Ατομική Διπλωματική Εργασία

ΑΝΑΠΤΥΞΗ ΛΕΙΤΟΥΡΓΙΩΝ ΚΑΙ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ ΠΡΟΕΓΓΡΑΦΗΣ ΜΑΘΗΜΑΤΩΝ ΠΕΡΙΟΡΙΣΜΕΝΗΣ ΕΠΙΛΟΓΗΣ

Στυλιανός Αντωνουδιού

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Λειτουργιών και Αρχιτεκτονική Συστήματος Προεγγραφής Μαθημάτων Περιορισμένης Επιλογής

Στυλιανός Αντωνουδιού

Επιβλέπων Καθηγητής
Άννα Φιλίππου

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

Μάιος 2025

Ευχαριστίες

Πρωτίστως, θα ήθελα να ευχαριστήσω την επιβλέπουσα καθηγήτρια μου, Κ. Άννα Φιλίππου, για την καθοδήγηση, την εμπιστοσύνη και την πολύτιμη υποστήριξη που μου παρείχε καθ' όλη τη διάρκεια της εργασίας. Οι χρήσιμες υποδείξεις της και το ευχάριστο κλίμα συνεργασίας που δημιούργησε συνέβαλαν καθοριστικά στην ολοκλήρωση αυτού του έργου.

Ιδιαίτερη μνεία θέλω να κάνω στον αδερφό μου, Κυριάκο Αντωνουδιού, με τον οποίο συνεργαστήκαμε στενά στην ανάπτυξη του συστήματος.Η συνεργασία μας υπήρξε καθοριστική και η αμοιβαία υποστήριξη που προσφέραμε ο ένας στον άλλον αποτέλεσε τον βασικό παράγοντα για την επιτυχή ολοκλήρωση αυτής της εργασίας. Ακόμη, θα ήθελα να ευχαριστήσω τον κ. Γιάννη Δημόπουλο, καθηγητή του μαθήματος ΕΠΛ433 - Προγραμματισμός και Ικανοποίηση Περιορισμών, για την εισαγωγή του στις θεμελιώδεις έννοιες σχετικά με την ικανοποίηση περιορισμών, οι οποίες αποδείχτηκαν κρίσιμες για την υλοποίηση του αλγορίθμου και την επιτυχή ολοκλήρωση της εργασίας.

Τέλος, ευχαριστώ την οικογένειά μου και τους φίλους μου για την αμέριστη στήριξή τους και την ψυχολογική υποστήριξη που μου προσέφεραν καθ' όλη τη διάρκεια των σπουδών μου. Η στήριξή τους ήταν αναπόσπαστο κομμάτι της ολοκλήρωσης αυτής της προσπάθειας και τους οφείλω ένα μεγάλο ευχαριστώ.

Περίληψη

Σκοπός της Ατομικής Διπλωματικής Εργασίας είναι η ανάπτυξη ενός συστήματος προεγγραφών σε μαθήματα περιορισμένης επιλογής για τους φοιτητές του Τμήματος Πληροφορικής. Η ανάγκη για τη δημιουργία του συστήματος προέκυψε από την παρατηρούμενη ταλαιπωρία κατά τη διαδικασία εγγραφής, τόσο από την πλευρά των φοιτητών όσο και από την πλευρά του Διαχειριστή του Συστήματος και του Συντονιστή του Τμήματος. Ο κύριος στόχος της εργασίας είναι η αυτοματοποίηση της διαδικασίας προεγγραφής, ελαχιστοποιώντας το άγχος και την ταλαιπωρία που προκαλούνται κατά την εγγραφή σε μαθήματα περιορισμένης επιλογής.

Η εργασία εστιάζει κυρίως στην ανάπτυξη των λειτουργιών του συστήματος, συμπεριλαμβανομένου του τρόπου με τον οποίο οι φοιτητές μπορούν να καταχωρούν τις προτιμήσεις τους για τα μαθήματα και να τις ταξινομούν με βάση την προτεραιότητά τους. Αν και το σύστημα υποστηρίζει τη χρήση ενός αλγορίθμου για την αυτόματη κατανομή των φοιτητών στα μαθήματα, η παρούσα εργασία επικεντρώνεται κυρίως στην αρχιτεκτονική και την υλοποίηση των βασικών λειτουργιών που εξασφαλίζουν την ευχρηστία και την ευελιξία του συστήματος. Το αποτέλεσμα είναι ένα σύστημα που επιτρέπει την καλύτερη ικανοποίηση των φοιτητών και την απλοποίηση της διαδικασίας για τους υπεύθυνους, παρέχοντας μια πιο ομαλή και αποδοτική εμπειρία προεγγραφής.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Παρουσίαση του προβλήματος και του σκοπού της εργασίας

Η διαδικασία εγγραφής σε μαθήματα περιορισμένης επιλογής αποτελεί μια αγχωτική διαδικασία για τους φοιτητές, αλλά και για τους Συντονιστές του Τμήματος Πληροφορικής. Τα προβλήματα που προκύπτουν από την ύπαρξη περιορισμένων θέσεων στα μαθήματα, οι οποίες μοιράζονται μεταξύ των φοιτητών βάσει πιστωτικών μονάδων, δημιουργούν συχνά άγχος, τόσο για τους φοιτητές όσο και για τους υπεύθυνους διαχείρισης της διαδικασίας.

Το πρόβλημα εντείνεται καθώς οι φοιτητές καλούνται να καταχωρήσουν τις επιλογές τους, χωρίς την εγγύηση ότι θα επιλεγούν στα μαθήματα της προτίμησής τους αφού μπορεί να γεμίσουν όλες οι θέσεις των μαθημάτων αυτών και να πρέπει να εγγραφούν σε άλλα μαθήματα. Επιπλέον, οι Διαχειριστές και οι Συντονιστές διαχειρίζονται μεγάλο φόρτο εργασίας, καθώς καλούνται να επιλύσουν προβλήματα που σχετίζονται με την κατανομή των μαθημάτων.

Ο σκοπός της εργασίας αυτής είναι η ανάπτυξη ενός συστήματος προεγγραφών σε μαθήματα περιορισμένης επιλογής, το οποίο θα αυτοματοποιήσει και θα βελτιώσει τη διαδικασία εγγραφής, μειώνοντας τα προβλήματα και την ταλαιπωρία τόσο για τους φοιτητές όσο και για τους υπεύθυνους.

1.2 Σκοπός και στόχοι του συστήματος

Ο βασικός σκοπός του αναπτυσσόμενου συστήματος είναι να αυτοματοποιήσει την διαδικασία προεγγραφής των φοιτητών σε μαθήματα περιορισμένης επιλογής, με στόχο την εξοικονόμηση χρόνου, τη μείωση του άγχους των φοιτητών και τη βελτίωση της αποτελεσματικότητας για τους Διαχειριστές και των Συντονιστών του Τμήματος.

Οι κύριοι στόχοι του συστήματος περιλαμβάνουν:

- Αυτοματοποίηση της διαδικασίας προεγγραφής: Οι φοιτητές θα μπορούν να δηλώσουν τις προτιμήσεις τους για τα μαθήματα και να τις ταξινομούν κατά προτεραιότητα.
- Διασφάλιση δίκαιης κατανομής μαθημάτων: Ο αλγόριθμος που θα χρησιμοποιείται για την κατανομή των φοιτητών θα εξασφαλίζει ότι τα μαθήματα κατανεμηθούν δίκαια, σύμφωνα με προκαθορισμένους περιορισμούς και κριτήρια.
- Απλοποίηση της διαχείρισης: Οι Διαχειριστές του Συστήματος θα μπορούν να παρακολουθούν και να ενεργοποιούν τον αλγόριθμο, μειώνοντας τον φόρτο εργασίας τους και εξασφαλίζοντας πιο αποτελεσματική λειτουργία του συστήματος.
- Ενίσχυση της φοιτητικής ικανοποίησης: Το σύστημα στοχεύει στην ελαχιστοποίηση της ταλαιπωρίας των φοιτητών κατά την εγγραφή τους σε μαθήματα περιορισμένης επιλογής, βελτιώνοντας την εμπειρία τους.
- Κατανόηση προτιμήσεων: Το σύστημα συμβάλλει στην έγκαιρη και σαφή αποτύπωση των προτιμήσεων των φοιτητών, επιτρέποντας στους διαχειριστές να εντοπίζουν εγκαίρως τάσεις ή προβλήματα και να λαμβάνουν διορθωτικά μέτρα. Με αυτόν τον τρόπο, διευκολύνεται η καλύτερη ικανοποίηση των αναγκών των φοιτητών και η βελτίωση της διαδικασίας επιλογής μαθημάτων.

1.3 Μεθοδολογία

- **Σεπτέμβριος - Οκτώβριος 2024:** Ξεκίνησε η φάση ανάλυσης και πρωτοτυποποίησης. Πραγματοποιήθηκε συλλογή απαιτήσεων, ανάλυση του προβλήματος και καταγραφή βασικών σεναρίων χρήσης. Παράλληλα, έγινε χρήση του εργαλείου **Justinmind** για τη δημιουργία ενός πρώιμου πρωτοτύπου διεπαφής, με σκοπό την αρχική απεικόνιση της λειτουργικότητας του συστήματος. Την ίδια περίοδο, δημιουργήθηκε και διανεμήθηκε ένα **ερωτηματολόγιο** προς φοιτητές, με στόχο τη συλλογή απόψεων και εμπειριών σχετικών με το υπάρχον σύστημα επιλογής μαθημάτων.

- **Νοέμβριος - Δεκέμβριος 2024:** Ασχολήθηκα εντατικά με τη **μελέτη και ανάπτυξη αλγορίθμων κατανομής** μαθημάτων χρησιμοποιώντας τη γλώσσα **MiniZinc**. Πραγματοποιήθηκαν πειραματισμοί με διαφορετικές διαμορφώσεις περιορισμών, ώστε να επιτευχθεί μια λύση που θα προάγει τη δικαιοσύνη και την ικανοποίηση των φοιτητών.

- **Ιανουάριος - Μάρτιος 2025:** Ξεκίνησε η **υλοποίηση του λογισμικού** σε περιβάλλον **ASP.NET Core MVC**. Δημιουργήθηκαν τα κύρια **Views και Controllers**, τα οποία υποστήριζαν βασικές λειτουργίες όπως η είσοδος χρηστών, η καταγραφή προτιμήσεων, και η διαχείριση κατανομών.

- **Απρίλιος - Μάιος 2025:** Πραγματοποιήθηκαν **δοκιμές του συστήματος**. Επίσης, συντάχθηκε το **τελικό έγγραφο της διπλωματικής εργασίας**, το οποίο περιλαμβάνει την τεκμηρίωση των τεχνικών επιλογών, τη μεθοδολογία, και την αξιολόγηση του συστήματος.

Για την καλύτερη διαχείριση του έργου, χρησιμοποιήθηκε το εργαλείο **Kanban Flow** για τον προγραμματισμό και παρακολούθηση εργασιών, καθώς και το **GitHub** για την έκδοση και αποθήκευση του κώδικα.

Ακολουθεί το **Gantt chart** της χρονικής εξέλιξης του έργου:

Gantt Chart – Χρονοδιάγραμμα Εργασιών

Μήνας	**Σεπ 2024**	**Οκτ 2024**	**Νοε 2024**	**Δεκ 2024**	**Ιαν 2025**	**Φεβ 2025**	**Μαρ 2025**	**Απρ 2025**	**Μάι 2025**
Ανάλυση απαιτήσεων	●●	●●							
Δημιουργία πρωτοτύπου (Justinmind)	●●	●●							
Ερωτηματολόγιο	●●	●●							
Μελέτη CSP και MiniZinc		●●	●●	●●					
Ανάπτυξη αλγορίθμων MiniZinc			●●	●●					
Ανάπτυξη ASP.NET Core MVC					●●	●●	●●		
Υλοποίηση λειτουργιών					●●	●●	●●		
Δοκιμές συστήματος								●●	●●
Συγγραφή διπλωματικής								●●	●●

Σημείωση: Το σύμβολο ●● δηλώνει την περίοδο κατά την οποία πραγματοποιήθηκε η εκάστοτε εργασία.

1.4 Δομή Διπλωματικής Εργασίας

Η παρούσα διπλωματική εργασία είναι οργανωμένη σε επτά κεφάλαια, με στόχο να καλύψει όλες τις σημαντικές πτυχές της ανάπτυξης και της υλοποίησης του συστήματος προεγγραφών. Παρακάτω παρατίθεται μια σύντομη περιγραφή του περιεχομένου κάθε κεφαλαίου:

Κεφάλαιο 1: Εισαγωγή
Το πρώτο κεφάλαιο εισάγει το πρόβλημα και τον σκοπό της εργασίας, καθώς και τους στόχους του συστήματος που αναπτύχθηκε. Επίσης, περιγράφεται η δομή της διπλωματικής εργασίας.

Κεφάλαιο 2: Ανάλυση Απαιτήσεων
Στο κεφάλαιο αυτό αναλύονται οι απαιτήσεις του συστήματος και του αλγορίθμου, ενώ παρουσιάζεται και το ερωτηματολόγιο που χρησιμοποιήθηκε για την κατανόηση των αναγκών των χρηστών του συστήματος.

Κεφάλαιο 3: Θεωρητικό Υπόβαθρο
Εδώ γίνεται εισαγωγή στον προγραμματισμό ικανοποίησης περιορισμών, και παρουσιάζονται τα εργαλεία MiniZinc και Google OR-Tools, τα οποία χρησιμοποιήθηκαν για την ανάπτυξη του αλγορίθμου. Επιπλέον, γίνεται συγκριτική ανάλυση αυτών των εργαλείων.

Κεφάλαιο 4: Σχεδίαση του Συστήματος
Στο κεφάλαιο αυτό περιγράφεται η σχεδίαση του συστήματος, συμπεριλαμβανομένων των οντοτήτων και της σχέσης τους, της βάσης δεδομένων, της αρχιτεκτονικής του συστήματος και της διάρθρωσης των Views, Controllers και Models.

Κεφάλαιο 5: Υλοποίηση του Συστήματος
Εδώ περιγράφεται η διαδικασία ανάπτυξης του συστήματος και η εφαρμογή της διεπαφής χρήστη και της λογικής του συστήματος.

Κεφάλαιο 6: Αξιολόγηση και Δοκιμές

Στο κεφάλαιο αυτό παρουσιάζεται η στρατηγική δοκιμών του συστήματος και η αξιολόγηση της ακρίβειας, αποδοτικότητας και χρηστικότητας του αλγορίθμου, μέσω των χρηστών του συστήματος.

Κεφάλαιο 7: Συμπεράσματα και Μελλοντικές Εργασίες

Στο τελευταίο κεφάλαιο παρατίθεται μια συνολική αξιολόγηση του συστήματος και προτάσεις για μελλοντικές βελτιώσεις και επεκτάσεις.

Κεφάλαιο 2

Ανάλυση Απαιτήσεων

2.1 Ανασκόπηση του συστήματος

Το σύστημα προεγγραφής σε μαθήματα περιορισμένης επιλογής αναπτύχθηκε μέσω μιας συνεργατικής προσέγγισης μεταξύ εμένα και του αδελφού μου Κυριάκου Αντωνουδιού , με τον καθένα να αναλαμβάνει συγκεκριμένες ευθύνες και αρμοδιότητες. Συγκεκριμένα:

Αρμοδιότητες του Στυλιανού Αντωνούδιου:

- Ανάλυση και μοντελοποίηση του προβλήματος προεγγραφής μαθημάτων ως πρόβλημα ικανοποίησης περιορισμών (CSP) στο Minizinc.
- Σχεδίαση και ανάπτυξη του συστήματος σε ASP.NET Core MVC, συμπεριλαμβανομένων των Models, Views, και Controllers.
- Διαχείριση του αποθετηρίου στο GitHub και χρήση του Kanban Flow για την παρακολούθηση της προόδου του έργου.
- Σχεδίαση πρωτοτύπου με το εργαλείο Justinmind.

Αρμοδιότητες του Κυριάκου Αντωνούδιου:

- Συνεισφορά στον αρχικό σχεδιασμό του αλγορίθμου κατανομής και τη βελτιστοποίηση των επιδόσεών του.
- Υλοποίηση του αλγορίθμου κατανομής χρησιμοποιώντας το Google OR-Tools, λαμβάνοντας υπόψη τις προτεραιότητες των φοιτητών και τους περιορισμούς του συστήματος.

- Υποστήριξη στην υλοποίηση των Models, Views και Controllers για τη διαχείριση των δεδομένων φοιτητών και μαθημάτων.
- Προετοιμασία και καθαρισμό των δεδομένων για την αποτελεσματική εκτέλεση του αλγορίθμου.
- Δοκιμή και αξιολόγηση του συστήματος,χρησιμοποιώντας το xUnit ώστε να εξασφαλιστεί η ορθότητα και η αποδοτικότητα της τελικής λύσης.
- Διαχείριση του αποθετηρίου στο GitHub και χρήση του Kanban Flow για την παρακολούθηση της προόδου του έργου.

Η συνεργασία αυτή επέτρεψε την αποτελεσματική ολοκλήρωση του έργου, συνδυάζοντας τις δεξιότητες και τις γνώσεις και των δύο, διασφαλίζοντας ένα ολοκληρωμένο και λειτουργικό σύστημα.

2.2 Περιγραφή του ερωτηματολογίου και των αποτελεσμάτων

Για την κατανόηση των αναγκών των φοιτητών και των απαιτήσεων που σχετίζονται με τη διαδικασία προεγγραφής σε μαθήματα περιορισμένης επιλογής, δημιουργήσαμε και διανείμαμε ένα ερωτηματολόγιο μέσω της γραμματείας του Τμήματος Πληροφορικής. Το ερωτηματολόγιο σχεδιάστηκε από εμάς με στόχο τη συλλογή πολύπλευρων πληροφοριών με τις εμπειρίες, τις προτιμήσεις και τις προτάσεις των φοιτητών. Συνολικά, συμμετείχαν 32 φοιτητές του Τμήματος Πληροφορικής, εκ των οποίων οι 20 ήταν τεταρτοετείς, οι 8 τριτοετείς, ενώ οι υπόλοιποι ήταν πρωτοετείς ή δευτεροετείς.
Το ερωτηματολόγιο περιλάμβανε ερωτήσεις κλειστού και ανοιχτού τύπου, καλύπτοντας τις παρακάτω θεματικές ενότητες:

- Εμπειρία από προηγούμενες διαδικασίες εγγραφής σε μαθήματα περιορισμένης επιλογής.
- Βαθμός ικανοποίησης από τις υπάρχουσες διαδικασίες.
- Προβλήματα που αντιμετωπίστηκαν (π.χ. αδυναμία εγγραφής στο επιθυμητό μάθημα).
- Προτάσεις για βελτίωση της διαδικασίας.
- Προθυμία χρήσης ενός αυτοματοποιημένου συστήματος προεγγραφών.

Παρακάτω παρατίθενται ορισμένα στιγμιότυπα (screenshots) από το ερωτηματολόγιο:

Ερώτηση 1)

Εικόνα 1 – Αποτελέσματα ερώτησης 1

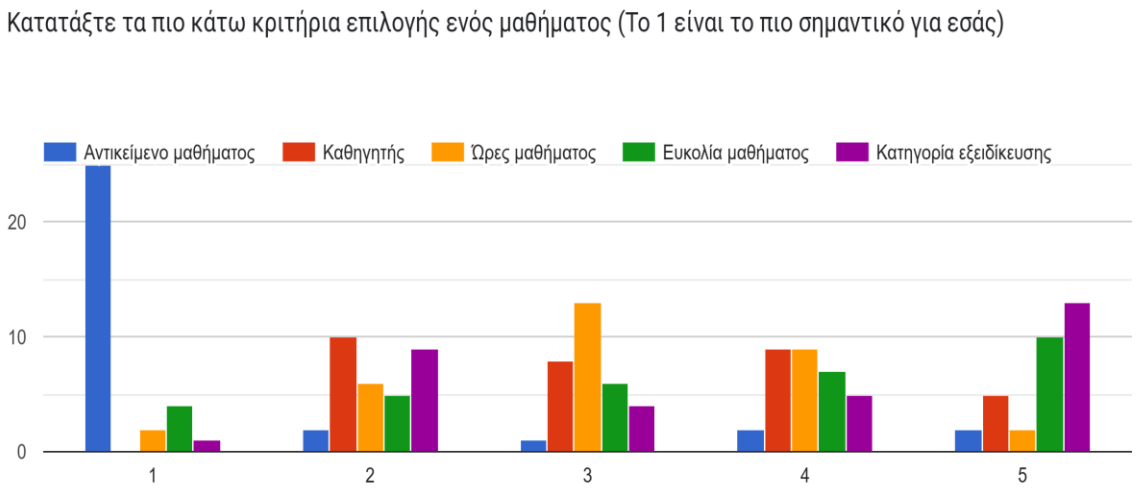
Από αυτή την ερώτηση βλέπουμε ότι οι περισσότεροι φοιτητές που παίρνουν μαθήματα περιορισμένης επιλογής επιλέγουν είτε 1 είτε 2 ενώ ελάχιστοι είναι αυτοί που θα επιλέξουν 3 μαθήματα.

Ερώτηση 2)

Εικόνα 2 – Αποτελέσματα ερώτησης 2

Εδώ βλέπουμε ότι λιγότεροι από το 20% των φοιτητών εγγράφονται στα μαθήματα που θέλουν.

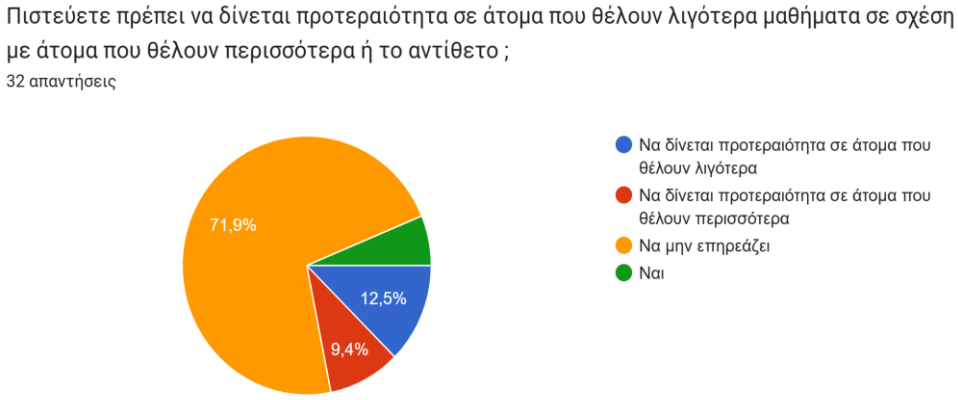
Ερώτηση 3)



Εικόνα 3 – Αποτελέσματα ερώτησης 3

Εδώ βλέπουμε ότι επί το πλείστον οι φοιτητές έχουν ως πρώτο κριτήριο επιλογής ενός μαθήματος το αντικείμενο του μαθήματος και τα υπόλοιπα κριτήρια είναι ισορροπημένα στις πιο κάτω θέσεις.

