Ατομική Διπλωματική Εργασία

ΑΝΑΠΤΥΞΗ ΑΛΓΟΡΙΘΜΟΥ ΚΑΙ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ ΓΙΑ ΤΟ ΣΥΣΤΗΜΑ ΠΡΟΕΓΓΡΑΦΗΣ ΣΕ ΜΑΘΗΜΑΤΑ ΠΕΡΙΟΡΙΣΜΕΝΗΣ ΕΠΙΛΟΓΗΣ

Κυριάκος Αντωνουδιού

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Αλγορίθμου και Βάσης Δεδομένων για το Σύστημα προεγγραφής σε μαθήματα περιορισμένης επιλογής

Κυριάκος 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

Καταρχάς, θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στην επιβλέπουσα καθηγήτριά μου, κυρία Φιλίππου, για την πολύτιμη καθοδήγηση, την αμέριστη υποστήριξη και τις ουσιαστικές της παρατηρήσεις, οι οποίες συνέβαλαν αποφασιστικά στην ολοκλήρωση της παρούσας διπλωματικής εργασίας.

Ιδιαίτερες ευχαριστίες οφείλω επίσης στον συνεργάτη μου, Στυλιανό Αντωνουδιού, για τη δημιουργική συνεργασία, την αμοιβαία υποστήριξη και την αφοσίωση που επιδείξαμε καθ' όλη τη διάρκεια της ανάπτυξης του συστήματος.

Θα ήθελα ακόμη να ευχαριστήσω το Τμήμα Πληροφορικής για την υψηλού επιπέδου εκπαίδευση που μου παρείχε όλα αυτά τα χρόνια. Η γνώση και οι δεξιότητες που απέκτησα κατά τη διάρκεια των σπουδών μου αποτέλεσαν το θεμέλιο πάνω στο οποίο βασίστηκε η υλοποίηση της παρούσας εργασίας.

Ιδιαίτερη μνεία αξίζει να γίνει στον κ. Δημόπουλο, για τη διδασκαλία του μαθήματος ΕΠΛ 433 - Προγραμματισμός και Ικανοποίηση Περιορισμών, οι γνώσεις του οποίου αποδείχθηκαν καθοριστικές για την κατανόηση και την επιτυχή υλοποίηση του συστήματος προεγγραφών.

Τέλος, ένα μεγάλο ευχαριστώ στην οικογένειά μου για την αδιάκοπη στήριξη, την υπομονή και την πίστη στις προσπάθειές μου, στοιχεία που με ενθάρρυναν να συνεχίσω και να ολοκληρώσω αυτήν την πορεία.

Περίληψη

Η παρούσα Ατομική Διπλωματική Εργασία έχει ως στόχο την ανάπτυξη μιας διαδικτυακής εφαρμογής για την υποστήριξη της διαδικασίας προεγγραφών σε μαθήματα περιορισμένης επιλογής για τους φοιτητές του Τμήματος Πληροφορικής. Η ανάγκη για την υλοποίηση της εφαρμογής προέκυψε από την επιθυμία για βελτίωση της εμπειρίας εγγραφής τόσο για τους φοιτητές όσο και για τα μέλη της Γραμματείας και τους Συντονιστές του Τμήματος, μέσω της αυτοματοποίησης και της ορθολογικότερης διαχείρισης της διαδικασίας.

Κύριος στόχος του συστήματος είναι η ικανοποίηση όσο το δυνατόν περισσότερων φοιτητών όσον αφορά την εγγραφή τους στα επιθυμητά μαθήματα, ελαχιστοποιώντας το άγχος και την ταλαιπωρία που συνοδεύει την παραδοσιακή διαδικασία. Η εφαρμογή υποστηρίζει δύο τύπους χρηστών: τους φοιτητές και τον διαχειριστή του συστήματος (π.χ. κάποιο αρμόδιο μέλος της γραμματείας). Οι φοιτητές μπορούν να υποβάλλουν λίστα προτεραιότητας με τα μαθήματα που επιθυμούν να παρακολουθήσουν συμπεριλαμβανομένου ειδικού όρου για παρακολούθηση ενός μαθήματος(π.χ. κρίνεται απαραίτητο για εκπόνηση διπλωματικής εργασίας). Από την άλλη πλευρά, ο διαχειριστής του συστήματος έχει τη δυνατότητα να ορίζει τα διαθέσιμα μαθήματα, να δημοσιεύει ανακοινώσεις καθώς και να δει και να παρατηρήσει στατιστικά στοιχεία με βάση τις προτιμήσεις των φοιτητών.

Η υλοποίηση βασίστηκε σε σύγχρονες τεχνολογίες ανάπτυξης διαδικτυακών εφαρμογών και αξιοποιεί γνώσεις που αποκτήθηκαν κατά τη διάρκεια των σπουδών, με ιδιαίτερη έμφαση στις τεχνικές προγραμματισμού και ικανοποίησης περιορισμών που διδάχθηκαν στο πλαίσιο του μαθήματος ΕΠΛ 433. Τα αποτελέσματα δείχνουν ότι το σύστημα μπορεί να συμβάλει σημαντικά στη διευκόλυνση της διαδικασίας προεγγραφών και στη βελτίωση της συνολικής φοιτητικής εμπειρίας.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Παρουσίαση του προβλήματος και του σκοπού της εργασίας

Στο Τμήμα Πληροφορικής του Πανεπιστημίου Κύπρου, οι φοιτητές καλούνται να επιλέξουν και να παρακολουθήσουν με επιτυχία συνολικά 6 μαθήματα περιορισμένης επιλογής, τα οποία ανήκουν σε διάφορους τομείς εξειδίκευσης. Σύμφωνα με το πρόγραμμα σπουδών, οι φοιτητές πρέπει να εγγραφούν σε 1 μάθημα περιορισμένης επιλογής κατά το 5ο και το 6ο εξάμηνο, καθώς και σε 2 μαθήματα περιορισμένης επιλογής κατά το 7ο και το 8ο εξάμηνο. Τα μαθήματα αυτά έχουν περιορισμένες θέσεις και οι εγγραφές μέχρι και σήμερα πραγματοποιούνται μέσω του συστήματος Banner, όπως και για τα υπόλοιπα μαθήματα.

Το πρόβλημα που αντιμετωπίζουν σήμερα οι φοιτητές στο Τμήμα Πληροφορικής του Πανεπιστημίου Κύπρου είναι ο έντονος ανταγωνισμός για τις περιορισμένες θέσεις στα μαθήματα περιορισμένης επιλογής. Αυτό έχει ως αποτέλεσμα πολλοί φοιτητές να μην μπορούν να εγγραφούν στα μαθήματα που επιθυμούν, δημιουργώντας αίσθημα απογοήτευσης και αναστάτωσης. Η διαδικασία της εγγραφής, που πραγματοποιείται μέσω του συστήματος Banner, δεν καταφέρνει να εξασφαλίσει τη δικαιοσύνη και την ικανοποίηση των αναγκών των φοιτητών, οδηγώντας σε επιπρόσθετες και χρονοβόρες διευθετήσεις μετά τις εγγραφές. Αυτό έχει αντίκτυπο τόσο στους φοιτητές, οι οποίοι βιώνουν έντονο άγχος και σύγχυση, όσο και στη γραμματεία και τους καθηγητές, οι οποίοι καλούνται να διαχειριστούν και να επιλύσουν τα πολλαπλά αιτήματα και αλλαγές, με αποτέλεσμα να καταναλώνεται

πολύτιμος χρόνος και πόροι. Ουσιαστικά, η όλη διαδικασία προκαλεί μια άσκοπη σπατάλη χρόνου και ενέργειας για όλους τους εμπλεκόμενους, δημιουργώντας περιττό άγχος και επιβαρύνοντας τη συνολική εμπειρία των φοιτητών και των διοικητικών και ακαδημαϊκών υπηρεσιών.

Ο σκοπός της παρούσας εργασίας είναι η ανάπτυξη ενός συστήματος διαδικτυακής εφαρμογής, το οποίο θα επιτρέπει στους φοιτητές να υποβάλλουν τις προσωπικές τους λίστες προτεραιότητας για τα μαθήματα περιορισμένης επιλογής. Μέσω της εφαρμογής αυτής, οι φοιτητές θα μπορούν να καταχωρούν τις προτιμήσεις τους για τα μαθήματα, ενώ ο διαχειριστής του συστήματος θα εκτελεί έναν αλγόριθμο ικανοποίησης περιορισμών, προκειμένου να διασφαλιστεί η αποδοτική και δίκαιη κατανομή των διαθέσιμων θέσεων. Στόχος της ανάπτυξης αυτού του συστήματος είναι η βελτίωση της διαδικασίας εγγραφής, μειώνοντας την ταλαιπωρία για τους φοιτητές και διευκολύνοντας το έργο της γραμματείας και των καθηγητών στην οργανωμένη και δίκαιη κατανομή των θέσεων. Η πλήρης υλοποίηση αυτής της εργασίας έγινε σε συνεργασία με τον Στυλιανό Αντωνουδιού.

1.2 Σκοπός και στόχοι του συστήματος

Ο σκοπός αυτού του συστήματος είναι να αποτελεί μία διαδικτυακή εφαρμογή που να δύναται να τρέχει στα διάφορα προγράμματα περιήγησης όπως είναι το Google Chrome, το Microsoft Edge και το Firefox τα οποία υποστηρίζουν συσκευές όπως ηλεκτρονικοί υπολογιστές, φορητοί υπολογιστές, κινητά τηλέφωνα αλλά και ταμπλέτες. Αυτή η διαδικτυακή εφαρμογή θα υπάρχει ως κουμπί πλοήγησης στην ήδη υπάρχουσα διαδικτυακή σελίδα του Τμήματος Πληροφορικής,την my CS Portal ούτως ώστε να μπορούν να έχουν πρόσβαση στην εφαρμογή τόσο ο διαχειριστής του συστήματος όσο και οι φοιτητές του Τμήματος που αποτελούν και τους δύο τύπους χρηστών της εφαρμογής μας.

Ο τρόπος λειτουργίας της εφαρμογής θα γίνεται μέσω κάποιων διαδικάσιών που προβλέπεται να επιτελέσουν από τη μία ο διαχειριστής του συστήματος και από την άλλη οι φοιτητές που επιθυμούν στο κάθε επερχόμενο εξάμηνο να εγγραφούν σε μαθήματα περιορισμένης επιλογής. Συγκεκριμένα, θα μπορούν οι φοιτητές να βλέπουν τα διαθέσιμα μαθήματα για το επερχόμενο εξάμηνο και τις απαραίτητες πληροφορίες για αυτά, επίσης θα μπορούν να βλέπουν τις ανακοινώσεις του διαχειριστή του συστήματος, όπως επίσης να συμπληρώνουν τη λίστα προτεραιότητας τους με τα μαθήματα που επιθυμούν να εγγραφούν , τον αριθμό

των μαθημάτων αλλά και να δηλώνουν αν θέλουν η πρώτη τους επιλογή για να σχετίζεται με την διπλωματική τους εργασία παραθέτοντας σχετικές πληροφορίες, ενώ στο τέλος θα μπορούν να δουν τα μαθήματα που εν τέλει έχουν εγγραφεί. Από τη μεριά του ο διαχειριστής του συστήματος θα μπορεί να βλέπει στατιστικά γύρω από τις επιλογές των φοιτητών και να κάνει απαραίτητες αλλαγές αν χρειάζονται στο μέγεθος των ακροατηρίων.Ακόμη, θα μπορεί να δημοσιεύει ανακοινώσεις, να εκτελεί τον αλγόριθμο για ανάθεση των μαθημάτων στους φοιτητές, να δημοσιεύει τα διαθέσιμα μαθήματα αλλά και να αναθέτει με το χέρι φοιτητές στα μαθήματα εκτός του αλγόριθμου.

Τέλος, στόχος αυτού του συστήματος είναι να προσφέρει μια δίκαιη, διαφανή και αποδοτική διαδικασία ανάθεσης των φοιτητών στα μαθήματα περιορισμένης επιλογής, λαμβάνοντας υπόψη τις προσωπικές τους προτιμήσεις και τους διαθέσιμους περιορισμούς. Μέσω της αυτοματοποίησης της διαδικασίας με τη χρήση αλγορίθμων, επιδιώκεται να περιοριστεί ο άσκοπος ανταγωνισμός, να μειωθεί το άγχος και η ταλαιπωρία για φοιτητές, γραμματεία και καθηγητές, και να αξιοποιηθεί καλύτερα ο διαθέσιμος χρόνος και οι πόροι του Τμήματος. Παράλληλα, η εφαρμογή ενισχύει τη διαφάνεια και την αξιοπιστία στη διαδικασία εγγραφής, συμβάλλοντας στη γενικότερη αναβάθμιση της ακαδημαϊκής εμπειρίας των φοιτητών.

1.3 Μεθοδολογία

Σεπτέμβριος – Οκτώβριο: Κατά τους πρώτους δύο μήνες, εστιάσαμε στην κατανόηση των βασικών αρχών του μαθήματος «Προγραμματισμός Ικανοποίησης Περιορισμών» (CSP), με στόχο να διερευνήσουμε πώς αυτές οι τεχνικές μπορούν να αξιοποιηθούν για τη δημιουργία ενός αποτελεσματικού αλγορίθμου κατανομής φοιτητών στα μαθήματα περιορισμένης επιλογής. Παράλληλα, προχωρήσαμε σε ανάλυση προδιαγραφών και απαιτήσεων του υπό ανάπτυξη συστήματος, ώστε να διαμορφώσουμε ένα σαφές πλαίσιο εντός του οποίου θα μπορούσε να εφαρμοστεί ο προγραμματισμός περιορισμών. Η προσέγγιση αυτή μας βοήθησε να κατανοήσουμε τις ιδιαιτερότητες του προβλήματος και να ξεκινήσουμε τη διαμόρφωση μιας λύσης που να συνδυάζει ακαδημαϊκή μεθοδολογία με τις πραγματικές ανάγκες του Τμήματος.

Νοέμβριος - Δεκέμβριος 2024: Στη συνέχεια, προχωρήσαμε στην ακριβή αποσαφήνιση των λειτουργιών του συστήματος, ώστε να διασφαλίσουμε ότι όλες οι απαιτήσεις των εμπλεκόμενων χρηστών καλύπτονται αποτελεσματικά. Παράλληλα, πραγματοποιήσαμε δοκιμές με διάφορους αλγορίθμους κατανομής, συγκρίνοντας τα αποτελέσματά τους τόσο ως

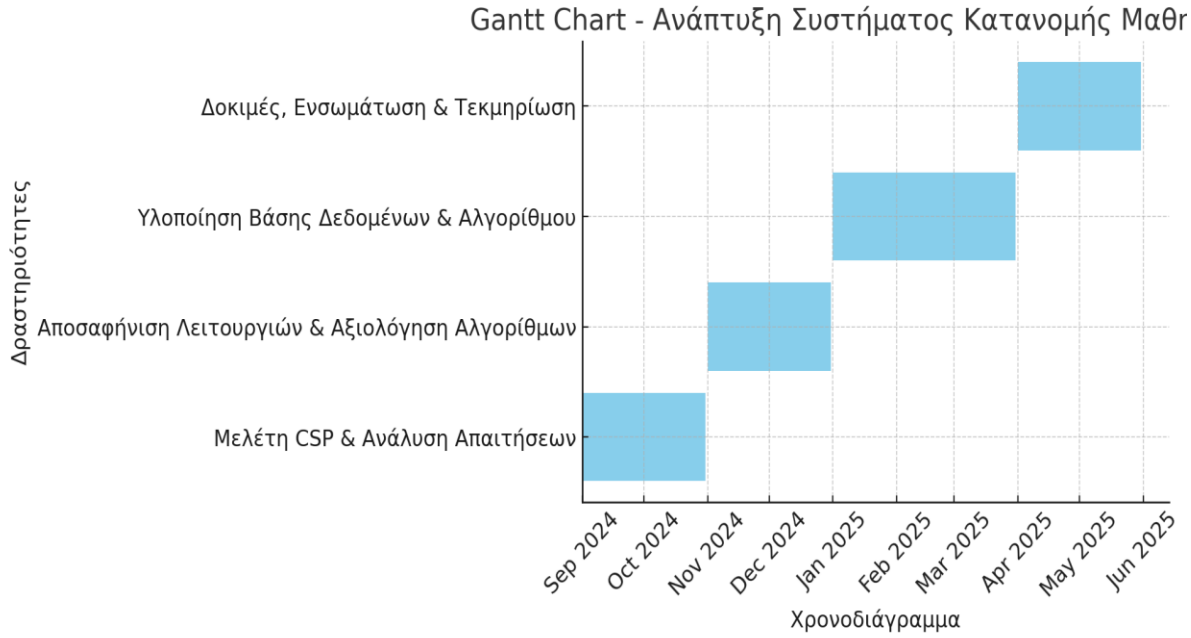
προς τη δικαιοσύνη όσο και ως προς την αποτελεσματικότητα. Μέσα από αυτή τη διαδικασία, καταλήξαμε ότι η πιο κατάλληλη προσέγγιση για την περίπτωσή μας είναι αυτή που βασίζεται στη μέτρηση της ικανοποίησης των φοιτητών, δίνοντας έμφαση στην κάλυψη των προτιμήσεών τους στον μέγιστο δυνατό βαθμό.

Ιανουάριος - Μάρτιος 2025: Ακολούθως, περάσαμε στο στάδιο της υλοποίησης του συστήματος, με εμένα να αναλαμβάνω κυρίως τη δημιουργία της βάσης δεδομένων, χρησιμοποιώντας τον Microsoft SQL Server σε συνδυασμό με το Entity Framework για την αλληλεπίδραση με την εφαρμογή. Παράλληλα, ανέλαβα την υλοποίηση του βασικού αλγορίθμου κατανομής φοιτητών, τον οποίο υλοποίησα σε .NET αξιοποιώντας τη βιβλιοθήκη Google OR-Tools, ώστε να μπορέσουμε να διαχειριστούμε αποτελεσματικά τους περιορισμούς και τις προτιμήσεις που απαιτεί η διαδικασία ανάθεσης.

Απρίλιος - Μάιος 2025: Πραγματοποιήθηκαν δοκιμές του συστήματος Unit Testing και η τελική ενσωμάτωση του αλγορίθμου στη διαδικτυακή εφραμογή. Επίσης, συντάχθηκε το τελικό έγγραφο της διπλωματικής εργασίας, το οποίο περιλαμβάνει την τεκμηρίωση των τεχνικών επιλογών, τη μεθοδολογία, και την αξιολόγηση του συστήματος.

Ακολουθεί το Gantt chart της χρονικής εξέλιξης του έργου:

Gantt Chart – Χρονοδιάγραμμα Εργασιών



1.4 Δομή Διπλωματικής Εργασίας

Η παρούσα διπλωματική εργασία οργανώνεται σε επτά κεφάλαια, τα οποία καλύπτουν συστηματικά όλα τα στάδια της ανάπτυξης του συστήματος ανάθεσης μαθημάτων, από τον εντοπισμό του προβλήματος μέχρι την αξιολόγηση των αποτελεσμάτων και τις προτάσεις για μελλοντική εργασία.

Στο Κεφάλαιο 1 πραγματοποιείται η εισαγωγή στο αντικείμενο μελέτης. Παρουσιάζεται το πρόβλημα που αντιμετωπίζουν οι φοιτητές κατά τη διαδικασία επιλογής μαθημάτων περιορισμένης επιλογής, προσδιορίζεται ο σκοπός και οι στόχοι της εργασίας και παρατίθεται συνοπτικά η δομή της διπλωματικής.

Το Κεφάλαιο 2 αφορά την ανάλυση απαιτήσεων του συστήματος. Περιλαμβάνει την περιγραφή και τα αποτελέσματα σχετικού ερωτηματολογίου, την ανάλυση λειτουργικών και μη λειτουργικών απαιτήσεων, την τεκμηρίωση της ανάγκης για μια δίκαιη και διαφανή διαδικασία κατανομής μαθημάτων, καθώς και μια ευρύτερη αναφορά στη μεθοδολογία που ακολουθήθηκε και το διαμεροισμό των εργασιών μέσω συγκεκριμένων εργαλείων.

Στο Κεφάλαιο 3 παρατίθεται το θεωρητικό υπόβαθρο που υποστηρίζει την υλοποίηση του συστήματος. Γίνεται εισαγωγή στον προγραμματισμό ικανοποίησης περιορισμών, αναλύονται το εργαλείο Google OR-Tools το οποίο χρησιμοπιήθηκε στην τελική υλοποίηση του συστήματος, και πραγματοποιείται συγκριτική αξιολόγηση των δυνατοτήτων τους ως προς την εφαρμογή τους στο συγκεκριμένο πρόβλημα.

Το Κεφάλαιο 4 επικεντρώνεται στη σχεδίαση της Βάσης Δεδομένων και του Αλγορίθμου στο σύστημα. Περιγράφονται οι βασικές οντότητες και οι μεταξύ τους σχέσεις, παρουσιάζεται η σχεδίαση της βάσης δεδομένων και αναλύεται η αρχιτεκτονική της εφαρμογής, βασισμένη στο πρότυπο .NET MVC. Παράλληλα, γίνεται αναλυτική περιγραφή του σχεδιασμού του Αλγόριθμου ως Πρόβλημα Ικανοποίησης Περιορισμών , όπως επίσης και η αλληλεπίδραση της Βάσης Δεδομένων με τον Αλγόριθμο.

Στο Κεφάλαιο 5 αναπτύσσεται η υλοποίηση του αλγορίθμου. Γίνεται αναλυτική παρουσίαση της μεθοδολογίας σχεδίασης για τη δίκαιη κατανομή μαθημάτων, της υλοποίησής του με το εργαλείο OR-Tools, και την αλληλεπίδραση με τη Βάση Δεδομένων σε επίπεδο υλοποίησης.

Το Κεφάλαιο 6 περιγράφει την αξιολόγηση που υλοποιήθηκε με Unit Testing και συγκεκριμένα το xUnit Framework που χρησιμοποιήθηκε . Επιπρόσθετα , γίνετα μία αναλυτική σύγκριση μεταξύ του επιλεγμένου αλγορίθμου και άλλων παραλλαγών που σχεδιάστηκαν για το πρόβλημα.

