Ατομική Διπλωματική Εργασία

ΕΞΟΡΥΞΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΑΝΑΛΥΣΗ ΚΟΙΝΩΝΙΚΟΥ ΔΙΚΤΥΟΥ LINKEDIN

Ανδρέου Κ. Ανδρέας

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2016

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Εξόρυξη Δεδομένων και Ανάλυση Κοινωνικού Δικτύου LinkedIn

Ανδρέου Κ. Ανδρέας

Επιβλέπων Καθηγητής
Δρ. Μάριος Δικαιάκος

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2016

Ευχαριστίες

Καταρχάς, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της Διπλωματικής μου κ. Μάριο Δικαιάκο για την συνεχή του υποστήριξη, καθοδήγηση και τις συμβουλές που μου πρόσφερε κατά την διάρκεια της Ατομικής Διπλωματικής μου Εργασίας.

Επίσης θα ήθελα να ευχαριστήσω δύο ερευνητές του εργαστηρίου Διαδικτυακού Υπολογισμού (LInC), τους κύριους Δημήτρη Αντωνιάδη και Χάρη Ευσταθιάδη, για την βοήθεια τους και την καθοδήγησή τους σε διάφορα ζητήματα που παρουσιάστηκαν στην ανάπτυξη της Διπλωματικής Εργασίας.

Τέλος, ένα μεγάλο ευχαριστώ στην οικογένεια μου, για την οικονομική και ψυχολογική υποστήριξή τους καθώς και για την υπομονή τους καθ' όλη την διάρκεια των σπουδών μου.

Περίληψη

Αναμφίβολα, τα κοινωνικά δίκτυα έχουν γίνει πλέον αναπόσπαστο κομμάτι στην καθημερινότητά μας. Ανεξαρτήτως πλατφόρμας, Facebook, Twitter, LinkedIn και τα διάφορα άλλα, όλα έχουν ένα κοινό στοιχείο και αυτό είναι η πληθώρα πληροφοριών που μοιράζονται οι χρήστες τους οικειοθελώς.

Συγκεκριμένα, η παρούσα διπλωματική εργασία επικεντρώνεται στη συλλογή και ανάλυση πληροφοριών των χρηστών του κοινωνικού δικτύου LinkedIn. Αντιθέτως με τον ανταγωνισμό του, το LinkedIn είναι ένα πιο επιχειρηματικά προσανατολισμένα κοινωνικό δίκτυο όπου οι χρήστες του μπορούν να ανεβάσουν πληροφορίες για την μόρφωσή τους, τις δεξιότητές τους, τις θέσεις εργασίας τους, τα ενδιαφέροντά τους, να δημιουργήσουν συνδέσεις με άλλους χρήστες όπως επίσης και να φέρει κοντά εταιρείες και κυνηγούς εργασίας.

Επομένως, απώτερος σκοπός είναι η εκμετάλλευση αυτών των μέχρι τώρα αναξιοποίητο πληροφοριών και την διάθεσή τους με γραφικό τρόπο, μέσω ιστοσελίδας, έτσι ώστε να είναι προσβάσιμες από τον οποιοδήποτε και οποτεδήποτε.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1. Κοινωνικά Δίκτυα και το LinkedIn

Στην γενικευμένη του μορφή, ο όρος «κοινωνικό δίκτυο» ορίζεται σαν ένα σύνολο κοινωνικών αλληλεπιδράσεων και διαπροσωπικών σχέσεων. Παρόλα αυτά, ο όρος σήμερα χρησιμοποιείται κυρίως για να περιγράψει ιστοσελίδες ή/και εφαρμογές οι οποίες αποτελούν την διεπαφή ανάμεσα στους χρήστες και επιτρέπουν την επικοινωνία μεταξύ τους μέσω ανάρτησης προσωπικών τους στοιχείων, σχόλιων, μηνυμάτων και διαφόρων μορφών πολυμέσων. Υπάρχουν διάφορα παραδείγματα τέτοιων ιστοσελίδων, εκ των οποίων οι πιο γνωστές είναι το Facebook, Twitter, Instagram και LinkedIn.

Συγκεκριμένα, η παρούσα διπλωματική εργασία επικεντρώνεται στη συλλογή και ανάλυση πληροφοριών των χρηστών του κοινωνικού δικτύου LinkedIn. Αντιθέτως με τον ανταγωνισμό του, το LinkedIn είναι ένα πιο επιχειρηματικά προσανατολισμένα κοινωνικό δίκτυο όπου οι χρήστες του μπορούν να ανεβάσουν πληροφορίες για την μόρφωσή τους, τις δεξιότητές τους, τις θέσεις εργασίας τους, τα ενδιαφέροντά τους, να δημιουργήσουν συνδέσεις με άλλους χρήστες όπως επίσης και να φέρει κοντά εταιρείες και κυνηγούς εργασίας.

1.2. Κίνητρο Διπλωματικής Εργασίας

Ανεξαρτήτως πλατφόρμας, όλα τα κοινωνικά δίκτυα έχουν ένα κοινό στοιχείο και αυτό είναι η πληθώρα πληροφοριών που μοιράζονται οι χρήστες τους οικειοθελώς. Εκμεταλλευόμενοι αυτές τις πληροφορίες, μπορούμε να εξάγουμε διάφορα συμπεράσματα όπως το επίπεδο μόρφωσης των χρηστών μιας χώρας, τις δεξιότητες που κατέχουν, τα επίπεδα εργασίας καθώς και την ροή των εργαζομένων εντός και εκτός μιας χώρας.

Όπως φανερώνει και η έκθεση του ινστιτούτου McKinsey με τίτλο «Global Flows in a digital age: How trade, finance, people and data connect the world economy» [1], [2], οι σχέσεις μεταξύ χωρών έχουν γίνει πιο στενά συνδεδεμένες, με ροές αγαθών και υπηρεσιών να αποτελούν μέχρι και το 36% του παγκόσμιου ΑΕΠ (Ακαθάριστου Εγχώριου Προϊόντος).

Επομένως εκμεταλλευόμενοι την πληθώρα πληροφοριών που παρέχεται ελεύθερα από τους χρήστες του LinkedIn, μπορούμε να πάρουμε σημαντικές αποφάσεις που μπορούν να βοηθήσουν στην ανάπτυξη της οικονομίας, στα επίπεδα γνώσεις και καινοτομίας και την όλη ευημερία μιας χώρας.

1.3. Στόχοι Διπλωματικής Εργασίας

Η Διπλωματικής Εργασίας έχει χωριστεί σε δύο κυρίως στόχους. Αρχικά ήταν αναγκαία η συλλογή δεδομένων από τα προφίλ των χρηστών του LinkedIn και η αποθήκευσή τους σε μια Βάση Δεδομένων για την μεταγενέστερή τους ανάλυση. Λόγω του μεγάλου αριθμού των χρηστών που έχει η κάθε χώρα αλλά και περιορισμών του LinkedIn για τον αριθμό των χρηστών που μπορεί κάποιος να επισκεφθεί κάθε μέρα, οι χώρες προς μελέτη που έχουν επιλεχθεί είναι η Κύπρος και η Εσθονία.

Δεύτερος στόχος, ήταν η δημιουργία ιστοσελίδας όπου θα λειτουργούσε ως διεπαφή για τον χρήστη. Μέσω της ιστοσελίδας ο χρήστης θα μπορεί να επιλέξει διάφορα στατιστικά που τον ενδιαφέρουν, ομαδοποιημένα ανά κατηγορία/θέματα. Αφού ο χρήστης συμπληρώσει τα απαραίτητα πεδία για την δημιουργία τους (π.χ. χώρα, πτυχίο

κ.τ.λ.) θα μπορεί να δει τα αποτελέσματα σε δύο μορφές, σε μορφή γραφικής παράστασης και σε πίνακα. Επίσης ο χρήστης θα έχει την ευκαιρία να κάνει αλλαγές στα αποτελέσματα, εξατομικεύοντας τόσο τα δεδομένα όσο και την μορφή των αποτελεσμάτων.

Στα επόμενα κεφάλαια, αναπτύσσεται σε βάθος η μεθοδολογία για την επίτευξη των πιο πάνω στόχων, επεξηγώντας τις διάφορες γλώσσες και τεχνολογίες που έχουν χρησιμοποιηθεί, την αρχιτεκτονική του συστήματος, επεξήγηση των πιο σημαντικών προγραμματιστικών σημείων καθώς και των συμπερασμάτων που εξάχθηκαν με το πέρας της διπλωματικής.

1.4. Δομή Διπλωματικής Εργασίας

Η παρούσα Διπλωματική Εργασία έχει χωριστεί στα ακόλουθα κεφάλαια τα οποία συμπεριλαμβάνουν:

Κεφάλαιο 2: γίνεται αναφορά στις διάφορες γλώσσες και τεχνολογίες που χρησιμοποιήθηκαν για την συλλογή των δεδομένων από το LinkedIn, την αποθήκευσή τους, την δημιουργία της ιστοσελίδας καθώς και την τρόπο απεικόνισης στων γραφικών παραστάσεων.

Κεφάλαιο 3: γίνεται αναφορά στην αρχιτεκτονική του συστήματος χρησιμοποιώντας την αρχιτεκτονική Πελάτη-Εξυπηρετητή καθώς επίσης και στην δομή του συστήματος στον εξυπηρετητή και στην Βάση Δεδομένων.

Κεφάλαιο 4: γίνεται αναφορά στην υλοποίηση του συστήματος καλύπτοντας θέματα ασφάλειας, σχέσεων κλάσεων, χρησιμοποιώντας την Ενοποιημένη Γλώσσα Μοντελισμού (UML), καθώς και σημαντικών σημείων του κώδικα.

Κεφάλαιο 5: γίνεται αναφορά στα συμπεράσματα των αποτελεσμάτων σύγκρισης των δύο χωρών, Κύπρου και Εσθονίας, ανά κατηγορία αλλά και συμπεράσματα για την κάθε μια ξεχωριστά.

Παράρτημα Α: Το Παράρτημα Α περιλαμβάνει στιγμιότυπα οθονών του συστήματος.

Παράρτημα Β: Το Παράρτημα Β περιλαμβάνει το κώδικα δημιουργίας Βάσης Δεδομένων.

Παράρτημα Γ: Το Παράρτημα Γ περιλαμβάνει το κώδικα δημιουργίας διαπροσωπιών και κλάσεων PHP για την δημιουργία γραφικών παραστάσεων.

Κεφάλαιο 2

Μεθοδολογία: Γλώσσες και Τεχνολογίες που έχουν χρησιμοποιηθεί

2.1. Εισαγωγή

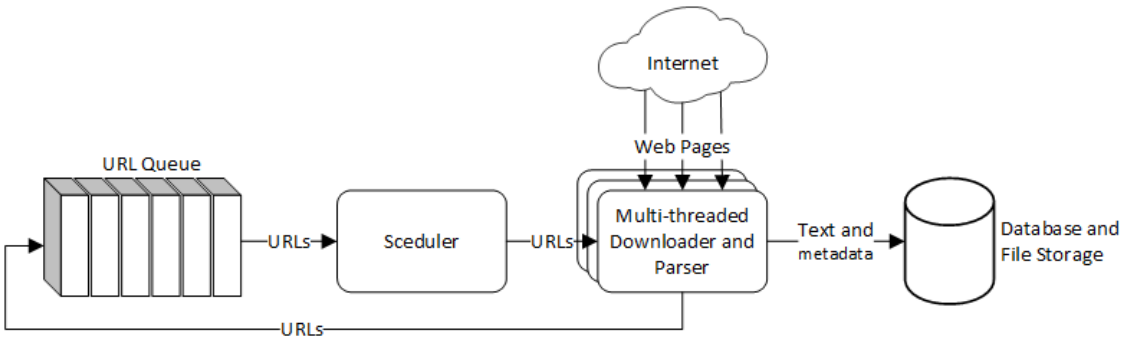
Όπως προϊδεάζει και ο τίτλος, αυτό το κεφάλαιο είναι αφιερωμένο στις διάφορες μεθοδολογίες που έχουν χρησιμοποιηθεί για την επίτευξη των στόχων της διπλωματικής εργασίας που αναφέρονται στο πρώτο κεφάλαιο.

Συγκεκριμένα, το κεφάλαιο αυτό επεξηγεί τον τρόπο και τις διάφορες προκλήσεις που παρουσιάστηκαν κατά την διάρκεια συλλογής δεδομένων, την αποθήκευσή τους στην βάση δεδομένων, της διάφορες γλώσσες που έχουν χρησιμοποιηθεί για την ανάπτυξη της ιστοσελίδας καθώς και τον τρόπο απεικόνισης στων γραφικών παραστάσεων και την έρευνα που έγινε πίσω από την επιλογή αυτή.

2.2. Συλλογή Δεδομένων

Για τον λόγο ότι το LinkedIn δεν παρέχει κάποια αυτοματοποιημένη λειτουργία για την παροχή πληροφοριών από τα προφίλ των χρηστών του, ήταν αναγκαία η χρήση ενός Web Crawler[9].

Ένας Web Crawler (βλ. Σχήμα 2.1) είναι ένα λογισμικό το οποίο σαρώνει συστηματικά το διαδίκτυο ακολουθώντας συνδέσμους (links) σε ιστοσελίδες τις οποίες κατεβάζει, αναλύει, αξιολογεί και έπειτα αποθηκεύει ότι χρειάζεται ενώ οι διάφοροι σύνδεσμοι σε άλλες ιστοσελίδες που ανακαλύφθηκαν προθέτονται στην ουρά συνδέσμων για να υποστούν και αυτές την ίδια μεταχείριση.



Σχήμα 2.1: High-level αρχιτεκτονική ενός προτύπου Web Crawler

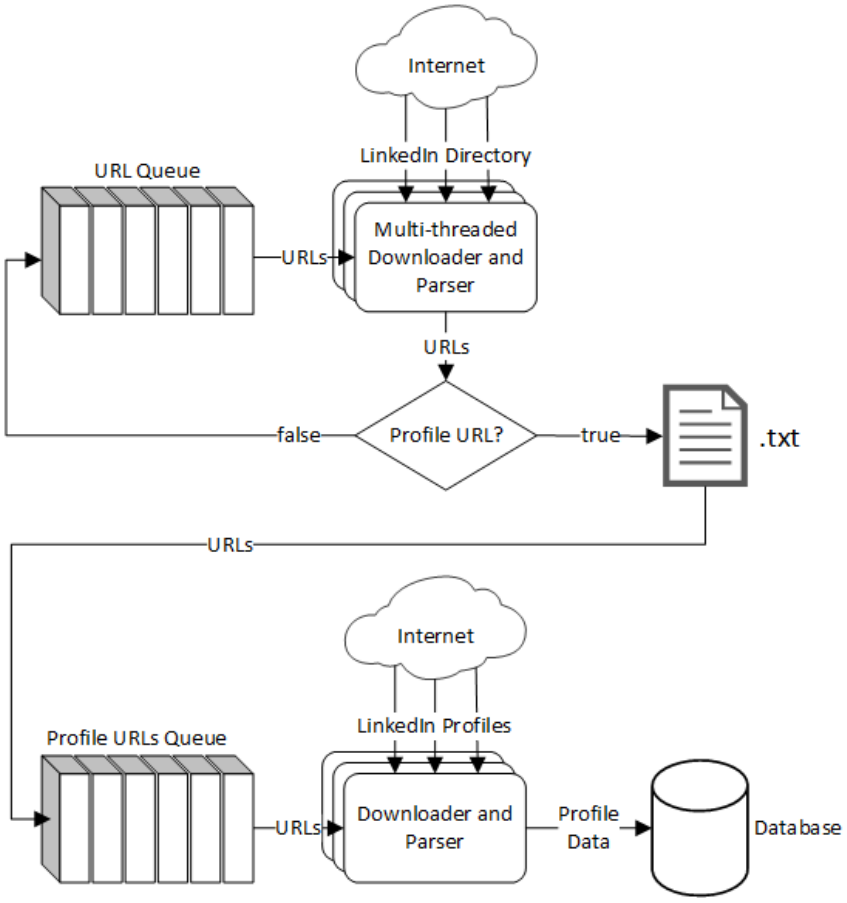
Οι Web Crawlers αποτελούν κεντρικό κομμάτι στις μηχανές αναζήτησης (π.χ. Google), οι οποίες τους χρησιμοποιούν κυρίως για σκοπούς ευρετηρίασης ιστοσελίδων. Παρόλα αυτά, συνήθως οι λεπτομέρειες για τους αλγορίθμους και την αρχιτεκτονική τους αποτελούν επιχειρηματικό απόρρητο και όταν δημοσιεύονται τα σχέδιά τους, συνήθως υστερούν λεπτομέρειες-κλειδιά για αποτροπή αναπαραγωγής του έργου τους από τυχών ανταγωνιστές.

Η δημιουργία ενός υψηλής απόδοσης συστήματος που να μπορεί να κατεβάζει εκατοντάδες εκατομμύρια σελίδες κατά τη διάρκεια αρκετών εβδομάδων παρουσιάζει μια σειρά από προκλήσεις στον σχεδιασμό του, αποτελεσματικότητα εισόδων/εξόδων και δικτύου αλλά και διαχειρισιμότητα. Παρόλα αυτά, είναι εύκολο να δημιουργηθεί ένας αργός crawler που να μπορεί να κατεβάζει λίγες σελίδες ανά δευτερόλεπτο για σύντομο χρονικό διάστημα.

Στην περίπτωση crawling του LinkedIn έχει δημιουργηθεί ένας απλός crawler, χρησιμοποιώντας την Java[8]. Η Java είναι μια γενικού σκοπού γλώσσα προγραμματισμού η οποία είναι συντρέχουσα, class-based, αντικειμενοστρεφής και έχει σχεδιαστεί για να τρέχει παντού, ανεξαρτήτως συσκευής ή λειτουργικού συστήματος. Για σκοπούς parsing των ιστοσελίδων, έχει χρησιμοποιηθεί η βιβλιοθήκη jsoup[10]. Η

jsoup είναι μια ανοιχτού-κώδικα Java βιβλιοθήκη που περιέχει τις απαραίτητες μεθόδους για εξαγωγή και διαχείριση δεδομένων σε HTML έγγραφα (περισσότερα για την HTML στο υποκεφαλαίου 2.4. Δημιουργία Ιστοσελίδας).

Ο crawler του LinkedIn αποτελείται από δύο κυρίως μέρη (βλ. Σχήμα 2.2), το Directory και το Profile Crawler. Ο Directory Crawler χρησιμοποιώντας πολυνηματικό προγραμματισμό, σαρώνει το κατάλογο με τα προφίλ των χρηστών του LinkedIn. Το κάθε νήμα είναι υπεύθυνο για ένα γράμμα του λατινικού αλφαβήτου για το οποίο και δημιουργεί ένα αρχείο .txt στο οποίο αποθηκεύονται οι σύνδεσμοι στα προφίλ των χρηστών. Εξερευνώντας τον κατάλογο με τα προφίλ, αν κάποιος σύνδεσμος οδηγεί σε χρήστη, τότε αποθηκεύεται στο αρχείο .txt, διαφορετικά αν οδηγεί σε άλλο κατάλογο, τότε μπαίνει στην ουρά με τους συνδέσμους για να υποστεί την ίδια μεταχείριση. Η διαδικασία τελειώνει με το άδειασμα τις ουράς των συνδέσμων.



Σχήμα 2.2: High-level αρχιτεκτονική του LinkedIn Crawler

Ο Profile Crawler, παίρνει στην συνέχεια τα αρχεία .txt με τους συνδέσμους στα προφίλ και τα τοποθετεί στην δικιά του ουρά. Στην συνέχεια επισκέπτεται το προφίλ του κάθε χρήστη, εξάγει τα στοιχεία του και τα αποθηκεύει σε μια Βάση Δεδομένων (βλ. 2.3. Αποθήκευση Δεδομένων). Η διαδικασία τελειώνει με το άδειασμα τις ουράς και αφού έχουν τελειώσει όλα τα αρχεία με τους συνδέσμους στα προφίλ.

Λόγο μεγάλου αριθμού χρηστών που έχει κάθε χώρα, έχουν επιλεγεί δύο μικρές χώρες για το crawling, η Κύπρος και η Εσθονία. Από το crawling της Κύπρου έχει μαζευτεί ένα δείγμα **140,017** χρηστών, ενώ από της Εσθονίας ένα δείγμα **125,253** χρηστών.

2.3. Αποθήκευση Δεδομένων

Κατά την διάρκεια του crawling, απαραίτητη είναι και η αποθήκευσή των δεδομένων που συλλέγονται σε κάποια Βάση Δεδομένων για την μεταγενέστερη τους ανάλυση. Για τον σκοπό αυτό έχει επιλεγεί η MySQL[11], η οποία είναι ένα δωρεάν και ανοιχτού κώδικα σύστημα διαχείριση σχεσιακών βάσεων δεδομένων. Μέσω του crawling έχουν συλλεχθεί τα ακόλουθα στοιχεία για τον κάθε χρήστη:

- Προσωπικά Στοιχεία όπως ονοματεπώνυμο, επικεφαλίδα, περιγραφή, τοποθεσία, βιομηχανία εργασίας, URL προφίλ, URL φωτογραφίας προφίλ, αριθμό συνδέσεων και συστάσεις.
- Δεξιότητες, πιστοποιητικά και διάφορες σειρές μαθημάτων που έχει παρακολουθήσει.
- Θέσεις εργασίας, περίοδο εργασίας και στοιχεία των εταιρειών στις οποίες εργάστηκε όπως: όνομα, τύπο, μέγεθος, διεύθυνση, βιομηχανία, ιστοσελίδα, περιγραφή, έτος ίδρυσης και το LinkedIn URL της.
- Ομιλούμενες γλώσσες και γλωσσική επάρκεια.
- Στοιχεία μόρφωσης τους όπως πτυχίο, major, χρονολογία και το σχολείο στο οποίο έχει φοιτήσει.
- Διάφορες δημοσιεύσεις και έργα στα οποία ήταν πρωτουργοί ή έλαβαν μέρος.
- Διάφορα άλλα στοιχεία όπως τα ενδιαφέροντα τους, εθελοντισμό, οργανώσεις και groups τα οποία παρακολουθεί.

Όσο αφορά την αρχιτεκτονική της Βάσης Δεδομένων, γίνεται αναφορά στο Κεφάλαιο 3: Αρχιτεκτονική Συστήματος στο υποκεφάλαιο 3.3 Δομή Βάσης Δεδομένων (Relational Model).

2.4. Δημιουργία Διαδικτυακής Εφαρμογής/Ιστοσελίδας

Για τις ανάγκες πρόσβασης στην ανάλυση των δεδομένων του LinkedIn αλλά και για την απεικόνιση των γραφικών παραστάσεων των δεδομένων αυτών έχει δημιουργηθεί μια διαδικτυακή εφαρμογή, μια ιστοσελίδα, που θα εκπληρώνει τις ανάγκες αυτές. Μια διαδικτυακή εφαρμογή ονομάζεται κάθε εφαρμογή η οποία είναι διαθέσιμη στους χρήστες της μέσω του Διαδικτύου και ο χρήστης χρησιμοποιεί μόνο τον περιηγητή (browser) του για να την χρησιμοποιήσει.

Αν και διαχρονικά, οι περισσότεροι χρήστες προτιμούσαν την εγκατάσταση και χρήση τοπικών εφαρμογών, παρόλα αυτά, με την πάροδο του χρόνου παρατηρείται μια μετάβαση στις διαδικτυακές εφαρμογές. Πράγματι, συγκρίνοντας τις δύο, τα πλεονεκτήματα χρήσης διαδικτυακής εφαρμογής είναι πολυάριθμα, εκ των οποίων συμπεριλαμβάνουν:

- **Πρόσβαση ανεξαρτήτως συσκευής:** οι χρήστες έχουν άμεση πρόσβαση από οποιονδήποτε υπολογιστή ή άλλη συσκευή πού έχει πρόσβαση στο διαδίκτυο. Δεν υπάρχει κάποια απαραίτητη ανάγκη για εγκατάσταση επιπρόσθετου λογισμικού εκτός από την εφαρμογή του περιηγητή διαδικτύου (browser) πού είναι ήδη προ-εγκαταστημένος σε όλες τις μηχανές.

- **Πρόσβαση ανεξαρτήτως τοποθεσίας:** λόγο χρήσης τους διαδικτύου, οι διαδικτυακές εφαρμογές δίνουν ευελιξία στους χρήστες τους ώστε να τις χρησιμοποιούν οπουδήποτε αυτοί επιθυμούν.

- **Πρόσβαση ανεξαρτήτως λειτουργικού συστήματος:** καθώς η εκτέλεση της εφαρμογής γίνεται εντός του περιηγητή διαδικτύου και όχι στον υπολογιστή του χρήστη την κάνει ικανή να εκτελείται σε όλα τα λειτουργικά συστήματα.

- **Δεν καταναλώνουν πόρους:** ακολουθώντας την πιο πάνω λογική, αφού η διαδικτυακές εφαρμογές βρίσκονται σε εξυπηρετητές και δεν εκτελούνται στον υπολογιστή του χρήστη, αυτό σημαίνει ότι και δεν καταναλώνουν πόρους από το σύστημα του, τόσο υπολογιστικούς όσο και αποθηκευτικούς.

- **Γρήγορη και Καθολική Αναβάθμιση:** ένα από τα σημαντικότερα πλεονεκτήματα των διαδικτυακών εφαρμογών έναντι των τοπικών, είναι οι περιπτώσεις ανάγκης αναβάθμισης της εφαρμογής. Η αναβάθμιση στην τοπική εφαρμογή θα πρέπει να γίνει ξεχωριστά σε κάθε υπολογιστή με την έγκριση του κάθε χρήστη και με την πιθανότητα αδυναμία πρόσβασης στην εφαρμογή χωρίς την αναβαθμισμένη έκδοση. Αντιθέτως, στις διαδικτυακές εφαρμογές η αναβάθμιση γίνεται στον εξυπηρετητή που φιλοξενεί την εφαρμογή. Έτσι έχουν όλοι χρήστες ταυτόχρονα πρόσβαση στην αναβαθμισμένη της έκδοση.

Αν και πράγματι, η υλοποίηση ενός συστήματος σαν διαδικτυακή εφαρμογή υπερτερεί σε πολλούς τομείς, παρόλα αυτά υπάρχουν και μειονεκτήματα όπως:

- **Αδυναμία χρήσης χωρίς πρόσβαση σε διαδίκτυο:** η ανάγκη πρόσβασης στο διαδίκτυο για την λειτουργία της διαδικτυακή εφαρμογής εκτός από πλεονέκτημα είναι ταυτόχρονα και αδυναμία της, μιας και η απουσία διαδικτύου την καθιστά μη λειτουργική.

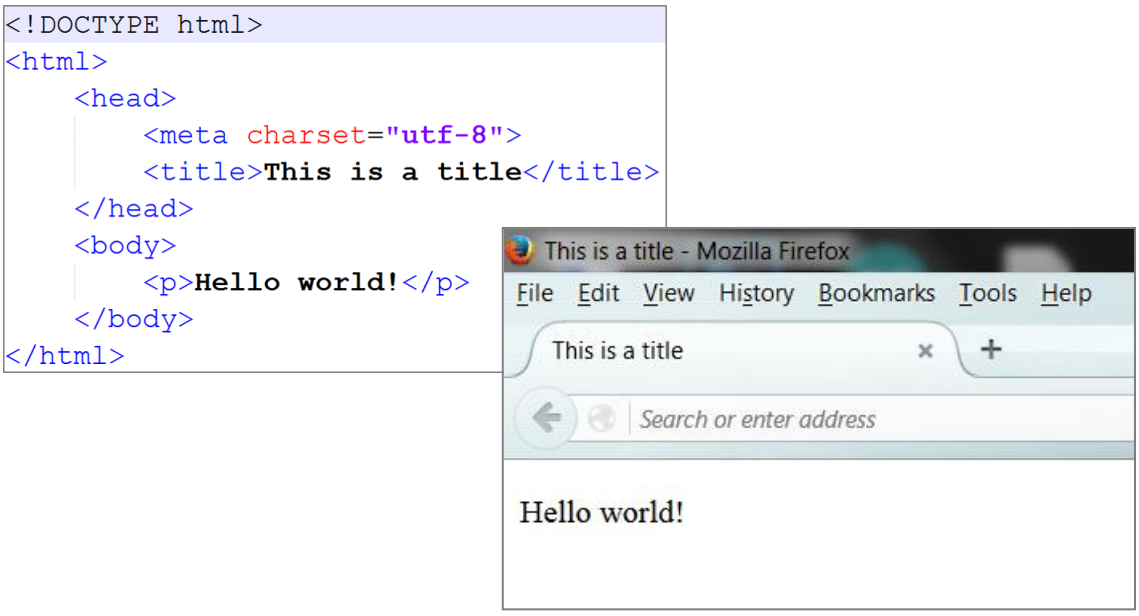
- **Συμβατότητα περιηγητών (browsers):** το συγκεκριμένο μειονέκτημα αφορά την τελευταία έκδοση της HTML, την HTML5, η οποία δεν υποστηρίζεται από όλους του περιηγητές. Αυτό δημιουργεί προβλήματα στην κατασκευή διαδικτυακών εφαρμογών που θέλουν να εκμεταλλευτούν τις καινούργιες δυνατότητες τις HTML5. Οι κατασκευαστές της εφαρμογής πρέπει να αποφασίσουν με τους χρήστες από κοινού ποιος περιηγητής θα είναι ο προτεινόμενος και ταυτόχρονα να προβλέπεται η χρήση άλλων περιηγητών.

- **Αναβάθμιση:** όπως η ανάγκη πρόσβασης στο διαδίκτυο, έτσι και η αναβάθμιση των διαδικτυακών εφαρμογών μπορεί να αποβεί μειονέκτημα. Αν υπάρχει κάποιο πρόβλημα στις κλασικές τοπικές εφαρμογές, τότε μια εταιρεία μπορεί να επιλέξει να

μην αναβαθμίσει την εφαρμογή αυτή μέχρι να επιλυθεί το πρόβλημα. Παρόλα αυτά, στην περίπτωση των διαδικτυακών εφαρμογών οι εταιρείες και γενικά οι χρήστες δεν μπορούν να αποτρέψουν την αναβάθμιση αυτή.

Η υλοποίηση μιας τέτοιας διαδικτυακής εφαρμογής, μιας ιστοσελίδας, συμπεριλαμβάνει χρήση διάφορων γλωσσών προγραμματισμού, κανόνων και τεχνολογιών. Παρόλα αυτά, ανεξαρτήτως πολυπλοκότητας, υπάρχουν τρείς κύριες γλώσσες/κανόνες που αποτελούν τον πυρήνα της δημιουργίας τους, η HTML, η CSS και η JavaScript. Επομένως, συμπεριλαμβανομένου των τριών αυτών, το συγκεκριμένο σύστημα χρησιμοποιεί τα ακόλουθα:

- **HyperText Markup Language (HTML):** Η Γλώσσα Σήμανσης Υπερκειμένου είναι η κύρια γλώσσας σήμανσης για τις ιστοσελίδες και τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων. Γράφεται υπό την μορφή ετικετών (tags) όπου συνήθως λειτουργούν ανά ζεύγη, με την ετικέτα έναρξης π.χ.: `<h1>` και την ετικέτα λήξης π.χ.: `</h1>`, με κάποιες εξαιρέσεις όπου η ετικέτα λήξης ενσωματώνεται στην ετικέτα έναρξης π.χ.: `<input />`. Το Σχήμα 2.3 παρουσιάζει παράδειγμα αρχείου HTML και τον τρόπο παρουσίασης του μέσω περιηγητή διαδικτύου:



Σχήμα 2.3: Παράδειγμα γλώσσας HTML

- **Cascading Style Sheets (CSS):** οι κανόνες τεχνοτροπίας χρησιμοποιούνται για τον έλεγχο της εμφάνιση ενός εγγράφου HTML. Χρησιμοποιώντας CSS μπορούμε να κάνουμε στιλιστικές αλλαγές στην ιστοσελίδα, διαμορφώνοντας χαρακτηριστικά όπως: χρώματα, περιγράμματα, σκίαση, στοίχιση, μέγεθος γραμματοσειράς κ.τ.λ. Το Σχήμα 2.4 παρουσιάζει τρείς διαφορετικούς τρόπους που μπορούν να συμπεριληφθούν οι κανόνες τεχνοτροπίας, (1) απευθείας σε κάποιο στοιχείο, (2) στην κορυφή του αρχείου HTML ή (3) σε κάποιο εξωτερικό αρχείο, κάτι που προτιμάται για διευκόλυνση συντήρησης του κώδικα.

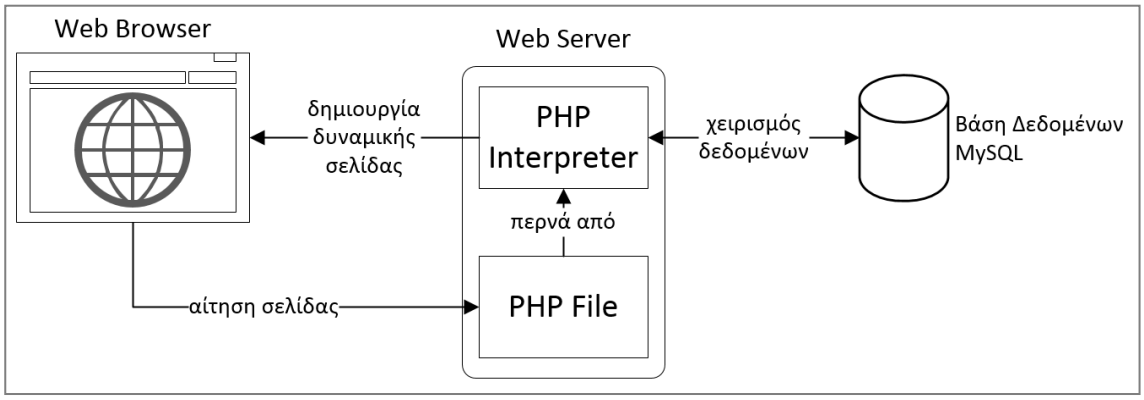
Σχήμα 2.4: Παράδειγμα γλώσσας CSS για αλλαγή χρώματος

- **JavaScript:** είναι μια υψηλού-επιπέδου γλώσσα (high-level), δυναμική (dynamic), με ασθενείς τύπους (untyped) και διερμηνευμένη (interpreted). Χρησιμοποιείται για προγραμματισμό από την πλευρά του πελάτη, στο περιηγητή (browser), και χαρακτηρίζεται ως client-side γλώσσα προγραμματισμού. Το Σχήμα 2.5 παρουσιάζει παράδειγμα όπου πατώντας ένα κουπί, εμφανίζεται η μέρα και η ώρα:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>This is a title</title>
    </head>
    <body>
        <h1>My First JavaScript</h1>
        <button
            type="button"
            onclick="document.getElementById('demo').innerHTML = Date()">
            Click me to display Date and Time.
        </button>
        <p id="demo"></p>
    </body>
</html>
```

Σχήμα 2.5: Παράδειγμα γλώσσας JavaScript για εμφάνιση μέρας και ώρας

- **Hypertext Preprocessor (PHP):** είναι μια γλώσσα προγραμματισμού για την δημιουργία διαδικτυακών σελίδων με δυναμικό περιεχόμενο η οποία τρέχει στην μεριά του εξυπηρετητή (server-side). Όταν ζητηθεί μια σελίδα PHP (π.χ. με κατάληξη .php, php4, κ.ά.) από τον περιηγητή διαδικτύου (βλ. Σχήμα 2.6) περνά από τον PHP Interpreter ώστε να παραχθεί σε πραγματικό χρόνο το τελικό περιεχόμενο, το οποίο μπορεί να συμπεριλαμβάνει δεδομένα και από κάποια Βάση Δεδομένων, που θα σταλεί στην συνέχεια στον περιηγητή διαδικτύου.



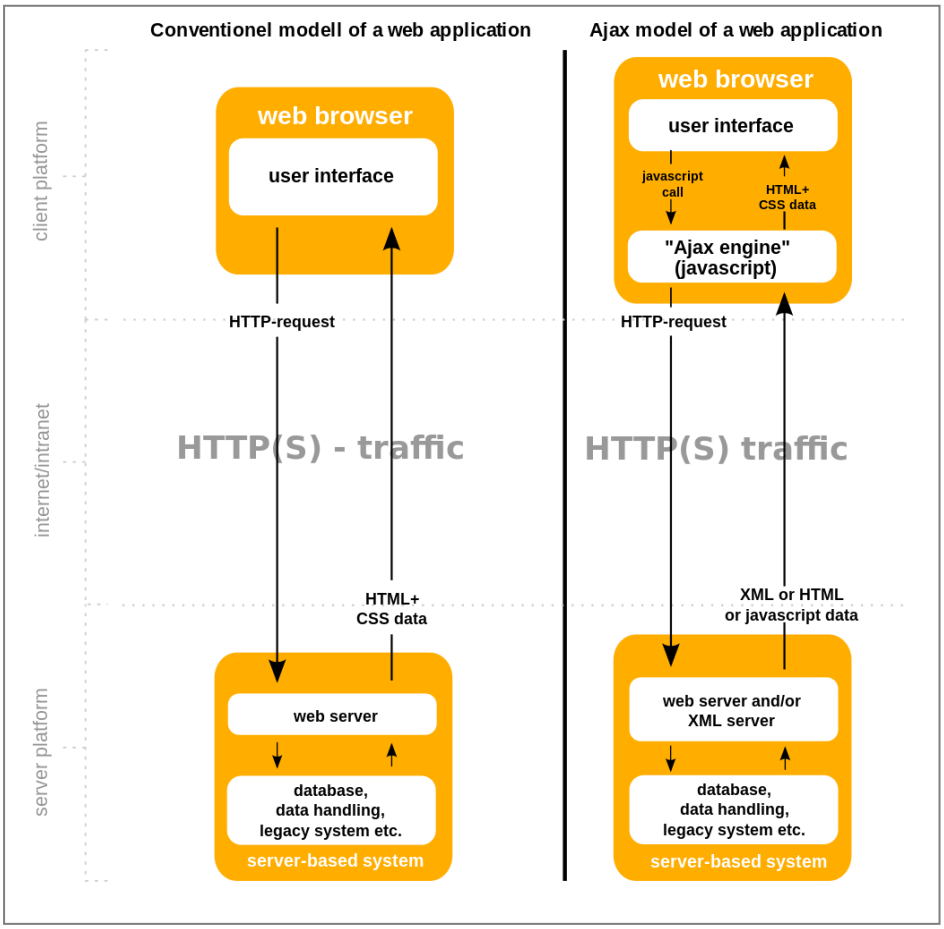
Σχήμα 2.6: Παράδειγμα αίτησης σελίδας PHP

Ο κώδικας της PHP μπορεί να είναι ενσωματωμένος (embedded) στον HTML κώδικα ή γίνει συνδυασμός με διάφορα συστήματα προτύπων ιστού, διαχείρισης ή πλαισίου (frameworks). Το Σχήμα 2.7 παρουσιάζει ένα τέτοιο παράδειγμα ενσωματωμένου κώδικα PHP σε HTML:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>PHP Test</title>
    </head>
    <body>
        <?php echo '<p>Hello from PHP</p>'; ?>
    </body>
</html>
```

Σχήμα 2.7: Παράδειγμα ενσωματωμένου κώδικα PHP σε HTML

- **Asynchronous JavaScript and XML (AJAX):** η AJAX είναι ένα σετ από τεχνικές ανάπτυξης ιστοσελίδων που χρησιμοποίει πολλές τεχνολογίες διαδικτύου στην μεριά του χρήστη (client-side) για να δημιουργηθούν ασύγχρονες διαδικτυακές εφαρμογές, αφού επιτρέπει την αμφίδρομη επικοινωνία μεταξύ του εξυπηρετητή και του χρήστη (Σχήμα 2.8). Έτσι, χρησιμοποιώντας AJAX, επιτρέπεται η ανανέωση ενός μέρους μιας ιστοσελίδας, χωρίς αυτή να χρειαστεί να ξαναφορτωθεί (reload).



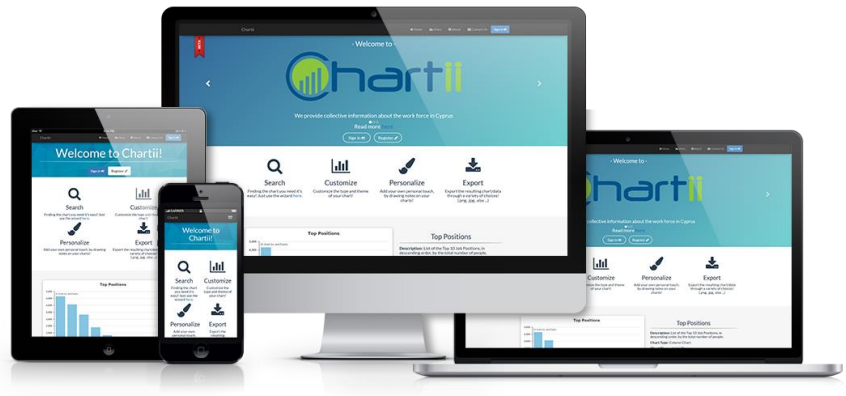
Σχήμα 2.8: Συμβατικό μοντέλο διαδικτυακής εφαρμογής έναντι εφαρμογής που χρησιμοποιεί τεχνολογία AJAX
(Πηγή: https://en.wikipedia.org/wiki/File:Ajax-vergleich-en.svg)

2.5. Responsive Web Design

Με την ραγδαία χρήση των έξυπνων συσκευών (smartphones) και ταμπλετών (tablets) οι χρήστες μπορούν να επισκεφθούν ιστοσελίδες από διάφορες συσκευές, οι οποίες διαφέρουν τόσο σε μέγεθος οθόνης όσο και με τον τρόπο αλληλεπίδρασης τους με τον χρήστη. Η πρόσβαση σε κλασικού τύπου ιστοσελίδες μπορεί να αποβεί δυσάρεστη εμπειρία για τον χρήστη μιας και υπάρχει η ανάγκη συνεχών προσαρμογών για την βέλτιστη οπτική προβολή όπως συνεχή μεγέθυνση και κύλιση της ιστοσελίδας.

Σε αυτή την περίπτωση έρχεται το Responsive Web Design, όπου είναι μια προσέγγιση για το σχεδιασμό ιστοσελίδων που παρέχει μια βέλτιστη προβολή περιεχομένου και καλύτερη εμπειρία αλληλεπίδρασης με ευκολότερη ανάγνωση και πλοήγηση ελαχιστοποιώντας τις ανάγκες προσαρμογής σε ένα ευρύ φάσμα συσκευών. Αν για παράδειγμα κάποιος επισκεφθεί μια ιστοσελίδα με responsive σχεδιασμό μέσω του smartphone, η ιστοσελίδα θα διαμορφωθεί με τέτοιο τρόπο ώστε να μην χρειάζεται ο χρήστης να κάνει μεγέθυνση ή να κάνει πλάγιο scroll για να διαβάσει το περιεχόμενο. Επίσης σημαντικό πλεονέκτημα είναι και το γεγονός ότι δημιουργείται μια ιστοσελίδα η οποία μπορεί να εξυπηρετήσει διαφόρου μεγέθους συσκευές, διευκολύνοντας έτσι την ενημέρωσή της αλλά και μειώνοντας το κόστος ανάπτυξής της.

Για την responsive λειτουργία του συστήματος έχει χρησιμοποιηθεί το Bootstrap v3, το οποίο είναι μια συλλογή εργαλείων ανοιχτού κώδικα για την δημιουργία τέτοιου είδους ιστοσελίδων και διαδικτυακών εφαρμογών. Το Bootstrap παρέχει HTML και CSS σχεδιαστικά πρότυπα, καθώς και προαιρετικές επεκτάσεις JavaScript, για τις μορφές τυπογραφίας, κουμπιών πλοήγησης και άλλων στοιχείων του περιβάλλοντος. Το πιο κάτω Σχήμα 2.9 παρουσιάζει τη μορφή του συγκεκριμένου συστήματος από διάφορες συσκευές:



Σχήμα 2.9: Παράδειγμα Responsive Web Design με Bootstrap v3.

2.6. Απεικόνιση Γραφικών Παραστάσεων

Ένα σημαντικό κομμάτι στην υλοποίηση του συγκεκριμένου συστήματος είναι ο τρόπος απεικόνισης γραφικών παραστάσεων. Για την επιλογή της κατάλληλης βιβλιοθήκης απεικόνισης γραφικών παραστάσεων έπρεπε να γίνει μια μικρή έρευνα στην οποία η υποψήφιες βιβλιοθήκες έπρεπε να πληρούν τα πιο κάτω κριτήρια:

- Υποστήριξη από όλους τους περιηγητές διαδικτύου.
- Συχνότητα αναβαθμίσεων και ενημερώσεων.
- Υποστήριξη διαφόρων τύπων γραφικών παραστάσεων.
- Δυνατότητα εξατομίκευσης από τον χρήστη.
- Δυνατότητα αποθήκευσης της γραφικής παράστασης σε μορφή εικόνας.
- Να παρέχεται δωρεάν.

Οι βιβλιοθήκες που δοκιμάστηκαν βάση των πιο πάνω κριτηρίων ήταν:

- Google Charts [https://developers.google.com/chart/]
- HighCharts [http://www.highcharts.com/]
- amCharts [https://www.amcharts.com/]
- Morris Charts [http://morrisjs.github.io/morris.js/]
- AnyChart [http://www.anychart.com/]
- Chartist [https://gionkunz.github.io/chartist-js/]
- D3.js [https://d3js.org/]
- Cytoscape [http://www.cytoscape.org/]
- Vis.js [http://visjs.org/]

Από τις πιο πάνω βιβλιοθήκες, καταλληλότερη κρίθηκε η amCharts, η οποία κάνει χρήση JavaScript και παρεμφερή τεχνολογίες. Ως cross-platform standard, η JavaScript υποστηρίζεται από την συντριπτική πλειοψηφία των περιηγητών διαδικτύου και ως επακόλουθο και συσκευών, πληρώντας έτσι το πρώτο κριτήριο επιλογής. Η βιβλιοθήκη amCharts δέχεται συχνές αναβαθμίσεις, παρέχει πλήθος διαφορετικών γραφικών παραστάσεων, επιτρέπει την εξατομίκευση από τον χρήστη, παρέχεται δωρεάν και επιτρέπει την αποθήκευση των γραφικών παραστάσεων και των δεδομένων τους σε διάφορους τύπους όπως .png, .jpg, .svg, .json, .xlsx, κ.ά., πληρώντας έτσι και τα υπόλοιπα κριτήρια επιλογής της.

Παρόλα αυτά, κατά την διάρκεια υλοποίησης του συστήματος, παρατηρήθηκε η αδυναμία παροχής ενός συγκεκριμένου τύπου γραφικής παράστασης, Sankey διπλής κατεύθυνσης και γι' αυτό χρειάστηκε να γίνει συνδυασμός με ακόμα μια βιβλιοθήκη, την D3.js, η οποία ήταν και η μόνη που παρείχε τέτοιου είδους γραφική παράσταση. Όπως η amCharts βιβλιοθήκη, έτσι και η D3.js κάνει χρήση JavaScript και καλύπτει τα περισσότερα από τα κριτήρια, με την εξαίρεση της δυνατότητας εξατομίκευσης από τον χρήστης και την αποθήκευση σε μορφή εικόνας, η οποία χρειάστηκε να υλοποιηθεί.

Κεφάλαιο 3

Αρχιτεκτονική Συστήματος

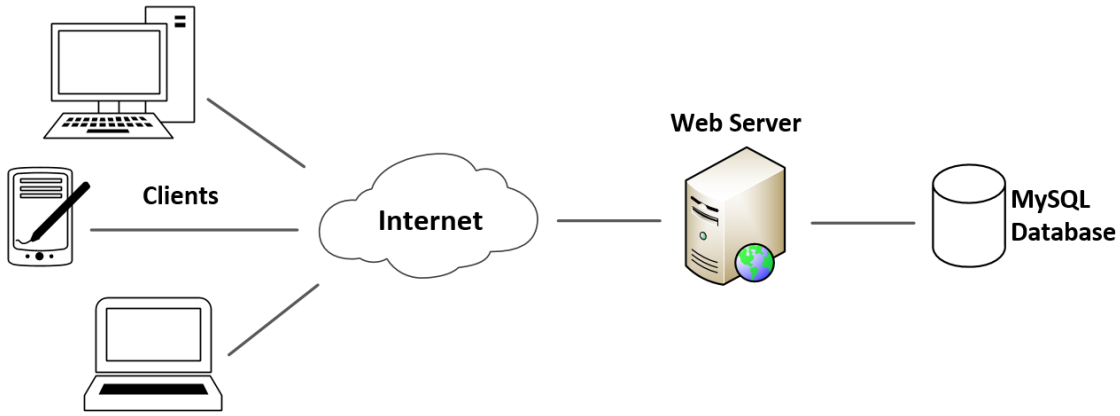
3.1. Εισαγωγή

Το συγκεκριμένο κεφάλαιο είναι αφιερωμένο στην επεξήγηση της αρχιτεκτονικής του συστήματος καθώς και της δομής του. Όσο αφορά την αρχιτεκτονική του συστήματος, γίνεται αναφορά στο μοντέλο πελάτη-εξυπηρετητή που χρησιμοποιήθηκε, τα πλεονεκτήματα μιας τέτοιας κεντροποιημένης υλοποίησης καθώς και την επικοινωνία μεταξύ πελάτη και εξυπηρετητή και τα μηνύματα που ανταλλάζονται μεταξύ τους.

Όσο αφορά την δομή του συστήματος, γίνεται αναφορά στην δομή της ιστοσελίδας στον εξυπηρετητή καθώς και της δομής της Βάσης Δεδομένων. Στο υποκεφάλαιο της δομής της ιστοσελίδας γίνεται παρουσίαση του χάρτη της ιστοσελίδας, επεξήγηση των διάφορων σελίδων και αρχείων καθώς και τα δικαιώματα πρόσβασης σε αυτά, ενώ στο υποκεφάλαιο της δομής της Βάσης Δεδομένων γίνεται παρουσίαση των πινάκων σε μορφή Σχεσιακού Μοντέλου (Relational Model), ένα μοντέλο που βοηθά καλύτερα στην παρουσίαση των δεδομένων που αποθηκεύονται σε κάθε πίνακα, τα πρωτεύων κλειδιά τους και την σχέση μεταξύ των κλειδιών αυτών.

3.2. Αρχιτεκτονική Πελάτη-Εξυπηρετητή (Client-Server)

Η αρχιτεκτονική Πελάτη-Εξυπηρετητή αποτελεί μια συνήθη μέθοδο ανάπτυξης λογισμικού στην οποία ο πελάτης θεωρείται ο περιηγητής ιστού που χρησιμοποίει ο χρήστης και ο εξυπηρετητής είναι ο εξυπηρετητής Ιστού του παροχέα ο οποίος φιλοξενεί την ιστοσελίδα. Όπως φαίνεται και στο Σχήμα 3.1, αυτό το είδος αρχιτεκτονικής έχει ένα ή περισσότερους πελάτες, ο καθένας με διαφορετικό τύπο συσκευής και λειτουργικό σύστημα, οι οποίοι επικοινωνούν με ένα κεντρικό εξυπηρετητή μέσω δικτύου ή μέσω σύνδεσης διαδικτύου.



Σχήμα 3.1: Αρχιτεκτονική Πελάτη-Εξυπηρετητή.

Η αρχιτεκτονική Πελάτη-Εξυπηρετητή θεωρείται και μια αρχιτεκτονική παραγωγού-καταναλωτή όπου ο εξυπηρετητής φιλοξενεί, παράγει, παραδίδει και διαχειρίζεται το μεγαλύτερο μέρος των πόρων και υπηρεσιών που καταναλώνονται από τον πελάτη. Ένας πελάτης στέλνει αίτημα για πόρο στον εξυπηρετητή μέσω διαδικτύου, το οποίο επεξεργάζεται χρησιμοποιώντας δεδομένα αποθηκευμένα στον εξυπηρετητή ή και σε μια Βάση Δεδομένων και στην συνέχεια παραδίδεται απάντηση πίσω στον πελάτη.

Βασικά πλεονεκτήματα μιας τέτοιας αρχιτεκτονικής είναι ότι ένας εξυπηρετητής μπορεί να διαχειριστεί πολλούς πελάτες ταυτόχρονα, ενώ παράλληλα ένας πελάτης μπορεί να συνδεθεί με διάφορους εξυπηρετητές, με τον κάθε ένα να παρέχει διαφορετικό σύνολο υπηρεσιών. Η κεντροποιημένη μορφή της αρχιτεκτονικής αυτής διευκολύνει επίσης την αποθήκευσης των δεδομένων αλλά και την εύκολη ενημέρωσή τους.