Ερώτηση 4)



Εικόνα 4 – Αποτελέσματα ερώτησης 4

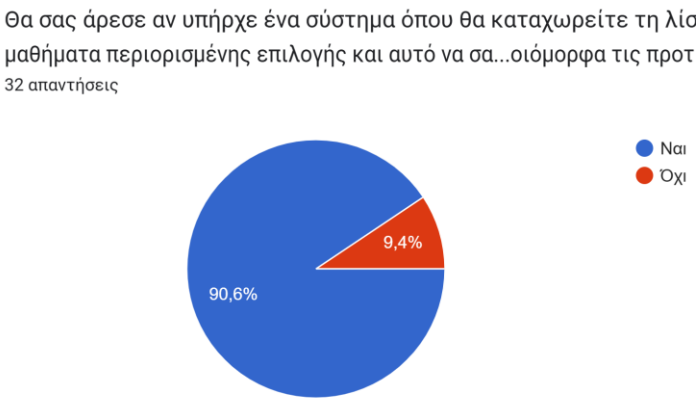
Από την εικόνα 4 βλέπουμε ότι η πλειοψηφία των φοιτητών δεν θέλει ο αριθμός των μαθημάτων περιορισμένης επιλογής που θέλει ένας φοιτητής να επηρεάζει τη προτεραιότητα του.

Ερώτηση 5)

Εικόνα 5 – Αποτελέσματα ερώτησης 5

Η μεγάλη πλειοψηφία των φοιτητών θέλει να έχουν προτεραιότητα εάν χρειάζονται ένα μάθημα για την εκπλήρωση της διπλωματικής τους εργασίας.

Ερώτηση 6)



Εικόνα 6 – Αποτελέσματα ερώτησης 6

Η μεγάλη πλειοψηφία των φοιτητών θέλει ένα σύστημα όπου θα καταχωρούν τις προτιμήσεις τους και αυτό θα τους αναθέσει στα μαθήματα με τον πιο δίκαιο τρόπο.

Ακόμη είχαμε και κάποια σχόλια/ εισηγήσεις από τους φοιτητές όπως :

“Κυρίως να ελευθερώνονται θέσεις ανάλογα με τη ζήτηση του μαθήματος και όχι βάσει χρονικής προτεραιότητας. Επίσης, η κατάταξη μέσω πιστωτικών μονάδων δυσλειτουργεί και δεν είναι δίκαιη, αφού δεν είναι σταθερός ο οδηγός σπουδών. Να υπάρχει ελαστικότητα με τα μαθήματα και επικοινωνία με τους φοιτητές.”

Στη συνέχεια, αναλύθηκαν οι απαντήσεις των φοιτητών. Τα βασικά αποτελέσματα που προέκυψαν ήταν τα εξής:

- Ένα σημαντικό ποσοστό φοιτητών εξέφρασε δυσαρέσκεια σχετικά με την τρέχουσα διαδικασία εγγραφής.
- Οι περισσότεροι φοιτητές δήλωσαν ότι έχουν αντιμετωπίσει προβλήματα μη διαθεσιμότητας θέσεων στα επιθυμητά μαθήματα.
- Υπήρξε έντονη υποστήριξη για την ανάγκη αυτοματοποίησης της διαδικασίας προεγγραφών με χρήση ενός δικαιότερου συστήματος.

Τα δεδομένα αυτά αποτέλεσαν θεμέλιο για τη διαμόρφωση των απαιτήσεων του συστήματος που αναπτύχθηκε.

2.3 Ανάλυση των απαιτήσεων του συστήματος και του αλγορίθμου

Η διαδικασία συλλογής και διαμόρφωσης των απαιτήσεων πραγματοποιήθηκε με συνεχή ανατροφοδότηση και καθοδήγηση. Συγκεκριμένα, οι απαιτήσεις συζητούνταν σε εβδομαδιαία βάση με την επιβλέπουσα καθηγήτρια, κ. Φιλίππου, ώστε να διασφαλιστεί η πληρότητα και η λειτουργικότητα του συστήματος. Παράλληλα, πραγματοποιήθηκε συνάντηση με τη γραμματεία του Τμήματος Πληροφορικής, όπου παρουσιάστηκε μια πρώτη εκδοχή της λειτουργικότητας του συστήματος.

Κατά τη διάρκεια της συνάντησης, εκφράστηκε ιδιαίτερο ενδιαφέρον για τη δυνατότητα παραγωγής στατιστικών με βάση τις λίστες προτιμήσεων των φοιτητών. Η συγκεκριμένη δυνατότητα κρίθηκε ιδιαίτερα χρήσιμη, καθώς μπορεί να υποστηρίξει τη λήψη αποφάσεων σε μελλοντικές διαδικασίες σχεδιασμού μαθημάτων, προσφέροντας καλύτερη εικόνα για τη ζήτηση και τις ανάγκες των φοιτητών.

Για την ανάπτυξη του συστήματος προεγγραφών και την εφαρμογή του αλγορίθμου κατανομής των θέσεων στα μαθήματα περιορισμένης επιλογής, αναλύθηκαν οι εξής απαιτήσεις:

Λειτουργικές Απαιτήσεις του Συστήματος:

Διαχείριση φοιτητών: Το σύστημα πρέπει να επιτρέπει την καταχώρηση και διαχείριση των δεδομένων των φοιτητών, όπως ονοματεπώνυμο, UC Number , και προτιμώμενα μαθήματα.

Διαχείριση μαθημάτων: Η δυνατότητα του διαχειριστή για καταχώρηση, επεξεργασία και διαγραφή μαθημάτων από τη βάση δεδομένων.

Διαχείριση ανακοινώσεων: Η δυνατότητα του διαχειριστή για καταχώρηση, επεξεργασία και διαγραφή ανακοινώσεων από τη βάση δεδομένων.

Δημιουργία Λίστας Προτίμησης: Οι φοιτητές πρέπει να έχουν τη δυνατότητα να δημιουργήσουν τη δική τους λίστα με τις προτιμήσεις τους.

Αλγόριθμος κατανομής θέσεων: Ο αλγόριθμος πρέπει να αναθέτει τις θέσεις στους φοιτητές με βάση τις προκαθορισμένες προτεραιότητες, εξασφαλίζοντας τη δίκαιη κατανομή και με ακρίβεια ώστε να μην υπάρχουν λάθη.

Επικοινωνία με φοιτητές: Το σύστημα πρέπει να ενημερώνει τους φοιτητές για την κατάσταση των εγγραφών τους μέσω ανακοινώσεων.

Χειροκίνητη ανάθεση: Ο διαχειριστής θα έχει τη δυνατότητα να αναθέσει χειροκίνητα τα μαθήματα σε ένα φοιτητή.

Πληροφορίες φοιτητών: Ο διαχειριστής θα μπορεί να βλέπει όλες τις πληροφορίες του κάθε φοιτητή.

Στατιστικά προτιμήσεων: Ο διαχειριστής θα μπορεί να βλέπει σημαντικές πληροφορίες σχετικά με τη ζήτηση του κάθε μαθήματος, τη συνολική ζήτηση και τον αριθμό των φοιτητών που ζητούν τη πρώτη τους επιλογή για την εκπλήρωση της διπλωματικής τους εργασίας.

Τεχνικές Απαιτήσεις:

Αρχιτεκτονική: Το σύστημα θα υλοποιηθεί χρησιμοποιώντας τη μοντέρνα αρχιτεκτονική .NET MVC Core για τη δημιουργία μιας ευέλικτης και κλιμακούμενης εφαρμογής.

Βάση Δεδομένων: Θα χρησιμοποιηθεί σύστημα διαχείρισης βάσεων δεδομένων (MSSQL Server) για την αποθήκευση των δεδομένων των φοιτητών και των μαθημάτων.

Εφαρμογή αλγορίθμου: Ο αλγόριθμος κατανομής θα πρέπει να υλοποιηθεί χρησιμοποιώντας τεχνικές ικανοποίησης περιορισμών, όπως αυτές που παρέχουν τα εργαλεία MiniZinc και Google OR-Tools.

Φιλικότητα προς τον χρήστη: Η διεπαφή χρήστη πρέπει να είναι απλή και εύκολη στη χρήση, παρέχοντας σαφείς επιλογές για τους φοιτητές και τους διαχειριστές του συστήματος.

Απαιτήσεις του Αλγορίθμου:

Δίκαιη κατανομή θέσεων: Ο αλγόριθμος πρέπει να εξασφαλίσει ότι οι θέσεις κατανομής στα μαθήματα δεν βασίζονται στην χρονική προτεραιότητα, αλλά στην αναγνώριση της ζήτησης του μαθήματος και των προτεραιοτήτων των φοιτητών.

Ορθότητα: Ο αλγόριθμος πρέπει να εξασφαλίζει ότι οι φοιτητές κατανέμονται στα μαθήματα σύμφωνα με τις δηλωμένες προτιμήσεις τους, λαμβάνοντας υπόψη τους προκαθορισμένους περιορισμούς και κανόνες του συστήματος, όπως η μέγιστη χωρητικότητα των μαθημάτων και οι λίστες προτεραιότητας, διασφαλίζοντας μια δίκαιη και συνεπή κατανομή.

Αποτελεσματικότητα: Ο αλγόριθμος πρέπει να είναι αποδοτικός, με στόχο την ταχύτατη και σωστή κατανομή των θέσεων για όλους τους φοιτητές, ακόμα και όταν ο αριθμός των φοιτητών και των μαθημάτων είναι μεγάλος.

Η υλοποίηση αυτών των απαιτήσεων διασφαλίζει την ομαλή και αποτελεσματική λειτουργία του συστήματος, παρέχοντας μια δίκαιη διαδικασία για όλους τους φοιτητές.

2.4 Ανάλυση της ανάγκης για δίκαιη κατανομή των μαθημάτων

Η δίκαιη κατανομή των μαθημάτων στους φοιτητές αποτελεί έναν από τους βασικούς στόχους του συστήματος προεγγραφών, καθώς οι φοιτητές πρέπει να έχουν ίσες ευκαιρίες

πρόσβασης στα μαθήματα περιορισμένης επιλογής, χωρίς να επηρεάζονται από παράγοντες όπως η χρονική προτεραιότητα της εγγραφής ή άλλοι μη αντικειμενικοί παράγοντες. Για την εξασφάλιση δίκαιης κατανομής των θέσεων, η διαδικασία αυτή πρέπει να βασίζεται σε ένα σύνολο αντικειμενικών και διαφανών κριτηρίων.

Παράγοντες για δίκαιη κατανομή

Ζήτηση μαθημάτων: Τα μαθήματα περιορισμένης επιλογής συνήθως διαθέτουν περιορισμένες θέσεις, οπότε η ζήτηση από τους φοιτητές είναι υψηλή. Έτσι, το σύστημα πρέπει να καταγράφει και να αναλύει τη ζήτηση για κάθε μάθημα, ώστε να εξασφαλίζεται ότι οι θέσεις είναι ικανοποιητικές για κάθε μάθημα.

Προτιμήσεις φοιτητών: Οι φοιτητές πρέπει να έχουν τη δυνατότητα να δηλώσουν τη λίστα προτιμήσεων τους για τα μαθήματα, γεγονός που επιτρέπει στο σύστημα να κατατάξει τις επιλογές τους. Αν ένα μάθημα δεν είναι διαθέσιμο στην πρώτη τους επιλογή, το σύστημα πρέπει να εξετάσει τις επόμενες προτιμήσεις, διασφαλίζοντας ότι κανένας φοιτητής δεν θα μείνει χωρίς μάθημα.

Αναγνώριση των αναγκών και των προτεραιοτήτων: Ειδικές ανάγκες (π.χ. φοιτητές που χρειάζονται συγκεκριμένα μαθήματα για να ολοκληρώσουν το πρόγραμμα σπουδών τους, ανάγκη για εκπλήρωση της διπλωματικής τους εργασίας) πρέπει να λαμβάνονται υπόψη κατά την κατανομή των θέσεων. Η δίκαιη κατανομή δεν σημαίνει μόνο ίση μεταχείριση όλων των φοιτητών, αλλά και την εξασφάλιση ότι όλοι θα έχουν πρόσβαση στις αναγκαίες ευκαιρίες, ανάλογα με την προσωπική τους κατάσταση.

Αξιολόγηση πιστωτικών μονάδων: Η ομαδοποίηση με βάση τις πιστωτικές μονάδες που χρησιμοποιείται τώρα , είναι σημαντικό να μην αποτελεί παράγοντα για τη δίκαιη κατανομή.

Ελαστικότητα: Το σύστημα πρέπει να είναι ελαστικό, δηλαδή να επιτρέπει αλλαγές και τροποποιήσεις της κατανομής σε περίπτωση αλλαγών στις ανάγκες ή τις προτιμήσεις των φοιτητών (π.χ. αλλαγές στους περιορισμούς ή στην προσφορά θέσεων).

Αλγόριθμος για δίκαιη κατανομή

Ο αλγόριθμος κατανομής θέσεων θα βασίζεται σε τεχνικές ικανοποίησης περιορισμών (Constraint Satisfaction), ώστε να διασφαλίζεται ότι οι θέσεις θα κατανεμηθούν με αντικειμενικό και δίκαιο τρόπο. Η διαδικασία αυτή θα πρέπει να επιτρέπει τη μέγιστη δυνατή ικανοποίηση των προτιμήσεων των φοιτητών, ενώ ταυτόχρονα να τηρεί τους περιορισμούς

της ζήτησης και της προσφοράς θέσεων. Ο αλγόριθμος θα πρέπει να προσφέρει λύσεις που να είναι ισότιμες και να μην ευνοούν συγκεκριμένες ομάδες φοιτητών ή μεμονωμένα άτομα.

Συμπεράσματα

Η δίκαιη κατανομή των μαθημάτων είναι καθοριστική για την επιτυχία του συστήματος προεγγραφών, καθώς εξασφαλίζει ότι όλοι οι φοιτητές έχουν ίσες ευκαιρίες συμμετοχής στα μαθήματα. Χρησιμοποιώντας έναν αλγόριθμο που βασίζεται σε αντικειμενικά κριτήρια και σε σύγχρονες τεχνικές ικανοποίησης περιορισμών, το σύστημα μπορεί να διαχειριστεί τη ζήτηση των μαθημάτων με τρόπο που να διασφαλίζει την ισότητα και τη δικαιοσύνη για όλους τους φοιτητές.

Κεφάλαιο 3

Θεωρητικό Υπόβαθρο

3.1 Εισαγωγή στον προγραμματισμό ικανοποίησης περιορισμών

Ο προγραμματισμός ικανοποίησης περιορισμών (CSP) είναι ένας τομέας της τεχνητής νοημοσύνης και της θεωρίας υπολογισμού που ασχολείται με την επίλυση προβλημάτων όπου πρέπει να ικανοποιηθούν συγκεκριμένοι περιορισμοί ή συνθήκες. Στον προγραμματισμό ικανοποίησης περιορισμών, στόχος είναι να βρεθεί μια λύση σε ένα πρόβλημα που να πληροί όλους τους περιορισμούς που έχουν τεθεί, χωρίς να παραβιάζεται κανένας από αυτούς.

Ένα πρόβλημα CSP αποτελείται από:

Μεταβλητές: Αυτές είναι οι άγνωστες τιμές που πρέπει να προσδιοριστούν.

Domains (Πεδίο ορισμού): Κάθε μεταβλητή έχει ένα σύνολο από επιτρεπτές τιμές, που ονομάζεται το "domain" της.

Περιορισμοί: Οι περιορισμοί είναι οι συνθήκες που καθορίζουν ποιες συνδυαστικές τιμές για τις μεταβλητές είναι αποδεκτές ή όχι. Αυτοί οι περιορισμοί μπορεί να είναι αυστηροί (όπου πρέπει να ικανοποιούνται αυστηρά) ή ευέλικτοι (όπου επιτρέπονται κάποια σφάλματα).

Στη διαδικασία επίλυσης ενός προβλήματος CSP, η ιδέα είναι να βρεθεί ένας συνδυασμός τιμών για τις μεταβλητές που να ικανοποιεί όλους τους περιορισμούς του συστήματος. Αυτό συνήθως περιλαμβάνει την αναζήτηση μιας λύσης μέσω μεθόδων αναζήτησης ή/και αλγορίθμων βελτιστοποίησης.

Στο πλαίσιο αυτής της διπλωματικής εργασίας, ο προγραμματισμός ικανοποίησης περιορισμών χρησιμοποιείται για την επίλυση του προβλήματος κατανομής των μαθημάτων στους φοιτητές. Η χρήση του CSP επιτρέπει την ανάπτυξη ενός αλγορίθμου που εξασφαλίζει ότι οι θέσεις στα μαθήματα κατανεμηθούν δίκαια και σύμφωνα με τις προτιμήσεις των φοιτητών, ενώ παράλληλα τηρούνται οι περιορισμοί που προκύπτουν από τον αριθμό θέσεων κάθε μαθήματος και τις ειδικές ανάγκες που μπορεί να έχουν οι φοιτητές.

Σχετικά με την εφαρμογή του CSP στην προεγγραφή μαθημάτων
Ο προγραμματισμός ικανοποίησης περιορισμών είναι ιδανικός για την επίλυση προβλημάτων που περιλαμβάνουν περιορισμούς και απαιτήσεις, όπως η κατανομή περιορισμένων πόρων (θέσεων σε μαθήματα) μεταξύ διαφορετικών "αιτούντων" (φοιτητών). Με την εφαρμογή τεχνικών CSP, το σύστημα προεγγραφών μπορεί να εξετάσει κάθε πιθανό συνδυασμό κατανομής θέσεων και να βρει τη βέλτιστη λύση που εξασφαλίζει τη δίκαιη κατανομή σύμφωνα με τις προκαθορισμένες προτεραιότητες και περιορισμούς.

3.2 Αναπαράσταση του προβλήματος ως ΠΙΠ

Το πρόβλημα που καλούμαστε να λύσουμε αποτελεί ένα υβριδικό CSP-SAT πρόβλημα βελτιστοποίησης, καθώς συνδυάζει χαρακτηριστικά του Προγραμματισμού Ικανοποίησης Περιορισμών (CSP) και του Satisfiability (SAT). Το μοντέλο μας ενσωματώνει μεταβλητές με πεδία τιμών {0,1} (όπως στον SAT), καθώς και άλλες μεταβλητές που έχουν πιο σύνθετα πεδία τιμών, όπως τα ακέραια ή υποσύνολά τους. Η αναπαράσταση του προβλήματος διαρθρώνεται ως εξής:

1. Μεταβλητές και Πεδία Τιμών
Κύριες Μεταβλητές του Προβλήματος: Κάθε μεταβλητή στο πρόβλημά μας αντιστοιχεί σε έναν φοιτητή και ένα μάθημα (π.χ. Φοιτητής 1 – ΕΠΛ 412). Αν έχουμε 100 φοιτητές και 10 μαθήματα, τότε η σύνολο των μεταβλητών είναι 1000 (100 * 10). Κάθε μεταβλητή έχει πεδίο τιμών {0,1}, όπου:

0 σημαίνει ότι ο φοιτητής δεν έχει εγγραφεί στο μάθημα.
1 σημαίνει ότι ο φοιτητής έχει εγγραφεί στο μάθημα. Αυτές οι μεταβλητές εντάσσονται στο SAT κομμάτι του μοντέλου.

Βοηθητικές Μεταβλητές:

- Πίνακας Satisfaction: Ο πίνακας αυτός αντιπροσωπεύει το βαθμό ικανοποίησης του κάθε φοιτητή, υπολογισμένος βάσει των προτιμήσεών του.

- Μεταβλητή Minimum Satisfaction: Αυτή η μεταβλητή αποθηκεύει την ελάχιστη τιμή από τον πίνακα satisfaction.
- Μεταβλητή Total Satisfaction: Αυτή η μεταβλητή αποθηκεύει το άθροισμα όλων των στοιχείων του πίνακα satisfaction.

2. Περιορισμοί

Το πρόβλημα συνοδεύεται από τους εξής περιορισμούς:

- Αριθμός Μαθημάτων που Θέλει Ο Φοιτητής να Εγγραφεί: Κάθε φοιτητής πρέπει να εγγραφεί σε ακριβώς τον αριθμό των μαθημάτων που έχει δηλώσει.

- Μέγεθος ακροατηρίων: Ο αριθμός των φοιτητών που εγγράφονται σε κάθε μάθημα δεν μπορεί να ξεπερνά τη χωρητικότητα του μαθήματος.

- Περιορισμός Satisfaction: Ο υπολογισμός του πίνακα satisfaction βασίζεται στους βαθμούς ικανοποίησης των φοιτητών και πρέπει να τηρείται σωστά.

- Υπολογισμός Minimum Satisfaction: Ο υπολογισμός της ελάχιστης τιμής του πίνακα satisfaction.

- Προτεραιότητα Μαθημάτων: Ο κάθε φοιτητής μπορεί να εγγραφεί μόνο στα μαθήματα που περιλαμβάνονται στη λίστα προτεραιότητάς του (5 μαθήματα ανά φοιτητή).

- Υπολογισμός Total Satisfaction: Ο υπολογισμός του συνολικού satisfaction για όλους τους φοιτητές.

- Υποχρεωτική Εγγραφή στην 1η Επιλογή: Υπάρχει περιορισμός που απαιτεί την εγγραφή του κάθε φοιτητή στο μάθημα που έχει επιλέξει ως 1η επιλογή για τη διπλωματική του εργασία.

3. Αντικειμενοστρεφής Συνάρτηση

Η αντικειμενοστρεφής συνάρτηση του προβλήματος είναι σχεδιασμένη ώστε να μεγιστοποιεί δύο βασικούς στόχους:

Minimum Satisfaction: Η μεγιστοποίηση της ελάχιστης ικανοποίησης των φοιτητών.
Total Satisfaction: Η μεγιστοποίηση του συνολικού βαθμού ικανοποίησης όλων των φοιτητών.

Η αντικειμενοστρεφής συνάρτηση εκφράζεται ως εξής:
Maximize((minSatisfaction×numStudents)+totalSatisfaction)
Maximize((minSatisfaction×numStudents)+totalSatisfaction)
Όπου:
minSatisfaction είναι η ελάχιστη ικανοποίηση μεταξύ όλων των φοιτητών.
totalSatisfaction είναι το συνολικό άθροισμα των ικανοποιήσεων όλων των φοιτητών.
numStudents είναι ο αριθμός των φοιτητών.

4. Παράμετροι του Μοντέλου

Για την επίλυση του προβλήματος χρησιμοποιούνται οι εξής παράμετροι:

- Αριθμός φοιτητών: Ο συνολικός αριθμός φοιτητών.

- Αριθμός μαθημάτων: Ο συνολικός αριθμός μαθημάτων που προσφέρονται.

- Πίνακας με τα ονόματα των μαθημάτων.

- Πίνακας με τα μεγέθη των ακροατηρίων.

- Πίνακας με τον αριθμό των μαθημάτων που θέλει να εγγραφεί κάθε φοιτητής.