Τέλος, το Κεφάλαιο 7 συνοψίζει τα συμπεράσματα της εργασίας και παρουσιάζει διάφορες γνώσεις και εμπειρίες που απέκτησα μέσω της διπλωματικής εργασίας, όπως επίσης και τις σημαντικότερες δυσκολίες και προκλήσεις που κλήθηκα να αντιμετωπίσω. Τέλος προτείνει κατευθύνσεις για πιθανές μελλοντικές βελτιώσεις και επεκτάσεις του συστήματος.

Κεφάλαιο 2

Ανάλυση Απαιτήσεων

2.1 Περιγραφή του ερωτηματολογίου και των αποτελεσμάτων

Για τη συλλογή δεδομένων σχετικά με τις απόψεις, τις ανάγκες και τις προτιμήσεις των φοιτητών ως προς τη διαδικασία επιλογής μαθημάτων περιορισμένης επιλογής, δημιουργήθηκε και διανεμήθηκε ένα ερωτηματολόγιο. Το ερωτηματολόγιο εστάλη στους φοιτητές μέσω της Γραμματείας του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου και αποτέλεσε βασικό εργαλείο στην παρούσα έρευνα. Στόχος του ήταν να καταγράψει τις εμπειρίες των φοιτητών από τη διαδικασία εγγραφής, να αναδείξει τα προβλήματα που αντιμετωπίζουν, να αποτυπώσει τις προτεραιότητες που θέτουν κατά την επιλογή μαθημάτων, καθώς και να συγκεντρώσει προτάσεις για βελτίωση του υφιστάμενου συστήματος. Οι ερωτήσεις του ερωτηματολογίου κάλυπταν ένα ευρύ φάσμα θεμάτων, όπως το έτος σπουδών, την ικανοποίηση από τη διαδικασία εγγραφής, τη συχνότητα με την οποία οι φοιτητές εγγράφονται στα επιθυμητά μαθήματα, και την επίδραση της διαθεσιμότητας μαθημάτων στην ακαδημαϊκή τους πορεία. Επιπλέον, διερευνήθηκαν οι απόψεις των φοιτητών σχετικά με κριτήρια όπως το αντικείμενο του μαθήματος, ο διδάσκων, η δυσκολία, καθώς και η δυνατότητα υιοθέτησης ενός νέου συστήματος κατάθεσης λίστας προτιμήσεων και αυτόματης ανάθεσης μέσω αλγορίθμου ικανοποίησης περιορισμών. Τα δεδομένα που συγκεντρώθηκαν από το ερωτηματολόγιο αποτέλεσαν πολύτιμη πηγή πληροφόρησης για την ανάπτυξη ενός σύγχρονου και αποδοτικού συστήματος προεγγραφής μαθημάτων, που αποτελεί αντικείμενο διπλωματικής εργασίας .

Τα αποτελέσματα της έρευνας, στην οποία συμμετείχαν φοιτητές του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου με το 85% των συμμετεχόντων να είναι φοιτητές 3ου και 4ου έτους —δηλαδή φοιτητές με άμεση εμπειρία των υφιστάμενων δυσκολιών στην εγγραφή σε μαθήματα περιορισμένης επιλογής— ανέδειξαν σημαντικά προβλήματα στη διαδικασία. Συγκεκριμένα, περίπου το 50% των φοιτητών που συμμετείχε στην έρευνα δήλωσε ότι στη τελευταία περίοδο εγγραφών γράφτηκε σε 2 μαθήματα ενώ οι υπόλοιποι γράφτηκαν σε 1 ή 3 μαθήματα, με κανέναν φοιτητή να αναφέρει ότι γράφτηκε σε 4 μαθήματα. Ιδιαίτερα αξιοσημείωτο είναι το γεγονός ότι μόνο το 15,6% των φοιτητών καταφέρνει πάντοτε να εγγράφεται στα μαθήματα της πρώτης του επιλογής, ενώ οι υπόλοιποι αναγκάζονται να στραφούν σε εναλλακτικά μαθήματα ή να περιμένουν την επόμενη περίοδο εγγραφών. Παράλληλα, πάνω από τους μισούς συμμετέχοντες δήλωσαν ότι η έλλειψη διαθέσιμων θέσεων σε μαθήματα έχει επηρεάσει αρνητικά την ακαδημαϊκή τους πρόοδο. Όσον αφορά τα κριτήρια επιλογής μαθημάτων, το αντικείμενο του μαθήματος αναδείχθηκε ως ο σημαντικότερος παράγοντας για τους φοιτητές, ενώ η πλειοψηφία εξέφρασε την αντίθεσή της στη χρήση πιστωτικών μονάδων ή γενικού μέσου όρου ως κριτήρια προτεραιότητας. Επιπλέον, οι περισσότεροι φοιτητές θεωρούν ότι ο αριθμός των μαθημάτων που επιθυμεί να δηλώσει ένας φοιτητής δεν θα πρέπει να επηρεάζει τη σειρά προτεραιότητας στην εγγραφή. Τέλος, η συντριπτική πλειοψηφία των φοιτητών δήλωσε ότι είναι αναγκαία η παροχή προτεραιότητας σε φοιτητές των οποίων η επιλογή μαθημάτων σχετίζεται άμεσα με τη διπλωματική τους εργασία. Τα αποτελέσματα αυτά αναδεικνύουν την ανάγκη για τη δημιουργία ενός νέου, σύγχρονου και δίκαιου συστήματος προεγγραφής μαθημάτων, το οποίο αναπτύχθηκε στο πλαίσιο της παρούσας διπλωματικής εργασίας σε συνεργασία με τον συμφοιτητή μου, Στυλιανό Αντωνουδιού.

➔Στο **Παράρτημα Α** παρατίθονται όλα τα αποτελέσματα από την έρευνα

2.2 Ανάλυση των Απαιτήσεων του συστήματος και του αλγορίθμου

Κατά τη διάρκεια των συζητήσεών μας με την κ. Δώρα από τη γραμματεία και την κ. Φιλίππου, συγκεντρώσαμε σημαντικά σχόλια και προτάσεις που συνέβαλαν στη διαμόρφωση των απαιτήσεων του υπό ανάπτυξη συστήματος. Η κ. Δώρα επεσήμανε ιδιαίτερα τη σημασία της ύπαρξης ενός εργαλείου που να αυτοματοποιεί τη διαδικασία προεγγραφής, μειώνοντας τον φόρτο εργασίας και τις παρεμβάσεις της γραμματείας, ενώ δήλωσε ότι της άρεσε πολύ η ιδέα να υπάρχουν στατιστικά στοιχεία διαθέσιμα πριν από την εκτέλεση του αλγορίθμου,

ώστε να υπάρχει καλύτερη εικόνα για τη ζήτηση και την κατανομή των προτιμήσεων. Η κ. Φιλίππου από την πλευρά της έδωσε έμφαση στην ανάγκη για διαφάνεια και ισότιμη μεταχείριση των φοιτητών, ειδικά σε περιπτώσεις όπου υπάρχουν περιορισμοί χωρητικότητας ή ειδικές προτεραιότητες. Και οι δύο συμφώνησαν πως το νέο σύστημα πρέπει να είναι απλό στη χρήση αλλά και αρκετά ευέλικτο ώστε να καλύπτει τις λειτουργικές ανάγκες τόσο της γραμματείας όσο και των διδασκόντων.

Το υπό ανάπτυξη σύστημα αποτελεί μια διαδικτυακή εφαρμογή (web app), η οποία προορίζεται να χρησιμοποιηθεί επισήμως από το Τμήμα Πληροφορικής του Πανεπιστημίου Κύπρου για τη διαχείριση των εγγραφών στα μαθήματα περιορισμένης επιλογής. Οι βασικές απαιτήσεις του συστήματος απορρέουν τόσο από τα αποτελέσματα της έρευνας που διεξήχθη μεταξύ των φοιτητών όσο και από τις λειτουργικές ανάγκες της γραμματείας και των διδασκόντων του Τμήματος. Ο κύριος στόχος είναι η παροχή ενός σύγχρονου, διαφανούς και αποδοτικού εργαλείου που θα διευκολύνει τη διαδικασία προεγγραφής, μειώνοντας τον φόρτο εργασίας, την ταλαιπωρία και τις αδικίες που παρατηρούνται στο υπάρχον σύστημα εγγραφών.

Η βασική λειτουργικότητα του συστήματος προβλέπει ότι οι φοιτητές θα καταχωρούν ηλεκτρονικά τη λίστα προτίμησής τους για τα διαθέσιμα μαθήματα περιορισμένης επιλογής. Κάθε φοιτητής θα μπορεί να ταξινομεί τα μαθήματα κατά σειρά προτίμησης, επιτρέποντας έτσι στο σύστημα να συλλέγει και να οργανώνει τις ατομικές προτιμήσεις σε μια ενιαία βάση δεδομένων. Η γραμματεία, μέσω ενός απλού περιβάλλοντος διαχείρισης, θα μπορεί να εξάγει αυτές τις προτιμήσεις και να ενεργοποιεί την εκτέλεση του αλγορίθμου ανάθεσης.

Ο αλγόριθμος, ο οποίος αναπτύχθηκε στο πλαίσιο αυτής της διπλωματικής εργασίας, καλείται να ικανοποιήσει μια σειρά από κρίσιμες απαιτήσεις. Συγκεκριμένα, λαμβάνοντας υπόψη τις προτεραιότητες των φοιτητών, πρέπει να αναθέτει τους φοιτητές στα μαθήματα με τρόπο όσο το δυνατόν πιο δίκαιο και ισόρροπο. Η δικαιοσύνη ορίζεται εδώ ως η μέγιστη δυνατή ικανοποίηση των προτιμήσεων όλων των συμμετεχόντων, ελαχιστοποιώντας ταυτόχρονα περιπτώσεις όπου κάποιος φοιτητής δεν εγγράφεται σε κανένα από τα μαθήματα προτίμησής του. Επιπλέον, ο αλγόριθμος πρέπει να λαμβάνει υπόψη περιορισμούς όπως ο μέγιστος αριθμός φοιτητών ανά μάθημα, πιθανές προτεραιότητες φοιτητών που χρειάζονται συγκεκριμένα μαθήματα για τη διπλωματική τους εργασία, καθώς και την ανάγκη για ισότιμη μεταχείριση ανεξαρτήτως αριθμού μαθημάτων που επιθυμεί να παρακολουθήσει κάθε φοιτητής.

Η επιτυχής ολοκλήρωση και λειτουργία του συστήματος αναμένεται να οδηγήσει σε μια σημαντική βελτίωση της διαδικασίας εγγραφών, αυξάνοντας την ικανοποίηση των φοιτητών και μειώνοντας τη διοικητική επιβάρυνση για το προσωπικό του Τμήματος.

2.3 Ανάλυση της ανάγκης για δίκαιη κατανομή των μαθημάτων

Η ανάγκη για δίκαιη κατανομή των φοιτητών στα μαθήματα περιορισμένης επιλογής προκύπτει κυρίως από τις αδυναμίες της υφιστάμενης διαδικασίας, η οποία βασίζεται στη λογική του «όποιος προλάβει». Σε αυτό το πλαίσιο, η ταχύτητα σύνδεσης και η διαθεσιμότητα των φοιτητών τη συγκεκριμένη στιγμή παίζουν καθοριστικό ρόλο, με αποτέλεσμα η διαδικασία να είναι συχνά άδικη και άνιση. Ένας φοιτητής με καλύτερη πρόσβαση στο διαδίκτυο ή που ήταν απλώς πιο τυχερός, μπορεί να εγγραφεί σε όλα τα μαθήματα που επιθυμεί, ενώ κάποιος άλλος, λόγω καθυστέρησης ή τεχνικού προβλήματος, μπορεί να μείνει εκτός όλων των επιλογών του. Αυτή η κατάσταση προκαλεί απογοήτευση, αίσθημα αδικίας και συχνά δημιουργεί προβλήματα στην ακαδημαϊκή πορεία των φοιτητών. Συνεπώς, η μετάβαση σε ένα σύστημα που βασίζεται σε αντικειμενικά κριτήρια και εξετάζει τις προτιμήσεις όλων πριν την κατανομή, αποτελεί κρίσιμη ανάγκη για την ενίσχυση της ισότητας και της αξιοπιστίας της διαδικασίας.

Η χρήση του γενικού μέσου όρου (GPA) ως κριτήριο προτεραιότητας δεν επιλύει ουσιαστικά το πρόβλημα, αφού δημιουργεί νέα στρώματα ανισότητας, δίνοντας άδικο πλεονέκτημα σε φοιτητές που ενδεχομένως να έχουν διαφορετικές ακαδημαϊκές συνθήκες ή ρυθμούς προόδου. Αντίθετα, ένα σύστημα που βασίζεται στη δίκαιη κατανομή στοχεύει στο να μεγιστοποιεί τη συνολική ικανοποίηση των φοιτητών, διασφαλίζοντας ότι ακόμη και ο λιγότερο ικανοποιημένος φοιτητής απολαμβάνει ένα ικανοποιητικό επίπεδο εκπλήρωσης των προτιμήσεών του.

Η δίκαιη κατανομή είναι σημαντική για διάφορους λόγους. Πρώτον, ενισχύει το αίσθημα ισότητας και δικαιοσύνης μέσα στο πανεπιστημιακό περιβάλλον, το οποίο είναι κρίσιμο για τη διατήρηση ενός θετικού και υποστηρικτικού κλίματος μεταξύ φοιτητών και προσωπικού. Δεύτερον, μειώνει τα επίπεδα άγχους και απογοήτευσης στους φοιτητές, οι οποίοι συχνά ανησυχούν αν θα μπορέσουν να ολοκληρώσουν εγκαίρως το πρόγραμμα σπουδών τους λόγω

αδυναμίας πρόσβασης σε βασικά μαθήματα. Τρίτον, υποστηρίζει την ακαδημαϊκή πρόοδο και αποτρέπει καθυστερήσεις στην αποφοίτηση, καθώς όλοι οι φοιτητές έχουν πιο ίσες ευκαιρίες να εγγραφούν στα απαιτούμενα ή επιθυμητά μαθήματα.

Επιπλέον, ένα δίκαιο σύστημα, που να διασφαλίζει ίδια επίπεδα δικαιοσύνης ανάμεσα στους φοιτητές και να μην βασίζεται σε τυχαιότητα, όπως ισχύει τώρα. Επιπλέον το σύστημα πρέπει να προάγει τη διαφάνεια και να μειώνει την ανάγκη για συνεχείς διαπραγματεύσεις και αιτήσεις αλλαγών προς τη γραμματεία και τους καθηγητές, κάτι που με τη σειρά του εξοικονομεί πολύτιμο χρόνο και πόρους για όλο το ακαδημαϊκό προσωπικό. Τέλος, ένα δίκαιο σύστημα κατανομής ενισχύει την εικόνα του ιδρύματος προς τα έξω, δείχνοντας ότι επενδύει στη βελτίωση των διαδικασιών και στην υποστήριξη των φοιτητών του με σύγχρονες, αντικειμενικές και αποτελεσματικές λύσεις.

Η ανάπτυξη και εφαρμογή ενός συστήματος που θα στηρίζεται στην αρχή της δίκαιης κατανομής, με χρήση αλγορίθμων ικανοποίησης περιορισμών, αποτελεί επομένως αναγκαία και σημαντική εξέλιξη για τη βελτίωση της συνολικής εκπαιδευτικής εμπειρίας στο Τμήμα Πληροφορικής του Πανεπιστημίου Κύπρου.

2.4 Μεθοδολογία Ανάπτυξης και Συνεργατικά Εργαλεία

Κατά την ανάπτυξη της διαδικτυακής εφαρμογής για την υποστήριξη της διαδικασίας προεγγραφών, επιλέξαμε να ακολουθήσουμε πρακτικές που εντάσσονται στο πλαίσιο των Agile μεθοδολογιών ανάπτυξης λογισμικού, με στόχο την ευελιξία, τη σταδιακή βελτίωση του συστήματος και την καλύτερη συνεργασία της ομάδας.

Η φιλοσοφία του Agile μας επέτρεψε να χωρίσουμε την εργασία σε διαδοχικούς κύκλους ανάπτυξης (iterations/sprints), όπου κάθε κύκλος εστίαζε σε συγκεκριμένες λειτουργίες του συστήματος. Κατά την έναρξη κάθε κύκλου, καθορίζαμε με σαφήνεια τις απαιτήσεις και τα παραδοτέα, ενώ στο τέλος του κύκλου αξιολογούσαμε το παραχθέν αποτέλεσμα και εντοπίζαμε σημεία προς βελτίωση. Η συνεχής ανατροφοδότηση μας βοήθησε να προσαρμόζουμε την κατεύθυνση της ανάπτυξης ανάλογα με τις δυσκολίες που προέκυπταν ή τις ιδέες που αναδύονταν στην πορεία.

Η συνεργασία με τον συμφοιτητή μου ήταν ιδιαίτερα αποδοτική, καθώς καταφέραμε να λειτουργούμε σαν μια πραγματική ομάδα ανάπτυξης λογισμικού, μοιράζοντας τις εργασίες, οργανώνοντας τα tasks με σαφήνεια και ακολουθώντας διαδικασίες που μοιάζουν με εκείνες που εφαρμόζονται σε επαγγελματικά περιβάλλοντα. Κατά τη διάρκεια κάθε εβδομάδας, κάναμε σύντομα meetings για να συζητήσουμε την πρόοδο, να αναθέσουμε νέες εργασίες και να λύσουμε απορίες ή προβλήματα που εντοπίστηκαν. Αυτό συνέβαλε όχι μόνο στην τεχνική πρόοδο του έργου, αλλά και στην απόκτηση πολύτιμης εμπειρίας ομαδικής και επαγγελματικής συνεργασίας.

Στο πλαίσιο των ευέλικτων (Agile) προσεγγίσεων που υιοθετήσαμε κατά την ανάπτυξη της εφαρμογής, εφαρμόσαμε και την Kanban μεθοδολογία, ένα απλό αλλά εξαιρετικά αποτελεσματικό μοντέλο διαχείρισης ροής εργασίας, που μας επέτρεψε να έχουμε άμεση εποπτεία της προόδου του έργου και να ελαχιστοποιήσουμε τα σημεία συμφόρησης στην παραγωγή.

Η βασική φιλοσοφία του Kanban βασίζεται στην οπτική απεικόνιση των εργασιών και στην προώθησή τους μέσα από διακριτά στάδια της διαδικασίας ανάπτυξης. Η χρήση αυτής της μεθοδολογίας μας βοήθησε να εντοπίζουμε εύκολα καθυστερήσεις ή bottlenecks, να προτεραιοποιούμε τις πιο κρίσιμες εργασίες και να εστιάζουμε στον περιορισμό του "Work in Progress" (WIP), ώστε να διατηρείται μια σταθερή και παραγωγική ροή.

Για την πρακτική εφαρμογή της μεθοδολογίας επιλέξαμε να χρησιμοποιήσουμε το εργαλείο KanbanFlow, το οποίο προσφέρει ένα εξαιρετικά ευέλικτο περιβάλλον οργάνωσης εργασιών και είναι προσαρμόσιμο στις ανάγκες κάθε ομάδας.

Το KanbanFlow μας έδωσε τη δυνατότητα να δημιουργήσουμε έναν κοινό πίνακα εργασιών (board), τον οποίο διαμορφώσαμε με τις βασικές στήλες:

- To Do (εργασίες προς υλοποίηση),
- In Progress (εργασίες σε εξέλιξη),
- Testing (δοκιμές λειτουργιών),

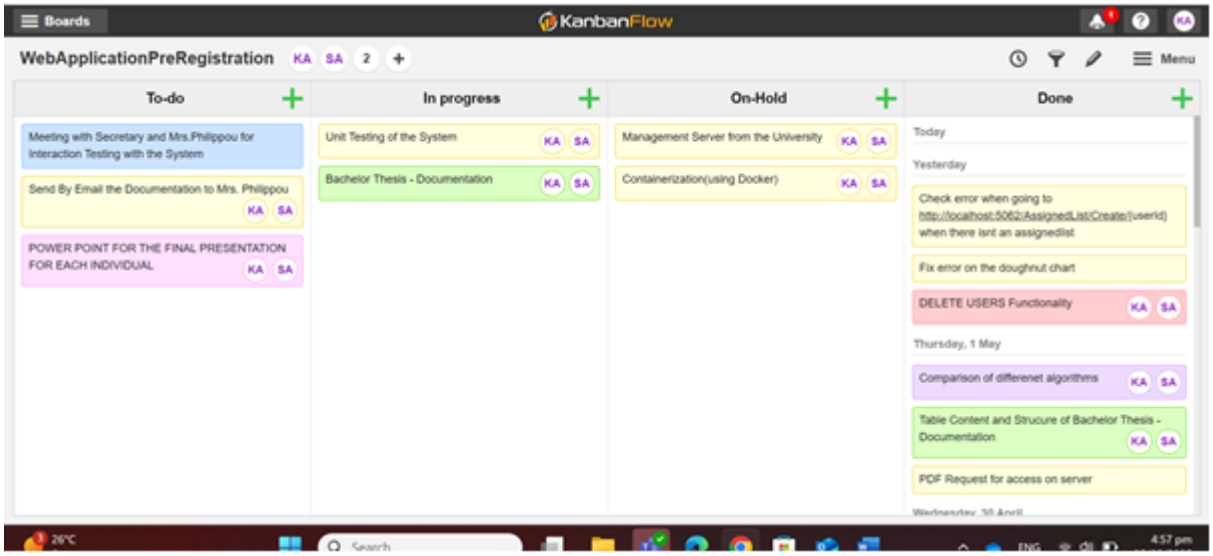
- Done (ολοκληρωμένες εργασίες).

Κάθε εργασία αναπαριστάτο ως μια κάρτα, στην οποία μπορούσαμε να ορίσουμε τίτλο, περιγραφή, ημερομηνία παράδοσης, αλλά και χρονόμετρο Pomodoro, χαρακτηριστικό που μας βοήθησε να εστιάζουμε καλύτερα σε μικρά χρονικά διαστήματα παραγωγικής δουλειάς.

Επιπλέον, αξιοποιήσαμε τις δυνατότητες κατηγοριοποίησης με χρώματα (labels) για τον διαχωρισμό μεταξύ frontend, backend, σχεδιασμού, τεκμηρίωσης κ.λπ., και ορίσαμε υπευθύνους ανά εργασία, διευκολύνοντας έτσι τη συνεργασία και την κατανομή ρόλων. Η συνεχής ενημέρωση του πίνακα μας κρατούσε οργανωμένους και ενημερωμένους για την πρόοδο της ομάδας συνολικά.