Παρόλη την διευκόλυνση και ταχύτητα που παρέχει ένα κεντροποιημένο σύστημα, δεν παύει να μειονεκτεί σε διάφορα άλλα θέματα. Στην περίπτωση συνεχών αιτημάτων από πολλούς πελάτες ταυτόχρονα, μπορεί να προκαλέσει υπερχείλιση του εξυπηρετητή και την μη δυνατή εξυπηρέτησή τους. Το γεγονός ότι η μορφή της αρχιτεκτονικής είναι κεντροποιημένη, καθιστά το σύστημα ευάλωτο στην περίπτωση τέτοιου προβλήματος αφού η κατάρρευση τέτοιου κεντρικού σημείου του συστήματος, επακολουθεί και κατάρρευση ολόκληρου του συστήματος.

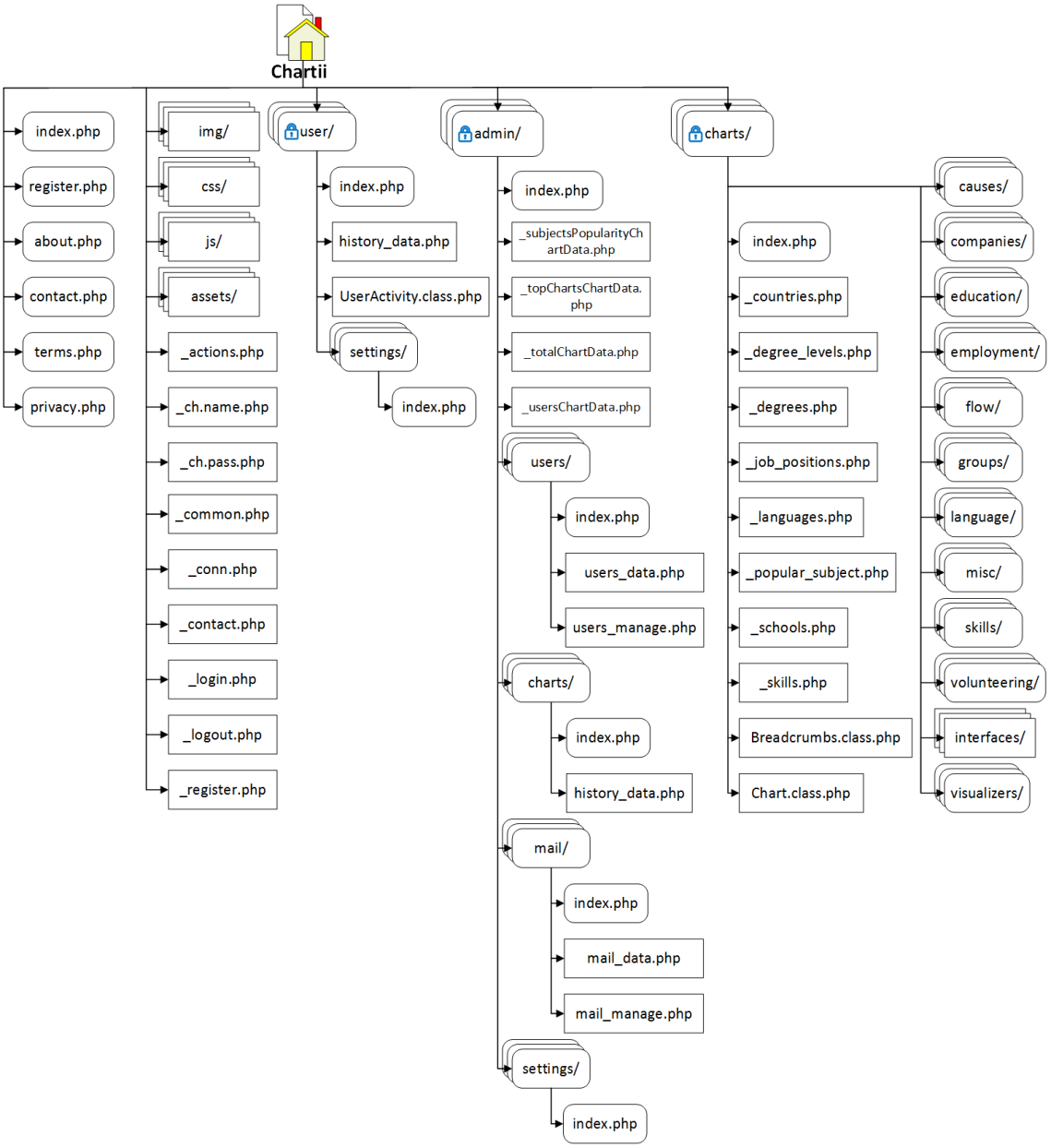
3.3. Δομή Ιστοσελίδας στον εξυπηρετητή

Το Σχήμα 3.3 παρουσιάζει την δομή της ιστοσελίδας στον εξυπηρετητή σε μορφή χάρτη ιστοσελίδας. Όπως φαίνεται και στο υπόμνημα του χάρτη (Σχήμα 3.2), τα στρογγυλεμένα ορθογώνια αναπαριστούν σελίδες και ομάδες σελίδων που είναι προσβάσιμες με τον περιηγητή ιστού, ενώ τα απλά ορθογώνια αναπαριστούν στοιχεία και ομάδες στοιχείων που χρησιμοποιούνται από τις σελίδες.

Σχήμα 3.2: Υπόμνημα Χάρτη Ιστοσελίδας.

Το σύνολο των στοιχείων που ξεκινούν με κάτω παύλα '_' αποτελούν στοιχεία τα οποία χρησιμοποιούνται για εκτέλεση διαφόρων ενεργειών (π.χ. ένωση με την Βάση Δεδομένων), ενώ τα στοιχεία που ξεκινούν με κεφαλαίο γράμμα και έχουν κατάληξη .class αποτελούν κλάσεις αρχείων PHP (PHP class files).

Οι ιστοσελίδες και τα στοιχεία που βρίσκονται κάτω από τον κατάλογο **user/** και **charts/** είναι προσβάσιμα από τους κανονικούς χρήστες μόνο με την δημιουργία λογαριασμού ενώ αυτά που είναι κάτω από τον κατάλογο **admin/** είναι προσβάσιμα μονό με δικαιώματα διαχειριστή. Πιο κάτω γίνεται περιγραφή της κάθε σελίδας και κάθε στοιχείου.



Σχήμα 3.3: Χάρτης ιστοσελίδας στον εξυπηρετητή.

Κεντρικός Κατάλογος:

- index.php: κεντρική σελίδα του συστήματος η οποία χρησιμοποιείται σαν σημείο αναφοράς για τις υπόλοιπες σελίδες.
- register.php: σελίδα που παρέχει την φόρμα δημιουργίας λογαριασμού.
- about.php: σελίδα που παρουσιάζει τα εμπλεκόμενα άτομα στην δημιουργία του συστήματος.

- contact.php: σελίδα που παρέχει την φόρμα επικοινωνίας καθώς και χάρτη τοποθεσίας του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου και ώρες λειτουργίας.
- terms.php: σελίδα με τους όρους αποδοχής δημιουργίας λογαριασμού.
- privacy.php: σελίδα με το privacy policy, μια δήλωση που αποκαλύπτει την χρήση των δεδομένων των πελατών της ιστοσελίδας.
- _actions.php: στοιχείο με την λίστα των ενεργειών που δικαιούται κάθε χρήστης και που εμφανίζεται στο navigation bar της ιστοσελίδας.
- _ch.name.php: στοιχείο που αναλαμβάνει τις απαιτούμενες ενέργειες για την αλλαγή ονόματος κάποιου χρήστη.
- _ch.pass.php: στοιχείο που αναλαμβάνει τις απαιτούμενες ενέργειες για την αλλαγή κωδικού κάποιου χρήστη.
- _common.php: στοιχείο που συμπεριλαμβάνεται σε όλα τα αρχεία και που παρέχει κώδικα που ξεκινά το session στην PHP.
- _conn.php: στοιχείο που συμπεριλαμβάνει τα πιστοποιητικά για σύνδεση με την Βάση Δεδομένων.
- _contact.php: στοιχείο που αναλαμβάνει την αίτηση για επικοινωνία που προέρχεται από την σελίδα contact.php.
- _login.php: στοιχείο που αναλαμβάνει την αίτηση για σύνδεση (login) χρήστη.
- _logout.php: στοιχείο που αναλαμβάνει την αίτηση για αποσύνδεση (logout) χρήστη.
- _register.php: στοιχείο που αναλαμβάνει την αίτηση για δημιουργία λογαριασμού που προέρχεται από την σελίδα register.php.
- img/: κατάλογος με όλες τις εικόνες που χρησιμοποιεί το σύστημα.
- css/: κατάλογος με όλα τα αρχεία CSS που χρησιμοποιεί το σύστημα.
- js/: κατάλογος με όλα τα αρχεία JavaScript που χρησιμοποιεί το σύστημα.
- assets/: κατάλογος με όλες τις βιβλιοθήκες που χρησιμοποιεί το σύστημα.

Κατάλογος User:

- index.php: κύρια σελίδα στον λογαριασμό κανονικών χρηστών. Δίνεται πρόσβαση σε γραφική παράσταση με τον αριθμό των γραφικών παραστάσεων σε μορφή timeline καθώς και σε πίνακα με το ιστορικό των γραφικών παραστάσεων αυτών.

- history_data.php: στοιχείο που ανακτά το ιστορικό των γραφικών παραστάσεων από την Βάση Δεδομένων για τον πίνακα στην κύρια σελίδα χρήστη.
- UserActivity.php: κλάση PHP που ανακτά το ιστορικό του πλήθους των γραφικών παραστάσεων από την Βάση Δεδομένων για την γραφική παράσταση στην κύρια σελίδα χρήστη.
- settings/index.php: σελίδα που παρέχει τις φόρμες για αλλαγή ονόματος και κωδικού πρόσβασης για τον χρήστη.

Κατάλογος Administrator:

- index.php: κύρια σελίδα στον λογαριασμό των διαχειριστών. Δίνεται πρόσβαση σε γραφικές παραστάσεις που παρουσιάζουν το πλήθος των χρηστών χρονικά, το πλήθος των γραφικών παραστάσεων που έχουν δημιουργηθεί χρονικά, την δημοσιότητα των θεμάτων των γραφικών παραστάσεων καθώς και τις πέντε πρώτες σε δημιουργία γραφικές παραστάσεις.
- _subjectsPopularityChartData: στοιχείο που ανακτά την δημοσιότητα των θεμάτων των γραφικών παραστάσεων από την Βάση Δεδομένων για την αντίστοιχη γραφική παράσταση για την κύρια σελίδα των διαχειριστών.
- _topChartsChartData: στοιχείο που ανακτά τις πέντε πρώτες γραφικές παραστάσεις σε δημιουργία από την Βάση Δεδομένων για την αντίστοιχη γραφική παράσταση για την κύρια σελίδα των διαχειριστών.
- _totalChartData: στοιχείο που ανακτά το πλήθος των γραφικών παραστάσεων από την Βάση Δεδομένων για την αντίστοιχη γραφική παράσταση για την κύρια σελίδα των διαχειριστών.
- _usersChartData: στοιχείο που ανακτά το πλήθος των χρηστών από την Βάση Δεδομένων για την αντίστοιχη γραφική παράσταση για την κύρια σελίδα των διαχειριστών.
- users/index.php: σελίδα διαχείρισης χρηστών.
- users/users_data.php: στοιχείο που ανακτά τα στοιχεία των χρηστών από την Βάση Δεδομένων για την σελίδα διαχείρισης χρηστών.
- users/users_manage.php: στοιχείο που αναλαμβάνει την διαχείριση των λογαριασμών των χρηστών στην Βάση Δεδομένων χρησιμοποιώντας AJAX.
- charts/index.php: σελίδα με το ιστορικό όλων των γραφικών παραστάσεων που έχουν δημιουργηθεί.

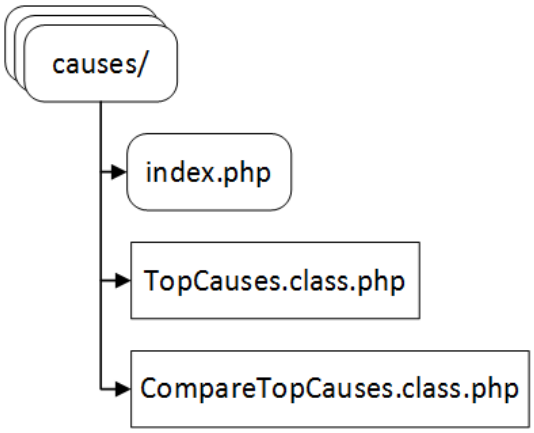
- charts/history_data.php: στοιχείο που ανακτά το ιστορικό όλων των γραφικών παραστάσεων που έχουν δημιουργηθεί για χρήση στην αντίστοιχη σελίδα.
- mail/index.php: σελίδα διαχείρισης αλληλογραφίας.
- mail/mail_data.php: στοιχείο που ανακτά την αλληλογραφία από την Βάση Δεδομένων.
- mail/mail_manage.php: στοιχείο που αναλαμβάνει την διαχείριση της αλληλογραφίας στην Βάση Δεδομένων χρησιμοποιώντας AJAX.
- settings/index.php: σελίδα που παρέχει τις φόρμες για αλλαγή ονόματος και κωδικού πρόσβασης για τον διαχειριστή.

Κατάλογος Charts:

- index.php: σελίδα επιλογής θέματος δημιουργίας γραφικής παράστασης.
- _countries.php: στοιχείο με λίστα των χωρών που υποστηρίζονται από το σύστημα η οποία συμπεριλαμβάνεται στις διάφορες φόρμες.
- _degree_levels.php: στοιχείο με την λίστα με διάφορα επίπεδα πτυχίου η οποία συμπεριλαμβάνεται στις διάφορες φόρμες.
- _degrees.php: στοιχείο με τα ονόματα πτυχίων που συμπεριλαμβάνονται στις διάφορες φόρμες.
- _job_positions.php: στοιχείο με την λίστα ονομάτων θέσεων εργασίας που συμπεριλαμβάνονται στις διάφορες φόρμες.
- _languages.php: στοιχείο με την λίστα ομιλούμενων γλωσσών που συμπεριλαμβάνονται στις διάφορες φόρμες.
- _popular_subject.php: στοιχείο που επικοινωνεί με την Βάση Δεδομένων για ανεύρεση του δημοφιλέστερου θέματος δημιουργίας γραφικής παράστασης.
- _schools.php: στοιχείο με την λίστα σχολείων που συμπεριλαμβάνονται στις διάφορες φόρμες.
- _skills.php: στοιχείο με την λίστα δεξιοτήτων που συμπεριλαμβάνονται στις διάφορες φόρμες.
- Breadcrumbs.class.php: κλάση PHP που δημιουργεί την διαδρομή καταλόγου που ακολουθήθηκε (directory path) στην γραμμή πλοήγησης.
- Chart.class.php: η συγκεκριμένη κλάση PHP αποτελεί πατρική κλάση για όλες τις κλάσεις που εμπλέκονται στην δημιουργία γραφικών παραστάσεων

ανάλυσης του LinkedIn, υλοποιώντας συναρτήσεις που χρησιμοποιούνται από όλες τις άλλες κλάσεις.

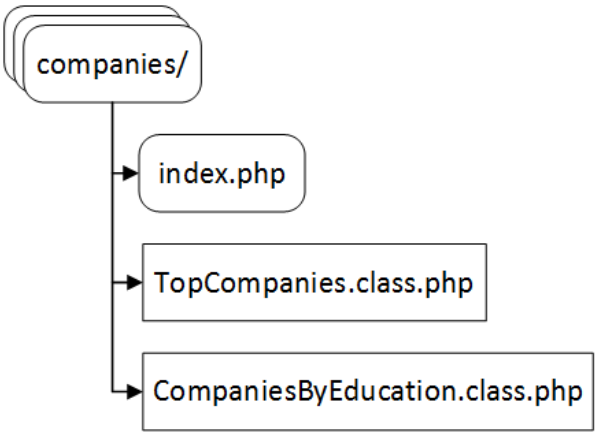
Υποκατάλογος causes:



Σχήμα 3.4: Υποκατάλογος causes.

- index.php: σελίδα επιλογής γραφικής παράστασης με θέμα «Ενδιαφέροντα».
- TopCauses.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης των «100 Δημοφιλέστερων Ενδιαφερόντων».
- CompareTopCauses.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης συγκρίσεως των «100 Δημοφιλέστερων Ενδιαφερόντων» μεταξύ χωρών.

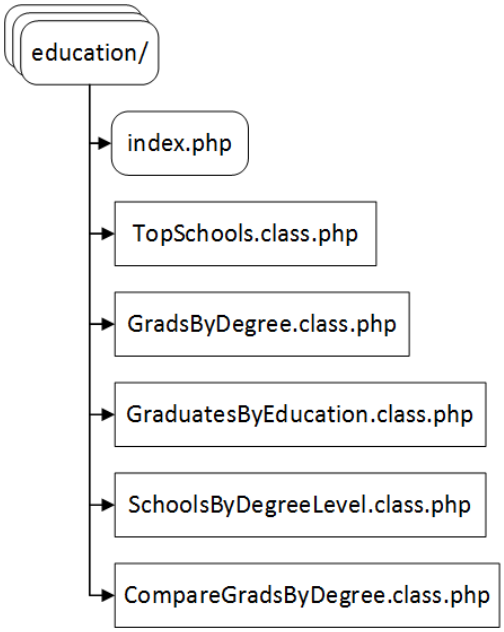
Υποκατάλογος companies:



Σχήμα 3.5: Υποκατάλογος companies.

- index.php: σελίδα επιλογής γραφικής παράστασης με θέμα «Εταιρείες».
- TopCompanies.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης των «100 Πρώτες Εταιρείες» βάση αριθμό εργαζομένων.
- CompaniesByEducation.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «Εταιρείες βάση εργαζομένων με συγκεκριμένο πτυχίο».

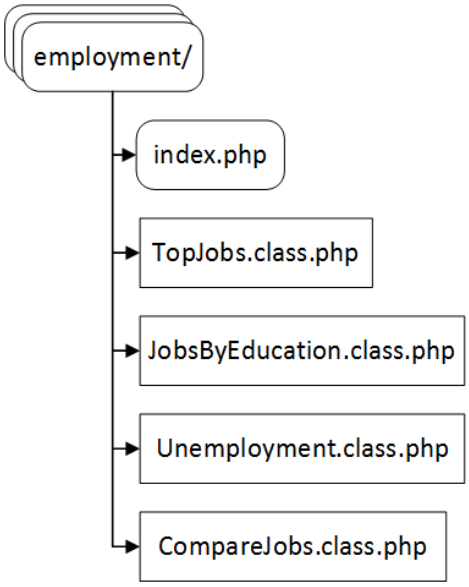
Υποκατάλογος education:



Σχήμα 3.6: Υποκατάλογος education.

- index.php: σελίδα επιλογής γραφικής παράστασης με θέμα «Μόρφωση».
- TopSchools.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης των «100 Πρώτων Σχολείων» βάση αριθμό φοιτητών.
- GradsByDegree.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «Πτυχιούχοι Βάση Επιπέδου Πτυχίου».
- GraduatesByEducation.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «Πτυχιούχοι Βάση Μόρφωσης».
- SchoolsByDegreeLevel.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «Σχολεία βάση πτυχιούχων με συγκεκριμένο επίπεδο πτυχίου».
- CompareGradsByDegree.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης συγκρίσεως «Πτυχιούχοι Βάση Επιπέδου Πτυχίου» μεταξύ χωρών.

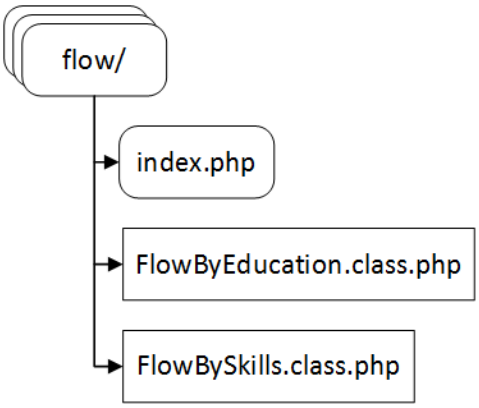
Υποκατάλογος employment:



Σχήμα 3.7: Υποκατάλογος employment.

- index.php: σελίδα επιλογής γραφικής παράστασης με θέμα «Εργασία».
- TopJobs.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης των «100 Πρώτων Θέσεων Εργασίας» βάση αριθμό εργαζομένων.
- JobsByEducation.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «Θέσεις Εργασίας Βάση Πτυχίου».
- Unemployment.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «Ανεργία».
- CompareJobs.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «Σύγκριση Θέσεων Εργασίας».

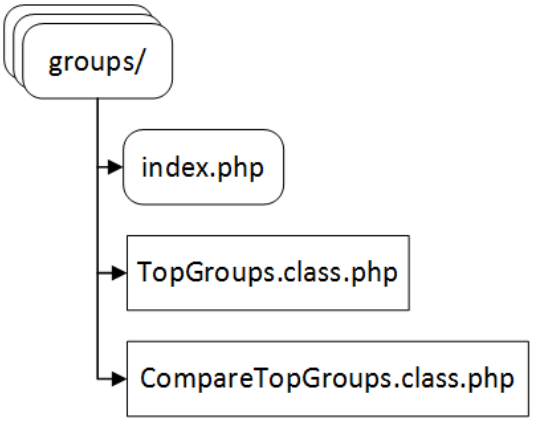
Υποκατάλογος flow:



Σχήμα 3.8: Υποκατάλογος flow.

- index.php: σελίδα επιλογής γραφικής παράστασης με θέμα «Ροή Εργαζομένων».
- FlowByEducation.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «Ροή Εργαζομένων Βάση Μόρφωσης».
- FlowBySkills.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «Ροή Εργαζομένων Βάση Δεξιοτήτων».

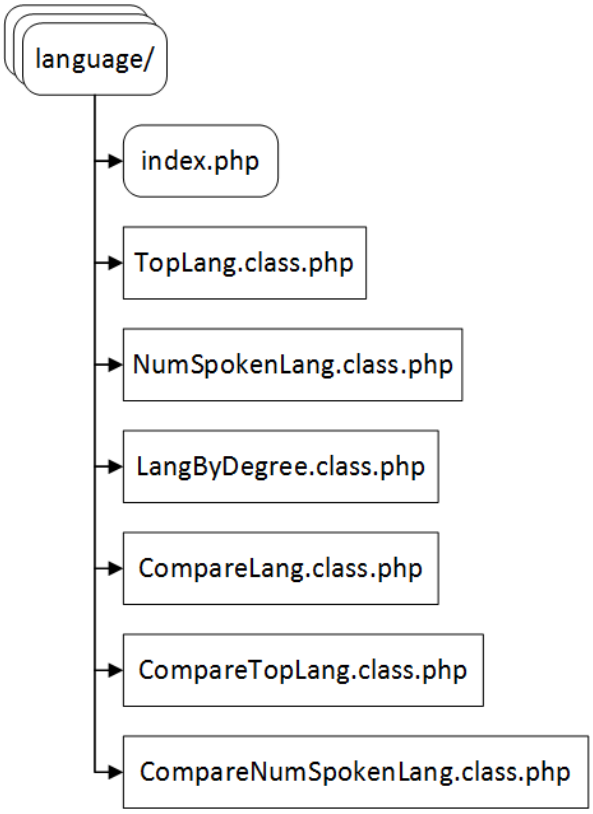
Υποκατάλογος groups:



Σχήμα 3.9: Υποκατάλογος groups.

- index.php: σελίδα επιλογής γραφικής παράστασης με θέμα «Ομάδες».
- TopGroups.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «100 Πρώτες Ομάδες» για τις οποίες ανήκουν οι χρήστες του LinkedIn.
- CompareTopGroups.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης σύγκρισης μεταξύ χωρών των «100 Πρώτων Ομάδων» στις οποίες ανήκουν οι χρήστες του LinkedIn.

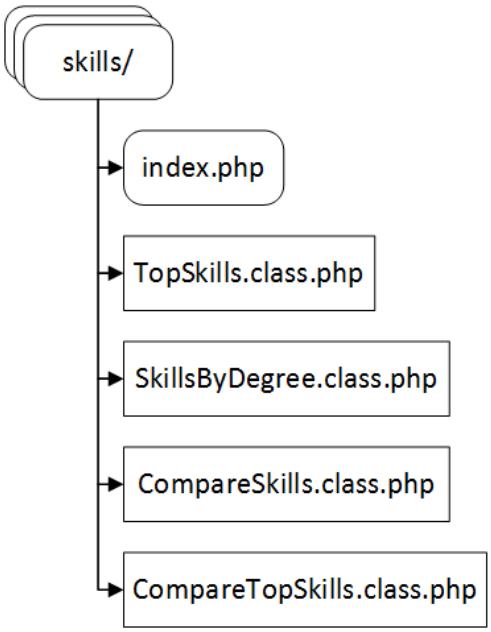
Υποκατάλογος language:



Σχήμα 3.10: Υποκατάλογος language.

- index.php: σελίδα επιλογής γραφικής παράστασης με θέμα «Γλώσσες».
- TopLang.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «100 Πρώτες Ομιλούμενες Γλώσσες».
- NumSpokenLang.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «Γνώση Πολλαπλών Γλωσσών».
- LangByDegree.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «Γνώση Γλωσσών Βάση Πτυχίου».
- CompareLang.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «Σύγκριση Ομιλούμενων Γλωσσών».
- CompareTopLang.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «Σύγκριση Ομιλούμενων Γλωσσών» μεταξύ χωρών.
- CompareNumSpokenLang.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «Σύγκριση Γνώσης Πολλαπλών Γλωσσών» μεταξύ χωρών.

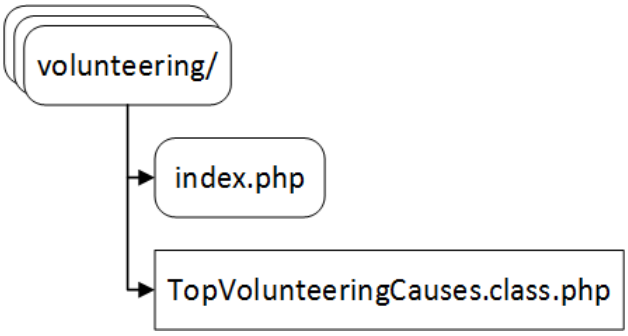
Υποκατάλογος skills:



Σχήμα 3.11: Υποκατάλογος skills.

- index.php: σελίδα επιλογής γραφικής παράστασης με θέμα «Δεξιότητες».
- TopSkills.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «100 Πρώτες Δεξιότητες».
- SkillsByDegree.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «Δεξιότητες Βάση Πτυχίου».
- CompareSkills.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «Σύγκριση Δεξιοτήτων».
- CompareTopSkills.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης σύγκρισης «100 Πρώτων Δεξιοτήτων» μεταξύ χωρών.

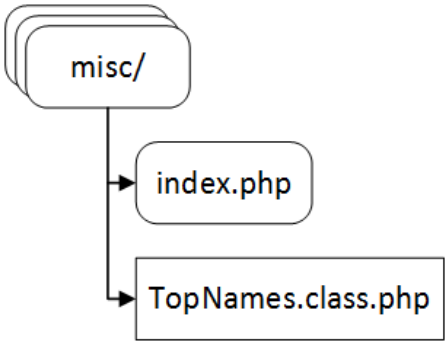
Υποκατάλογος volunteering:



Σχήμα 3.12: Υποκατάλογος volunteering.

- index.php: σελίδα επιλογής γραφικής παράστασης με θέμα «Εθελοντισμός».
- TopVolunteeringCauses.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «100 Πρώτα Θέματα Εθελοντισμού».

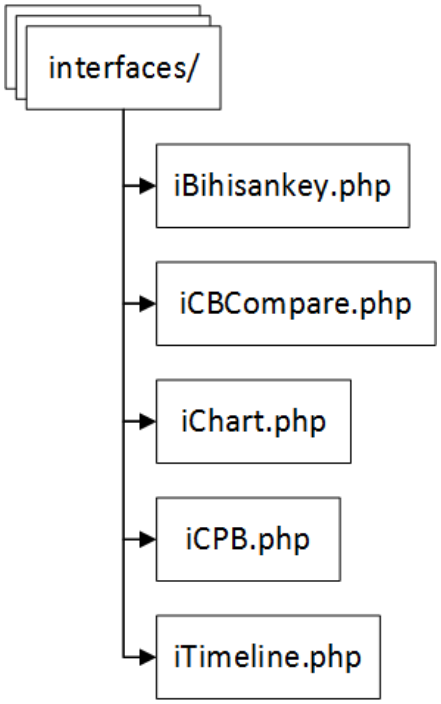
Υποκατάλογος miscellaneous:



Σχήμα 3.13: Υποκατάλογος miscellaneous.

- index.php: σελίδα επιλογής γραφικής παράστασης με θέμα «Διάφορα».
- TopNames.class.php: κλάση PHP για την δημιουργία γραφικής παράστασης «100 Πρώτα Δημοφιλέστερα Ονόματα».

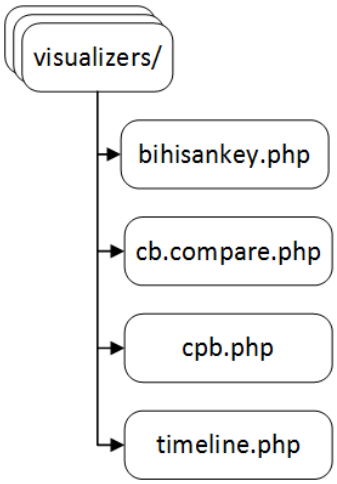
Υποκατάλογος interfaces:



Σχήμα 3.14: Υποκατάλογος interfaces.

- iChart.php: πατρική PHP class interface στην οποία γίνονται extend όλες οι υπόλοιπες.
- iCPB.php: PHP class interface την οποία υλοποιούν οι PHP classes που χρησιμοποιούν τον cpb visualizer.
- iTimeline.php: PHP class interface την οποία υλοποιούν οι PHP classes που χρησιμοποιούν τον timeline visualizer.
- iCBCompare.php: PHP class interface την οποία υλοποιούν οι PHP classes που χρησιμοποιούν τον cb.compare visualizer.
- iBihisankey.php: PHP class interface την οποία υλοποιούν οι PHP classes που χρησιμοποιούν τον bihisankey visualizer.

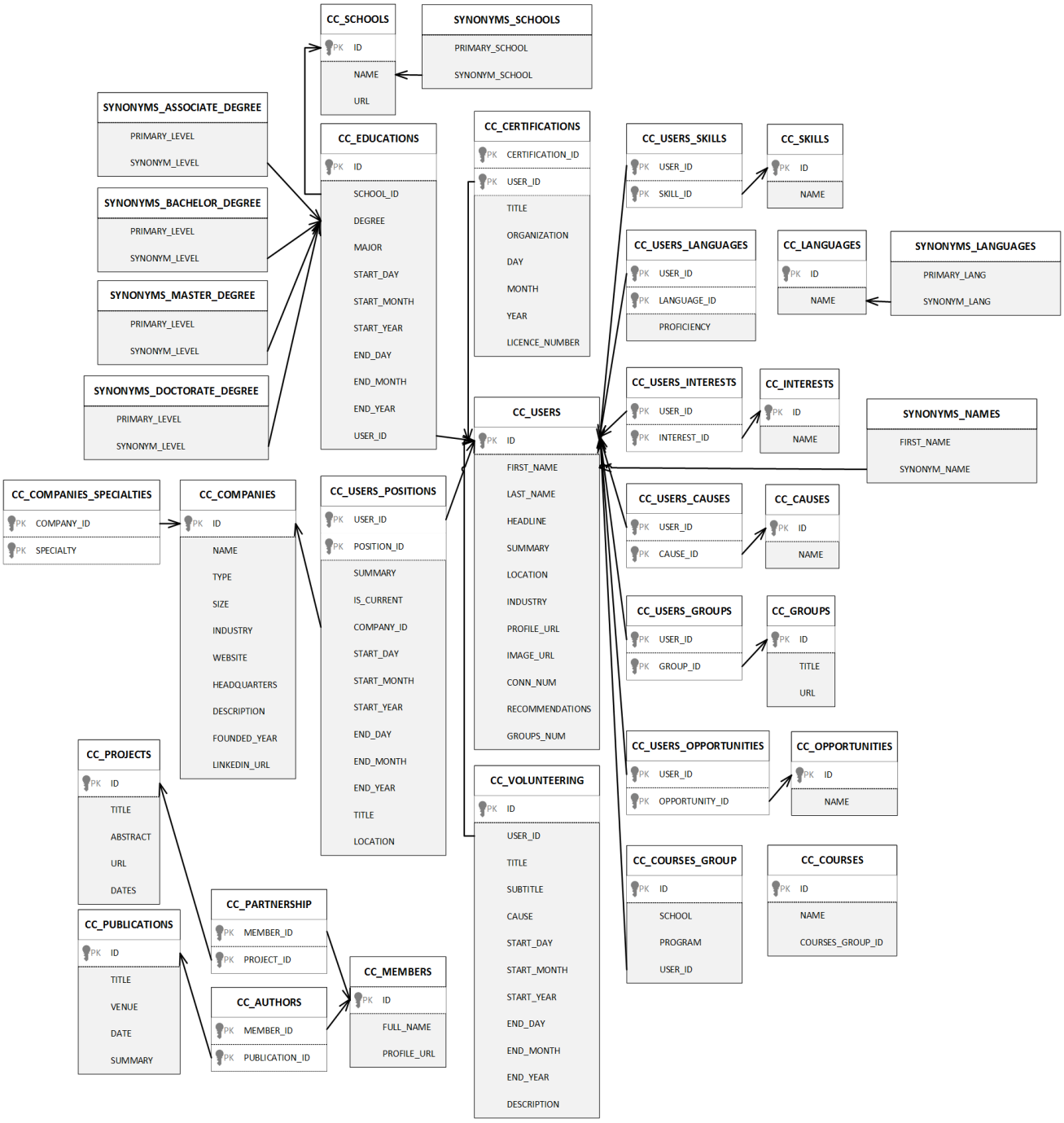
Υποκατάλογος visualizers:



Σχήμα 3.15: Υποκατάλογος visualizers.

- bihisankey.php: σελίδα με τον visualizer που αναπαριστά την ροή χρησιμοποιώντας Sankey diagram.
- cb.compare.php: σελίδα με τον visualizer σύγκρισης χωρών χρησιμοποιώντας Column Chart και Bar Chart.
- cpb.php: σελίδα με τον visualizer που χρησιμοποιεί Column Chart, Pie Chart και Bar Chart.
- timeline.php: σελίδα με τον visualizer που χρησιμοποιεί γραφική αναπαράσταση με γραμμή χρόνου.

3.4. Δομή Βάσης Δεδομένων (Relational Model)

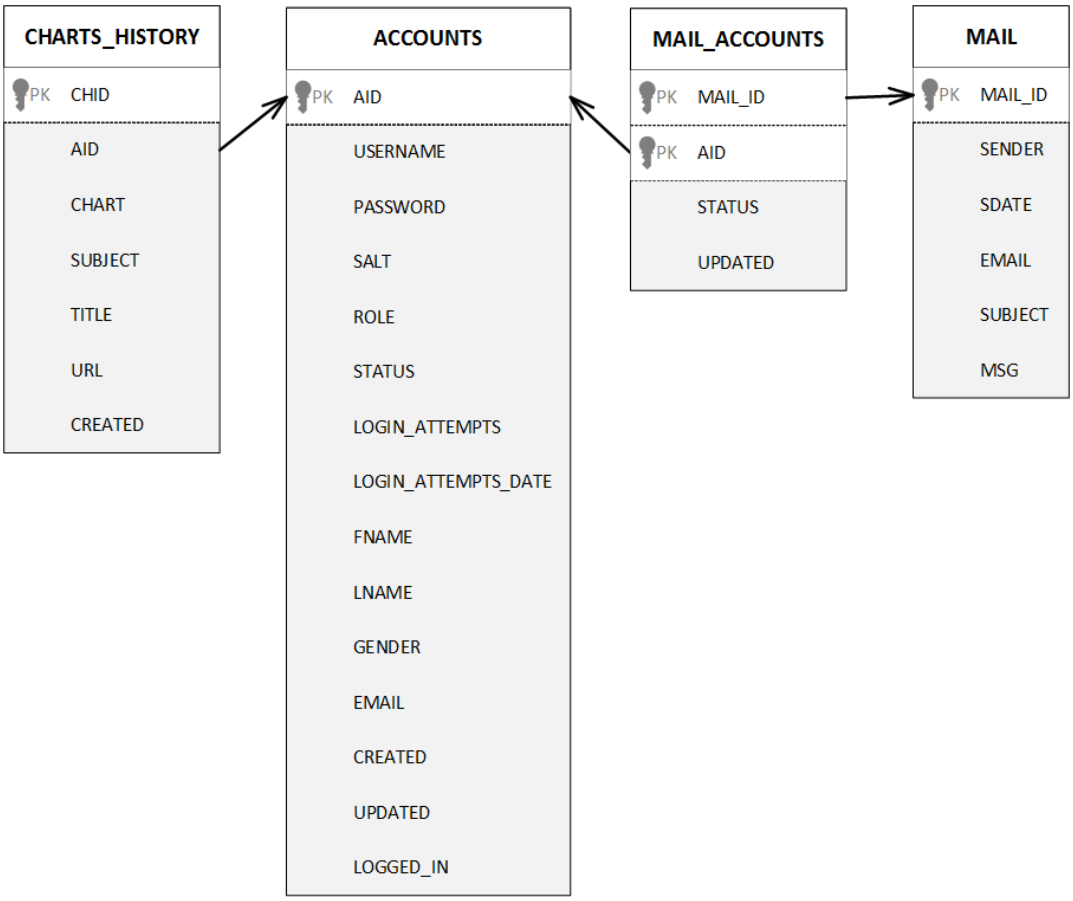


Σχήμα 3.16: Database Relational Model (Part A).

Το Σχήμα 3.16 παρουσιάζει το πρώτο μέρος του Σχεσιακού Μοντέλου της Βάσης Δεδομένων. Στους συγκεκριμένους πίνακες αποθηκεύονται όλες οι πληροφορίες που συλλέγονται από τον crawler (βλ. 2.3 Αποθήκευση Δεδομένων). Για διευκόλυνση της παρουσίασης των πινάκων και για τις δύο χώρες έχει αντικατασταθεί το πρόθεμα των πινάκων για την κάθε χώρα με το πρόθεμα "CC_". Αυτό σημαίνει ότι υπάρχει ένα σετ με τους πιο πάνω πίνακες που ξεκινά με "CY_" πρόθεμα για την Κύπρο και ακόμη ένα σετ που ξεκινά με "EE_" για την Εσθονία.

Αξίζει να σημειωθεί ότι μέσα από την ανάλυση των πληροφοριών που συλλέχθηκαν από το LinkedIn έχει παρατηρηθεί η εμφάνιση εγγραφών που περιέχουν συνώνυμα. Για τον λόγο ότι η MySQL δεν υποστηρίζει την δημιουργία και συσχέτιση συνωνύμων, υπήρχε η ανάγκη δημιουργίας πινάκων που βοηθούν στην επίλυση αυτού του προβλήματος. Ο τρόπος με τον οποίο δουλεύει η συγκεκριμένη λύση έχει ως εξής: δημιουργείται ένας πίνακας με δύο στήλες, μια για τα διάφορα συνώνυμα και μια για την πρωτεύων λέξη που αντιστοιχούν. Στην συνέχει ο «προβληματικός» πίνακας γίνεται LEFT JOIN με τον πίνακα συνωνύμων βάση της στήλης συνωνύμων και GROUP BY (αν χρειάζεται) βάση της στήλης με την πρωτεύων λέξη.

Πιο κάτω, το Σχήμα 3.17 παρουσιάζει το δεύτερο μέρος του Σχεσιακού Μοντέλου της Βάσης Δεδομένων. Στους συγκεκριμένους πίνακες αποθηκεύονται όλες οι σχετικές πληροφορίες για την ιστοσελίδα, οι οποίες συμπεριλαμβάνουν τους λογαριασμούς των χρηστών, το ιστορικό δημιουργίας των γραφικών παραστάσεων καθώς επίσης και την αλληλογραφία με την διαχείρισή της.



Σχήμα 3.17: Database Relational Model (Part B).

Κεφάλαιο 4

Υλοποίηση Συστήματος

4.1. Εισαγωγή

Το συγκεκριμένο κεφάλαιο είναι αφιερωμένο στην περιγραφή των κυριότερων σημείων υλοποίησης του συστήματος. Συγκεκριμένα, γίνεται αναφορά σε διάφορα θέματα ασφάλειας όπως η αποθήκευση κωδικού πρόσβασης των χρηστών του συστήματος, επεξήγηση των κλάσεων PHP για την δημιουργία των γραφικών παραστάσεων χρησιμοποιώντας την Ενοποιημένη Γλώσσα Μοντελοποίησης (UML) καθώς και κομμάτια κώδικα με κάποια ιδιαίτερη σημασία. Παρόλα αυτά, όλος ο κώδικας καθώς και στιγμιότυπα οθόνης συμπεριλαμβάνονται στα Παραρτήματα.

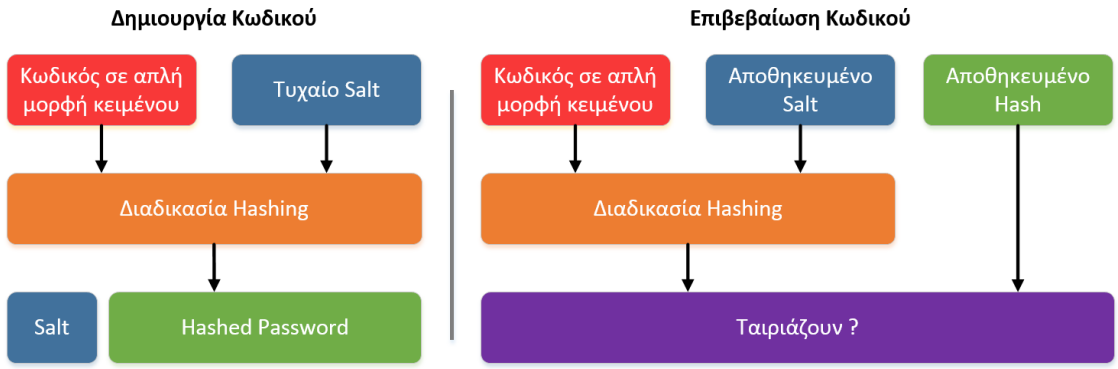
4.2. Θέματα Ασφάλειας

Κρυπτογραφημένη Αποθήκευση Κωδικού Πρόσβασης

Ένα από τα σημαντικότερα θέματα ασφάλειας είναι η αποθήκευση του κωδικού πρόσβασης των χρηστών στην Βάση Δεδομένων. Για την επίτευξη αυτής της διαδικασίας είναι απαραίτητη η χρήση κρυπτογράφησης του κωδικού μέσω μιας διαδικασίας που ονομάζεται Hashing and Salt[13]. Η κρυπτογράφηση ενός κωδικού μόνο με την διαδικασία Hashing είναι ευάλωτη σε πολλές επιθέσεις όπως dictionary-attacks[12] και pre-computed rainbow table attacks[14]. Παρόλα αυτά με την πρόσθεση τυχαίου αριθμού/λέξης στην εξίσωση, γνωστό ως Salt, μπορεί να υπερασπιστεί ενάντιας τέτοιων επιθέσεων.

Η διαδικασία έχει ως εξής (βλ. Σχήμα 4.1): Κατά την δημιουργία του κωδικού, ο κωδικός σε απλή μορφή κειμένου από την φόρμα εγγραφής (register) μαζί με ένα τυχαίο Salt, που στην περίπτωση του παρών συστήματος είναι ο χρόνος σε milliseconds, περνούν μέσο της διαδικασίας Hashing. Για τον λόγω ότι το Salt είναι μοναδικό για τον κάθε κωδικό, αυτό σημαίνει ότι όμοιοι κωδικοί θα κρυπτογραφηθούν διαφορετικά. Στην συνέχεια αποθηκεύεται στην Βάση Δεδομένων ο κρυπτογραφημένος κωδικός και το Salt που έχει χρησιμοποιηθεί.

Στην συνέχεια για την επιβεβαίωση του κωδικού η διαδικασία έχει ως εξής: Βάση του username (αν υπάρχει) γίνεται λήψη του αποθηκευμένου Hash κωδικού και Salt. Στην συνέχεια, ο κωδικός σε απλή μορφή κειμένου από την φόρμα σύνδεσης (login), μαζί με το αποθηκευμένο Salt, περνούν από την διαδικασία Hashing. Αν ο παραγόμενος hashed κωδικός ταιριάζει με τον αποθηκευμένο, τότε θα γίνει επιτρεπτή η σύνδεση στο χρήστη.



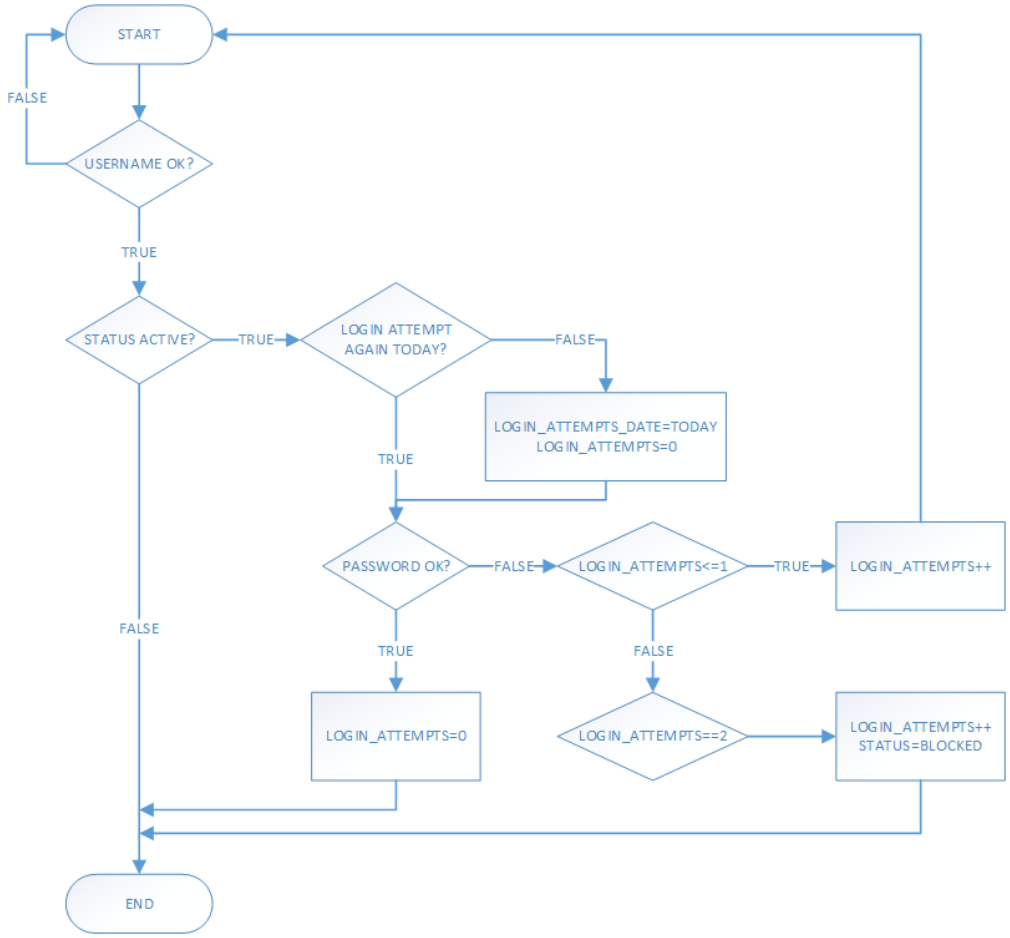
Σχήμα 4.1: Διαδικασία Hashing and Salt κωδικού.

Διαδικασία Σύνδεσης Λογαριασμού

Σε συνδυασμό με το πιο πάνω μέτρο ασφάλειας είναι σημαντική και η φραγή κάποιου λογαριασμού μετά από κάποιο αριθμό λανθασμένων καταχωρήσεων κωδικού. Στην περίπτωση που κάποιος προσπαθεί να μαντέψει το κωδικό δοκιμάζοντας συνεχώς διάφορους κωδικούς, θα πρέπει να υπάρχει μια δικλίδα ασφαλείας για την φραγή αυτού του λογαριασμού μετά από τρείς λανθασμένες προσπάθειες.

Επομένως η διαδικασία σύνδεσης στο σύστημα έχει ως εξής (Βλ. Σχήμα 4.2): Με την υποβολή της φόρμας σύνδεσης εξετάζεται πρώτα το username. Αν δεν υπάρχει τότε το

σύστημα επιστρέφει το ανάλογο μήνυμα λάθους και επιστρέφει στην σελίδα σύνδεσης. Αν υπάρχει, γίνεται έλεγχος κατάστασης του λογαριασμού. Αν ο λογαριασμός είναι φραγμένος λόγω εξάντλησης των προσπαθειών εισαγωγής κωδικού ή έχει ανακαλεστεί από τον διαχειριστή, το σύστημα επιστρέφει το ανάλογο μήνυμα λάθους. Εναλλακτικά, γίνεται έλεγχο αν ο χρήστης έχει ξαναδοκιμάσει να συνδεθεί την παρών ημέρα. Ο έλεγχος αυτός αποσκοπεί στον μηδενισμό των λανθασμένων προσπαθειών με την αλλαγή της ημέρας.



Σχήμα 4.2: Διαδικασία Σύνδεσης Λογαριασμού.

Προχωρώντας, στην συνέχεια γίνεται έλεγχος του κωδικού με τον τρόπο πού έχει περιγράφει πιο πάνω. Αν ο κωδικός είναι σωστός, τότε οι αποτυχημένες προσπάθειες για αυτή την ημέρα μηδενίζονται και παραχωρείται πρόσβαση στον χρήστη. Στην περίπτωση που ο κωδικός είναι λανθασμένος, αυξάνεται ο αριθμός των προσπαθειών πρόσβασης για την συγκεκριμένη ημέρα. Αν ο αριθμός έχει φτάσει τις τρείς φορές, φράζεται ο λογαριασμός και το σύστημα επιστρέφει το ανάλογο μήνυμα.

SQL Injection

Ακόμη μια διαδεδομένη μέθοδος επίθεσης σε διαδικτυακές εφαρμογές είναι η SQL Injection. Ο τρόπος που επιτυγχάνεται τέτοιου είδους επίθεση είναι εισάγοντας κάποιες συγκεκριμένες εντολές σε γλώσσα SQL στα δεδομένα εισόδου. Στην περίπτωση ελλιπούς επαλήθευσης των δεδομένων εισόδου, είναι δυνατό σε κάποιες περιπτώσεις να περαστούν extra εντολές SQL, κακόβουλες για το σύστημα. Ο τρόπος που προστατεύεται το σύστημα από τέτοιου είδους επιθέσεις είναι εκτελώντας εξονυχιστικούς ελέγχους επαλήθευσης στα δεδομένα εισόδου και χρησιμοποιώντας την συνάρτηση (για PHP) `mysql_real_escape_string($value)` όπου μετατρέπει τους ειδικούς χαρακτήρες (π.χ. μονά ή διπλά εισαγωγικά) που χρησιμοποιούνται στις εντολές SQL σε χαρακτήρες κειμένου.

4.3. Περιγραφή Υλοποίησης Συστήματος

Σε αυτό το υποκεφάλαιο θα γίνει αναφορά στα πιο σημαντικά σημεία του κώδικα του συστήματος. Για ολοκληρωμένο κώδικα για κάθε αρχείο μπορείτε να αποταθείτε στα Παραρτήματα.

Στην κορυφή του κάθε αρχείου συμπεριλαμβάνονται σύνδεσμοι για τα αρχεία _common.php και _conn.php. Το αρχείο _common.php περιλαμβάνει τον κώδικα για την έναρξη του session της PHP, τα links για τα social media που χρησιμοποιούνται στα footers όλων των σελίδων καθώς και την συνάρτηση για αντιμετώπιση των SQL Injections.

_common.php:

```
<?php
	session_start();
	$Social_Mail = "mailto:chartii@example.com";
	$Social_Facebook = "#";
	$Social_Twitter = "#";

	function dbSafe($value) {
		return '"'.mysql_real_escape_string($value).'"';
	}
?>
```

Το αρχείο _conn.php περιλαμβάνει τα credentials και την συνάρτηση για ένωση με την Βάση Δεδομένων.