- Δυσδιάστατος πίνακας που περιέχει τις λίστες προτεραιότητας των φοιτητών (5 μαθήματα ανά φοιτητή).

- Boolean πίνακας που υποδεικνύει αν ο φοιτητής θέλει η 1η του επιλογή να είναι για τη διπλωματική του εργασία.

5. Λύση

Η λύση του προβλήματος δεν είναι απλώς feasible (δηλαδή πληρούνται όλοι οι περιορισμοί), αλλά optimal (δηλαδή βέλτιστη), καθώς επιτυγχάνεται η μεγιστοποίηση της αντικειμενοστρεφούς συνάρτησης, που συνδυάζει το minimum satisfaction και το total satisfaction.

3.3 Παρουσίαση του MiniZinc

Το MiniZinc είναι μια υψηλού επιπέδου γλώσσα μοντελοποίησης για προβλήματα βελτιστοποίησης και ικανοποίησης περιορισμών[2]. Παρέχει έναν τρόπο για να περιγράψουμε το πρόβλημα με σαφή και αφαιρετικό τρόπο, ανεξάρτητα από τον επιλυτή (solver) που θα χρησιμοποιηθεί για την επίλυσή του. Το MiniZinc υποστηρίζει πολλαπλούς επιλυτές όπως οι Gecode, Chuffed, και CBC, και επιτρέπει στον χρήστη να εστιάσει στη μοντελοποίηση χωρίς να χρειάζεται να ασχοληθεί με λεπτομέρειες υλοποίησης.

Η βασική φιλοσοφία του MiniZinc είναι να διαχωρίσει το μοντέλο από τα δεδομένα[1]. Το μοντέλο περιέχει τη δομή του προβλήματος, δηλαδή τις μεταβλητές, τους περιορισμούς και τη συνάρτηση στόχο, ενώ τα δεδομένα παρέχουν τις τιμές εισόδου για το συγκεκριμένο σενάριο.

Παράδειγμα 1: Κατανομή αριθμών

Ένα απλό πρόβλημα όπου θέλουμε να κατανείμουμε τρεις ακέραιες μεταβλητές έτσι ώστε να είναι διακριτές μεταξύ τους και να ελαχιστοποιείται το άθροισμά τους:

Code:

array[1..3] of var 1..10: x;

% Χειροκίνητος περιορισμός διαφορετικών τιμών
constraint x[1] != x[2];
constraint x[1] != x[3];
constraint x[2] != x[3];

solve minimize sum(i in 1..3)(x[i]);
Αυτό το παράδειγμα:
Δημιουργεί έναν πίνακα τριών μεταβλητών x[1..3], όπου κάθε τιμή μπορεί να είναι από 1 έως 10.
Ορίζει ότι κάθε ζεύγος μεταβλητών πρέπει να έχει διαφορετική τιμή, υλοποιώντας τον περιορισμό "όλες διαφορετικές" με ρητούς περιορισμούς ανισότητας.
Τέλος, επιλύει το πρόβλημα ελαχιστοποιώντας το άθροισμα των τιμών των μεταβλητών.

Παράδειγμα 2: Αντιστοίχιση φοιτητών σε μαθήματα

Code:
% Αριθμός φοιτητών
int: num_students = 2;
% Αριθμός μαθημάτων
int: num_courses = 3;

% Δυαδική μεταβλητή: enrolled[i,j] = 1 αν ο φοιτητής i εγγράφηκε στο μάθημα j, αλλιώς 0
array[1..num_students, 1..num_courses] of var 0..1: enrolled;

% Κάθε φοιτητής πρέπει να εγγραφεί σε ακριβώς ένα μάθημα
constraint
forall(i in 1..num_students)(
sum(j in 1..num_courses)(enrolled[i,j]) = 1
);

% Πίνακας με τη μέγιστη χωρητικότητα κάθε μαθήματος
array[1..num_courses] of int: capacities = [1, 1, 2];

% Η χωρητικότητα κάθε μαθήματος δεν πρέπει να ξεπεραστεί
constraint
forall(j in 1..num_courses)(
sum(i in 1..num_students)(enrolled[i,j]) <= capacities[j]
);

% Επίλυση προβλήματος χωρίς βελτιστοποίηση – αρκεί να βρεθεί μια έγκυρη ανάθεση
solve satisfy;

Παράδειγμα 3: Μια αρχική έκδοση του αλγορίθμου

Αυτή η αρχική έκδοση του αλγορίθμου για την κατανομή μαθημάτων σε φοιτητές, αποσκοπεί στη μεγιστοποίηση της ελάχιστης ικανοποίησης των φοιτητών, διασφαλίζοντας παράλληλα ότι τηρούνται οι βασικοί περιορισμοί του προβλήματος. Η προσέγγιση βασίζεται σε μια μαθηματική αναπαράσταση του προβλήματος, όπου κάθε φοιτητής πρέπει να εγγραφεί σε έναν προκαθορισμένο αριθμό μαθημάτων, σύμφωνα με τις προτιμήσεις του, χωρίς να υπερβαίνει τις διαθέσιμες θέσεις των μαθημάτων.

% ==========
% ΠΑΡΑΜΕΤΡΟΙ ΜΟΝΤΕΛΟΥ
% ==========

% 1. Αριθμός φοιτητών και μαθημάτων:
int: num_students; % Συνολικός αριθμός φοιτητών
int: num_courses; % Συνολικός αριθμός μαθημάτων

% 2. Ονόματα και χωρητικότητες μαθημάτων:
array[1..num_courses] of string: course_name; % Ονόματα των μαθημάτων
array[1..num_courses] of int: course_capacity; % Χωρητικότητα κάθε μαθήματος

% 3. Αριθμός μαθημάτων ανά φοιτητή:
array[1..num_students] of int: classes_to_take; % Πόσα μαθήματα επιθυμεί κάθε φοιτητής

% 4. Λίστα προτιμήσεων:

array[1..num_students, 1..5] of int: preferences; % Οι 5 κορυφαίες προτιμήσεις κάθε φοιτητή

% ==========

% ΜΕΤΑΒΛΗΤΕΣ ΑΠΟΦΑΣΗΣ

% ==========

% 1. Ανάθεση φοιτητών σε μαθήματα:

array[1..num_students, 1..num_courses] of var bool: assigned; % assigned[i,j] = 1 αν ο φοιτητής i πήρε το μάθημα j

% 2. Σκορ ικανοποίησης φοιτητών:

array[1..num_students] of var int: satisfaction; % Επίπεδο ικανοποίησης κάθε φοιτητή (μέγιστο 100)

% 3. Ελάχιστο επίπεδο ικανοποίησης:

var int: min_satisfaction; % Ελάχιστη ικανοποίηση ανάμεσα σε όλους τους φοιτητές

% ==========

% ΠΕΡΙΟΡΙΣΜΟΙ

% ==========

% 1. Πλήθος μαθημάτων ανά φοιτητή:

% Κάθε φοιτητής εγγράφεται ακριβώς σε όσα μαθήματα επιθυμεί (classes_to_take[i])

```
constraint forall(i in 1..num_students) (
   sum(j in 1..num_courses) (assigned[i, j]) = classes_to_take[i]
);
```

% 2. Περιορισμός χωρητικότητας μαθημάτων:

% Το σύνολο των εγγραφών ανά μάθημα δεν πρέπει να ξεπερνά το course_capacity[j]

```
constraint forall(j in 1..num_courses) (
   sum(i in 1..num_students) (assigned[i, j]) <= course_capacity[j]
);
```

% 3. Συμμόρφωση με τις λίστες προτιμήσεων:

% Οι φοιτητές μπορούν να εγγραφούν μόνο σε μαθήματα της λίστας τους

```
constraint forall(i in 1..num_students, j in 1..num_courses) (
   assigned[i, j] -> (exists(k in 1..5) (preferences[i, k] = j))
);

% 4. Υπολογισμός ικανοποίησης:
% Βασίζεται στη θέση του μαθήματος στη λίστα προτιμήσεων.
% 1η επιλογή → 100, 2η → 50, 3η → 25, 4η → 12, 5η → 6
constraint forall(i in 1..num_students) (
   satisfaction[i] = sum(j in 1..num_courses) (
      if assigned[i, j] then
         let {
            int: rank = min([k | k in 1..5 where preferences[i, k] = j])
         } in
         if exists(k in 1..5)(preferences[i, k] = j) then
            if rank == 1 then
               100
            elseif rank == 2 then
               50
            elseif rank == 3 then
               25
            elseif rank == 4 then
               12
            elseif rank == 5 then
               6
            else
               0
            endif
         else
            0
         endif
      else
         0
      endif
   )
);
```

```
% 5. Ελαχιστοποίηση ανισότητας:
% Η ελάχιστη ικανοποίηση πρέπει να είναι <= από κάθε φοιτητή
constraint forall(i in 1..num_students) (
   satisfaction[i] >= min_satisfaction
);

% ====================================
% ΣΥΝΑΡΤΗΣΗ ΣΤΟΧΟΥ
% ====================================

% Στόχος: Μέγιστη ελάχιστη ικανοποίηση
% Δηλαδή, να αυξηθεί όσο το δυνατόν περισσότερο η χαμηλότερη ικανοποίηση φοιτητή
solve maximize min_satisfaction;
```

Αυτό το μοντέλο είναι μια πρώτη προσπάθεια για την υλοποίηση του προβλήματος, επικεντρωμένη στη διατήρηση υψηλού επιπέδου ικανοποίησης για όλους τους φοιτητές. Ωστόσο, δεν λαμβάνει ακόμη υπόψη ορισμένους άλλους σημαντικούς παράγοντες, όπως η προτίμηση για διπλωματικές εργασίες ή η βελτιστοποίηση του συνολικού επιπέδου ικανοποίησης, τα οποία εισάγονται σε πιο εξελιγμένες εκδόσεις του αλγορίθμου.

3.4 Εργαλεία και IDE

Το MiniZinc διαθέτει το δικό του IDE (MiniZinc IDE), το οποίο υποστηρίζει:

- Γραφικό περιβάλλον για τη σύνταξη μοντέλων.
- Εκτέλεση μοντέλων με διαφορετικούς επιλυτές.
- Οπτικοποίηση αποτελεσμάτων.

Η εκμάθησή του είναι σχετικά απλή για άτομα με βασικές γνώσεις προγραμματισμού, και ενδείκνυται ιδιαίτερα για ακαδημαϊκή χρήση και πρωτότυπη μοντελοποίηση προβλημάτων ικανοποίησης περιορισμών.

Μετάβαση στα Google OR-Tools

Αν και το MiniZinc αποτελεί εξαιρετικό εργαλείο για τη μοντελοποίηση και κατανόηση προβλημάτων CSP, κατά τη διάρκεια της ανάπτυξης του έργου προχωρήσαμε στη χρήση των

Google OR-Tools για την τελική υλοποίηση του αλγορίθμου. Οι βασικοί λόγοι για αυτή την απόφαση ήταν:

- **Υποστήριξη από την ASP.NET**: Τα OR-Tools διαθέτουν βιβλιοθήκες για πολλές γλώσσες προγραμματισμού, μεταξύ των οποίων και η C#, γεγονός που διευκολύνει την ενσωμάτωσή τους σε web εφαρμογές βασισμένες σε ASP.NET.
- **Ευκολία στη χρήση μέσω κώδικα**: Επιτρέπουν άμεση προγραμματιστική υλοποίηση και δυναμική παραμετροποίηση, χωρίς ανάγκη για ξεχωριστή γλώσσα μοντελοποίησης.
- **Επιδόσεις**: Τα OR-Tools είναι ιδιαίτερα αποδοτικά και βελτιστοποιημένα για μεγάλα προβλήματα.

Ως εκ τούτου, η χρήση των OR-Tools κρίθηκε πιο πρακτική και αποτελεσματική για την παραγωγική φάση του έργου, ενώ το MiniZinc αξιοποιήθηκε κυρίως στην αρχική φάση μοντελοποίησης και πειραματισμού.

Παρακάτω είναι ένας συγκριτικός πίνακας μεταξύ **MiniZinc** και **Google OR-Tools**, εστιάζοντας σε βασικά χαρακτηριστικά, δυνατότητες και πρακτική χρηστικότητα:

Χαρακτηριστικό	MiniZinc	Google OR-Tools
Τύπος εργαλείου	Γλώσσα μοντελοποίησης περιορισμών	Framework βελτιστοποίησης και επίλυσης περιορισμών
Υποστήριξη γλωσσών	Αυτόνομη (με υποστήριξη σε Python/C++)	Πλήρης υποστήριξη σε Python, C++, Java, C#, κ.ά.
Ενσωμάτωση σε εφαρμογές	Περιορισμένη – κυρίως για ερευνητικούς/ακαδημαϊκούς σκοπούς	Πολύ καλή – εύκολη ενσωμάτωση σε πραγματικές εφαρμογές όπως ASP.NET
Καμπύλη εκμάθησης	Μέτρια – απαιτεί εξοικείωση με μοντελοποίηση CSP	Ευκολότερη – βασισμένη σε δημοφιλείς γλώσσες προγραμματισμού
Υποστήριξη SAT/CP/Solver APIs	Χρήση εξωτερικών solvers (π.χ. Gecode, Chuffed, CBC)	Περιλαμβάνει έτοιμους και ενσωματωμένους solvers (CP-SAT, MIP κ.λπ.)
Απόδοση (performance)	Πολύ καλή με ισχυρούς solvers	Εξαιρετική, ειδικά με τον CP-SAT για μεγάλες κλίμακες
Ευκολία debugging	Περιορισμένη (μέσω του MiniZinc IDE)	Πολύ καλή – αξιοποιεί debugging εργαλείων της γλώσσας επιλογής
Τεκμηρίωση / Κοινότητα	Καλή, αλλά μικρότερη	Πολύ εκτενής τεκμηρίωση και ενεργή κοινότητα
Καταλληλότητα για .NET	Όχι άμεσα κατάλληλο	Άψογη ενσωμάτωση με C# και ASP.NET
Κατάλληλο για παραγωγικές λύσεις	Περισσότερο για πρωτότυπα και πειράματα	Ιδανικό για παραγωγικά συστήματα

Κεφάλαιο 4

Σχεδίαση του Συστήματος

4.1 Τεχνολογίες και εργαλεία που χρησιμοποιήθηκαν

Κατά την υλοποίηση του συστήματος, επιλέχθηκαν τεχνολογίες και εργαλεία που εξασφαλίζουν ευελιξία, αξιοπιστία και συμβατότητα με διαφορετικά περιβάλλοντα ανάπτυξης και φιλοξενίας. Πιο συγκεκριμένα:

- ASP.NET Core MVC: Επιλέχθηκε για τη δυνατότητά του να εκτελείται σε πολλαπλές πλατφόρμες και να υποστηρίζει φιλοξενία τόσο σε IIS (Windows Server) όσο και σε Apache/Nginx (Linux Server). Αυτή η ευελιξία καθιστά το σύστημα εύκολα συνδέσιμο με τους υφιστάμενους διακομιστές του πανεπιστημίου, ανεξάρτητα από το λειτουργικό σύστημα που χρησιμοποιείται.

- GitHub: Χρησιμοποιήθηκε για τον έλεγχο εκδόσεων (version control) και τη συνεργατική ανάπτυξη του λογισμικού. Η χρήση αποθετηρίων Git διευκόλυνε την παρακολούθηση αλλαγών και την ασφαλή αποθήκευση του πηγαίου κώδικα. https://github.com/KyrAnt02/bachelor-thesis-course-prebooking

- Kanban Flow: Υιοθετήθηκε ως μεθοδολογία διαχείρισης έργου, με στόχο τη σαφή οργάνωση των εργασιών και τη βελτιστοποίηση της ροής ανάπτυξης μέσω απλών και οπτικών πινάκων (Kanban boards).

- Justinmind: Αξιοποιήθηκε κατά το αρχικό στάδιο σχεδιασμού για την πρωτοτυποποίηση ιδεών και τη χαρτογράφηση των βασικών λειτουργιών του συστήματος. Η δυνατότητα δημιουργίας διαδραστικών διαγραμμάτων βοήθησε στην κατανόηση και οπτικοποίηση της συνολικής αρχιτεκτονικής και της ροής της εφαρμογής.

4.2 Σχεδίαση της αρχιτεκτονικής του συστήματος (.NET MVC)

Η αρχιτεκτονική **Model-View-Controller (MVC)** αποτελεί μία από τις πιο διαδεδομένες και αξιόπιστες προσεγγίσεις για την ανάπτυξη σύγχρονων διαδικτυακών εφαρμογών[4]. Η βασική της φιλοσοφία βασίζεται στον διαχωρισμό των ευθυνών της εφαρμογής σε τρία διακριτά επίπεδα, με στόχο την καλύτερη οργάνωση του κώδικα, τη διευκόλυνση της συντήρησης και την ενίσχυση της επεκτασιμότητας του συστήματος.

Στο οικοσύστημα της Microsoft, η MVC υλοποιείται μέσω του ASP.NET MVC Framework, το οποίο αξιοποιεί τη γλώσσα C# και παρέχει ένα ισχυρό περιβάλλον ανάπτυξης. Μέσω αυτής της αρχιτεκτονικής, ο προγραμματιστής μπορεί να διαχειρίζεται την επιχειρησιακή λογική, την παρουσίαση και την αλληλεπίδραση του χρήστη με τρόπο που είναι σαφής, ξεκάθαρος και επαναχρησιμοποιήσιμος.

Τα τρία βασικά συστατικά της MVC είναι τα εξής:

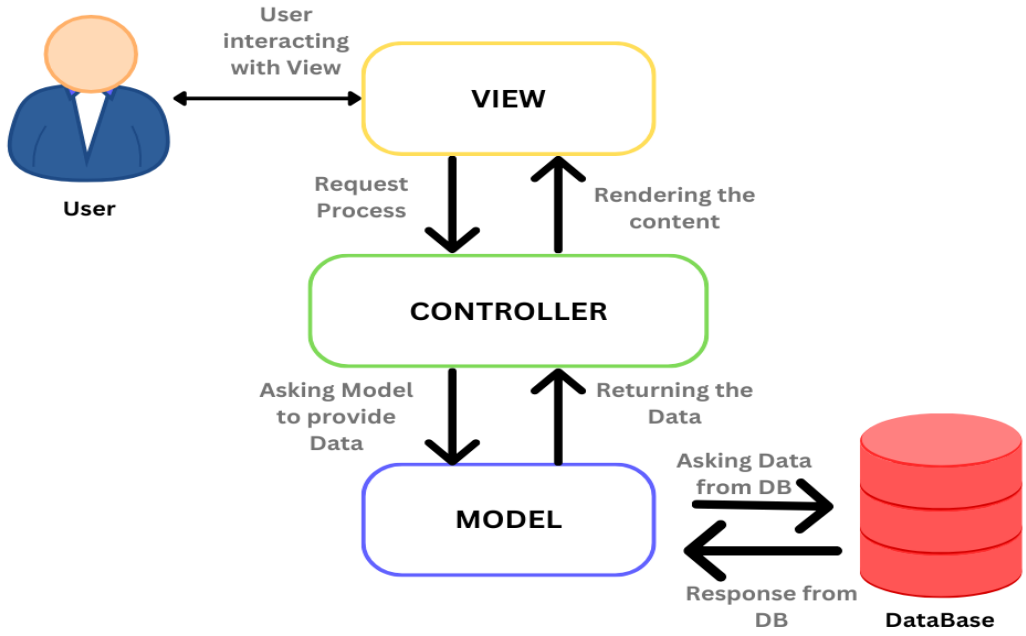
Model: Το Model αποτελεί την καρδιά της επιχειρησιακής λογικής της εφαρμογής[3]. Περιλαμβάνει τις δομές δεδομένων, τους κανόνες επεξεργασίας, και τις διεπαφές με τη βάση δεδομένων. Για παράδειγμα, σε ένα σύστημα κατανομής μαθημάτων, τα μοντέλα περιγράφουν έννοιες όπως "Φοιτητής", "Μάθημα", "Επιλογές φοιτητή", και ενσωματώνουν κανόνες όπως η συσχέτιση φοιτητών με μαθήματα ή οι περιορισμοί στον αριθμό εγγραφών ανά μάθημα. Επιπλέον, μέσω του Entity Framework, το Model επιτρέπει την ευέλικτη αλληλεπίδραση με τις υποκείμενες βάσεις δεδομένων χωρίς την ανάγκη άμεσης συγγραφής SQL.

View: Οι Views είναι υπεύθυνες για την παρουσίαση των δεδομένων στον τελικό χρήστη. Πρόκειται για σελίδες που δημιουργούνται με χρήση HTML και Razor syntax, και δέχονται δεδομένα από τον Controller. Οι Views είναι δυναμικές και ευέλικτες, επιτρέποντας την

εύκολη απεικόνιση λιστών μαθημάτων, προτιμήσεων φοιτητών, ή αποτελεσμάτων κατανομής. Η αποσύνδεση της λογικής της παρουσίασης από τα υπόλοιπα μέρη της εφαρμογής εξασφαλίζει ότι το front-end μπορεί να εξελίσσεται ανεξάρτητα από το back-end.

Controller: Ο Controller παίζει το ρόλο του συντονιστή. Επεξεργάζεται τα αιτήματα του χρήστη (HTTP requests), ανακτά ή επεξεργάζεται δεδομένα μέσω των Models, και επιλέγει ποια View θα εμφανιστεί στον χρήστη. Ουσιαστικά, λειτουργεί ως η "κόλλα" που ενώνει τα υπόλοιπα μέρη, χειριζόμενος τη ροή της εφαρμογής και τις διαδράσεις με τον χρήστη. Μέσα από τα Controllers γίνεται και η σύνδεση με εξωτερικά εργαλεία ή βιβλιοθήκες (π.χ. OR-Tools), ενορχηστρώνοντας την επίλυση προβλημάτων και την απόδοση αποτελεσμάτων.

Αξίζει να σημειωθεί πως η αρχιτεκτονική MVC ευνοεί τις βέλτιστες πρακτικές ανάπτυξης λογισμικού, όπως η ανάπτυξη βάσει δοκιμών (TDD), η καλύτερη αναγνωσιμότητα του κώδικα, και η παράλληλη ανάπτυξη μεταξύ ομάδων frontend και backend. Η επεκτασιμότητα και η συντηρησιμότητά της την καθιστούν ιδανική για συστήματα με αυξημένη πολυπλοκότητα, όπως ένα σύστημα κατανομής μαθημάτων βασισμένο σε αλγορίθμους βελτιστοποίησης.



4.3 Περιγραφή των Models

Model: User

Το μοντέλο User αντιπροσωπεύει έναν φοιτητή ή τον διαχειριστή του συστήματος. Περιέχει βασικές πληροφορίες ταυτοποίησης και χρησιμοποιείται για login και προσωποποιημένες λειτουργίες.