Η χρήση του KanbanFlow, σε συνδυασμό με τη μεθοδολογία Kanban, συνέβαλε σημαντικά στη διατήρηση της πειθαρχίας και της οργάνωσης καθ' όλη τη διάρκεια της ανάπτυξης, προσφέροντας ταυτόχρονα ευελιξία, σαφήνεια και μετρήσιμη πρόοδο.

Πιο κάτω παρατίθεται σχετικό στιγμιότυπο οθόνης:



Εικόνα 1 – KanbanFlow

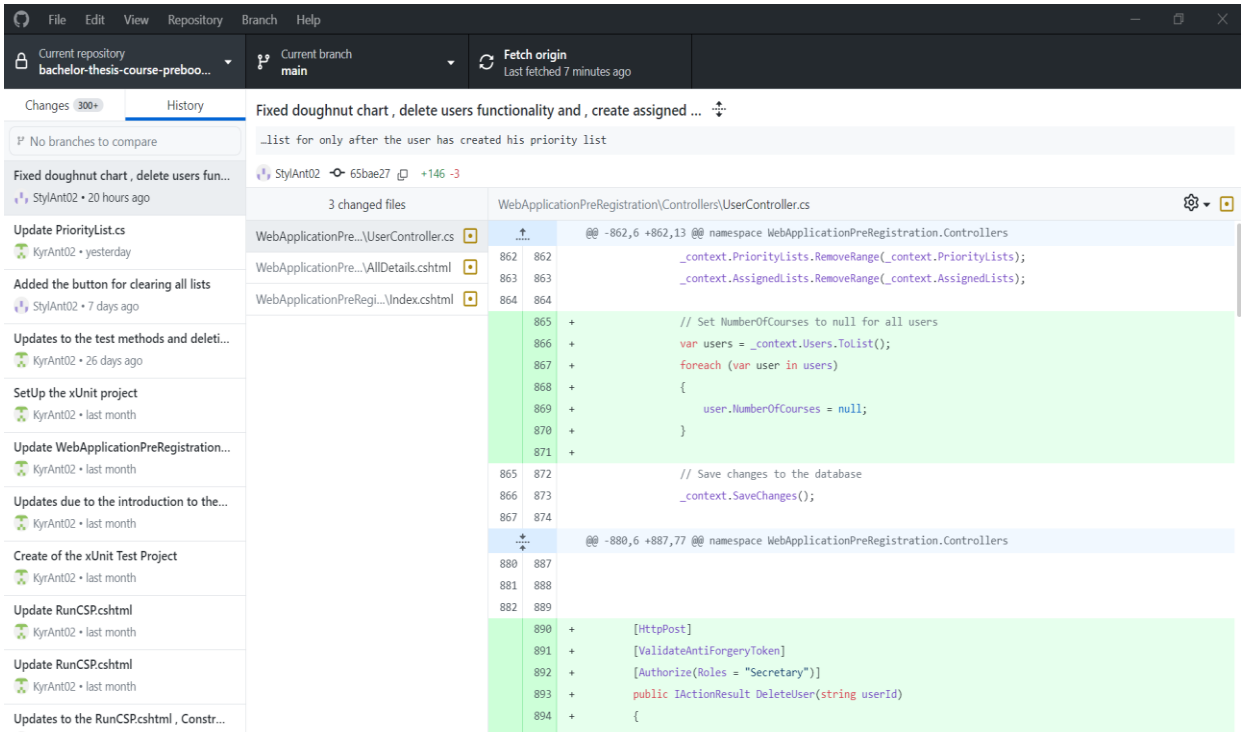
Καθ' όλη τη διάρκεια της ανάπτυξης της εφαρμογής, το GitHub αποτέλεσε το κύριο εργαλείο μας για έλεγχο έκδοσης, διαχείριση κώδικα και συνεργατική εργασία. Πρόκειται

για μια πλατφόρμα βασισμένη στο σύστημα ελέγχου έκδοσης Git, που μας επέτρεψε να παρακολουθούμε τις αλλαγές στον πηγαίο κώδικα, να δουλεύουμε παράλληλα σε διαφορετικά μέρη του έργου και να διατηρούμε μια σταθερή και ασφαλή πορεία ανάπτυξης. Εκτός από την αποθήκευση και διαχείριση του κώδικα, το GitHub αποδείχθηκε ένα εξαιρετικά χρήσιμο εργαλείο για την ομαδική εργασία και τη συνεργασία μεταξύ των μελών της ομάδας. Μέσω της δυνατότητας δημιουργίας διαφορετικών branches, μπορέσαμε να εργαζόμαστε παράλληλα σε επιμέρους χαρακτηριστικά της εφαρμογής, χωρίς να διαταράσσεται η σταθερή έκδοση του έργου. Με αυτόν τον τρόπο, διασφαλίσαμε τόσο την ευελιξία όσο και την ασφάλεια κατά την ανάπτυξη του λογισμικού.

Επιπλέον, αξιοποιήθηκαν λειτουργίες όπως τα pull requests και τα code reviews για να γίνεται έλεγχος και συγχώνευση αλλαγών με οργανωμένο τρόπο, ενισχύοντας την ποιότητα του κώδικα και ενθαρρύνοντας τη συνεχή ανατροφοδότηση μεταξύ μας. Καθώς η εργασία μας εξελισσόταν, χρησιμοποιήσαμε το GitHub επίσης για την τεκμηρίωση της προόδου μέσω issues και milestones, επιτρέποντάς μας να παρακολουθούμε τις εκκρεμότητες, να οργανώνουμε τις προτεραιότητες και να καθορίζουμε χρονικά πλαίσια.

Η συμβολή του GitHub ήταν καθοριστική όχι μόνο για την τεχνική πτυχή του project αλλά και για την ενίσχυση της επαγγελματικής μας εμπειρίας, προσφέροντάς μας τη δυνατότητα να εφαρμόσουμε βέλτιστες πρακτικές ανάπτυξης λογισμικού όπως αυτές συμβαίνουν σε επαγγελματικά περιβάλλοντα ανάπτυξης.

Πιο κάτω παρτίθεται και σχετικό στιγμιότυπο οθόνης από την εφαρμογή GitHub Desktop στην Εικόνα 2, και το σχετικό link στο repository:



Εικόνα 2 – GitHub Desktop

https://github.com/KyrAnt02/bachelor-thesis-course-prebooking

Σύνδεσμος GitHub Repository

2.5 Ανασκόπηση Συστήματος

Το σύστημα προεγγραφής σε μαθήματα περιορισμένης επιλογής αναπτύχθηκε μέσω μιας συνεργατικής προσέγγισης μεταξύ εμένα και του συμφοιτητή μου Στυλιανού 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 πρωτοτύπου, χρησιμοποιώντας το εργαλείο πρωτοτυποποίησης JustInMind.
- 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.

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,Controllers και Views 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 Framework 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

Κεφάλαιο 3

Θεωρητικό Υπόβαθρο και Σχεδίαση Αλγορίθμου

3.1 Εισαγωγή στον προγραμματισμό ικανοποίησης περιορισμών

Ο Προγραμματισμός Ικανοποίησης Περιορισμών (Constraint Satisfaction Programming - CSP) αποτελεί ένα ισχυρό και ευρέως χρησιμοποιούμενο υπολογιστικό μοντέλο για την επίλυση προβλημάτων όπου η λύση πρέπει να πληροί ένα σύνολο περιορισμών (constraints). Τυπικά, ένα πρόβλημα CSP ορίζεται από τρία βασικά στοιχεία:

Μεταβλητές (Variables): Το σύνολο των στοιχείων του προβλήματος που πρέπει να αποκτήσουν τιμές. Για παράδειγμα, σε ένα πρόβλημα ανάθεσης μαθημάτων, οι μεταβλητές μπορεί να είναι οι φοιτητές ή τα μαθήματα στα οποία πρέπει να εγγραφούν.

Πεδία Τιμών (Domains): Για κάθε μεταβλητή, το σύνολο των επιτρεπόμενων τιμών που μπορεί να λάβει. Στο παράδειγμα μας, το πεδίο τιμών για κάθε φοιτητή μπορεί να είναι το σύνολο των διαθέσιμων μαθημάτων.

Περιορισμοί (Constraints): Κανόνες που πρέπει να ικανοποιούνται για να θεωρηθεί μια λύση αποδεκτή. Οι περιορισμοί μπορεί να είναι δυαδικοί (μεταξύ δύο μεταβλητών) ή πιο

πολύπλοκοι (μεταξύ πολλών μεταβλητών) και μπορεί να εκφράζουν σχέσεις όπως αμοιβαίο αποκλεισμό, υποχρεωτικές επιλογές, όρια πλήθους, ή προτεραιότητες.

Η διαδικασία επίλυσης ενός CSP στοχεύει στο να ανατεθούν τιμές στις μεταβλητές με τρόπο που να ικανοποιούνται όλοι οι περιορισμοί. Αν δεν υπάρχει λύση που να καλύπτει όλους τους περιορισμούς, μπορεί να αναζητηθεί η καλύτερη δυνατή προσέγγιση μέσω προγραμματισμού βελτιστοποίησης.

Προγραμματισμός Βελτιστοποίησης

Ο Προγραμματισμός Βελτιστοποίησης (Constraint Optimization Programming - COP) αποτελεί επέκταση του κλασικού CSP. Σε ένα πρόβλημα βελτιστοποίησης, εκτός από τους περιορισμούς, υπάρχει και μια συνάρτηση στόχου (objective function) την οποία πρέπει να μεγιστοποιήσουμε ή να ελαχιστοποιήσουμε.

Για παράδειγμα, σε ένα σύστημα ανάθεσης φοιτητών σε μαθήματα, μπορεί να θέλουμε όχι μόνο να ικανοποιήσουμε τις προτιμήσεις των φοιτητών, αλλά και να το κάνουμε με τρόπο που να ελαχιστοποιείται η συνολική δυσαρέσκεια. Έτσι, το πρόβλημα δεν είναι απλά να βρούμε μία λύση, αλλά τη βέλτιστη λύση ως προς ένα κριτήριο.

Ο Προγραμματισμός Βελτιστοποίησης (Constraint Optimization Programming - COP) αποτελεί επέκταση του κλασικού CSP. Σε ένα πρόβλημα βελτιστοποίησης, εκτός από τους περιορισμούς, υπάρχει και μια συνάρτηση στόχου (objective function) την οποία πρέπει να μεγιστοποιήσουμε ή να ελαχιστοποιήσουμε.
Για παράδειγμα, σε ένα σύστημα ανάθεσης φοιτητών σε μαθήματα, μπορεί να θέλουμε όχι μόνο να ικανοποιήσουμε τις προτιμήσεις των φοιτητών, αλλά και να το κάνουμε με τρόπο που να ελαχιστοποιείται η συνολική δυσαρέσκεια. Έτσι, το πρόβλημα δεν είναι απλά να βρούμε μία λύση, αλλά τη βέλτιστη λύση ως προς ένα κριτήριο.

Τα SAT προβλήματα (Satisfiability Problems) αποτελούν ειδική κατηγορία CSP, όπου ζητείται να προσδιοριστεί αν υπάρχει μια ανάθεση τιμών σε μεταβλητές που καθιστά μια λογική φόρμουλα αληθή.

Το πιο βασικό πρόβλημα SAT είναι το κλασικό Boolean SAT Problem, όπου κάθε μεταβλητή μπορεί να πάρει την τιμή TRUE ή FALSE και η λογική έκφραση αποτελείται από συνδυασμούς αυτών μέσω λογικών τελεστών (AND, OR, NOT).

Η σημασία των SAT προβλημάτων είναι τεράστια, επειδή:

Πολλά πρακτικά προβλήματα (προγραμματισμός, επαλήθευση κυκλωμάτων, ανάλυση λογισμικού) μπορούν να αναχθούν σε SAT προβλήματα.

Υπάρχουν εξαιρετικά αποδοτικοί SAT solvers που μπορούν να χειριστούν πολύ μεγάλα προβλήματα σε σύντομο χρόνο.SAT είναι ένα από τα πιο γνωστά NP-πλήρη προβλήματα, άρα η μελέτη του σχετίζεται άμεσα με τη θεωρία υπολογισιμότητας και πολυπλοκότητας.
Σε πρακτικά συστήματα, τεχνικές από τον SAT προγραμματισμό (ή επεκτάσεις του, όπως το SMT - Satisfiability Modulo Theories) συχνά συνδυάζονται με προσεγγίσεις CSP και COP για την επίλυση σύνθετων βιομηχανικών ή ακαδημαϊκών προβλημάτων. [4] [5]

3.2 Αναπαράσταση του προβλήματος μας ως ΠΙΠ

Το πρόβλημα της δίκαιης και αποτελεσματικής ανάθεσης φοιτητών σε μαθήματα περιορισμένης επιλογής στο Τμήμα Πληροφορικής του Πανεπιστημίου Κύπρου, αναπαρίσταται με φυσικό και αποτελεσματικό τρόπο ως ένα Πρόβλημα Ικανοποίησης Περιορισμών (CSP). Μάλιστα, πρόκειται για ένα υβριδικό CSP-SAT πρόβλημα βελτιστοποίησης, καθώς περιλαμβάνει τόσο δυαδικές μεταβλητές με τιμές {0,1}, χαρακτηριστικές των SAT προβλημάτων, όσο και ακέραιες μεταβλητές που σχετίζονται με μέτρα ικανοποίησης και βελτιστοποίησης.

Μεταβλητές και Πεδία Τιμών:

- Κύριες Μεταβλητές:

Οι κύριες μεταβλητές του προβλήματός μας αναπαριστούν την πιθανή εγγραφή κάθε φοιτητή σε κάθε διαθέσιμο μάθημα. Συγκεκριμένα, κάθε μεταβλητή εκφράζει ένα μοναδικό συνδυασμό φοιτητή-μαθήματος, όπως για παράδειγμα η μεταβλητή (Φοιτητής 1 – Μάθημα ΕΠΛ412), η οποία δηλώνει αν ο συγκεκριμένος φοιτητής έχει εγγραφεί ή όχι στο εν λόγω μάθημα. Το πεδίο τιμών αυτών των μεταβλητών είναι δυαδικό: {0,1}, όπου η τιμή 0 σημαίνει ότι ο φοιτητής δεν έχει εγγραφεί στο μάθημα, ενώ η τιμή 1 σημαίνει ότι έχει εγγραφεί. Σε ένα πρόβλημα με N φοιτητές και M μαθήματα, ο συνολικός αριθμός τέτοιων μεταβλητών είναι N × M, αφού εξετάζονται όλοι οι δυνατοί συνδυασμοί εγγραφών.

- Βοηθητικές Μεταβλητές:

Οι βοηθητικές μεταβλητές του προβλήματος χρησιμοποιούνται για την αξιολόγηση και τη βελτιστοποίηση της κατανομής των μαθημάτων. Συγκεκριμένα, ο πίνακας satisfaction

αποδίδει σε κάθε φοιτητή έναν βαθμό ικανοποίησης, ανάλογα με την προτεραιότητα του μαθήματος στο οποίο τελικά εγγράφηκε (100 για 1η επιλογή, 75 για 2η, 50 για 3η, 25 για 4η και 0 για 5η), με πεδίο τιμών από 0 έως 250. Η μεταβλητή minimum satisfaction κρατά τη μικρότερη τιμή του πίνακα satisfaction μεταξύ όλων των φοιτητών, ενώ η μεταβλητή total satisfaction εκφράζει το άθροισμα όλων των τιμών του πίνακα satisfaction, αποτυπώνοντας έτσι το συνολικό επίπεδο ικανοποίησης. Οι μεταβλητές αυτές δεν ανήκουν στο SAT σκέλος του προβλήματος, αλλά ενσωματώνονται για να καθορίσουν τον στόχο της βελτιστοποίησης, εξασφαλίζοντας μια δίκαιη και αποδοτική κατανομή.

Περιορισμοί του Προβλήματος:
Το σύστημα πρέπει να ικανοποιεί τους ακόλουθους περιορισμούς:

1. Αριθμός Μαθημάτων:

   Κάθε φοιτητής πρέπει να εγγραφεί σε ακριβώς τόσα μαθήματα όσα έχει δηλώσει ότι επιθυμεί.
2. Χωρητικότητα Μαθημάτων:

   Ο συνολικός αριθμός φοιτητών που εγγράφονται σε κάθε μάθημα δεν πρέπει να υπερβαίνει τη μέγιστη επιτρεπόμενη χωρητικότητα του μαθήματος.
3. Υπολογισμός Satisfaction:

   Ο πίνακας satisfaction πρέπει να ενημερώνεται σωστά βάσει της θέσης κάθε μαθήματος στη λίστα προτεραιότητας του φοιτητή.
4. Υπολογισμός Minimum Satisfaction:

   Η μεταβλητή minimum satisfaction πρέπει να αντιπροσωπεύει τη μικρότερη τιμή στον πίνακα satisfaction.
5. Συμμόρφωση με Λίστα Προτεραιότητας:

   Οι φοιτητές μπορούν να εγγραφούν μόνο σε μαθήματα που βρίσκονται στη λίστα προτεραιότητας τους (5 επιλογές ανά φοιτητή).
6. Υπολογισμός Total Satisfaction:

   Το άθροισμα της συνολικής ικανοποίησης όλων των φοιτητών πρέπει να υπολογίζεται σωστά.
7. Προτεραιότητα για Διπλωματική Εργασία:

   Ειδικός περιορισμός επιβάλλει την εγγραφή φοιτητή στο μάθημα που έχει επιλέξει ως 1η προτεραιότητα για την εκπόνηση διπλωματικής εργασίας.

Αυτοί οι περιορισμοί διασφαλίζουν ότι κάθε λύση που προκύπτει είναι έγκυρη (feasible).

Αντικειμενοστρεφής Συνάρτηση Βελτιστοποίησης:

Η αντικειμενοστρεφής συνάρτηση του προβλήματος επιδιώκει όχι μόνο να βρει μια έγκυρη λύση αλλά και να μεγιστοποιήσει την συνολική και την ελάχιστη ικανοποίηση. Συγκεκριμένα:

model.Maximize((minSatisfaction×numStudents)+totalSatisfaction)

Αυτός ο συνδυασμός δίνει ίσου βάρους σημασία τόσο στη συνολική ευτυχία όλων των φοιτητών όσο και στη διασφάλιση ότι ακόμα και ο λιγότερο ικανοποιημένος φοιτητής θα είναι όσο το δυνατόν πιο ικανοποιημένος. Αυτό επιτυγχάνεται με το ότι πολλαπλασιάζουμε το minSatisfaction με τον αριθμό των φοιτητών ούτως ώστε ώστε αυτή η ποσότητα να ισοζυγίζεται με το totalSatisfaction με βάση το πεδίο τιμών τους. Η στρατηγική αυτή υποστηρίζει την επιδίωξη για δίκαιη και ισορροπημένη ανάθεση.

Για την πλήρη και ακριβή μοντελοποίηση του προβλήματος απαιτούνται ορισμένες βασικές παράμετροι. Πιο συγκεκριμένα, περιλαμβάνονται: (1) ο συνολικός αριθμός των φοιτητών και (2) ο συνολικός αριθμός των μαθημάτων, (3) μία λίστα με τα ονόματα όλων των διαθέσιμων μαθημάτων, καθώς και (4) ένας πίνακας που καταγράφει τη χωρητικότητα (δηλαδή το μέγιστο αριθμό φοιτητών) για κάθε μάθημα. Επίσης, απαιτείται (5) ένας πίνακας που καθορίζει πόσα μαθήματα επιθυμεί να παρακολουθήσει κάθε φοιτητής, (6) ένας δισδιάστατος πίνακας που περιέχει τις λίστες προτεραιότητας όλων των φοιτητών (με πέντε επιλεγμένα μαθήματα ανά φοιτητή), και τέλος, (7) ένας boolean πίνακας που δείχνει για κάθε φοιτητή εάν το μάθημα πρώτης επιλογής του σχετίζεται με τη διπλωματική του εργασία. Όλες αυτές οι παράμετροι είναι απαραίτητες για να μπορέσει το μοντέλο να αναπαραστήσει με ακρίβεια την πραγματική διαδικασία εγγραφής και να εφαρμόσει τους αντίστοιχους περιορισμούς και στόχους βελτιστοποίησης.

Φύση της Λύσης:

Η επιδιωκόμενη λύση δεν είναι απλώς feasible (δηλαδή μια λύση που ικανοποιεί όλους τους περιορισμούς), αλλά optimal: επιτυγχάνει το μέγιστο δυνατό άθροισμα του συνδυαστικού στόχου που περιλαμβάνει την ελάχιστη και τη συνολική ικανοποίηση.

3.3 Παρουσίαση του εργαλείου Google – OR-Tools

Στο αρχικό στάδιο της ανάπτυξης του συστήματος, ο συνεργάτης μου, Στυλιανός, εργάστηκε με το εργαλείο MiniZinc, ένα δημοφιλές και ευρέως χρησιμοποιούμενο εργαλείο μοντελοποίησης για προβλήματα ικανοποίησης περιορισμών και βελτιστοποίησης. Το MiniZinc προσφέρει μια υψηλού επιπέδου γλώσσα μοντελοποίησης που είναι ανεξάρτητη από συγκεκριμένους επιλυτές, επιτρέποντας την εύκολη μεταφορά μοντέλων σε διάφορα συστήματα επίλυσης. Χρησιμοποιώντας το MiniZinc, έγινε μια πρώτη αναπαράσταση του προβλήματος και δοκιμάστηκαν βασικές στρατηγικές ικανοποίησης των περιορισμών.

Στη συνέχεια, για την πλήρη ανάπτυξη του τελικού μας αλγορίθμου και συστήματος, εργάστηκα πάνω στο εργαλείο Google OR-Tools, το οποίο επέλεξα για την ευελιξία, τη δύναμη και τις δυνατότητές του σε πραγματικά προβλήματα βελτιστοποίησης μεγάλης κλίμακας.