_conn.php:

```
<?php
	$host='localhost';
	$dbu='[redacted]';
	$dbp='[redacted]';
	$db='test';

	$conn = mysqli_connect($host, $dbu, $dbp, $db);
?>
```

Στην συνέχεια, εντός του <head> tag των αρχείων συμπεριλαμβάνεται οι διάφορες μεταπληροφορίες (charset, description, author), σύνδεσμοι στα αρχεία με τους κανόνες μορφοποίησης καθώς και σύνδεσμοι στις διάφορες βιβλιοθήκες που χρησιμοποιούνται (bootstrap, amCharts).

<head>...</head>:

```
<!DOCTYPE html>
<html lang="en">
<!-- HEAD -->
<head>
	<meta charset="utf-8">
	<meta http-equiv="X-UA-Compatible" content="IE=edge">
	<meta name="viewport" content="width=device-width, initial-
scale=1">
	<meta name="description" content="Welcome to the Chartii Home
page">
	<meta name="author" content="Andreas Andreou">
	<link rel="icon" href="img/favicon.ico">
	<title>Chartii: Home</title>
	<link rel="stylesheet" href="css/bootstrap.min.css" >
	<link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/font-
awesome/4.3.0/css/font-awesome.min.css">
	<link rel='stylesheet'
href='http://fonts.googleapis.com/css?family=Lato' type='text/css'>
	<link rel="stylesheet" href="css/style.css">
	<link rel="stylesheet" href="css/animate.css">
	<link rel="stylesheet" href="css/scrolling-nav.css">
	<script src="js/wow.min.js"></script>
	<script> new WOW().init(); </script>
	<!-- amCharts -->
	<script type="text/javascript"
src="assets/amcharts/amcharts.js"></script>
	<script type="text/javascript"
src="assets/amcharts/serial.js"></script>
```

```
      <script type="text/javascript"
src="assets/amcharts/pie.js"></script>
      <script type="text/javascript"
src="assets/amcharts/themes/light.js"></script>
      <script type="text/javascript"
src="assets/amcharts/themes/black.js"></script>
      <script type="text/javascript"
src="assets/amcharts/plugins/responsive/responsive.js"></script>
</head>
<!-- END OF HEAD -->
```

Προχωρώντας εντός του <body> tag, σε όλα τα αρχεία υπάρχει το navigation bar, το οποίο παρέχει καθολική πλοήγηση στο σύστημα παρέχοντας συνδέσμους για την Οικοσελίδα, τις γραφικές παραστάσεις, το About Us page, το Contact Us page, την λίστα ενεργειών του κάθε χρήστη καθώς και για το login pop up.

<nav>...</nav>:

```
<!-- Navigation Bar -->
<nav class="navbar navbar-inverse navbar-fixed-top">
 <div class="container">
  <div class="navbar-header">
   <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#navbar" aria-expanded="false" aria-
controls="navbar">
    <span class="sr-only">Toggle navigation</span>
    <span class="icon-bar"></span>
    <span class="icon-bar"></span>
    <span class="icon-bar"></span>
   </button>
   <a class="navbar-brand" href="index.php">Chartii</a>
  </div>
  <div id="navbar" class="navbar-collapse collapse">
   <ul class="nav navbar-nav navbar-right">
    <li class="active"><a href="index.php"><i class="fa fa-home">
</i> Home</a></li>
    <li><a href="charts/"><i class="fa fa-area-chart"></i> 
Charts</a></li>
    <li><a href="about.php"><i class="fa fa-exclamation-circle">
</i> About Us</a></li>
```

```
    <li><a href="contact.php"><i class="fa fa-envelope"> </i> 
Contact Us</a></li>
    <!-- If user hasn't logged in, show login/register
buttons/options -->
    <?php if (!isset($_SESSION['AID'])){ ?>
    <li><button type="button" class="btn btn-primary navbar-btn
hidden-xs" data-toggle="modal" data-target="#modal_login" aria-
expanded="false">Sign in <i class="fa fa-sign-in"></i>
</button></li>
    <li><a href="register.php" class="visible-xs"><i class="fa fa-
pencil"></i> Register</a></li>
    <li><a href="#" type="button" class="visible-xs" data-
toggle="modal" data-target="#modal_login" aria-expanded="false"><i
class="fa fa-sign-in"></i> Sign in</a></li>
    <?php }else{ echo $actions_ls; } ?>
   </ul>
  </div>
 </div>
</nav>
<!-- /Navigation Bar -->
```

Αμέσως μετά το navigation bar, ακολουθεί το login modal (pop up), όπου ο χρήστης μπορεί να συμπληρώσει το username και password του για πρόσβαση στον λογαριασμό του.

Login modal:

```
<!-- Login Modal -->
<div class="modal fade" id="modal_login" tabindex="-1"
role="dialog" aria-labelledby="myModalLabel" aria-hidden="true">
 <div class="modal-dialog">
  <div class="modal-content">
   <div class="modal-header">
    <button type="button" class="close" data-dismiss="modal" aria-
label="Close"
onclick="getElementById('error_msg').innerHTML=''"><span aria-
hidden="true">&times;</span></button>
```

```
    <h4   class="modal-title"   id="myModalLabel">Sign   in <i
class="fa fa-sign-in"></i></h4>
   </div>
   <form method="post" action="_login.php">
    <div class="modal-body">
     <p  class="text-danger"  name="error_msg"  id="error_msg"><?php
echo $err_msg; ?></p>
     <div class="form-group">
      <label    for="username"><i    class="fa    fa-user"></i> 
Username</label>
      <input   type="text"   class="form-control"   name="username"
id="username" placeholder="username..." required>
     </div>
     <div class="form-group">
      <label         for="password"><i         class="fa         fa-
lock"></i> Password</label>
      <input  type="password"  class="form-control"  name="password"
id="password" placeholder="password..." required>
     </div>
     <div class="form-group checkbox">
      <label><input  type="checkbox"  name="remember"  id="remember"
value="1">Remember me</label>
     </div>
     <p><i  class="fa  fa-exclamation-circle"></i> If  you  are
not registered yet, click <a href="register.php">here to join.</a>
</p>
     <input   type="hidden"   name="current_page"   id="current_page"
value="index.php">
    </div>
    <div class="modal-footer">
     <button type="submit" class="btn btn-primary">Login</button>
     <button  type="button"  class="btn  btn-default"  data-dismiss=
"modal"
onclick="getElementById('error_msg').innerHTML=''">Cancel</button>
    </div>
   </form>
  </div>
 </div>
```

```
</div>
<!-- /Login Modal -->
```

Στο κάτω μέρος κάθε ιστοσελίδας υπάρχει το footer, το οποίο παρέχει συνδέσμους για τα κοινωνικά δίκτυα, τα terms, το privacy καθώς και κουμπί για κύλιση (scroll) στο πάνω μέρος της κάθε σελίδας.

Footer:

```
<!-- #Footer -->
<div id="footer">
 <div class="container">
  <div class="col-xs-12 col-sm-4 text-center">
   <h4>&copy; Chartii 2015<br>
   [ <a href="terms.php" target="_blank">Terms</a> | <a href="privacy.php" target="_blank">Privacy</a> ]</h4><br>
  </div>
  <div class="col-xs-12 col-sm-4 col-sm-push-4 text-center">
   <a class="mailBtn socialBtn" href="<?php echo $Social_Mail;?>"><i class="fa fa-envelope"></i></a>
   <a class="facebookBtn socialBtn" href="<?php echo $Social_Facebook;?>"><i class="fa fa-facebook"></i></a>
   <a class="twitterBtn socialBtn" href="<?php echo $Social_Twitter;?>" ><i class="fa fa-twitter"></i></a><br><br>
  </div>
  <div class="col-xs-12 col-sm-4 col-sm-pull-4 text-center">
   <a role="button" class="btn btn-default btn-lg btn-tr btn-tr-default btn-circle page-scroll wow rubberBand" href="#wrapper"><i class="fa fa-angle-double-up fa-2x"></i></a>
   <br>
  </div>
 </div>
</div>
<!-- /#Footer -->
```

Η υλοποίηση της διαδικασίας δημιουργίας λογαριασμού πραγματοποιείται στο αρχείο _register.php και η διαδικασία του login πραγματοποιείται στο αρχείο _login.php. Και τα δύο αρχεία ακολουθούν τα βήματα που περιγράφονται στο υποκεφάλαιο 4.2 Θέματα Ασφάλειας, για προστασία από διάφορες επιθέσεις.

_register.php:

```
<?php
	include '_common.php';
	include('_conn.php');

	/***** Get user's submission *****/
	// Username
	if (isset($_POST['username'])){
		$Username = dbSafe($_POST['username']);
		$_SESSION['RF_Username']=$_POST['username'];
	}
	// Password1
	if (isset($_POST['password'])){
		$Password = $_POST['password'];
		$_SESSION['RF_Password']=$_POST['password'];
	}
	// Password2
	if (isset($_POST['password2'])){
		$Password2 = $_POST['password2'];
		$_SESSION['RF_Password2']=$_POST['password2'];
	}
	// Name
	if (isset($_POST['fname'])){
		$Fname = dbSafe($_POST['fname']);
		$_SESSION['RF_Fname']=$_POST['fname'];
	}
	// Surname
	if (isset($_POST['lname'])){
		$Lname = dbSafe($_POST['lname']);
		$_SESSION['RF_Lname']=$_POST['lname'];
	}
	// Gender
	if (isset($_POST['gender'])){
		$Gender = dbSafe($_POST['gender']);
		$_SESSION['RF_Gender']=$_POST['gender'];
	}
	// Email
	if (isset($_POST['email'])){
		$Email = dbSafe($_POST['email']);
		$_SESSION['RF_Email']=$_POST['email'];
	}
	// Terms
	if (isset($_POST['terms']) && $_POST['terms']){
		$Terms = 1;
		$_SESSION['RF_Terms']="checked";
	}
	// Role
	if (isset($_POST['role'])) dbSafe($Role=$_POST['role']);
```

```
      else $Role=dbSafe(1);

      $goPage = $_POST['current_page'];

      /***** Check user's submission *****/
      // Check if user accepted terms and conditions
      if ($Terms!=1){
            $_SESSION['Error_code']="RF1";
            $_SESSION['Error_msg']="Registration failed! You haven't
accept the terms and conditions. (Error: RF1)";
            header('location:'.$goPage);
            die();
      }

      // Check if password was confirmed correctly
      if ($_SESSION['RF_Password']!=$_SESSION['RF_Password2']){
            $_SESSION['Error_code']="RF2";
            $_SESSION['Error_msg']="Registration failed! Password
mismatch. (Error: RF2)";
            header('location:'.$goPage);
            die();
      }

      // Check Database Connection
      if (!$conn) {
            $_SESSION['Error_code']="DB1";
            $_SESSION['Error_msg']="Connection with the database
failed. (Error: DB1)";
            header('location:'.$goPage);
            die();
      }

      // Check if username exists
      $sql='SELECT * FROM ACCOUNTS WHERE USERNAME='.$Username.'';
      $result=mysqli_query($conn, $sql);
      $count=mysqli_num_rows($result);

      if ($count>0){
            $_SESSION['Error_code']="RF3";
            $_SESSION['Error_msg']="Registration failed! Username
already exists. (Error: RF3)";
            header('location:'.$goPage);
            die();
      }

      // Check username length
      if (strlen($_SESSION['RF_Username'])<6 ||
strlen($_SESSION['RF_Username'])>25){
            $_SESSION['Error_code']="RF4";
            $_SESSION['Error_msg']="Registration failed! Username
must be 6 - 25 characters. (Error: RF4)";
            header('location:'.$goPage);
            die();
      }

      // Check password length
```

```
      if (strlen($_SESSION['RF_Password'])<6 ||
strlen($_SESSION['RF_Password'])>25){
            $_SESSION['Error_code']="RF5";
            $_SESSION['Error_msg']="Registration failed! Password
must be 6 - 25 characters. (Error: RF5)";
            header('location:'.$goPage);
            die();
      }

      // Check name length
      if (strlen($_SESSION['RF_Fname'])<2 ||
strlen($_SESSION['RF_Fname'])>25){
            $_SESSION['Error_code']="RF6";
            $_SESSION['Error_msg']="Registration failed! Name must
be 2 - 25 characters. (Error: RF6)";
            header('location:'.$goPage);
            die();
      }

      // Check surname length
      if (strlen($_SESSION['RF_Lname'])<2 ||
strlen($_SESSION['RF_Lname'])>25){
            $_SESSION['Error_code']="RF7";
            $_SESSION['Error_msg']="Registration failed! Surname
must be 2 - 25 characters. (Error: RF7)";
            header('location:'.$goPage);
            die();
      }

      // Check email length
      if (strlen($_SESSION['RF_Email'])<5 ||
strlen($_SESSION['RF_Email'])>50){
            $_SESSION['Error_code']="RF8";
            $_SESSION['Error_msg']="Registration failed! Email must
be 5 - 50 characters. (Error: RF8)";
            header('location:'.$goPage);
            die();
      }

      // Hash and Salt Password
      $tsalt = time();
      $hashedPass = sha1( $Password . $tsalt );

      $pass = dbSafe($hashedPass);
      $salt = dbSafe($tsalt);

      // Set created and logged_in datetime
      $Created = dbSafe(date('Y-m-d H:i:s'));
      $Logged_in = dbSafe(date('Y-m-d H:i:s'));


      // Check Database Connection
      if (!$conn) {
            $_SESSION['Error_code']="DB1";
            $_SESSION['Error_msg']="Connection with the database
failed. (Error: DB1)";
            header('location:'.$goPage);
```

```
            die();
      }

      // Insert to database
      $sql='INSERT INTO ACCOUNTS (USERNAME, PASSWORD, SALT, ROLE,
FNAME, LNAME, GENDER, EMAIL, CREATED, LOGGED_IN) VALUES

      ('.$Username.','.$pass.','.$salt.','.$Role.','.$Fname.','.$Ln
ame.','.$Gender.','.$Email.','.$Created.','.$Logged_in.')';

      if (!mysqli_query($conn,$sql)){
            $_SESSION['Error_code']="RF9";
            $_SESSION['Error_msg']="Registration failed! Database
error. (Error: RF9)";
            header('location:'.$goPage);
            die();
      }
      else{
            // Clear Registration Failure's Variables from Session
            unset($_SESSION['RF_Username']);
            unset($_SESSION['RF_Password']);
            unset($_SESSION['RF_Password2']);
            unset($_SESSION['RF_Fname']);
            unset($_SESSION['RF_Lname']);
            unset($_SESSION['RF_Gender']);
            unset($_SESSION['RF_Email']);
            unset($_SESSION['RF_Terms']);
            mysqli_close($conn);
            header('location:success.php?go='.$goPage);
      }
?>
```

_login.php:

```
<?php
      include('_common.php');
      include('_conn.php');

      $Username = dbSafe($_POST['username']);
      $Password=$_POST['password'];
      if (isset($_POST['remember']) && $_POST['remember'] == 1){
            $Remember=1;
      }
      $goPage = $_POST['current_page'];

      // Check Database Connection
      if (!$conn) {
            $_SESSION['Error_code']="DB1";
            $_SESSION['Error_msg']="Connection with the database
failed. (Error: DB1)";
            header('location:'.$goPage);
            die();
      }

      // Run db query
      $sql='SELECT * FROM ACCOUNTS WHERE USERNAME='.$Username;
```

```
      $result=mysqli_query($conn, $sql);
      $count=mysqli_num_rows($result);
      $row=mysqli_fetch_assoc($result);

      ///// Check Credentials
      // Check Username
      if ($count == 1){

            // Check if account is Blocked
            if ($row['STATUS']=="Blocked"){
                  mysqli_close($conn);
                  $_SESSION['Error_code']="AF1";
                  $_SESSION['Error_msg']='Your account has been
blocked. Please contact the website\'s administrator for further
actions.';
                  header('location:'.$goPage);
                  die();
            }

            // Check if account is Deactivated
            if ($row['STATUS']=="Deactivated"){
                  mysqli_close($conn);
                  $_SESSION['Error_code']="AF1";
                  $_SESSION['Error_msg']='Your access has been
revoked. Please contact the website\'s administrator for further
actions.';
                  header('location:'.$goPage);
                  die();
            }

            // Check if login was attempted again today
            // If not, set login_attempts to zero and
login_attempts_date to today
            $last_login_attempt_date = date("Y-m-d",
strtotime($row['LOGIN_ATTEMPTS_DATE']));
            $today = date("Y-m-d H:i:s");
            $today_date = date('Y-m-d', $today);
            if ($last_login_attempt_date==$today_date){
                  $sql = "UPDATE ACCOUNTS SET LOGIN_ATTEMPTS='0',
LOGIN_ATTEMPTS_DATE='$today' WHERE AID='".$row['AID']."' ";
                  mysqli_query($conn, $sql);
            }

            $storedPassword = $row['PASSWORD'];
            $salt = $row['SALT'];
            // use the stored salt to hash the user's submitted
password
            $hashedPassword = sha1($Password.$salt);

            // Check Password
            if ( $storedPassword == $hashedPassword ) {

                  $sql = "UPDATE ACCOUNTS SET LOGIN_ATTEMPTS='0',
LOGIN_ATTEMPTS_DATE='$today', LOGGED_IN='$today' WHERE
AID='".$row['AID']."' ";
                  mysqli_query($conn, $sql);
```

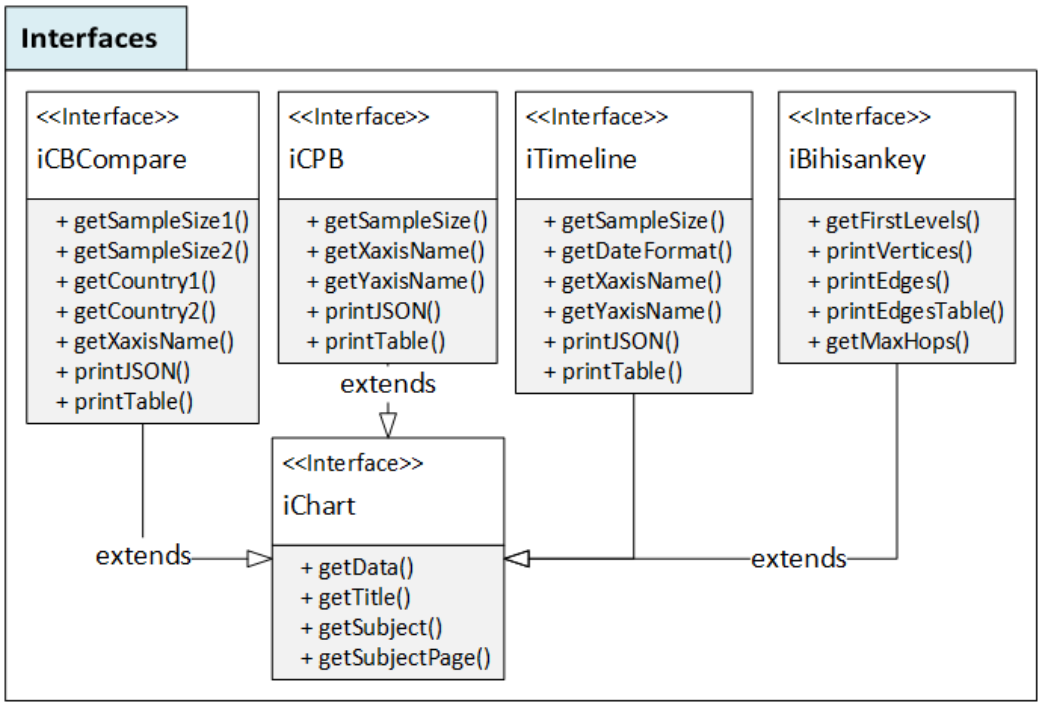
```
                  $_SESSION['AID']=$row['AID'];
                  $_SESSION['USERNAME']=$row['USERNAME'];
                  $_SESSION['FNAME']=$row['FNAME'];
                  $_SESSION['LNAME']=$row['LNAME'];
                  $_SESSION['GENDER']=$row['GENDER'];
                  $_SESSION['EMAIL']=$row['EMAIL'];

                  // Check Role
                  if ($row['ROLE'] == 1) $_SESSION['ROLE']=1; //
normal user
                  else if ($row['ROLE'] == 2) $_SESSION['ROLE']= 2;
// admin
                  header('location:'.$goPage);
                  mysqli_close($conn);
            }else{
                  // wrong password
                  $login_attempts = $row['LOGIN_ATTEMPTS'];
                  if ($login_attempts<=1){
                        $login_attempts++;
                        $remaining = 3 - $login_attempts;
                        $sql = "UPDATE ACCOUNTS SET
LOGIN_ATTEMPTS='$login_attempts' WHERE AID='".$row['AID']."' ";
                        mysqli_query($conn, $sql);
                        $_SESSION['Error_code']="AF2";
                        $_SESSION['Error_msg']="Authentication
failed! Password does not match (Error: AF2). <b>$remaining</b>
remaining attempt(s).";
                  }else if($login_attempts==2){
                        $login_attempts++;
                        $sql = "UPDATE ACCOUNTS SET
LOGIN_ATTEMPTS='$login_attempts', STATUS='Blocked' WHERE
AID='".$row['AID']."' ";
                        mysqli_query($conn, $sql);
                        $_SESSION['Error_code']="AF2";
                        $_SESSION['Error_msg']='Authentication
failed! Password does not match (Error: AF2).<br>Your account has
been locked. Please contact the website\'s administrator for
further actions.';
                  }
                  header('location:'.$goPage);
                  mysqli_close($conn);
            }
      }else{
            // Username does not exist
            $_SESSION['Error_code']="AF1";
            $_SESSION['Error_msg']='Authentication failed! Username
does not exist. (Error: AF1)';
            header('location:'.$goPage);
            mysqli_close($conn);
      }
?>
```

Η διαδικασία του logout γίνεται πολύ εύκολα με την καταστροφή του PHP Session. Στην συνέχεια, γίνεται ανακατεύθυνση του χρήστη στην Οικοσελίδα.

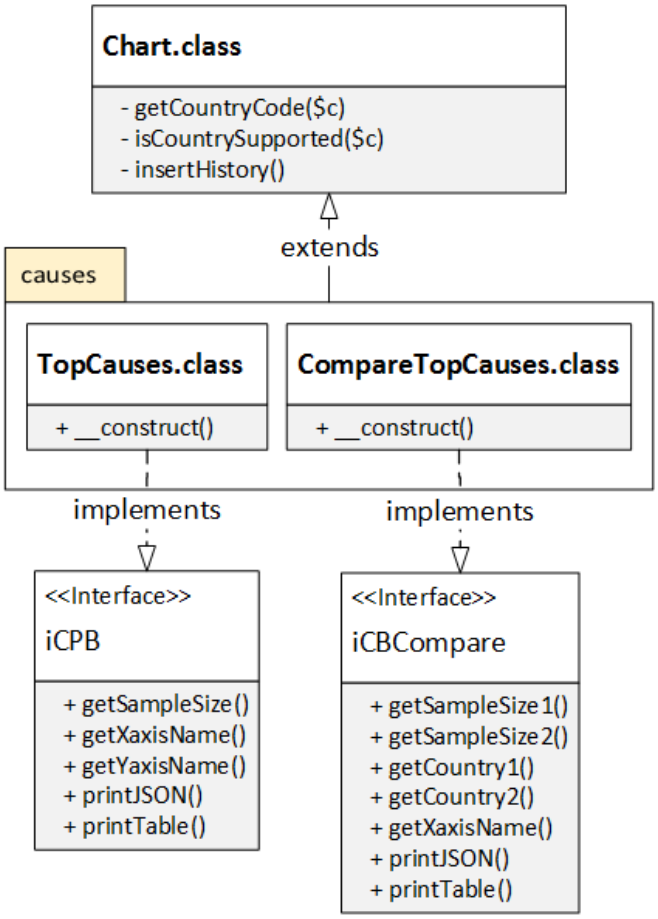
```
<?php
      session_start();
      session_destroy();
      header('location:index.php');
?>
```

Πιο κάτω γίνεται παρουσίαση σε μορφή UML της σχέσης των κλάσεων και των interfaces που έχουν χρησιμοποιηθεί για την υλοποίηση των γραφικών παραστάσεων για την ανάλυση του LinkedIn. Όλα τα interfaces γίνονται extend σε ένα πατρικό interface, το iChart.php (Βλ. Σχήμα 4.3), ενώ όλες οι κλάσεις γίνονται extend σε μια πατρική κλάση, την Chart.class.php. Η κάθε κλάση PHP κάνει implement το ανάλογο interface για τον visualizer στον οποίο συμπεριλαμβάνεται.



Σχήμα 4.3: UML Interfaces.

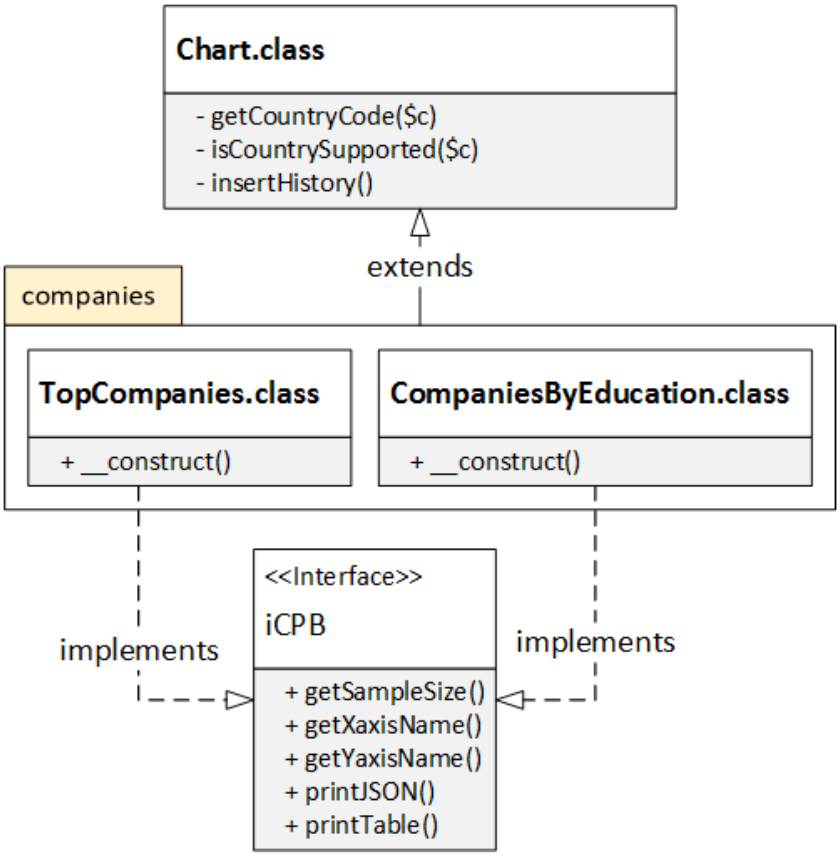
Κλάσεις κατηγορίας causes:



Σχήμα 4.4: UML causes.

- TopCauses.class implements iCPB interface extends to Chart.class.
- CompareTopCauses.class implements iCBCompare interface extends to Chart.class.

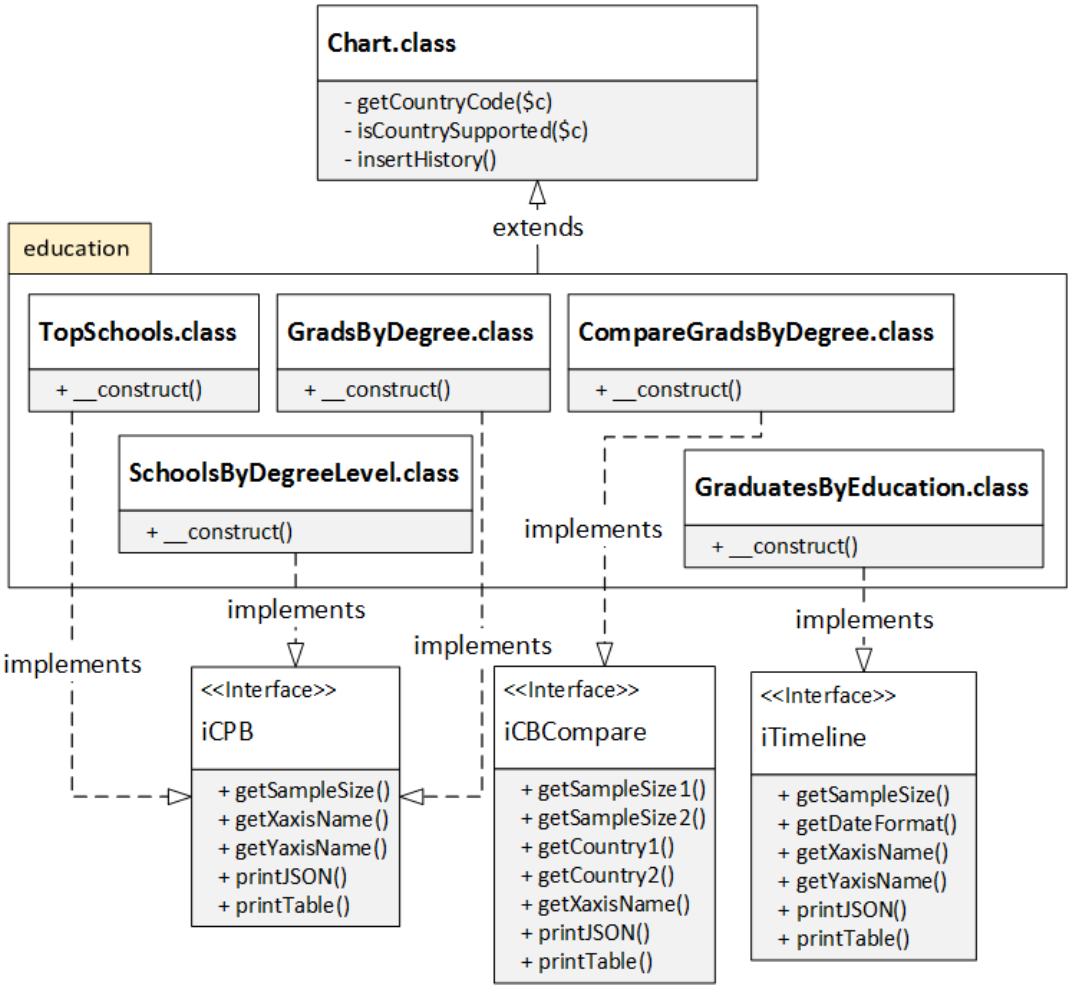
Κλάσεις Κατηγορίας companies:



Σχήμα 4.5: UML companies.

- TopCompanies.class implements iCPB extends Chart.class.
- CompaniesByEducation.class implements iCPB extends Chart.class.

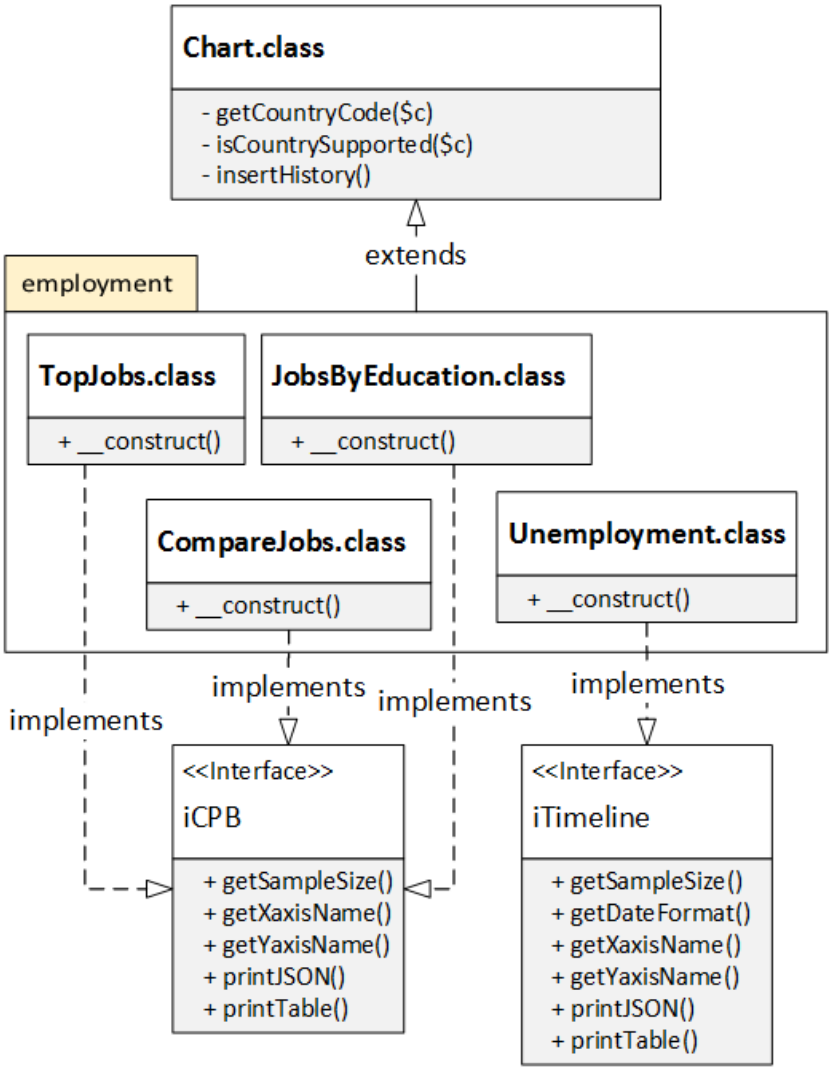
Κλάσεις Κατηγορίας education:



Σχήμα 4.6: UML education.

- TopSchools.class implements iCPB extends Chart.class.
- SchoolsByDegreeLevel.class implements iCPB extends Chart.class.
- GradsByDegree.class implements iCPB extends Chart.class.
- GraduatesByEducation.class implements iTimeline extends Chart.class.
- CompareGradsByDegree.class implements iCBCompare extends Chart.class.

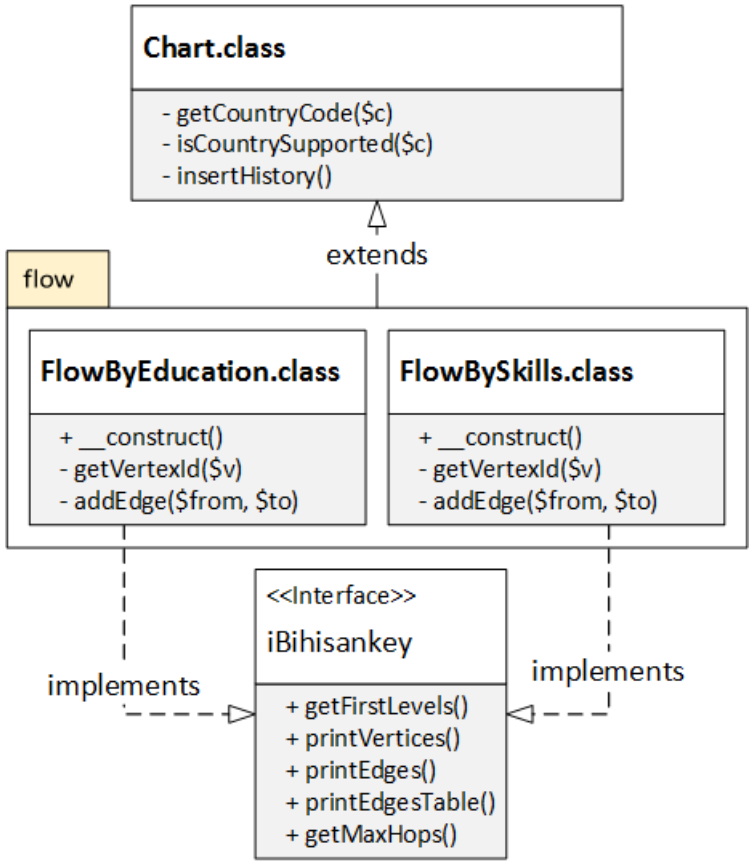
Κλάσεις Κατηγορίας employment:



Σχήμα 4.7: UML employment.

- TopJobs.class implements iCPB extends Chart.class.
- JobsByEducation.class implements iCPB extends Chart.class.
- CompareJobs.class implements iCPB extends Chart.class.
- GraduatesByEducation.class implements iTimeline extends Chart.class.

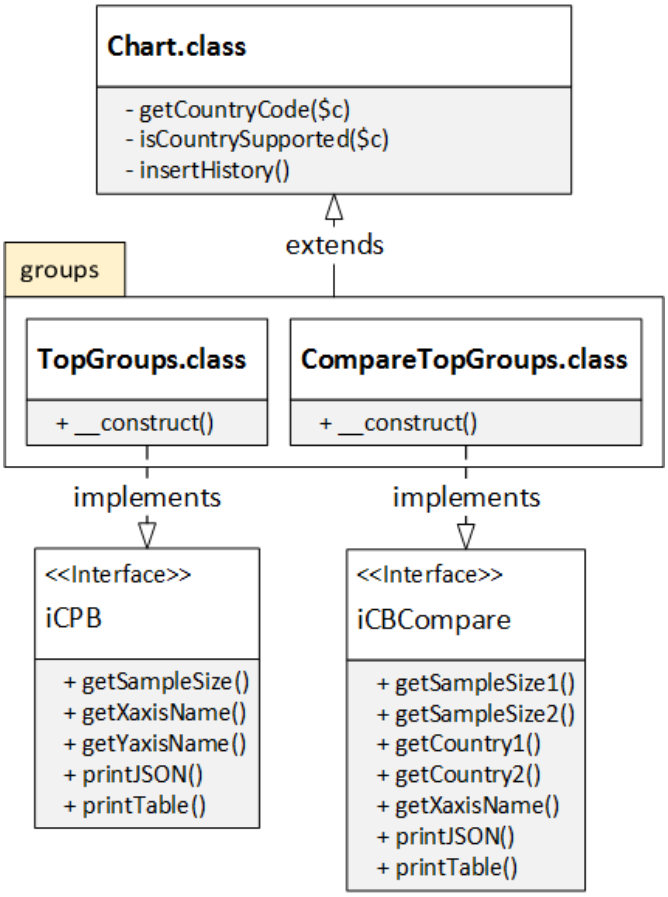
Κλάσεις Κατηγορίας flow:



Σχήμα 4.8: UML flow.

- FlowByEducation.class implements iBihisankey extends Chart.class.
- FlowBySkills.class implements iBihisankey extends Chart.class.

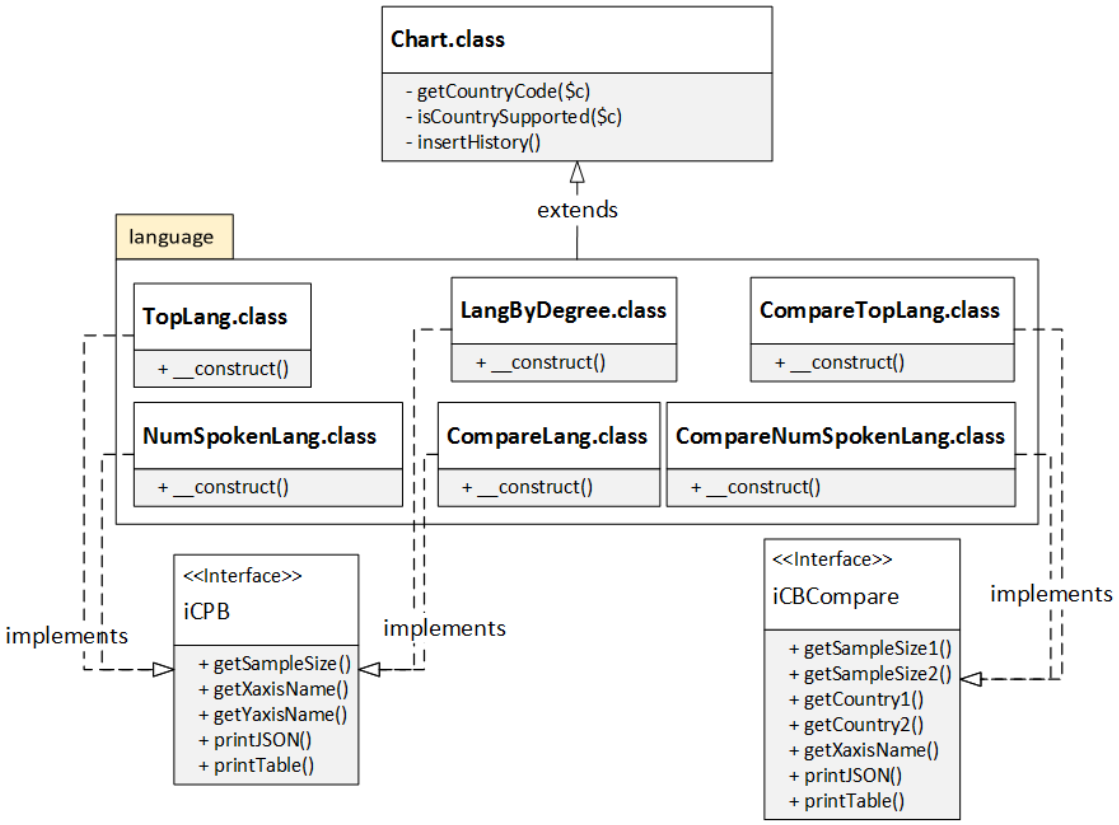
Κλάσεις κατηγορίας groups:



Σχήμα 4.9: UML groups.

- TopGroups.class implements iCPB interface extends to Chart.class.
- CompareTopGroups.class implements iCBCompare interface extends to Chart.class.

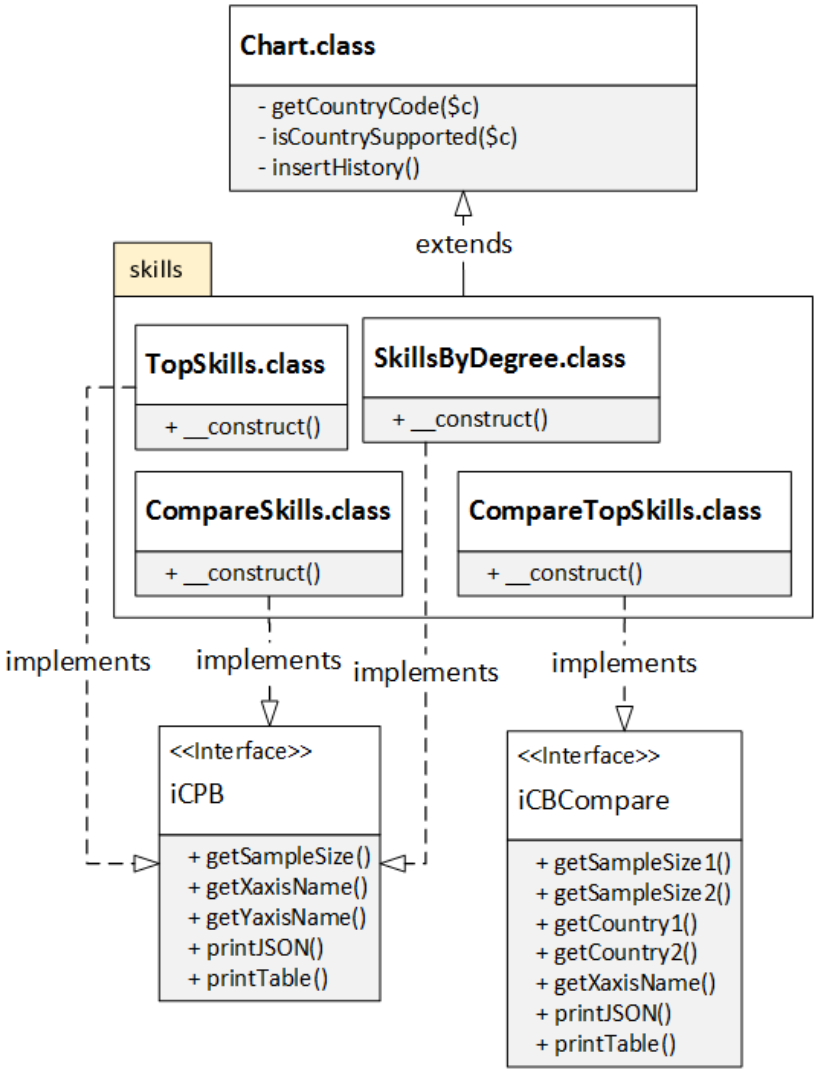
Κλάσεις Κατηγορίας language:



Σχήμα 4.10: UML language.

- TopLang.class implements iCPB extends Chart.class.
- NumSpokenLang.class implements iCPB extends Chart.class.
- LangByDegree.class implements iCPB extends Chart.class.
- CompareLang.class implements iCPB extends Chart.class.
- CompareTopLang.class implements iCBCompare extends Chart.class.
- CompareNumSpokenLang.class implements iCBCompare extends Chart.class.

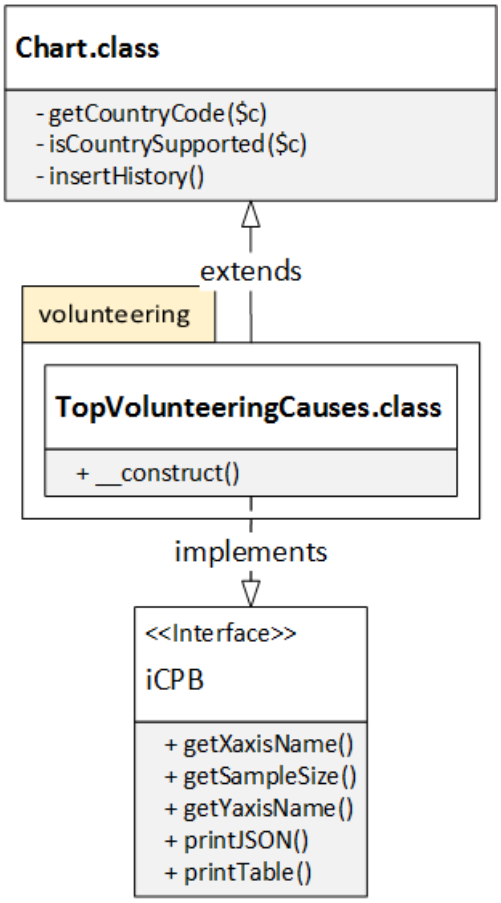
Κλάσεις Κατηγορίας skills:



Σχήμα 4.11: UML skills.

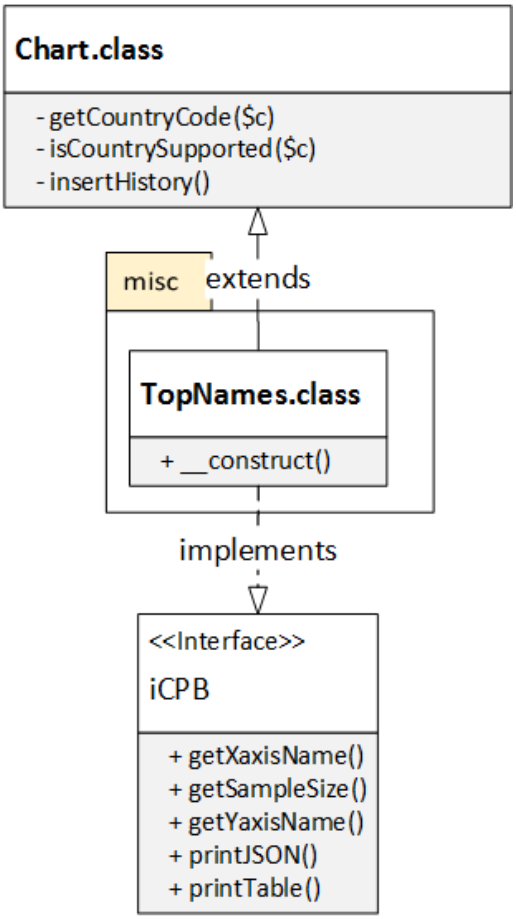
- TopSkills.class implements iCPB extends Chart.class.
- SkillsByDegree.class implements iCPB extends Chart.class.
- CompareSkills.class implements iCPB extends Chart.class.
- CompareTopSkills.class implements iCBCompare extends Chart.class.

Κλάσεις Κατηγορίας volunteering:



Σχήμα 4.12: UML volunteering.

Κλάσεις Κατηγορίας misc:



Σχήμα 4.13: UML misc.

- TopVolunteeringCauses.class implements iCPB extends Chart.class.
- TopNames.class implements iCPB extends Chart.class.

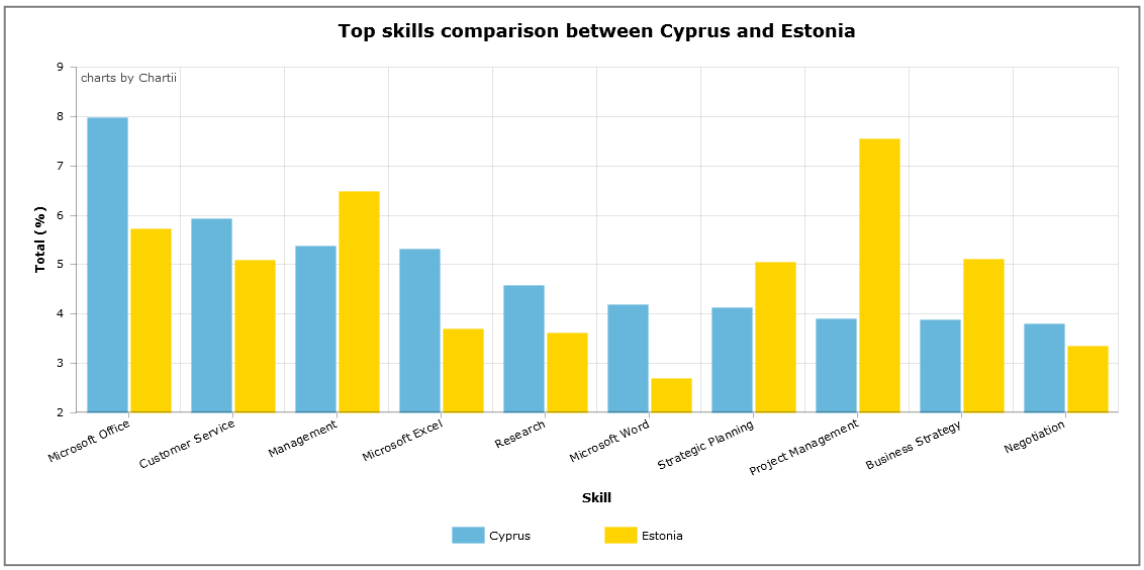
Κεφάλαιο 5

Συμπεράσματα

5.1. Συμπεράσματα Ανάλυσης του LinkedIn

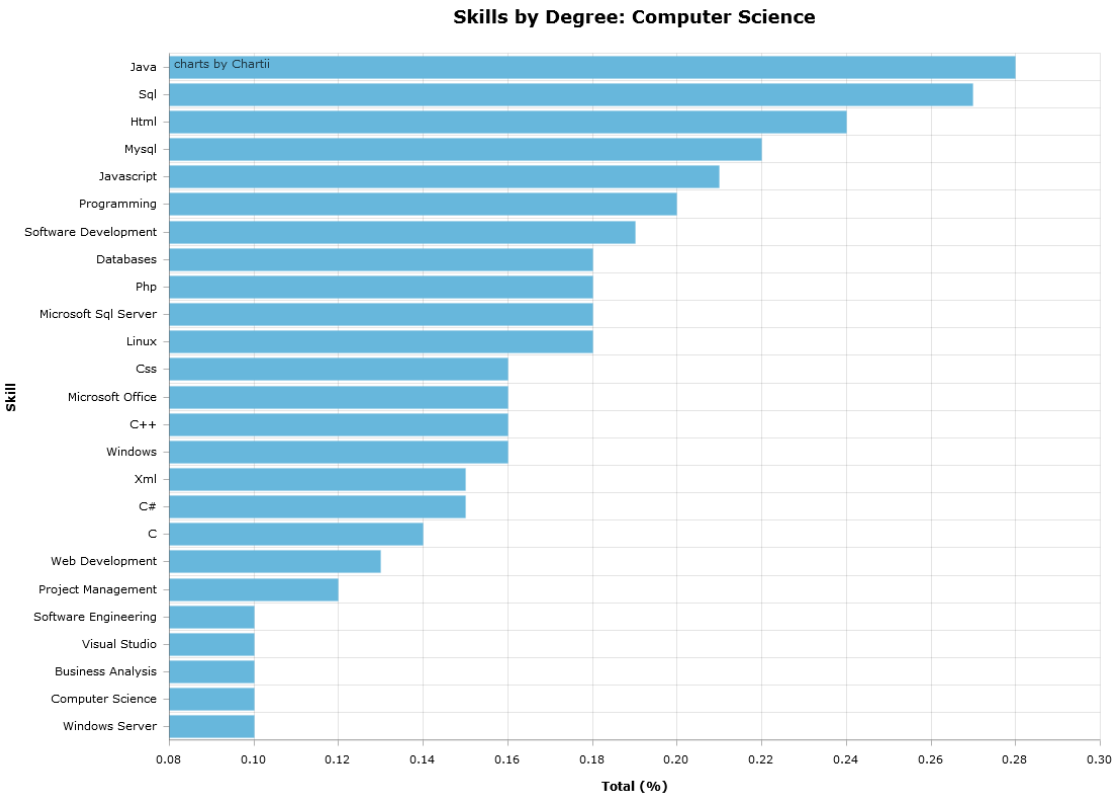
Όπως έχει αναφερθεί και στο υποκεφάλαιο 2.2 Συλλογή Δεδομένων, λόγο μεγάλου αριθμού χρηστών που έχει κάθε χώρα, έχουν επιλεγεί δύο μικρές χώρες για σύγκριση, η Κύπρος και η Εσθονία. Από το crawling της Κύπρου έχει μαζευτεί ένα δείγμα **140,017** χρηστών, ενώ από της Εσθονίας ένα δείγμα **125,253** χρηστών. Πιο κάτω ακολουθούν μερικά από τα συμπεράσματα των αποτελεσμάτων σύγκρισης των δύο χωρών ανά κατηγορία αλλά και συμπεράσματα για την κάθε μια ξεχωριστά. Όλες οι τιμές είναι σε συνάρτηση του μεγέθους δείγματος της κάθε χώρας.