- Βασικά πεδία:
    - UserId (Primary Key) όπου είναι το UC Number του φοιτητή
    - FirstName
    - LastName
    - Email
    - NumberOfCourses
    - Password
    - Role (Student or Secretary)
- Σχέσεις:
    - Έχει μία PriorityList
    - Έχει μία AssignedList

  Εκτός από τον διαχειριστή του συστήματος που δεν έχει κάποια σχέση

Model: PriorityList

Το μοντέλο PriorityList περιέχει τις 5 προτιμήσεις μαθημάτων κάθε φοιτητή, με σειρά προτεραιότητας.

- Βασικά πεδία:
    - ListId (Primary Key)
    - UserId (Foreign Key)
    - FirstChoice
    - SecondChoice
    - ThirdChoice
    - FourthChoice
    - FifthChoice
    - IsFirstChoiceForBachelorThesis μόνο όταν χρειάζεται ένα μάθημα για την εκπλήρωση της διπλωματικής του εργασίας
    - professorBachelorThesis
    - TitleBachelorThesis

- Σχέσεις:
    - Συνδέεται με έναν User μέσω του UserId

Model: AssignedList

Το AssignedList καταγράφει τα μαθήματα στα οποία τελικά έχει εγγραφεί ο φοιτητής μετά την εκτέλεση του αλγορίθμου.

- Βασικά πεδία:
    - AssignedListId (Primary Key)
    - UserId (Foreign Key)
    - Courses (Λίστα με τα μαθήματα όπου εν τέλει εγγράφηκε ο φοιτητής)
- Σχέσεις:
    - Συνδέεται με User

Model: Course

Το μοντέλο Course αντιπροσωπεύει ένα πανεπιστημιακό μάθημα στο σύστημα. Κάθε μάθημα έχει όνομα, χωρητικότητα και κωδικό.

- **Βασικά πεδία:**
    - Code (Primary Key)
    - Title
    - Instructor
    - Capacity
    - DaysHours
    - Prerequisites
    - Description
- **Σχέσεις:**
    - Αντιστοιχίζεται με φοιτητές μέσω των PriorityList και AssignedList.

Model: Announcement

Το Announcement χρησιμοποιείται από τον διαχειριστή του συστήματος για την αποστολή ανακοινώσεων προς τους φοιτητές.

- Βασικά πεδία:

- AnnouncementId (Primary Key)
- AnnouncementTitle
- AnnouncementDescription
- dateTime

- Σχέσεις:
  - Δεν έχει άμεσες σχέσεις αλλά εμφανίζεται στους χρήστες στο Dashboard.

4.4 Περιγραφή των Views

Η εφαρμογή περιλαμβάνει διακριτά Views για δύο βασικές κατηγορίες χρηστών: τους φοιτητές και τον διαχειριστή. Κάθε View σχεδιάστηκε με βάση τις ανάγκες και τα δικαιώματα πρόσβασης του αντίστοιχου ρόλου.

Views:

1. User/Login

- Περιγραφή: Αρχική σελίδα της εφαρμογής, εμφανίζει τη φόρμα για το User Id και το Password του χρήστη , αλλά και την επιλογή να κάνει Register.
- Χρήστες: Όλοι οι επισκέπτες της εφαρμογής.
- Controller/Action: UserController -> Login() GET and POST

2. User/Register

- Περιγραφή: Εμφανίζει τη φόρμα για την εγγραφή του χρήστη.Πρέπει να συμπληρώσει το User Id , το Password του ,Όνομα,Επίθετο και Email
- Χρήστες: Όλοι οι επισκέπτες της εφαρμογής.
- Controller/Action: UserController -> Register() GET and POST

Views Φοιτητών:

1. User/Dashboard

- Περιγραφή: Κύρια σελίδα της εφαρμογής για τους φοιτητές, εμφανίζει στο πάνω μέρος ένα navigation bar για όλες τις σελίδες που μπορεί να μεταβεί ένας φοιτητής.Επίσης ο φοιτητής βλέπει τις πιο πρόσφατες ανακοινώσεις , τη λίστα προτιμήσεων του και την λίστα με τα μαθήματα που εν τέλει εγγράφηκε.
- Χρήστες: Φοιτητές.
- Controller/Action: UserController -> Dashboard() GET

2. User/Courses

- Περιγραφή: Σε αυτή τη σελίδα υπάρχει η λίστα με όλα τα μαθήματα που προσφέρονται το τρέχον εξάμηνο και μπορεί να επιλέξει ένα για να δει περισσότερες πληροφορίες.
- Χρήστες: Φοιτητές.
- Controller/Action: UserController -> Courses() GET

3. User/Course/412

- Περιγραφή: Σε αυτή τη σελίδα υπάρχουν όλες οι πληροφορίες για το συγκεκριμένο μάθημα.
- Χρήστες: Φοιτητές.
- Controller/Action: UserController -> Course/412() GET

4. User/Announcements

- Περιγραφή: Σε αυτή τη σελίδα υπάρχει η λίστα με όλες τις ανακοινώσεις και μπορεί να επιλέξει μία για να δει περισσότερες πληροφορίες.
- Χρήστες: Φοιτητές.
- Controller/Action: UserController -> Announcements() GET

5. User/AnnouncementDetails/ANN001

- Περιγραφή: Σε αυτή τη σελίδα υπάρχουν όλες οι πληροφορίες για τη συγκεκριμένη ανακοίνωση.
- Χρήστες: Φοιτητές.
- Controller/Action: UserController -> AnnouncementDetails/ANN001() GET

6.User/Details

- Περιγραφή: Σε αυτή τη σελίδα υπάρχουν όλες οι πληροφορίες σχετικά με τον χρήστη.Από εδώ μπορεί να αλλάξει τα στοιχεία του ή να δημιουργήσει ή να αλλάξει τη λίστα προτιμήσεων του.
- Χρήστες: Φοιτητές.
- Controller/Action: UserController -> Details() GET

7.User/Edit

- Περιγραφή: Σε αυτή τη σελίδα μπορεί να αλλάξει τις προσωπικές πληροφορίες του όλες οι πληροφορίες σχετικά με τον χρήστη.
- Χρήστες: Φοιτητές.
- Controller/Action: UserController -> Edit() GET and POST

8.PriorityList/Create

- Περιγραφή: Σε αυτή τη σελίδα ο φοιτητής δημιουργεί τη λίστα προτιμήσεων του.
- Χρήστες: Φοιτητές.
- Controller/Action: PriorityListController -> Create() GET and POST

9.PriorityList/Edit

- Περιγραφή: Σε αυτή τη σελίδα ο φοιτητής μπορεί να αλλάξει τη λίστα προτιμήσεων του.
- Χρήστες: Φοιτητές.
- Controller/Action: PriorityListController -> Edit() GET and POST

Views Διαχειριστή:

1. User/SecretaryDashboard

- Περιγραφή: Κύρια σελίδα της εφαρμογής για τον διαχειριστή, εμφανίζει στο πάνω μέρος ένα navigation bar για όλες τις σελίδες που μπορεί να μεταβεί.Επίσης βλέπει κάποια στατιστικά με βάση τις λίστες προτιμήσεων των φοιτητών.

- Χρήστες: Διαχειριστής Συστήματος.
- Controller/Action: UserController -> SecretaryDashboard() GET

2. User/Index

- Περιγραφή: Σε αυτή τη σελίδα ο διαχειριστής του συστήματος βλέπει όλους τους φοιτητές , μπορεί να αναζητήσει για κάποιο συγκεκριμένο φοιτητή και μπορεί να επιλέξει ένα συγκεκριμένο φοιτητή για να δει όλες τις πληροφορίες του.
- Χρήστες: Διαχειριστής Συστήματος.
- Controller/Action: UserController -> Index() GET

3. User/AllDetails?userId={userid}

- Περιγραφή: Σε αυτή τη σελίδα ο διαχειριστής βλέπει όλες τις πληροφορίες ενός φοιτητή όπου μπορεί και να αλλάξει την λίστα με τα μαθήματα που εγγράφηκε ένας φοιτητής.
- Χρήστες: Διαχειριστής Συστήματος.
- Controller/Action: UserController -> AllDeatils?userId={userid}() GET

4. User/StudentsBT

- Περιγραφή: Σε αυτή τη σελίδα ο διαχειριστής βλέπει όλους τους φοιτητές όπου θέλουν την πρώτη τους επιλογή για την εκπλήρωση της διπλωματικής τους εργασίας, μπορεί να αναζητήσει για κάποιο συγκεκριμένο φοιτητή.
- Χρήστες: Διαχειριστής Συστήματος.
- Controller/Action: UserController -> StudentsBT() GET

5. User/RunCSPPage

- Περιγραφή: Σε αυτή τη σελίδα ο διαχειριστής θα έρθει για να ενεργοποιήσει τον αλγόριθμο έτσι ώστε να ανατεθούν τα μαθήματα στους φοιτητές.
- Χρήστες: Διαχειριστής Συστήματος.
- Controller/Action: UserController -> RunCSPPage() GET

6. Course/Index

- Περιγραφή: Σε αυτή τη σελίδα ο διαχειριστής βλέπει τη λίστα με όλα τα μαθήματα του τρέχον εξαμήνου και μπορεί να μεταβεί στη σελίδα για δημιουργία νέου μαθήματος , να μεταβεί στη σελίδα για περισσότερες πληροφορίες για ένα μάθημα , να αλλάξει τις πληροφορίες ενός μαθήματος και να διαγράψει ένα μάθημα.
- Χρήστες: Διαχειριστής Συστήματος.
- Controller/Action: CourseController -> Index() GET

7. Course/Create

- Περιγραφή: Σε αυτή τη σελίδα ο διαχειριστής μπορεί να δημιουργήσει ένα καινούργιο μάθημα.
- Χρήστες: Διαχειριστής Συστήματος.
- Controller/Action: CourseController -> Create() GET and POST

8. Course/Details/412

- Περιγραφή: Σε αυτή τη σελίδα ο διαχειριστής μπορεί να δει όλες τις πληροφορίες ενός μαθήματος.
- Χρήστες: Διαχειριστής Συστήματος.
- Controller/Action: CourseController -> Details/412() GET

9. Course/Edit/412

- Περιγραφή: Σε αυτή τη σελίδα ο διαχειριστής μπορεί να αλλάξει τις πληροφορίες ενός μαθήματος.
- Χρήστες: Διαχειριστής Συστήματος.
- Controller/Action: CourseController -> Edit/412() GET and POST

10. Announcement/Index

- Περιγραφή: Σε αυτή τη σελίδα ο διαχειριστής βλέπει τη λίστα με όλες τις ανακοινώσεις και μπορεί να μεταβεί στη σελίδα για δημιουργία νέας ανακοίνωσης, να μεταβεί στη σελίδα για περισσότερες πληροφορίες για μία ανακοίνωση, να αλλάξει τις πληροφορίες μίας ανακοίνωσης και να διαγράψει μία ανακοίνωση.
- Χρήστες: Διαχειριστής Συστήματος.

- Controller/Action: AnnouncementController -> Index() GET

11. Announcement/Create

- Περιγραφή: Σε αυτή τη σελίδα ο διαχειριστής μπορεί να δημιουργήσει μια καινούργια ανακοίνωση.
- Χρήστες: Διαχειριστής Συστήματος.
- Controller/Action: AnnouncementController -> Create() GET and POST

12. Announcement/Details/ANN001

- Περιγραφή: Σε αυτή τη σελίδα ο διαχειριστής μπορεί να δει όλες τις πληροφορίες μιας ανακοίνωσης.
- Χρήστες: Διαχειριστής Συστήματος.
- Controller/Action: AnnouncementController -> Details/ANN001() GET

13. Announcement/Edit/ANN001

- Περιγραφή: Σε αυτή τη σελίδα ο διαχειριστής μπορεί να αλλάξει τις πληροφορίες μιας ανακοίνωσης.
- Χρήστες: Διαχειριστής Συστήματος.
- Controller/Action: AnnouncementController -> Edit/ANN001() GET and POST

14. AssignedList/Create/{userId}

- Περιγραφή: Σε αυτή τη σελίδα ο διαχειριστής μπορεί να αναθέσει μαθήματα σε ένα συγκεκριμένο φοιτητή.
- Χρήστες: Διαχειριστής Συστήματος.
- Controller/Action: AssignedListController -> Create/{userId}() GET and POST

Εκτέλεση αλγορίθμου:

Στον φάκελο Services της εφαρμογής μας, έχει υλοποιηθεί μια εξειδικευμένη κλάση υπεύθυνη για την εκτέλεση του αλγορίθμου κατανομής φοιτητών στα μαθήματα, αξιοποιώντας την πλατφόρμα **Google OR-Tools**. Η κλάση αυτή αποτελεί τον πυρήνα της λογικής βελτιστοποίησης και περιλαμβάνει μεθόδους για τη μετατροπή των δεδομένων από

τη βάση (όπως προτιμήσεις φοιτητών, χωρητικότητες μαθημάτων και απαιτήσεις εγγραφής) σε μεταβλητές, περιορισμούς και αντικειμενικές συναρτήσεις κατάλληλες για το εργαλείο. Στο εσωτερικό της, δημιουργείται ένα CpModel όπου ορίζονται οι binary decision variables για κάθε πιθανή αντιστοίχιση φοιτητή-μαθήματος, οι constraints που διασφαλίζουν τη σωστή και δίκαιη κατανομή, και η objective function που μεγιστοποιεί την ικανοποίηση των φοιτητών. Με την ολοκλήρωση της μοντελοποίησης, χρησιμοποιείται ο CpSolver για την αναζήτηση της βέλτιστης λύσης. Η κλάση αυτή είναι πλήρως ενσωματωμένη με το υπόλοιπο σύστημα μέσω Dependency Injection, και καλείται από τον UserController όταν ο διαχειριστής επιλέξει την εκτέλεση του αλγορίθμου.

Κεφάλαιο 5

Υλοποίηση του Συστήματος

5.1 Περιγραφή της διαδικασίας ανάπτυξης του συστήματος

Η ανάπτυξη του συστήματος πραγματοποιήθηκε με βάση τις αρχές της αρχιτεκτονικής Model-View-Controller (MVC), αξιοποιώντας το περιβάλλον του ASP.NET Core MVC για την κατασκευή της εφαρμογής και το Entity Framework για την επικοινωνία με τη βάση δεδομένων. Παράλληλα, για την επίλυση του προβλήματος κατανομής μαθημάτων, χρησιμοποιήθηκαν τα Google OR-Tools, τα οποία προσφέρουν δυνατότητες μοντελοποίησης προβλημάτων βελτιστοποίησης μέσα σε περιβάλλον .NET.

Η διαδικασία ανάπτυξης ακολούθησε την προσέγγιση modular και iterative development, ξεκινώντας από τον σχεδιασμό των βασικών μονάδων και προχωρώντας σε σταδιακή ολοκλήρωση και δοκιμή των λειτουργιών του συστήματος. Η εργασία χωρίστηκε σε κύριες φάσεις:

- Δημιουργία της βάσης δεδομένων και των Models: Ορίστηκαν οι βασικές οντότητες του συστήματος, όπως οι χρήστες (User), τα μαθήματα (Course), οι λίστες προτεραιότητας (PriorityList), οι αναθέσεις (AssignedList) και οι ανακοινώσεις (Announcement). Οι συσχετίσεις και τα primary/foreign keys καθορίστηκαν μέσω του Entity Framework, ώστε να υποστηρίζεται η λειτουργική αλληλεπίδραση των δεδομένων.
- Υλοποίηση του μηχανισμού authentication και authorization: Χρησιμοποιήθηκε το authorize system της .NET για την εγγραφή, είσοδο και διαφοροποίηση των χρηστών με ρόλους (φοιτητές, διαχειριστής).
- Σχεδίαση Views και Controllers: Δημιουργήθηκαν ξεχωριστές όψεις για φοιτητές και διαχειριστή, και αναπτύχθηκαν οι ανάλογοι Controllers που χειρίζονται τη λογική των

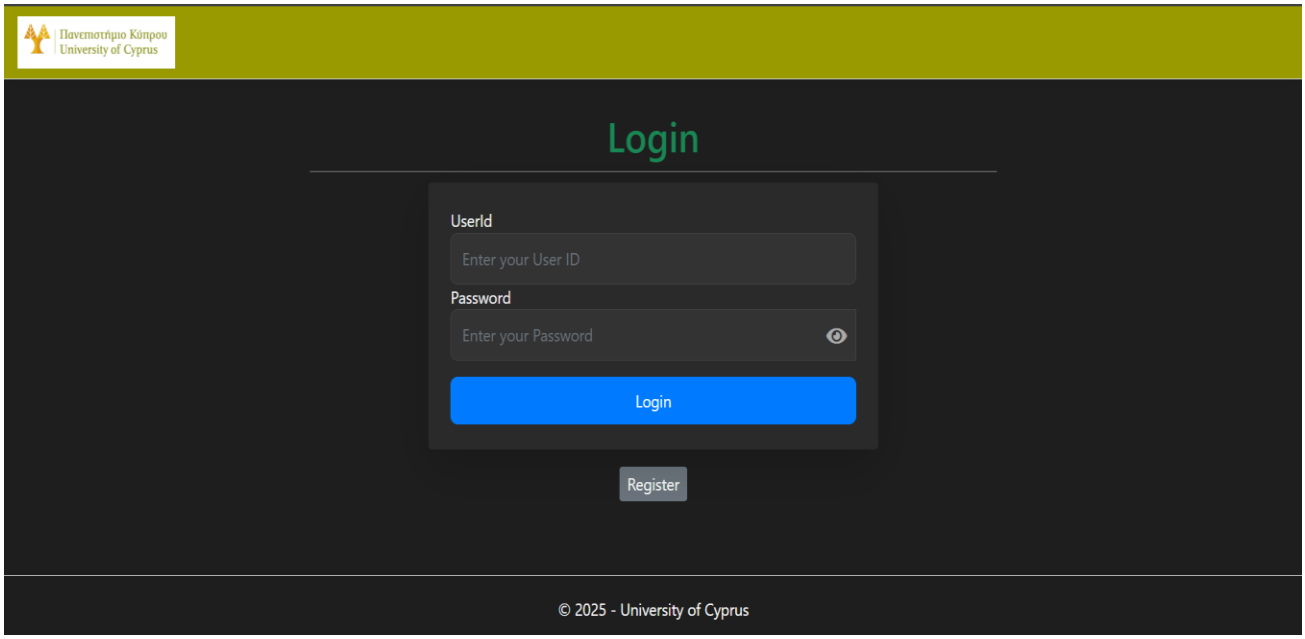
αιτημάτων, όπως η καταχώριση προτιμήσεων, η ανάρτηση ανακοινώσεων, και η εκκίνηση του αλγορίθμου.

- Ενσωμάτωση του αλγορίθμου Google OR-Tools: Η λογική του αλγορίθμου τοποθετήθηκε σε ξεχωριστή Service class, η οποία λαμβάνει τις απαιτούμενες παραμέτρους από τη βάση δεδομένων, μοντελοποιεί το πρόβλημα ως Constraint Optimization Problem, και επιστρέφει τις βέλτιστες αναθέσεις στους φοιτητές. Η χρήση των OR-Tools επιλέχθηκε τελικά έναντι του MiniZinc, λόγω της καλύτερης ενσωμάτωσης στο .NET οικοσύστημα.
- Δοκιμές και διορθώσεις: Καθ' όλη τη διάρκεια της ανάπτυξης εφαρμόστηκαν λειτουργικές δοκιμές, κυρίως unit tests και χειροκίνητες δοκιμές ροής από την πλευρά των χρηστών, ώστε να διασφαλιστεί η ορθότητα των επιλογών και η αντοχή του συστήματος σε εσφαλμένες εισόδους ή περιπτώσεις χωρίς λύση.

Το τελικό αποτέλεσμα είναι ένα πλήρως λειτουργικό σύστημα, το οποίο υποστηρίζει με ακρίβεια την εισαγωγή προτιμήσεων από φοιτητές, την παρακολούθηση των αιτήσεων από το διαχειριστή και την αυτόματη, δίκαιη και βέλτιστη κατανομή των μαθημάτων.

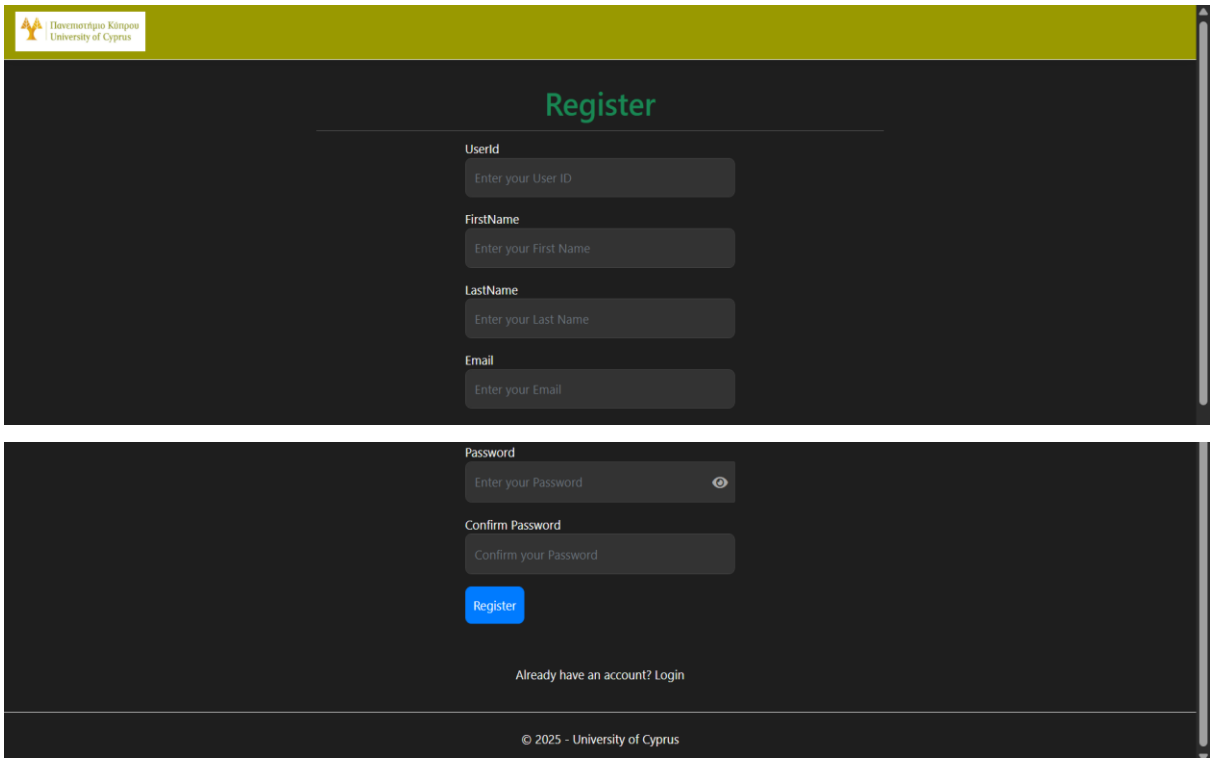
5.2 Εφαρμογή της διεπαφής χρήστη και της λογικής του συστήματος

Στην ενότητα αυτή παρουσιάζεται η διεπαφή χρήστη της εφαρμογής, μέσα από αναλυτικά screenshots που αναδεικνύουν τη ροή των λειτουργιών για τους δύο βασικούς ρόλους του συστήματος: τον φοιτητή και το διαχειριστή. Θα εξεταστούν τα βήματα που ακολουθεί κάθε χρήστης, από τη σύνδεση στο σύστημα έως την αλληλεπίδραση με τις βασικές δυνατότητες, όπως η καταχώριση προτιμήσεων και η εκτέλεση της κατανομής. Μέσα από τη ροή των οθονών γίνεται εμφανής η οργάνωση, η ευχρηστία και η λειτουργικότητα του συστήματος.