Το Google OR-Tools είναι ένα ανοιχτού κώδικα σύνολο βιβλιοθηκών που αναπτύχθηκε από τη Google, ειδικά σχεδιασμένο για την επίλυση προβλημάτων Operations Research (Έρευνας Επιχειρησιακών Λειτουργιών) και Optimization. Το OR-Tools υποστηρίζει πολλά είδη προβλημάτων, όπως:

- Προβλήματα Ικανοποίησης Περιορισμών (CSP)
- Προβλήματα Γραμμικής και Μη Γραμμικής Βελτιστοποίησης
- Προβλήματα SAT (Boolean Satisfiability)
- Routing & Scheduling Problems (Προβλήματα διαδρομολόγησης και προγραμματισμού)

Το OR-Tools, και ειδικότερα ο ισχυρός επιλύτης CP-SAT Solver, αποτελεί ένα από τα κορυφαία εργαλεία για την επίλυση πολύπλοκων προβλημάτων ικανοποίησης περιορισμών (CSP) και βελτιστοποίησης. Το εργαλείο υποστηρίζει ακέραιες, boolean και συνεχείς μεταβλητές και παρέχει έναν πλούσιο σύνολο έτοιμων περιορισμών όπως άθροισμα, ισότητα, μέγιστο, ελάχιστο και λογικές εκφράσεις, διευκολύνοντας σημαντικά τη μοντελοποίηση σύνθετων προβλημάτων. Επιπλέον, παρέχει προηγμένες δυνατότητες για τη διαχείριση στόχων βελτιστοποίησης, επιτρέποντας την εύκολη μεγιστοποίηση ή ελαχιστοποίηση μίας ή περισσότερων αντικειμενικών συναρτήσεων — υποστηρίζοντας και weighted συνδυασμούς στόχων, όπως στο δικό μας πρόβλημα. Ο CP-SAT Solver αξιοποιεί σύγχρονες τεχνικές όπως SAT-solving, Branch and Bound και Local Search, προσφέροντας υψηλή απόδοση ακόμα και για προβλήματα με χιλιάδες μεταβλητές. Το πιο σημαντικό όμως είναι η ευελιξία του

OR-Tools στη μοντελοποίηση σύνθετων σεναρίων, όπως αυτό της δικής μας εφαρμογής, όπου συνυπάρχουν συνδυαστικοί και λογικοί περιορισμοί εγγραφών, ικανοποίησης και χωρητικότητας.

3.4 Συγκριτική ανάλυση των εργαλείων και των δυνατοτήτων τους

Η συγκριτική ανάλυση ανάμεσα στα εργαλεία MiniZinc και Google OR-Tools αναδεικνύει με σαφήνεια την υπεροχή του δεύτερου, ειδικά όταν πρόκειται για την ανάπτυξη εφαρμογών που απαιτούν ενσωμάτωση με πραγματικά συστήματα και τεχνολογίες παραγωγής. Το MiniZinc είναι ένα ισχυρό και ευέλικτο μοντελιστικό εργαλείο, ιδανικό για ακαδημαϊκή χρήση ή για την αρχική φάση πειραματισμού σε προβλήματα περιορισμών, καθώς επιτρέπει τη γρήγορη περιγραφή ενός μοντέλου χωρίς να ασχολείται κανείς με λεπτομέρειες υλοποίησης. Ωστόσο, περιορίζεται σημαντικά όταν απαιτείται ενσωμάτωση με σύγχρονες γλώσσες προγραμματισμού, διαχείριση εισόδου/εξόδου ή σύνδεση με βάσεις δεδομένων, web interfaces και άλλα modules.

Αντίθετα, το Google OR-Tools αποτελεί ένα σύγχρονο, ανοιχτού κώδικα εργαλείο επίλυσης βελτιστοποιητικών και περιοριστικών προβλημάτων, σχεδιασμένο όχι μόνο για μοντελοποίηση αλλά και για πλήρη ενσωμάτωση σε λογισμικά παραγωγής. Το OR-Tools προσφέρει bindings για πολλές γλώσσες προγραμματισμού, όπως Python, C++, Java και C#. Η υποστήριξη αυτών των γλωσσών το καθιστά εξαιρετικά ευέλικτο, επιτρέποντας στον προγραμματιστή να ενσωματώσει τον επιλύτη σε μεγαλύτερα συστήματα, με πλήρη έλεγχο εισόδου/εξόδου, διαχείρισης εξαιρέσεων, λογικής ροής και οπτικοποίησης. Στο πλαίσιο του δικού μας project, επέλεξα να χρησιμοποιήσω τη γλώσσα C#, λόγω του ότι ανήκει στο οικοσύστημα των τεχνολογιών που αξιοποιήθηκαν για την υλοποίηση της web εφαρμογής μας. Αυτή η επιλογή επέτρεψε την άμεση και αποδοτική ενσωμάτωση του αλγορίθμου κατανομής στο backend του συστήματος, διατηρώντας παράλληλα υψηλή απόδοση και αξιοπιστία.

Επιπλέον, το OR-Tools διαθέτει έναν από τους ταχύτερους και πιο εξελιγμένους επιλύτες προβλημάτων περιορισμών — τον CP-SAT Solver — που προσφέρει βελτιστοποίηση με τεχνικές SAT-solving, Branch and Bound, και Local Search. Αυτές οι τεχνικές είναι ιδιαίτερα αποτελεσματικές για μεγάλης κλίμακας προβλήματα όπως το δικό μας, που περιλαμβάνει χιλιάδες μεταβλητές και συνδυαστικούς περιορισμούς. Τέλος, ενώ το MiniZinc απαιτεί τη χρήση εξωτερικών solvers (όπως Gecode ή Chuffed), το OR-Tools περιλαμβάνει δικούς του

solvers, μειώνοντας την πολυπλοκότητα και τα εξωτερικά dependencies. Συνοψίζοντας, το Google OR-Tools αποτελεί μια πολύ πιο ολοκληρωμένη και πρακτική λύση για υλοποιήσεις παραγωγής, με δυνατότητες που καλύπτουν πλήρως τις ανάγκες μοντελοποίησης, επίλυσης και ενσωμάτωσης σε πραγματικά πληροφοριακά συστήματα.

Κεφάλαιο 4

Σχεδίαση του Συστήματος

4.1 Εισαγωγή στην Σχεδίαση του Συστήματος

Η σχεδίαση του συστήματος βασίστηκε στη φιλοσοφία ότι θέλουμε να προχωρήσουμε με την ιδέα της κατασκευής μίας μοντέρνας web εφαρμογής που να μπορεί να χρησιμοποιηθεί από το Τμήμα Πληροφορικής του Πανεπιστημίου Κύπρου για τη δίκαιη κατανομή μαθημάτων περιορισμένης επιλογής στους φοιτητές σύμφωνα με τις προτιμήσεις τους. Η βασική ιδέα ήταν η δημιουργία ενός εύχρηστου, διαδραστικού και επεκτάσιμου περιβάλλοντος, το οποίο θα επιτρέπει την καταχώρηση δεδομένων, την εφαρμογή του αλγορίθμου κατανομής και την παρουσίαση των αποτελεσμάτων με τρόπο φιλικό προς το χρήστη. Η web εφαρμογή σχεδιάστηκε με χρήση του ASP.NET MVC Framework, αξιοποιώντας το πρότυπο αρχιτεκτονικής Model–View–Controller, το οποίο διαχωρίζει την επιχειρησιακή λογική, την παρουσίαση και τη διαχείριση δεδομένων. Αυτός ο διαχωρισμός επέτρεψε σε εμένα και τον Στυλιανό να εργαστούμε παράλληλα σε διαφορετικά μέρη του συστήματος.[1] [2] [3]

Πιο συγκεκριμένα, η δική μου συνεισφορά εστιάστηκε στον πυρήνα της εφαρμογής, δηλαδή στο σχεδιασμό των Models, των βασικών οντοτήτων (entities) και του σχήματος της βάσης δεδομένων. Το τμήμα αυτό είναι κρίσιμο, καθώς αποτελεί τον θεμέλιο λίθο τόσο για τη λογική του αλγορίθμου, όσο και για την επικοινωνία μεταξύ χρήστη, εφαρμογής και solver. Ο σωστός σχεδιασμός των οντοτήτων διασφαλίζει την ορθή αποθήκευση, ανάκτηση και συσχέτιση των δεδομένων, τα οποία στη συνέχεια τροφοδοτούν τον αλγόριθμο κατανομής.

Τα δεδομένα αυτά περιλαμβάνουν πληροφορίες για φοιτητές, διαθέσιμα μαθήματα, χωρητικότητες μαθημάτων, προτιμήσεις φοιτητών, καθώς και το αποτέλεσμα της κατανομής.

Ο σχεδιασμός ακολουθήθηκε με γνώμονα τη λειτουργικότητα, την επεκτασιμότητα και την υποστήριξη της αλγοριθμικής λογικής, η οποία υλοποιήθηκε μέσω του Google OR-Tools CP-SAT solver. Στόχος ήταν να διαμορφωθεί μία δομή δεδομένων που θα επιτρέπει την εύκολη εξαγωγή όλων των απαραίτητων παραμέτρων για τον αλγόριθμο, όπως ο πίνακας προτεραιοτήτων, οι εγγραφές των φοιτητών και οι ικανοποιήσεις, αλλά και να υποστηρίζεται η επιστροφή και η αποθήκευση των αποτελεσμάτων με τρόπο διαφανή για τον τελικό χρήστη.

Στο πλαίσιο αυτό, σχεδιάστηκαν με προσοχή τα Entities και τα αντίστοιχα Models σε C#, χρησιμοποιώντας τη μεθοδολογία Code First του Entity Framework, γεγονός που προσέφερε την ευελιξία να οριστούν οι πίνακες της βάσης δεδομένων κατευθείαν από τον ορισμό των κλάσεων. Η βάση δεδομένων που προέκυψε αντανακλά με ακρίβεια τη λογική του προβλήματος, επιτρέποντας την αποθήκευση πολλαπλών λιστών προτεραιότητας ανά φοιτητή, την παρακολούθηση των διαθέσιμων θέσεων ανά μάθημα και την καταγραφή των αποτελεσμάτων ικανοποίησης (satisfaction) μετά την εκτέλεση του αλγορίθμου.

Ο ρόλος των models είναι ιδιαίτερα κρίσιμος σε μία MVC εφαρμογή, καθώς γεφυρώνουν το χάσμα μεταξύ των δεδομένων και της λογικής. Κάθε βασική οντότητα – όπως User, Course, Announcement, PriorityList, AssignedList – μοντελοποιήθηκε ώστε να υποστηρίζει τον στόχο του προβλήματος, δηλαδή τη δίκαιη και βέλτιστη κατανομή των μαθημάτων, ενώ δόθηκε ιδιαίτερη προσοχή στην αναπαράσταση των σχέσεων (one-to-many, many-to-many), στην ορθότητα των αναφορών (foreign keys), αλλά και στην οργάνωση του σχήματος με βάση τις αρχές της κανονικοποίησης.

Συμπερασματικά, η παρούσα ενότητα εστιάζει στη δική μου συμβολή στον σχεδιασμό του συστήματος, και ειδικότερα στη δομή των οντοτήτων και της βάσης δεδομένων, οι οποίες είναι άρρηκτα συνδεδεμένες με τον αλγόριθμο βελτιστοποίησης και συνιστούν απαραίτητο στοιχείο για την επιτυχία της συνολικής εφαρμογής.

4.2 Οντότητες και Συσχέτιση

Στον πυρήνα της εφαρμογής βρίσκεται το σχήμα της βάσης δεδομένων, το οποίο έχει σχεδιαστεί προσεκτικά ώστε να καλύπτει όλες τις απαιτήσεις του προβλήματος κατανομής μαθημάτων και να εξυπηρετεί τόσο τις ανάγκες των φοιτητών όσο και της διοίκησης. Ακολουθεί αναλυτική περιγραφή των βασικών οντοτήτων που υλοποιήθηκαν στο πλαίσιο του μοντέλου:

Users

Η οντότητα Users αποτελεί τον ακρογωνιαίο λίθο του συστήματος, καθώς κάθε εγγραφή αντιστοιχεί σε έναν χρήστη της εφαρμογής, είτε πρόκειται για φοιτητή είτε για διαχειριστή. Περιλαμβάνει τα ακόλουθα πεδία:

- UserId: Πρωτεύον κλειδί, μοναδικό αναγνωριστικό χρήστη.
- FirstName, LastName: Προσωπικά στοιχεία.

- Email, Password: Πιστοποιητικά πρόσβασης.
- NumberOfCourses: Ο αριθμός μαθημάτων στα οποία επιθυμεί να εγγραφεί ο φοιτητής.
- Role: Ο ρόλος του χρήστη (π.χ., “Student” ή “Secretary”).

Η διαφοροποίηση των ρόλων επιτρέπει την υλοποίηση δικαιωμάτων πρόσβασης και λειτουργιών ανάλογα με την ταυτότητα του χρήστη.

Courses

Η οντότητα Courses αποθηκεύει τα διαθέσιμα μαθήματα προς επιλογή και περιλαμβάνει κρίσιμες πληροφορίες που είναι απαραίτητες για τον αλγόριθμο εγγραφής:

- Code: Πρωτεύον κλειδί, μοναδικός κωδικός μαθήματος.
- Title: Τίτλος μαθήματος.
- Instructor: Υπεύθυνος διδασκαλίας.
- Capacity: Μέγιστος αριθμός φοιτητών που μπορεί να εγγραφεί.
- DaysHours: Ώρες και ημέρες διεξαγωγής.
- Prerequisites: Προαπαιτούμενα μαθήματα.

- Description: Περιγραφή του μαθήματος.

Η πληροφορία της χωρητικότητας (Capacity) αξιοποιείται από τον αλγόριθμο κατανομής ως περιορισμός εγγραφής.

PriorityLists

Η οντότητα PriorityLists αναπαριστά τις προτιμήσεις των φοιτητών και σχετίζεται άμεσα με τον βασικό μηχανισμό βελτιστοποίησης:

- ListId: Πρωτεύον κλειδί.
- FirstChoice έως FifthChoice: Οι πέντε κορυφαίες επιλογές μαθημάτων του φοιτητή, με φθίνουσα σειρά προτεραιότητας.
- IsFirstChoiceForBachelorThesis: Boolean τιμή που δηλώνει αν το μάθημα 1ης προτεραιότητας σχετίζεται με τη διπλωματική εργασία.
- TitleBachelorThesis: Ο τίτλος της διπλωματικής (αν υπάρχει).
- UserId: Ξένο κλειδί προς την οντότητα Users.

Αυτή η οντότητα είναι ουσιώδης για την εκτέλεση του αλγορίθμου, καθώς από εδώ αντλούνται οι πίνακες προτιμήσεων που τροφοδοτούν τον solver.

AssignedLists

Η οντότητα AssignedLists καταγράφει το αποτέλεσμα της εκτέλεσης του αλγορίθμου και τις τελικές αναθέσεις μαθημάτων σε κάθε φοιτητή:

- AssignedListId: Πρωτεύον κλειδί.
- UserId: Αναφορά στον φοιτητή.
- Courses: Συγκεντρωτική λίστα με τα μαθήματα στα οποία τελικά έχει εγγραφεί ο φοιτητής.

Η συγκεκριμένη οντότητα εξυπηρετεί την αποθήκευση του αποτελέσματος και τη μεταγενέστερη προβολή ή εξαγωγή.

Announcements

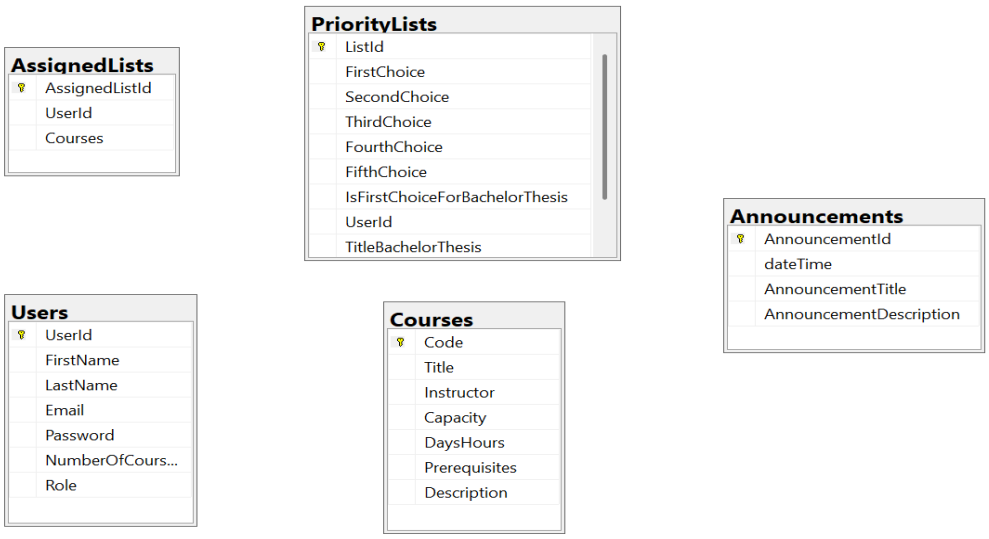
Η οντότητα Announcements υποστηρίζει τη δημοσίευση ενημερώσεων και ανακοινώσεων μέσα στην πλατφόρμα από την γραμματέα του τμήματος:

- AnnouncementId: Πρωτεύον κλειδί.
- dateTime: Ημερομηνία και ώρα ανάρτησης.
- AnnouncementTitle: Τίτλος ανακοίνωσης.
- AnnouncementDescription: Το περιεχόμενο της ανακοίνωσης.

Η λειτουργικότητα αυτή είναι ιδιαίτερα χρήσιμη για ενημερώσεις σχετικά με περιόδους εγγραφών, αποτελέσματα, οδηγίες, κ.λπ.

Το παραπάνω σχήμα συνθέτει ένα ολοκληρωμένο μοντέλο που υποστηρίζει πλήρως τις απαιτήσεις του συστήματος. Οι οντότητες συνδέονται μέσω σχέσεων τύπου one-to-many (π.χ., κάθε φοιτητής μπορεί να έχει μόνο μία λίστα προτεραιότητας, αλλά πολλές εγγραφές), ενώ η οργάνωση των δεδομένων υποστηρίζει την αποδοτική εξαγωγή παραμέτρων και την εκτέλεση του αλγορίθμου μέσω των Google OR-Tools.

Πιο κάτω παρατίθεται και το Σχήμα - Διάγραμμα της Βάσης Δεδομένων (Schema - Diagram) που έχει εξαχθεί από το στούντιο διαχείρισης σχέσεων της Microsoft που χρησιμοποιήσαμε στην υλοποίηση, διακρίνοντας τα σχετικά αντικείμενα , τα πεδία και τις σχέσεις που υπάρχουν μεταξύ των αντικειμένων :



Εικόνα 3 – Σχήμα Βάσης Δεδομένων

4.3 Σχεδίαση της Βάσης Δεδομένων

Η βάση δεδομένων του συστήματος διαδραματίζει καθοριστικό ρόλο στην οργάνωση, αποθήκευση και ανάκτηση όλων των πληροφοριών που σχετίζονται με τους φοιτητές, τα μαθήματα, τις επιλογές προτεραιότητας, τα αποτελέσματα κατανομής και τις ανακοινώσεις. Ο σχεδιασμός της υλοποιήθηκε με γνώμονα τη σαφήνεια, την αποδοτικότητα και την επεκτασιμότητα.

Η βάση σχεδιάστηκε με βάση το σχεσιακό μοντέλο και υλοποιήθηκε μέσω του Microsoft SQL Server, ακολουθώντας την αρχή της κανονικοποίησης ώστε να διασφαλιστεί η ακεραιότητα και να αποτραπεί η πλεοναστική αποθήκευση δεδομένων. Η επικοινωνία με την εφαρμογή πραγματοποιήθηκε μέσω του Entity Framework σε προσέγγιση Code-First, που επέτρεψε τον απευθείας καθορισμό των πινάκων και σχέσεων μέσω C# κλάσεων (Models).

Η σχεδίαση περιλαμβάνει τις εξής βασικές οντότητες:

- **Users**: Η οντότητα Users αποτελεί τη θεμελιώδη βάση για την αναπαράσταση όλων των χρηστών που αλληλεπιδρούν με την πλατφόρμα. Στο σύστημα υπάρχουν δύο κύριοι ρόλοι: φοιτητές και διαχειριστές (admins). Οι φοιτητές χρησιμοποιούν την εφαρμογή για να δηλώσουν τα μαθήματα στα οποία επιθυμούν να εγγραφούν σε σειρά προτεραιότητας και να βλέπουν τα μαθήματα στα οποία έχουν εγγραφεί εν τέλει, ενώ οι διαχειριστές έχουν τη δυνατότητα να προσθέτουν/τροποποιούν μαθήματα, να βλέπουν τα αποτελέσματα των εγγραφών, να δημοσιεύουν ανακοινώσεις και να ενεργοποιούν τον αλγόριθμο κατανομής.
  Συγκεκριμένα, ο πίνακας Users, περιέχει τα εξής πεδία: το πεδίο Role που καθορίζει αν πρόκειται για φοιτητή ή διαχειριστή. Το NumberOfCourses εκφράζει πόσα μαθήματα επιθυμεί να εγγραφεί ο φοιτητής, κάτι που λαμβάνεται υπόψη στον αλγόριθμο. Το UserId είναι μοναδικό και χρησιμοποιείται για να συνδέεται ο φοιτητής με άλλες οντότητες, όπως οι επιλογές προτίμησης ή τα ανατεθειμένα μαθήματα. Τέλος, τα πεδία FirstName, LastName, Email, Password χρησιμοποιούνται για τις προσωπικές πληροφορίες του φοιτητή.
- **Courses**: Η οντότητα Courses αντιπροσωπεύει όλα τα μαθήματα που προσφέρονται για δήλωση από τους φοιτητές μέσα από την πλατφόρμα. Είναι ένα από τα κεντρικότερα στοιχεία του συστήματος, καθώς γύρω από αυτό περιστρέφονται οι

επιλογές των φοιτητών και η λειτουργία του αλγορίθμου. Ο κάθε εγγεγραμμένος φοιτητής έχει τη δυνατότητα να δηλώσει προτιμήσεις για συγκεκριμένα μαθήματα με βάση προσωπικά και ακαδημαϊκά κριτήρια.