Δεξιότητες:



Σχήμα 5.1: Σύγκριση Δεξιοτήτων μεταξύ Κύπρου και Εσθονίας.

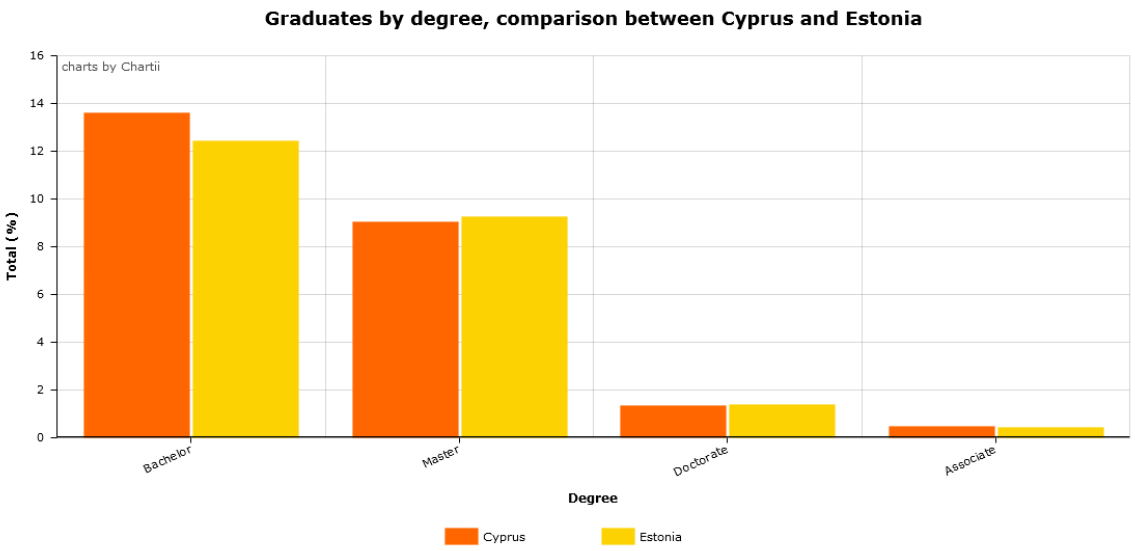
Από την σύγκριση των δύο χωρών στις πρώτες 10 δεξιότητές τους παρατηρούμε περισσότερο δεξιότητες επιχειρηματικά προσανατολισμένες καθώς και κοινή γνώση διαφόρων εργαλείων που βοηθούν στην παραγωγικότητα γραφειακής δουλειάς.



Σχήμα 5.2: Σύγκριση Δεξιοτήτων Βάση Πτυχίου.

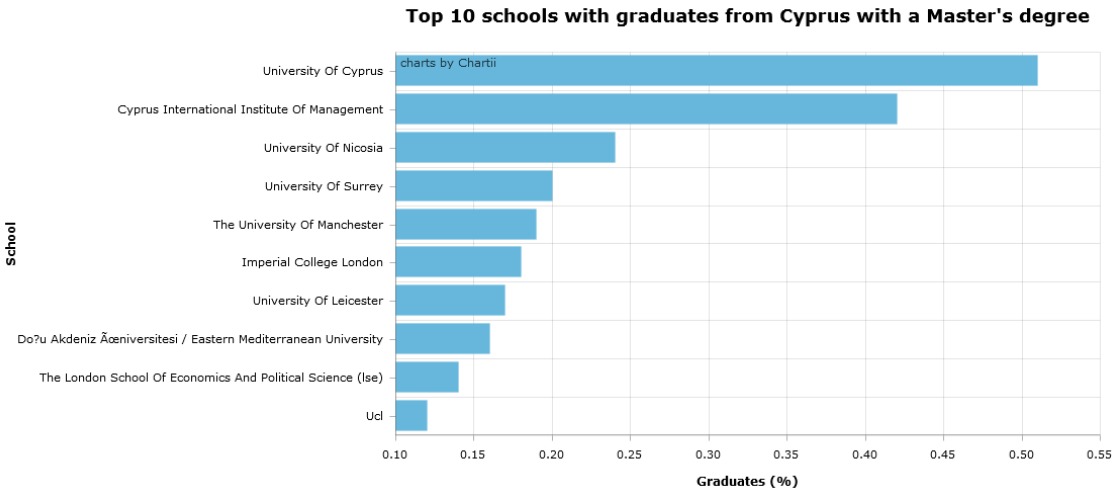
Το σύστημα παρέχει επίσης πρόσβαση στις δεξιότητες βάση ορισμένου πτυχίου. Τιμής ένεκεν, έχει επιλεγεί το πτυχίο «Επιστήμη Πληροφορικής» και όπως παρατηρούμε στην πρώτη θέση βρίσκεται η γλώσσα προγραμματισμού Java ακολουθούμενη από γλώσσες διαχείρισης βάσεων δεδομένων, διαδικτυακού προγραμματισμού και άλλες.

Μόρφωση:



Σχήμα 5.3: Σύγκριση βάση επιπέδων πτυχίου μεταξύ Κύπρου και Εσθονίας.

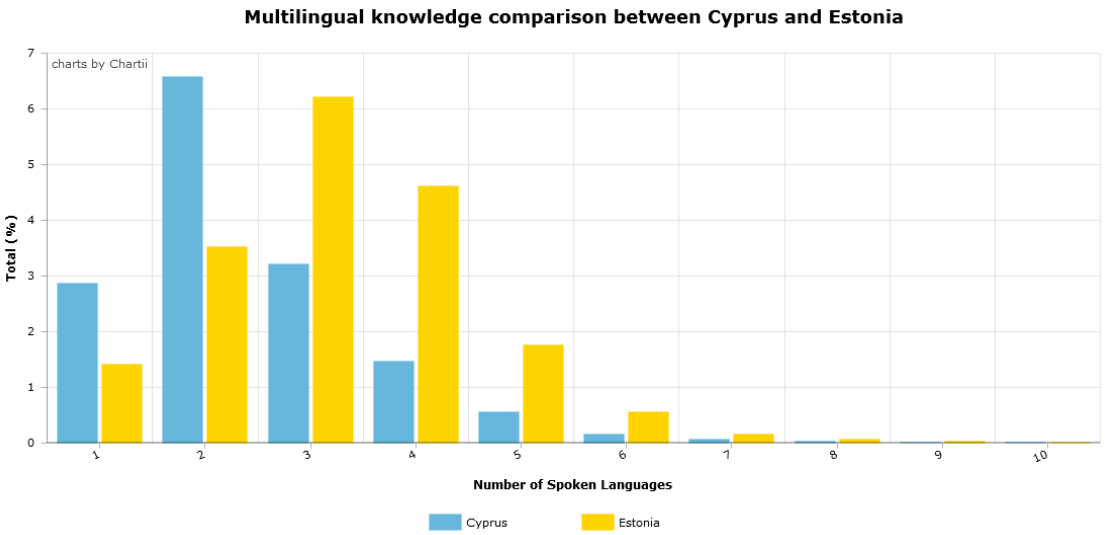
Αρκετά ενδιαφέρον αποτελεί και η σύγκριση των δύο χωρών βάση των αποφοίτων στα διάφορα επίπεδα πτυχίου. Παρατηρούμε ότι σε συνάρτηση του δείγματος, η Κύπρος διαθέτει ελαφρώς περισσότερα άτομα με πτυχίο. Αξιοσημείωτο είναι επίσης και το γεγονός ότι καθώς το επίπεδο αυξάνεται, όλο και λιγότερα άτομα προχωρούν στο επόμενο στάδιο.



Σχήμα 5.4: Δέκα Πρώτα Σχολεία με κύπριους απόφοιτους με master.

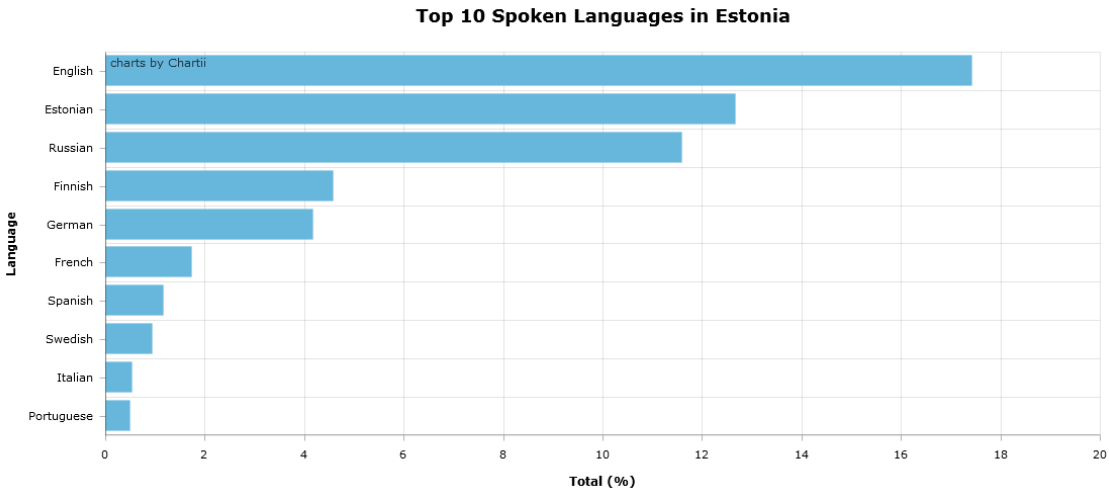
Μελετώντας σε πια πανεπιστήμια/κολλέγια επιλέγουν οι κύπριοι για σπουδές master, παρατηρούμε ότι οι περισσότεροι επιλέγουν κυπριακά πανεπιστήμια με πρώτο το Πανεπιστήμιο Κύπρου, ενώ οι υπόλοιποι επιλέγουν σπουδές στην Αγγλία.

Γνώση Γλωσσών:



Σχήμα 5.5: Σύγκριση γνώσης πολλαπλών γλωσσών μεταξύ Κύπρου και Εσθονίας.

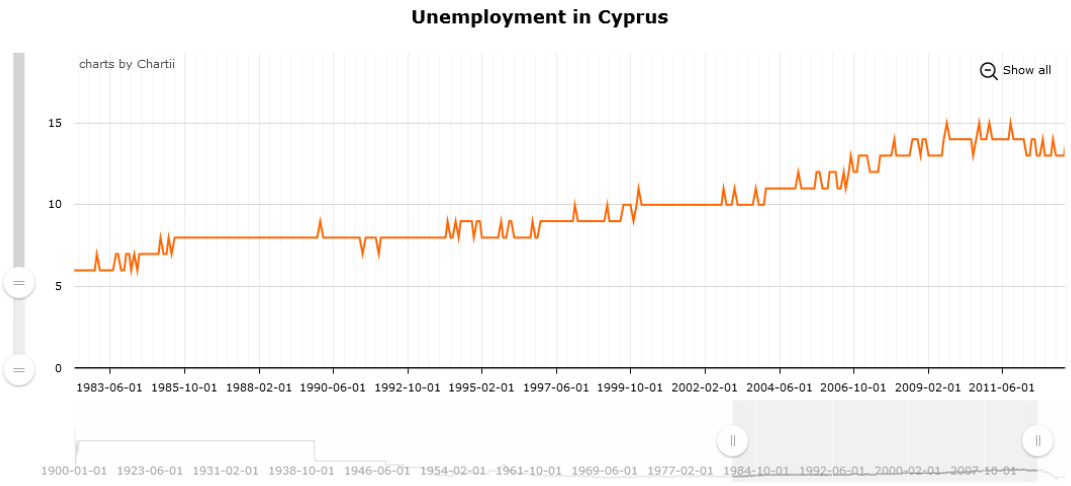
Ένα ιδιαίτερα ενδιαφέρον συμπέρασμα προέκυψε από την ανάλυση της σύγκρισης των δύο αυτών χωρών στην γνώση πολλαπλών γλωσσών. Παρατηρούμε ότι οι Εσθονοί υπερτερούν των Κυπρίων στην γνώση περισσοτέρων από δύο γλωσσών.



Σχήμα 5.6: Δέκα πρώτες ομιλούμενες γλώσσες στην Εσθονία.

Σε συνέχεια της προηγούμενης γραφικής παράστασης, μπορούμε μέσω του συστήματος να δούμε ποιες είναι οι πιο ομιλούμενες γλώσσες στην Εσθονία. Μέσω της ανάλυσης συμπεραίνουμε ότι η πλειονότητα των Εσθονών μιλάνε Αγγλικά, Εσθονικά και Ρωσικά.

Εργασία:

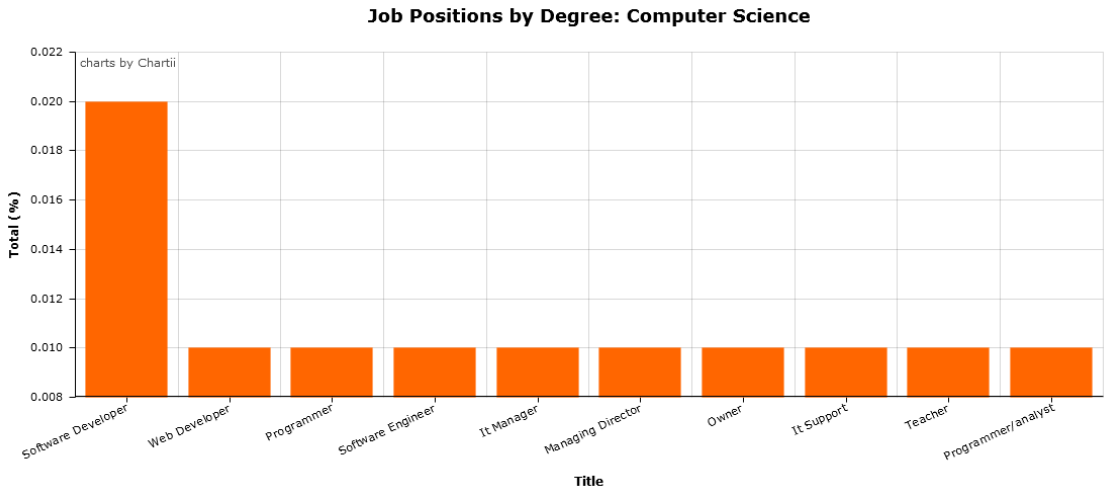


Σχήμα 5.7: Ανεργία στην Κύπρο.

Μια άλλη σημαντική στατιστική που μπορούμε να εξάγουμε από την ανάλυση του LinkedIn είναι η ανεργία στην Κύπρο. Μιας και ο κάθε χρήστης του LinkedIn μπορεί ανεβάσει πληροφορίες για την θέση εργασίας του, μπορούμε να χαρτογραφήσουμε αυτή την πορεία και να υπολογίσουμε το ποσοστό ανεργίας στην κάθε χώρα. Η φόρμουλα υπολογισμού της ανεργίας ορίζεται ως ο αριθμός των ανέργων δια το άθροισμα τον εργαζομένων και ανέργων, δηλαδή δια του εργατικού δυναμικού:

$$Ποσοστό\ Ανεργίας = \frac{Αρ.Ανέργων}{Αρ.Εργαζομένων + Αρ.Ανέργων}$$

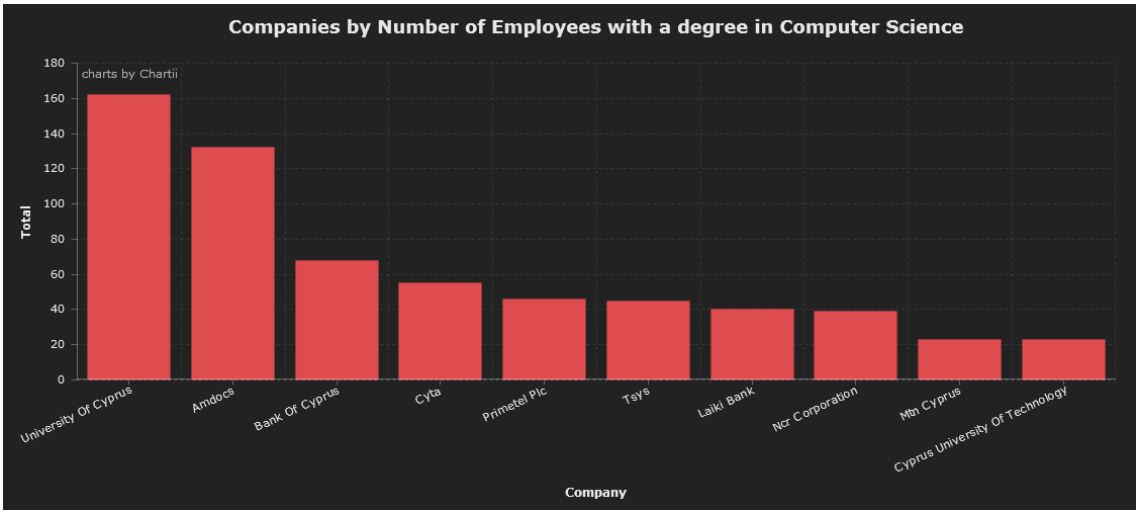
Όπως αναμένεται, τα τελευταία χρόνια παρατηρείτε αύξηση της ανεργίας με το 2011 να φτάνουμε ποσοστά 15%.



Σχήμα 5.8: Θέσεις Εργασίας Βάση Πτυχίου.

Μέσα από την ανάλυση του LinkedIn μπορούμε, εκτός άλλων, να δούμε επίσης τις πρώτες θέσεις εργασίας βάση μόρφωσης καθώς επίσης τις τελευταίες αλλά και όλες διαχρονικά. Στην πιο πάνω περίπτωση βλέπουμε την πρώτη θέση εργασία των Κυπρίων με πτυχίο στην Επιστήμη Πληροφορικής. Παρατηρούμε ότι η πλειονότητα βρήκε σαν πρώτη θέση εργασίας σαν προγραμματιστές λογισμικού.

Εταιρείες:

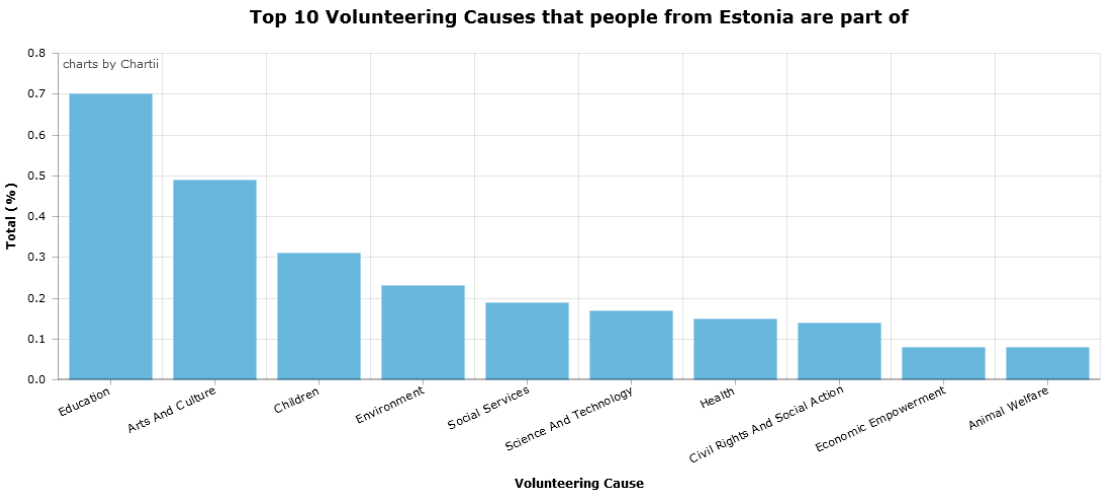


Σχήμα 5.8: Εταιρείες με εργαζόμενους με πτυχίο «Επιστήμη Πληροφορικής».

Μια άλλη γραφική παράσταση που προσφέρει το σύστημα είναι λίστα με εταιρείες στις οποίες εργάζονται άτομα με συγκεκριμένη μόρφωση. Το Σχήμα 5.8 παρουσιάζει τις

δέκα πρώτες εταιρείες που εργάζονται οι πτυχιούχοι Πληροφορικής, η πλειονότητα των οποίων εργάζονται στο Πανεπιστήμιο Κύπρου και στην εταιρεία Amdocs.

Εθελοντισμός:



Σχήμα 5.9: Εθελοντισμός στην Εσθονία.

Αναλύοντας το LinkedIn μπορούμε να βρούμε πληροφορίες για το εθελοντικό ενδιαφέρον κάποιας χώρας. Το Σχήμα 5.9 παρουσιάζει τις δέκα πρώτες κατηγορίες εθελοντισμού για τις οποίες ενδιαφέρονται οι Εσθονοί. Όπως φαίνεται και από την γραφική παράσταση, υπάρχει ιδιαίτερο ενδιαφέρον στην εκπαίδευση, στην τέχνη και στον πολιτισμό.

Τα πιο πάνω συμπεράσματα ήταν απλά ένα δείγμα του τι μπορεί να εξαχθεί από το συγκεκριμένο σύστημα ανάλυσης του κοινωνικού δικτύου LinkedIn έχοντας μόνο συλλέξει πληροφορίες από δύο χώρες. Λόγω της δυνατότητας εξατομικευμένων συγκρίσεων σε διάφορα θέματα, μπορούν να δημιουργηθούν πολυάριθμες γραφικές παραστάσεις όπου η κάθε μια να εξυπηρετεί τον κάθε χρήστη διαφορετικά, δίνοντας έτσι στο σύστημα απεριόριστες δυνατότητες.

Βιβλιογραφία

[1] L. Tyson and S. Lund, "Project Syndicate" Global Flows and Global Growth, 15 May 2014. [Online]. Available: https://www.project-syndicate.org/commentary/laura-tyson-and-susan-lund-show-how-countries-and-economic-sectors-benefit-from-greater-international-interconnectedness?barrier=true. [Accessed 14 September 2015].

[2] J. Manyika, J. Bughin, S. Lund, O. Nottebohm, D. Poulter, S. Jauch and S. Ramaswamy, "Global flows in a digital age: How trade, finance, people, and data connect the world economy", McKinsey & Company, New York, 2014.

[3] E. Commission, "European Commission" 07 May 2015. [Online]. Available: http://europa.eu/rapid/press-release_MEMO-15-4928_en.htm?locale=en. [Accessed 14 September 2015].

[4] "Tech Immigrants: A Map of Silicon Valley's Imported Talent" 6 June 2014. [Online]. Available: http://www.bloomberg.com/news/articles/2014-06-05/tech-immigrants-a-map-of-silicon-valleys-imported-talent. [Accessed 14 September 2015].

[5] "Business cluster", Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Business_cluster. [Accessed 15 September 2015].

[6] "What Are Clusters?", Harvard Business School, [Online]. Available: http://www.isc.hbs.edu/competitiveness-economic-development/frameworks-and-key-concepts/Pages/clusters.aspx. [Accessed 15 September 2015].

[7] "Cluster Observatory", [Online]. Available: http://www.clusterobservatory.eu/index.html. [Accessed 15 September 2015].

[8] "Java Programming Language", Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Java_%28programming_language%29. [Accessed 18 April 2016].

[9] "Web Crawler", Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Web_crawler. [Accessed 18 April 2016].

[10] "jsoup: Java HTML Parser", Open Source Project, [Online]. Available: https://jsoup.org/. [Accessed 18 April 2016].

[11] "MySQL", MySQL AB, [Online]. Available: https://www.mysql.com/. [Accessed 18 April 2016].

[12] "Dictionary Attack", Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Dictionary_attack. [Accessed 10 May 2016].

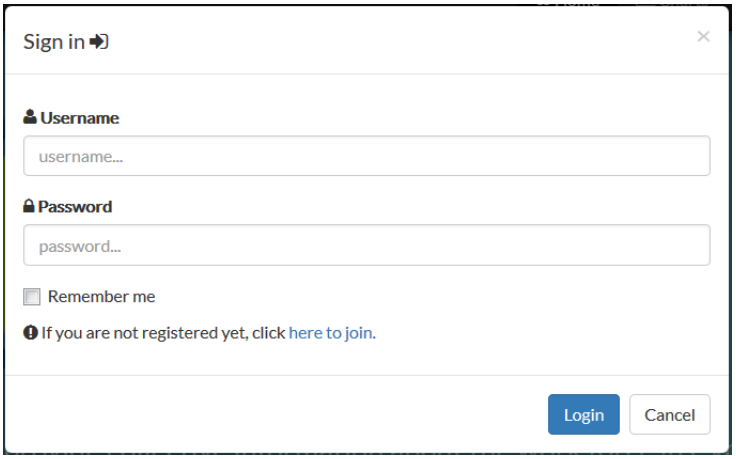
[13] "Salt (Cryptography)," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Salt_%28cryptography%29. [Accessed 11 May 2016].

[14] "Rainbow Table," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Rainbow_table. [Accessed 10 May 2016].

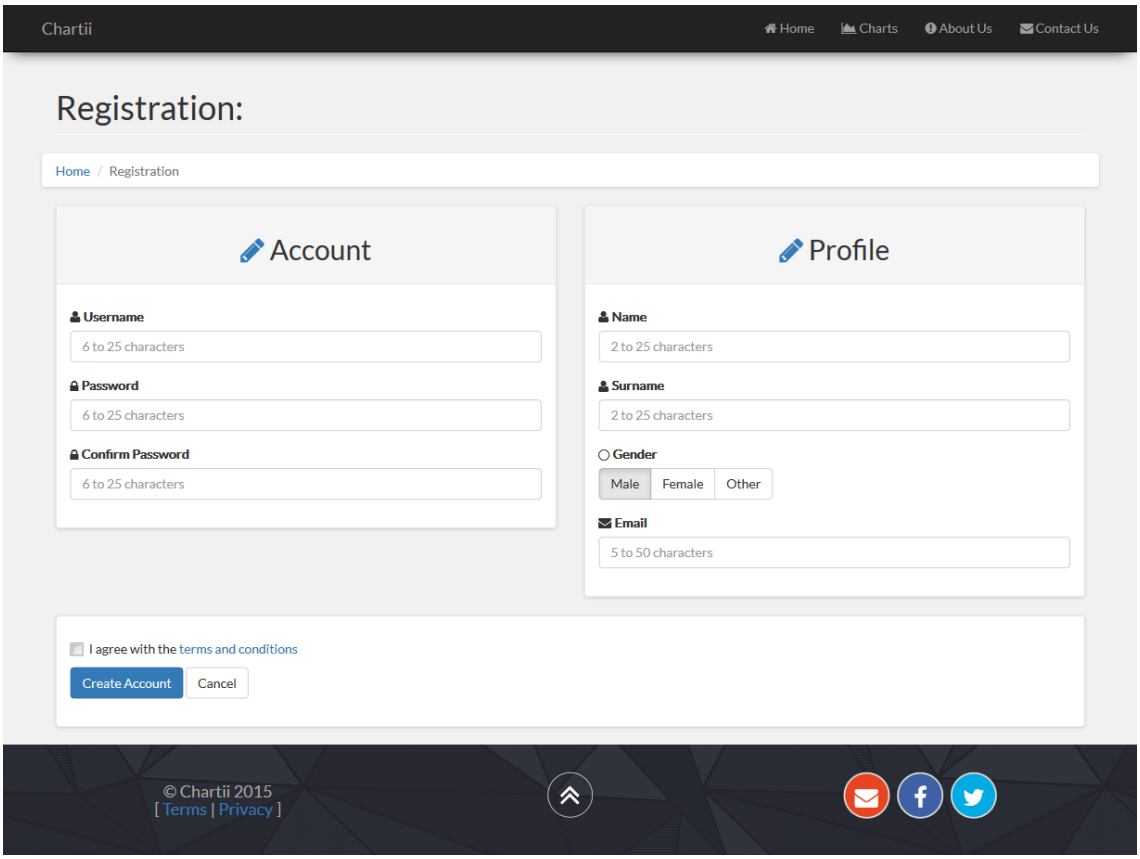
Παράρτημα Α

Το Παράρτημα Α περιλαμβάνει στιγμιότυπα οθονών του συστήματος:

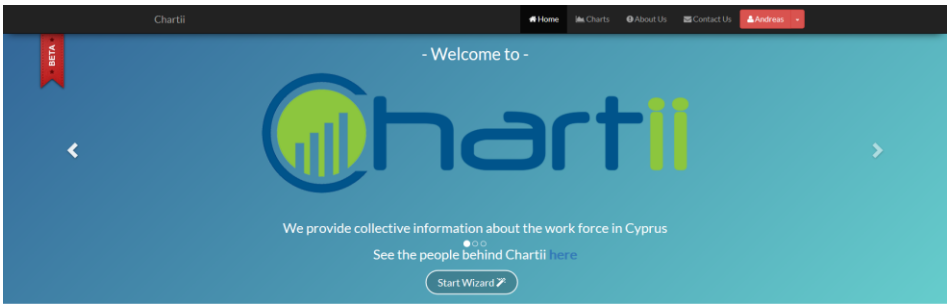
Login:

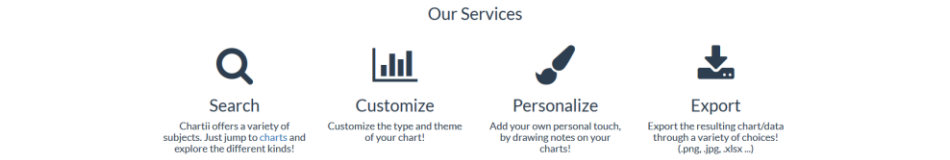


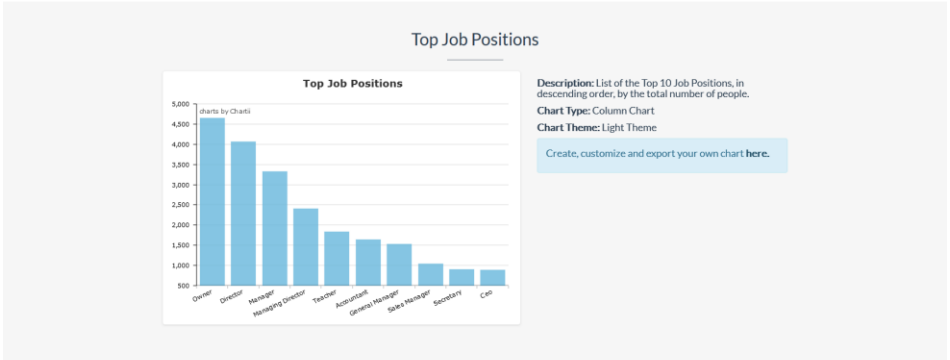
Register:



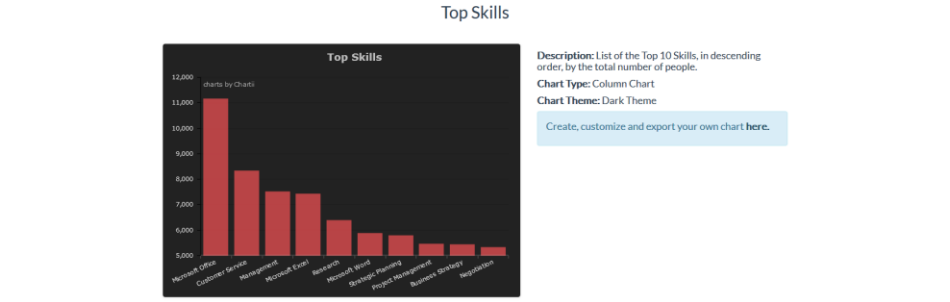
Index:

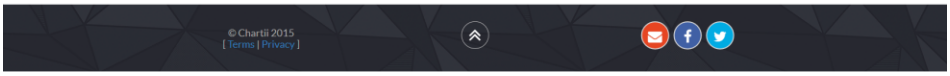

Chartii
Home
Charts
About Us
Contact Us
Andreas
BETA
- Welcome to -
We provide collective information about the work force in Cyprus
See the people behind Chartii here
Start Wizard


Our Services
Search
Chartii offers a variety of subjects. Just jump to charts and explore the different kinds!
Customize
Customize the type and theme of your chart!
Personalize
Add your own personal touch, by drawing notes on your charts!
Export
Export the resulting chart/data through a variety of choices! (.png, .jpg, .xlsx ...)

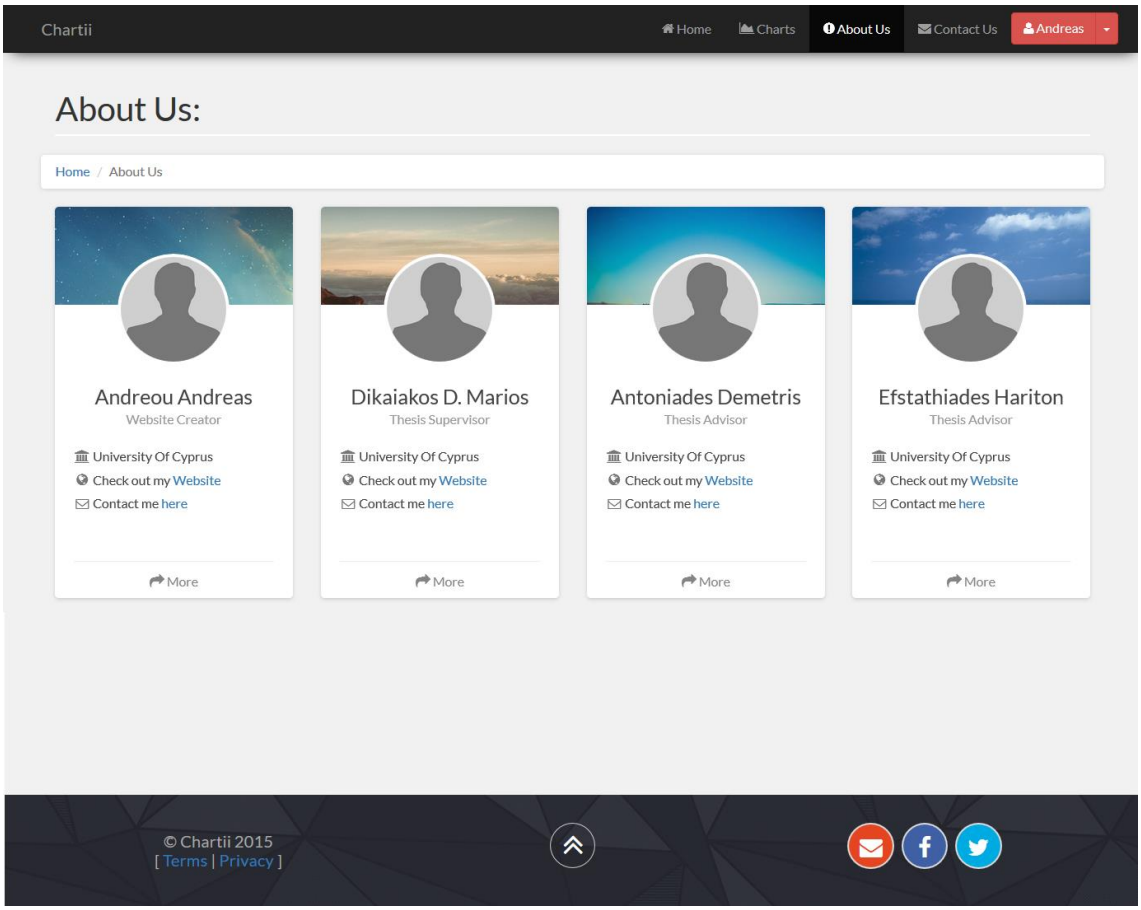

Top Job Positions
Top Job Positions
charts by Chartii
5,000
4,500
4,000
3,500
3,000
2,500
2,000
1,500
1,000
500
Owner
Director
Manager
Managing Director
Teacher
Accountant
General Manager
Sales Manager
Secretary
Ceo
Description: List of the Top 10 Job Positions, in descending order, by the total number of people.
Chart Type: Column Chart
Chart Theme: Light Theme
Create, customize and export your own chart here.


Top Spoken Languages
Description: Chart that shows the Top 6 Spoken Languages and the aggregate result of the rest shown as 'Others'.
Chart Type: Pie Chart
Chart Theme: Default Theme
Create, customize and export your own chart here.
charts by Chartii
Top Spoken Languages
Others: 16.03%
Spanish: 3.02%
German: 3.71%
Russian: 4.47%
French: 7.34%
Greek: 27.81%
English: 37.62%
English 19,298
Greek 14,268
French 3,768
Russian 2,292
German 1,903
Spanish 1,547
Others 8,225

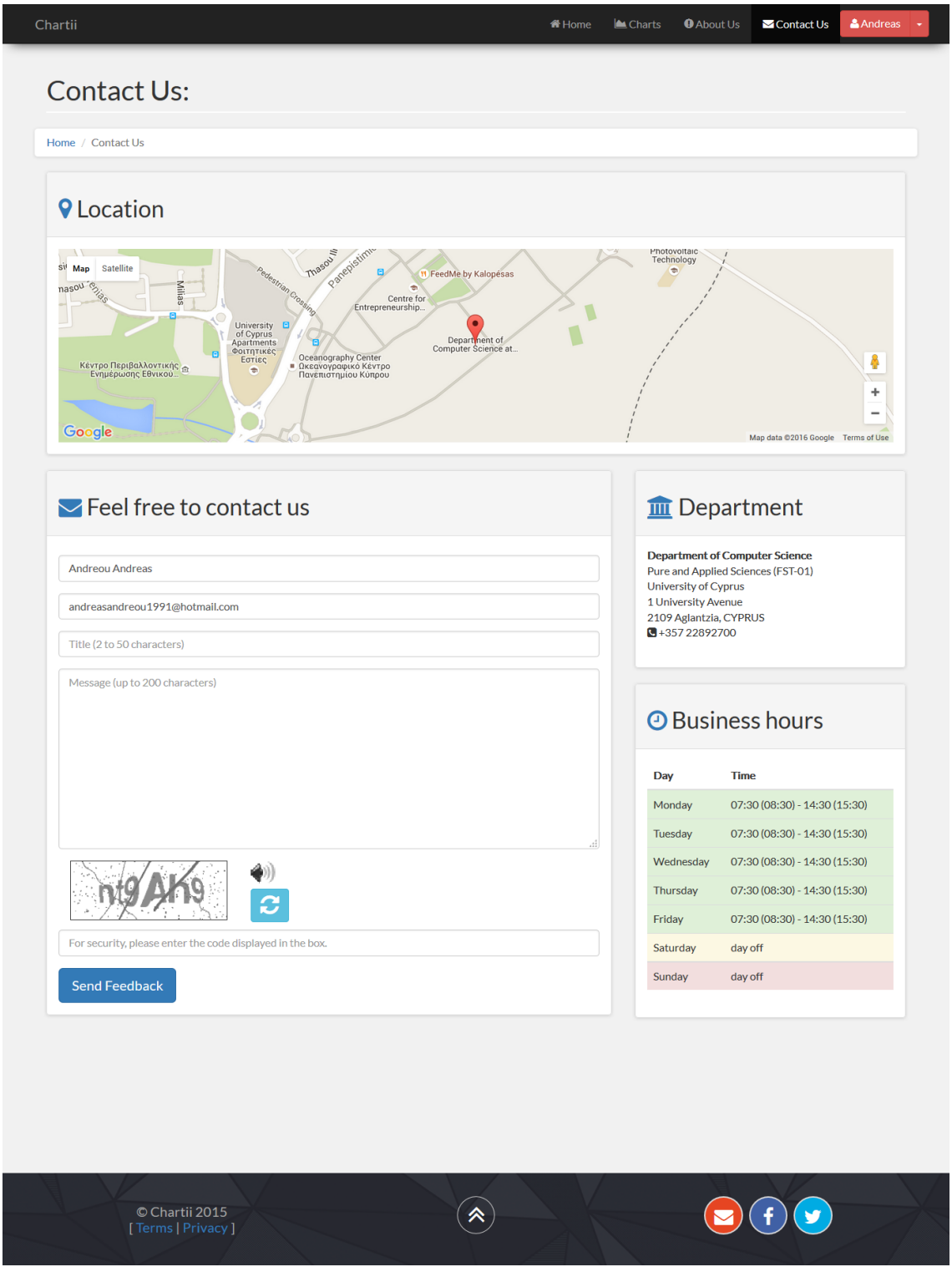

Top Skills
Top Skills
charts by Chartii
12,000
11,000
10,000
9,000
8,000
7,000
6,000
5,000
Microsoft Office
Customer Service
Management
Microsoft Excel
Research
Microsoft Word
Strategic Planning
Project Management
Business Strategy
Negotiation
Description: List of the Top 10 Skills, in descending order, by the total number of people.
Chart Type: Column Chart
Chart Theme: Dark Theme
Create, customize and export your own chart here.


© Chartii 2015
[Terms | Privacy]

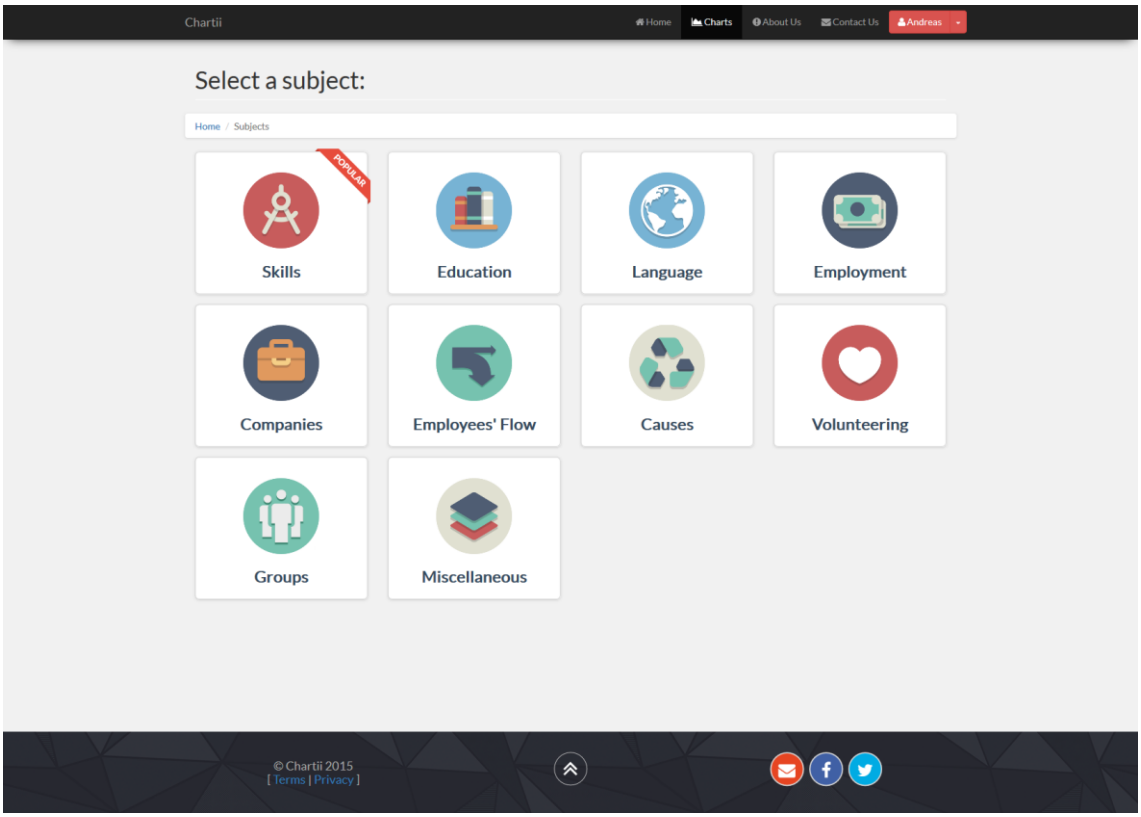
About Us:


Chartii
Home
Charts
About Us
Contact Us
Andreas
About Us:
Home / About Us
Andreou Andreas
Website Creator
University Of Cyprus
Check out my Website
Contact me here
More
Dikaiakos D. Marios
Thesis Supervisor
University Of Cyprus
Check out my Website
Contact me here
More
Antoniades Demetris
Thesis Advisor
University Of Cyprus
Check out my Website
Contact me here
More
Efstathiades Hariton
Thesis Advisor
University Of Cyprus
Check out my Website
Contact me here
More
© Chartii 2015
[Terms | Privacy]

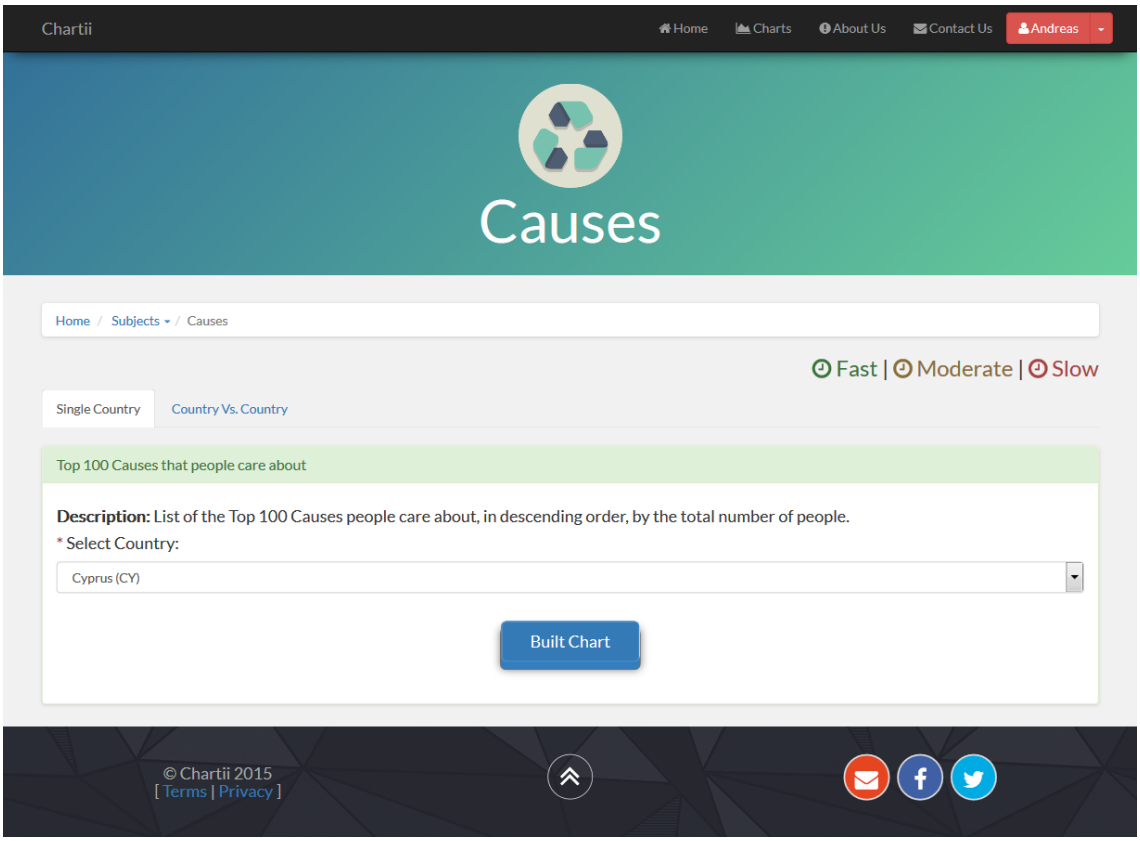
Contact Us:


Chartii
Home
Charts
About Us
Contact Us
Andreas
Contact Us:
Home / Contact Us
Location
Map
Satellite
Pedestrian Crossing
Thasou
Panepistimiou
FeedMe by Kalopésas
Centre for Entrepreneurship...
Photovoltaic Technology
University of Cyprus Apartments Φοιτητικές Εστίες
Department of Computer Science at...
Oceanography Center Ωκεανογραφικό Κέντρο Πανεπιστημίου Κύπρου
Κέντρο Περιβαλλοντικής Ενημέρωσης Εθνικού...
Google
Map data ©2016 Google
Terms of Use
Feel free to contact us
Andreou Andreas
andreasandreou1991@hotmail.com
Title (2 to 50 characters)
Message (up to 200 characters)
For security, please enter the code displayed in the box.
Send Feedback
Department
Department of Computer Science
Pure and Applied Sciences (FST-01)
University of Cyprus
1 University Avenue
2109 Aglantzia, CYPRUS
+357 22892700
Business hours
Day
Time
Monday
07:30 (08:30) - 14:30 (15:30)
Tuesday
07:30 (08:30) - 14:30 (15:30)
Wednesday
07:30 (08:30) - 14:30 (15:30)
Thursday
07:30 (08:30) - 14:30 (15:30)
Friday
07:30 (08:30) - 14:30 (15:30)
Saturday
day off
Sunday
day off
© Chartii 2015
[Terms | Privacy]

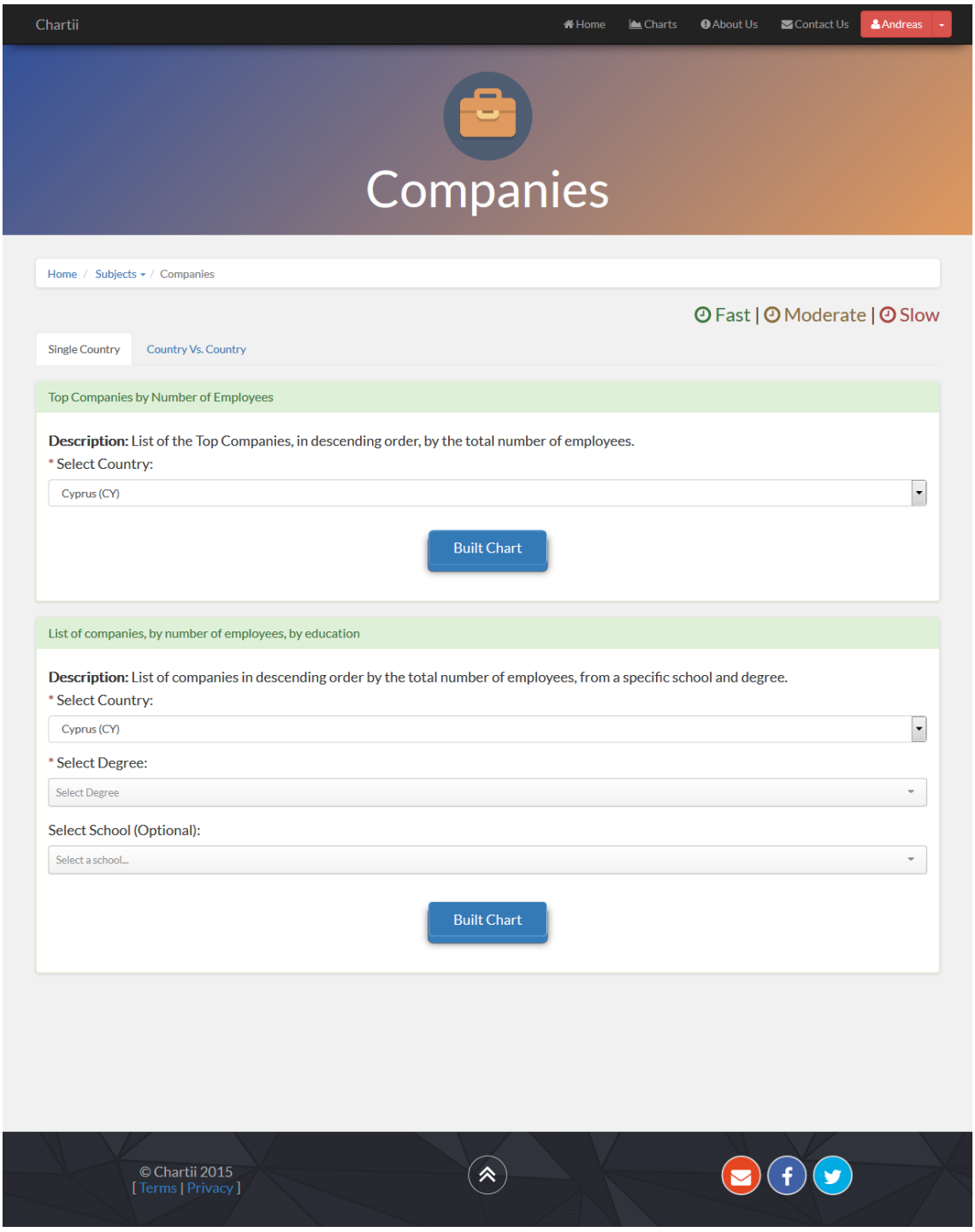
Charts:



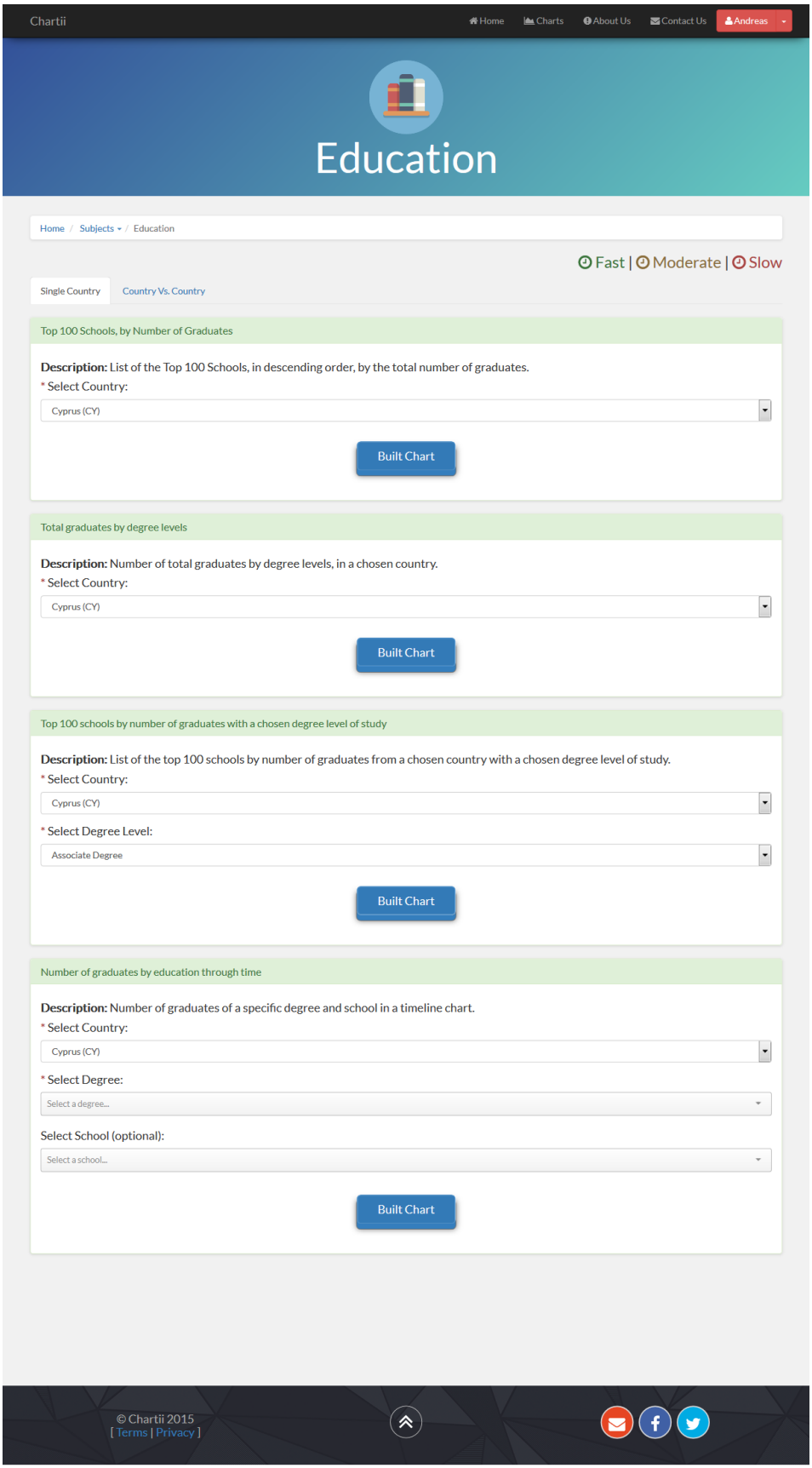
Causes:



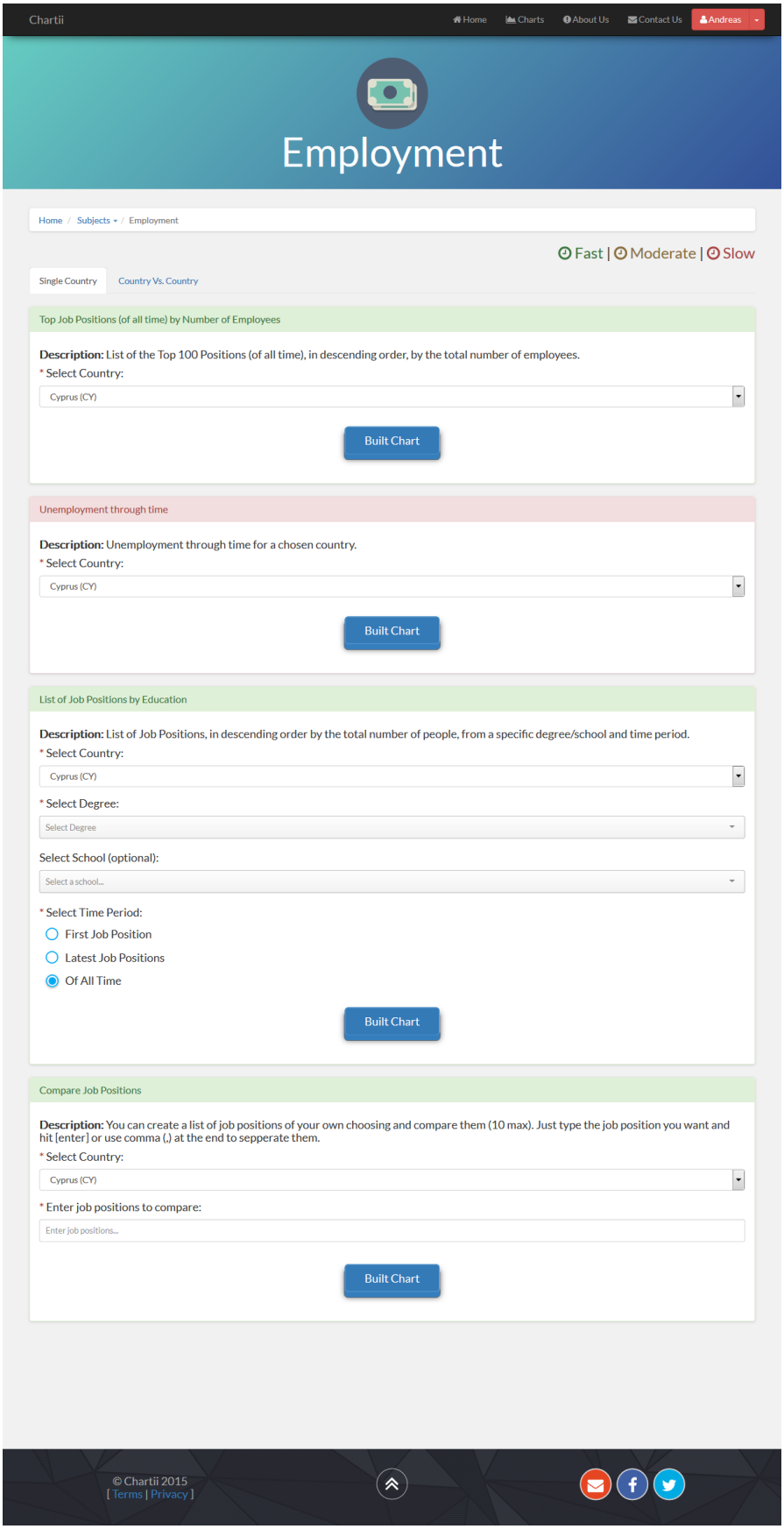
Companies:


Chartii
Home
Charts
About Us
Contact Us
Andreas
Companies
Home / Subjects / Companies
Fast | Moderate | Slow
Single Country
Country Vs. Country
Top Companies by Number of Employees
Description: List of the Top Companies, in descending order, by the total number of employees.
* Select Country:
Cyprus (CY)
Built Chart
List of companies, by number of employees, by education
Description: List of companies in descending order by the total number of employees, from a specific school and degree.
* Select Country:
Cyprus (CY)
* Select Degree:
Select Degree
Select School (Optional):
Select a school...
Built Chart
© Chartii 2015
[Terms | Privacy]

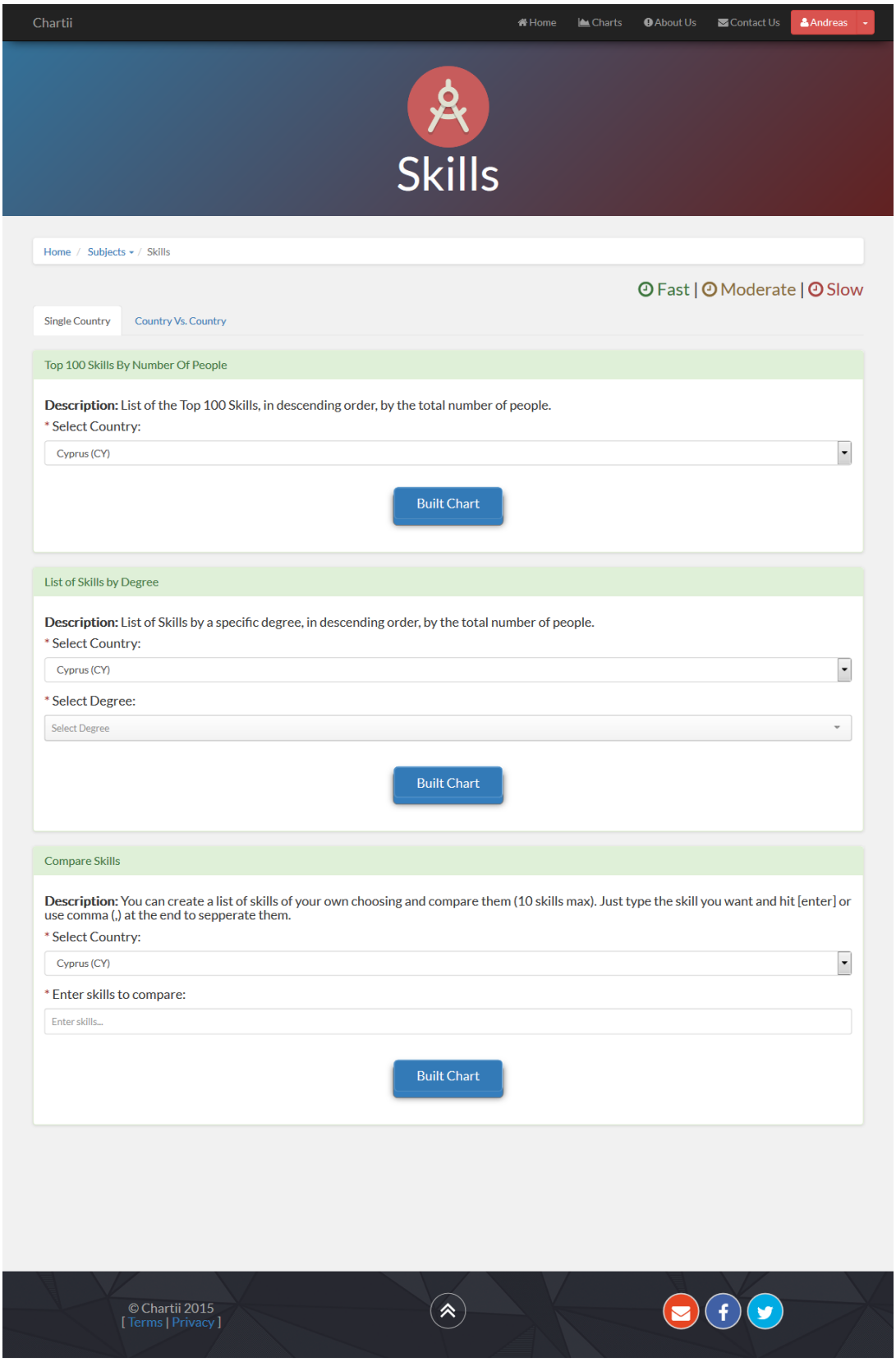
Education:


Chartii
Home
Charts
About Us
Contact Us
Andreas
Education
Home / Subjects / Education
Fast | Moderate | Slow
Single Country
Country Vs. Country
Top 100 Schools, by Number of Graduates
Description: List of the Top 100 Schools, in descending order, by the total number of graduates.
* Select Country:
Cyprus (CY)
Built Chart
Total graduates by degree levels
Description: Number of total graduates by degree levels, in a chosen country.
* Select Country:
Cyprus (CY)
Built Chart
Top 100 schools by number of graduates with a chosen degree level of study
Description: List of the top 100 schools by number of graduates from a chosen country with a chosen degree level of study.
* Select Country:
Cyprus (CY)
* Select Degree Level:
Associate Degree
Built Chart
Number of graduates by education through time
Description: Number of graduates of a specific degree and school in a timeline chart.
* Select Country:
Cyprus (CY)
* Select Degree:
Select a degree...
Select School (optional):
Select a school...
Built Chart
© Chartii 2015
[Terms | Privacy]

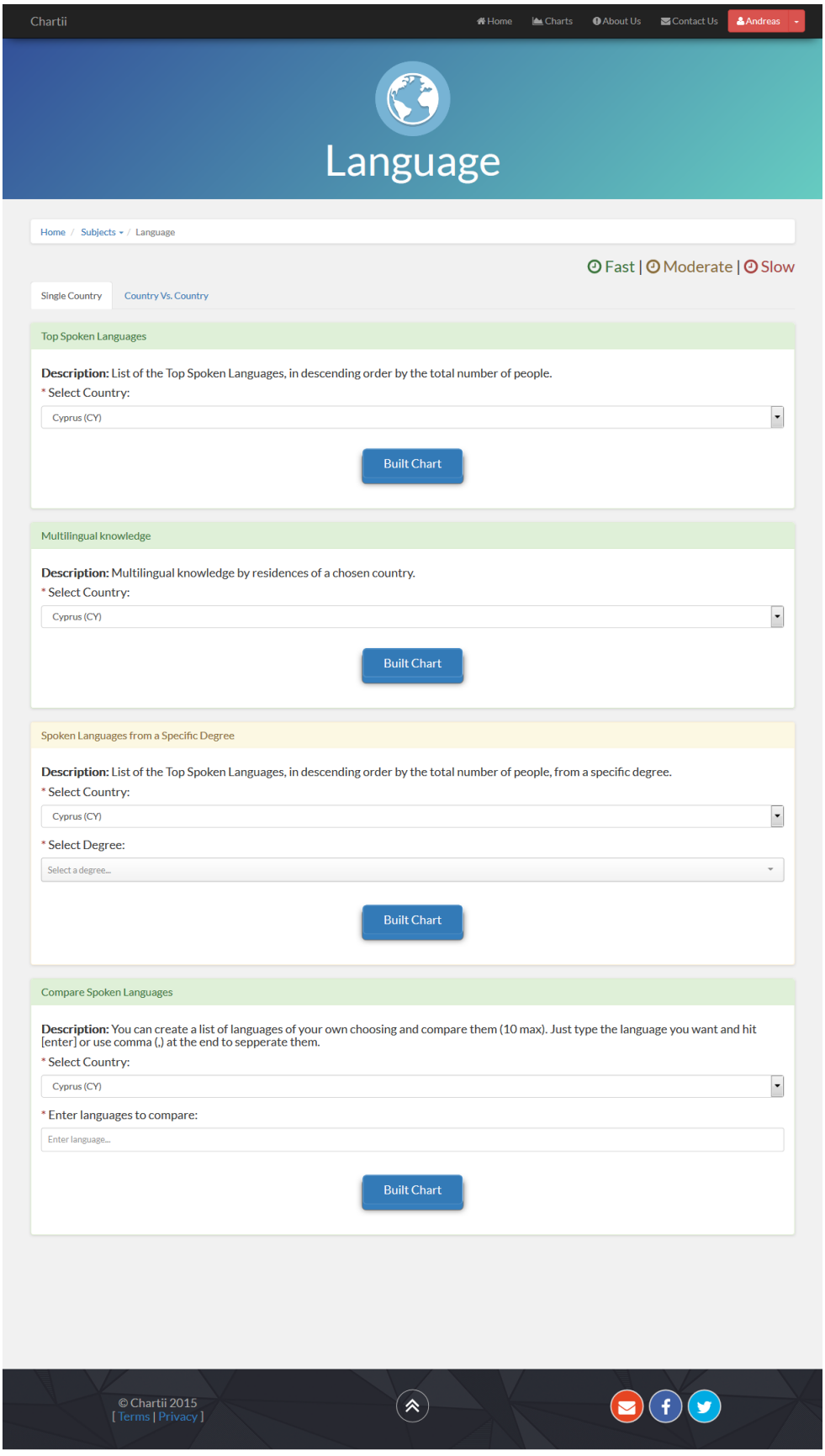
Employment:


Chartii
Home
Charts
About Us
Contact Us
Andreas
Employment
Home / Subjects / Employment
Fast | Moderate | Slow
Single Country
Country Vs. Country
Top Job Positions (of all time) by Number of Employees
Description: List of the Top 100 Positions (of all time), in descending order, by the total number of employees.
* Select Country:
Cyprus (CY)
Built Chart
Unemployment through time
Description: Unemployment through time for a chosen country.
* Select Country:
Cyprus (CY)
Built Chart
List of Job Positions by Education
Description: List of Job Positions, in descending order by the total number of people, from a specific degree/school and time period.
* Select Country:
Cyprus (CY)
* Select Degree:
Select Degree
Select School (optional):
Select a school...
* Select Time Period:
First Job Position
Latest Job Positions
Of All Time
Built Chart
Compare Job Positions
Description: You can create a list of job positions of your own choosing and compare them (10 max). Just type the job position you want and hit [enter] or use comma (,) at the end to sepperate them.
* Select Country:
Cyprus (CY)
* Enter job positions to compare:
Enter job positions...
Built Chart
© Chartii 2015
[Terms | Privacy]

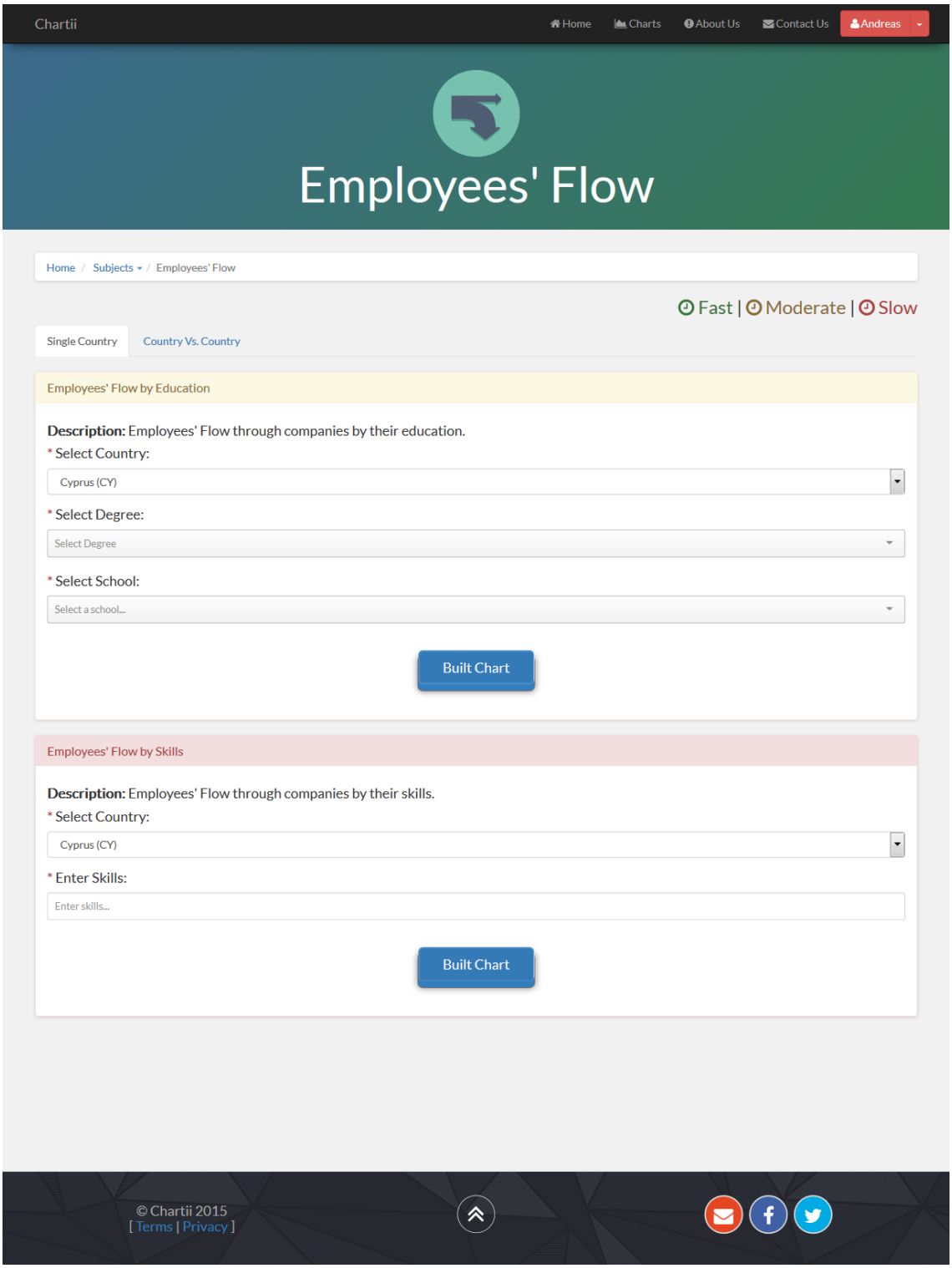
Skills:


Chartii
Home
Charts
About Us
Contact Us
Andreas
Skills
Home / Subjects / Skills
Fast | Moderate | Slow
Single Country
Country Vs. Country
Top 100 Skills By Number Of People
Description: List of the Top 100 Skills, in descending order, by the total number of people.
* Select Country:
Cyprus (CY)
Built Chart
List of Skills by Degree
Description: List of Skills by a specific degree, in descending order, by the total number of people.
* Select Country:
Cyprus (CY)
* Select Degree:
Select Degree
Built Chart
Compare Skills
Description: You can create a list of skills of your own choosing and compare them (10 skills max). Just type the skill you want and hit [enter] or use comma (,) at the end to sepperate them.
* Select Country:
Cyprus (CY)
* Enter skills to compare:
Enter skills...
Built Chart
© Chartii 2015
[Terms | Privacy]

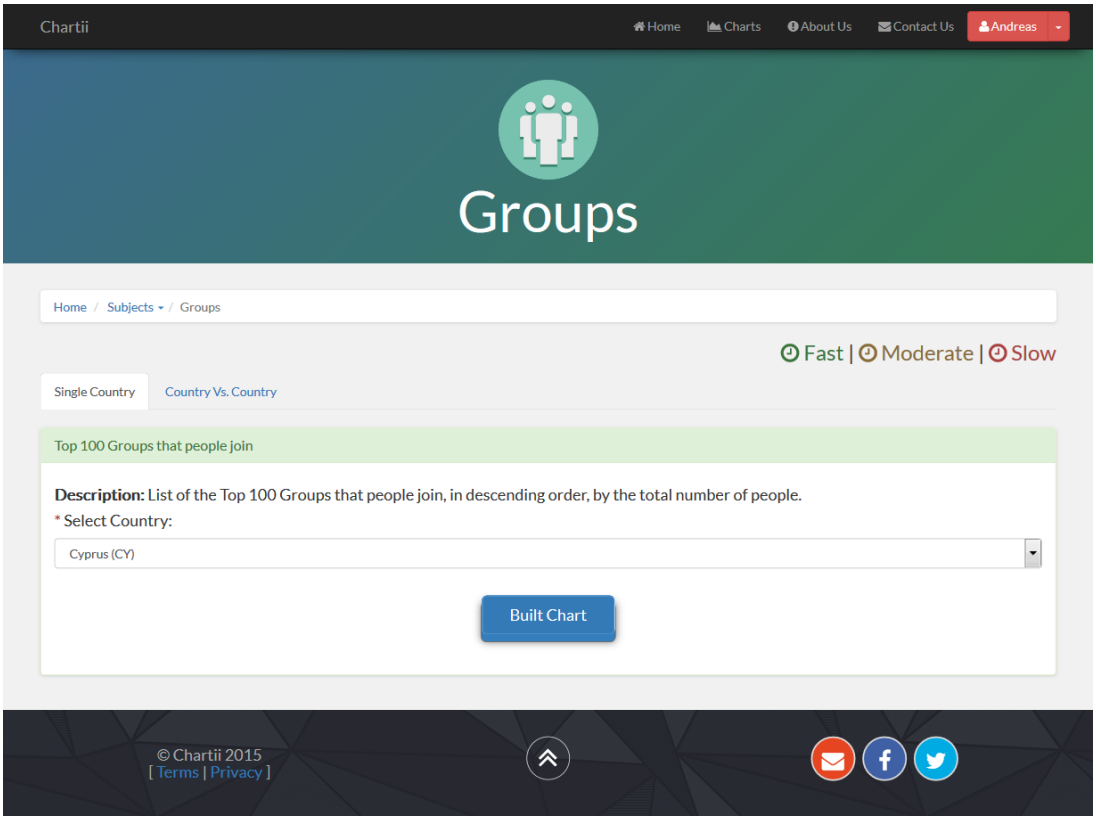
Language:


Chartii
Home
Charts
About Us
Contact Us
Andreas
Language
Home / Subjects / Language
Fast | Moderate | Slow
Single Country
Country Vs. Country
Top Spoken Languages
Description: List of the Top Spoken Languages, in descending order by the total number of people.
* Select Country:
Cyprus (CY)
Built Chart
Multilingual knowledge
Description: Multilingual knowledge by residences of a chosen country.
* Select Country:
Cyprus (CY)
Built Chart
Spoken Languages from a Specific Degree
Description: List of the Top Spoken Languages, in descending order by the total number of people, from a specific degree.
* Select Country:
Cyprus (CY)
* Select Degree:
Select a degree...
Built Chart
Compare Spoken Languages
Description: You can create a list of languages of your own choosing and compare them (10 max). Just type the language you want and hit [enter] or use comma (,) at the end to sepperate them.
* Select Country:
Cyprus (CY)
* Enter languages to compare:
Enter language...
Built Chart
© Chartii 2015
[Terms | Privacy]

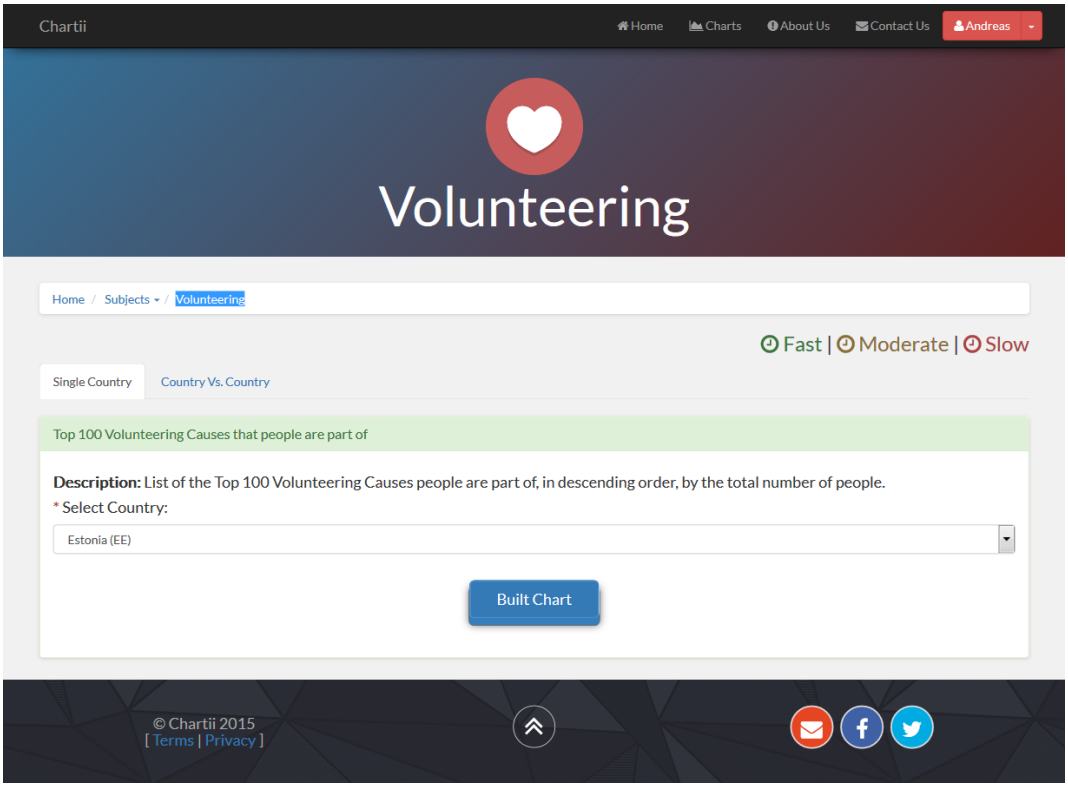
Employees' Flow:



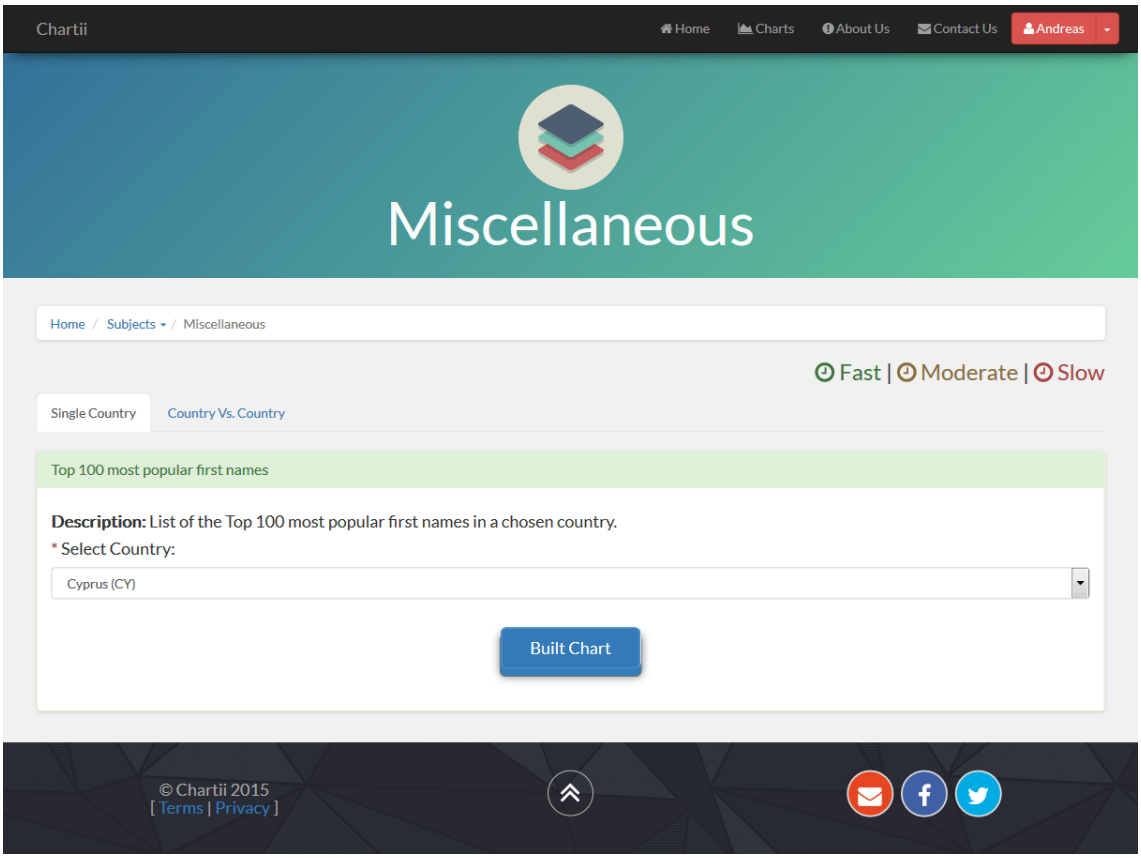
Groups:



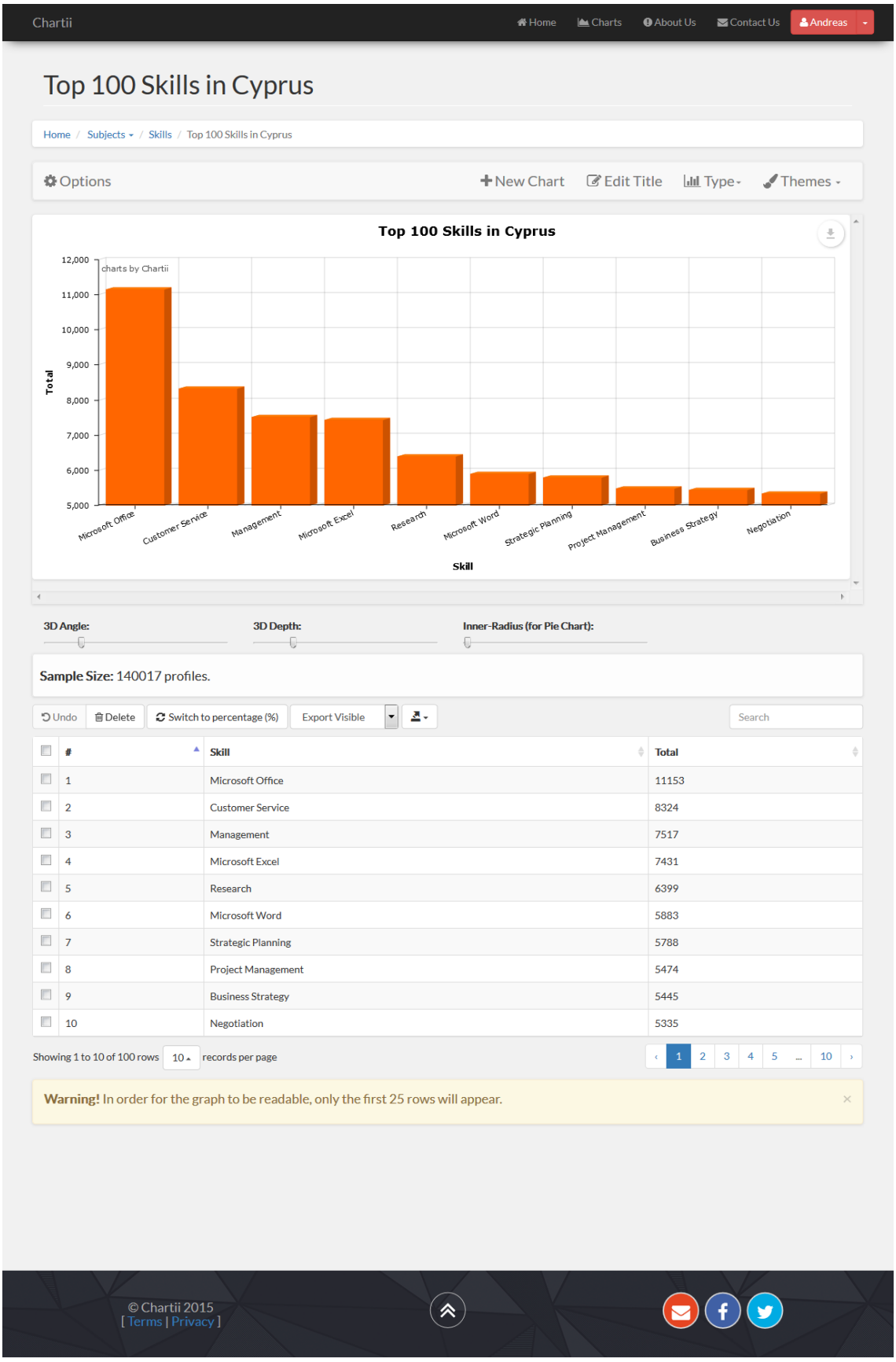
Volunteering:



Miscellaneous:



Visualizer1:



Visualizer2:

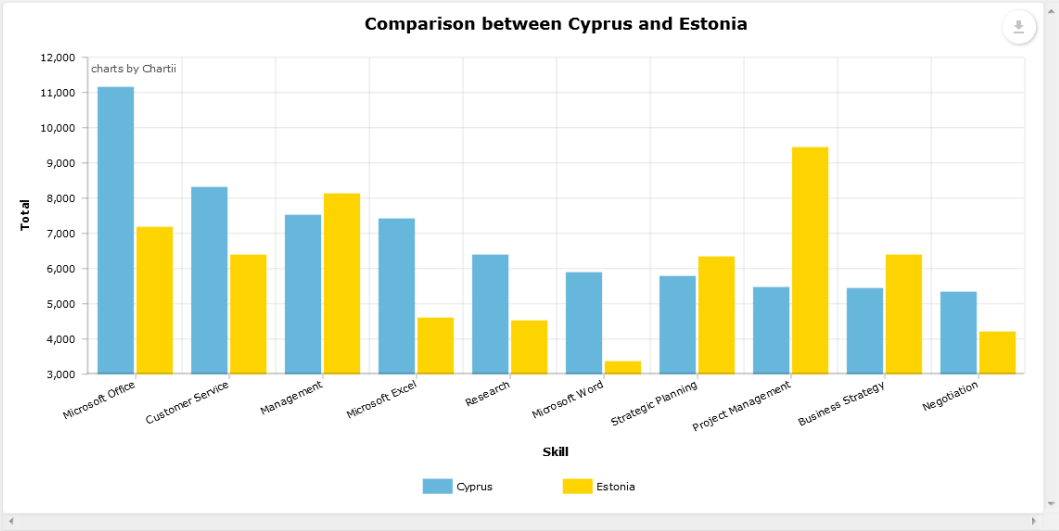
Chartii

Home | Charts | About Us | Contact Us | Andreas

Comparison between Cyprus and Estonia

Home / Subjects / Skills / Comparison between Cyprus and Estonia

Options | New Chart | Edit Title | Type | Themes



3D Angle: 3D Depth:

Sample Size 1 (Cyprus): 140017 profiles.
Sample Size 2 (Estonia): 125253 profiles.

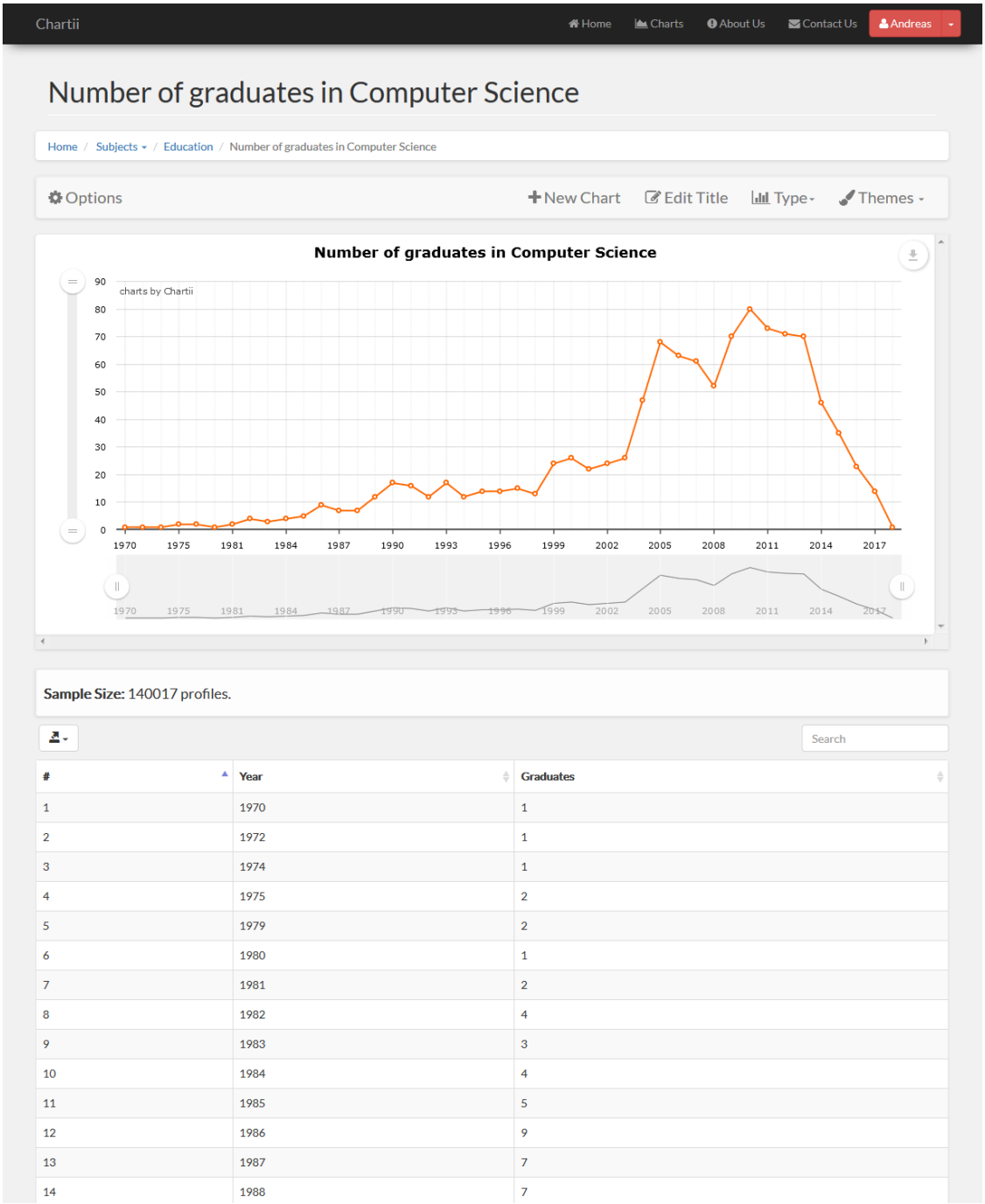
Undo | Delete | Switch to percentage (%) | Export Visible | Search

#	Skill	Cyprus	Estonia
1	Microsoft Office	11153	7181
2	Customer Service	8324	6385
3	Management	7517	8127
4	Microsoft Excel	7431	4618
5	Research	6399	4523
6	Microsoft Word	5883	3374
7	Strategic Planning	5788	6340
8	Project Management	5474	9443
9	Business Strategy	5445	6395
10	Negotiation	5335	4208

Showing 1 to 10 of 100 rows 10 records per page ‹ 1 2 3 4 5 … 10 ›

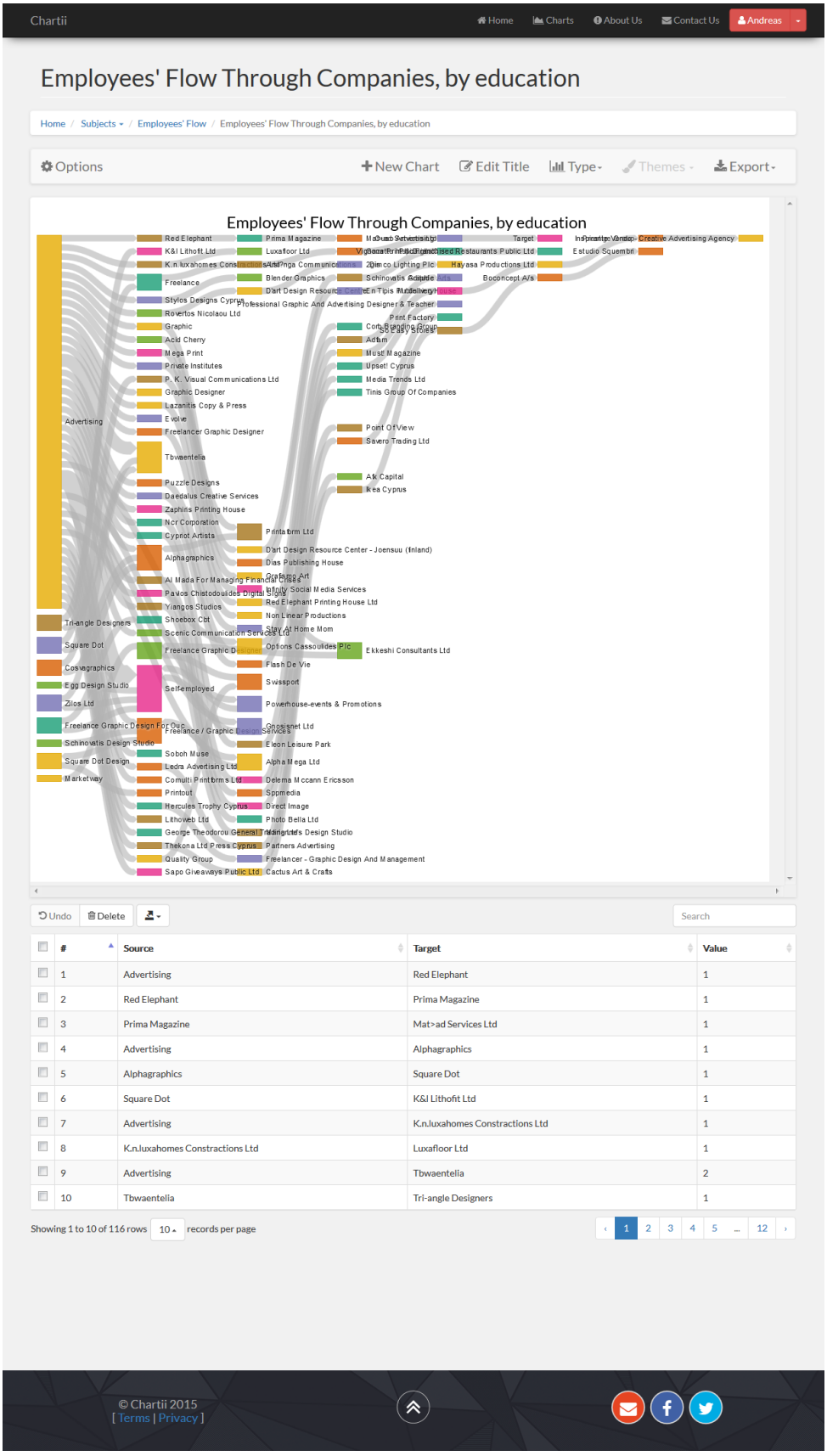
Warning! In order for the graph to be readable, only the first 25 rows will appear. ×

Visualizer3:



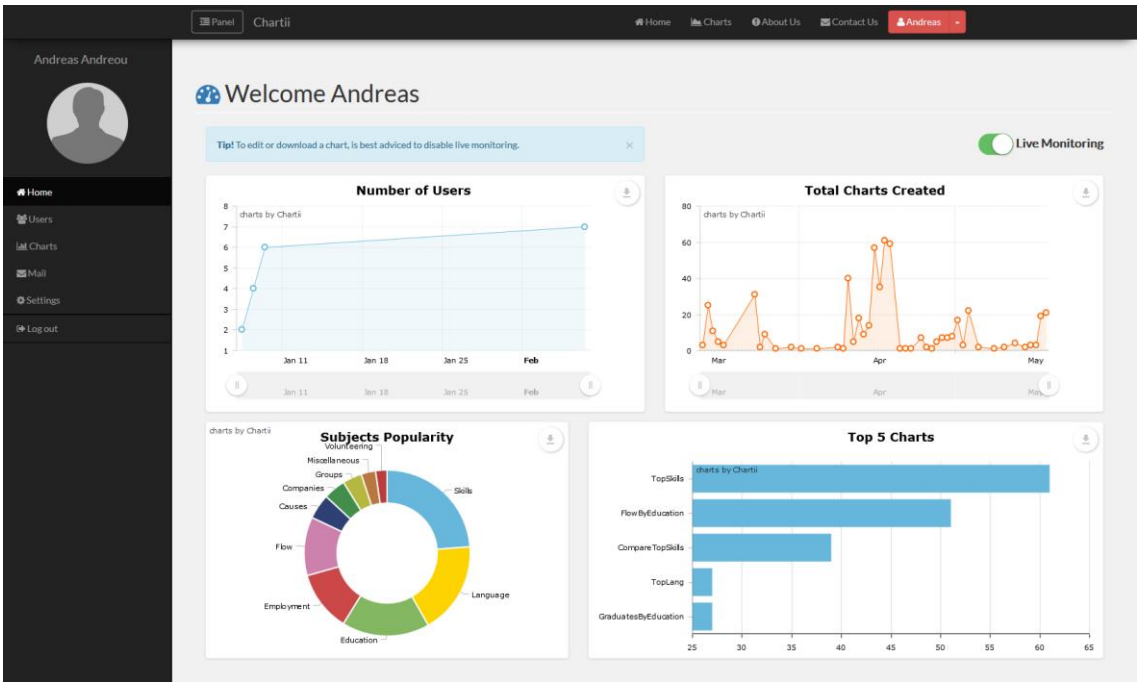
#	Year	Graduates
1	1970	1
2	1972	1
3	1974	1
4	1975	2
5	1979	2
6	1980	1
7	1981	2
8	1982	4
9	1983	3
10	1984	4
11	1985	5
12	1986	9
13	1987	7
14	1988	7

Visualizer4:

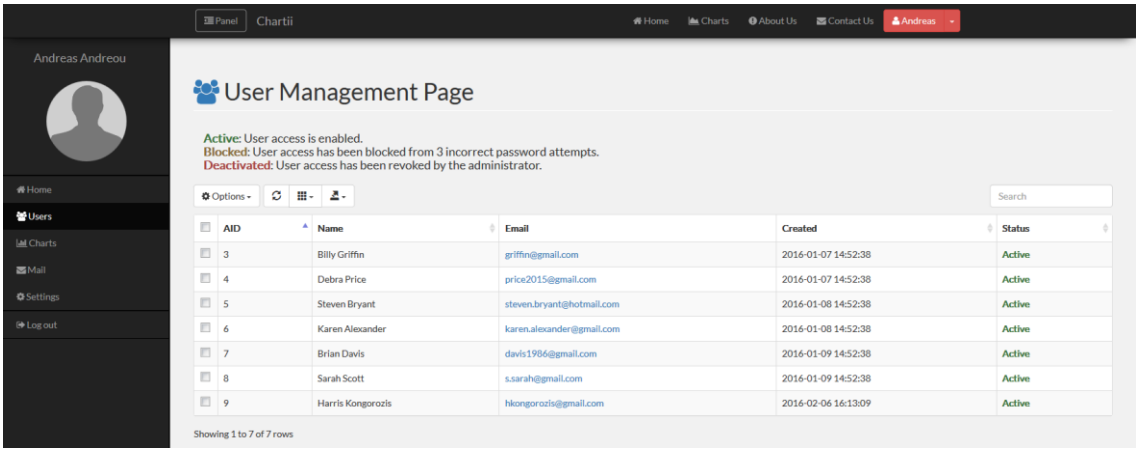


#	Source	Target	Value
1	Advertising	Red Elephant	1
2	Red Elephant	Prima Magazine	1
3	Prima Magazine	Mat>ad Services Ltd	1
4	Advertising	Alphagraphics	1
5	Alphagraphics	Square Dot	1
6	Square Dot	K&I Lithofit Ltd	1
7	Advertising	K.n.luxahomes Constractions Ltd	1
8	K.n.luxahomes Constractions Ltd	Luxafloor Ltd	1
9	Advertising	Tbwaentelia	2
10	Tbwaentelia	Tri-angle Designers	1