Εικόνα 7 – Διεπαφή Σύνδεσης Χρήστη

Στην εικόνα 7 έχουμε την αρχική οθόνη του συστήματος είναι αυτή όπου ο χρήστης πρέπει να εισάγει το User Id του (UC Number) και το κωδικό πρόσβασης του.Αν είναι η πρώτη φορά του χρήστη στο σύστημα τότε θα πατήσει το κουμπί Register κάτω όπου θα μεταφερθεί στη σελίδα της εγγραφής.

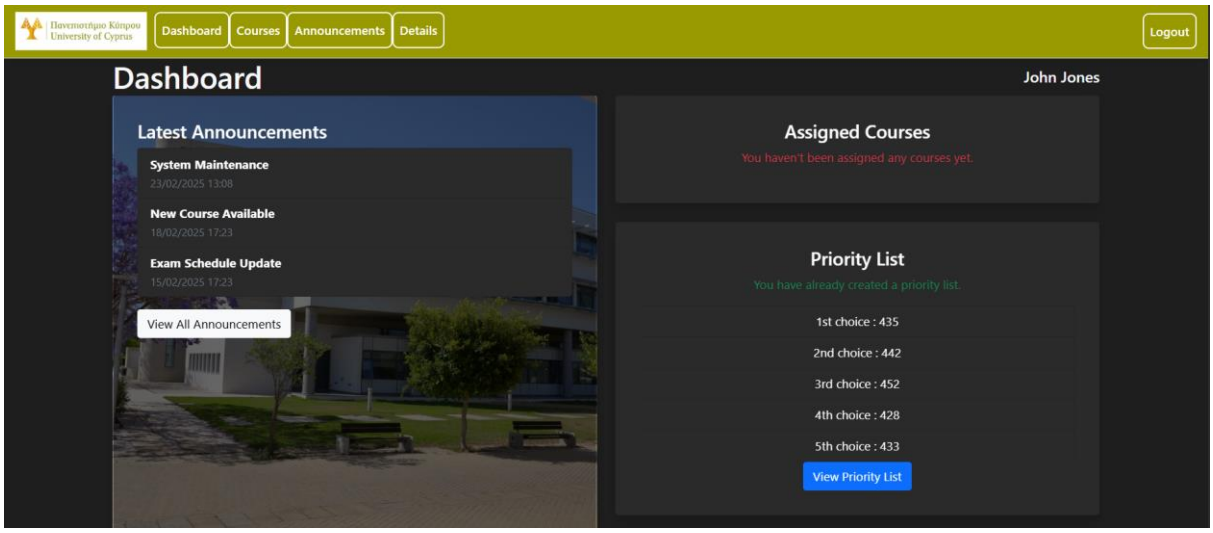


Εικόνα 8 – Διεπαφή Εγγραφής Χρήστη

Στην εικόνα 8 έχουμε την οθόνη του Register όπου ο χρήστης πρέπει να εισάγει το User Id του, Όνομα, Επίθετο , Email και κωδικό πρόσβασης.

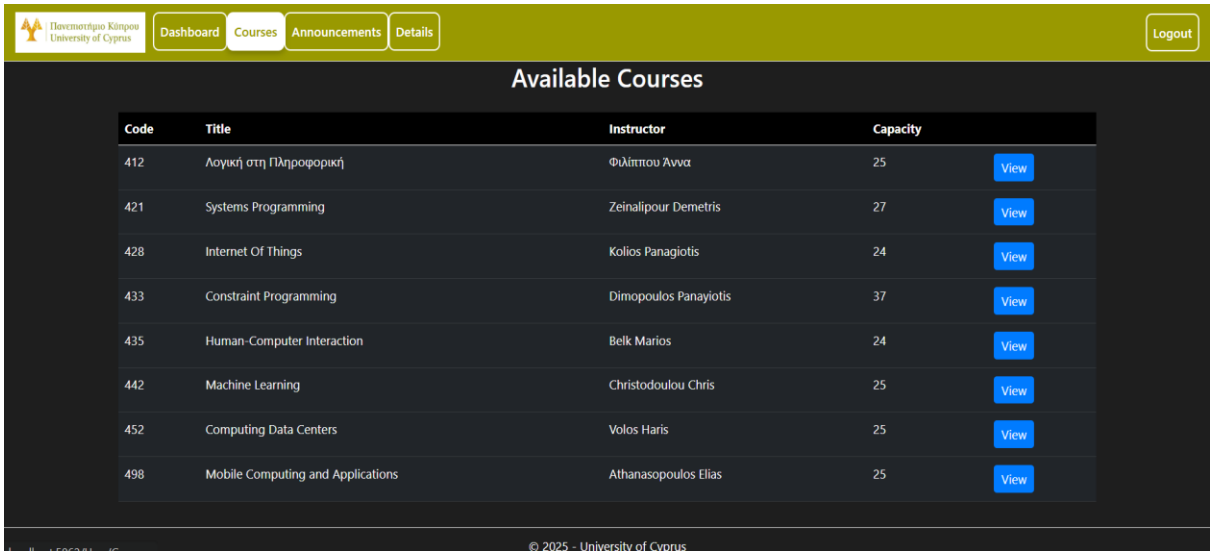
Ροή οθονών Φοιτητών

Με την είσοδο ενός φοιτητή στο σύστημα θα δει μπροστά του τη κύρια οθόνη του συστήματος.



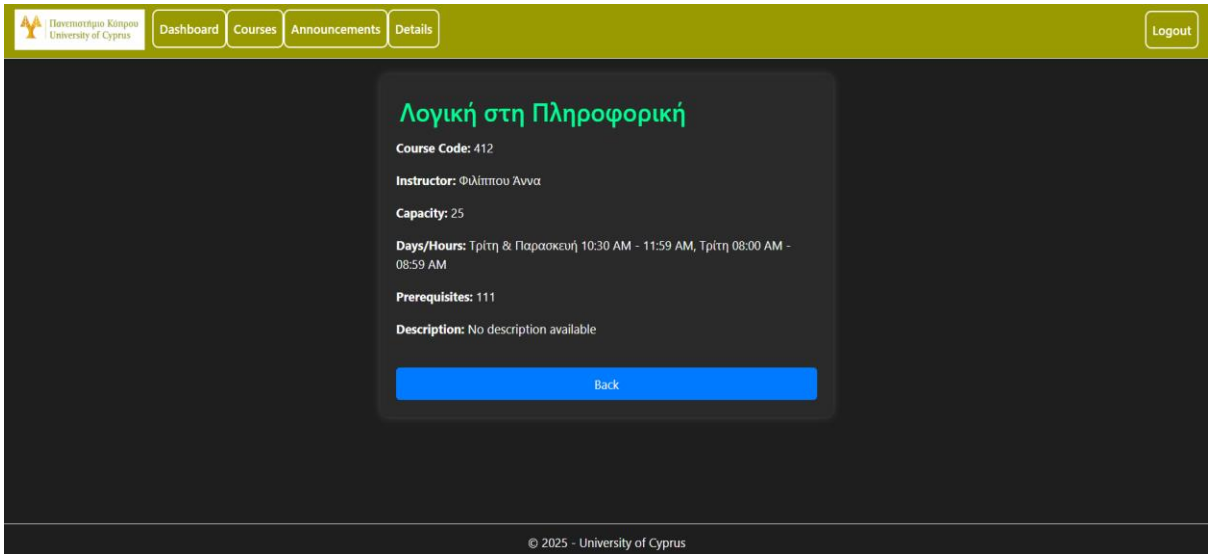
Εικόνα 9 – Κεντρική Διεπαφή Φοιτητή

Στην εικόνα 9 έχουμε το τι θα βλέπει στη κεντρική του οθόνη ο φοιτητής. Ο φοιτητής θα βλέπει μπροστά του τις τελευταίες ανακοινώσεις , τη λίστα προτίμησης του και όταν εκτελεστεί ο αλγόριθμος θα βλέπει και τα μαθήματα που του ανατεθήκαν. Στο
πάνω μέρος υπάρχει το navigation bar όπου μπορεί να μεταφερθεί σε όποια άλλη σελίδα θέλει ή να αποσυνδεθεί.



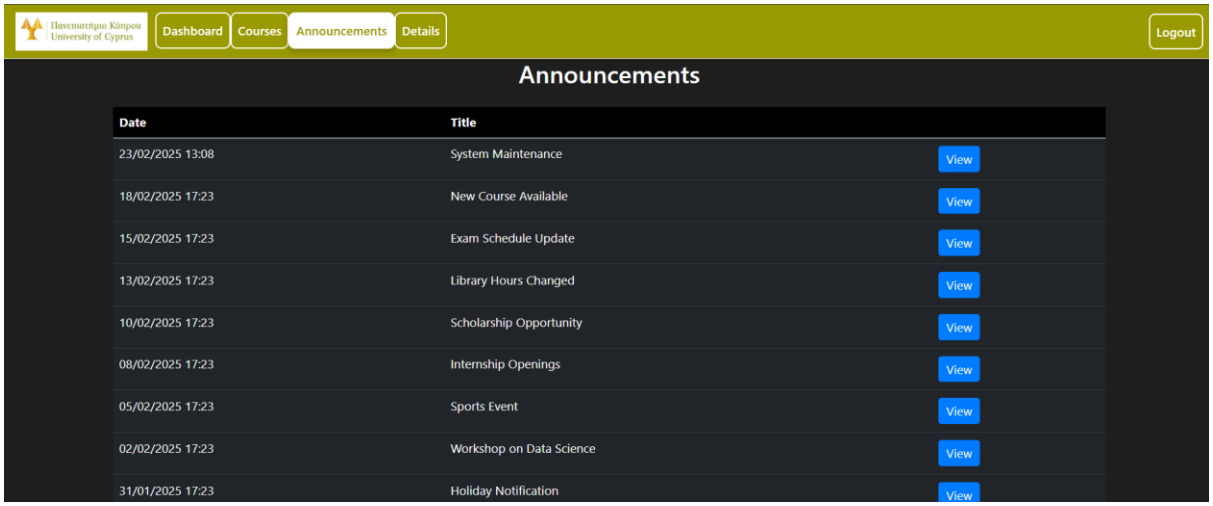
Εικόνα 10 – Διεπαφή Μαθημάτων Εξαμήνου

Στην εικόνα 10 έχουμε τη διεπαφή όπου ο φοιτητής βλέπει όλα τα μαθήματα που προσφέρονται το τρέχον εξάμηνο όπου αν θέλει να δει όλες τις πληροφορίες ενός από τα μαθήματα μπορεί να πατήσει το κουμπί View και να έρθει σε αυτή τη σελίδα:



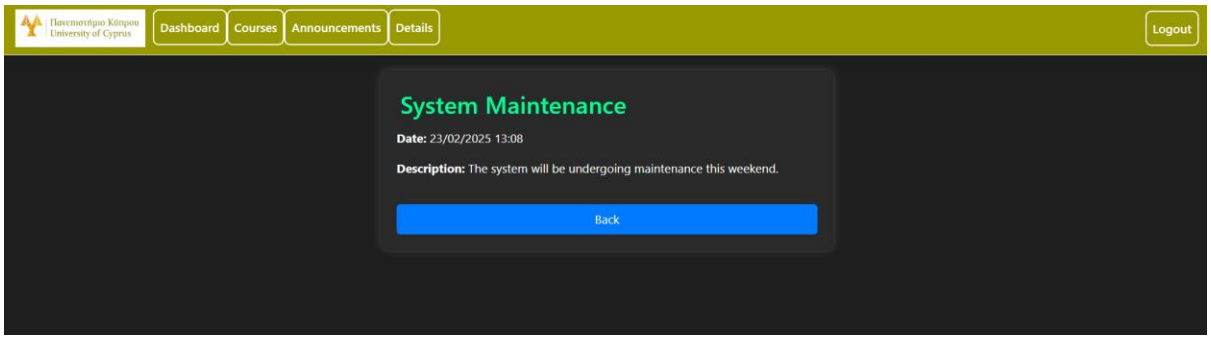
Εικόνα 11 – Διεπαφή Πληροφοριών Συγκεκριμένου Μαθήματος

Στην εικόνα 11 έχουμε τη διεπαφή όπου ο φοιτητής βλέπει όλες τις πληροφορίες που σχετίζονται με το συγκεκριμένο μάθημα.



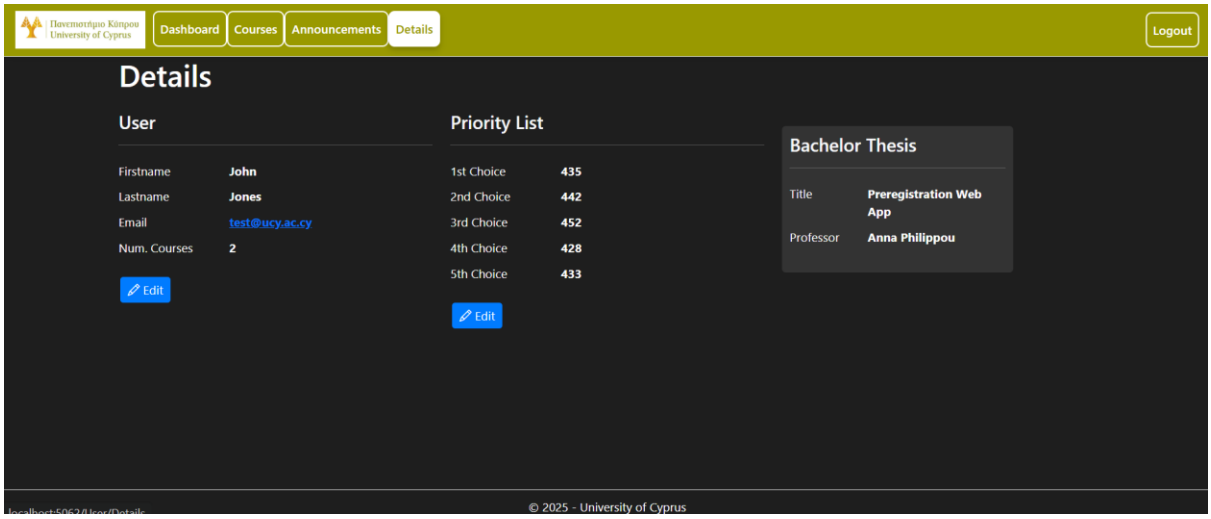
Εικόνα 12 – Διεπαφή Ανακοινώσεων

Αντίστοιχα στην εικόνα 12 έχουμε τη διεπαφή όπου μπορεί να δει όλες τις ανακοινώσεις και αν πατήσει το View να δει το κείμενο της ανακοίνωσης όπως πιο κάτω:



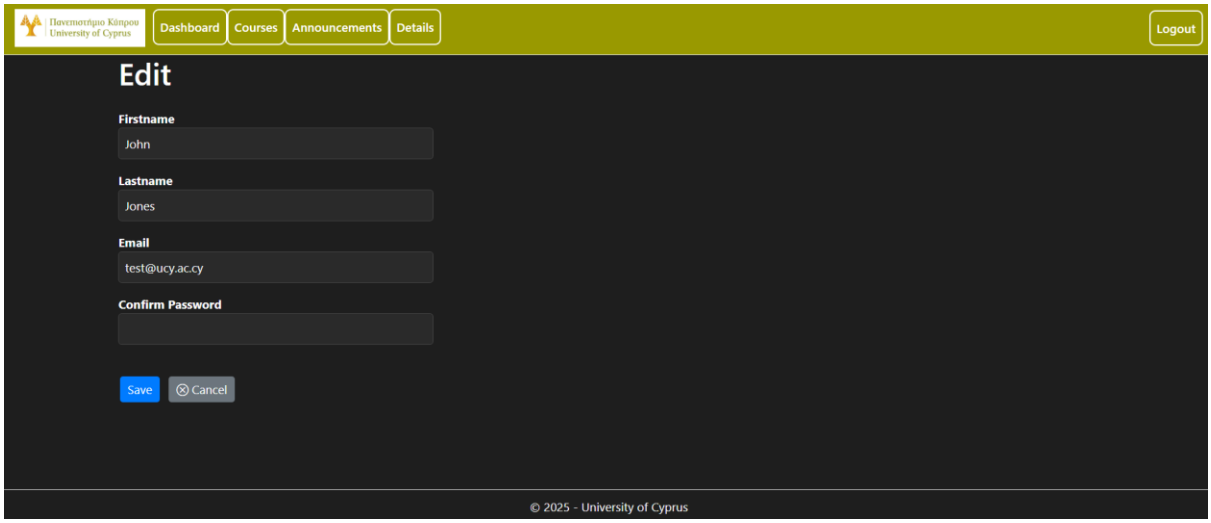
Εικόνα 13 – Διεπαφή Πληροφοριών Συγκεκριμένης Ανακοίνωσης

Στην εικόνα 13 έχουμε τη διεπαφή όπου ο φοιτητής βλέπει όλες τις πληροφορίες που σχετίζονται με τη συγκεκριμένη ανακοίνωση .



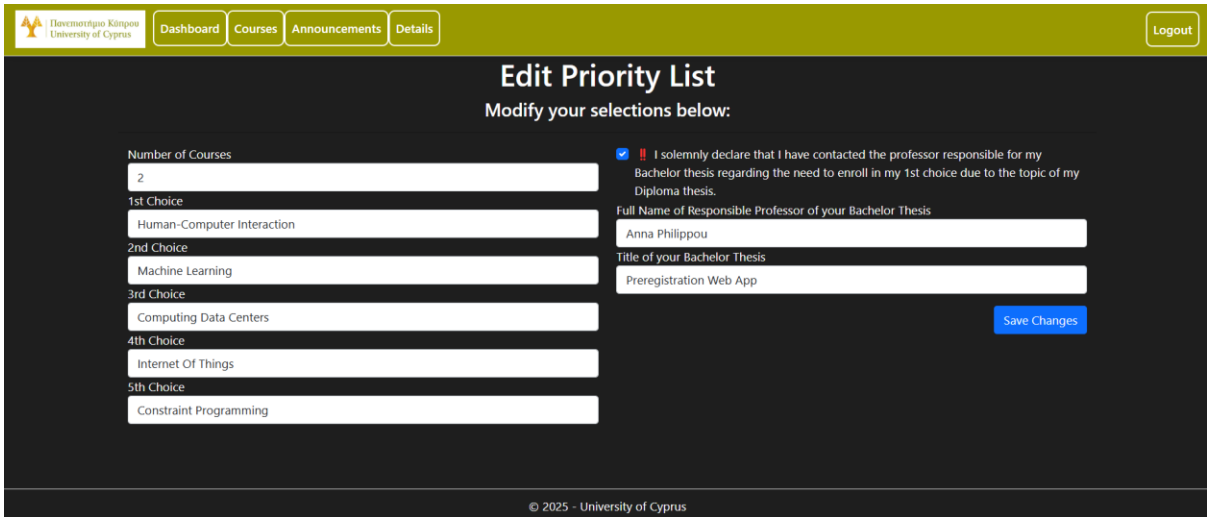
Εικόνα 14 – Διεπαφή Πληροφοριών Φοιτητή

Στην εικόνα 14 έχουμε τη διεπαφή όπου ο χρήστης βλέπει τα στοιχεία του και μπορεί να τα επεξεργαστεί.



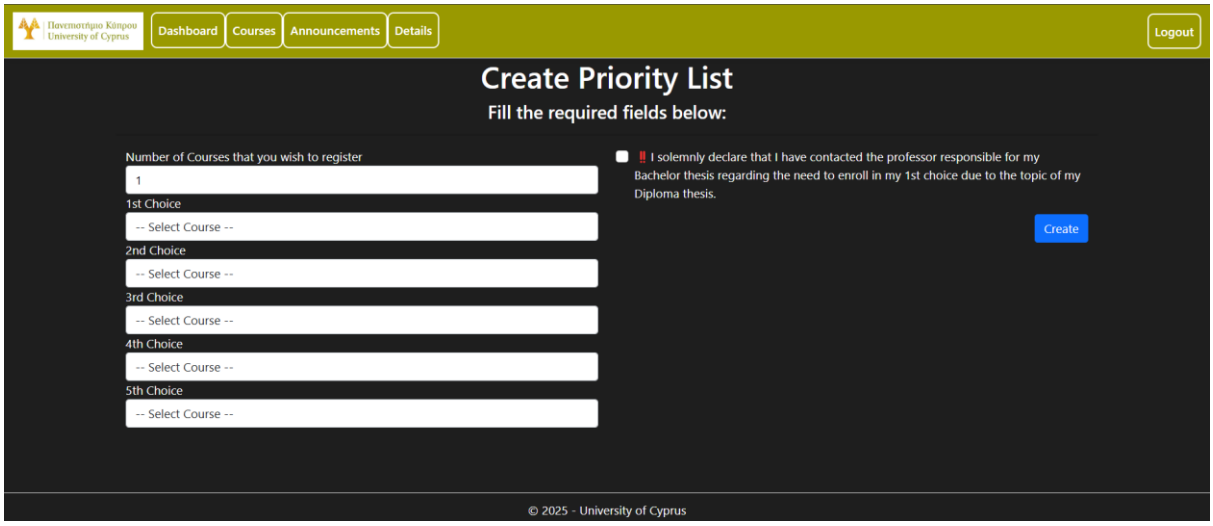
Εικόνα 15 – Διεπαφή Επεξεργασίας Προσωπικών Πληροφοριών

Στην εικόνα 15 έχουμε τη διεπαφή όπου ο φοιτητής μπορεί να επεξεργαστεί τα προσώπικα του στοιχεία σε περίπτωση που χρειάζεται.



Εικόνα 16 – Διεπαφή Επεξεργασίας Λίστας Προτιμήσεων

Στην εικόνα 16 έχουμε τη διεπαφή όπου αν ο φοιτητής έχει δημιουργήσει τη λίστα προτίμησης του, μπορεί να επεξεργαστεί αυτήν όποτε θέλει από εδώ.