Ο κάθε εγγραφή στο Courses περιλαμβάνει πλήρη πληροφόρηση για το εκάστοτε μάθημα, όπως ο τίτλος, ο διδάσκων, η μέγιστη χωρητικότητα (Capacity), οι ημέρες και ώρες διεξαγωγής, οι προαπαιτούμενοι και η περιγραφή του. Αυτή η πληροφορία δεν είναι μόνο χρήσιμη για την παρουσίαση στο UI, αλλά χρησιμοποιείται και στον αλγόριθμο για να διασφαλιστούν περιορισμοί όπως η χωρητικότητα ή οι συγκρούσεις προαπαιτουμένων.

- **PriorityLists**: Η οντότητα PriorityLists καταγράφει τις δηλώσεις προτίμησης των φοιτητών. Ο κάθε φοιτητής καλείται να επιλέξει έως και πέντε μαθήματα με σειρά προτεραιότητας (FirstChoice έως FifthChoice), αντικατοπτρίζοντας την επιθυμία του για κάθε μάθημα. Αυτό το σύνολο δεδομένων αποτελεί τη βάση για την εκτέλεση του αλγορίθμου κατανομής, καθώς μέσω αυτού το σύστημα προσπαθεί να ικανοποιήσει όσο το δυνατόν περισσότερες από τις πρώτες επιλογές των φοιτητών, σε συνδυασμό με τους περιορισμούς.Επιπλέον, υπάρχουν πεδία που σχετίζονται με πιθανή ανάθεση διπλωματικής εργασίας, όπως το IsFirstChoiceForBachelorThesis και το TitleBachelorThesis, που επιτρέπουν την εξειδικευμένη διαχείριση φοιτητών που εντάσσουν κάποιο μάθημα ως υποστήριξη της πτυχιακής τους. Η σύνδεση της λίστας προτιμήσεων με τον UserId διασφαλίζει ότι κάθε εγγραφή σχετίζεται μοναδικά με έναν φοιτητή.
- **AssignedLists**: Η οντότητα AssignedLists είναι το τελικό αποτέλεσμα της διαδικασίας κατανομής. Εφόσον ο αλγόριθμος OR-Tools επεξεργαστεί τα δεδομένα των φοιτητών, των μαθημάτων και των περιορισμών, επιστρέφει μια λίστα με τα μαθήματα στα οποία τελικά εγγράφεται ο κάθε φοιτητής. Αυτή η λίστα καταγράφεται στον πίνακα AssignedLists.

  Ουσιαστικά, πρόκειται για τη «λύση» του προβλήματος βελτιστοποίησης. Η ύπαρξή της επιτρέπει στον φοιτητή να ενημερωθεί για την τελική του κατάσταση, και στον διαχειριστή να έχει εικόνα της κατανομής. Αν και ο τρόπος αποθήκευσης είναι σε μορφή συμβολοσειράς (string), ενσωματώνει πλήρως την απόδοση του αλγορίθμου στον πραγματικό κόσμο.
- **Announcements**: Η οντότητα Announcements έχει υποστηρικτικό αλλά κρίσιμο ρόλο στην επικοινωνία του διαχειριστή με τους χρήστες του συστήματος. Κάθε εγγραφή στον πίνακα αυτό περιλαμβάνει έναν τίτλο, μια περιγραφή και μια χρονική

σήμανση (dateTime) και αφορά την ενημέρωση των φοιτητών για κρίσιμες πληροφορίες — όπως π.χ. την ημερομηνία λήξης δηλώσεων, τη δημοσίευση των αποτελεσμάτων, τεχνικές βελτιώσεις, οδηγίες, ή άλλες σχετικές ανακοινώσεις.
Αποτελεί ένα βασικό στοιχείο για τη διατήρηση της διαφάνειας και την εύρυθμη λειτουργία της διαδικασίας, διατηρώντας ανοικτό δίαυλο επικοινωνίας μέσα στην εφαρμογή.

Η βάση δεν περιέχει σύνθετες σχέσεις τύπου many-to-many, γεγονός που διατηρεί τη σχεδίαση απλή και προσανατολισμένη στις ανάγκες της εφαρμογής. Όπου απαιτείται η αποθήκευση πολλών τιμών σε ένα πεδίο (όπως τα μαθήματα στις επιλογές ή στην τελική κατανομή), γίνεται χρήση αναφορών σε Course Code.Η συγκεκριμένη αρχιτεκτονική επιτρέπει την εύκολη εξαγωγή των δεδομένων σε μορφές που είναι συμβατές με τον επιλυτή Google OR-Tools. [6] [7] [8]

4.4 Αλληλεπίδραση Βάσης Δεδομένων με τον Αλγόριθμο ΠΠ

Η βάση δεδομένων διαδραματίζει κεντρικό ρόλο στην προετοιμασία και υποστήριξη του αλγορίθμου επίλυσης προβλήματος ικανοποίησης περιορισμών (CSP-SAT) που χρησιμοποιείται για την αντιστοίχιση φοιτητών σε μαθήματα. Το πρόβλημα μας είναι υβριδικής μορφής, καθώς περιέχει μεταβλητές με δυαδικά πεδία τιμών (SAT - {0,1}) αλλά και ακέραιες μεταβλητές που σχετίζονται με επίπεδα ικανοποίησης, συνθέτοντας ένα πρόβλημα βελτιστοποίησης. Η βάση δεδομένων παρέχει τα απαραίτητα δεδομένα για την κατασκευή του μοντέλου του αλγορίθμου και παραλαμβάνει τα αποτελέσματα μετά την επίλυσή του.

Μεταβλητές και Πεδία Τιμών από τη Βάση Δεδομένων:
→Οι κύριες και βοηθητικές μεταβλητές του αλγορίθμου αντλούνται από συγκεκριμένα πεδία των πινάκων της βάσης δεδομένων:

- **Users.UserId** και **Courses.CourseId**: Αυτά τα δύο πεδία χρησιμοποιούνται για τον ορισμό των δυαδικών μεταβλητών απόφασης ως τα πεδία που κάνουν μοναδικό ένα χρήστη και ένα μάθημα. Κάθε τέτοια μεταβλητή εκφράζει το αν ο φοιτητής UserId έχει εγγραφεί στο μάθημα CourseId (τιμή 1) ή όχι (τιμή 0).

- **Users.NumberOfCourses**: Χρησιμοποιείται για την επιβολή του περιορισμού ότι κάθε φοιτητής πρέπει να εγγραφεί ακριβώς στον αριθμό μαθημάτων που έχει δηλώσει.
- **Courses.Capacity**: Αποτελεί το όριο που χρησιμοποιείται περιορισμό ότι το κάθε μάθημα μπορεί να δεχτεί έως έναν συγκεκριμένο αριθμό φοιτητών με βάση την χωρητικότητα που είναι δηλωμένη σε αυτό εξ αρχής.
- **PriorityLists.FirstChoice** έως **FifthChoice**: Τα πεδία αυτά συνιστούν τις πέντε προτιμήσεις κάθε φοιτητή και χρησιμοποιούνται για να περιορίσουν τις μεταβλητές ώστε να δηλώνουν μόνο μαθήματα που βρίσκονται στη Λίστα Προτεραιότητας του κάθε χρήστη, αλλά και να υπολογίζουν τον βαθμό ικανοποίησης του κάθε φοιτητή κατά την διάρκεια εκτέλεσης του αλγορίθμου .
- **PriorityLists.IsFirstChoiceForBachelorThesis**: Το πεδίο αυτό ενεργοποιεί τον υποχρεωτικό περιορισμό ο οποίος διασφαλίζει ότι ο φοιτητής θα εγγραφεί στη συγκεκριμένη πρώτη επιλογή, εφόσον την έχει δηλώσει ως σχετική με τη διπλωματική εργασία του και έχει καταχωρήσει στο σύστημα τον τίτλο της Διπλωματικής του εργασίας , αλλά και το ονοματεπώνυμο του επιβλέπων καθηγητή.

Ο πίνακας satisfaction κατασκευάζεται δυναμικά κατά την εκτέλεση του αλγορίθμου αντιστοίχισης, με σκοπό να μετρήσει την ικανοποίηση κάθε φοιτητή ανάλογα με το πόσο κοντά βρίσκονται οι τελικές του εγγραφές στις αρχικές του προτιμήσεις. Ο πίνακας αυτός δεν προέρχεται άμεσα από κάποιον πίνακα της βάσης, αλλά υπολογίζεται μέσω της συσχέτισης των εγγραφών από τον πίνακα PriorityLists (FirstChoice έως FifthChoice) και των μεταβλητών εγγραφής (που συνδέουν UserId και CourseId). Αν, για παράδειγμα, ο φοιτητής εγγραφεί στο μάθημα που έχει δηλώσει ως FirstChoice, λαμβάνει 100 μονάδες ικανοποίησης, 50 για SecondChoice, 25 για ThirdChoice, 12 για FourthChoice και 0 για FifthChoice. Οι τιμές αυτές συγκροτούν τη βάση για τον υπολογισμό δύο βοηθητικών μεταβλητών: η μεταβλητή minimum_satisfaction, που κρατά την ελάχιστη τιμή από τον πίνακα satisfaction μεταξύ όλων των φοιτητών , και η total_satisfaction, η οποία προκύπτει ως άθροισμα των επιμέρους τιμών satisfaction και αποτελεί το βασικό κριτήριο βελτιστοποίησης. Στόχος του συστήματος είναι η μεγιστοποίηση της total_satisfaction, και του minimum_satisfaction με στόχο μια δίκαιη και ισόρροπη κατανομή μαθημάτων που να ικανοποιεί όσο το δυνατότερο όλους τους φοιτητές. Η διαδικασία ξεκινά με την ανάκτηση των απαραίτητων δεδομένων από τη βάση: όλα τα ενεργά Users, οι εγγραφές προτιμήσεων από τον πίνακα PriorityLists και οι διαθέσιμες πληροφορίες μαθημάτων από τον πίνακα Courses, όπως χωρητικότητα και

προαπαιτούμενα. Στη φάση της μοντελοποίησης, δημιουργούνται οι δυαδικές μεταβλητές που δηλώνουν την πιθανή εγγραφή ενός φοιτητή σε ένα μάθημα (π.χ. x_3_ΕΠΛ412 = 1 σημαίνει ότι ο φοιτητής 3 εγγράφηκε στο μάθημα ΕΠΛ412), ορίζονται όλοι οι απαραίτητοι περιορισμοί και διαμορφώνεται η συνάρτηση στόχου. Στη συνέχεια, η επίλυση γίνεται μέσω του εργαλείου Google OR-Tools, το οποίο επιστρέφει τις τελικές τιμές των μεταβλητών. Τέλος, τα αποτελέσματα αποθηκεύονται στον πίνακα AssignedLists, με τη μορφή μιας λίστας μαθημάτων ανά φοιτητή, συσχετισμένα με το UserId. Η συνεργασία αυτή ανάμεσα στη σχεδίαση της βάσης δεδομένων και το μαθηματικό μοντέλο βελτιστοποίησης καθιστά δυνατή την αυτόματη, δίκαιη και αποδοτική αντιστοίχιση φοιτητών σε μαθήματα, διαχειριζόμενη πολλαπλούς περιορισμούς και στόχους με μαθηματική ακρίβεια.

Κεφάλαιο 5

Υλοποίηση Βάσης Δεδομένων και Αλγόριθμου στο Σύστημα

5.1 Εισαγωγή στην Υλοποίηση

Η φάση της υλοποίησης αποτελεί κρίσιμο στάδιο στην ανάπτυξη του προτεινόμενου συστήματος, καθώς μεταφράζει τη θεωρητική σχεδίαση και τις απαιτήσεις σε ένα πλήρως λειτουργικό λογισμικό. Στην παρούσα εργασία, η υλοποίηση βασίστηκε σε σύγχρονες και αξιόπιστες τεχνολογίες που επιτρέπουν την αποδοτική διαχείριση δεδομένων, τη σταθερή λειτουργία του συστήματος και την ακριβή εκτέλεση του αλγορίθμου κατανομής μαθημάτων με βάση την ικανοποίηση περιορισμών.

Η ανάπτυξη της web εφαρμογής πραγματοποιήθηκε με τη χρήση του ASP.NET Core MVC, ενός ισχυρού framework της Microsoft που συνδυάζει τη δομή Model–View–Controller για την καθαρή οργάνωση του κώδικα και την ευκολία στην επεκτασιμότητα του λογισμικού. Η επιλογή του ASP.NET Core έγινε λόγω της υψηλής απόδοσής του, της διαλειτουργικότητάς του με άλλα εργαλεία της Microsoft και της ευκολίας ενσωμάτωσης με συστήματα βάσεων δεδομένων.

Για τη διαχείριση και μοντελοποίηση της βάσης δεδομένων χρησιμοποιήθηκε το Microsoft SQL Server, ενώ για την αλληλεπίδραση με αυτήν επιλέχθηκε το Entity Framework Core ως Object-Relational Mapper (ORM). Το EF Core επέτρεψε την υλοποίηση των οντοτήτων του συστήματος ως C# κλάσεις (models) και την αυτόματη δημιουργία του αντίστοιχου σχήματος βάσης μέσω της χρήσης migrations. Με τον τρόπο αυτό μειώθηκε η ανάγκη για

χειροκίνητη παρέμβαση στη βάση, ενώ παράλληλα διασφαλίστηκε η συνέπεια μεταξύ της εφαρμογής και των δεδομένων.

Κομβικής σημασίας στοιχείο της υλοποίησης αποτέλεσε η ενσωμάτωση του αλγορίθμου επίλυσης Προβλήματος Ικανοποίησης Περιορισμών (CSP), ο οποίος αναπτύχθηκε με τη χρήση του εργαλείου Google OR-Tools. Ο αλγόριθμος, όπως αναλύθηκε σε προηγούμενα κεφάλαια, μοντελοποιεί την αντιστοίχιση φοιτητών–μαθημάτων ως ένα βελτιστοποιημένο CSP–SAT πρόβλημα, με στόχο τη μέγιστη δυνατή ικανοποίηση των φοιτητών, λαμβάνοντας υπόψη περιορισμούς όπως η χωρητικότητα των μαθημάτων και οι ατομικές προτιμήσεις.

Ένα από τα σημαντικά πλεονεκτήματα της αρχιτεκτονικής που υλοποιήθηκε είναι ότι ο αλγόριθμος λαμβάνει δυναμικά τα δεδομένα του από τη βάση, δηλαδή τις προτιμήσεις των φοιτητών, τις πληροφορίες των μαθημάτων και τα όρια εγγραφής. Με την ολοκλήρωση της εκτέλεσης, τα αποτελέσματα της αντιστοίχισης αποθηκεύονται επίσης στη βάση δεδομένων, εξασφαλίζοντας έτσι τη συνέπεια και την καταγραφή των τελικών επιλογών.

Η σύνδεση όλων αυτών των στοιχείων (βάση – εφαρμογή – αλγόριθμος) έγινε με τρόπο διαφανή για τον χρήστη, επιτρέποντας την αυτοματοποιημένη και δίκαιη διαδικασία εγγραφών μέσω μιας μοντέρνας και επεκτάσιμης αρχιτεκτονικής.

5.2 Υλοποίηση της Βάσης Δεδομένων

Η υλοποίηση της βάσης δεδομένων πραγματοποιήθηκε χρησιμοποιώντας τον Microsoft SQL Server σε συνδυασμό με το Entity Framework Core (EF Core), ένα ORM εργαλείο που επέτρεψε την αντιστοίχιση των αντικειμένων της εφαρμογής (C# κλάσεις) με πίνακες στη σχεσιακή βάση. Μέσω του μηχανισμού των migrations, το σχήμα δημιουργήθηκε αυτόματα βάσει των C# μοντέλων, διασφαλίζοντας συνέπεια και ευκολία στη διαχείριση της βάσης.

Κάθε βασική οντότητα της εφαρμογής (π.χ. Users, Courses, PriorityLists) υλοποιήθηκε ως μοντέλο με τις απαραίτητες αναγωγές σε πρωτεύοντα και ξένα κλειδιά. Οι συσχετίσεις που υπάρχουν μεταξύ των διάφορων αντικειμένων – μοντέλων υλοποιούνται με τον πάρακατω τρόπο:

Η αντιστοίχιση σχέσεων EF Core έχει να κάνει με την αντιστοίχιση της αναπαράστασης πρωτεύοντος κλειδιού/ξένου κλειδιού που χρησιμοποιείται σε μια σχεσιακή βάση δεδομένων

στις αναφορές μεταξύ αντικειμένων που χρησιμοποιούνται σε ένα μοντέλο αντικειμένου , όπως η προσθήκη ιδιότητας πρωτεύοντος κλειδιού σε κάθε τύπο οντότητας, η προσθήκη ιδιότητας ξένου κλειδιού σε έναν τύπο οντότητας και συσχέτιση των αναφορών μεταξύ των τύπων οντοτήτων με το πρωτεύον και το ξένο κλειδί για να σχηματιστεί μια ενιαία διαμόρφωση σχέσης. Μόλις γίνει αυτή η αντιστοίχιση, στη περίπτωση που αλλάζει η τιμή στο πρωτεύον κλειδί μίας οντότητας , τότε το EF αλλάζει αυτόματα τη τιμή και στο αντίστοιχο ξένο κλειδί.

Η υλοποίηση της βάσης δεδομένων πραγματοποιήθηκε με χρήση του Microsoft SQL Server και του Entity Framework Core, αξιοποιώντας το μοντέλο code-first και τα migrations για τη δημιουργία του σχήματος, δηλαδή ότι ολόκληρο το σχήμα , οι οντότητες , τα πεδία , τα κλειδία , οι περιορισμοί και οι συσχετίσεις δηλώθηκαν στον κώδικα ως C# κλάσεις και χρησιμοποιώντας τις εντολές migrations του EF αυτόματα οι αλλαγές στην Βάση Δεδομένων γίνονταν και στον Microsoft SQL Server. Οι βασικές οντότητες του συστήματος, όπως οι `Users`, `Courses`, `PriorityLists`, `AssignedLists` και `Announcements`, υλοποιήθηκαν με έμφαση στην ακεραιότητα δεδομένων και τη λειτουργική συνέπεια με τον αλγόριθμο κατανομής , αλλά και στο τι χρειαζόταν το σύστημα ούτως ώστε να το σύστημα να τηρεί τις αρχικές απαιτήσεις και προδιαγραφές. Η οντότητα `Users` περιλαμβάνει στοιχεία ελέγχου πρόσβασης και καθορισμού ρόλου, επιτρέποντας την παραμετροποίηση των λειτουργιών ανά χρήστη. Ο πίνακας `Courses` αποθηκεύει όλα τα δεδομένα που χρειάζονται για να μπορεί να αναπαρασταθεί ένα μάθημα περιορισμένης επιλογής όπως `Capacity` και `Prerequisites`. Η `PriorityLists` συνδέεται με τους φοιτητές και περιλαμβάνει τις πέντε κορυφαίες επιλογές μαθημάτων τους, με κάποια πεδία να έχουν και validation μέσω data annotations. Η `AssignedLists` καταγράφει τις τελικές αναθέσεις, αποθηκεύοντας τα μαθήματα ως ενιαίο πεδίο με τύπο δεδομένων λίστα από συμβολοσειρές για λόγους ευελιξίας. Οι `Announcements` υλοποιούνται για τη διαχείριση ενημερώσεων από τη γραμματεία και θα μπορούσαμε να πούμε ότι είναι η μοναδική οντότητα του συστήματος η οποία δεν έχει ουδεμία σχέση με την εκτέλεση του αλγορίθμου και απλώς αποθηκεύει όλα τα απαραίτητα δεδομένα σχετικά με τις ανακοινώσεις που θα δημοσιεύει η γραμματέας. Συνολικά, η βάση δεδομένων συνδέεται άρρηκτα με το υπόλοιπο σύστημα ASP.NET MVC Core, υποστηρίζοντας πλήρως τη λογική λειτουργία και την επικοινωνία με τον αλγόριθμο επίλυσης.

5.3 Entity Framework και Ανάκτηση/Αποθήκευση Δεδομένων

Ένα ORM (Object-Relational Mapping) είναι ένα εργαλείο ή τεχνική που επιτρέπει στους προγραμματιστές να αλληλεπιδρούν με σχεσιακές βάσεις δεδομένων μέσω αντικειμενοστραφούς κώδικα. Με άλλα λόγια, το ORM γεφυρώνει το χάσμα μεταξύ του αντικειμενοστραφούς μοντέλου του λογισμικού (π.χ., κλάσεις C#) και του σχεσιακού μοντέλου της βάσης δεδομένων (π.χ., πίνακες SQL). Το πλεονέκτημα αυτής της προσέγγισης είναι ότι ο προγραμματιστής δεν χρειάζεται να γράφει χειροκίνητα SQL εντολές για βασικές λειτουργίες (όπως SELECT, INSERT, UPDATE, DELETE), καθώς το ORM αναλαμβάνει αυτόματα τη μετατροπή των αντικειμένων σε σχεσιακά δεδομένα και αντίστροφα. Έτσι, μειώνεται σημαντικά η πολυπλοκότητα, ενώ αυξάνεται η αναγνωσιμότητα και η συντηρησιμότητα του κώδικα.

Στο πλαίσιο του έργου, χρησιμοποιήθηκε το Entity Framework Core (EF Core), το οποίο αποτελεί το επίσημο ORM framework της Microsoft για την πλατφόρμα .NET. Το EF Core υποστηρίζει την προσέγγιση code-first, επιτρέποντας τον ορισμό των μοντέλων (classes) στον κώδικα και την αυτόματη δημιουργία του αντίστοιχου σχήματος της βάσης δεδομένων μέσω migrations. Η επιλογή του EF Core έγινε λόγω της στενής ενσωμάτωσής του με το ASP.NET MVC Core, της υποστήριξης LINQ (για εκφραστικά queries), καθώς και της δυνατότητας διαχείρισης των σχέσεων, constraints και foreign keys με αντικειμενοστραφή τρόπο.