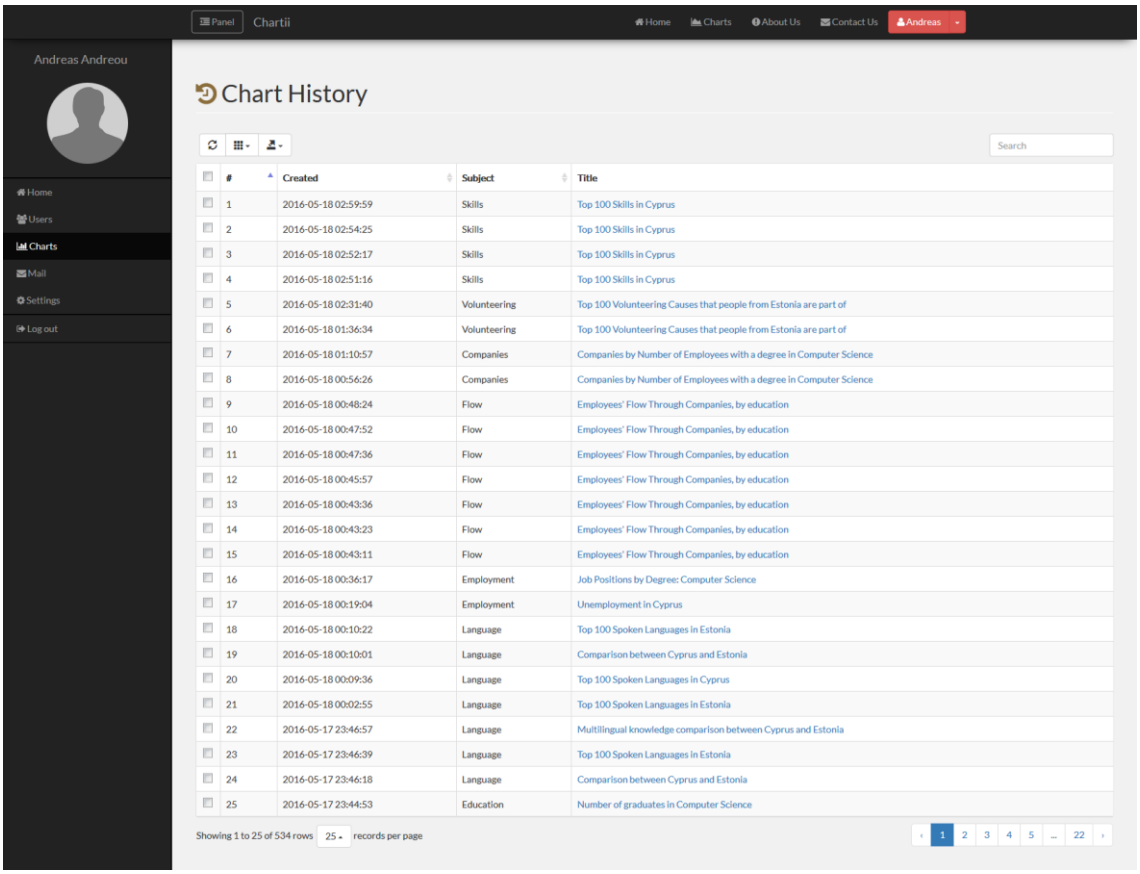
Admin/index:



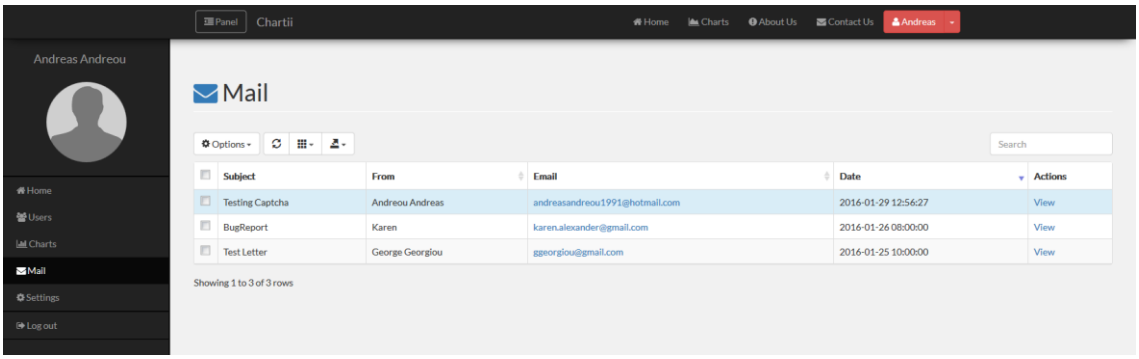
Admin/users:



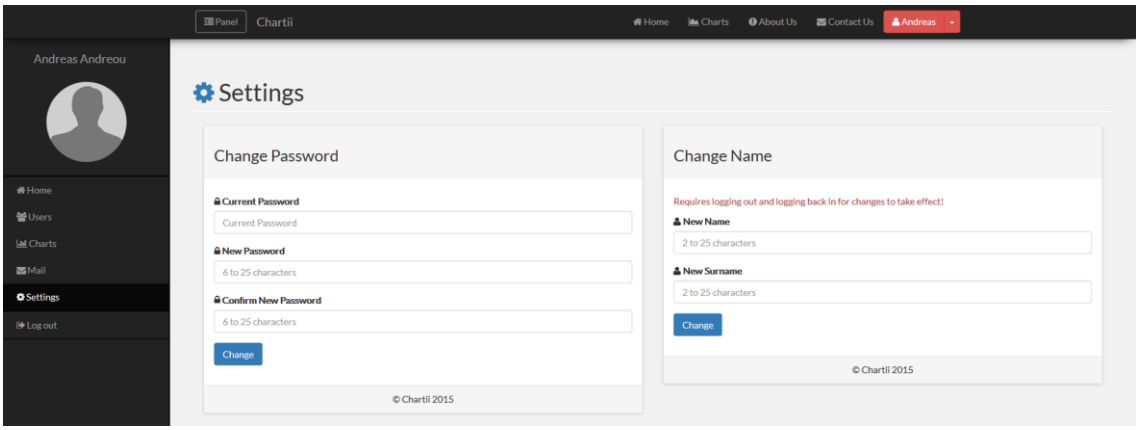
Admin/charts:



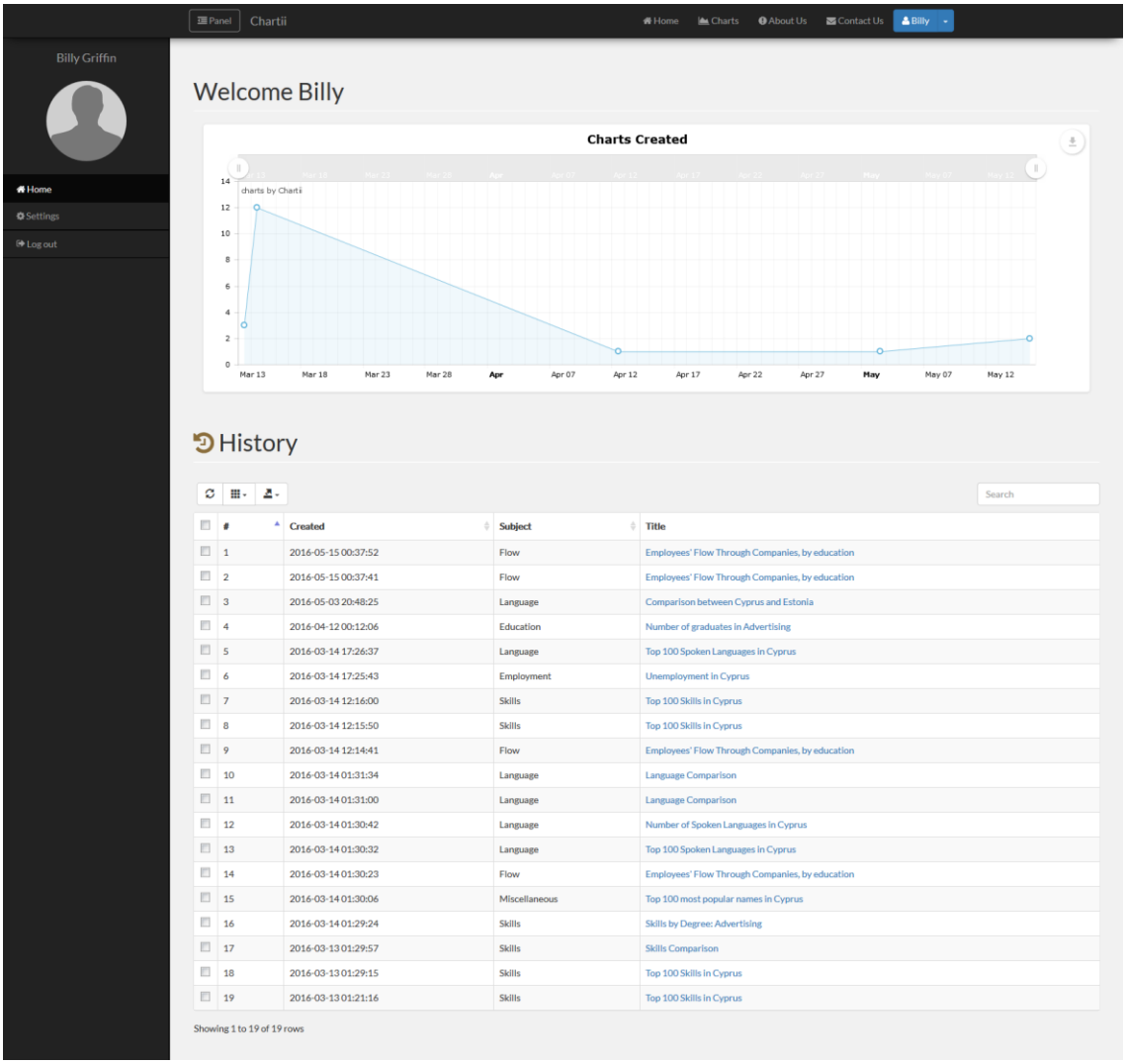
Admin/mail:



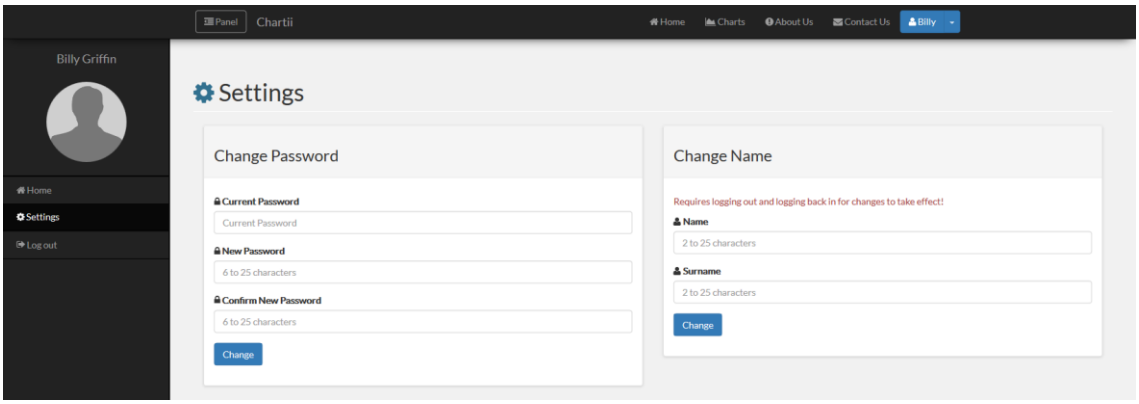
Admin/settings:



User/index:



User/settings:



Παράρτημα Β

Το Παράρτημα Β περιλαμβάνει το κώδικα δημιουργίας Βάσης Δεδομένων. Το πρόθεμα "CC_" αναπαριστά τα αρχικά της κάθε χώρας:

```
DROP TABLE IF EXISTS `EE_AUTHORS`;

CREATE TABLE IF NOT EXISTS `EE_AUTHORS` (
  `MEMBER_ID` bigint(20) NOT NULL,
  `PUBLICATION_ID` varchar(45) NOT NULL,
  PRIMARY KEY (`MEMBER_ID`,`PUBLICATION_ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `EE_CAUSES`;

CREATE TABLE IF NOT EXISTS `EE_CAUSES` (
  `ID` int(50) NOT NULL,
  `NAME` varchar(200) NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `EE_CERTIFICATIONS`;

CREATE TABLE IF NOT EXISTS `EE_CERTIFICATIONS` (
  `CERTIFICATION_ID` varchar(45) COLLATE utf8_unicode_ci NOT NULL,
  `USER_ID` bigint(20) NOT NULL,
  `TITLE` varchar(1000) COLLATE utf8_unicode_ci DEFAULT NULL,
  `ORGANIZATION` varchar(1000) COLLATE utf8_unicode_ci DEFAULT
NULL,
  `DAY` int(11) DEFAULT NULL,
  `MONTH` int(11) DEFAULT NULL,
  `YEAR` int(11) DEFAULT NULL,
  `LICENCE_NUMBER` varchar(200) COLLATE utf8_unicode_ci DEFAULT
NULL,
  PRIMARY KEY (`certification_id`,`user_id`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

DROP TABLE IF EXISTS `EE_COMPANIES`;

CREATE TABLE `EE_COMPANIES` (
  `ID` bigint(20) NOT NULL UNIQUE,
  `NAME` varchar(1000) DEFAULT NULL,
  `TYPE` varchar(45) DEFAULT NULL,
  `SIZE` varchar(45) DEFAULT NULL,
  `INDUSTRY` varchar(100) DEFAULT NULL,
  `WEBSITE` varchar(1000) DEFAULT NULL,
  `HEADQUARTERS` varchar(1000) DEFAULT NULL,
  `DESCRIPTION` varchar(10000) DEFAULT NULL,
  `FOUNDED_YEAR` varchar(20) DEFAULT NULL,
  `LINKEDIN_URL` varchar(1000) DEFAULT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

```
DROP TABLE IF EXISTS `EE_COMPANIES_SPECIALTIES`;

CREATE TABLE IF NOT EXISTS `EE_COMPANIES_SPECIALTIES` (
  `COMPANY_ID` varchar(45) COLLATE utf8_unicode_ci NOT NULL,
  `SPECIALTY` varchar(250) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`COMPANY_ID`,`SPECIALTY`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

DROP TABLE IF EXISTS `EE_COURSES`;

CREATE TABLE IF NOT EXISTS `EE_COURSES` (
  `ID` bigint(11) NOT NULL AUTO_INCREMENT,
  `NAME` varchar(1000) DEFAULT NULL,
  `COURSES_GROUP_ID` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=MyISAM  DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `EE_COURSES_GROUP`;

CREATE TABLE IF NOT EXISTS `EE_COURSES_GROUP` (
  `ID` varchar(45) NOT NULL,
  `SCHOOL` varchar(1000) DEFAULT NULL,
  `PROGRAM` varchar(1000) DEFAULT NULL,
  `USER_ID` bigint(20) NOT NULL,
  PRIMARY KEY (`ID`),
  KEY `USER_ID` (`USER_ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `EE_EDUCATIONS`;

CREATE TABLE IF NOT EXISTS `EE_EDUCATIONS` (
  `ID` varchar(45) NOT NULL,
  `SCHOOL_ID` bigint(20) DEFAULT NULL,
  `DEGREE` varchar(150) DEFAULT NULL,
  `MAJOR` varchar(200) DEFAULT NULL,
  `START_DAY` int(11) DEFAULT NULL,
  `START_MONTH` int(11) DEFAULT NULL,
  `START_YEAR` int(11) DEFAULT NULL,
  `END_DAY` int(11) DEFAULT NULL,
  `END_MONTH` int(11) DEFAULT NULL,
  `END_YEAR` int(11) DEFAULT NULL,
  `USER_ID` bigint(20) NOT NULL,
  PRIMARY KEY (`ID`),
  KEY `SCHOOL_ID` (`SCHOOL_ID`),
  KEY `DEGREE` (`DEGREE`),
  KEY `START_YEAR` (`START_YEAR`),
  KEY `END_YEAR` (`END_YEAR`),
  KEY `USER_ID` (`USER_ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `EE_GROUPS`;

CREATE TABLE IF NOT EXISTS `EE_GROUPS` (
  `ID` varchar(45) NOT NULL,
  `TITLE` varchar(1000) COLLATE utf8_unicode_ci DEFAULT NULL,
  `URL` varchar(1000) COLLATE utf8_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`ID`)
```

```
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

DROP TABLE IF EXISTS `EE_GROUPS_DETAILS`;

CREATE TABLE IF NOT EXISTS `EE_GROUPS_DETAILS` (
  `GROUP_ID` varchar(45) COLLATE utf8_unicode_ci NOT NULL,
  `CREATED_DATE` varchar(45) COLLATE utf8_unicode_ci DEFAULT NULL,
  `TYPE` varchar(45) COLLATE utf8_unicode_ci DEFAULT NULL,
  `MEMBERS` int(11) DEFAULT NULL,
  `SUBGROUPS` int(11) DEFAULT NULL,
  `OWNER_ID` varchar(45) COLLATE utf8_unicode_ci DEFAULT NULL,
  `WEBSITE` varchar(1000) COLLATE utf8_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`GROUP_ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

DROP TABLE IF EXISTS `EE_INTERESTS`;

CREATE TABLE IF NOT EXISTS `EE_INTERESTS` (
  `ID` int(50) NOT NULL,
  `NAME` varchar(400) NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `EE_LANGUAGES`;

CREATE TABLE IF NOT EXISTS `EE_LANGUAGES` (
  `ID` int(50) NOT NULL,
  `NAME` varchar(200) NOT NULL,
  PRIMARY KEY (`ID`),
  KEY `ID` (`ID`),
  KEY `NAME` (`NAME`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `EE_MEMBERS`;

CREATE TABLE IF NOT EXISTS `EE_MEMBERS` (
  `ID` bigint(20) NOT NULL,
  `FULL_NAME` varchar(1000) NOT NULL,
  `PROFILE_URL` varchar(1000) NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `EE_OPPORTUNITIES`;

CREATE TABLE IF NOT EXISTS `EE_OPPORTUNITIES` (
  `ID` int(50) NOT NULL,
  `NAME` varchar(200) NOT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `EE_PARTNERSHIP`;

CREATE TABLE IF NOT EXISTS `EE_PARTNERSHIP` (
  `MEMBER_ID` bigint(20) NOT NULL,
  `PROJECT_ID` varchar(45) NOT NULL,
  PRIMARY KEY (`MEMBER_ID`,`PROJECT_ID`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

```
DROP TABLE IF EXISTS `EE_PROJECTS`;

CREATE TABLE IF NOT EXISTS `EE_PROJECTS` (
  `ID` varchar(45) COLLATE utf8_unicode_ci NOT NULL,
  `TITLE` TEXT COLLATE utf8_unicode_ci DEFAULT NULL,
  `ABSTRACT` TEXT COLLATE utf8_unicode_ci DEFAULT NULL,
  `URL` varchar(1000) COLLATE utf8_unicode_ci DEFAULT NULL,
  `DATES` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

DROP TABLE IF EXISTS `EE_PUBLICATIONS`;

CREATE TABLE IF NOT EXISTS `EE_PUBLICATIONS` (
  `ID` varchar(45) NOT NULL,
  `TITLE` TEXT DEFAULT NULL,
  `VENUE` varchar(1000) DEFAULT NULL,
  `DATE` varchar(45) DEFAULT NULL,
  `SUMMARY` TEXT DEFAULT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `EE_SCHOOLS`;

CREATE TABLE IF NOT EXISTS `EE_SCHOOLS` (
  `ID` bigint(20) NOT NULL,
  `NAME` varchar(200) NOT NULL,
  `URL` varchar(1000) DEFAULT NULL,
  PRIMARY KEY (`ID`),
  KEY `ID` (`ID`),
  KEY `NAME` (`NAME`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `EE_SKILLS`;

CREATE TABLE IF NOT EXISTS `EE_SKILLS` (
  `ID` int(50) NOT NULL,
  `NAME` varchar(200) NOT NULL,
  PRIMARY KEY (`ID`),
  KEY `NAME` (`NAME`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `EE_USERS`;

CREATE TABLE IF NOT EXISTS `EE_USERS` (
  `ID` bigint(20) NOT NULL UNIQUE,
  `FIRST_NAME` varchar(45) DEFAULT NULL,
  `LAST_NAME` varchar(100) DEFAULT NULL,
  `HEADLINE` varchar(500) DEFAULT NULL,
  `SUMMARY` TEXT DEFAULT NULL,
  `LOCATION` varchar(45) DEFAULT NULL,
  `INDUSTRY` varchar(45) DEFAULT NULL,
  `PROFILE_URL` varchar(1000) NOT NULL,
  `IMAGE_URL` varchar(1000) DEFAULT NULL,
  `CONN_NUM` int(11) DEFAULT NULL,
  `RECOMMENDATIONS` int(11) DEFAULT NULL,
```

```
  `GROUPS_NUM` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE EE_USERS ADD INDEX (LOCATION);
ALTER TABLE EE_USERS ADD INDEX (FIRST_NAME);
ALTER TABLE EE_USERS ADD INDEX (ID);

DROP TABLE IF EXISTS `EE_USERS_CAUSES`;

CREATE TABLE IF NOT EXISTS `EE_USERS_CAUSES` (
  `USER_ID` bigint(20) NOT NULL,
  `CAUSE_ID` int(50) NOT NULL,
  PRIMARY KEY (`USER_ID`,`CAUSE_ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

DROP TABLE IF EXISTS `EE_USERS_GROUPS`;

CREATE TABLE IF NOT EXISTS `EE_USERS_GROUPS` (
  `USER_ID` bigint(20) NOT NULL,
  `GROUP_ID` varchar(45) COLLATE utf8_unicode_ci NOT NULL,
  PRIMARY KEY (`USER_ID`,`GROUP_ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

DROP TABLE IF EXISTS `EE_USERS_INTERESTS`;

CREATE TABLE IF NOT EXISTS `EE_USERS_INTERESTS` (
  `USER_ID` bigint(20) NOT NULL,
  `INTEREST_ID` int(50) NOT NULL,
  PRIMARY KEY (`USER_ID`,`INTEREST_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

DROP TABLE IF EXISTS `EE_USERS_LANGUAGES`;

CREATE TABLE IF NOT EXISTS `EE_USERS_LANGUAGES` (
  `USER_ID` bigint(20) NOT NULL,
  `LANGUAGE_ID` int(50) NOT NULL,
  `PROFICIENCY` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`USER_ID`,`LANGUAGE_ID`),
  KEY `USER_ID` (`USER_ID`),
  KEY `LANGUAGE_ID` (`LANGUAGE_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

DROP TABLE IF EXISTS `EE_USERS_OPPORTUNITIES`;

CREATE TABLE IF NOT EXISTS `EE_USERS_OPPORTUNITIES` (
  `USER_ID` bigint(20) NOT NULL,
  `OPPORTUNITY_ID` int(50) NOT NULL,
  PRIMARY KEY (`USER_ID`,`OPPORTUNITY_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

DROP TABLE IF EXISTS `EE_USERS_POSITIONS`;

CREATE TABLE IF NOT EXISTS `EE_USERS_POSITIONS` (
  `USER_ID` bigint(20) NOT NULL,
  `POSITION_ID` varchar(45) NOT NULL,
  `SUMMARY` varchar(5000) DEFAULT NULL,
```

```
  `IS_CURRENT` int(11) DEFAULT NULL,
  `COMPANY_ID` varchar(45) DEFAULT NULL,
  `START_DAY` int(11) DEFAULT NULL,
  `START_MONTH` int(11) DEFAULT NULL,
  `START_YEAR` int(11) DEFAULT NULL,
  `END_DAY` int(11) DEFAULT NULL,
  `END_MONTH` int(11) DEFAULT NULL,
  `END_YEAR` int(11) DEFAULT NULL,
  `TITLE` varchar(500) DEFAULT NULL,
  `LOCATION` varchar(200) DEFAULT NULL,
  PRIMARY KEY (`USER_ID`,`POSITION_ID`),
  KEY `USER_ID` (`USER_ID`),
  KEY `POSITION_ID` (`POSITION_ID`),
  KEY `COMPANY_ID` (`COMPANY_ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

DROP TABLE IF EXISTS `EE_USERS_SKILLS`;

CREATE TABLE IF NOT EXISTS `EE_USERS_SKILLS` (
  `USER_ID` bigint(20) NOT NULL,
  `SKILL_ID` int(50) NOT NULL,
  PRIMARY KEY (`USER_ID`,`SKILL_ID`),
  KEY `USER_ID` (`USER_ID`),
  KEY `SKILL_ID` (`SKILL_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

DROP TABLE IF EXISTS `EE_VOLUNTEERING`;

CREATE TABLE IF NOT EXISTS `EE_VOLUNTEERING` (
  `ID` varchar(45) NOT NULL,
  `USER_ID` bigint(20) NOT NULL,
  `TITLE` varchar(500) DEFAULT NULL,
  `SUBTITLE` varchar(500) DEFAULT NULL,
  `CAUSE` varchar(500) DEFAULT NULL,
  `START_DAY` int(11) DEFAULT NULL,
  `START_MONTH` int(11) DEFAULT NULL,
  `START_YEAR` int(11) DEFAULT NULL,
  `END_DAY` int(11) DEFAULT NULL,
  `END_MONTH` int(11) DEFAULT NULL,
  `END_YEAR` int(11) DEFAULT NULL,
  `DESCRIPTION` TEXT DEFAULT NULL,
  PRIMARY KEY (`ID`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Παράρτημα Γ

Το Παράρτημα Γ περιλαμβάνει το κώδικα δημιουργίας διαπροσωπιών και κλάσεων PHP για την δημιουργία γραφικών παραστάσεων:

Chart.class.php:

```
<?php

class Chart{

	/**
	 * Function that returns the code of a given country that
	 * is supported by Chartii's database (meaning having that
	 * country crawled).
	 */
	protected function getCountryCode($c) {
		switch($c){
			case("Cyprus"): return "CY_"; break;
			case("Estonia"): return "EE_"; break;
			default: return "";
		}
	}

	/**
	 * Function that returns true/false if the given country
	 * is supported by Chartii's database (meaning having that
	 * country crawled).
	 */
	protected function isCountrySupported($c){
		switch($c){
			case("Cyprus"): return true; break;
			case("Estonia"): return true; break;
			default: return false;
		}
	}

	/**
	 * Function to keep history of the requested charts
	 * (ignoring the refresh of a tab).
	 */
	protected function insertHistory() {
		include '../../_conn.php';
		$aid = dbSafe($_SESSION['AID']);
		$chart = dbSafe(get_class($this));
		$subject = dbSafe($this->getSubject());
		$title = dbSafe($this->getTitle());
		$url =
dbSafe("http://".$_SERVER['SERVER_NAME'].$_SERVER['PHP_SELF']."?".$
_SERVER['QUERY_STRING']);
		$sql="INSERT INTO CHARTS_HISTORY (AID, CHART, SUBJECT,
TITLE, URL) VALUES
				($aid, $chart, $subject, $title, $url)";
```

```
            $pageRefresh = isset($_SERVER['HTTP_CACHE_CONTROL']) &&
                           $_SERVER['HTTP_CACHE_CONTROL'] ===
'max-age=0';
            if ($pageRefresh==0){
                  if ($conn) {
                        mysqli_query($conn, $sql);
                        mysqli_close($conn);
                  }
            }
      }
}
?>
```

iChart.php:

```
<?php
interface iChart{

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData();

      /**
       * Returns the chart's title.
       */
      public function getTitle();

      /**
       * Returns the chart's Subject.
       */
      public function getSubject();

      /**
       * Returns the chart's Subject Page.
       */
      public function getSubjectPage();
}
?>
```

iBihisankey.php:

```
<?php

require('iChart.php');

interface iBihisankey extends iChart{

      /**
       * Returns the number of first level leafs in the graph
       */
      public function getFirstLevels();

      /**
       * Prints the Vertices table in JSON format.
```

```
        */
       public function printVertices();

       /**
        * Prints the Edges table in JSON format.
        */
       public function printEdges();

       /**
        * Prints the Edges table in a table format.
        */
       public function printEdgesTable();

       /**
        * Returns the maximum path taken by an employee.
        */
       public function getMaxHops();
}
?>
```

iCBCompare.php:

```
<?php

require('iChart.php');

interface iCBCompare extends iChart{

       /**
        * Returns the sample size for the first
        * country of the current graph.
        */
       public function getSampleSize1();

       /**
        * Returns the sample size for the second
        * country of the current graph.
        */
       public function getSampleSize2();

       /**
        * Returns the chart's X axis name.
        */
       public function getXaxisName();

       /**
        * Returns the name of the first country.
        */
       public function getCountry1();

       /**
        * Returns the name of the second country.
        */
       public function getCountry2();

       /**
```

```
        * Prints the array in JSON format.
        */
       public function printJSON();

       /**
        * Prints the array in a HTML table format.
        */
       public function printTable();
}
?>
```

iCPB.php:

```
<?php

require('iChart.php');

interface iCPB extends iChart{

       /**
        * Returns the sample size of the current
        * graph.
        */
       public function getSampleSize();

       /**
        * Returns the chart's X axis name.
        */
       public function getXaxisName();

       /**
        * Returns the chart's Y axis name.
        */
       public function getYaxisName();

       /**
        * Prints the array in JSON format.
        */
       public function printJSON();

       /**
        * Prints the array in a HTML table format.
        */
       public function printTable();

}
?>
```

iTimeline.php:

```
<?php

require('iChart.php');

interface iTimeline extends iChart{

	/**
	 * Returns the sample size of the current
	 * graph.
	 */
	public function getSampleSize();

	/**
	 * Returns the chart's Date Format for timeline graph.
	 */
	public function getDateFormat();

	/**
	 * Returns the chart's X axis name.
	 */
	public function getXaxisName();

	/**
	 * Returns the chart's Y axis name.
	 */
	public function getYaxisName();

	/**
	 * Prints the array in JSON format.
	 */
	public function printJSON();

	/**
	 * Prints the array in a HTML table format.
	 */
	public function printTable();

}
?>
```

TopCauses.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCPB.php');

class TopCauses extends Chart implements iCPB{
	private $country;
	private $array = array ();

	/**
	 * Class constructor. Checks required variables and
	 * calls the getData() function.
	 */
	public function __construct() {
		if (!isset($_GET['country']) || $_GET['country']==""){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: 'country' variable was not
given.";
			die();
		}
		$this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
		if (!$this->isCountrySupported($this->country)){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
			die();
		}
		$this->getData();
		if (!isset($_GET['count']))
			$this->insertHistory();
	}

	/**
	 * Communicates with database to return
	 * the appropriate data needed.
	 */
	public function getData() {
		include '../../_conn.php';
		$sql = "";
		$country = $this->country;
		$country_code = $this->getCountryCode($country);
		if ($country == "Cyprus")
			$sql = "SELECT * FROM CY_TOP_CAUSES LIMIT 0, 100";
		else if ($country == "Estonia")
			$sql = "SELECT * FROM EE_TOP_CAUSES LIMIT 0, 100";
		else
			$sql = "SELECT
					C.NAME As Cause,
					Count(UC.USER_ID) AS Total
				FROM ".$country_code."USERS U
				INNER JOIN
".$country_code."USERS_CAUSES UC ON U.ID=UC.USER_ID
				INNER JOIN ".$country_code."CAUSES C
ON UC.CAUSE_ID=C.ID
```

```
                              WHERE
                                    U.LOCATION LIKE '%$country%'
                              GROUP BY Cause
                              ORDER BY Total DESC
                              LIMIT 0, 100";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "Cause" => ucwords( strtolower(
mysql_real_escape_string($row['Cause']) ) ),
                                    "Total" => $row['Total']
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
       * Returns the sample size of the current
       * graph.
       */
      public function getSampleSize() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code."USERS U
                        WHERE LOCATION LIKE '%$country%'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        $row = mysqli_fetch_assoc ( $result );
                        $total = $row['TOTAL'];
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
            return $total;
      }

      /**
```

```
       * Returns the chart's title.
       */
      public function getTitle() {
            return "Top 100 Causes people from ".$this->country ."
care about";
      }

      /**
       * Returns the chart's Subject.
       */
      public function getSubject() {
            return 'Causes';
      }

      /**
       * Returns the chart's Subject Page.
       */
      public function getSubjectPage() {
            return '<a href="../causes/">Causes</a>';
      }

      /**
       * Returns the chart's X axis name.
       */
      public function getXaxisName(){
            return "Cause";
      }

      /**
       * Returns the chart's Y axis name.
       */
      public function getYaxisName(){
            return "Total";
      }

      /**
       * Prints the array in JSON format.
       */
      public function printJSON() {
            echo json_encode($this->array);
      }

      /**
       * Prints the array in a HTML table format.
       */
      public function printTable() {
            for($i = 0; $i < sizeof ( $this->array ); $i ++) {
                  $cause = $this->array[$i]["Cause"];
                  $total = $this->array[$i]["Total"];
                  echo '
                  <tr>
                        <td></td>
                        <td>'.($i+1).'</td>
                        <td>'.$cause.'</td>
                        <td>'.$total.'</td>
                  </tr>
                  ';
```

```
            }
      }
}
?>
```

CompareTopCauses.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCBCompare.php');

class CompareTopCauses extends Chart implements iCBCompare{
      private $country1;
      private $country2;
      private $array = array ();

      /**
       * Class constructor. Checks required variables and
       * calls the getData() function.
       */
      public function __construct() {
            if (!isset($_GET['country1']) || $_GET['country1']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country1' variable was not
given.";
                  die();
            }
            if (!isset($_GET['country2']) || $_GET['country2']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country2' variable was not
given.";
                  die();
            }
            $this->country1 = ucwords( strtolower(
mysql_real_escape_string($_GET['country1']) ) );
            $this->country2 = ucwords( strtolower(
mysql_real_escape_string($_GET['country2']) ) );
            if (!$this->isCountrySupported($this->country1)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country1 ." is
<b>not</b> a supported country.";
                  die();
            }
            if (!$this->isCountrySupported($this->country2)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country2 ." is
<b>not</b> a supported country.";
                  die();
            }
            $this->getData();
            if (!isset($_GET['count']))
                  $this->insertHistory();
      }

      /**
```

```
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $sql = "";
            $country1 = $this->country1;
            $country2 = $this->country2;
            $country_code1 = $this->getCountryCode($country1);
            $country_code2 = $this->getCountryCode($country2);
            $sql = "SELECT T1.Cause, T1.TOTAL AS TOTAL1,
IFNULL(T2.TOTAL, '0') AS TOTAL2
                        FROM (
                              SELECT * FROM
".$country_code1."TOP_CAUSES LIMIT 0,100
                        )T1
                        LEFT JOIN ".$country_code2."TOP_CAUSES T2 ON
T1.Cause=T2.Cause
                        ORDER BY TOTAL1 DESC";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "Cause" => ucwords( strtolower(
mysql_real_escape_string($row['Cause']) ) ),
                                    "Total1" => $row['TOTAL1'],
                                    "Total2" => $row['TOTAL2']
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
       * Returns the sample size for the first
       * country of the current graph.
       */
      public function getSampleSize1() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country1 = $this->country1;
            $country_code1 = $this->getCountryCode($country1);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code1."USERS U
                        WHERE LOCATION LIKE '%$country1%'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
```

```
                $result = mysqli_query ( $conn, $sql );
                if (mysqli_num_rows ( $result ) > 0) {
                      $row = mysqli_fetch_assoc ( $result );
                      $total = $row['TOTAL'];
                }
                mysqli_close($conn);
          }else{
                header('HTTP/1.1 503 Service Unavailable');
                echo "503 Service Unavailable";
                die();
          }
          return $total;
     }

     /**
      * Returns the sample size for the second
      * country of the current graph.
      */
     public function getSampleSize2() {
          include '../../_conn.php';
          $total = 0;
          $sql = "";
          $country2 = $this->country2;
          $country_code2 = $this->getCountryCode($country2);
          $sql = "SELECT COUNT(*) AS TOTAL
                      FROM ".$country_code2."USERS U
                      WHERE LOCATION LIKE '%$country2%'";
          if ($conn) {
                mysqli_set_charset ( $conn, "utf8" );
                $result = mysqli_query ( $conn, $sql );
                if (mysqli_num_rows ( $result ) > 0) {
                      $row = mysqli_fetch_assoc ( $result );
                      $total = $row['TOTAL'];
                }
                mysqli_close($conn);
          }else{
                header('HTTP/1.1 503 Service Unavailable');
                echo "503 Service Unavailable";
                die();
          }
          return $total;
     }

     /**
      * Returns the chart's title.
      */
     public function getTitle() {
          return "Causes Comparison between ".$this->country1 ."
and ".$this->country2;
     }

     /**
      * Returns the chart's Subject.
      */
     public function getSubject() {
          return 'Causes';
     }
```

```
	/**
	 * Returns the chart's Subject Page.
	 */
	public function getSubjectPage() {
		return '<a href="../causes/">Causes</a>';
	}

	/**
	 * Returns the chart's X axis name.
	 */
	public function getXaxisName(){
		return "Cause";
	}

	/**
	 * Returns the name of the first country.
	 */
	public function getCountry1(){
		return $this->country1;
	}

	/**
	 * Returns the name of the second country.
	 */
	public function getCountry2(){
		return $this->country2;
	}

	/**
	 * Prints the array in JSON format.
	 */
	public function printJSON() {
		echo json_encode($this->array);
	}

	/**
	 * Prints the array in a HTML table format.
	 */
	public function printTable() {
		for($i = 0; $i < sizeof ( $this->array ); $i ++) {
			$cause = $this->array[$i]["Cause"];
			$total1 = $this->array[$i]["Total1"];
			$total2 = $this->array[$i]["Total2"];
			echo '
			<tr>
				<td></td>
				<td>'.($i+1).'</td>
				<td>'.$cause.'</td>
				<td>'.$total1.'</td>
				<td>'.$total2.'</td>
			</tr>
			';
		}
	}
}
?>
```

TopCompanies.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCPB.php');

class TopCompanies extends Chart implements iCPB{
	private $country;
	private $array = array ();

	/**
	 * Class constructor. Checks required variables and
	 * calls the getData() function.
	 */
	public function __construct() {
		if (!isset($_GET['country']) || $_GET['country']==""){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: 'country' variable was not
given.";
			die();
		}
		$this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
		if (!$this->isCountrySupported($this->country)){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
			die();
		}
		$this->getData ();
		if (!isset($_GET['count']))
			$this->insertHistory();
	}

	/**
	 * Communicates with database to return
	 * the appropriate data needed.
	 */
	public function getData() {
		include '../../_conn.php';
		$sql = "";
		$country = $this->country;
		$country_code = $this->getCountryCode($country);
		if ($country == "Cyprus")
			$sql = "SELECT * FROM CY_TOP_COMPANIES LIMIT 0,
100";
		else if ($country == "Estonia")
			$sql = "SELECT * FROM EE_TOP_COMPANIES LIMIT 0,
100";
		else
			$sql = "SELECT
					C.NAME AS Company,
					COUNT(UP.USER_ID) AS Total
				FROM ".$country_code."USERS U
```

```
                              INNER JOIN
".$country_code."USERS_POSITIONS UP ON U.ID=UP.USER_ID
                              INNER JOIN ".$country_code."COMPANIES
C ON UP.COMPANY_ID=C.ID
                              WHERE
                                    U.LOCATION LIKE '%$country%'
                              GROUP BY Company
                              ORDER BY Total DESC
                              LIMIT 0, 100";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "Company" => ucwords( strtolower(
mysql_real_escape_string($row['Company']) ) ),
                                    "Total" => ucwords( strtolower(
mysql_real_escape_string($row['Total']) ) )
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
       * Returns the sample size of the current
       * graph.
       */
      public function getSampleSize() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code."USERS U
                        WHERE LOCATION LIKE '%$country%'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        $row = mysqli_fetch_assoc ( $result );
                        $total = $row['TOTAL'];
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
```

```
            }
            return $total;
      }

      /**
       * Returns the chart's title.
       */
      public function getTitle() {
            return "Top 100 Companies, by Number of Employees, in
".$this->country;
      }

      /**
       * Returns the chart's Subject.
       */
      public function getSubject() {
            return 'Companies';
      }

      /**
       * Returns the chart's Subject Page.
       */
      public function getSubjectPage() {
            return '<a href="../companies/">Companies</a>';
      }

      /**
       * Returns the chart's X axis name.
       */
      public function getXaxisName(){
            return "Company";
      }

      /**
       * Returns the chart's Y axis name.
       */
      public function getYaxisName(){
            return "Total";
      }

      /**
       * Prints the array in JSON format.
       */
      public function printJSON() {
            echo json_encode($this->array);
      }

      /**
       * Prints the array in a HTML table format.
       */
      public function printTable() {
            for($i = 0; $i < sizeof ( $this->array ); $i ++) {
                  $company = $this->array[$i]["Company"];
                  $total = $this->array[$i]["Total"];
                  echo '
                  <tr>
                        <td></td>
```

```
                        <td>'.($i+1).'</td>
                        <td>'.$company.'</td>
                        <td>'.$total.'</td>
                  </tr>
                  ';
            }
      }
}
?>
```

CompaniesByEducation.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCPB.php');

class CompaniesByEducation extends Chart implements iCPB{
      private $country;
      private $degree;
      private $school;
      private $array = array ();

      /**
       * Class constructor. Checks required variables and
       * calls the getData() function.
       */
      public function __construct() {
            if (!isset($_GET['country']) || $_GET['country']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country' variable was not
given.";
                  die();
            }
            if (!isset($_GET['degree']) || $_GET['degree']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'degree' variable was not
given.";
                  die();
            }
            if (!isset($_GET['school'])){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'school' variable was not
given.";
                  die();
            }
            $this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
            if (!$this->isCountrySupported($this->country)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
                  die();
            }
            $this->degree = ucwords( strtolower(
mysql_real_escape_string($_GET['degree']) ) );
```

```
            $this->school = ucwords( strtolower(
mysql_real_escape_string($_GET['school']) ) );
            $this->getData ();
            if (!isset($_GET['count']))
                  $this->insertHistory();
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $degree = $this->degree;
            $sql = "SELECT
                              C.NAME AS Company,
                              COUNT(UP.USER_ID) AS Total
                        FROM ".$country_code."USERS U
                        INNER JOIN ".$country_code."USERS_POSITIONS
UP ON U.ID=UP.USER_ID
                        INNER JOIN ".$country_code."COMPANIES C ON
UP.COMPANY_ID=C.ID
                        INNER JOIN ".$country_code."EDUCATIONS E ON
U.ID=E.USER_ID
                        INNER JOIN ".$country_code."SCHOOLS S ON
E.SCHOOL_ID=S.ID
                        WHERE
                              U.LOCATION LIKE '%$country%' AND(
                                    E.DEGREE LIKE '%$degree%' OR
                                    E.MAJOR LIKE '%$degree%'
                              )";
            if ($this->school !== ""){
                  $school = $this->school;
                  $sql = $sql."AND S.NAME LIKE '%$school%'";
            }
            $sql = $sql."GROUP BY Company ORDER BY Total DESC";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "Company" => ucwords( strtolower(
mysql_real_escape_string($row['Company']) ) ),
                                    "Total" => ucwords( strtolower(
mysql_real_escape_string($row['Total']) ) )
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
```

```
                echo "503 Service Unavailable";
                die();
        }
    }

    /**
     * Returns the sample size of the current
     * graph.
     */
    public function getSampleSize() {
        include '../../_conn.php';
        $total = 0;
        $sql = "";
        $country = $this->country;
        $country_code = $this->getCountryCode($country);
        $sql = "SELECT COUNT(*) AS TOTAL
                    FROM ".$country_code."USERS U
                    WHERE LOCATION LIKE '%$country%'";
        if ($conn) {
            mysqli_set_charset ( $conn, "utf8" );
            $result = mysqli_query ( $conn, $sql );
            if (mysqli_num_rows ( $result ) > 0) {
                $row = mysqli_fetch_assoc ( $result );
                $total = $row['TOTAL'];
            }
            mysqli_close($conn);
        }else{
            header('HTTP/1.1 503 Service Unavailable');
            echo "503 Service Unavailable";
            die();
        }
        return $total;
    }

    /**
     * Returns the chart's title.
     */
    public function getTitle() {
        $str = "Companies by Number of Employees";
        if ($this->school !== ""){
            $str = $str." from ".$this->school;
        }
        $str = $str."  with a degree in ".$this->degree;
        return $str;
    }

    /**
     * Returns the chart's Subject.
     */
    public function getSubject() {
        return 'Companies';
    }

    /**
     * Returns the chart's Subject Page.
     */
    public function getSubjectPage() {
```

```
            return '<a href="../companies/">Companies</a>';
      }

      /**
       * Returns the chart's X axis name.
       */
      public function getXaxisName(){
            return "Company";
      }

      /**
       * Returns the chart's Y axis name.
       */
      public function getYaxisName(){
            return "Total";
      }

      /**
       * Prints the array in JSON format.
       */
      public function printJSON() {
            echo json_encode($this->array);
      }

      /**
       * Prints the array in a HTML table format.
       */
      public function printTable() {
            for($i = 0; $i < sizeof ( $this->array ); $i ++) {
                  $company = $this->array[$i]["Company"];
                  $total = $this->array[$i]["Total"];
                  echo '
                  <tr>
                        <td></td>
                        <td>'.($i+1).'</td>
                        <td>'.$company.'</td>
                        <td>'.$total.'</td>
                  </tr>
                  ';
            }
      }
}
?>
```

TopSchools.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCPB.php');

class TopSchools extends Chart implements iCPB{
      private $country;
      private $array = array ();

      /**
```

```
       * Class constructor. Checks required variables and
       * calls the getData() function.
       */
      public function __construct() {
            if (!isset($_GET['country']) || $_GET['country']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country' variable was not
given.";
                  die();
            }
            $this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
            if (!$this->isCountrySupported($this->country)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
                  die();
            }
            $this->getData ();
            if (!isset($_GET['count']))
                  $this->insertHistory();
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            if ($country == "Cyprus")
                  $sql = "SELECT * FROM CY_TOP_SCHOOLS LIMIT 0,
100";
            else if ($country == "Estonia"){
                  $sql = "SELECT * FROM EE_TOP_SCHOOLS LIMIT 0,
100";
            }
            else
                  $sql = "SELECT
                                    S.NAME AS School,
                                    COUNT(E.USER_ID) AS Graduates
                              FROM USERS U
                              INNER JOIN EDUCATIONS E ON
U.ID=E.USER_ID
                              INNER JOIN SCHOOLS S ON
E.SCHOOL_ID=S.ID
                              WHERE
                                    U.LOCATION LIKE '%$country%' AND
                                    E.START_YEAR < E.END_YEAR
                              GROUP BY School
                              ORDER BY Graduates DESC
                              LIMIT 0, 100";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
```

```
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "School" => ucwords( strtolower(
mysql_real_escape_string($row['School']) ) ),
                                    "Graduates" => ucwords(
strtolower( mysql_real_escape_string($row['Graduates']) ) )
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
       * Returns the sample size of the current
       * graph.
       */
      public function getSampleSize() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code."USERS U
                        WHERE LOCATION LIKE '%$country%'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        $row = mysqli_fetch_assoc ( $result );
                        $total = $row['TOTAL'];
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
            return $total;
      }

      /**
       * Returns the chart's title.
       */
      public function getTitle() {
            return "Top 100 Schools by Number of Graduates in
".$this->country;
      }
```

```
	/**
	 * Returns the chart's Subject.
	 */
	public function getSubject() {
		return 'Education';
	}

	/**
	 * Returns the chart's Subject Page.
	 */
	public function getSubjectPage() {
		return '<a href="../education/">Education</a>';
	}

	/**
	 * Returns the chart's X axis name.
	 */
	public function getXaxisName(){
		return "School";
	}

	/**
	 * Returns the chart's Y axis name.
	 */
	public function getYaxisName(){
		return "Graduates";
	}

	/**
	 * Prints the array in JSON format.
	 */
	public function printJSON() {
		echo json_encode($this->array);
	}

	/**
	 * Prints the array in a HTML table format.
	 */
	public function printTable() {
		for($i = 0; $i < sizeof ( $this->array ); $i ++) {
			$school = $this->array[$i]["School"];
			$grads = $this->array[$i]["Graduates"];
			echo '
			<tr>
				<td></td>
				<td>'.($i+1).'</td>
				<td>'.$school.'</td>
				<td>'.$grads.'</td>
			</tr>
			';
		}
	}
}
?>
```

SchoolsByDegreeLevel.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCPB.php');

class SchoolsByDegreeLevel extends Chart implements iCPB{
	private $country;
	private $degree_level;
	private $array = array ();

	/**
	 * Class constructor. Checks required variables and
	 * calls the getData() function.
	 */
	public function __construct() {
		if (!isset($_GET['country']) || $_GET['country']==""){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: 'country' variable was not
given.";
			die();
		}
		$this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
		if (!$this->isCountrySupported($this->country)){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
			die();
		}
		$this->degree_level = ucwords( strtolower(
mysql_real_escape_string($_GET['degree_level']) ) );
		if ($this->degree_level!="Associate" && $this-
>degree_level!="Bachelor" &&
			$this->degree_level!="Master" && $this-
>degree_level!="Doctoral"){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: ".$this->degree_level ." is
<b>not</b> a supported degree level.";
			die();
		}
		$this->getData ();
		if (!isset($_GET['count']))
			$this->insertHistory();
	}

	/**
	 * Communicates with database to return
	 * the appropriate data needed.
	 */
	public function getData() {
		include '../../_conn.php';
		$sql = "";
		$country = $this->country;
		$degree_level = $this->degree_level;
		$country_code = $this->getCountryCode($country);
```

```
            $sql = "SELECT COUNT(T.USER_ID) AS Graduates, T.NAME AS
School
                        FROM(
                              SELECT E.ID, E.USER_ID, E.DEGREE,
E.MAJOR, S.NAME
                              FROM ".$country_code."USERS U
                              INNER JOIN ".$country_code."EDUCATIONS
E ON U.ID=E.USER_ID ";
            switch($degree_level){
                  case("Associate"):  $sql = $sql."LEFT JOIN
SYNONYMS_ASSOCIATE_DEGREE SYN ON "; break;
                  case("Bachelor"):  $sql = $sql."LEFT JOIN
SYNONYMS_BACHELOR_DEGREE SYN ON "; break;
                  case("Master"):  $sql = $sql."LEFT JOIN
SYNONYMS_MASTER_DEGREE SYN ON "; break;
                  case("Doctoral"):  $sql = $sql."LEFT JOIN
SYNONYMS_DOCTORATE_DEGREE SYN ON "; break;
            }
            $sql = $sql."     E.DEGREE LIKE SYN.SYNONYM_LEVEL OR
                              E.MAJOR LIKE SYN.SYNONYM_LEVEL
                              INNER JOIN ".$country_code."SCHOOLS S
ON E.SCHOOL_ID=S.ID
                              WHERE
                                    U.LOCATION LIKE '%$country%' AND
                                    E.START_YEAR!='0' AND
                                    E.END_YEAR!='0' AND
                                    SYN.PRIMARY_LEVEL IS NOT NULL
                              GROUP BY E.ID
                        ) T
                        GROUP BY School
                        ORDER BY Graduates DESC
                        LIMIT 0,100
                        ";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "School" => ucwords( strtolower(
mysql_real_escape_string($row['School']) ) ),
                                    "Graduates" => ucwords(
strtolower( mysql_real_escape_string($row['Graduates']) ) )
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
```

```
       * Returns the sample size of the current
       * graph.
       */
      public function getSampleSize() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code."USERS U
                        WHERE LOCATION LIKE '%$country%'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        $row = mysqli_fetch_assoc ( $result );
                        $total = $row['TOTAL'];
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
            return $total;
      }

      /**
       * Returns the chart's title.
       */
      public function getTitle() {
            return "Top 100 schools with graduates from ".$this-
>country." with a ".$this->degree_level."'s degree";
      }

      /**
       * Returns the chart's Subject.
       */
      public function getSubject() {
            return 'Education';
      }

      /**
       * Returns the chart's Subject Page.
       */
      public function getSubjectPage() {
            return '<a href="../education/">Education</a>';
      }

      /**
       * Returns the chart's X axis name.
       */
      public function getXaxisName(){
            return "School";
      }
```

```
	/**
	 * Returns the chart's Y axis name.
	 */
	public function getYaxisName(){
		return "Graduates";
	}

	/**
	 * Prints the array in JSON format.
	 */
	public function printJSON() {
		echo json_encode($this->array);
	}