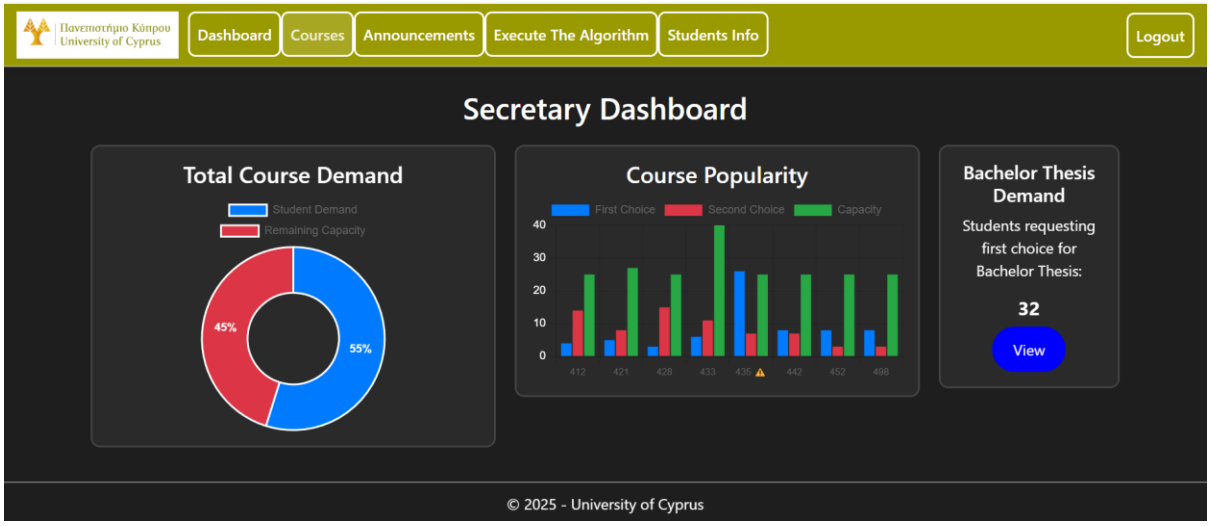


Εικόνα 17 – Διεπαφή Δημιουργίας Λίστας Προτιμήσεων

Στην εικόνα 17 έχουμε τη διεπαφή όπου αν ο φοιτητής δεν δημιούργησε ακόμη τη λίστα προτίμησης του, τότε θα έρχεται στην οθόνη για τη δημιουργία αυτής.

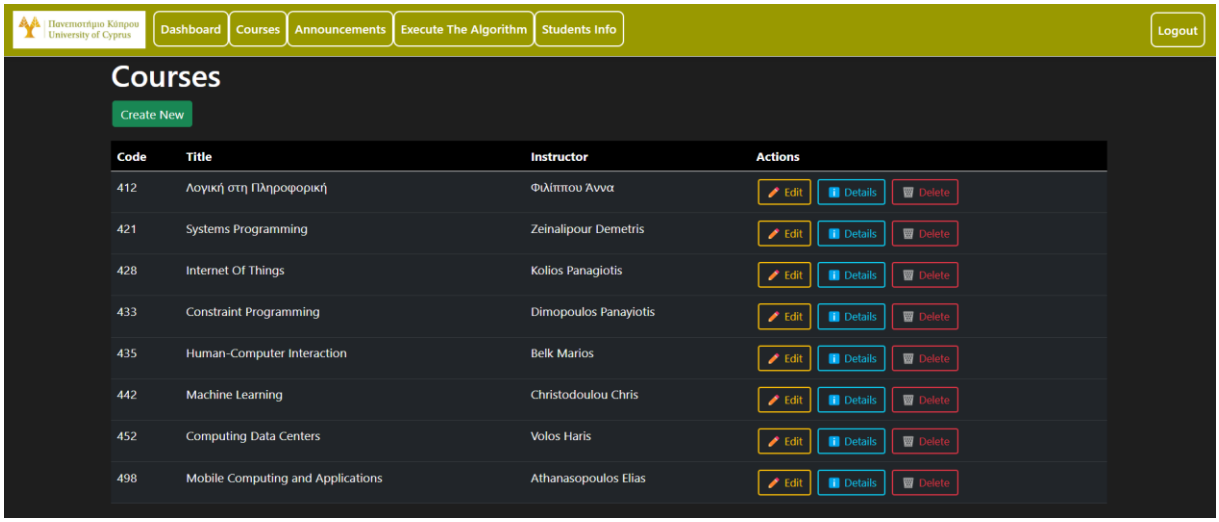
Ροή οθονών Διαχειριστή

Με την είσοδο του διαχειριστή στο σύστημα θα δει μπροστά του τη κύρια οθόνη του συστήματος.



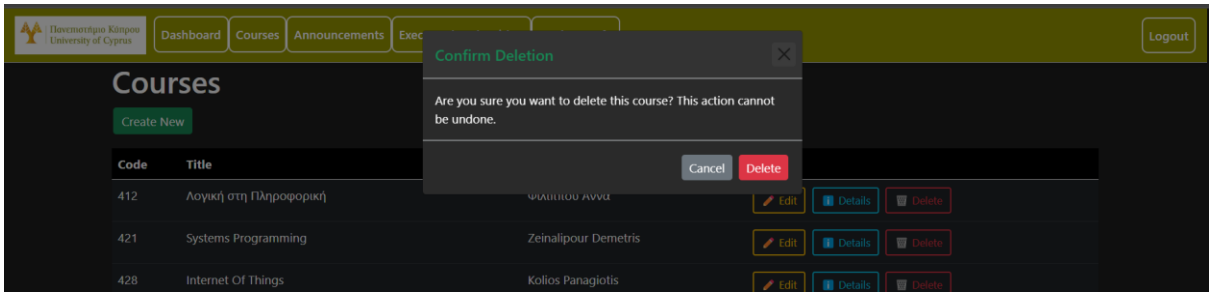
Εικόνα 18 – Κεντρική Διεπαφή Διαχειριστή Συστήματος

Στην εικόνα 18 έχουμε τη κεντρική διεπαφή του διαχειριστή. Ο διαχειριστής στην αρχική του οθόνη θα βλέπει κάποια στατιστικά σχετικά με τις λίστες προτίμησης των φοιτητών. Πιο συγκεκριμένα θα βλέπει το ποσοστό της ζήτησης μαθημάτων από τους φοιτητές έναντι της συνολικής χωρητικότητας των μαθημάτων. Ακόμη θα παρουσιάζεται η ζήτηση ως πρώτη και δεύτερη επιλογή όλων των μαθημάτων και ο αριθμός των φοιτητών που θέλουν τη πρώτη τους επιλογή για την εκπλήρωση της διπλωματικής τους εργασίας. Σε περίπτωση που υπάρχουν προβλήματα που μπορεί να προκαλέσουν πρόβλημα στην εκτέλεση του αλγορίθμου θα εμφανίζονται μηνύματα στην οθόνη έτσι ώστε ο διαχειριστής να γνωρίζει πως να τα διορθώσει.



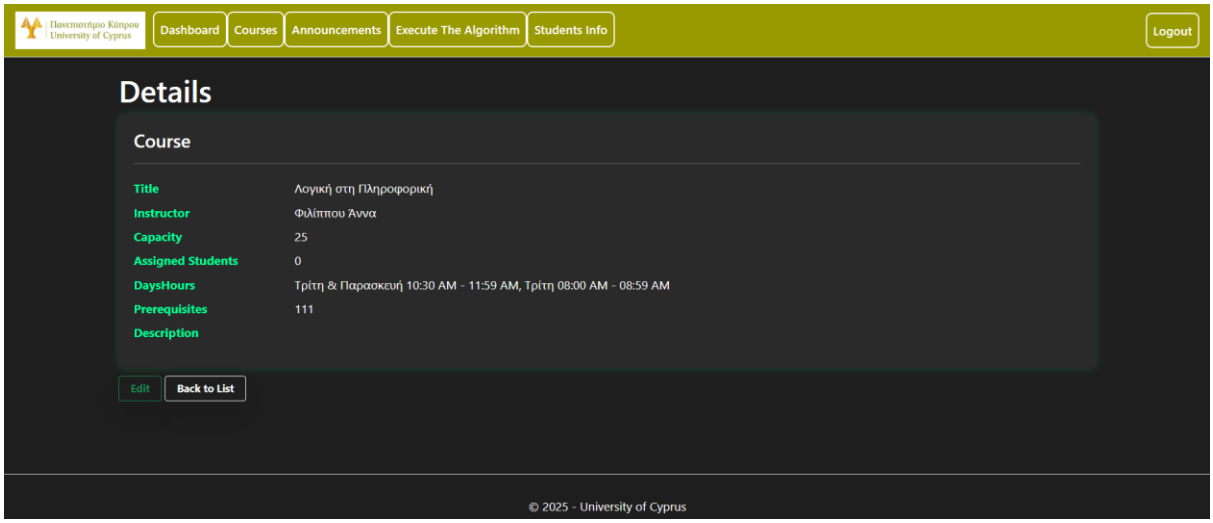
Εικόνα 19 – Διεπαφή Λίστας Ακροατηρίων

Στην εικόνα 19 έχουμε τη διεπαφή όπου ο διαχειριστής θα βλέπει όλα τα μαθήματα του τρέχον εξαμήνου και θα μπορεί να μεταφερθεί στις οθόνες για δημιουργία νέου μαθήματος , προβολής και επεξεργασίας υπάρχον μαθήματος αλλά και να διαγράψει κάποιο μάθημα.



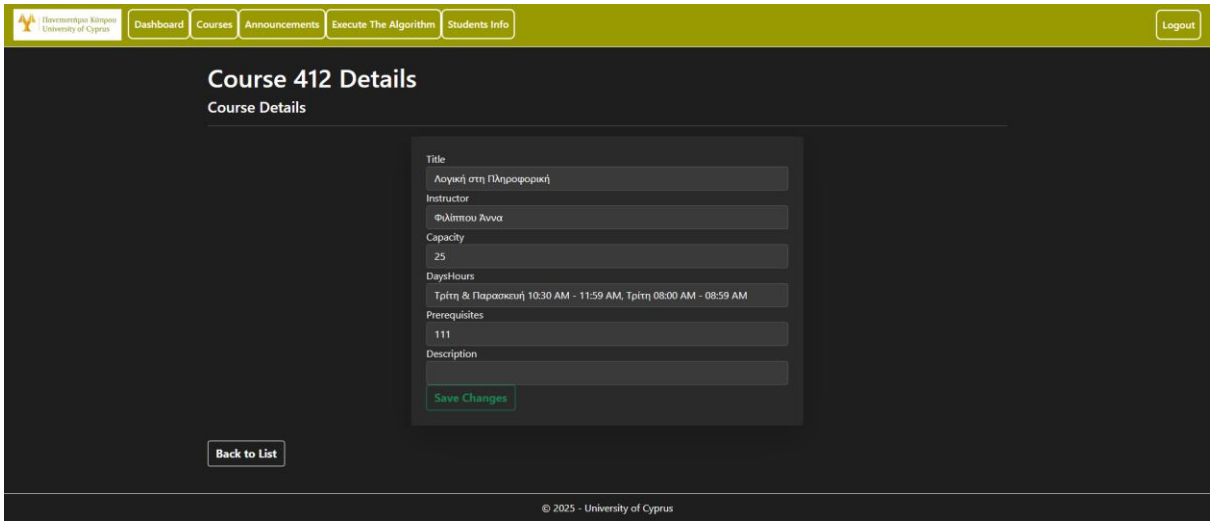
Εικόνα 20 – Αναδυόμενη Προειδοποίηση

Στην εικόνα 20 έχουμε τη προειδοποίηση που θα υπάρχει πριν διαγράψει ένα μάθημα. Παρόμοιες προειδοποιήσεις υπάρχουν σε όλες τις "επικίνδυνες" ενέργειες, όπως η διαγραφή δεδομένων, η αλλαγή κρίσιμων παραμέτρων ή η επιβεβαίωση σημαντικών αποφάσεων, προκειμένου να αποτραπούν ακούσια λάθη και να διασφαλιστεί η ορθή λειτουργία του συστήματος.



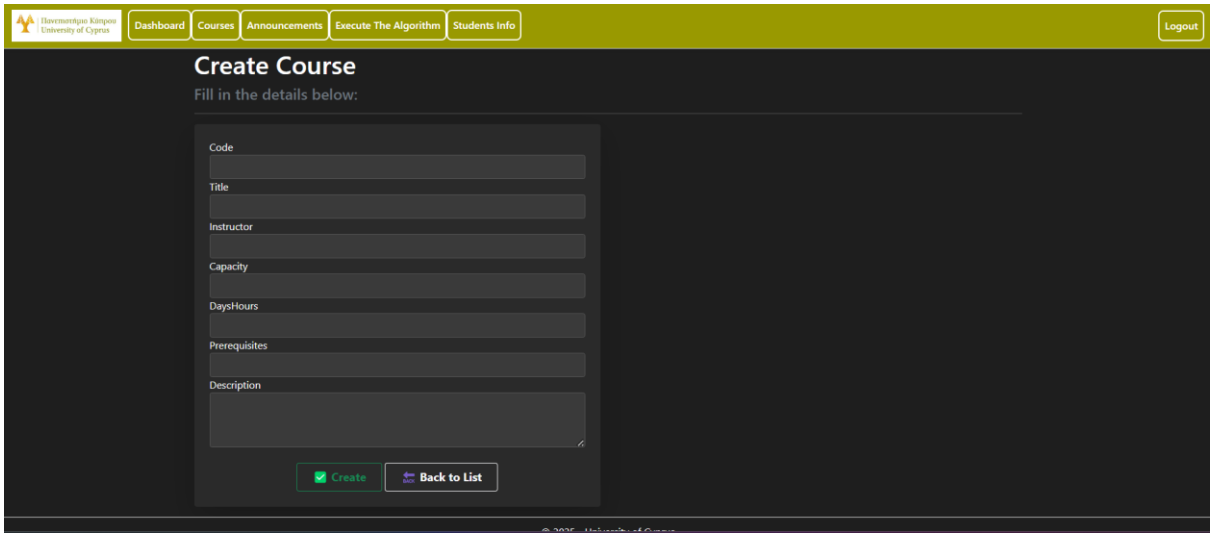
Εικόνα 21 – Διεπαφή Πληροφοριών Συγκεκριμένου Ακροατηρίου

Στην εικόνα 21 έχουμε τη διεπαφή όπου ο διαχειριστής βλέπει όλες τις πληροφοριές σχετικά με το συγκεκριμένο μάθημα.



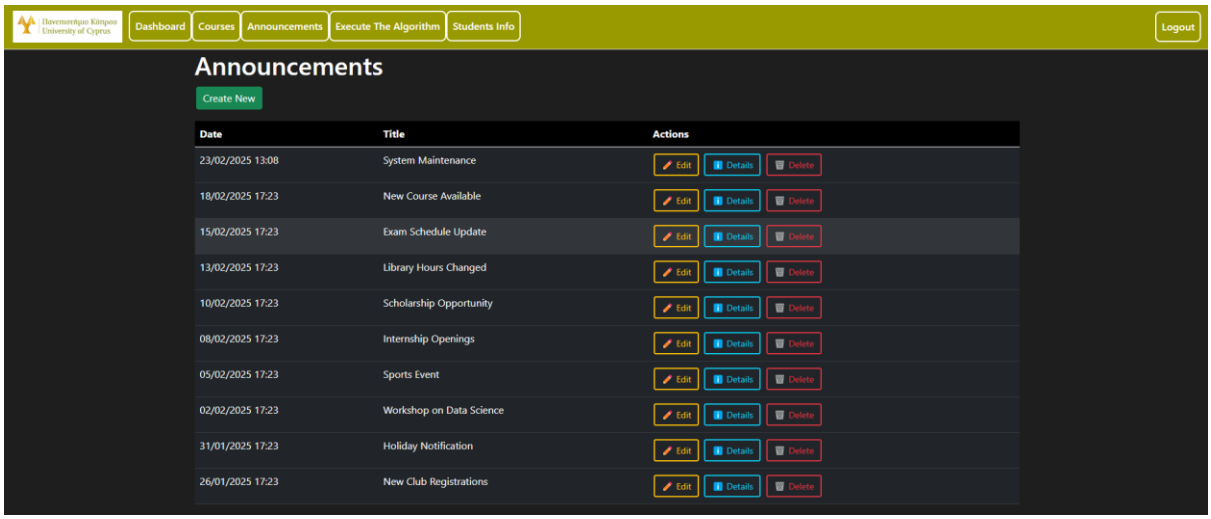
Εικόνα 22 – Διεπαφή Επεξεργασίας Συγκεκριμένου Ακροατηρίου

Στην εικόνα 22 έχουμε τη διεπαφή όπου ο διαχειριστής μπορεί να αλλάξει τις πληροφοριές σχετικά με το συγκεκριμένο μάθημα.



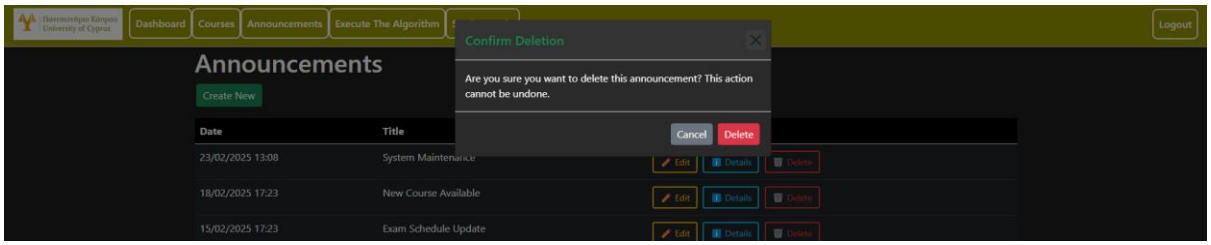
Εικόνα 23 – Διεπαφή Δημιουργίας Ακροατηρίου

Στην εικόνα 23 έχουμε τη διεπαφή όπου ο διαχειριστής μπορεί να δημιουργήσει ένα νέο μάθημα.

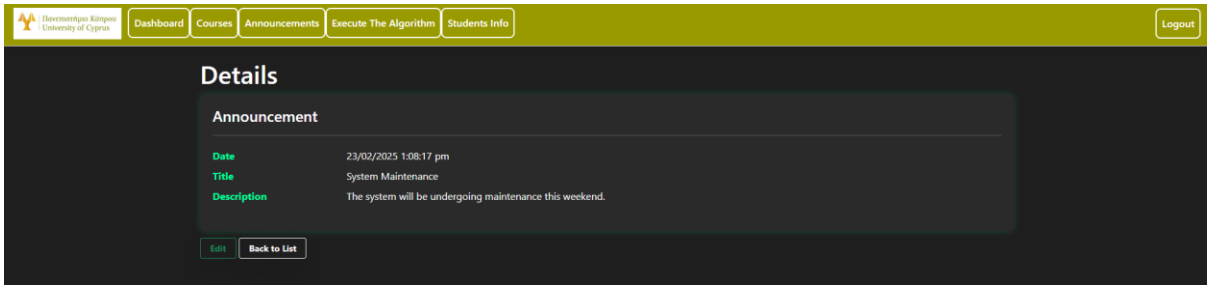


Εικόνα 24 – Διεπαφή Λίστας Ανακοινώσεων

Αντίστοιχα με τα μαθήματα θα γίνεται το ίδιο και για τις ανακοινώσεις. Στην εικόνα 24 έχουμε τη διεπαφή όπου ο διαχειριστής θα βλέπει όλες τις ανακοινώσεις και θα μπορεί να μεταφερθεί στις οθόνες για δημιουργία νέας ανακοίνωσης , προβολής και επεξεργασίας υπάρχον ανακοίνωσης αλλά και να διαγράψει κάποια ανακοίνωση.

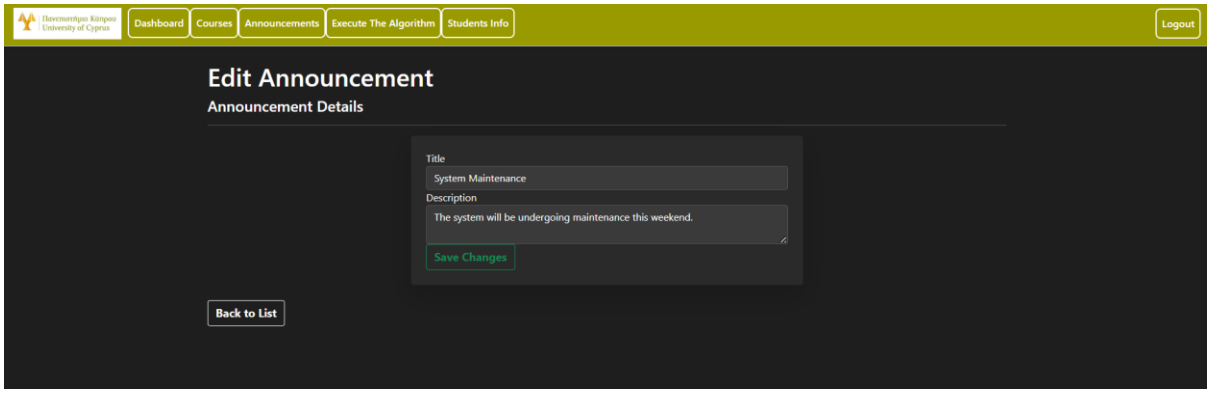


Εικόνα 25 – Αναδυόμενη Προειδοποίηση



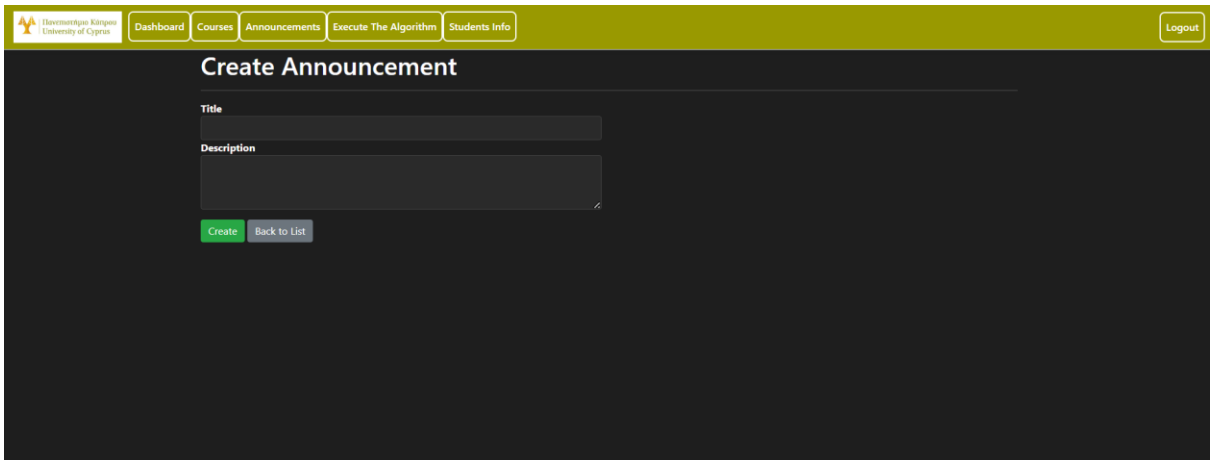
Εικόνα 26 – Διεπαφή Πληροφοριών Συγκεκριμένης Ανακοίνωσης

Στην εικόνα 26 έχουμε τη διεπαφή όπου ο διαχειριστής βλέπει όλες τις πληροφοριές σχετικά με τη συγκεκριμένη ανακοίνωση.