Μέσω του EF Core υλοποιούνται όλες οι βασικές λειτουργίες ανάκτησης και αποθήκευσης δεδομένων. Για παράδειγμα, με μία απλή DbContext κλάση, είναι δυνατή η πρόσβαση στους πίνακες ως collections (π.χ., DbSet<User>), ενώ οι κλήσεις προς τη βάση δεδομένων υλοποιούνται με κώδικα όπως context.Users.Where(u => u.Role == "Student").ToList() αποθηκεύοντας επιστρεφόμενο αποτέλεσμα σε μία λίστα. Η προσέγγιση αυτή μειώνει τον κίνδυνο SQL injection, ενισχύει την ασφάλεια και κάνει τον κώδικα πιο ευανάγνωστο και συντηρήσιμο. Επιπλέον, το EF Core επιτρέπει τη χρήση των λεγόμενων Data Annotations ([Required], [MaxLength], [ForeignKey],[PrimaryKey] κ.λπ.) για οριοθέτηση για το τι αντιπροσωπεύει το κάθε πεδίο, τυχόν περιορισμούς που θέλουμε να θέσουμε στις δυνατές τιμές που μπορούν αποθηκευτούν σε ένα πεδίο, έλεγχο εγκυρότητας και ενίσχυση της ακεραιότητας των δεδομένων σε επίπεδο μοντέλου.

Συνολικά, το Entity Framework Core αποτέλεσε το βασικό εργαλείο για την ασφαλή, αξιόπιστη και αποδοτική επικοινωνία του συστήματος με τη βάση δεδομένων, ενώ υποστήριξε πλήρως την αρχιτεκτονική της εφαρμογής βάσει του MVC προτύπου.

5.4 Υλοποίηση και Ενσωμάτωση του Αλγορίθμου μέσω OR-Tools

Καταρχάς, τα Google OR-Tools αποτελούν ένα πανίσχυρο, ανοιχτού κώδικα εργαλείο που αναπτύχθηκε από την Google για την επίλυση προβλημάτων ικανοποιήσης περιορισμών και βελτισποποίησης σε υψηλού επιπέδου εφαρμογές αφού υποστηρίζεται από object-oriented γλώσσες προγραμματισμού όπως είναι η C#,C++,Java και Python. Η βιβλιοθήκη αυτή υποστηρίζει διάφορους τύπους προβλημάτων, όπως Constraint Programming (CSP), Linear Programming (LP), Mixed Integer Programming (MIP), Scheduling, Vehicle Routing και άλλα. Στην παρούσα εργασία γίνεται χρήση του CP-SAT Solver, ενός υπερσύγχρονου και αποδοτικού επιλυτή για Constraint Satisfaction προβλήματα. Τα CSPs είναι κατάλληλα για περιπτώσεις όπου πρέπει να ανατεθούν τιμές σε μεταβλητές, υπό συγκεκριμένους περιορισμούς, ώστε να ικανοποιούνται όλοι οι όροι ταυτόχρονα – όπως ακριβώς απαιτείται και στην παρούσα περίπτωση κατανομής φοιτητών σε μαθήματα.
Για την υλοποίηση του δικού μας προβλήματος , όπως περιγράψαμε και προηγουμένως, αναπαραστήσαμε ένα υβριδικό CSP-SAT πρόβλημα βελτιστοποίησης και ικανοποίησης περιορισμών, αφού εμπεριέχει όχι μόνο μεταβλήτες που το πεδίο τιμών τους είναι {0,1} , αλλά και μεταβλητές που έχουν ως πεδία τιμών τους αλλά υποσύνολα των ακεραίων εκτός από 0 ή 1 και γι'αυτό το πρόβλημα μας δεν μπορεί να θεωρηθεί ως ένα απλό CSP-SAT πρόβλημα.

Η ενσωμάτωση αυτής της λογικής στην εφαρμογή ASP.NET MVC Core γίνεται μέσω μιας service class με την ονομασία ConstraintsSatisfactionProblem. Οι service classes γενικά αποτελούν ένα σχεδιαστικό μοτίβο (design pattern) στο ASP.NET MVC που επιτρέπει την απομόνωση της επιχειρηματικής λογικής(business logic) από τους controllers που αποτελούν τον συνδετικό κρίκο μεταξύ της Βάσης Δεδομένων και της οθόνης(User Interface) που αλληλεπιδρά ο χρήστης δημιουργώντας έτσι αντίκτυπο στις ενέργειες του χρήστη είτε αυτές έχουν να κάνουν με εισαγωγή , ενημέρωση , διαγραφή ή απλώς αίτημα για ανάκτηση και παρουσίαση στην οθόνη. Αντί να επιβαρύνεται ο controller με πολύπλοκη επεξεργασία ή υλοποίηση αλγορίθμων, αυτή μεταφέρεται σε ειδικές κλάσεις, οι οποίες προσφέρουν καθαρότητα, επαναχρησιμοποίηση και δυνατότητα εύκολου ελέγχου (unit testing) και φυσικά οι controllers μπορούν να καλούν μέσω των μεθόδων τους (Action Methods) τις αντίστοιχες μεθόδους των Service Classes για να επιτευχθεί η κατάλληλη αλληλουχία στον

κώδικα. Στην προκειμένη περίπτωση, η κλάση ConstraintsSatisfactionProblem βρίσκεται στον φάκελο Services της εφαρμογής και προσφέρει τη στατική μέθοδο ExecuteCSP, η οποία εκτελεί τον αλγόριθμο με βάση τα εισερχόμενα δεδομένα και προφανώς αυτή η μέθοδος καλείται από ένα controller.
Το namespace `Google.OrTools.Sat` ανήκει στη βιβλιοθήκη **Google OR-Tools** και παρέχει όλα τα απαραίτητα εργαλεία για τη μοντελοποίηση και επίλυση **προβλημάτων ικανοποίησης περιορισμών (Constraint Satisfaction Problems - CSP)** μέσω του **CP-SAT Solver**. Περιλαμβάνει βασικές κλάσεις όπως `CpModel` για τον ορισμό του μοντέλου, `IntVar` και `BoolVar` για τις μεταβλητές απόφασης, `LinearExpr` για εκφράσεις γραμμικής μορφής, καθώς και την κλάση `CpSolver` για την επίλυση του μοντέλου. Το συγκεκριμένο namespace είναι κατάλληλο για προβλήματα που περιλαμβάνουν πλήθος από λογικούς και αριθμητικούς περιορισμούς, επιτρέποντας την εύκολη έκφραση σύνθετων συνθηκών και τη βελτιστοποίηση συναρτήσεων στόχου. Χρησιμοποιείται ευρέως σε εφαρμογές όπως κατανομές, χρονοπρογραμματισμός, logistics και – όπως στην παρούσα εργασία – σε εκπαιδευτικά σενάρια κατανομής φοιτητών.

Η υλοποίηση του αλγορίθμου αρχίζει με τη δημιουργία ενός μοντέλου περιορισμών (CpModel) που το παρέζει το πιο πάνω namespace, το οποίο θα περιέχει όλες τις μεταβλητές απόφασης, δηλαδή της μεταβήτές που θέλουμε να παρουν τιμή έπειτα από την εκτέλεση του αλγορίθμου και τους περιορισμούς που ελέγχουν τις τιμές που θα πάρουν αυτές οι μεταβλητές απόφασης. Με αυτό το τρόπο ορίζονται πρώτα οι μεταβλητές απόφασης (IntVar[,] assigned), οι οποίες καθορίζουν εάν ένας φοιτητής εγγράφεται σε ένα συγκεκριμένο μάθημα (αληθής/ψευδής – Boolean), όπως επίσης και οι υπόλοιπες συμπηρωματικές μεταβλητές απόφασης του προβλήματος όπως η satisfaction , minimum_satisfaction και total satisfaction . Συγεκριμένα, για κάθε συνδιασμό κάθε φοιτητή με κάθε μάθημα δημιουργείται μία μεταβλητή και όλες αυτές οι μεταβλητές αποτελούν μέρος ενός δυσδιάστατου πίνακα, του πίνακα IntVar[,] assigned . Στη συνέχεια, για κάθε φοιτητή δημιουργείται και μία μεταβλητή ικανοποίησης (satisfaction[i]), η οποία εκφράζει το πόσο "ικανοποιημένος" είναι ανάλογα με το αν πήρε τα μαθήματα που ήθελε ,δηλαδή εάν ένας φοιτητής εγγραφεί στην πρώτη του επιλογή , θα πάρει 100 μονάδες, αν εγγραφεί στην δεύτερη του επιλογή θα πάρει 50 μονάδες , αν εγγραφεί στην Τρίτη του επιλογή θα πάρει 25 μονάδες , αν εγγραφεί την Τέταρτη του επιλογή θα πάρει 12 μονάδες και αν εγγραφεί στην Πέμπτη του επιλογή θα πάρει 0 μονάδες. Ενώ, οι μεταβλητές minSatisfaction (για τον

ελάχιστο βαθμό ικανοποίησης) και totalSatisfaction (συνολική ικανοποίηση όλων των φοιτητών), αξιοποιούνται στη συνάρτηση βελτιστοποίησης.

Η μοντελοποίηση συνεχίζεται με την προσθήκη περιορισμών:

- Περιορισμός 1: Κάθε φοιτητής πρέπει να εγγράφεται σε ακριβώς τόσα μαθήματα όσα έχει ζητήσει (μέσω της μεταβλητής classesToTake[i]).
- Περιορισμός 2: Κάθε μάθημα δεν μπορεί να ξεπεράσει τη μέγιστη χωρητικότητα (courseCapacity[j]).
- Περιορισμός 3: Υπολογίζεται η ικανοποίηση κάθε φοιτητή, με βάση τη θέση του κάθε μαθήματος στη λίστα προτιμήσεών του. Τα μαθήματα σε υψηλότερη θέση προσφέρουν μεγαλύτερη βαθμολογία ικανοποίησης (π.χ. 100 μονάδες για πρώτη επιλογή, 50 για δεύτερη, κ.λπ.).
- Περιορισμός 4: Καθορίζεται ότι το minSatisfaction δεν μπορεί να είναι μεγαλύτερο από καμία επιμέρους ικανοποίηση φοιτητή, ώστε να μπορεί να χρησιμοποιηθεί στη συνάρτηση στόχου.
- Περιορισμός 5: Οι φοιτητές επιτρέπεται να ανατεθούν μόνο σε μαθήματα που βρίσκονται στις πρώτες 5 προτιμήσεις τους, μέσω κατάλληλων λογικών μεταβλητών (isInPreferenceList) και συνεπαγωγών (AddImplication).
- Περιορισμός 6: Υπολογίζεται η συνολική ικανοποίηση ως το άθροισμα των επιμέρους ικανοποιήσεων.
- Περιορισμός 7: Αν ο φοιτητής έχει δηλώσει ως πρώτη επιλογή μάθημα που σχετίζεται με πτυχιακή εργασία, εξασφαλίζεται η ανάθεσή του σε αυτό το μάθημα.

Ακολουθεί η συνάρτηση στόχου (model.Maximize(...)), η οποία στοχεύει στη μέγιστη συνολική ικανοποίηση όλων των φοιτητών, ενώ ταυτόχρονα επιδιώκει και τη μέγιστη ελαχιστοποίηση της διαφοράς ικανοποίησης μεταξύ των φοιτητών μέσω του όρου minSatisfaction * numStudents. Έτσι, δεν μεγιστοποιείται απλώς το συνολικό σκορ, αλλά διασφαλίζεται και η δικαιοσύνη (fairness) στην κατανομή.

Η μέθοδος καλεί στη συνέχεια τον CP-SAT Solver για την επίλυση του προβλήματος. Αν βρεθεί εφικτή ή βέλτιστη λύση (CpSolverStatus.Optimal ή Feasible), επιστρέφει τον πίνακα των μεταβλητών assigned και το αντικείμενο του solver για περαιτέρω ανάλυση ή εξαγωγή τιμών. Σε περίπτωση αποτυχίας, επιστρέφεται null, και παράγεται εξαίρεση με λεπτομέρειες

για ευκολότερο debugging όπου πριν από την εκτέλεση του αλγορίθμου αν τα δεδομένα εισόδου είναι διαμορφωμένα με τέτοι τρόπο που δεν θα μπορεί ο αλγόριθμος να βρει λύση θα ενημερώνεται η γραμματέας με σχετικό μήνυμα στην οθόνη της , ούτως ώστε να κάνει τις απαραίτητες αλλαγές , συνήθως στις θέσεις των μαθημάτων για να μπορεί ο αλγόριθμος να έχει λύση.

Πιο κάτω παρατίθεται η service class ConstraintsSatisfactionProblem για την εκτέλεση του αλγορίθμου:

```
using System;
using System.Linq;
using System.Linq.Expressions;
using Google.OrTools.Sat;

namespace WebApplicationPreRegistration.Services
{
   public static class ConstraintsSatisfactionProblem
   {
      public static (IntVar[,], CpSolver) ExecuteCSP(int numStudents, int numCourses, string[] courseName,
int[] courseCapacity, int[] classesToTake, int[,] preferences, bool[] isFirstChoiceForBachelorThesis)
      {
         try
         {
            // Create the model
            CpModel model = new CpModel();

            // Decision variables: assigned[i, j]
            IntVar[,] assigned = new IntVar[numStudents, numCourses];
            for (int i = 0; i < numStudents; i++)
            {
               for (int j = 0; j < numCourses; j++)
               {
                  assigned[i, j] = model.NewBoolVar($"assigned_{i}_{j}");
               }
            }

            // Satisfaction variables
            var satisfaction = new IntVar[numStudents];
            for (int i = 0; i < numStudents; i++)
            {
               satisfaction[i] = model.NewIntVar(0, 187, $"satisfaction_{i}");
            }

            // Minimum satisfaction variable
            var minSatisfaction = model.NewIntVar(0, 187, "min_satisfaction");

            // Create a sum variable to hold the total satisfaction
            var totalSatisfaction = model.NewIntVar(0, 187 * numStudents, "total_satisfaction");


            // Constraint 1: Each student is assigned to the exact number of classes they want
            for (int i = 0; i < numStudents; i++)
```

```
        {
            model.Add(LinearExpr.Sum(Enumerable.Range(0, numCourses).Select(j => assigned[i, j])) ==
classesToTake[i]);
        }

        // Constraint 2: Course capacity constraint
        for (int j = 0; j < numCourses; j++)
        {
            model.Add(LinearExpr.Sum(Enumerable.Range(0, numStudents).Select(i => assigned[i, j])) <=
courseCapacity[j]);
        }

        // Constraint 3: Satisfaction calculation
        for (int i = 0; i < numStudents; i++)
        {
            var satisfactionTerms = new List<IntVar>();
            for (int j = 0; j < numCourses; j++)
            {
                for (int rank = 0; rank < 5; rank++)
                {
                    if (preferences[i, rank] - 1 == j)
                    {
                        var satisfactionProduct = model.NewIntVar(0, 500, $"satisfaction_product_{i}_{j}");
                        int rankScore = (rank == 0) ? 100 : (rank == 1) ? 50 : (rank == 2) ? 25 : (rank == 3) ? 12 :
6;
                        model.AddMultiplicationEquality(satisfactionProduct, new IntVar[] { assigned[i, j],
model.NewConstant(rankScore) });
                        satisfactionTerms.Add(satisfactionProduct);
                    }
                }
            }
            model.Add(satisfaction[i] == LinearExpr.Sum(satisfactionTerms));
        }

        // Constraint 4: Min satisfaction is the minimum of all satisfactions
        foreach (var s in satisfaction)
        {
            model.Add(minSatisfaction <= s);
        }

        // Constraint 5: Students can only be assigned to courses in their preference list
        for (int i = 0; i < numStudents; i++)
        {
            for (int j = 0; j < numCourses; j++)
            {
                // Create a Boolean condition that checks if course j is in the top 5 preferences for student i
                BoolVar isInPreferenceList = model.NewBoolVar($"is_in_preference_list_{i}_{j}");
                var preferenceConditions = new List<ILiteral>();

                for (int k = 0; k < 5; k++) // Check top 5 preferences
                {
                    if (preferences[i, k] - 1 == j)
                    {
                        // If course j matches a preference, add the condition
                        preferenceConditions.Add(isInPreferenceList);
                    }
```

```
                }

                // Add a boolean constraint for isInPreferenceList
                if (preferenceConditions.Count > 0)
                {
                    // If course j is in preferences, set isInPreferenceList to true
                    model.AddBoolOr(preferenceConditions).OnlyEnforceIf(isInPreferenceList);
                }
                else
                {
                    // If course j is NOT in preferences, set isInPreferenceList to false
                    model.Add(isInPreferenceList == 0);
                }

                // Add the implication: If assigned[i, j], then isInPreferenceList must be true
                model.AddImplication((ILiteral)assigned[i, j], isInPreferenceList);
            }
        }

        // Constraint 6: Calculation of the totalSatisfaction variable
        model.Add(totalSatisfaction == LinearExpr.Sum(satisfaction));

        // Constraint 7: Reserved the first choice of bachelor thesis
        for (int i = 0; i < numStudents; i++)
        {
            if (isFirstChoiceForBachelorThesis[i] == true)
            {
                int firstChoiceCourseIndexInCourseName = preferences[i, 0] - 1; // Get the course index from
preferences
                if (firstChoiceCourseIndexInCourseName >= 0 && firstChoiceCourseIndexInCourseName <
numCourses)
                {
                    model.Add(assigned[i, firstChoiceCourseIndexInCourseName] == 1);
                }
            }
        }

        // Objective Function: Maximize the formulatedObjectiveFunction
        model.Maximize((minSatisfaction * numStudents) + totalSatisfaction);

        // Solve the model
        CpSolver solver = new CpSolver();
        var status = solver.Solve(model);

        // Output results
        if (status == CpSolverStatus.Optimal || status == CpSolverStatus.Feasible)
        {
            return (assigned, solver);
        }
        else
        {
            //No solution found
            return (null, null);
        }
    }
    catch (Exception ex)
```

```
        {
            throw new Exception("Number of Students: " + numStudents + ". Error in
ConstraintsSatisfactionProblem class. Message: " + ex.Message + "\nStackTrace: " + ex.StackTrace, ex);
        }
    }
  }
}
```

5.5 Αποθήκευση των Αποτελεσμάτων στην Βάση Δεδομένων

Η φάση της αποθήκευσης των αποτελεσμάτων του αλγορίθμου στην βάση δεδομένων αποτελεί κρίσιμο βήμα της διαδικασίας υλοποίησης και ολοκληρώνει την αλυσίδα μεταξύ επίλυσης του προβλήματος ικανοποίησης περιορισμών (CSP) και χρηστικής αξιοποίησης των αποτελεσμάτων από τους τελικούς χρήστες του συστήματος.
Αφού εκτελεστεί ο αλγόριθμος μέσω του ConstraintsSatisfactionProblem.ExecuteCSP() και παραχθεί ένα αποτέλεσμα – δηλαδή ένα δισδιάστατο πίνακα assigned[i,j] όπου οι τιμές 1 δηλώνουν την ανάθεση του φοιτητή i στο μάθημα j , ενώ αντιθέτως οι τιμές 0 δηλώνουν την μη εγγραφή του φοιτητή i στο μάθημα j – η εφαρμογή οφείλει να επεξεργαστεί και να αποθηκεύσει αυτά τα δεδομένα με τρόπο συνεπή και αποδοτικό στη Βάση Δεδομένων. Αυτό γίνεται μέσα από την παρέμβαση ενός Controller ο οποίος όπως είπαμε προηγουμένως χρησιμοποιεί ένα ενδιάμεσο Service Layer, αφού απαιτείται το σύστημα μας να προσφέρει και να λειτουργεί με καθαρό διαχωρισμό ευθυνών, όπως είναι επιθυμητό σε πιο σύνθετα ή επεκτάσιμα συστήματα τα οποία δεν έχουν απλώς μία αλληλεπίδραση με του χρήστη με τη Βάση Δεδομένων , αλλά παρέχουν μία σύνθετη επιχειρισιακή λογική , όπως το δικό μας σύστημα.

Κατά την επεξεργασία των αποτελεσμάτων, ο controller αναλύει τον επιστρεφόμενο πίνακα assigned από την service class, εντοπίζει τις περιπτώσεις όπου η τιμή είναι 1 και προβαίνει στη δημιουργία εγγραφών στον πίνακα AssignedLists. Για κάθε φοιτητή δημιουργείται μία νέα εγγραφή, στην οποία αποθηκεύεται η λίστα των μαθημάτων στα οποία έχει εγγραφεί επιτυχώς. Η αποθήκευση αυτή γίνεται με έναν από τους εξής δύο τρόπους:

Απλουστευμένη μορφή: Όπου η λίστα μαθημάτων αποθηκεύεται ως ένα ενιαίο string (π.χ., comma-separated codes ή τίτλοι μαθημάτων), στο πεδίο Courses της AssignedLists. Αυτή η προσέγγιση διευκολύνει την υλοποίηση και την ανάγνωση, αν και μειώνει την κανονικοποίηση της βάσης δεδομένων, όμως ήταν απαραίτητο για τον λόγο ότι ο κάθε

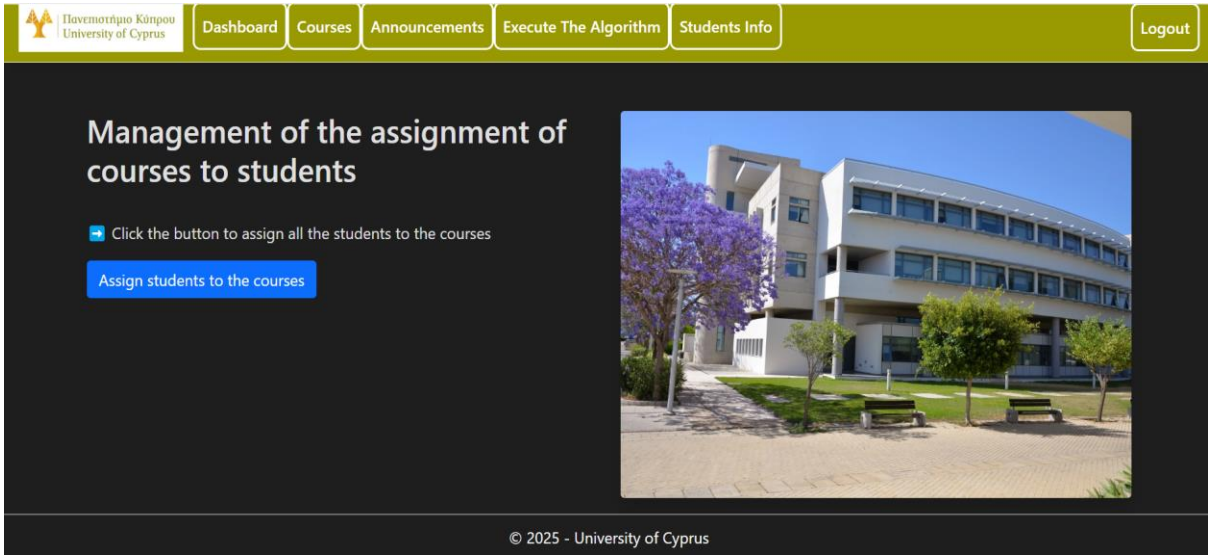
φοιτητής εγγράφεται σε διαφορετικό αριθμό μαθημάτων και γι'αυτό χρειαζόταν η χρήση μίας δυναμική λίστας συμβολοσειρών .