	/**
	 * Prints the array in a HTML table format.
	 */
	public function printTable() {
		for($i = 0; $i < sizeof ( $this->array ); $i ++) {
			$school = $this->array[$i]["School"];
			$grads = $this->array[$i]["Graduates"];
			echo '
			<tr>
				<td></td>
				<td>'.($i+1).'</td>
				<td>'.$school.'</td>
				<td>'.$grads.'</td>
			</tr>
			';
		}
	}
}
?>
```

GraduatesByEducation.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iTimeline.php');

class GraduatesByEducation extends Chart implements iTimeline{
	private $country;
	private $degree;
	private $school;
	private $array = array ();

	/**
	 * Class constructor. Checks required variables and
	 * calls the getData() function.
	 */
	public function __construct() {
		if (!isset($_GET['country']) || $_GET['country']==""){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: 'country' variable was not
given.";
```

```
                die();
            }
            if (!isset($_GET['degree']) || $_GET['degree']==""){
                header('HTTP/1.1 400 Bad Request');
                echo "400 Bad Request: 'degree' variable was not
given.";
                die();
            }
            $this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
            $this->degree = ucwords( strtolower(
mysql_real_escape_string($_GET['degree']) ) );
            if (!$this->isCountrySupported($this->country)){
                header('HTTP/1.1 400 Bad Request');
                echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
                die();
            }
            if (isset($_GET['school']) && $_GET['school']!=""){
                $this->school = ucwords( strtolower(
mysql_real_escape_string($_GET['school']) ) );
            }
            $this->getData ();
            if (!isset($_GET['count']))
                $this->insertHistory();
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $degree = $this->degree;
            $school = $this->school;
            $sql = "SELECT
                              E.END_YEAR AS GRADUATE_YEAR,
                              COUNT( E.USER_ID ) AS GRADUATES
                        FROM ".$country_code."USERS U
                        INNER JOIN ".$country_code."EDUCATIONS E ON
U.ID=E.USER_ID ";
            if ($this->school!="")
                $sql = $sql."INNER JOIN ".$country_code."SCHOOLS S
ON E.SCHOOL_ID = S.ID ";
            $sql = $sql."WHERE
                              U.LOCATION LIKE '%$country%' AND
                              E.END_YEAR!='0' AND ";
            if ($this->school!="")
                $sql = $sql."S.NAME LIKE '%$school%' AND ";
            $sql = $sql."(
                                    E.DEGREE LIKE '%$degree%' OR
                                    E.MAJOR LIKE '%$degree%'
                              )
                        GROUP BY GRADUATE_YEAR
                        ORDER BY GRADUATE_YEAR ASC";
```

```
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $row = array (
                                    "Year" => $row['GRADUATE_YEAR'],
                                    "Graduates" => $row['GRADUATES']
                              );
                              array_push( $this->array, $row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
       * Returns the sample size of the current
       * graph.
       */
      public function getSampleSize() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code."USERS U
                        WHERE LOCATION LIKE '%$country%'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        $row = mysqli_fetch_assoc ( $result );
                        $total = $row['TOTAL'];
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
            return $total;
      }

      /**
       * Returns the chart's title.
       */
      public function getTitle() {
            $title = "Number of graduates in ".$this->degree;
            if ($this->school!="")
                  $title = $title.", from ".$this->school;
```

```
            return $title;
      }

      /**
       * Returns the chart's Subject.
       */
      public function getSubject() {
            return 'Education';
      }

      /**
       * Returns the chart's Subject Page.
       */
      public function getSubjectPage() {
            return '<a href="../education/">Education</a>';
      }

      /**
       * Returns the chart's X axis name.
       */
      public function getXaxisName(){
            return "Year";
      }

      /**
       * Returns the chart's Y axis name.
       */
      public function getYaxisName(){
            return "Graduates";
      }

      /**
       * Returns the chart's Date Format for timeline graph.
       */
      public function getDateFormat(){
            return "YYYY";
      }

      /**
       * Prints the array in JSON format.
       */
      public function printJSON() {
            echo json_encode($this->array);
      }

      /**
       * Prints the array in a HTML table format.
       */
      public function printTable() {
            for($i = 0; $i < sizeof ( $this->array ); $i ++) {
                  $year = $this->array[$i]["Year"];
                  $total = $this->array[$i]["Graduates"];
                  echo '
                  <tr>
                        <td>'.($i+1).'</td>
                        <td>'.$year.'</td>
                        <td>'.$total.'</td>
```

```
                    </tr>
                    ';
              }
        }
}
?>
```

GradsByDegree.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCPB.php');

class GradsByDegree extends Chart implements iCPB{
      private $country;
      private $array = array ();

      /**
       * Class constructor. Checks required variables and
       * calls the getData() function.
       */
      public function __construct() {
            if (!isset($_GET['country']) || $_GET['country']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country' variable was not
given.";
                  die();
            }
            $this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
            if (!$this->isCountrySupported($this->country)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
                  die();
            }
            $this->getData ();
            if (!isset($_GET['count']))
                  $this->insertHistory();
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            if ($country == "Cyprus")
                  $sql = "SELECT * FROM CY_GRADS_BY_DEGREE";
            else if ($country == "Estonia"){
                  $sql = "SELECT * FROM EE_GRADS_BY_DEGREE";
            }
```

```
            else
                  $sql = "-- Associate
                              SELECT 'Associate' AS DEGREE_LEVEL,
COUNT(DISTINCT E.USER_ID) AS TOTAL
                              FROM ".$country_code."USERS U
                              INNER JOIN ".$country_code."EDUCATIONS
E ON U.ID=E.USER_ID
                              LEFT JOIN SYNONYMS_ASSOCIATE_DEGREE
SAD ON
                                          E.DEGREE LIKE
SAD.SYNONYM_LEVEL OR
                                          E.MAJOR LIKE
SAD.SYNONYM_LEVEL
                              WHERE
                                    U.LOCATION LIKE '%$country%' AND
                                    E.START_YEAR!='0' AND
                                    E.END_YEAR!='0' AND
                                    SAD.PRIMARY_LEVEL = 'Associate'
                              UNION
                              -- Bachelor
                              SELECT 'Bachelor' AS DEGREE_LEVEL,
COUNT(DISTINCT E.USER_ID) AS TOTAL
                              FROM ".$country_code."USERS U
                              INNER JOIN ".$country_code."EDUCATIONS
E ON U.ID=E.USER_ID
                              LEFT JOIN SYNONYMS_BACHELOR_DEGREE SBD
ON
                                          E.DEGREE LIKE
SBD.SYNONYM_LEVEL OR
                                          E.MAJOR LIKE
SBD.SYNONYM_LEVEL
                              WHERE
                                    U.LOCATION LIKE '%$country%' AND
                                    E.START_YEAR!='0' AND
                                    E.END_YEAR!='0' AND
                                    SBD.PRIMARY_LEVEL = 'Bachelor'
                              UNION
                              -- Master
                              SELECT 'Master' AS DEGREE_LEVEL,
COUNT(DISTINCT E.USER_ID) AS TOTAL
                              FROM ".$country_code."USERS U
                              INNER JOIN ".$country_code."EDUCATIONS
E ON U.ID=E.USER_ID
                              LEFT JOIN SYNONYMS_MASTER_DEGREE SMD
ON
                                          E.DEGREE LIKE
SMD.SYNONYM_LEVEL OR
                                          E.MAJOR LIKE
SMD.SYNONYM_LEVEL
                              WHERE
                                    U.LOCATION LIKE '%$country%' AND
                                    E.START_YEAR!='0' AND
                                    E.END_YEAR!='0' AND
                                    SMD.PRIMARY_LEVEL = 'Master'
                              UNION
                              -- Doctorate
```

```
                              SELECT 'Doctor' AS DEGREE_LEVEL,
COUNT(DISTINCT E.USER_ID) AS TOTAL
                              FROM ".$country_code."USERS U
                              INNER JOIN ".$country_code."EDUCATIONS
E ON U.ID=E.USER_ID
                              LEFT JOIN SYNONYMS_DOCTORATE_DEGREE
SDD ON
                                          E.DEGREE LIKE
SDD.SYNONYM_LEVEL OR
                                          E.MAJOR LIKE
SDD.SYNONYM_LEVEL
                              WHERE
                                    U.LOCATION LIKE '%$country%' AND
                                    E.START_YEAR!='0' AND
                                    E.END_YEAR!='0' AND
                                    SDD.PRIMARY_LEVEL = 'Doctor'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "Degree" => ucwords( strtolower(
mysql_real_escape_string($row['DEGREE_LEVEL']) ) ),
                                    "Graduates" => $row['TOTAL']
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
       * Returns the sample size of the current
       * graph.
       */
      public function getSampleSize() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code."USERS U
                        WHERE LOCATION LIKE '%$country%'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        $row = mysqli_fetch_assoc ( $result );
                        $total = $row['TOTAL'];
```

```
                }
                mysqli_close($conn);
        }else{
                header('HTTP/1.1 503 Service Unavailable');
                echo "503 Service Unavailable";
                die();
        }
        return $total;
}

/**
 * Returns the chart's title.
 */
public function getTitle() {
        return "Graduates by degree levels in ".$this->country;
}

/**
 * Returns the chart's Subject.
 */
public function getSubject() {
        return 'Education';
}

/**
 * Returns the chart's Subject Page.
 */
public function getSubjectPage() {
        return '<a href="../education/">Education</a>';
}

/**
 * Returns the chart's X axis name.
 */
public function getXaxisName(){
        return "Degree";
}

/**
 * Returns the chart's Y axis name.
 */
public function getYaxisName(){
        return "Graduates";
}

/**
 * Prints the array in JSON format.
 */
public function printJSON() {
        echo json_encode($this->array);
}

/**
 * Prints the array in a HTML table format.
 */
public function printTable() {
        for($i = 0; $i < sizeof ( $this->array ); $i ++) {
```

```
                  $degree = $this->array[$i]["Degree"];
                  $grads = $this->array[$i]["Graduates"];
                  echo '
                  <tr>
                        <td></td>
                        <td>'.($i+1).'</td>
                        <td>'.$degree.'</td>
                        <td>'.$grads.'</td>
                  </tr>
                  ';
            }
      }
}
?>
```

CompareGradsByDegree.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCBCompare.php');

class CompareGradsByDegree extends Chart implements iCBCompare{
      private $country1;
      private $country2;
      private $array = array ();

      /**
       * Class constructor. Checks required variables and
       * calls the getData() function.
       */
      public function __construct() {
            if (!isset($_GET['country1']) || $_GET['country1']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country1' variable was not
given.";
                  die();
            }
            if (!isset($_GET['country2']) || $_GET['country2']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country2' variable was not
given.";
                  die();
            }
            $this->country1 = ucwords( strtolower(
mysql_real_escape_string($_GET['country1']) ) );
            $this->country2 = ucwords( strtolower(
mysql_real_escape_string($_GET['country2']) ) );
            if (!$this->isCountrySupported($this->country1)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country1 ." is
<b>not</b> a supported country.";
                  die();
            }
            if (!$this->isCountrySupported($this->country2)){
                  header('HTTP/1.1 400 Bad Request');
```

```
                  echo "400 Bad Request: ".$this->country2 ." is
<b>not</b> a supported country.";
                  die();
            }
            $this->getData();
            if (!isset($_GET['count']))
                  $this->insertHistory();
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $sql = "";
            $country1 = $this->country1;
            $country2 = $this->country2;
            $country_code1 = $this->getCountryCode($country1);
            $country_code2 = $this->getCountryCode($country2);
            $sql = "SELECT T1.DEGREE_LEVEL, T1.TOTAL AS TOTAL1,
IFNULL(T2.TOTAL, '0') AS TOTAL2
                        FROM (
                              SELECT * FROM
".$country_code1."GRADS_BY_DEGREE
                        )T1
                        LEFT JOIN ".$country_code2."GRADS_BY_DEGREE
T2 ON T1.DEGREE_LEVEL=T2.DEGREE_LEVEL
                        ORDER BY TOTAL1 DESC
                        ";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "Degree" => ucwords( strtolower(
mysql_real_escape_string($row['DEGREE_LEVEL']) ) ),
                                    "Total1" => $row['TOTAL1'],
                                    "Total2" => $row['TOTAL2']
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
       * Returns the sample size for the first
       * country of the current graph.
       */
```

```
	public function getSampleSize1() {
		include '../../_conn.php';
		$total = 0;
		$sql = "";
		$country1 = $this->country1;
		$country_code1 = $this->getCountryCode($country1);
		$sql = "SELECT COUNT(*) AS TOTAL
				FROM ".$country_code1."USERS U
				WHERE LOCATION LIKE '%$country1%'";
		if ($conn) {
			mysqli_set_charset ( $conn, "utf8" );
			$result = mysqli_query ( $conn, $sql );
			if (mysqli_num_rows ( $result ) > 0) {
				$row = mysqli_fetch_assoc ( $result );
				$total = $row['TOTAL'];
			}
			mysqli_close($conn);
		}else{
			header('HTTP/1.1 503 Service Unavailable');
			echo "503 Service Unavailable";
			die();
		}
		return $total;
	}

	/**
	 * Returns the sample size for the second
	 * country of the current graph.
	 */
	public function getSampleSize2() {
		include '../../_conn.php';
		$total = 0;
		$sql = "";
		$country2 = $this->country2;
		$country_code2 = $this->getCountryCode($country2);
		$sql = "SELECT COUNT(*) AS TOTAL
				FROM ".$country_code2."USERS U
				WHERE LOCATION LIKE '%$country2%'";
		if ($conn) {
			mysqli_set_charset ( $conn, "utf8" );
			$result = mysqli_query ( $conn, $sql );
			if (mysqli_num_rows ( $result ) > 0) {
				$row = mysqli_fetch_assoc ( $result );
				$total = $row['TOTAL'];
			}
			mysqli_close($conn);
		}else{
			header('HTTP/1.1 503 Service Unavailable');
			echo "503 Service Unavailable";
			die();
		}
		return $total;
	}

	/**
	 * Returns the chart's title.
	 */
```

```
	public function getTitle() {
		return "Graduates by degree, comparison between ".$this-
>country1 ." and ".$this->country2;
	}

	/**
	 * Returns the chart's Subject.
	 */
	public function getSubject() {
		return 'Education';
	}

	/**
	 * Returns the chart's Subject Page.
	 */
	public function getSubjectPage() {
		return '<a href="../education/">Education</a>';
	}

	/**
	 * Returns the chart's X axis name.
	 */
	public function getXaxisName(){
		return "Degree";
	}

	/**
	 * Returns the name of the first country.
	 */
	public function getCountry1(){
		return $this->country1;
	}

	/**
	 * Returns the name of the second country.
	 */
	public function getCountry2(){
		return $this->country2;
	}

	/**
	 * Prints the array in JSON format.
	 */
	public function printJSON() {
		echo json_encode($this->array);
	}

	/**
	 * Prints the array in a HTML table format.
	 */
	public function printTable() {
		for($i = 0; $i < sizeof ( $this->array ); $i ++) {
			$degree = $this->array[$i]["Degree"];
			$total1 = $this->array[$i]["Total1"];
			$total2 = $this->array[$i]["Total2"];
			echo '
			<tr>
```

```
                        <td></td>
                        <td>'.($i+1).'</td>
                        <td>'.$degree.'</td>
                        <td>'.$total1.'</td>
                        <td>'.$total2.'</td>
                  </tr>
                  ';
            }
      }
}
?>
```

TopJobs.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCPB.php');

class TopJobs extends Chart implements iCPB{
      private $country;
      private $array = array ();

      /**
       * Class constructor. Checks required variables and
       * calls the getData() function.
       */
      public function __construct() {
            if (!isset($_GET['country']) || $_GET['country']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country' variable was not
given.";
                  die();
            }
            $this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
            if (!$this->isCountrySupported($this->country)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
                  die();
            }
            $this->getData ();
            if (!isset($_GET['count']))
                  $this->insertHistory();
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
```

```
            if ($country == "Cyprus")
                  $sql = "SELECT * FROM CY_TOP_POSITIONS LIMIT 0,
100";
            else if ($country == "Estonia")
                  $sql = "SELECT * FROM EE_TOP_POSITIONS LIMIT 0,
100";
            else
                  $sql = "SELECT
                                    UP.TITLE AS Title,
                                    COUNT( UP.USER_ID ) AS Total
                              FROM ".$country_code."USERS U
                              INNER JOIN
".$country_code."USERS_POSITIONS UP ON U.ID=UP.USER_ID
                              WHERE
                                    U.LOCATION LIKE '%$country%'
                              GROUP BY Title
                              ORDER BY Total DESC
                              LIMIT 0, 100";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "Title" => ucwords( strtolower(
mysql_real_escape_string($row['Title']) ) ),
                                    "Total" => ucwords( strtolower(
mysql_real_escape_string($row['Total']) ) )
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
       * Returns the sample size of the current
       * graph.
       */
      public function getSampleSize() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code."USERS U
                        WHERE LOCATION LIKE '%$country%'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
```

```
                if (mysqli_num_rows ( $result ) > 0) {
                      $row = mysqli_fetch_assoc ( $result );
                      $total = $row['TOTAL'];
                }
                mysqli_close($conn);
          }else{
                header('HTTP/1.1 503 Service Unavailable');
                echo "503 Service Unavailable";
                die();
          }
          return $total;
    }

    /**
     * Returns the chart's title.
     */
    public function getTitle() {
          return "Top 100 Job Positions in ".$this->country;
    }

    /**
     * Returns the chart's Subject.
     */
    public function getSubject() {
          return 'Employment';
    }

    /**
     * Returns the chart's Subject Page.
     */
    public function getSubjectPage() {
          return '<a href="../employment/">Employment</a>';
    }

    /**
     * Returns the chart's X axis name.
     */
    public function getXaxisName(){
          return "Title";
    }

    /**
     * Returns the chart's Y axis name.
     */
    public function getYaxisName(){
          return "Total";
    }

    /**
     * Prints the array in JSON format.
     */
    public function printJSON() {
          echo json_encode($this->array);
    }

    /**
     * Prints the array in a HTML table format.
```

```
       */
      public function printTable() {
            for($i = 0; $i < sizeof ( $this->array ); $i ++) {
                  $title = $this->array[$i]["Title"];
                  $total = $this->array[$i]["Total"];
                  echo '
                  <tr>
                        <td></td>
                        <td>'.($i+1).'</td>
                        <td>'.$title.'</td>
                        <td>'.$total.'</td>
                  </tr>
                  ';
            }
      }
}
?>
```

Unemployment.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iTimeline.php');

class Unemployment extends Chart implements iTimeline{
      private $country;
      private $unemployed = array ();
      private $employed = array ();
      private $unemployment = array ();

      /**
       * Class constructor. Checks required variables and
       * calls the getData() function.
       */
      public function __construct() {
            if (!isset($_GET['country']) || $_GET['country']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country' variable was not
given.";
                  die();
            }
            $this->country = ucwords(strtolower($_GET['country']));
            if (!$this->isCountrySupported($this->country)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
                  die();
            }
            $this->getData();
            if (!isset($_GET['count']))
                  $this->insertHistory();
      }

      /**
       *
```

```
       */
      private function addUnemploymentDates($fromDate, $toDate) {
            // Set timezone
            date_default_timezone_set('UTC');

            // Fix unknown month/day
            $fromDate = str_replace("-00","-01",$fromDate);
            $toDate = str_replace("-00-","-12-",$toDate);
            $toDate = str_replace("-00","-01",$toDate);

            while (strtotime($fromDate) <= strtotime($toDate)) {
                  if (array_key_exists($fromDate,$this->unemployed)
){
                        $this->unemployed[$fromDate]++;
                  }else{
                        $this->unemployed[$fromDate]=1;
                  }
                  $fromDate = date("Y-m-d", strtotime("+1 month",
strtotime($fromDate)));
            }
      }

      /**
       *
       */
      private function addEmploymentDates($fromDate, $toDate) {
            // Set timezone
            date_default_timezone_set('UTC');

            // Fix unknown month/day
            $fromDate = str_replace("-00","-01",$fromDate);
            $toDate = str_replace("-00-","-12-",$toDate);
            $toDate = str_replace("-00","-01",$toDate);

            while (strtotime($fromDate) <= strtotime($toDate)) {
                  if (array_key_exists($fromDate, $this->employed)
){
                        $this->employed[$fromDate]++;
                  }else{
                        $this->employed[$fromDate]=1;
                  }
                  $fromDate = date("Y-m-d", strtotime("+1 month",
strtotime($fromDate)));
            }
      }

      /**
       *
       */
      private function getUnemployment() {
            foreach ($this->unemployed as $date => $unemployees) {
                  if (array_key_exists($date, $this->employed) ){
                        $workforce = $this->employed[$date] +
$unemployees;
                        $this->unemployment[$date] = round(
($unemployees / $workforce) * 100 );
                  }
```

```
            }
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT DISTINCT
                        UP.USER_ID AS UID,
                        CONCAT(
                              LPAD(UP.START_YEAR,4,'0000'), '-',
                              LPAD(UP.START_MONTH,2,'00'), '-',
                              LPAD(UP.START_DAY,2,'00')
                        ) AS STARTED,
                        CONCAT(
                              LPAD(UP.END_YEAR,4,'0000'), '-',
                              LPAD(UP.END_MONTH,2,'00'), '-',
                              LPAD(UP.END_DAY,2,'00')
                        ) AS ENDED,
                        UP.TITLE
                  FROM ".$country_code."USERS U
                  INNER JOIN ".$country_code."USERS_POSITIONS UP ON
U.ID=UP.USER_ID
                  WHERE
                        U.LOCATION LIKE '%$country%' AND
                        UP.START_YEAR!='0' AND
                        UP.END_YEAR!='0'
                  ORDER BY UID ASC, STARTED ASC";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        ini_set('max_execution_time', 300); // max
execution time = 5min
                        $t_uid = 0;
                        $unemp_started ='';
                        $unemp_ended ='';
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $uid = $row ['UID'];
                              if ($uid != $t_uid){
                                    $this->addEmploymentDates($row
['STARTED'], $row ['ENDED']);
                                    $unemp_started = $row['ENDED'];
                                    $t_uid = $uid;
                                    continue;
                              }
                              $jobTitle = strtolower($row['TITLE']);
                              if ( (
                                          (strpos($jobTitle,
'unemployed') !== false) ||
                                          (strpos($jobTitle, 'no
job') !== false) ||
```

```
                                          (strpos($jobTitle, 'job
seeking') !== false) ||
                                          (strpos($jobTitle,
'anergos') !== false)
                                    )
                              ){
                                    $unemp_ended = $row['ENDED'];
                              }else{
                                    $unemp_ended = $row['STARTED'];
                                    $this->addEmploymentDates($row
['STARTED'], $row ['ENDED']);
                              }
                              $this-
>addUnemploymentDates($unemp_started, $unemp_ended);
                              $unemp_started = $row ['ENDED'];
                        }
                        ksort($this->unemployed);
                        ksort($this->employed);
                        $this->getUnemployment();
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
       * Returns the sample size of the current
       * graph.
       */
      public function getSampleSize() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code."USERS U
                        WHERE LOCATION LIKE '%$country%'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        $row = mysqli_fetch_assoc ( $result );
                        $total = $row['TOTAL'];
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
            return $total;
      }
```

```
	/**
	 * Returns the chart's title.
	 */
	public function getTitle() {
		return 'Unemployment in '.$this->country;
	}

	/**
	 * Returns the chart's Subject.
	 */
	public function getSubject() {
		return 'Employment';
	}

	/**
	 * Returns the chart's Subject Page.
	 */
	public function getSubjectPage() {
		return '<a href="../employment/">Employment</a>';
	}

	/**
	 * Returns the chart's X axis name.
	 */
	public function getXaxisName(){
		return "Date";
	}

	/**
	 * Returns the chart's Y axis name.
	 */
	public function getYaxisName(){
		return "Unemployment (%)";
	}

	/**
	 * Returns the chart's Date Format for timeline graph.
	 */
	public function getDateFormat(){
		return "YYYY-MM";
	}

	/**
	 * Prints the array in JSON format.
	 */
	public function printJSON() {
		//echo json_encode($this->unemployment);
		$str = "[";
		foreach($this->unemployment as $date => $unemployees){
			$str = $str.'{"Date": "'.$date.'", "Unemployment
(%)": '.$unemployees.'}, ';
		}
		$str = substr($str, 0, -2);
		$str = $str."]";
		echo $str;
	}
```

```
	/**
	 * Prints the array in a HTML table format.
	 */
	public function printTable() {
		$i=1;
		foreach ($this->unemployment as $date => $unemployees) {
			echo '
			<tr>
				<td>'.($i).'</td>
				<td>'.$date.'</td>
				<td>'.$unemployees.'</td>
			</tr>
			';
			$i++;
		}
	}
}
?>
```

JobsByEducation.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCPB.php');

class JobsByEducation extends Chart implements iCPB{
	private $country;
	private $degree;
	private $school;
	private $period;
	private $array = array ();

	/**
	 * Class constructor. Checks required variables and
	 * calls the getData() function.
	 */
	public function __construct() {
		if (!isset($_GET['country']) || $_GET['country']==""){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: 'country' variable was not
given.";
			die();
		}
		if (!isset($_GET['degree']) || $_GET['degree']==""){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: 'degree' variable was not
given.";
			die();
		}
		if (!isset($_GET['period']) || $_GET['period']==""){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: 'period' variable was not
given.";
			die();
		}
```

```
            $this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
            if (!$this->isCountrySupported($this->country)){
                  header('HTTP/1.1 400 Bad Request');
                  echo ".$this->country." is
<b>not</b> a supported country.";
                  die();
            }
            $this->degree = ucwords( strtolower(
mysql_real_escape_string($_GET['degree']) ) );
            if (isset($_GET['school']) && $_GET['school']!=""){
                  $this->school = ucwords( strtolower(
mysql_real_escape_string($_GET['school']) ) );
            }
            $this->period =
mysql_real_escape_string($_GET['period']);
            $this->getData ();
            if (!isset($_GET['count']))
                  $this->insertHistory();
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $country = $this->country;
            $degree = $this->degree;
            $period = $this->period;
            $country_code = $this->getCountryCode($country);
            $sql="";
            if ($period=="latest"){
                  $sql = "SELECT T2.Title, COUNT(T2.UID) AS Total
                              FROM (
                                    SELECT DISTINCT
                                          UP.USER_ID AS UID,
                                          MAX(CONCAT(
                                                UP.START_YEAR, '-',

      LPAD(UP.START_MONTH,2,'00'), '-',

      LPAD(UP.START_DAY,2,'00')
                                          )) AS STARTED
                                    FROM
".$country_code."USERS_POSITIONS UP
                                    GROUP BY UID
                                    ORDER BY UID ASC, STARTED ASC
                              ) T1
                              INNER JOIN (
                                    SELECT DISTINCT
                                    UP.USER_ID AS UID,
                                    CONCAT(
                                          UP.START_YEAR, '-',

      LPAD(UP.START_MONTH,2,'00'), '-',
                                          LPAD(UP.START_DAY,2,'00')
```

```
                                    ) AS STARTED,
                                    UP.TITLE AS Title
                                    FROM ".$country_code."USERS U
                                    INNER JOIN
".$country_code."USERS_POSITIONS UP ON U.ID=UP.USER_ID
                                    INNER JOIN
".$country_code."EDUCATIONS E ON U.ID=E.USER_ID ";
                                    if ($this->school!="")
                                          $sql = $sql."INNER JOIN
".$country_code."SCHOOLS S ON E.SCHOOL_ID=S.ID ";
                                    $sql = $sql."
                                    WHERE
                                          U.LOCATION LIKE
'%$country%' AND
                                          E.START_YEAR < E.END_YEAR
AND ";
                                    if ($this->school!="")
                                          $sql = $sql."S.NAME LIKE
'%".$this->school."%' AND ";
                                    $sql = $sql."(
                                                E.DEGREE LIKE
'%$degree%' OR
                                                E.MAJOR LIKE
'%$degree%'
                                          )
                                    ORDER BY UID ASC, STARTED ASC
                              ) T2 ON T1.UID=T2.UID AND
T1.STARTED=T2.STARTED
                              GROUP BY T2.Title
                              ORDER BY Total DESC";
            }else if ($period=="first"){
                  $sql = "SELECT T2.Title, COUNT(T2.UID) AS Total
                              FROM (
                                    SELECT DISTINCT
                                          UP.USER_ID AS UID,
                                          MIN(CONCAT(
                                                UP.START_YEAR, '-',

      LPAD(UP.START_MONTH,2,'00'), '-',

      LPAD(UP.START_DAY,2,'00')
                                          )) AS STARTED
                                    FROM
".$country_code."USERS_POSITIONS UP
                                    GROUP BY UID
                                    ORDER BY UID ASC, STARTED ASC
                              ) T1
                              INNER JOIN (
                                    SELECT DISTINCT
                                    UP.USER_ID AS UID,
                                    CONCAT(
                                          UP.START_YEAR, '-',

      LPAD(UP.START_MONTH,2,'00'), '-',
                                          LPAD(UP.START_DAY,2,'00')
                                    ) AS STARTED,
                                    UP.TITLE AS Title
```

```
                                    FROM ".$country_code."USERS U
                                    INNER JOIN
".$country_code."USERS_POSITIONS UP ON U.ID=UP.USER_ID
                                    INNER JOIN
".$country_code."EDUCATIONS E ON U.ID=E.USER_ID ";
                                    if ($this->school!="")
                                          $sql = $sql."INNER JOIN
".$country_code."SCHOOLS S ON E.SCHOOL_ID=S.ID ";
                                    $sql = $sql."
                                    WHERE
                                          U.LOCATION LIKE
'%$country%' AND
                                          E.START_YEAR < E.END_YEAR
AND ";
                                    if ($this->school!="")
                                          $sql = $sql."S.NAME LIKE
'%".$this->school."%' AND ";
                                    $sql = $sql."(
                                                E.DEGREE LIKE
'%$degree%' OR
                                                E.MAJOR LIKE
'%$degree%'
                                          )
                                    ORDER BY UID ASC, STARTED ASC
                              ) T2 ON T1.UID=T2.UID AND
T1.STARTED=T2.STARTED
                              GROUP BY T2.Title
                              ORDER BY Total DESC";
            }else{ // $period=="all"
                  $sql = "SELECT
                              UP.TITLE AS Title,
                              COUNT( UP.USER_ID ) AS Total
                        FROM ".$country_code."USERS U
                        INNER JOIN ".$country_code."USERS_POSITIONS
UP ON U.ID=UP.USER_ID
                        INNER JOIN ".$country_code."EDUCATIONS E ON
U.ID=E.USER_ID ";
                        if ($this->school!="")
                              $sql = $sql."INNER JOIN
".$country_code."SCHOOLS S ON E.SCHOOL_ID=S.ID ";
                        $sql = $sql."
                        WHERE
                              U.LOCATION LIKE '%$country%' AND
                              E.START_YEAR < E.END_YEAR AND ";
                        if ($this->school!="")
                              $sql = $sql."S.NAME LIKE '%".$this-
>school."%' AND ";
                        $sql = $sql."(
                                    E.DEGREE LIKE '%$degree%' OR
                                    E.MAJOR LIKE '%$degree%'
                              )
                        GROUP BY Title
                        ORDER BY Total DESC";
            }
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
```

```
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "Title" => ucwords( strtolower(
mysql_real_escape_string($row['Title']) ) ),
                                    "Total" => ucwords( strtolower(
mysql_real_escape_string($row['Total']) ) )
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
       * Returns the sample size of the current
       * graph.
       */
      public function getSampleSize() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code."USERS U
                        WHERE LOCATION LIKE '%$country%'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        $row = mysqli_fetch_assoc ( $result );
                        $total = $row['TOTAL'];
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
            return $total;
      }

      /**
       * Returns the chart's title.
       */
      public function getTitle() {
            return "Job Positions by Degree: ".$this->degree;
      }

      /**
```

```
	 * Returns the chart's Subject.
	 */
	public function getSubject() {
		return 'Employment';
	}

	/**
	 * Returns the chart's Subject Page.
	 */
	public function getSubjectPage() {
		return '<a href="../employment/">Employment</a>';
	}

	/**
	 * Returns the chart's X axis name.
	 */
	public function getXaxisName(){
		return "Title";
	}

	/**
	 * Returns the chart's Y axis name.
	 */
	public function getYaxisName(){
		return "Total";
	}

	/**
	 * Prints the array in JSON format.
	 */
	public function printJSON() {
		echo json_encode($this->array);
	}

	/**
	 * Prints the array in a HTML table format.
	 */
	public function printTable() {
		for($i = 0; $i < sizeof ( $this->array ); $i ++) {
			$title = $this->array[$i]["Title"];
			$total = $this->array[$i]["Total"];
			echo '
			<tr>
				<td></td>
				<td>'.($i+1).'</td>
				<td>'.$title.'</td>
				<td>'.$total.'</td>
			</tr>
			';
		}
	}
}
?>
```

CompareJobs.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCPB.php');

class CompareJobs extends Chart implements iCPB{
	private $country;
	private $array = array ();

	/**
	 * Class constructor. Checks required variables and
	 * calls the getData() function.
	 */
	public function __construct() {
		if (!isset($_GET['country']) || $_GET['country']==""){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: 'country' variable was not
given.";
			die();
		}
		if ( !isset($_GET['tags']) ){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: no job positions were given
to compare.";
			die();
		}
		$this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
		if (!$this->isCountrySupported($this->country)){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
			die();
		}
		$this->getData ();
		if (!isset($_GET['count']))
			$this->insertHistory();
	}

	/**
	 * Communicates with database to return
	 * the appropriate data needed.
	 */
	public function getData() {
		include '../../_conn.php';
		$country = $this->country;
		$country_code = $this->getCountryCode($country);
		foreach ($_GET['tags'] as $tag){
			$tag = ucwords( strtolower(
mysql_real_escape_string($tag) ) );
			$sql = "SELECT
					UP.TITLE AS Title,
					COUNT( UP.USER_ID ) AS Total
				FROM ".$country_code."USERS U
```

```
                              INNER JOIN
".$country_code."USERS_POSITIONS UP ON U.ID=UP.USER_ID
                              WHERE
                                    U.LOCATION LIKE '%$country%' AND
                                    UP.TITLE LIKE '%$tag%'
                              GROUP BY Title
                              ORDER BY Total DESC";
                  if ($conn) {
                        $total = 0;
                        mysqli_set_charset($conn, "utf8");
                        $result=mysqli_query($conn, $sql);
                        if (mysqli_num_rows($result) > 0) {
                              while($row =
mysqli_fetch_assoc($result)) {
                                    $total = $total + $row['Total'];
                              }
                              $new_row = array (
                                    "Title" => $tag,
                                    "Total" => $total
                              );
                              array_push( $this->array, $new_row );
                        }
                  }else{
                        header('HTTP/1.1 503 Service Unavailable');
                        echo "503 Service Unavailable";
                        die();
                  }
            }
            mysqli_close($conn);
      }

      /**
       * Returns the sample size of the current
       * graph.
       */
      public function getSampleSize() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code."USERS U
                        WHERE LOCATION LIKE '%$country%'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        $row = mysqli_fetch_assoc ( $result );
                        $total = $row['TOTAL'];
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
```

```
            return $total;
      }

      /**
       * Returns the chart's title.
       */
      public function getTitle() {
            return "Jobs Comparison";
      }

      /**
       * Returns the chart's Subject.
       */
      public function getSubject() {
            return 'Employment';
      }

      /**
       * Returns the chart's Subject Page.
       */
      public function getSubjectPage() {
            return '<a href="../employment/">Employment</a>';
      }

      /**
       * Returns the chart's X axis name.
       */
      public function getXaxisName(){
            return "Title";
      }

      /**
       * Returns the chart's Y axis name.
       */
      public function getYaxisName(){
            return "Total";
      }

      /**
       * Prints the array in JSON format.
       */
      public function printJSON() {
            echo json_encode($this->array);
      }

      /**
       * Prints the array in a HTML table format.
       */
      public function printTable() {
            for($i = 0; $i < sizeof ( $this->array ); $i ++) {
                  $title = $this->array[$i]["Title"];
                  $total = $this->array[$i]["Total"];
                  echo '
                  <tr>
                        <td></td>
                        <td>'.($i+1).'</td>
                        <td>'.$title.'</td>
```

```
                        <td>'.$total.'</td>
                    </tr>
                    ';
            }
      }
}
?>
```

FlowByEducation.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iBihisankey.php');

class FlowByEducation extends Chart implements iBihisankey{
      private $country;
      private $degree;
      private $school;
      private $maxHops = 0;
      private $vertices = array ();
      private $edges = array ();
      private $firstLevels = 0;

      public function __construct() {
            if (!isset($_GET['country']) || $_GET['country']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country' variable was not
given.";
                  die();
            }
            if (!isset($_GET['degree']) || $_GET['degree']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'degree' variable was not
given.";
                  die();
            }
            if (!isset($_GET['school']) || $_GET['school']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'school' variable was not
given.";
                  die();
            }
            $this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
            if (!$this->isCountrySupported($this->country)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
                  die();
            }
            $this->degree = ucwords(strtolower($_GET['degree']));
            $this->school = ucwords(strtolower($_GET['school']));
            $this->getData ();
            if (!isset($_GET['count']))
                  $this->insertHistory();
```

```
	}

	/**
	 * Checks if the given vertex name exists in the vetices
table.
	 * If so, it returns the id; if not, the new vertex is added
	 * and the id is returned.
	 *
	 * @param unknown $v Given vertex name
	 */
	private function getVertexId($v) {
		for($i = 0; $i < sizeof ( $this->vertices ); $i ++) {
			if ($this->vertices [$i]["name"] == $v) {
				return $this->vertices [$i]["id"];
			}
		}
		$type = chr(65+rand(0,25));
		$newVertex = array (
			"type" => "$type",
			"id" => $i,
			"parent" => "null",
			"name" => $v
		);
		array_push( $this->vertices, $newVertex );
		return $this->vertices[$i]["id"];
	}

	/**
	 * Adds an edge between the 'from' and 'to' vertices.
	 * @param unknown $from Source Vertex name.
	 * @param unknown $to Target Vertex name.
	 */
	private function addEdge($from, $to) {

		// Get the IDs of 'from' and 'to' vertex
		$fromId = $this->getVertexId ( $from );
		$toId = $this->getVertexId ( $to );

		// If the edge exist, +1 the value
		for($i = 0; $i < sizeof ( $this->edges ); $i ++) {
			if ($this->edges [$i] ["source"] == $fromId &&
$this->edges [$i] ["target"] == $toId) {
				$this->edges [$i] ["value"] ++;
				return;
			}
		}

		// Create new edge between source and target vertex
		$newEdge = array(
			"source" => $fromId,
			"target" => $toId,
			"value" => 1
		);
		array_push( $this->edges, $newEdge );
	}

	/**
```

```
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $country = $this->country;
            $degree = $this->degree;
            $school = $this->school;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT DISTINCT
                              U.ID AS UID,
                              CONCAT(
                                    UP.START_YEAR, '-',
                                    LPAD(UP.START_MONTH,2,'00'), '-',
                                    LPAD(UP.START_DAY,2,'00')
                              ) AS STARTED,
                              UP.TITLE AS POSITION,
                              C.NAME AS COMPANY
                        FROM ".$country_code."USERS U
                        INNER JOIN ".$country_code."USERS_POSITIONS
UP ON U.ID=UP.USER_ID
                        INNER JOIN ".$country_code."COMPANIES C ON
UP.COMPANY_ID=C.ID
                        INNER JOIN ".$country_code."EDUCATIONS E ON
U.ID=E.USER_ID
                        INNER JOIN ".$country_code."SCHOOLS S ON
E.SCHOOL_ID=S.ID
                        WHERE
                              U.LOCATION LIKE '%$country%' AND
                              S.NAME LIKE '%$school%' AND
                              (
                                    E.DEGREE LIKE '%$degree%' OR
                                    E.MAJOR LIKE '%$degree%'
                              )
                        ORDER BY UID ASC, STARTED ASC";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        $t_uid = 0;
                        $t_maxHops = 0;
                        $from_company = "";
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $uid = $row ['UID'];
                              $started = $row ['STARTED'];
                              $position = ucwords ( strtolower (
$row ['POSITION'] ) );
                              $to_company =
mysql_real_escape_string( ucwords ( strtolower ( $row ['COMPANY'] )
) );

                              if ($from_company == $to_company)
                                    continue;

                              if ($uid != $t_uid){
                                    $from_company = $this->degree;
```

```
                                    $t_maxHops = 0;
                                    $this->firstLevels++;
                              }
                              $this->addEdge ( $from_company,
$to_company );
                              $t_uid = $uid;
                              $from_company = $to_company;
                              $t_maxHops++;

                              if ($t_maxHops>$this->maxHops){
                                    $this->maxHops=$t_maxHops;
                              }
                        }
                  }
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
       * Returns the number of first level leafs in the graph
       */
      public function getFirstLevels(){
            return $this->firstLevels;
      }

      /**
       * Returns the chart's title.
       */
      public function getTitle() {
            return "Employees' Flow Through Companies, by
education";
      }

      /**
       * Returns the chart's Subject.
       */
      public function getSubject() {
            return 'Flow';
      }

      /**
       * Returns the chart's Subject Page.
       */
      public function getSubjectPage() {
            return '<a href="../flow">Employees\' Flow</a>';
      }

      /**
       * Prints the Vertices table in JSON format.
       */
      public function printVertices() {
            //echo json_encode($this->vertices);
            $str = "[";
            foreach($this->vertices as $row => $v){
```

```
                  $str = $str.'{"type": "'.$v['type'].'", "id":
"'.$v['id'].'", "parent": "null", "name": "'.$v['name'].'"}, ';
            }
            $str = substr($str, 0, -2);
            $str = $str."]";
            echo $str;
      }

      /**
       * Prints the Edges table in JSON format.
       */
      public function printEdges() {
            echo json_encode($this->edges);
            /*$str = "[";
            foreach($this->edges as $row => $e){
                  $str = $str.'{"source": '.$e['source'].',
"target": '.$e['target'].', "value": '.$e['value'].'}, ';
            }
            $str = substr($str, 0, -2);
            $str = $str."]";
            echo $str;*/
      }

      /**
       * Prints the Edges table in a table format.
       */
      public function printEdgesTable() {
            for($i = 0; $i < sizeof ( $this->edges ); $i ++) {
                  $source = $this->vertices[$this-
>edges[$i]["source"]]["name"];
                  $target = $this->vertices[$this-
>edges[$i]["target"]]["name"];
                  $value = $this->edges[$i]["value"];
                  echo '
                  <tr>
                        <td></td>
                        <td>'.($i+1).'</td>
                        <td>'.$source.'</td>
                        <td>'.$target.'</td>
                        <td>'.$value.'</td>
                  </tr>
                  ';
            }
      }

      /**
       * Returns the maximum path taken by an employee.
       */
      public function getMaxHops(){
            echo $this->maxHops;
      }
}
?>
```

FlowBySkills.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iBihisankey.php');

class FlowBySkills extends Chart implements iBihisankey{
	private $country;
	private $degree;
	private $school;
	private $maxHops = 0;
	private $vertices = array ();
	private $edges = array ();
	private $firstLevels = 0;

	public function __construct() {
		if (!isset($_GET['country']) || $_GET['country']==""){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: 'country' variable was not
given.";
			die();
		}
		if ( !isset($_GET['tags']) ){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: no skills were given.";
			die();
		}
		$this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
		if (!$this->isCountrySupported($this->country)){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
			die();
		}
		$this->getData ();
		if (!isset($_GET['count']))
			$this->insertHistory();
	}

	/**
	 * Checks if the given vertex name exists in the vetices
table.
	 * If so, it returns the id; if not, the new vertex is added
	 * and the id is returned.
	 *
	 * @param unknown $v Given vertex name
	 */
	private function getVertexId($v) {
		for($i = 0; $i < sizeof ( $this->vertices ); $i ++) {
			if ($this->vertices [$i]["name"] == $v) {
				return $this->vertices [$i]["id"];
			}
		}
		$type = chr(65+rand(0,25));
		$newVertex = array (
```

```
                  "type" => "$type",
                  "id" => $i,
                  "parent" => "null",
                  "name" => $v
            );
            array_push( $this->vertices, $newVertex );
            return $this->vertices[$i]["id"];
      }

      /**
       * Adds an edge between the 'from' and 'to' vertices.
       * @param unknown $from Source Vertex name.
       * @param unknown $to Target Vertex name.
       */
      private function addEdge($from, $to) {

            // Get the IDs of 'from' and 'to' vertex
            $fromId = $this->getVertexId ( $from );
            $toId = $this->getVertexId ( $to );

            // If the edge exist, +1 the value
            for($i = 0; $i < sizeof ( $this->edges ); $i ++) {
                  if ($this->edges [$i] ["source"] == $fromId &&
$this->edges [$i] ["target"] == $toId) {
                        $this->edges [$i] ["value"] ++;
                        return;
                  }
            }

            // Create new edge between source and target vertex
            $newEdge = array(
                  "source" => $fromId,
                  "target" => $toId,
                  "value" => 1
            );
            array_push( $this->edges, $newEdge );
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $tags = "";
            foreach ($_GET['tags'] as $tag){
                  $tag = ucwords( strtolower(
mysql_real_escape_string($tag) ) );
                  $tags = $tags."S.NAME LIKE '%$tag%' OR ";
            }
            $tags = substr($tags, 0, -4);
            $sql = "SELECT DISTINCT
                              UT.ID AS UID,
                              CONCAT(
                                    UP.START_YEAR, '-',
```

```
                                    LPAD(UP.START_MONTH,2,'00'), '-',
                                    LPAD(UP.START_DAY,2,'00')
                              ) AS STARTED,
                              UP.TITLE AS POSITION,
                              C.NAME AS COMPANY
                        FROM(
                              SELECT U.ID
                              FROM ".$country_code."USERS U
                              WHERE U.LOCATION LIKE '%$country%'
                        ) UT
                        INNER JOIN ".$country_code."USERS_POSITIONS
UP ON UT.ID=UP.USER_ID
                        INNER JOIN ".$country_code."COMPANIES C ON
UP.COMPANY_ID=C.ID
                        INNER JOIN ".$country_code."USERS_SKILLS US
ON UT.ID=US.USER_ID
                        INNER JOIN (
                              SELECT S.ID
                              FROM ".$country_code."SKILLS S
                              WHERE
                                    $tags
                        ) ST ON US.SKILL_ID=ST.ID
                        ORDER BY UT.ID ASC, STARTED ASC";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        $t_uid = 0;
                        $t_maxHops = 0;
                        $from_company = "";
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $uid = $row ['UID'];
                              $started = $row ['STARTED'];
                              $position = ucwords ( strtolower (
$row ['POSITION'] ) );
                              $to_company =
mysql_real_escape_string( ucwords ( strtolower ( $row ['COMPANY'] )
) );

                              if ($from_company == $to_company)
                                    continue;

                              if ($uid != $t_uid){
                                    $from_company = "Start";
                                    $t_maxHops = 0;
                                    $this->firstLevels++;
                              }
                              $this->addEdge ( $from_company,
$to_company );
                              $t_uid = $uid;
                              $from_company = $to_company;
                              $t_maxHops++;

                              if ($t_maxHops>$this->maxHops){
                                    $this->maxHops=$t_maxHops;
                              }
```

```
                        }
                  }
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
       * Returns the number of first level leafs in the graph
       */
      public function getFirstLevels(){
            return $this->firstLevels;
      }

      /**
       * Returns the chart's title.
       */
      public function getTitle() {
            return "Employees' Flow Through Companies, by their
skills";
      }

      /**
       * Returns the chart's Subject.
       */
      public function getSubject() {
            return 'Flow';
      }

      /**
       * Returns the chart's Subject Page.
       */
      public function getSubjectPage() {
            return '<a href="../flow">Employees\' Flow</a>';
      }

      /**
       * Prints the Vertices table in JSON format.
       */
      public function printVertices() {
            //echo json_encode($this->vertices);
            $str = "[";
            foreach($this->vertices as $row => $v){
                  $str = $str.'{"type": "'.$v['type'].'", "id":
"'.$v['id'].'", "parent": "null", "name": "'.$v['name'].'"}, ';
            }
            $str = substr($str, 0, -2);
            $str = $str."]";
            echo $str;
      }

      /**
       * Prints the Edges table in JSON format.
       */
      public function printEdges() {
```

```
            echo json_encode($this->edges);
            /*$str = "[";
            foreach($this->edges as $row => $e){
                  $str = $str.'{"source": '.$e['source'].',
"target": '.$e['target'].', "value": '.$e['value'].'}, ';
            }
            $str = substr($str, 0, -2);
            $str = $str."]";
            echo $str;*/
      }

      /**
       * Prints the Edges table in a table format.
       */
      public function printEdgesTable() {
            for($i = 0; $i < sizeof ( $this->edges ); $i ++) {
                  $source = $this->vertices[$this-
>edges[$i]["source"]]["name"];
                  $target = $this->vertices[$this-
>edges[$i]["target"]]["name"];
                  $value = $this->edges[$i]["value"];
                  echo '
                  <tr>
                        <td></td>
                        <td>'.($i+1).'</td>
                        <td>'.$source.'</td>
                        <td>'.$target.'</td>
                        <td>'.$value.'</td>
                  </tr>
                  ';
            }
      }

      public function getMaxHops(){
            echo $this->maxHops;
      }
}
?>
```

TopGroups.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCPB.php');

class TopGroups extends Chart implements iCPB{
      private $country;
      private $array = array ();

      /**
       * Class constructor. Checks required variables and
       * calls the getData() function.
       */
      public function __construct() {
            if (!isset($_GET['country']) || $_GET['country']==""){
```

```
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country' variable was not
given.";
                  die();
            }
            $this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
            if (!$this->isCountrySupported($this->country)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
                  die();
            }
            $this->getData();
            if (!isset($_GET['count']))
                  $this->insertHistory();
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            if ($country == "Cyprus")
                  $sql = "SELECT * FROM CY_TOP_GROUPS LIMIT 0, 100";
            else if ($country == "Estonia")
                  $sql = "SELECT * FROM EE_TOP_GROUPS LIMIT 0, 100";
            else
                  $sql = "SELECT
                                    G.Title,
                                    Count(UG.USER_ID) AS Total
                              FROM ".$country_code."USERS U
                              INNER JOIN
".$country_code."USERS_GROUPS UG ON U.ID=UG.USER_ID
                              INNER JOIN ".$country_code."GROUPS G
ON UG.GROUP_ID=G.ID
                              WHERE
                                    U.LOCATION LIKE '%$country%'
                              GROUP BY G.Title
                              ORDER BY Total DESC
                              LIMIT 0, 100";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "Group" => ucwords( strtolower(
mysql_real_escape_string($row['Title']) ) ),
                                    "Total" => ucwords( strtolower(
mysql_real_escape_string($row['Total']) ) )
                              );
```

```
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
       * Returns the sample size of the current
       * graph.
       */
      public function getSampleSize() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code."USERS U
                        WHERE LOCATION LIKE '%$country%'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        $row = mysqli_fetch_assoc ( $result );
                        $total = $row['TOTAL'];
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
            return $total;
      }

      /**
       * Returns the chart's title.
       */
      public function getTitle() {
            return "Top 100 Groups with peopler from ".$this-
>country;
      }

      /**
       * Returns the chart's Subject.
       */
      public function getSubject() {
            return 'Groups';
      }

      /**
       * Returns the chart's Subject Page.
```

```
	 */
	public function getSubjectPage() {
		return '<a href="../groups/">Groups</a>';
	}

	/**
	 * Returns the chart's X axis name.
	 */
	public function getXaxisName(){
		return "Group";
	}

	/**
	 * Returns the chart's Y axis name.
	 */
	public function getYaxisName(){
		return "Total";
	}

	/**
	 * Prints the array in JSON format.
	 */
	public function printJSON() {
		echo json_encode($this->array);
	}