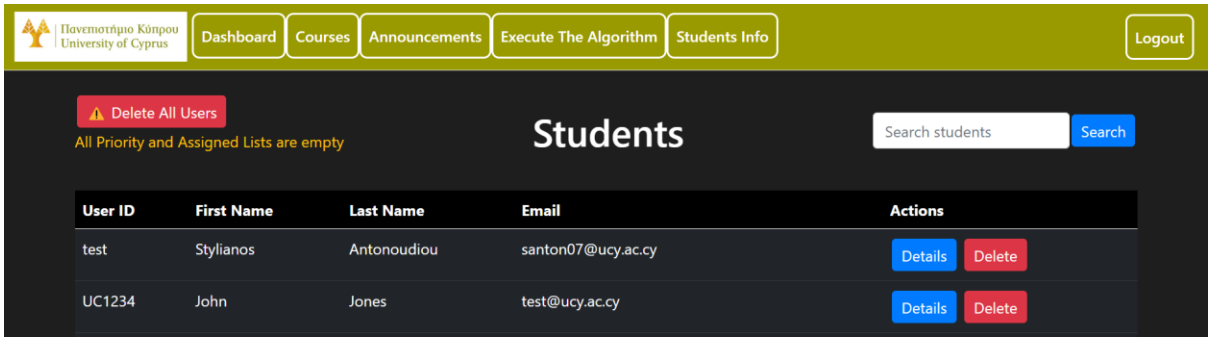
Εικόνα 27 - Διεπαφή Επεξεργασίας Συγκεκριμένου Ακροατηρίου

Στην εικόνα 27 έχουμε τη διεπαφή όπου ο διαχειριστής μπορεί να αλλάξει τις πληροφοριές σχετικά με τη συγκεκριμένη ανακοίνωση.



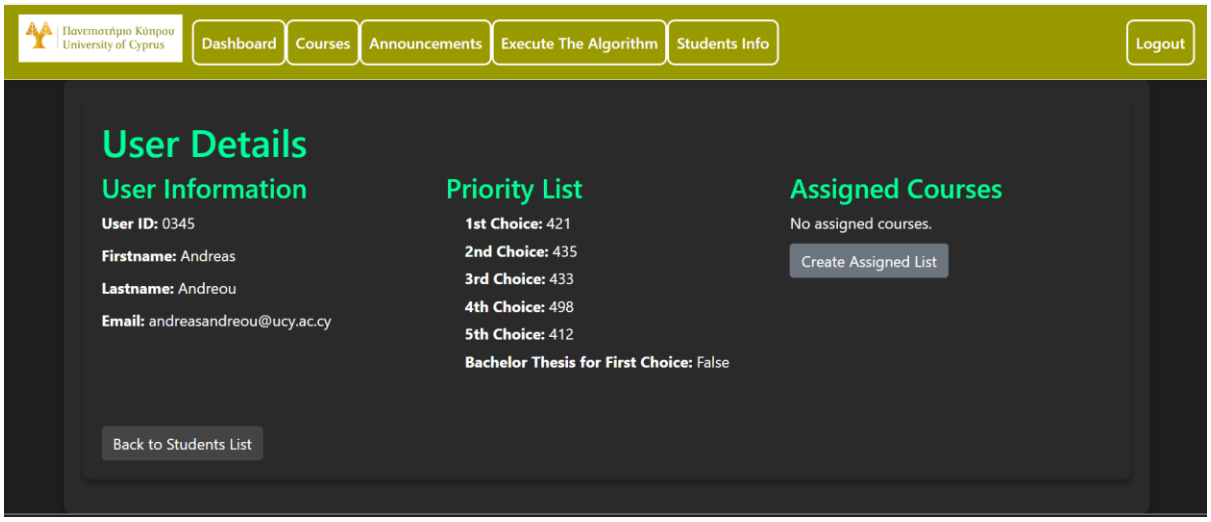
Εικόνα 28 - Διεπαφή Δημιουργίας Ανακοίνωσης

Στην εικόνα 28 έχουμε τη διεπαφή όπου ο διαχειριστής μπορεί να δημιουργήσει μία νέα ανακοίνωση.



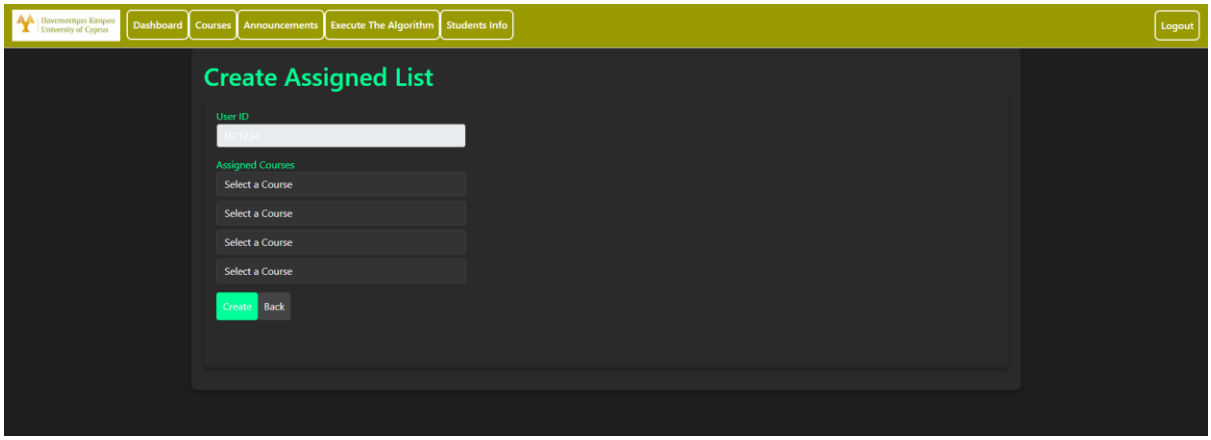
Εικόνα 29 - Διεπαφή Λίστας Φοιτητών

Στην εικόνα 29 έχουμε τη διεπαφή Students Info όπου ο διαχειριστής βλέπει τη λίστα με όλους τους φοιτητές , μπορεί να αναζητήσει κάποιο συγκεκριμένο και να δει όλες τις λεπτομέρειες που σχετίζονται με κάποιο φοιτητή εάν πατήσει το κουμπί Details.



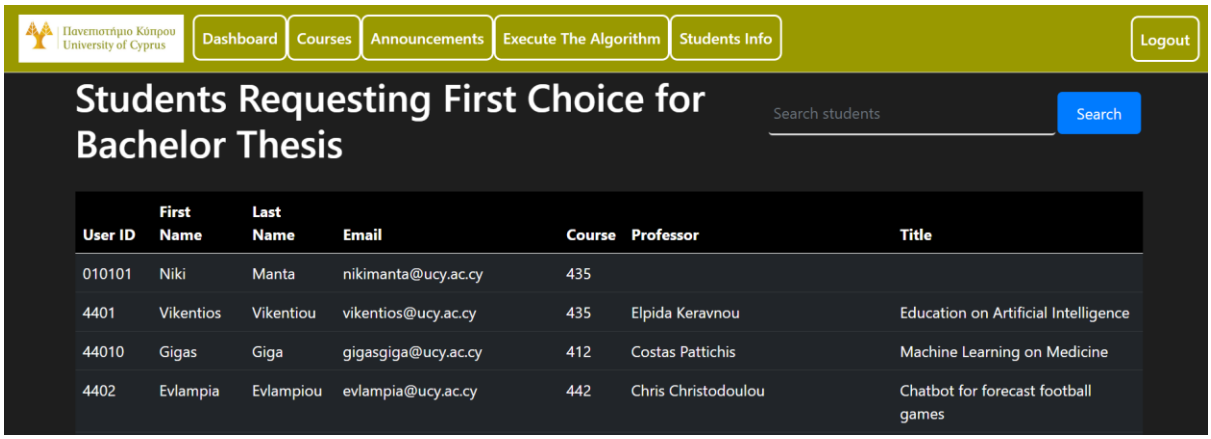
Εικόνα 30 - Διεπαφή Πληροφοριών Συγκεκριμένου Φοιτητή

Ο διαχειριστής έχει τη δυνατότητα να αναθέσει χειροκίνητα τα μαθήματα ενός φοιτητή και αυτός ο φοιτητής να μην συμπεριληφθεί στον αλγόριθμο.



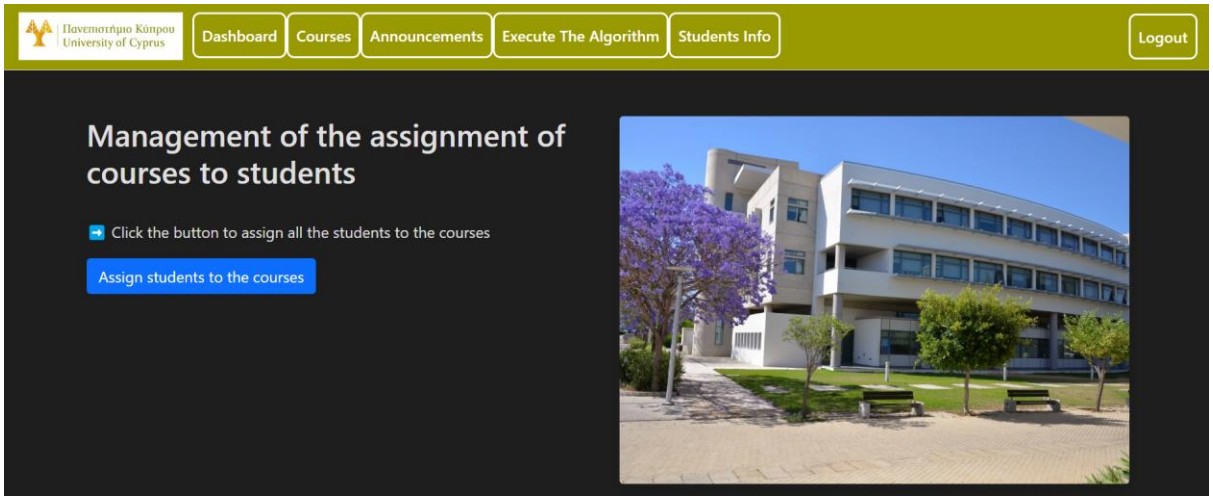
Εικόνα 31 - Διεπαφή Ανάθεσης Μαθημάτων Σε Συγκεκριμένο Φοιτητή

Στην εικόνα 31 έχουμε τη διεπαφή όπου ο διαχειριστής μπορεί να αναθέσει μαθήματα σε κάποιο φοιτητή χειροκίνητα και αυτός να μην συμπεριληφθεί στον αλγόριθμο.



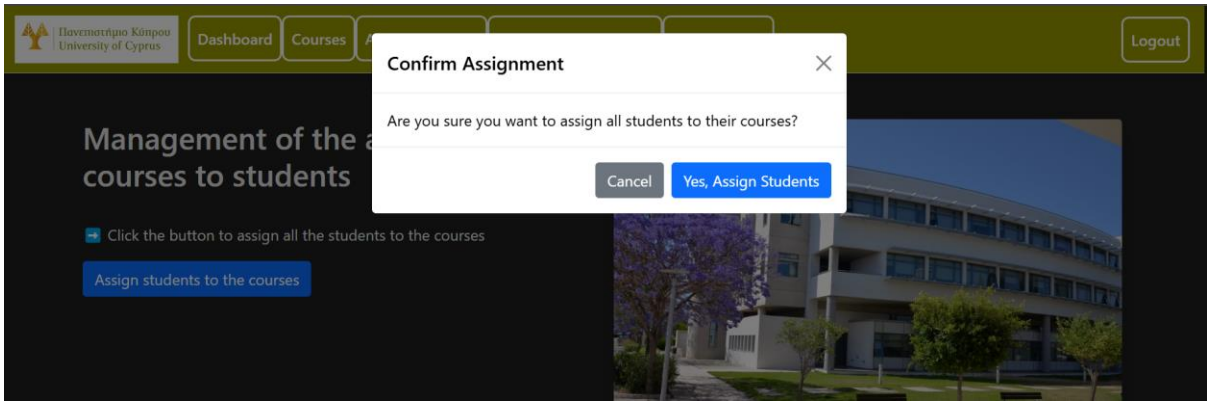
Εικόνα 32 - Διεπαφή Λίστας Φοιτητών για Διπλωματική

Ακόμη από τη κύρια οθόνη μπορεί να έρθει εδώ και να δει τη λίστα με τους φοιτητές που χρειάζονται την πρώτη τους επιλογή για την εκπλήρωση της διπλωματικής τους εργασίας όπως φαίνεται και στην εικόνα 32.

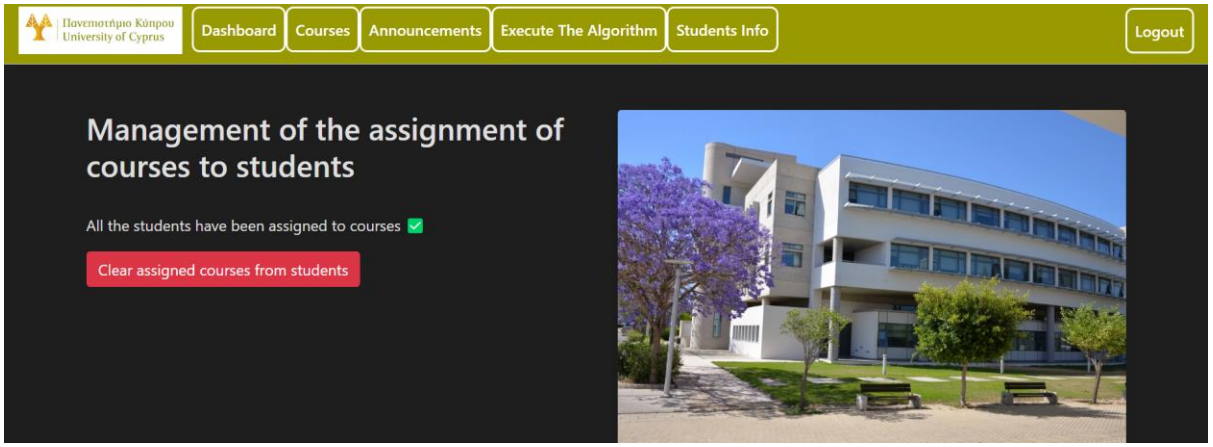


Εικόνα 33 - Διεπαφή Εκτέλεσης Αλγορίθμου

Στην εικόνα 33 έχουμε τη διεπαφή όπου ο διαχειριστής μπορεί να εκκινήσει τον αλγόριθμο.

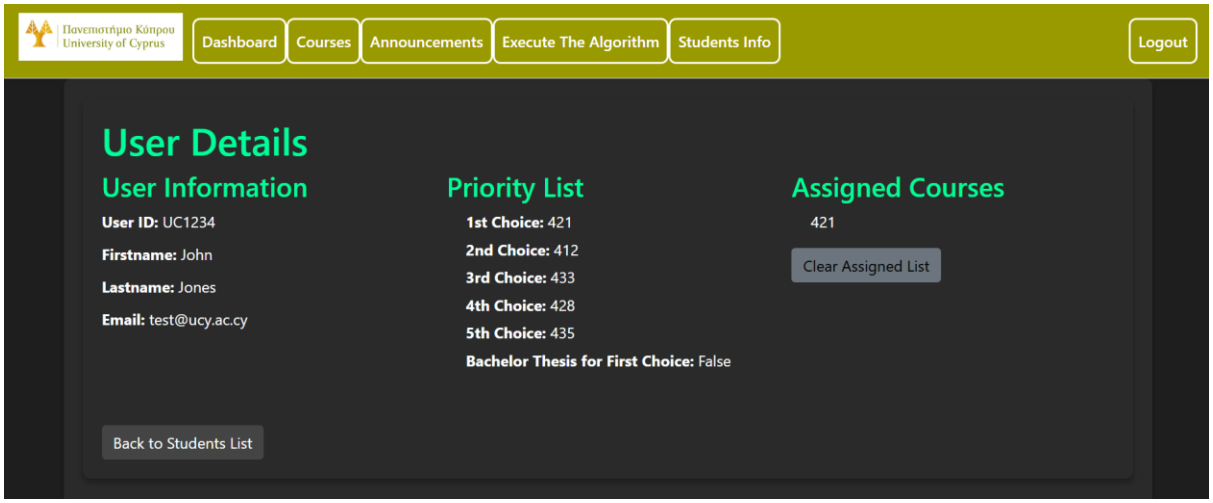


Εικόνα 34 – Αναδυόμενο παράθυρο



Εικόνα 35 - Διεπαφή Εκτέλεσης Αλγορίθμου

Μετά την εκτέλεση του αλγορίθμου ο διαχειριστής μπορεί να καθαρίσει τα μαθήματα που ανατέθηκαν στους φοιτητές αν υπάρχει η ανάγκη για κάτι τέτοιο όπως φαίνεται και στην εικόνα 35.



Εικόνα 36 - Διεπαφή Πληροφοριών Συγκεκριμένου Φοιτητή

Αν θέλει όμως να κάνει κάποια αλλαγή στα μαθήματα ενός συγκεκριμένου φοιτητή μετά την εκτέλεση του αλγορίθμου μπορεί να έρθει εδώ και να καθαρίσει τη λίστα του συγκεκριμένου φοιτητή και να του θέσει χειροκίνητα τα μαθήματα.

Κεφάλαιο 6

Αξιολόγηση και Δοκιμές

6.1 Στρατηγική δοκιμών του συστήματος

Η στρατηγική δοκιμών του συστήματος σχεδιάστηκε με σκοπό να διασφαλίσει την ορθότητα, τη σταθερότητα και την ευχρηστία της εφαρμογής σε όλα τα επίπεδα. Οι δοκιμές κάλυψαν τόσο τη λειτουργικότητα των επιμέρους τμημάτων του συστήματος όσο και τη συνολική εμπειρία του χρήστη κατά την αλληλεπίδρασή του με την εφαρμογή. Οι κύριες κατηγορίες δοκιμών που εφαρμόστηκαν ήταν οι εξής:

Μονάδες Ελέγχου (Unit Testing)
Ελέγχθηκαν ξεχωριστά τα επιμέρους μέρη της εφαρμογής, όπως οι μέθοδοι των Models, οι συναρτήσεις στο backend (Controllers και Services) και η σωστή απόδοση τιμών. Για τις δοκιμές αυτές χρησιμοποιήθηκε το εργαλείο xUnit, εξασφαλίζοντας ότι κάθε λειτουργία παράγει τα επιθυμητά αποτελέσματα σε ποικίλες συνθήκες.

Έλεγχος Επικύρωσης (Validation Testing)
Δοκιμάστηκε η σωστή λειτουργία των validation μηχανισμών σε όλες τις φόρμες εισαγωγής δεδομένων. Για παράδειγμα διασφαλίστηκε ότι οι φοιτητές δεν μπορούν να υποβάλουν άδειες ή ελλιπείς λίστες προτεραιότητας και ότι τα δεδομένα των μαθημάτων έχουν λογικές τιμές (όρια χωρητικότητας, μη διπλότυπες εγγραφές κ.λπ.).

Λειτουργικές Δοκιμές (Functional Testing)
Έγιναν δοκιμές πλήρους ροής του συστήματος από την πλευρά τόσο του φοιτητή όσο και του διαχειριστή. Ελέγχθηκε αν κάθε χρήστης μπορεί να ολοκληρώσει τα κρίσιμα σενάρια χρήσης

(όπως η καταχώριση προτιμήσεων, η κατανομή μέσω του αλγορίθμου, η προβολή ανακοινώσεων) χωρίς σφάλματα ή ασυνέπειες.

Δοκιμές Ολοκλήρωσης (Integration Testing)

Διασφαλίστηκε ότι όλα τα μέρη της εφαρμογής συνεργάζονται σωστά μεταξύ τους. Π.χ. η σύνδεση μεταξύ των Views και των Controllers, η αποθήκευση/ανάκτηση από τη βάση δεδομένων, και η επικοινωνία με το service που εκτελεί τον αλγόριθμο κατανομής μέσω των Google OR-Tools.

Έλεγχος Ανθεκτικότητας (Error & Exception Handling)

Πραγματοποιήθηκαν σκόπιμες δοκιμές με λανθασμένα ή ακραία δεδομένα για να αξιολογηθεί η συμπεριφορά του συστήματος σε μη αναμενόμενες συνθήκες. Το σύστημα ανταποκρίθηκε με σωστά διαχειριζόμενα σφάλματα και φιλικά μηνύματα προς τον χρήστη, χωρίς να προκύπτουν crash ή απώλεια δεδομένων.

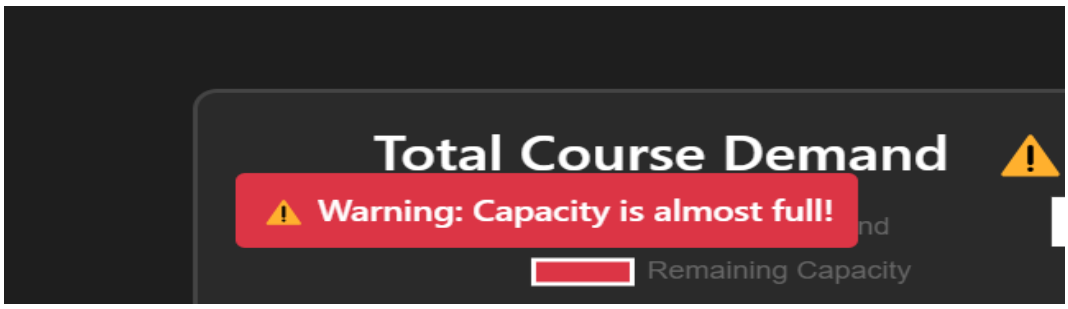
Χρήση δοκιμαστικών σεναρίων

Δημιουργήθηκαν σενάρια με εικονικούς φοιτητές, μαθήματα και δεδομένα εισόδου, ώστε να εκτελεστούν ρεαλιστικές δοκιμές κατανομής. Με αυτό τον τρόπο επαληθεύτηκε η ακρίβεια των αποτελεσμάτων και η συνοχή των δεδομένων που προκύπτουν μετά την εκτέλεση του αλγορίθμου.

Η στρατηγική δοκιμών κάλυψε έτσι ένα ευρύ φάσμα λειτουργιών, συμβάλλοντας στην παροχή ενός αξιόπιστου και εύχρηστου συστήματος τόσο για τους φοιτητές όσο και για το διαχειριστή.