Η εγγραφή στο AssignedLists περιλαμβάνει το UserId ως το ξένοπ κλειδί που συνδέει την κάθε λίστα μοναδικά με τον κάθε φοιτητή και την παραγόμενη λίστα μαθημάτων, ενώ προτού αποθηκευτεί κάθε εγγραφή, πραγματοποιούνται απαραίτητοι έλεγχοι εγκυρότητας (validation), όπως επιβεβαίωση ότι ο UserId υπάρχει και ότι τα μαθήματα είναι αποδεκτά entries στον πίνακα Courses.
Από προγραμματιστικής και μηχανικής λογισμικού άποψης, αυτή η διαδικασία αποθήκευσης στηρίζεται στη χρήση του Entity Framework. Με τη βοήθεια του EF context (ApplicationDbContext), κάθε νέα εγγραφή AssignedList προστίθεται μέσω του context.AssignedLists.Add() και η αποθήκευση ολοκληρώνεται με context.SaveChanges().
Επιπρόσθετα, η αποθήκευση των αποτελεσμάτων με αυτό τον τρόπο καθιστά δυνατή τη μεταγενέστερη προβολή από τον ίδιο τον φοιτητή, ο οποίος, μέσω του front-end της εφαρμογής, μπορεί να δει τα μαθήματα στα οποία του αποδόθηκαν θέσεις. Η υπηρεσία που εξυπηρετεί αυτή τη λειτουργία βασίζεται σε queries που ανακτούν τις αντίστοιχες εγγραφές AssignedList ανά χρήστη και τις προβάλλουν φιλτραρισμένα με βάση τον UserId.

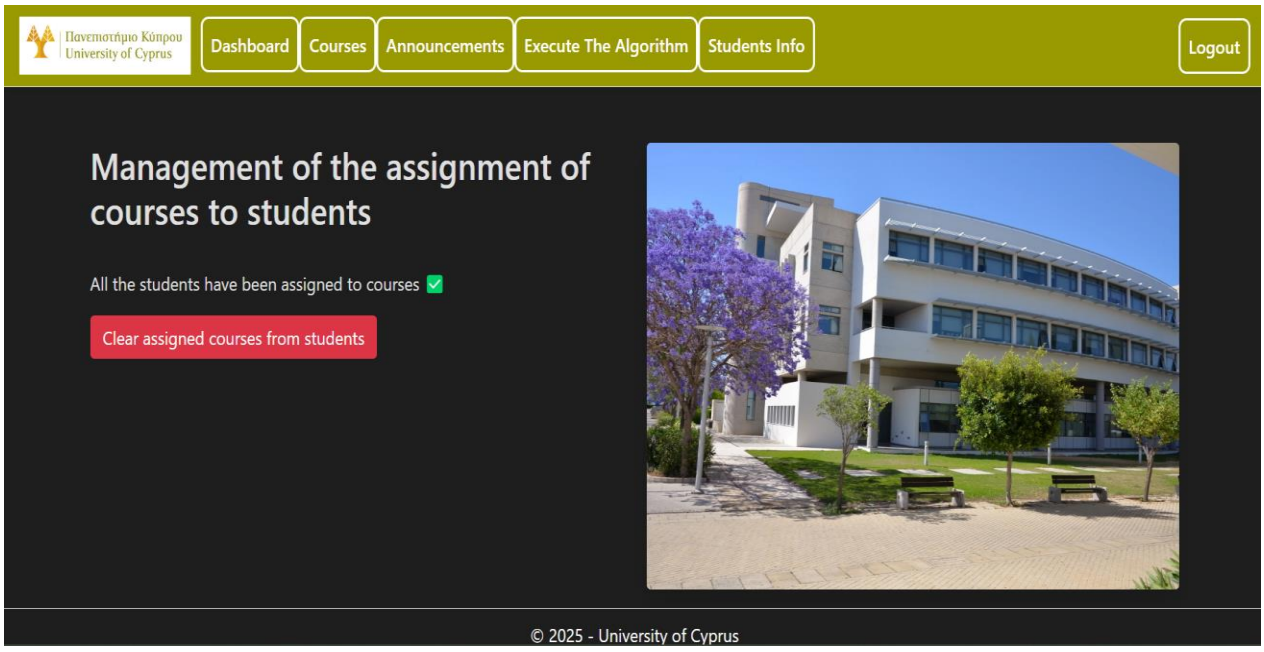
Τέλος, η διαδικασία αποθήκευσης των αποτελεσμάτων δεν είναι απλώς μία πράξη εγγραφής στη βάση. Είναι ένα κρίσιμο βήμα που γεφυρώνει τον αλγόριθμο με την εμπειρία χρήστη, που απαιτεί ακρίβεια, συνέπεια και δυνατότητα μελλοντικής επεκτασιμότητας. Μέσω της συνδυασμένης χρήσης του CpSolver, του ASP.NET MVC Core framework και του Entity Framework, το σύστημα παρέχει μια πλήρη, σταθερή και επαναχρησιμοποιήσιμη υποδομή κατανομής μαθημάτων βάσει προτιμήσεων και περιορισμών.

➔Η εκτέλεση του αλγορίθμου όπως είπαμε και προηγουμένως γίνεται από την γραμματέα , όπως στο δικό της Layout υπάρχει η επιλογή Execute the Algorithm , όπου μπορούν να τις εμφανιστούν οι 3 πιο κάτω περιπτώσεις:

1η περίπτωση: Όταν στη Βάση Δεδομένων δεν υπάρχει κανένας φοιτητής γραμμένος σε μάθημα είτε έπειτα από εκτέλεση του αλγορίθμου είτε απευθείας από ανάθεση της γραμματέας ένα φοιτητή σε μάθημα και για αυτό υπάρχει μόνο η επελογή 'Assign students to the courses', η οποία ουσιαστικά θα εγγράψει όλους τους φοιτητές στο σύστημα που έχουν καταχωρήσει priority list σε μαθήματα περιορισμένης επιλογής.

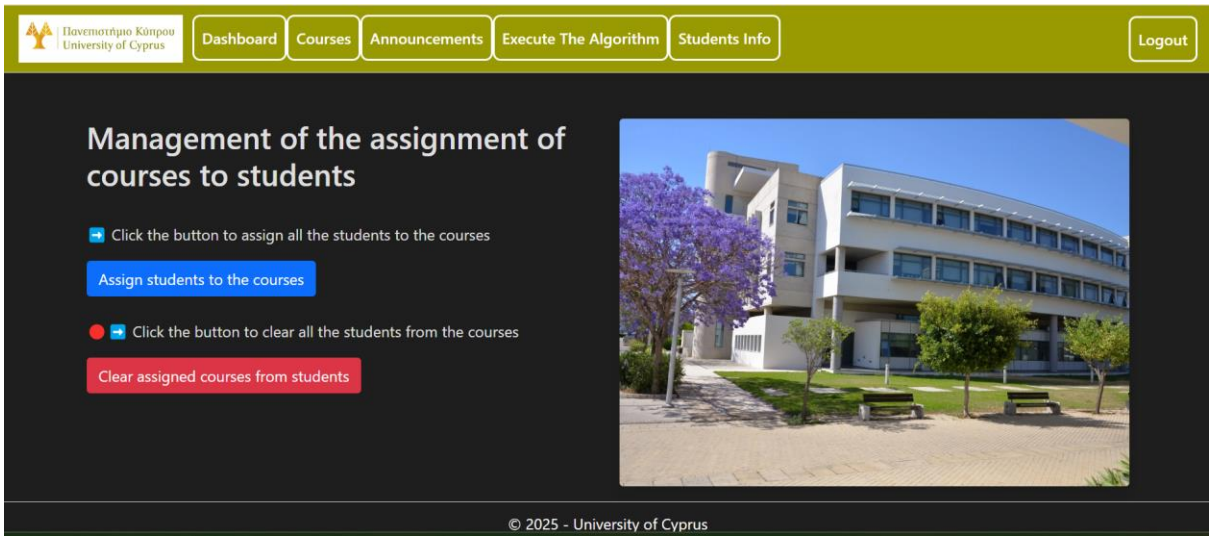


2η περίπτωση: Όταν στη Βάση Δεδομένων όλοι οι φοιτητές είναι γραμμένοι σε μαθήματα είτε έπειτα από εκτέλεση του αλγορίθμου είτε από απευθείας ανάθεση του διαχειριστή του συστήματος ενός φοιτητή σε μάθημα και για αυτό υπάρχει μόνο η επιλογή 'Clear assigned courses from students' που θα 'καθαρίσει' ουσιαστικά τη βάση δεδομένων.



3η περίπτωση: Όταν στη Βάση Δεδομένων υπάρχουν κάποιοι φοιτητές οι οποίοι είναι γραμμένοι σε μαθήματα είτε έπειτα από εκτέλεση του αλγορίθμου είτε απευθείας από

ανάθεση της γραμματέας σε ένα φοιτητή σε μάθημα και κάποιοι άλλοι φοιτητές που ενώ έχουν καταχωρημένη priority list στο σύστημα δεν είναι γραμμένοι σε κανένα μάθημα και για αυτό υπάρχει η επιλογή 'Clear assigned courses from students' που θα 'καθαρίσει' ουσιαστικά τη βάση δεδομένων , αλλά και η επιλογή 'Assign students to the courses', η οποία ουσιαστικά θα εγγράψει όλους τους φοιτητές που δεν είναι τώρα γραμμένοι σε μαθήματα χωρίς να επηρεαστούν οι φοιτητές που είναι ήδη γραμμένοι σε μαθήματα.



Κεφάλαιο 6

Αξιολόγηση και Δοκιμές με Unit Testing / Αξιολόγηση Αλγορίθμου και Σύγκριση με άλλους αλγορίθμους

6.1 Δοκιμές Συστήματος με xUnit(Unit Testing)

6.1.1 Εισαγωγή στο xUnit Framework

Το xUnit Framework είναι ένα από τα πιο δημοφιλή και ευρέως χρησιμοποιούμενα frameworks για Unit Testing στο περιβάλλον .NET και συγκεκριμένα στο ASP.NET Core. Το όνομα του προέρχεται από την οικογένεια "xUnit" frameworks (όπως JUnit για Java, NUnit, PyUnit κ.ά.), τα οποία ακολουθούν παρόμοια δομή και φιλοσοφία για την αυτοματοποιημένη δοκιμή κώδικα.

Η βασική αρχή πίσω από τη χρήση του xUnit και γενικώς το Unit Testing είναι ο διαχωρισμός της λογικής του προγράμματος σε μονάδες (units/modules), οι οποίες μπορούν να ελεγχθούν αυτόνομα μέσω μικρών, απομονωμένων σεναρίων χωρίς να υπάρχει κάποια

επιρροή μεταξύ των επιμέρους μονάδων του προγράμματος. Κάθε unit test επικεντρώνεται στο να επαληθεύσει μια συγκεκριμένη συμπεριφορά μιας συνάρτησης ή μεθόδου (μονάδας), συνήθως με συγκεκριμένες εισόδους και αναμενόμενες εξόδους. Αυτή η προσέγγιση βοηθά στον εντοπισμό λαθών νωρίς στην ανάπτυξη, μειώνει το τεχνικό χρέος και ενισχύει τη συντηρησιμότητα και αξιοπιστία του κώδικα.

Το xUnit είναι πλήρως συμβατό με το .NET Core και υποστηρίζεται επίσημα από τη Microsoft γι'αυτό και ήταν και το ιδανικό testing framework που μπορούσαμε να χρησιμοποιήσουμε. Προσφέρει ένα απλό αλλά ισχυρό API, υποστηρίζει attribute-based configuration ([Fact], [Theory], [InlineData] κ.λπ.), ενώ συνεργάζεται άψογα με mocking frameworks όπως το Moq και με εργαλεία μέτρησης κάλυψης κώδικα (π.χ. Coverlet). Παράλληλα, η ενσωμάτωσή του στο Visual Studio IDE , καθώς και η δυνατότητα εκτέλεσης των tests μέσω του command-line interface (dotnet test), καθιστούν τη χρήση του ιδιαίτερα ευέλικτη για μικρές και μεγάλες ομάδες ανάπτυξης εντοπίζοντας ξεκάθαρα τυχόν λάθος που μπορεί να υπάρξει στον κώδικα.

Στο πλαίσιο της παρούσας εφαρμογής, το xUnit αξιοποιήθηκε για να διασφαλιστεί ότι τόσο η επιχειρησιακή λογική (business logic), δηλαδή η λογική εκτέλεση του αλγόριθμου ικανοποίησης περιορισμών όσο και η διασύνδεση με τη βάση δεδομένων και τις υπηρεσίες της εφαρμογής λειτουργούν ορθά, ακόμα και κάτω από συνθήκες μη έγκυρων εισόδων ή ειδικών περιπτώσεων , με τον κώδικα να αντιδρά όπως πρέπει , δηλα΄δη με την εκτύπωση σχετικών μηνυμάτων 'ρίχνοντας' εξαιρέσεις (exceptions). Μέσω των tests διασφαλίζεται η αξιοπιστία του συστήματος και εντοπίζονται άμεσα σφάλματα ή παραβιάσεις λογικών περιορισμών.

6.1.2 Κατηγορίες Τεστ που υλοποιήθηκαν

Κατά την υλοποίηση της διαδικασίας δοκιμής και αξιολόγησης του συστήματος, κρίθηκε απαραίτητο να ενσωματωθούν επαρκή και ουσιαστικά unit tests που καλύπτουν τόσο τη λειτουργική λογική του συστήματος όσο και τα βασικά του controllers. Η βιβλιοθήκη xUnit Framework επιλέχθηκε για τη συγγραφή των δοκιμών, σε συνδυασμό με το εργαλείο Moq για τον εξομοιωτή εξαρτήσεων (mocking dependencies). Οι βασικές κατηγορίες δοκιμών περιλάμβαναν τον έλεγχο της κύριας service class ConstraintsSatisfactionProblem, καθώς και

των controllers: UserController, CourseController, PriorityListController, AssignedListController και AnnouncementController.

Η service class ConstraintsSatisfactionProblem αποτελεί το κεντρικό σημείο της επιχειρησιακής λογικής (business logic) για την επίλυση του προβλήματος κατανομής μαθημάτων μέσω τεχνικών Περιοριστικού Προγραμματισμού (Constraint Programming), αξιοποιώντας τη βιβλιοθήκη Google OR-Tools. Η μέθοδος ExecuteCSP() υλοποιεί ένα πλήρες μοντέλο βελτιστοποίησης που περιλαμβάνει μεταβλητές απόφασης, περιορισμούς και αντικειμενική συνάρτηση. Για τη διαδικασία δοκιμής αυτής της μεθόδου, υλοποιήθηκαν unit tests τα οποία βασίζονται στη δημιουργία ελεγχόμενων, συνθετικών σεναρίων εισόδου, χωρίς την ανάγκη χρήσης mocking, καθώς η κλάση είναι στατική και δεν εξαρτάται από εξωτερικές υπηρεσίες ή αποθετήρια δεδομένων.

Συγκεκριμένα, τα σενάρια περιλάμβαναν τη δημιουργία διαφορετικών συνδυασμών φοιτητών, μαθημάτων, χωρητικοτήτων και πινάκων προτιμήσεων, με στόχο να καλυφθούν όλες οι κρίσιμες περιπτώσεις συμπεριφοράς του αλγορίθμου. Παραδείγματα περιλαμβάνουν φοιτητές που ζητούν διαφορετικό αριθμό μαθημάτων, μαθήματα με περιορισμένη ή υπερβολική χωρητικότητα, καθώς και προτιμήσεις που επικαλύπτονται ή αποκλίνουν μεταξύ φοιτητών. Για κάθε τέτοιο σενάριο, η ExecuteCSP() καλείται και επιστρέφει δύο τιμές: τον πίνακα των μεταβλητών assigned[i,j] που δείχνουν αν ο φοιτητής i έχει ανατεθεί στο μάθημα j, και τον solver ο οποίος περιέχει τις βέλτιστες ή εφικτές τιμές για τις μεταβλητές του μοντέλου.

Κατά την αξιολόγηση της μεθόδου, χρησιμοποιούνται διάφορες μορφές assertions μέσω του xUnit framework. Οι δοκιμές ελέγχουν καταρχάς αν ο solver που επιστρέφεται δεν είναι null, υποδηλώνοντας ότι βρέθηκε κάποια λύση (είτε βέλτιστη είτε εφικτή). Επιπλέον, με χρήση του solver διαβάζονται οι τιμές των μεταβλητών assigned[i,j] και επαληθεύεται μέσω Assert.Equal() και Assert.True() ότι πληρούνται οι βασικοί περιορισμοί του μοντέλου: κάθε φοιτητής παρακολουθεί ακριβώς τον επιθυμητό αριθμό μαθημάτων (όπως ορίζεται από το classesToTake[i]), κανένα μάθημα δεν ξεπερνά τη χωρητικότητά του (courseCapacity[j]), και κάθε ανάθεση μαθημάτων γίνεται μόνο μέσα από τις 5 πρώτες προτιμήσεις του φοιτητή.

Ιδιαίτερη προσοχή δόθηκε στην επαλήθευση πιο εξειδικευμένων περιορισμών, όπως η αυτόματη ανάθεση της πρώτης επιλογής πτυχιακής εργασίας στους φοιτητές που το δικαιούνται, καθώς και η βελτιστοποίηση της ελάχιστης ικανοποίησης (minSatisfaction) για όλους τους φοιτητές. Η αντικειμενική συνάρτηση ελέγχεται έμμεσα, με συγκριτικά σενάρια

όπου διαφορετικές διατάξεις δεδομένων αναμένεται να παράγουν διαφορετικό συνολικό σκορ (totalSatisfaction), και αξιολογείται η απόδοση του αλγορίθμου.
Συνολικά, η διαδικασία αυτή αποδεικνύει ότι η ConstraintsSatisfactionProblem.ExecuteCSP() δεν είναι απλώς μία αριθμητική μέθοδος, αλλά ένας πλήρης μηχανισμός λήψης αποφάσεων, τον οποίο μπορούμε να δοκιμάσουμε, να προβλέψουμε και να βελτιώσουμε. Η χρήση του xUnit, η κατασκευή αναλυτικών σεναρίων και η αξιοποίηση assertions επιτρέπουν τη συστηματική επαλήθευση της ακρίβειας, της σταθερότητας και της αποτελεσματικότητας του μοντέλου.

Στον UserController, που χειρίζεται λειτουργίες διαχείρισης χρηστών, τα τεστ εστίασαν σε λειτουργίες όπως η δημιουργία νέων χρηστών και η ανάκτηση στοιχείων. Χρησιμοποιώντας mocking του IUserService ή του DbContext, προσομοιώθηκε η επιστροφή ενός χρήστη, ενώ η μέθοδος του controller καλούνταν κανονικά. Ο έλεγχος των αποτελεσμάτων γινόταν με χρήση Assert.IsType<OkObjectResult>, ώστε να διασφαλιστεί ότι η απόκριση ήταν της αναμενόμενης μορφής.

Αντίστοιχα, στον CourseController, υλοποιήθηκαν τεστ για μεθόδους όπως GetAllCourses, AddCourse και DeleteCourse. Με τη χρήση mock του ICourseService, επιστρεφόταν μία λίστα μαθημάτων κατά την κλήση του controller, και μέσω της μεθόδου Assert.Equal() γινόταν επαλήθευση του αριθμού των επιστρεφόμενων εγγραφών, επιβεβαιώνοντας ότι η λειτουργία του controller ανταποκρίνεται στα δεδομένα του mock.

Στον PriorityListController, η λογική εστιάζει στην υποβολή και έλεγχο λιστών προτιμήσεων από τους φοιτητές. Τα τεστ επικεντρώθηκαν στη σωστή επεξεργασία έγκυρων και μη έγκυρων δεδομένων. Με mocking των σχετικών services και την κατασκευή δεδομένων εισόδου που περιείχαν μη αποδεκτούς κωδικούς μαθημάτων, οι μέθοδοι του controller καλούνταν για να επιβεβαιωθεί ότι ορθώς επιστρεφόταν BadRequestResult μέσω του Assert.IsType<BadRequestResult>().

Η AssignedListController ήταν υπεύθυνη για την επιστροφή των αναθέσεων μαθημάτων που παρήγαγε ο αλγόριθμος. Στο πλαίσιο των τεστ, δημιουργήθηκαν δεδομένα ανάθεσης (π.χ. τρία μαθήματα σε έναν φοιτητή), και μέσω mocking του IAssignedListService επιστρεφόταν η προσομοιωμένη λίστα. Με τη χρήση του Assert.Contains() γινόταν επιβεβαίωση ότι τα επιστρεφόμενα μαθήματα ήταν τα αναμενόμενα και σωστά αποθηκευμένα.

Τέλος, στον AnnouncementController, ο οποίος διαχειρίζεται τις ανακοινώσεις της γραμματείας, δοκιμάστηκαν σενάρια προσθήκης και ανάκτησης ανακοινώσεων. Με τη χρήση mock του IAnnouncementService, κατασκευάστηκε μία νέα ανακοίνωση που προστίθεται μέσω της σχετικής μεθόδου, ενώ επιβεβαιώνεται ότι επιστρέφεται σωστά CreatedAtActionResult, δηλαδή το HTTP αποτέλεσμα προσθήκης πόρου.

Σε όλα τα παραπάνω τεστ, η διαδικασία περιλάμβανε τέσσερα βασικά βήματα: mocking εξαρτήσεων ώστε να απομονωθεί η μονάδα, δημιουργία δεδομένων για κάθε περίπτωση, εκτέλεση της μεθόδου που εξετάζεται και τέλος επαλήθευση αποτελεσμάτων μέσω Assert. Η επιλογή των συγκεκριμένων controllers και service αντικατοπτρίζει τα πιο κρίσιμα κομμάτια του συστήματος και διασφαλίζει ότι τόσο η λογική όσο και τα endpoints λειτουργούν όπως αναμένεται. Συνολικά, η εφαρμογή των unit tests συνέβαλε στην αξιοπιστία του συστήματος, στον έγκαιρο εντοπισμό σφαλμάτων και στη διατήρηση της σταθερότητας κατά τις επεκτάσεις του λογισμικού.