	/**
	 * Prints the array in a HTML table format.
	 */
	public function printTable() {
		for($i = 0; $i < sizeof ( $this->array ); $i ++) {
			$group = $this->array[$i]["Group"];
			$total = $this->array[$i]["Total"];
			echo '
			<tr>
				<td></td>
				<td>'.($i+1).'</td>
				<td>'.$group.'</td>
				<td>'.$total.'</td>
			</tr>
			';
		}
	}
}
?>
```

CompareTopGroups.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCBCompare.php');

class CompareTopGroups extends Chart implements iCBCompare{
	private $country1;
	private $country2;
	private $array = array ();

	/**
	 * Class constructor. Checks required variables and
	 * calls the getData() function.
	 */
	public function __construct() {
		if (!isset($_GET['country1']) || $_GET['country1']==""){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: 'country1' variable was not
given.";
			die();
		}
		if (!isset($_GET['country2']) || $_GET['country2']==""){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: 'country2' variable was not
given.";
			die();
		}
		$this->country1 = ucwords( strtolower(
mysql_real_escape_string($_GET['country1']) ) );
		$this->country2 = ucwords( strtolower(
mysql_real_escape_string($_GET['country2']) ) );
		if (!$this->isCountrySupported($this->country1)){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: ".$this->country1 ." is
<b>not</b> a supported country.";
			die();
		}
		if (!$this->isCountrySupported($this->country2)){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: ".$this->country2 ." is
<b>not</b> a supported country.";
			die();
		}
		$this->getData();
		if (!isset($_GET['count']))
			$this->insertHistory();
	}

	/**
	 * Communicates with database to return
	 * the appropriate data needed.
	 */
	public function getData() {
		include '../../_conn.php';
		$sql = "";
```

```
            $country1 = $this->country1;
            $country2 = $this->country2;
            $country_code1 = $this->getCountryCode($country1);
            $country_code2 = $this->getCountryCode($country2);
            $sql = "SELECT T1.Title, T1.TOTAL AS TOTAL1,
IFNULL(T2.TOTAL, '0') AS TOTAL2
                        FROM (
                              SELECT * FROM
".$country_code1."TOP_GROUPS LIMIT 0,100
                        )T1
                        LEFT JOIN ".$country_code2."TOP_GROUPS T2 ON
T1.Title=T2.Title
                        ORDER BY TOTAL1 DESC";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "Group" => ucwords( strtolower(
mysql_real_escape_string($row['Title']) ) ),
                                    "Total1" => $row['TOTAL1'],
                                    "Total2" => $row['TOTAL2']
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
       * Returns the sample size for the first
       * country of the current graph.
       */
      public function getSampleSize1() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country1 = $this->country1;
            $country_code1 = $this->getCountryCode($country1);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code1."USERS U
                        WHERE LOCATION LIKE '%$country1%'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        $row = mysqli_fetch_assoc ( $result );
                        $total = $row['TOTAL'];
                  }
                  mysqli_close($conn);
```

```
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
            return $total;
      }

      /**
       * Returns the sample size for the second
       * country of the current graph.
       */
      public function getSampleSize2() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country2 = $this->country2;
            $country_code2 = $this->getCountryCode($country2);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code2."USERS U
                        WHERE LOCATION LIKE '%$country2%'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        $row = mysqli_fetch_assoc ( $result );
                        $total = $row['TOTAL'];
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
            return $total;
      }

      /**
       * Returns the chart's title.
       */
      public function getTitle() {
            return "Groups Comparison between ".$this->country1 ."
and ".$this->country2;
      }

      /**
       * Returns the chart's Subject.
       */
      public function getSubject() {
            return 'Groups';
      }

      /**
       * Returns the chart's Subject Page.
       */
      public function getSubjectPage() {
            return '<a href="../groups/">Groups</a>';
```

```
	}

	/**
	 * Returns the chart's X axis name.
	 */
	public function getXaxisName(){
		return "Group";
	}

	/**
	 * Returns the name of the first country.
	 */
	public function getCountry1(){
		return $this->country1;
	}

	/**
	 * Returns the name of the second country.
	 */
	public function getCountry2(){
		return $this->country2;
	}

	/**
	 * Prints the array in JSON format.
	 */
	public function printJSON() {
		echo json_encode($this->array);
	}

	/**
	 * Prints the array in a HTML table format.
	 */
	public function printTable() {
		for($i = 0; $i < sizeof ( $this->array ); $i ++) {
			$group = $this->array[$i]["Group"];
			$total1 = $this->array[$i]["Total1"];
			$total2 = $this->array[$i]["Total2"];
			echo '
			<tr>
				<td></td>
				<td>'.($i+1).'</td>
				<td>'.$group.'</td>
				<td>'.$total1.'</td>
				<td>'.$total2.'</td>
			</tr>
			';
		}
	}
}
?>
```

TopLang.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCPB.php');

class TopLang extends Chart implements iCPB{
	private $country;
	private $array = array ();

	/**
	 * Class constructor. Checks required variables and
	 * calls the getData() function.
	 */
	public function __construct() {
		if (!isset($_GET['country']) || $_GET['country']==""){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: 'country' variable was not
given.";
			die();
		}
		$this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
		if (!$this->isCountrySupported($this->country)){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
			die();
		}
		$this->getData ();
		if (!isset($_GET['count']))
			$this->insertHistory();
	}

	/**
	 * Communicates with database to return
	 * the appropriate data needed.
	 */
	public function getData() {
		include '../../_conn.php';
		$sql = "";
		$country = $this->country;
		$country_code = $this->getCountryCode($country);
		if ($country == "Cyprus")
			$sql = "SELECT * FROM CY_TOP_LANGUAGES LIMIT 0,
100";
		else if ($country == "Estonia")
			$sql = "SELECT * FROM EE_TOP_LANGUAGES LIMIT 0,
100";
		else
			$sql = "SELECT
					IFNULL( SL.PRIMARY_LANG, L.NAME )
AS Lang,
					COUNT( UL.USER_ID ) AS Total
				FROM ".$country_code."USERS U
```

```
                              INNER JOIN
".$country_code."USERS_LANGUAGES UL ON U.ID=UL.USER_ID
                              INNER JOIN ".$country_code."LANGUAGES
L ON UL.LANGUAGE_ID = L.ID
                              LEFT JOIN SYNONYMS_LANGUAGES SL ON
L.NAME LIKE SL.SYNONYM_LANG
                              WHERE
                                    U.LOCATION LIKE '%$country%'
                              GROUP BY Lang
                              ORDER BY Total DESC
                              LIMIT 0, 100";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "Language" => ucwords(
strtolower( mysql_real_escape_string($row['Lang']) ) ),
                                    "Total" => ucwords( strtolower(
mysql_real_escape_string($row['Total']) ) )
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
       * Returns the sample size of the current
       * graph.
       */
      public function getSampleSize() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code."USERS U
                        WHERE LOCATION LIKE '%$country%'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        $row = mysqli_fetch_assoc ( $result );
                        $total = $row['TOTAL'];
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
```

```
                echo "503 Service Unavailable";
                die();
        }
        return $total;
    }

    /**
     * Returns the chart's title.
     */
    public function getTitle() {
        return "Top 100 Spoken Languages in ".$this->country;
    }

    /**
     * Returns the chart's Subject.
     */
    public function getSubject() {
        return 'Language';
    }

    /**
     * Returns the chart's Subject Page.
     */
    public function getSubjectPage() {
        return '<a href="../language/">Language</a>';
    }

    /**
     * Returns the chart's X axis name.
     */
    public function getXaxisName(){
        return "Language";
    }

    /**
     * Returns the chart's Y axis name.
     */
    public function getYaxisName(){
        return "Total";
    }

    /**
     * Prints the array in JSON format.
     */
    public function printJSON() {
        echo json_encode($this->array);
    }

    /**
     * Prints the array in a HTML table format.
     */
    public function printTable() {
        for($i = 0; $i < sizeof ( $this->array ); $i ++) {
                $lang = $this->array[$i]["Language"];
                $total = $this->array[$i]["Total"];
                echo '
                <tr>
```

```
                        <td></td>
                        <td>'.($i+1).'</td>
                        <td>'.$lang.'</td>
                        <td>'.$total.'</td>
                  </tr>
                  ';
            }
      }
}
?>
```

NumSpokenLang.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCPB.php');

class NumSpokenLang extends Chart implements iCPB{
      private $country;
      private $array = array ();

      /**
       * Class constructor. Checks required variables and
       * calls the getData() function.
       */
      public function __construct() {
            if (!isset($_GET['country']) || $_GET['country']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country' variable was not
given.";
                  die();
            }
            $this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
            if (!$this->isCountrySupported($this->country)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
                  die();
            }
            $this->getData ();
            if (!isset($_GET['count']))
                  $this->insertHistory();
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            if ($country == "Cyprus")
```

```
                  $sql = "SELECT * FROM CY_NUM_SPOKEN_LANG";
            else if ($country == "Estonia")
                  $sql = "SELECT * FROM EE_NUM_SPOKEN_LANG";
            else{
                  $sql = "SELECT
                                    TEMP.LANGUAGES_KNOW AS 'Number of
Spoken Languages',
                                    COUNT(TEMP.UID) AS 'Number of
People'
                              FROM (
                                    SELECT
                                          UL.USER_ID AS UID,
                                          COUNT( IFNULL(
SL.PRIMARY_LANG, L.NAME) ) AS LANGUAGES_KNOW
                                    FROM ".$country_code."USERS U
                                    INNER JOIN
".$country_code."USERS_LANGUAGES UL ON U.ID=UL.USER_ID
                                    INNER JOIN
".$country_code."LANGUAGES L ON UL.LANGUAGE_ID = L.ID
                                    LEFT JOIN SYNONYMS_LANGUAGES SL
ON L.NAME LIKE SL.SYNONYM_LANG
                                    WHERE
                                          U.LOCATION LIKE
'%$country%'
                                          GROUP BY UID
                                          ORDER BY LANGUAGES_KNOW
DESC
                              ) TEMP
                              GROUP BY TEMP.LANGUAGES_KNOW;";
            }
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "Number of Spoken Languages" =>
ucwords( strtolower( mysql_real_escape_string($row['Number of
Spoken Languages']) ) ),
                                    "Number of Peple" => ucwords(
strtolower( mysql_real_escape_string($row['Number of People']) ) )
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
       * Returns the sample size of the current
       * graph.
```

```
	 */
	public function getSampleSize() {
		include '../../_conn.php';
		$total = 0;
		$sql = "";
		$country = $this->country;
		$country_code = $this->getCountryCode($country);
		$sql = "SELECT COUNT(*) AS TOTAL
				FROM ".$country_code."USERS U
				WHERE LOCATION LIKE '%$country%'";
		if ($conn) {
			mysqli_set_charset ( $conn, "utf8" );
			$result = mysqli_query ( $conn, $sql );
			if (mysqli_num_rows ( $result ) > 0) {
				$row = mysqli_fetch_assoc ( $result );
				$total = $row['TOTAL'];
			}
			mysqli_close($conn);
		}else{
			header('HTTP/1.1 503 Service Unavailable');
			echo "503 Service Unavailable";
			die();
		}
		return $total;
	}

	/**
	 * Returns the chart's title.
	 */
	public function getTitle() {
		return "Multilingual knowledge in ".$this->country;
	}

	/**
	 * Returns the chart's Subject.
	 */
	public function getSubject() {
		return 'Language';
	}

	/**
	 * Returns the chart's Subject Page.
	 */
	public function getSubjectPage() {
		return '<a href="../language/">Language</a>';
	}

	/**
	 * Returns the chart's X axis name.
	 */
	public function getXaxisName(){
		return "Number of Spoken Languages";
	}

	/**
	 * Returns the chart's Y axis name.
	 */
```

```
	public function getYaxisName(){
		return "Number of People";
	}

	/**
	 * Prints the array in JSON format.
	 */
	public function printJSON() {
		echo json_encode($this->array);
	}

	/**
	 * Prints the array in a HTML table format.
	 */
	public function printTable() {
		for($i = 0; $i < sizeof ( $this->array ); $i ++) {
			$numLang = $this->array[$i]["Number of Spoken
Languages"];
			$numPeople = $this->array[$i]["Number of Peple"];
			echo '
			<tr>
				<td></td>
				<td>'.($i+1).'</td>
				<td>'.$numLang.'</td>
				<td>'.$numPeople.'</td>
			</tr>
			';
		}
	}
}
?>
```

LangByDegree.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCPB.php');

class LangByDegree extends Chart implements iCPB{
	private $country;
	private $degree;
	private $array = array ();

	/**
	 * Class constructor. Checks required variables and
	 * calls the getData() function.
	 */
	public function __construct() {
		if (!isset($_GET['country']) || $_GET['country']==""){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: 'country' variable was not
given.";
			die();
		}
		if (!isset($_GET['degree']) || $_GET['degree']==""){
```

```
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'degree' variable was not
given.";
                  die();
            }
            $this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
            if (!$this->isCountrySupported($this->country)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
                  die();
            }
            $this->degree = ucwords( strtolower(
mysql_real_escape_string($_GET['degree']) ) );
            $this->getData ();
            if (!isset($_GET['count']))
                  $this->insertHistory();
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $country = $this->country;
            $degree = $this->degree;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT
                              IFNULL( SL.PRIMARY_LANG, L.NAME ) AS
Lang,
                              COUNT( UL.USER_ID ) AS Total
                        FROM ".$country_code."USERS U
                        INNER JOIN ".$country_code."EDUCATIONS E ON
U.ID=E.USER_ID
                        INNER JOIN ".$country_code."USERS_LANGUAGES
UL ON U.ID=UL.USER_ID
                        INNER JOIN ".$country_code."LANGUAGES L ON
UL.LANGUAGE_ID=L.ID
                        LEFT JOIN SYNONYMS_LANGUAGES SL ON L.NAME
LIKE SL.SYNONYM_LANG
                        WHERE
                              U.LOCATION LIKE '%$country%' AND
                              E.START_YEAR < E.END_YEAR AND (
                                    E.DEGREE LIKE '%$degree%' OR
                                    E.MAJOR LIKE '%$degree%'
                              )
                        GROUP BY Lang
                        ORDER BY Total DESC";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
```

```
                                    "Language" => ucwords(
strtolower( mysql_real_escape_string($row['Lang']) ) ),
                                    "Total" => ucwords( strtolower(
mysql_real_escape_string($row['Total']) ) )
                                );
                                array_push( $this->array, $new_row );
                          }
                    }
                    mysqli_close($conn);
              }else{
                    header('HTTP/1.1 503 Service Unavailable');
                    echo "503 Service Unavailable";
                    die();
              }
        }

        /**
         * Returns the sample size of the current
         * graph.
         */
        public function getSampleSize() {
              include '../../_conn.php';
              $total = 0;
              $sql = "";
              $country = $this->country;
              $country_code = $this->getCountryCode($country);
              $sql = "SELECT COUNT(*) AS TOTAL
                          FROM ".$country_code."USERS U
                          WHERE LOCATION LIKE '%$country%'";
              if ($conn) {
                    mysqli_set_charset ( $conn, "utf8" );
                    $result = mysqli_query ( $conn, $sql );
                    if (mysqli_num_rows ( $result ) > 0) {
                          $row = mysqli_fetch_assoc ( $result );
                          $total = $row['TOTAL'];
                    }
                    mysqli_close($conn);
              }else{
                    header('HTTP/1.1 503 Service Unavailable');
                    echo "503 Service Unavailable";
                    die();
              }
              return $total;
        }

        /**
         * Returns the chart's title.
         */
        public function getTitle() {
              return "Spoken Languages by Degree: ".$this->degree;
        }

        /**
         * Returns the chart's Subject.
         */
        public function getSubject() {
              return 'Language';
```

```
	}

	/**
	 * Returns the chart's Subject Page.
	 */
	public function getSubjectPage() {
		return '<a href="../language/">Language</a>';
	}

	/**
	 * Returns the chart's X axis name.
	 */
	public function getXaxisName(){
		return "Language";
	}

	/**
	 * Returns the chart's Y axis name.
	 */
	public function getYaxisName(){
		return "Total";
	}

	/**
	 * Prints the array in JSON format.
	 */
	public function printJSON() {
		echo json_encode($this->array);
	}

	/**
	 * Prints the array in a HTML table format.
	 */
	public function printTable() {
		for($i = 0; $i < sizeof ( $this->array ); $i ++) {
			$lang = $this->array[$i]["Language"];
			$total = $this->array[$i]["Total"];
			echo '
			<tr>
				<td></td>
				<td>'.($i+1).'</td>
				<td>'.$lang.'</td>
				<td>'.$total.'</td>
			</tr>
			';
		}
	}
}
?>
```

CompareLang.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCPB.php');

class CompareLang extends Chart implements iCPB{
	private $country;
	private $array = array ();

	/**
	 * Class constructor. Checks required variables and
	 * calls the getData() function.
	 */
	public function __construct() {
		if (!isset($_GET['country']) || $_GET['country']==""){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: 'country' variable was not
given.";
			die();
		}
		if ( !isset($_GET['tags']) ){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: no languages were given to
compare.";
			die();
		}
		$this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
		if (!$this->isCountrySupported($this->country)){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
			die();
		}
		$this->getData ();
		if (!isset($_GET['count']))
			$this->insertHistory();
	}

	/**
	 * Communicates with database to return
	 * the appropriate data needed.
	 */
	public function getData() {
		include '../../_conn.php';
		$country = $this->country;
		$country_code = $this->getCountryCode($country);
		foreach ($_GET['tags'] as $tag){
			$tag = ucwords( strtolower(
mysql_real_escape_string($tag) ) );
			$sql = "SELECT
					IFNULL( SL.PRIMARY_LANG, L.NAME )
AS Lang,
					COUNT( UL.USER_ID ) AS Total
				FROM ".$country_code."USERS U
```

```
                              INNER JOIN
".$country_code."USERS_LANGUAGES UL ON U.ID=UL.USER_ID
                              INNER JOIN ".$country_code."LANGUAGES
L ON UL.LANGUAGE_ID=L.ID
                              LEFT JOIN SYNONYMS_LANGUAGES SL ON
L.NAME LIKE SL.SYNONYM_LANG
                              WHERE
                                    U.LOCATION LIKE '%$country%' AND
                                    SL.PRIMARY_LANG LIKE '%$tag%'
                              GROUP BY Lang
                              ORDER BY Total DESC";
                  if ($conn) {
                        $total = 0;
                        mysqli_set_charset($conn, "utf8");
                        $result=mysqli_query($conn, $sql);
                        if (mysqli_num_rows($result) > 0) {
                              while($row =
mysqli_fetch_assoc($result)) {
                                    $total = $total + $row['Total'];
                              }
                              $new_row = array (
                                    "Language" => $tag,
                                    "Total" => $total
                              );
                              array_push( $this->array, $new_row );
                        }
                  }else{
                        header('HTTP/1.1 503 Service Unavailable');
                        echo "503 Service Unavailable";
                        die();
                  }
            }
            mysqli_close($conn);
      }

      /**
       * Returns the sample size of the current
       * graph.
       */
      public function getSampleSize() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code."USERS U
                        WHERE LOCATION LIKE '%$country%'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        $row = mysqli_fetch_assoc ( $result );
                        $total = $row['TOTAL'];
                  }
                  mysqli_close($conn);
            }else{
```

```
                header('HTTP/1.1 503 Service Unavailable');
                echo "503 Service Unavailable";
                die();
        }
        return $total;
    }

    /**
     * Returns the chart's title.
     */
    public function getTitle() {
        return "Language Comparison";
    }

    /**
     * Returns the chart's Subject.
     */
    public function getSubject() {
        return 'Language';
    }

    /**
     * Returns the chart's Subject Page.
     */
    public function getSubjectPage() {
        return '<a href="../language/">Language</a>';
    }

    /**
     * Returns the chart's X axis name.
     */
    public function getXaxisName(){
        return "Language";
    }

    /**
     * Returns the chart's Y axis name.
     */
    public function getYaxisName(){
        return "Total";
    }

    /**
     * Prints the array in JSON format.
     */
    public function printJSON() {
        echo json_encode($this->array);
    }

    /**
     * Prints the array in a HTML table format.
     */
    public function printTable() {
        for($i = 0; $i < sizeof ( $this->array ); $i ++) {
                $lang = $this->array[$i]["Language"];
                $total = $this->array[$i]["Total"];
                echo '
```

```
                  <tr>
                        <td></td>
                        <td>'.($i+1).'</td>
                        <td>'.$lang.'</td>
                        <td>'.$total.'</td>
                  </tr>
                  ';
            }
      }
}
?>
```

CompareNumSpokenLang.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCBCompare.php');

class CompareNumSpokenLang extends Chart implements iCBCompare{
      private $country1;
      private $country2;
      private $array = array ();

      /**
       * Class constructor. Checks required variables and
       * calls the getData() function.
       */
      public function __construct() {
            if (!isset($_GET['country1']) || $_GET['country1']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country1' variable was not
given.";
                  die();
            }
            if (!isset($_GET['country2']) || $_GET['country2']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country2' variable was not
given.";
                  die();
            }
            $this->country1 = ucwords( strtolower(
mysql_real_escape_string($_GET['country1']) ) );
            $this->country2 = ucwords( strtolower(
mysql_real_escape_string($_GET['country2']) ) );
            if (!$this->isCountrySupported($this->country1)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country1 ." is
<b>not</b> a supported country.";
                  die();
            }
            if (!$this->isCountrySupported($this->country2)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country2 ." is
<b>not</b> a supported country.";
                  die();
```

```
            }
            $this->getData();
            if (!isset($_GET['count']))
                  $this->insertHistory();
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $sql = "";
            $country1 = $this->country1;
            $country2 = $this->country2;
            $country_code1 = $this->getCountryCode($country1);
            $country_code2 = $this->getCountryCode($country2);
            $sql = "SELECT
                              T1.`Number of Spoken Languages`,
                              T1.`Number of People` AS TOTAL1,
                              IFNULL(T2.`Number of People`, '0') AS
TOTAL2
                        FROM (
                              SELECT * FROM
".$country_code1."NUM_SPOKEN_LANG
                        )T1
                        LEFT JOIN ".$country_code2."NUM_SPOKEN_LANG
T2 ON T1.`Number of Spoken Languages`=T2.`Number of Spoken
Languages`
                        ORDER BY T1.`Number of Spoken Languages` ASC
                        ";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "LangNum" => ucwords( strtolower(
mysql_real_escape_string($row['Number of Spoken Languages']) ) ),
                                    "Total1" => ucwords( strtolower(
mysql_real_escape_string($row['TOTAL1']) ) ),
                                    "Total2" => ucwords( strtolower(
mysql_real_escape_string($row['TOTAL2']) ) )
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
```

```
     * Returns the sample size for the first
     * country of the current graph.
     */
    public function getSampleSize1() {
          include '../../_conn.php';
          $total = 0;
          $sql = "";
          $country1 = $this->country1;
          $country_code1 = $this->getCountryCode($country1);
          $sql = "SELECT COUNT(*) AS TOTAL
                      FROM ".$country_code1."USERS U
                      WHERE LOCATION LIKE '%$country1%'";
          if ($conn) {
                mysqli_set_charset ( $conn, "utf8" );
                $result = mysqli_query ( $conn, $sql );
                if (mysqli_num_rows ( $result ) > 0) {
                      $row = mysqli_fetch_assoc ( $result );
                      $total = $row['TOTAL'];
                }
                mysqli_close($conn);
          }else{
                header('HTTP/1.1 503 Service Unavailable');
                echo "503 Service Unavailable";
                die();
          }
          return $total;
    }

    /**
     * Returns the sample size for the second
     * country of the current graph.
     */
    public function getSampleSize2() {
          include '../../_conn.php';
          $total = 0;
          $sql = "";
          $country2 = $this->country2;
          $country_code2 = $this->getCountryCode($country2);
          $sql = "SELECT COUNT(*) AS TOTAL
                      FROM ".$country_code2."USERS U
                      WHERE LOCATION LIKE '%$country2%'";
          if ($conn) {
                mysqli_set_charset ( $conn, "utf8" );
                $result = mysqli_query ( $conn, $sql );
                if (mysqli_num_rows ( $result ) > 0) {
                      $row = mysqli_fetch_assoc ( $result );
                      $total = $row['TOTAL'];
                }
                mysqli_close($conn);
          }else{
                header('HTTP/1.1 503 Service Unavailable');
                echo "503 Service Unavailable";
                die();
          }
          return $total;
    }
```

```
	/**
	 * Returns the chart's title.
	 */
	public function getTitle() {
		return "Multilingual knowledge comparison between
".$this->country1 ." and ".$this->country2;
	}

	/**
	 * Returns the chart's Subject.
	 */
	public function getSubject() {
		return 'Language';
	}

	/**
	 * Returns the chart's Subject Page.
	 */
	public function getSubjectPage() {
		return '<a href="../language/">Language</a>';
	}

	/**
	 * Returns the chart's X axis name.
	 */
	public function getXaxisName(){
		return "Number of Spoken Languages";
	}

	/**
	 * Returns the name of the first country.
	 */
	public function getCountry1(){
		return $this->country1;
	}

	/**
	 * Returns the name of the second country.
	 */
	public function getCountry2(){
		return $this->country2;
	}

	/**
	 * Prints the array in JSON format.
	 */
	public function printJSON() {
		echo json_encode($this->array);
	}

	/**
	 * Prints the array in a HTML table format.
	 */
	public function printTable() {
		for($i = 0; $i < sizeof ( $this->array ); $i ++) {
			$NumLang = $this->array[$i]["LangNum"];
			$total1 = $this->array[$i]["Total1"];
```

```
                  $total2 = $this->array[$i]["Total2"];
                  echo '
                  <tr>
                        <td></td>
                        <td>'.($i+1).'</td>
                        <td>'.$NumLang.'</td>
                        <td>'.$total1.'</td>
                        <td>'.$total2.'</td>
                  </tr>
                  ';
            }
      }
}
?>
```

CompareTopLang.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCBCompare.php');

class CompareTopLang extends Chart implements iCBCompare{
      private $country1;
      private $country2;
      private $array = array ();

      /**
       * Class constructor. Checks required variables and
       * calls the getData() function.
       */
      public function __construct() {
            if (!isset($_GET['country1']) || $_GET['country1']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country1' variable was not
given.";
                  die();
            }
            if (!isset($_GET['country2']) || $_GET['country2']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country2' variable was not
given.";
                  die();
            }
            $this->country1 = ucwords( strtolower(
mysql_real_escape_string($_GET['country1']) ) );
            $this->country2 = ucwords( strtolower(
mysql_real_escape_string($_GET['country2']) ) );
            if (!$this->isCountrySupported($this->country1)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country1 ." is
<b>not</b> a supported country.";
                  die();
            }
            if (!$this->isCountrySupported($this->country2)){
                  header('HTTP/1.1 400 Bad Request');
```

```
                  echo "400 Bad Request: ".$this->country2 ." is
<b>not</b> a supported country.";
                  die();
            }
            $this->getData();
            if (!isset($_GET['count']))
                  $this->insertHistory();
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $sql = "";
            $country1 = $this->country1;
            $country2 = $this->country2;
            $country_code1 = $this->getCountryCode($country1);
            $country_code2 = $this->getCountryCode($country2);
            $sql = "SELECT T1.LANG, T1.TOTAL AS TOTAL1,
IFNULL(T2.TOTAL, '0') AS TOTAL2
                        FROM (
                              SELECT * FROM
".$country_code1."TOP_LANGUAGES LIMIT 0,100
                        )T1
                        LEFT JOIN ".$country_code2."TOP_LANGUAGES T2
ON T1.LANG=T2.LANG
                        ORDER BY TOTAL1 DESC
                        ";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "Language" => ucwords(
strtolower( mysql_real_escape_string($row['LANG']) ) ),
                                    "Total1" => $row['TOTAL1'],
                                    "Total2" => $row['TOTAL2']
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
       * Returns the sample size for the first
       * country of the current graph.
       */
```

```
	public function getSampleSize1() {
		include '../../_conn.php';
		$total = 0;
		$sql = "";
		$country1 = $this->country1;
		$country_code1 = $this->getCountryCode($country1);
		$sql = "SELECT COUNT(*) AS TOTAL
				FROM ".$country_code1."USERS U
				WHERE LOCATION LIKE '%$country1%'";
		if ($conn) {
			mysqli_set_charset ( $conn, "utf8" );
			$result = mysqli_query ( $conn, $sql );
			if (mysqli_num_rows ( $result ) > 0) {
				$row = mysqli_fetch_assoc ( $result );
				$total = $row['TOTAL'];
			}
			mysqli_close($conn);
		}else{
			header('HTTP/1.1 503 Service Unavailable');
			echo "503 Service Unavailable";
			die();
		}
		return $total;
	}

	/**
	 * Returns the sample size for the second
	 * country of the current graph.
	 */
	public function getSampleSize2() {
		include '../../_conn.php';
		$total = 0;
		$sql = "";
		$country2 = $this->country2;
		$country_code2 = $this->getCountryCode($country2);
		$sql = "SELECT COUNT(*) AS TOTAL
				FROM ".$country_code2."USERS U
				WHERE LOCATION LIKE '%$country2%'";
		if ($conn) {
			mysqli_set_charset ( $conn, "utf8" );
			$result = mysqli_query ( $conn, $sql );
			if (mysqli_num_rows ( $result ) > 0) {
				$row = mysqli_fetch_assoc ( $result );
				$total = $row['TOTAL'];
			}
			mysqli_close($conn);
		}else{
			header('HTTP/1.1 503 Service Unavailable');
			echo "503 Service Unavailable";
			die();
		}
		return $total;
	}

	/**
	 * Returns the chart's title.
	 */
```

```
	public function getTitle() {
		return "Comparison between ".$this->country1 ." and
".$this->country2;
	}

	/**
	 * Returns the chart's Subject.
	 */
	public function getSubject() {
		return 'Language';
	}

	/**
	 * Returns the chart's Subject Page.
	 */
	public function getSubjectPage() {
		return '<a href="../language/">Language</a>';
	}

	/**
	 * Returns the chart's X axis name.
	 */
	public function getXaxisName(){
		return "Language";
	}

	/**
	 * Returns the name of the first country.
	 */
	public function getCountry1(){
		return $this->country1;
	}

	/**
	 * Returns the name of the second country.
	 */
	public function getCountry2(){
		return $this->country2;
	}

	/**
	 * Prints the array in JSON format.
	 */
	public function printJSON() {
		echo json_encode($this->array);
	}

	/**
	 * Prints the array in a HTML table format.
	 */
	public function printTable() {
		for($i = 0; $i < sizeof ( $this->array ); $i ++) {
			$lang = $this->array[$i]["Language"];
			$total1 = $this->array[$i]["Total1"];
			$total2 = $this->array[$i]["Total2"];
			echo '
			<tr>
```

```
                        <td></td>
                        <td>'.($i+1).'</td>
                        <td>'.$lang.'</td>
                        <td>'.$total1.'</td>
                        <td>'.$total2.'</td>
                  </tr>
                  ';
            }
      }
}
?>
```

TopSkills.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCPB.php');

class TopSkills extends Chart implements iCPB{
      private $country;
      private $array = array ();

      /**
       * Class constructor. Checks required variables and
       * calls the getData() function.
       */
      public function __construct() {
            if (!isset($_GET['country']) || $_GET['country']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country' variable was not
given.";
                  die();
            }
            $this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
            if (!$this->isCountrySupported($this->country)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
                  die();
            }
            $this->getData();
            if (!isset($_GET['count']))
                  $this->insertHistory();
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
```

```
            if ($country == "Cyprus")
                  $sql = "SELECT * FROM CY_TOP_SKILLS LIMIT 0, 100";
            else if ($country == "Estonia")
                  $sql = "SELECT * FROM EE_TOP_SKILLS LIMIT 0, 100";
            else
                  $sql = "SELECT
                                    S.name As Skill,
                                    Count(US.user_id) AS Total
                              FROM ".$country_code."USERS U
                              INNER JOIN
".$country_code."USERS_SKILLS US ON U.ID=US.USER_ID
                              INNER JOIN ".$country_code."SKILLS S
ON US.skill_id=S.id
                              WHERE
                                    U.LOCATION LIKE '%$country%'
                              GROUP BY Skill
                              ORDER BY Total DESC
                              LIMIT 0, 100";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "Skill" => ucwords( strtolower(
mysql_real_escape_string($row['Skill']) ) ),
                                    "Total" => ucwords( strtolower(
mysql_real_escape_string($row['Total']) ) )
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
       * Returns the sample size of the current
       * graph.
       */
      public function getSampleSize() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code."USERS U
                        WHERE LOCATION LIKE '%$country%'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
```

```
                if (mysqli_num_rows ( $result ) > 0) {
                      $row = mysqli_fetch_assoc ( $result );
                      $total = $row['TOTAL'];
                }
                mysqli_close($conn);
          }else{
                header('HTTP/1.1 503 Service Unavailable');
                echo "503 Service Unavailable";
                die();
          }
          return $total;
    }

    /**
     * Returns the chart's title.
     */
    public function getTitle() {
          return "Top 100 Skills in ".$this->country;
    }

    /**
     * Returns the chart's Subject.
     */
    public function getSubject() {
          return 'Skills';
    }

    /**
     * Returns the chart's Subject Page.
     */
    public function getSubjectPage() {
          return '<a href="../skills/">Skills</a>';
    }

    /**
     * Returns the chart's X axis name.
     */
    public function getXaxisName(){
          return "Skill";
    }

    /**
     * Returns the chart's Y axis name.
     */
    public function getYaxisName(){
          return "Total";
    }

    /**
     * Prints the array in JSON format.
     */
    public function printJSON() {
          echo json_encode($this->array);
    }

    /**
     * Prints the array in a HTML table format.
```

```
       */
      public function printTable() {
            for($i = 0; $i < sizeof ( $this->array ); $i ++) {
                  $skill = $this->array[$i]["Skill"];
                  $total = $this->array[$i]["Total"];
                  echo '
                  <tr>
                        <td></td>
                        <td>'.($i+1).'</td>
                        <td>'.$skill.'</td>
                        <td>'.$total.'</td>
                  </tr>
                  ';
            }
      }
}
?>
```

SkillsByDegree.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCPB.php');

class SkillsByDegree extends Chart implements iCPB{
      private $country;
      private $degree;
      private $array = array ();

      /**
       * Class constructor. Checks required variables and
       * calls the getData() function.
       */
      public function __construct() {
            if (!isset($_GET['country']) || $_GET['country']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country' variable was not
given.";
                  die();
            }
            if (!isset($_GET['degree']) || $_GET['degree']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'degree' variable was not
given.";
                  die();
            }
            $this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
            $this->degree = ucwords( strtolower(
mysql_real_escape_string($_GET['degree']) ) );
            if (!$this->isCountrySupported($this->country)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
                  die();
```

```
            }
            $this->getData ();
            if (!isset($_GET['count']))
                  $this->insertHistory();
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $country = $this->country;
            $degree = $this->degree;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT
                              S.NAME As Skill,
                              COUNT(US.USER_ID) AS Total
                        FROM ".$country_code."USERS U
                        INNER JOIN ".$country_code."USERS_SKILLS US
ON U.ID=US.USER_ID
                        INNER JOIN ".$country_code."SKILLS S ON
US.SKILL_ID=S.ID
                        INNER JOIN ".$country_code."EDUCATIONS E ON
US.USER_ID=E.USER_ID
                        WHERE
                              U.LOCATION LIKE '%$country%' AND (
                                    E.DEGREE LIKE '%$degree%' OR
                                    E.MAJOR LIKE '%$degree%'
                              )
                        GROUP BY Skill
                        ORDER BY Total DESC
                        LIMIT 0, 1000";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "Skill" => ucwords( strtolower(
mysql_real_escape_string($row['Skill']) ) ),
                                    "Total" => ucwords( strtolower(
mysql_real_escape_string($row['Total']) ) )
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
```

```
	 * Returns the sample size of the current
	 * graph.
	 */
	public function getSampleSize() {
		include '../../_conn.php';
		$total = 0;
		$sql = "";
		$country = $this->country;
		$country_code = $this->getCountryCode($country);
		$sql = "SELECT COUNT(*) AS TOTAL
				FROM ".$country_code."USERS U
				WHERE LOCATION LIKE '%$country%'";
		if ($conn) {
			mysqli_set_charset ( $conn, "utf8" );
			$result = mysqli_query ( $conn, $sql );
			if (mysqli_num_rows ( $result ) > 0) {
				$row = mysqli_fetch_assoc ( $result );
				$total = $row['TOTAL'];
			}
			mysqli_close($conn);
		}else{
			header('HTTP/1.1 503 Service Unavailable');
			echo "503 Service Unavailable";
			die();
		}
		return $total;
	}

	/**
	 * Returns the chart's title.
	 */
	public function getTitle() {
		return "Skills by Degree: ".$this->degree;
	}

	/**
	 * Returns the chart's Subject.
	 */
	public function getSubject() {
		return 'Skills';
	}

	/**
	 * Returns the chart's Subject Page.
	 */
	public function getSubjectPage() {
		return '<a href="../skills/">Skills</a>';
	}

	/**
	 * Returns the chart's X axis name.
	 */
	public function getXaxisName(){
		return "Skill";
	}

	/**
```

```
	 * Returns the chart's Y axis name.
	 */
	public function getYaxisName(){
		return "Total";
	}

	/**
	 * Prints the array in JSON format.
	 */
	public function printJSON() {
		echo json_encode($this->array);
	}

	/**
	 * Prints the array in a HTML table format.
	 */
	public function printTable() {
		for($i = 0; $i < sizeof ( $this->array ); $i ++) {
			$skill = $this->array[$i]["Skill"];
			$total = $this->array[$i]["Total"];
			echo '
			<tr>
				<td></td>
				<td>'.($i+1).'</td>
				<td>'.$skill.'</td>
				<td>'.$total.'</td>
			</tr>
			';
		}
	}
}
?>
```

CompareSkills.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCPB.php');

class CompareSkills extends Chart implements iCPB{
	private $country;
	private $array = array ();

	/**
	 * Class constructor. Checks required variables and
	 * calls the getData() function.
	 */
	public function __construct() {
		if (!isset($_GET['country']) || $_GET['country']==""){
			header('HTTP/1.1 400 Bad Request');
			echo "400 Bad Request: 'country' variable was not
given.";
			die();
		}
		if ( !isset($_GET['tags']) ){
```

```
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: no skills were given to
compare.";
                  die();
            }
            $this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
            if (!$this->isCountrySupported($this->country)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
                  die();
            }
            $this->getData ();
            if (!isset($_GET['count']))
                  $this->insertHistory();
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            foreach ($_GET['tags'] as $tag){
                  $tag = ucwords( strtolower(
mysql_real_escape_string($tag) ) );
                  $sql = "SELECT
                                    S.NAME As Skill,
                                    Count(US.USER_ID) AS Total
                              FROM ".$country_code."USERS U
                              INNER JOIN
".$country_code."USERS_SKILLS US ON U.ID=US.USER_ID
                              INNER JOIN ".$country_code."SKILLS S
ON US.SKILL_ID=S.ID
                              WHERE
                                    U.LOCATION LIKE '%$country%' AND
                                    S.NAME LIKE '$tag'
                              GROUP BY Skill
                              ORDER BY Total DESC";
                  if ($conn) {
                        $total = 0;
                        mysqli_set_charset($conn, "utf8");
                        $result=mysqli_query($conn, $sql);
                        if (mysqli_num_rows($result) > 0) {
                              while($row =
mysqli_fetch_assoc($result)) {
                                    $total = $total + $row['Total'];
                              }
                              $new_row = array (
                                    "Skill" => $tag,
                                    "Total" => $total
                              );
                              array_push( $this->array, $new_row );
                        }
```

```
                }else{
                        header('HTTP/1.1 503 Service Unavailable');
                        echo "503 Service Unavailable";
                        die();
                }
        }
        mysqli_close($conn);
}

/**
 * Returns the sample size of the current
 * graph.
 */
public function getSampleSize() {
        include '../../_conn.php';
        $total = 0;
        $sql = "";
        $country = $this->country;
        $country_code = $this->getCountryCode($country);
        $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code."USERS U
                        WHERE LOCATION LIKE '%$country%'";
        if ($conn) {
                mysqli_set_charset ( $conn, "utf8" );
                $result = mysqli_query ( $conn, $sql );
                if (mysqli_num_rows ( $result ) > 0) {
                        $row = mysqli_fetch_assoc ( $result );
                        $total = $row['TOTAL'];
                }
                mysqli_close($conn);
        }else{
                header('HTTP/1.1 503 Service Unavailable');
                echo "503 Service Unavailable";
                die();
        }
        return $total;
}

/**
 * Returns the chart's title.
 */
public function getTitle() {
        return "Skills Comparison";
}

/**
 * Returns the chart's Subject.
 */
public function getSubject() {
        return 'Skills';
}

/**
 * Returns the chart's Subject Page.
 */
public function getSubjectPage() {
        return '<a href="../skills/">Skills</a>';
```

```
	}

	/**
	 * Returns the chart's X axis name.
	 */
	public function getXaxisName(){
		return "Skill";
	}

	/**
	 * Returns the chart's Y axis name.
	 */
	public function getYaxisName(){
		return "Total";
	}

	/**
	 * Prints the array in JSON format.
	 */
	public function printJSON() {
		echo json_encode($this->array);
	}

	/**
	 * Prints the array in a HTML table format.
	 */
	public function printTable() {
		for($i = 0; $i < sizeof ( $this->array ); $i ++) {
			$skill = $this->array[$i]["Skill"];
			$total = $this->array[$i]["Total"];
			echo '
			<tr>
				<td></td>
				<td>'.($i+1).'</td>
				<td>'.$skill.'</td>
				<td>'.$total.'</td>
			</tr>
			';
		}
	}
}
?>
```

CompareTopSkills.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCBCompare.php');

class CompareTopSkills extends Chart implements iCBCompare{
	private $country1;
	private $country2;
	private $array = array ();

	/**
```

```
       * Class constructor. Checks required variables and
       * calls the getData() function.
       */
      public function __construct() {
            if (!isset($_GET['country1']) || $_GET['country1']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country1' variable was not
given.";
                  die();
            }
            if (!isset($_GET['country2']) || $_GET['country2']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country2' variable was not
given.";
                  die();
            }
            $this->country1 = ucwords( strtolower(
mysql_real_escape_string($_GET['country1']) ) );
            $this->country2 = ucwords( strtolower(
mysql_real_escape_string($_GET['country2']) ) );
            if (!$this->isCountrySupported($this->country1)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country1 ." is
<b>not</b> a supported country.";
                  die();
            }
            if (!$this->isCountrySupported($this->country2)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country2 ." is
<b>not</b> a supported country.";
                  die();
            }
            $this->getData();
            if (!isset($_GET['count']))
                  $this->insertHistory();
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $sql = "";
            $country1 = $this->country1;
            $country2 = $this->country2;
            $country_code1 = $this->getCountryCode($country1);
            $country_code2 = $this->getCountryCode($country2);
            $sql = "SELECT T1.SKILL, T1.TOTAL AS TOTAL1,
IFNULL(T2.TOTAL, '0') AS TOTAL2
                        FROM (
                              SELECT * FROM
".$country_code1."TOP_SKILLS LIMIT 0,100
                        )T1
                        LEFT JOIN ".$country_code2."TOP_SKILLS T2 ON
T1.SKILL=T2.SKILL
                        ORDER BY TOTAL1 DESC
```

```
                        ";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "Skill" => ucwords( strtolower(
mysql_real_escape_string($row['SKILL']) ) ),
                                    "Total1" => ucwords( strtolower(
mysql_real_escape_string($row['TOTAL1']) ) ),
                                    "Total2" => ucwords( strtolower(
mysql_real_escape_string($row['TOTAL2']) ) )
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
      }

      /**
       * Returns the sample size for the first
       * country of the current graph.
       */
      public function getSampleSize1() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country1 = $this->country1;
            $country_code1 = $this->getCountryCode($country1);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code1."USERS U
                        WHERE LOCATION LIKE '%$country1%'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        $row = mysqli_fetch_assoc ( $result );
                        $total = $row['TOTAL'];
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
            return $total;
      }

      /**
       * Returns the sample size for the second
```

```
       * country of the current graph.
       */
      public function getSampleSize2() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country2 = $this->country2;
            $country_code2 = $this->getCountryCode($country2);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code2."USERS U
                        WHERE LOCATION LIKE '%$country2%'";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        $row = mysqli_fetch_assoc ( $result );
                        $total = $row['TOTAL'];
                  }
                  mysqli_close($conn);
            }else{
                  header('HTTP/1.1 503 Service Unavailable');
                  echo "503 Service Unavailable";
                  die();
            }
            return $total;
      }

      /**
       * Returns the chart's title.
       */
      public function getTitle() {
            return "Comparison between ".$this->country1 ." and
".$this->country2;
      }

      /**
       * Returns the chart's Subject.
       */
      public function getSubject() {
            return 'Skills';
      }

      /**
       * Returns the chart's Subject Page.
       */
      public function getSubjectPage() {
            return '<a href="../skills/">Skills</a>';
      }

      /**
       * Returns the chart's X axis name.
       */
      public function getXaxisName(){
            return "Skill";
      }

      /**
```

```
	 * Returns the name of the first country.
	 */
	public function getCountry1(){
		return $this->country1;
	}

	/**
	 * Returns the name of the second country.
	 */
	public function getCountry2(){
		return $this->country2;
	}

	/**
	 * Prints the array in JSON format.
	 */
	public function printJSON() {
		echo json_encode($this->array);
	}

	/**
	 * Prints the array in a HTML table format.
	 */
	public function printTable() {
		for($i = 0; $i < sizeof ( $this->array ); $i ++) {
			$skill = $this->array[$i]["Skill"];
			$total1 = $this->array[$i]["Total1"];
			$total2 = $this->array[$i]["Total2"];
			echo '
			<tr>
				<td></td>
				<td>'.($i+1).'</td>
				<td>'.$skill.'</td>
				<td>'.$total1.'</td>
				<td>'.$total2.'</td>
			</tr>
			';
		}
	}
}
?>
```

TopVolunteeringCauses.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCPB.php');

class TopVolunteeringCauses extends Chart implements iCPB{
	private $country;
	private $array = array ();

	/**
	 * Class constructor. Checks required variables and
	 * calls the getData() function.
```

```
       */
      public function __construct() {
            if (!isset($_GET['country']) || $_GET['country']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country' variable was not
given.";
                  die();
            }
            $this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
            if ($this->country!="Estonia"){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
                  die();
            }
            $this->getData();
            if (!isset($_GET['count']))
                  $this->insertHistory();
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT V.Cause, COUNT(V.USER_ID) AS Total
                        FROM ".$country_code."USERS U
                        INNER JOIN ".$country_code."VOLUNTEERING V
ON U.ID=V.USER_ID
                        WHERE
                              U.LOCATION LIKE '%$country%' AND
                              length(V.Cause)>0
                        GROUP BY V.Cause
                        ORDER BY Total DESC
                        LIMIT 0, 100";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "Cause" => ucwords( strtolower(
mysql_real_escape_string($row['Cause']) ) ),
                                    "Total" => ucwords( strtolower(
mysql_real_escape_string($row['Total']) ) )
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
```

```
                header('HTTP/1.1 503 Service Unavailable');
                echo "503 Service Unavailable";
                die();
            }
        }

        /**
         * Returns the sample size of the current
         * graph.
         */
        public function getSampleSize() {
            include '../../_conn.php';
            $total = 0;
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT COUNT(*) AS TOTAL
                        FROM ".$country_code."USERS U
                        WHERE LOCATION LIKE '%$country%'";
            if ($conn) {
                mysqli_set_charset ( $conn, "utf8" );
                $result = mysqli_query ( $conn, $sql );
                if (mysqli_num_rows ( $result ) > 0) {
                    $row = mysqli_fetch_assoc ( $result );
                    $total = $row['TOTAL'];
                }
                mysqli_close($conn);
            }else{
                header('HTTP/1.1 503 Service Unavailable');
                echo "503 Service Unavailable";
                die();
            }
            return $total;
        }

        /**
         * Returns the chart's title.
         */
        public function getTitle() {
            return "Top 100 Volunteering Causes that people from
".$this->country ." are part of";
        }

        /**
         * Returns the chart's Subject.
         */
        public function getSubject() {
            return 'Volunteering';
        }

        /**
         * Returns the chart's Subject Page.
         */
        public function getSubjectPage() {
            return '<a href="../Volunteering/">Volunteering</a>';
        }
```

```
	/**
	 * Returns the chart's X axis name.
	 */
	public function getXaxisName(){
		return "Volunteering Cause";
	}

	/**
	 * Returns the chart's Y axis name.
	 */
	public function getYaxisName(){
		return "Total";
	}

	/**
	 * Prints the array in JSON format.
	 */
	public function printJSON() {
		echo json_encode($this->array);
	}

	/**
	 * Prints the array in a HTML table format.
	 */
	public function printTable() {
		for($i = 0; $i < sizeof ( $this->array ); $i ++) {
			$cause = $this->array[$i]["Cause"];
			$total = $this->array[$i]["Total"];
			echo '
			<tr>
				<td></td>
				<td>'.($i+1).'</td>
				<td>'.$cause.'</td>
				<td>'.$total.'</td>
			</tr>
			';
		}
	}
}
?>
```

TopNames.class.php:

```
<?php

require('../Chart.class.php');
require('../interfaces/iCPB.php');

class TopNames extends Chart implements iCPB{
	private $country;
	private $array = array ();

	/**
	 * Class constructor. Checks required variables and
	 * calls the getData() function.
	 */
```

```
      public function __construct() {
            if (!isset($_GET['country']) || $_GET['country']==""){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: 'country' variable was not
given.";
                  die();
            }
            $this->country = ucwords( strtolower(
mysql_real_escape_string($_GET['country']) ) );
            if (!$this->isCountrySupported($this->country)){
                  header('HTTP/1.1 400 Bad Request');
                  echo "400 Bad Request: ".$this->country." is
<b>not</b> a supported country.";
                  die();
            }
            $this->getData ();
            if (!isset($_GET['count']))
                  $this->insertHistory();
      }

      /**
       * Communicates with database to return
       * the appropriate data needed.
       */
      public function getData() {
            include '../../_conn.php';
            $sql = "";
            $country = $this->country;
            $country_code = $this->getCountryCode($country);
            $sql = "SELECT
                              IFNULL(SN.FIRST_NAME, U.FIRST_NAME) AS
Name,
                              COUNT(U.ID) AS Total
                        FROM ".$country_code."USERS U
                        LEFT JOIN SYNONYMS_NAMES SN ON
U.FIRST_NAME=SN.SYNONYM_NAME
                        WHERE U.LOCATION LIKE '%$country%'
                        GROUP BY Name
                        ORDER BY Total DESC
                        LIMIT 0, 100";
            if ($conn) {
                  mysqli_set_charset ( $conn, "utf8" );
                  $result = mysqli_query ( $conn, $sql );
                  if (mysqli_num_rows ( $result ) > 0) {
                        while ( $row = mysqli_fetch_assoc ( $result
) ) {
                              $new_row = array (
                                    "Name" => ucwords( strtolower(
mysql_real_escape_string($row['Name']) ) ),
                                    "Total" => ucwords( strtolower(
mysql_real_escape_string($row['Total']) ) )
                              );
                              array_push( $this->array, $new_row );
                        }
                  }
                  mysqli_close($conn);
            }else{
```

```
                header('HTTP/1.1 503 Service Unavailable');
                echo "503 Service Unavailable";
                die();
        }
    }

    /**
     * Returns the sample size of the current
     * graph.
     */
    public function getSampleSize() {
        include '../../_conn.php';
        $total = 0;
        $sql = "";
        $country = $this->country;
        $country_code = $this->getCountryCode($country);
        $sql = "SELECT COUNT(*) AS TOTAL
                    FROM ".$country_code."USERS U
                    WHERE LOCATION LIKE '%$country%'";
        if ($conn) {
            mysqli_set_charset ( $conn, "utf8" );
            $result = mysqli_query ( $conn, $sql );
            if (mysqli_num_rows ( $result ) > 0) {
                $row = mysqli_fetch_assoc ( $result );
                $total = $row['TOTAL'];
            }
            mysqli_close($conn);
        }else{
            header('HTTP/1.1 503 Service Unavailable');
            echo "503 Service Unavailable";
            die();
        }
        return $total;
    }

    /**
     * Returns the chart's title.
     */
    public function getTitle() {
        return "Top 100 most popular names in ".$this->country;
    }

    /**
     * Returns the chart's Subject.
     */
    public function getSubject() {
        return 'Miscellaneous';
    }

    /**
     * Returns the chart's Subject Page.
     */
    public function getSubjectPage() {
        return '<a href="../misc/">Miscellaneous</a>';
    }

    /**
```

```
        * Returns the chart's X axis name.
        */
      public function getXaxisName(){
            return "Name";
      }

      /**
       * Returns the chart's Y axis name.
       */
      public function getYaxisName(){
            return "Total";
      }

      /**
       * Prints the array in JSON format.
       */
      public function printJSON() {
            echo json_encode($this->array);
      }

      /**
       * Prints the array in a HTML table format.
       */
      public function printTable() {
            for($i = 0; $i < sizeof ( $this->array ); $i ++) {
                  $name = $this->array[$i]["Name"];
                  $total = $this->array[$i]["Total"];
                  echo '
                  <tr>
                        <td></td>
                        <td>'.($i+1).'</td>
                        <td>'.$name.'</td>
                        <td>'.$total.'</td>
                  </tr>
                  ';
            }
      }
}
?>
```