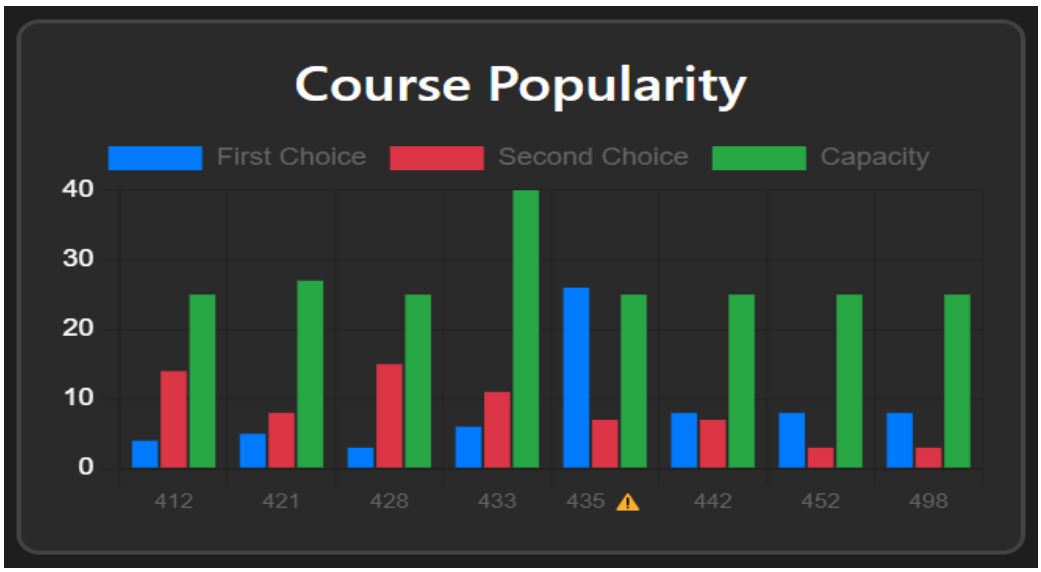
6.2 Διαχείριση σφαλμάτων και εξαιρέσεων σχετικά με τον αλγόριθμο

Σε αυτό το υποκεφάλαιο παρουσιάζονται ορισμένα στιγμιότυπα οθόνης (screenshots) που αφορούν προειδοποιητικά μηνύματα και ελέγχους που έχουν ενσωματωθεί στο σύστημα, με σκοπό την έγκαιρη ανίχνευση προβλημάτων τα οποία ενδέχεται να επηρεάσουν την ορθή εκτέλεση του αλγορίθμου κατανομής. Οι ειδοποιήσεις εμφανίζονται στη διεπαφή του διαχειριστή και προειδοποιούν για κρίσιμες καταστάσεις.



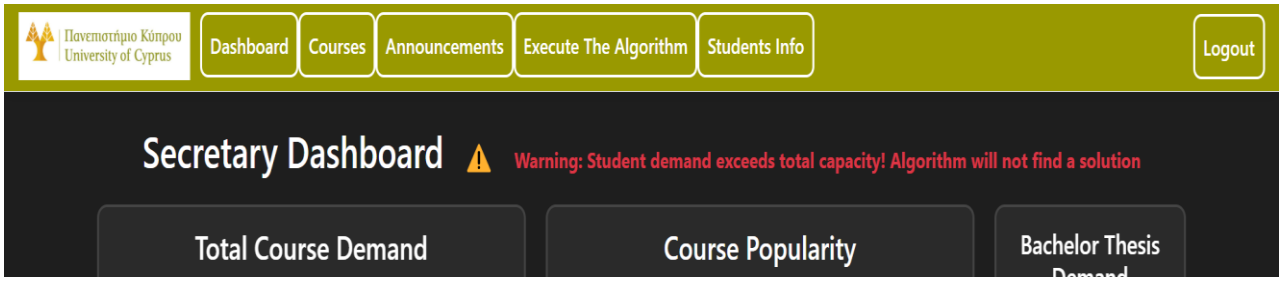
Εικόνα 37 – Προειδοποίηση Χωρητικότητας

Όταν η συνολική ζήτηση μαθημάτων ξεπεράσει το 80 % της συνολικής χωρητικότητας των μαθημάτων τότε φαίνεται αυτό το προειδοποιητικό μήνυμα στο διαχειριστή όπως βλέπουμε και στην εικόνα 37.



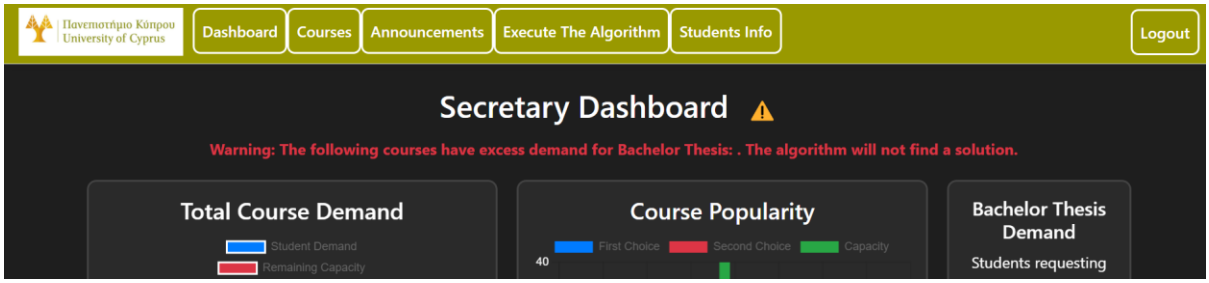
Εικόνα 38 – Προειδοποίηση Πρώτης Επιλογής

Όταν η ζήτηση ως πρώτη επιλογή ενός μαθήματος ξεπεράσει τη χωρητικότητα αυτού τότε φαίνεται η προειδοποίηση αυτή δίπλα από το κωδικό του συγκεκριμένου μαθήματος. Όταν θα εμφανίζεται αυτή η προειδοποίηση γνωρίζουμε ότι σίγουρα θα υπάρχει φοιτητής που δεν θα πάρει τη πρώτη του επιλογή. Βλέπουμε αυτή τη περίπτωση στην εικόνα 38.



Εικόνα 39 – Προειδοποίηση Μη Επιλυσιμότητας

Βλέπουμε στην εικόνα 39 ότι όταν η συνολική ζήτηση των φοιτητών ξεπεράσει το συνολικό μέγεθος των ακροατηρίων τότε έχουμε πρόβλημα όπου ο αλγόριθμος δεν θα μπορέσει να βρει λύση και κάτι πρέπει να αλλάξει στα μεγέθη των ακροατηρίων έτσι ώστε να διορθωθεί άμεσα.



Εικόνα 40 – Προειδοποίηση Μη Επιλυσιμότητας

Ο άλλος τρόπος που θα προκαλέσει πρόβλημα μη εύρεσης λύσης, είναι όταν ζητούν τη πρώτη τους επιλογή για την εκπλήρωση της διπλωματικής τους εργασίας για ένα συγκεκριμένο μάθημα, περισσότεροι φοιτητές σε σχέση με τη χωρητικότητα αυτού του μαθήματος.

6.3 Αξιολόγηση της χρηστικότητας του συστήματος μέσω των χρηστών

Για την αξιολόγηση της χρηστικότητας του συστήματος, ακολουθήθηκε μια ποιοτική προσέγγιση με βάση παρατηρήσεις από χρήστες-δοκιμαστές (φοιτητές και διαχειριστής), οι οποίοι χρησιμοποίησαν την εφαρμογή σε πραγματικά σενάρια χρήσης. Σκοπός ήταν να εντοπιστούν τυχόν δυσκολίες πλοήγησης, ασάφειες στη διεπαφή, αλλά και να αξιολογηθεί η συνολική εμπειρία από τη χρήση της πλατφόρμας.

Η διαδικασία βασίστηκε σε θεμελιώδεις **αρχές ευχρηστίας**, όπως αυτές διατυπώθηκαν από τον **Jakob Nielsen**:

Ορατότητα της κατάστασης του συστήματος: Το σύστημα ενημερώνει συνεχώς τον χρήστη για την κατάστασή του, π.χ. με ειδοποιήσεις μετά την υποβολή λίστας προτεραιότητας ή την ολοκλήρωση της κατανομής.

Αντιστοίχιση με τον πραγματικό κόσμο: Οι ορολογίες και οι επιλογές είναι σχεδιασμένες με γνώμονα τη γλώσσα που χρησιμοποιούν οι φοιτητές και ο διαχειριστής, αποφεύγοντας τεχνικούς όρους.

Έλεγχος και ελευθερία του χρήστη: Οι χρήστες έχουν τη δυνατότητα να διορθώσουν ή να αναιρέσουν ενέργειες όπου αυτό είναι κρίσιμο (π.χ. επαναυποβολή προτιμήσεων πριν την κατανομή).

Συνεκτικότητα και πρότυπα: Το περιβάλλον είναι συνεκτικό και ακολουθεί ομοιόμορφα πρότυπα πλοήγησης, διευκολύνοντας την εξοικείωση του χρήστη.

Πρόληψη λαθών: Το σύστημα δεν επιτρέπει την υποβολή ελλιπών ή λανθασμένων δεδομένων (π.χ. λίστα με λιγότερα από 5 μαθήματα).

Αντιμετώπιση λαθών με απλά μηνύματα: Όταν προκύπτει κάποιο σφάλμα, εμφανίζεται ένα φιλικό μήνυμα που εξηγεί τι πήγε στραβά και πώς να διορθωθεί.

Βοήθεια και τεκμηρίωση: Παρέχονται οδηγίες χρήσης και ενημερωτικά μηνύματα ώστε ο χρήστης να κατανοεί τα βήματα της διαδικασίας.

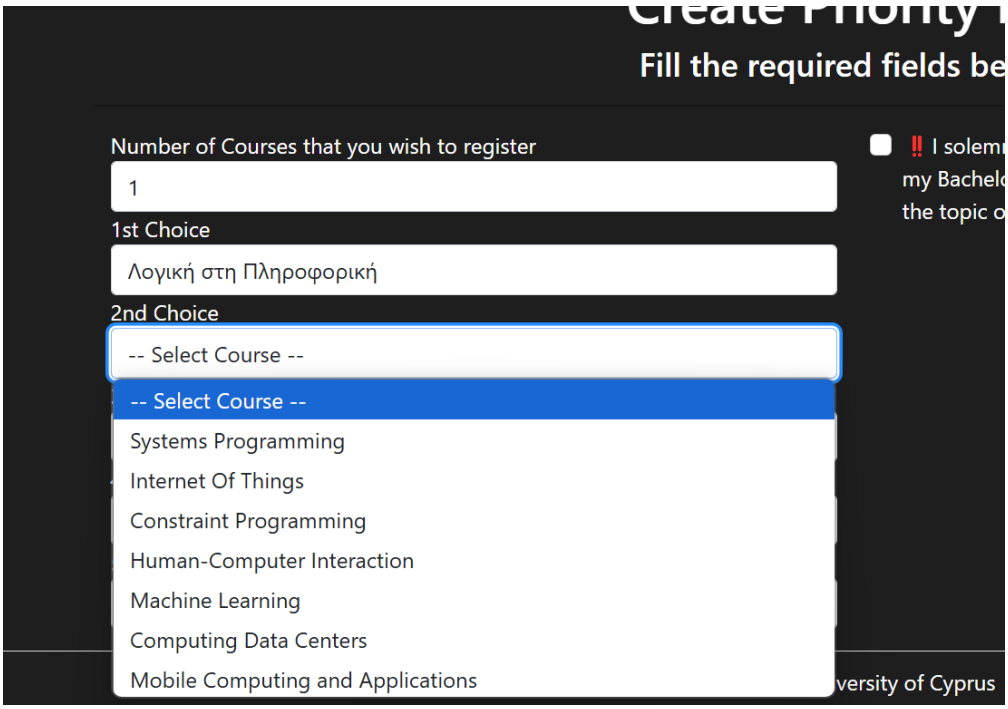
Κατά τη διάρκεια της ανάπτυξης του συστήματος, πραγματοποιήθηκαν δοκιμές χρηστικότητας με τη συμμετοχή 6 συμφοιτητών μας, προκειμένου να αξιολογηθεί η εμπειρία χρήσης και να εντοπιστούν πιθανά σημεία βελτίωσης. Οι φοιτητές που συμμετείχαν είχαν την ευκαιρία να πλοηγηθούν στο σύστημα, να δημιουργήσουν λογαριασμούς, να επιλέξουν μαθήματα και να δοκιμάσουν τη διαδικασία εγγραφής. Η γενική εικόνα που αποκομίσαμε ήταν θετική. Οι φοιτητές χαρακτήρισαν τη διαδικασία «σαφή» και «γρήγορη». Επιπλέον, αρκετοί ανέφεραν ότι το σύστημα μειώνει τα περιθώρια παρεξηγήσεων ή λαθών, καθώς οι

προτιμήσεις και οι επιλογές των φοιτητών είναι σαφώς καταγεγραμμένες και εύκολα διαχειρίσιμες.

Ωστόσο, κατά τη διάρκεια της ανατροφοδότησης προέκυψαν και κάποιες ενδιαφέρουσες παρατηρήσεις. Για παράδειγμα, ένας από τους φοιτητές πρότεινε να υπάρχει μεγαλύτερη ευελιξία στην επιλογή μαθημάτων, έτσι ώστε να μην είναι απαραίτητο να δηλώνονται ακριβώς 5 μαθήματα στη λίστα προτίμησης, καθώς αυτό μπορεί να περιορίζει τις επιλογές σε περιπτώσεις όπου κάποιος ενδιαφέρεται για λιγότερα μαθήματα.

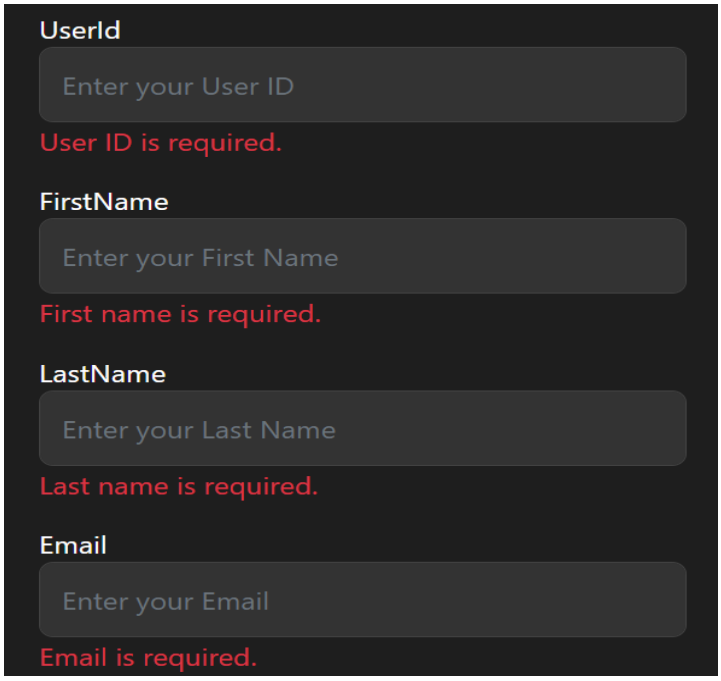
Συνοψίζοντας, η διεπαφή κρίθηκε φιλική, κατανοητή και κατάλληλη για χρήστες με βασική εμπειρία σε ηλεκτρονικά συστήματα.

Κάποια παραδείγματα είναι :



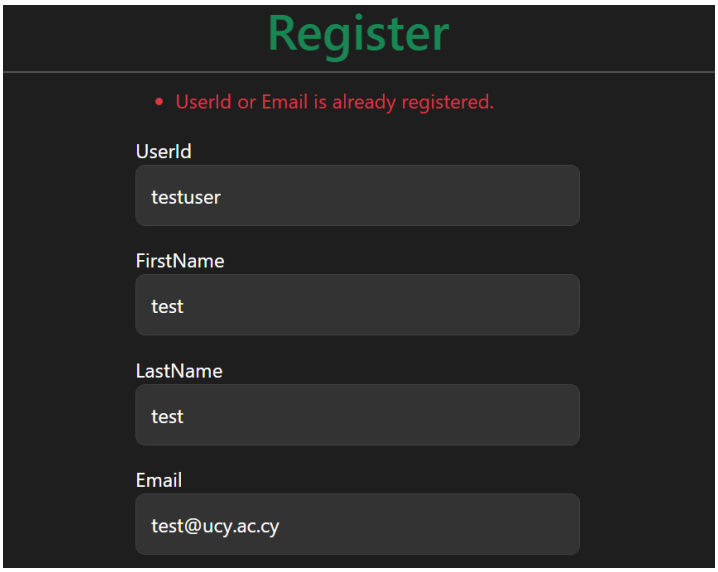
Εικόνα 41 – Περιορισμένες επιλογές

Στην εικόνα 41 βλέπουμε ότι κατά τη δημιουργία της λίστας ένας φοιτητής όταν επιλέξει ένα μάθημα τότε στις επόμενες επιλογές δεν του εμφανίζεται αυτό το μάθημα που ήδη επέλεγξε.



Εικόνα 42 – Αναγκαστικά Πεδία

Στην εικόνα 42 βλέπουμε ότι κατά την εγγραφή όλα τα πεδία είναι απαραίτητα και το email πρέπει να είναι στη σωστή μορφή.



Εικόνα 43 – Προειδοποίηση Μη Επιλυσιμότητας

Στην εικόνα 43 βλέπουμε ότι δεν μπορούν να υπάρχουν δύο χρήστες με το ίδιο UserId και Email.

Εικόνα 44 – Γραμμή Πλοήγησης

Στην εικόνα 44 έχουμε το navigation bar όπου χρησιμοποιείται έτσι ώστε να υπάρχει εύκολη πρόσβαση σε όλες τις οθόνες με ένα απλό πάτημα.

Κεφάλαιο 7

Συμπεράσματα και Μελλοντικές Εργασίες

7.1 Συνολική αξιολόγηση του συστήματος

Το σύστημα που αναπτύχθηκε καλύπτει με επιτυχία τις ανάγκες του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου για την κατανομή φοιτητών σε μαθήματα περιορισμένης επιλογής, με βάση τις προτιμήσεις τους και τους διαθέσιμους πόρους. Η χρήση βελτιστοποιητικού αλγορίθμου μέσω των Google OR-Tools διασφαλίζει τη δίκαιη και αποδοτική κατανομή, ενώ η πλατφόρμα που αναπτύχθηκε σε ASP.NET Core MVC προσφέρει ευχρηστία, επεκτασιμότητα και διαλειτουργικότητα με υπάρχοντες servers του πανεπιστημίου.

Κατά την υλοποίηση, δόθηκε έμφαση στη διασφάλιση της ορθής ροής λειτουργιών, ώστε ο χρήστης (φοιτητής ή διαχειριστής) να μπορεί να εκτελεί τις ενέργειές του χωρίς να δημιουργούνται σφάλματα ή ασάφειες. Το σύστημα εφαρμόζει βασικές αρχές ευχρηστίας, όπως αυτές του Jakob Nielsen: σαφής ιεραρχία πληροφορίας, ορατή ανατροφοδότηση, ελευθερία πλοήγησης, και πρόληψη σφαλμάτων.

Αξίζει να επισημανθεί η εξαιρετική συνεργασία με τον αδερφό μου, Κυριάκο Αντωνουδιού, ο οποίος συνέβαλε καθοριστικά τόσο στον σχεδιασμό του συστήματος όσο και στην υλοποίηση κρίσιμων τμημάτων του. Ιδιαίτερη ήταν η συμβολή του στην ανάπτυξη και ενσωμάτωση του αλγορίθμου, αλλά και στη γενικότερη τεχνική εποπτεία του project. Η ομαδική μας δουλειά επέτρεψε την ομαλή εξέλιξη του έργου και την αντιμετώπιση τεχνικών προκλήσεων με αποτελεσματικότητα.

Το σύστημα είναι ήδη σε λειτουργική κατάσταση, ωστόσο υπάρχουν επιπλέον δυνατότητες που μπορούν να εξεταστούν στο πλαίσιο μιας μελλοντικής επέκτασης.

7.2 Προτάσεις για βελτιώσεις

Ακολουθούν προτάσεις για μελλοντικές βελτιώσεις, οι οποίες μπορούν να ενισχύσουν τη χρηστικότητα και την αυτοματοποίηση του συστήματος, φέρνοντάς το πιο κοντά σε ένα πλήρως ενσωματωμένο οικοσύστημα ακαδημαϊκής διαχείρισης:

- Σύνδεση με Banner Web για την αυτόματη εγγραφή φοιτητών στα μαθήματα. Με την ολοκλήρωση της κατανομής, τα αποτελέσματα θα μπορούν να αποστέλλονται αυτόματα στο σύστημα του πανεπιστημίου, εξαλείφοντας κάθε ανάγκη χειροκίνητης εγγραφής.

- Ενοποίηση του συστήματος ταυτοποίησης με το Banner Web ή MyCSPortal, ώστε οι φοιτητές και οι διαχειριστές να συνδέονται χρησιμοποιώντας τα επίσημα διαπιστευτήρια του πανεπιστημίου, χωρίς την ανάγκη δημιουργίας λογαριασμού στην εφαρμογή.

- Αυτόματος έλεγχος προαπαιτούμενων μαθημάτων, μέσα από τα ακαδημαϊκά δεδομένα του Banner Web. Το σύστημα θα εμποδίζει την επιλογή μαθημάτων για τα οποία ο φοιτητής δεν έχει ολοκληρώσει τις απαραίτητες προϋποθέσεις.

- Ειδοποίηση των επιβλεπόντων καθηγητών σε περιπτώσεις διπλωματικής εργασίας. Όταν ένας φοιτητής δηλώσει ως πρώτη του επιλογή ένα μάθημα που σχετίζεται με τη διπλωματική του, το σύστημα θα αποστέλλει αυτόματα ειδοποίηση στον επιβλέποντα καθηγητή, ώστε να αξιολογήσει το αίτημα.

- Υλοποίηση notification system (μέσω email ή web notifications) για να ενημερώνονται οι φοιτητές σχετικά με την έκβαση της κατανομής, αλλαγές στο σύστημα ή σημαντικές ανακοινώσεις.

- Διερεύνηση εναλλακτικών αλγορίθμων κατανομής, όπως αλγόριθμοι βασισμένοι σε στατιστική μάθηση ή πολυκριτηριακή ανάλυση αποφάσεων, για ακόμα πιο δίκαιες ή προσαρμοσμένες κατανομές.

- Προσθήκη λειτουργίας για δήλωση φοιτητή σε περίπτωση που είναι τριτοετής και λόγω των προαπαιτούμενων μαθημάτων έχει ελάχιστες επιλογές σε μαθήματα περιορισμένης επιλογής.

Βιβλιογραφία

[1] N. Nethercote, P. J. Stuckey, R. Becket, S. Brand, G. J. Duck, and G. Tack, “MiniZinc: Towards a Standard CP Modelling Language,” *Principles and Practice of Constraint Programming (CP 2007)*, Springer, LNCS 4741, pp. 529–543, 2007.

[2] “MiniZinc: The Modelling Language,” [Online]. Available: https://www.minizinc.org/doc-2.6.0/en/index.html [Accessed: April 2025].

[3] D. Esposito, *Programming ASP.NET Core*, Microsoft Press, 2020.

[4] A. Freeman and J. H. Allen, *Pro ASP.NET Core MVC 2*, Apress, 2017.

Παράρτημα