6.2 Αξιολόγηση Αλγορίθμου Κατανομής Μαθημάτων

6.2.1 Απαιτήσεις Αλγόριθμου Ανάθεσης Μαθημάτων

Ο αλγόριθμος ανάθεσης μαθημάτων που υλοποιήσαμε στο πλαίσιο του Συστήματος Προεγγραφής για το Τμήμα Πληροφορικής βασίζεται σε ένα πρόβλημα Ικανοποίησης Περιορισμών και ταυτόχρονα σε ένα πρόβλημα Βελτιστοποίησης (CSP-SAT). Κεντρικός στόχος κατά τη σχεδίαση και υλοποίηση του αλγόριθμου ήταν να επιτευχθεί μέγιστη δυνατή συνολική ικανοποίηση των φοιτητών, διατηρώντας παράλληλα έναν υψηλό βαθμό δικαιοσύνης και ισοκατανομής ευκαιριών.

Για τον λόγο αυτό, ο αλγόριθμος δεν περιορίζεται στην εύρεση μίας οποιασδήποτε αποδεκτής λύσης (feasible solution), αλλά επιδιώκει τη βέλτιστη λύση (optimal solution)

βάσει μίας ειδικά σχεδιασμένης αντικειμενοστρεφούς συνάρτησης. Η συνάρτηση αυτή επιχειρεί να ισορροπήσει δύο σημαντικές πτυχές:

Το συνολικό άθροισμα ικανοποίησης όλων των φοιτητών (total satisfaction), το οποίο υπολογίζεται με βάση τις προτιμήσεις τους για τα μαθήματα στα οποία εγγράφονται.

Την ελάχιστη ικανοποίηση μεταξύ όλων των φοιτητών (minimum satisfaction), που διασφαλίζει ότι κανένας φοιτητής δεν θα αδικηθεί σημαντικά σε σχέση με τους υπόλοιπους.

Η διατύπωση της αντικειμενικής συνάρτησης είναι η εξής:

```
model.Maximize((minSatisfaction * numStudents) + totalSatisfaction);
```

Η προσέγγιση αυτή έχει ιδιαίτερη σημασία καθώς ενθαρρύνει τη δίκαιη μεταχείριση όλων των φοιτητών, προσπαθώντας όχι μόνο να αυξήσει τον μέσο όρο ικανοποίησης αλλά και να αποφύγει ακραίες καταστάσεις, όπου κάποιος φοιτητής μπορεί να έχει πολύ χαμηλή ικανοποίηση. Με άλλα λόγια, η λύση στο πρόβλημα δεν είναι αποδεκτή αν μερικοί φοιτητές έχουν πλήρη ικανοποίηση εις βάρος άλλων που μένουν εντελώς ανικανοποίητοι.

Επιπλέον, ο αλγόριθμος σέβεται πλήρως τους λειτουργικούς περιορισμούς του συστήματος, όπως:

- Ο αριθμός των μαθημάτων που θέλει να παρακολουθήσει κάθε φοιτητής.
- Η χωρητικότητα κάθε μαθήματος.
- Η απαίτηση κάθε φοιτητής να εγγράφεται μόνο σε μαθήματα από τις πέντε κορυφαίες προτιμήσεις του.
- Η ανάγκη υποχρεωτικής εγγραφής στην πρώτη επιλογή διπλωματικής εργασίας για όσους φοιτητές την έχουν επιλέξει.

Με βάση τα παραπάνω, διαμορφώθηκε ένας αλγόριθμος που δεν μεταχειρίζεται τους φοιτητές σαν απομονωμένες περιπτώσεις, αλλά λαμβάνει υπόψη του τη συνολική κατανομή ικανοποίησης στον πληθυσμό, με στόχο να προκύψει μια λύση που να είναι ταυτόχρονα αποτελεσματική και κοινωνικά δίκαιη.

6.2.2 Μετρικές Αξιολόγησης

Για την αξιολόγηση της ποιότητας των λύσεων που παράγει ο αλγόριθμος ανάθεσης μαθημάτων, σχεδιάστηκε ένα σύνολο από ποσοτικές μετρικές, οι οποίες μας επιτρέπουν να μετρήσουμε σε βάθος τον βαθμό επιτυχίας της κατανομής και να εντοπίσουμε περιπτώσεις ανισότητας ή χαμηλής ικανοποίησης. Οι μετρικές αυτές αξιολογούνται σε κάθε διαφορετικό σενάριο εισόδου ή παραμετροποίησης και περιγράφονται ως εξής:

- **Αριθμός φοιτητών που έλαβαν τη μέγιστη δυνατή ικανοποίηση**
  Αυτή η μετρική υπολογίζει πόσοι φοιτητές εγγράφηκαν ακριβώς στα μαθήματα που αποτελούσαν τις πρώτες τους προτιμήσεις, με αποτέλεσμα να έχουν την υψηλότερη δυνατή τιμή satisfaction (συνήθως 100). Αντικατοπτρίζει την ικανότητα του συστήματος να εξυπηρετεί τις κορυφαίες προτιμήσεις των φοιτητών.

- **Αριθμός φοιτητών που έλαβαν ικανοποίηση κάτω από το ήμισυ της μέγιστης δυνατής**
  Αυτή η μετρική αναδεικνύει περιπτώσεις φοιτητών που, λόγω περιορισμών, δεν μπόρεσαν να εγγραφούν στα μαθήματα υψηλής προτεραιότητας και κατέληξαν με χαμηλή συνολική ικανοποίηση (π.χ. κάτω από 50). Αποτελεί ένδειξη της «αποτυχίας» του συστήματος να ικανοποιήσει επαρκώς το σύνολο των φοιτητών.

- **Ελάχιστη Ικανοποίηση (Minimum Satisfaction)**
  Αναφέρεται στον φοιτητή με τη χαμηλότερη συνολική ικανοποίηση. Είναι κρίσιμη για την εκτίμηση της δικαιοσύνης της λύσης, καθώς επιθυμούμε να διατηρήσουμε το ελάχιστο επίπεδο ικανοποίησης όσο το δυνατόν υψηλότερο, αποφεύγοντας την απογοήτευση συγκεκριμένων φοιτητών.

- **Μέγιστη Ικανοποίηση (Maximum Satisfaction)**
  Αν και συχνά αναμενόμενο να υπάρχουν φοιτητές με μέγιστη ικανοποίηση, αυτή η μετρική μας επιβεβαιώνει ότι το σύστημα είναι ικανό να προσφέρει βέλτιστα

αποτελέσματα τουλάχιστον σε ένα μέρος του πληθυσμού. Ωστόσο, αν υπάρχει αλγόριθμος ικανοποίησης περιορισμών ο οποίος δεν έχει έχει τουλάχιστον ένα φοιτητή που να έχει την μέγιστη δυνατή ικανοποίηση μιλώντας στην γλώσσα των μαθηματικών τότε αυτός ο αλγόριθμος δεν είναι ο κατάλληλος . Για αυτό , μπορούμε να πούμε ότι αυτό που είναι πολύ σημαντικό είναι να έχουμε όσο το δυνατό περισσότερους φοιτητές οι οποίοι να λάβουν την μέγιστη δυνατή ικανοποίηση.

- **Μέση Ικανοποίηση (Average Satisfaction)**
  Δείχνει τον γενικό βαθμό ικανοποίησης του συνόλου των φοιτητών και αποτελεί συνοπτικό δείκτη επιτυχίας της λύσης. Όσο μεγαλύτερη είναι η μέση τιμή, τόσο καλύτερα εξυπηρετήθηκαν συνολικά οι φοιτητές. Δηλαδή η μεταβλήτή απόφασης total_satisfaction που χρησιμοποιούμε στον υφιστάμενο αλγόριθμο ικανοποίησης περιορισμών αποτελέι ουσιαστικά ανάλογη μετρική , αφού μαθηματικά μιλώντας , όσο μεγαλύτερη είναι η συνολική ικανοποίηση των φοιτητών , τόσο μεγαλύτερη θα είναι και η μέση ικανοποίηση των φοιτητών.

- **Τυπική Απόκλιση (Standard Deviation)**
  Η τυπική απόκλιση μετρά τη διασπορά των τιμών ικανοποίησης. Μία χαμηλή τιμή υποδηλώνει ότι η ικανοποίηση των φοιτητών ήταν σχετικά ομοιογενής, ενισχύοντας τη δικαιοσύνη της λύσης. Αντίθετα, μία υψηλή τιμή δείχνει μεγάλες διαφορές στην ικανοποίηση μεταξύ φοιτητών, κάτι που ενδέχεται να είναι ανεπιθύμητο.Η τυπική απόκλιση δεν μπορούσε να αποτελεί μέρος του αλγόριθμου ικανοποίησης περιορισμού λόγω των πολλών ενδιάμεσων μεταβλητών που χρειάζεται και με αυτό τον τρόπο αυξανόταν πάρα πολύ η πολυπλοκότητα του προβλήματος.

6.2.3 Σύγκριση Αλγορίθμου με Παραλλαγές του

Οι συγκρίσεις που έγιναν ήταν με βάση τους πιο κάτω αλγόριθμους οι οποίοι εκτελέστηκαν σε console applications της .NET Core χρησιμοποιώντας τα Google OR-Tools, όπου σε αντίθεση με το project ASP.NET MVC CORE που υλοποίηθηκε μία ολόκληρη διαδικτυακή

εφαρμογή, εδώ απλώς τα αποτελέσματα του αλγόριθμου (δηλαδή ικανοποίηση του κάθε φοιτητή, ανάθεση μαθημάτων, κ.α.) εκτυπώνονται στην οθόνη:

1) **Maximum Total Satisfaction:**

   Περιγραφή:

   Ο αλγόριθμος αυτός στοχεύει στη μέγιστη δυνατή συνολική ικανοποίηση όλων των φοιτητών.

2) **Maximize number of students that took their first choice**

   Περιγραφή:
   Εστιάζει στο να εγγραφούν όσο το δυνατόν περισσότεροι φοιτητές στο πρώτο μάθημα της λίστας προτεραιότητάς τους.

3) **Minimize Maximum Difference from best to worst satisfaction**

   Περιγραφή:

   Αυτός ο αλγόριθμος μειώνει τη μέγιστη διαφορά μεταξύ του πιο ικανοποιημένου και του λιγότερο ικανοποιημένου φοιτητή.

4) **Maximize Minimum Satisfaction**

   Περιγραφή:
   Εστιάζει στην βελτιστοποίηση του χειρότερου σεναρίου, δηλαδή προσπαθεί να διασφαλίσει ότι ο φοιτητής με τη μικρότερη ικανοποίηση έχει όσο το δυνατόν καλύτερη εμπειρία.

5) **Maximize Minimum Satisfaction Ratio**

   Περιγραφή:

   Αντί να βλέπει απόλυτη ικανοποίηση, εστιάζει στο ποσοστό (ratio) που καλύπτεται από τις επιθυμίες κάθε φοιτητή. Για παράδειγμα, αν ένας φοιτητής ζητά 4 μαθήματα και παίρνει 2 πρώτες επιλογές, έχει ικανοποίηση 50%.

6) **Maximize Total Satisfaction and Minimum Satisfaction**

Περιγραφή:

Πρόκειται για έναν υβριδικό αλγόριθμο που στοχεύει να πετύχει ισορροπία μεταξύ της συνολικής ικανοποίησης όλων των φοιτητών και της ικανοποίησης του λιγότερο ικανοποιημένου φοιτητή. Η αντικειμενική του συνάρτηση είναι σχεδιασμένη έτσι ώστε να

μεγιστοποιεί τόσο το total satisfaction όσο και το minimum satisfaction, δίνοντας ίσο βάρος και στις δύο διαστάσεις του προβλήματος. Αυτός αποτελεί και τον αλγόριθμο που έχουμε υλοποιήσει στο σύστημα μας.

Πιο κάτω ανατίθεται λεπτομερής περιγραφή των αποτελεσμάτων από την εκτέλεση των 6 αλγορίθμων:

1) Ο αλγόριθμος **Maximum Total Satisfaction**, παρουσίασε πολύ θετικά αποτελέσματα. Πιο συγκεκριμένα, 34 φοιτητές πέτυχαν τη μέγιστη δυνατή ικανοποίηση, ενώ κανένας φοιτητής δεν είχε ικανοποίηση κάτω από το μισό της μέγιστης, γεγονός που δείχνει ότι η λύση είναι γενικά ικανοποιητική για όλους. Η ελάχιστη τιμή ικανοποίησης ήταν 100, αρκετά υψηλή σε σχέση με το μέγιστο των 175, ενώ η μέση ικανοποίηση ήταν 142.12, αποδεικνύοντας ότι το σύνολο των φοιτητών έλαβε μαθήματα αρκετά κοντά στις προτιμήσεις του. Η τυπική απόκλιση ήταν 25.64, ένδειξη ότι υπάρχει κάποια διακύμανση ανάμεσα στις ικανοποιήσεις των φοιτητών, αλλά όχι υπερβολική. Συνολικά, ο αλγόριθμος πέτυχε εξαιρετικά επίπεδα συνολικής ικανοποίησης χωρίς να αποκλείσει φοιτητές από ικανοποιητικές αναθέσεις μαθημάτων, αν και επικεντρώνεται περισσότερο στη συνολική απόδοση παρά στην απόλυτη ισοκατανομή μεταξύ των φοιτητών.

2) Ο δεύτερος αλγόριθμος, **Maximize number of students that took their first choice,** είχε ως κύριο στόχο να εγγραφούν όσο το δυνατόν περισσότεροι φοιτητές στο μάθημα της πρώτης τους επιλογής και τα αποτελέσματα δείχνουν ότι κατάφερε να το πετύχει σε μεγάλο βαθμό. Συγκεκριμένα, 35 φοιτητές είχαν τη μέγιστη δυνατή ικανοποίηση, ενώ κανένας δεν βρέθηκε κάτω από το μισό της μέγιστης δυνατής τιμής, γεγονός που υποδεικνύει δίκαιη κατανομή. Η ελάχιστη τιμή ικανοποίησης ήταν 100 και η μέγιστη 175, όπως και στον πρώτο αλγόριθμο, ενώ η μέση ικανοποίηση ήταν 141.97, ελάχιστα χαμηλότερη σε σύγκριση με την προηγούμενη περίπτωση. Η τυπική απόκλιση ήταν 25.57, που δείχνει ότι η διακύμανση στις ικανοποιήσεις των φοιτητών παρέμεινε στα ίδια επίπεδα. Συνολικά, ο αλγόριθμος αυτός πέτυχε υψηλή απόδοση και παράλληλα εστίασε στην προτεραιότητα των φοιτητών, χωρίς να προκαλέσει σημαντικές ανισότητες στην κατανομή των μαθημάτων.

3) Ο τρίτος αλγόριθμος, **Minimize Maximum Difference from best to worst satisfaction,** είχε ως στόχο τη μείωση της διαφοράς μεταξύ της υψηλότερης και της χαμηλότερης ικανοποίησης, επιδιώκοντας μια ομοιόμορφη κατανομή. Αν και κατάφερε να πετύχει τον μικρότερο εύρος ικανοποίησης (μόλις 18 μονάδες διαφορά μεταξύ 50 και 68), αυτό επιτεύχθηκε εις βάρος της συνολικής ικανοποίησης των φοιτητών. Συγκεκριμένα, κανένας φοιτητής δεν έφτασε τη μέγιστη ικανοποίηση, ενώ 40 φοιτητές είχαν ικανοποίηση κάτω από το μισό του μέγιστου δυνατού, γεγονός που αναδεικνύει σοβαρή απώλεια ποιότητας για μεγάλο μέρος του συνόλου. Η μέση ικανοποίηση ήταν μόλις 58.85 και η τυπική απόκλιση εξαιρετικά χαμηλή (6.71), κάτι που δείχνει πράγματι ισότητα, αλλά σε πολύ χαμηλό επίπεδο ικανοποίησης. Συνολικά, ο αλγόριθμος αυτός πέτυχε την απόλυτη ισοκατανομή, αλλά σε βάρος της αποτελεσματικότητας και της ευημερίας των φοιτητών.

4) Ο τέταρτος αλγόριθμος, **Maximize Minimum Satisfaction**, εστιάζει στη βελτίωση της χαμηλότερης εμπειρίας των φοιτητών, προσπαθώντας να διασφαλίσει ότι κανείς δεν θα έχει εξαιρετικά χαμηλή ικανοποίηση. Το αποτέλεσμά του δείχνει ότι πέτυχε τον στόχο του, αφού η ελάχιστη ικανοποίηση ήταν 100, δηλαδή αρκετά υψηλή και πάνω από το ήμισυ του μέγιστου δυνατού. Κανένας φοιτητής δεν είχε ικανοποίηση κάτω του μισού, και 18 φοιτητές έφτασαν τη μέγιστη δυνατή ικανοποίηση. Η μέση τιμή της ικανοποίησης ήταν 126.75, μια ικανοποιητική επίδοση, ενώ και η τυπική απόκλιση ήταν 23.54, ένδειξη ότι υπήρχε κάποια διακύμανση, αλλά σε αποδεκτά πλαίσια. Ο αλγόριθμος αυτός πέτυχε μια ισορροπία μεταξύ της διασφάλισης καλών συνθηκών για τους λιγότερο ικανοποιημένους και της διατήρησης ικανοποιητικής συνολικής απόδοσης.

5) Ο πέμπτος αλγόριθμος, **Maximize Minimum Satisfaction Ratio**, στοχεύει στη μεγιστοποίηση του λόγου της ελάχιστης προς τη μέγιστη δυνατή ικανοποίηση, διασφαλίζοντας ότι ακόμη και ο λιγότερο ικανοποιημένος φοιτητής βρίσκεται όσο το δυνατόν πιο κοντά στο μέγιστο επίπεδο. Ο αλγόριθμος πέτυχε εξαιρετική απόδοση, με ελάχιστη ικανοποίηση 100 και καμία περίπτωση φοιτητή κάτω από το ήμισυ της μέγιστης ικανοποίησης. Επιπλέον, 17 φοιτητές πέτυχαν το μέγιστο επίπεδο ικανοποίησης, ενώ η μέση τιμή ήταν 127.67, ελαφρώς υψηλότερη από τον προηγούμενο αλγόριθμο. Η τυπική απόκλιση ήταν 21.48, η χαμηλότερη μέχρι στιγμής για αλγόριθμο με υψηλή μέση τιμή, γεγονός που δείχνει πιο ομοιόμορφη

κατανομή της ικανοποίησης. Συνολικά, ο αλγόριθμος παρουσιάζει πολύ ισορροπημένα αποτελέσματα, με έμφαση τόσο στη δίκαιη μεταχείριση όσο και στη συνολική ποιότητα κατανομής.

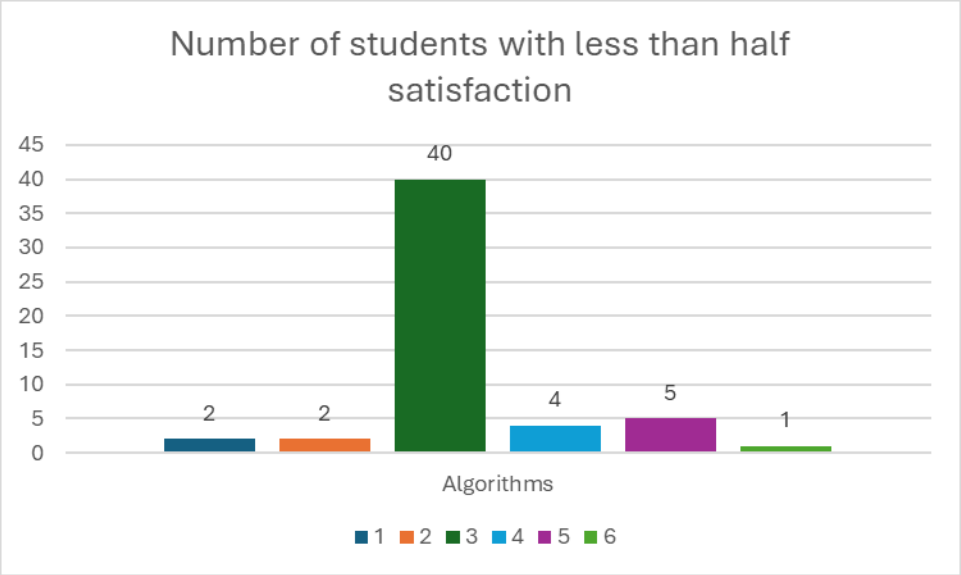
6) Ο έκτος αλγόριθμος, **Maximum Total Satisfaction and Maximum Minimum Satisfaction**, επιδιώκει να συνδυάσει ταυτόχρονα δύο κρίσιμους στόχους: τη μέγιστη συνολική ικανοποίηση και τη μέγιστη ελάχιστη ικανοποίηση. Με αυτόν τον τρόπο, εξισορροπεί την επίτευξη υψηλής απόδοσης στο σύνολο των φοιτητών, ενώ παράλληλα διασφαλίζει ότι κανείς δεν μένει πίσω. Τα αποτελέσματα το επιβεβαιώνουν, καθώς η ελάχιστη ικανοποίηση είναι 100, κανένας φοιτητής δεν έχει κάτω από τη μισή ικανοποίηση, ενώ 35 φοιτητές έφτασαν το μέγιστο επίπεδο ικανοποίησης. Η μέση ικανοποίηση είναι εξαιρετικά υψηλή (141.975) και η τυπική απόκλιση παραμένει σε λογικά επίπεδα (25.57), γεγονός που υποδηλώνει και καλή κατανομή. Ο συνδυασμός υψηλής μέσης τιμής, σημαντικού αριθμού φοιτητών στο μέγιστο και πλήρους αποφυγής χαμηλών τιμών καθιστά τον αλγόριθμο αυτό τον πιο αποτελεσματικό και ισορροπημένο από όλους. Για αυτόν τον λόγο επιλέχθηκε τελικά ως ο υφιστάμενος αλγόριθμος που υλοποιήθηκε στο σύστημα.

6.3 Συμπεράσματα Αξιολόγησης

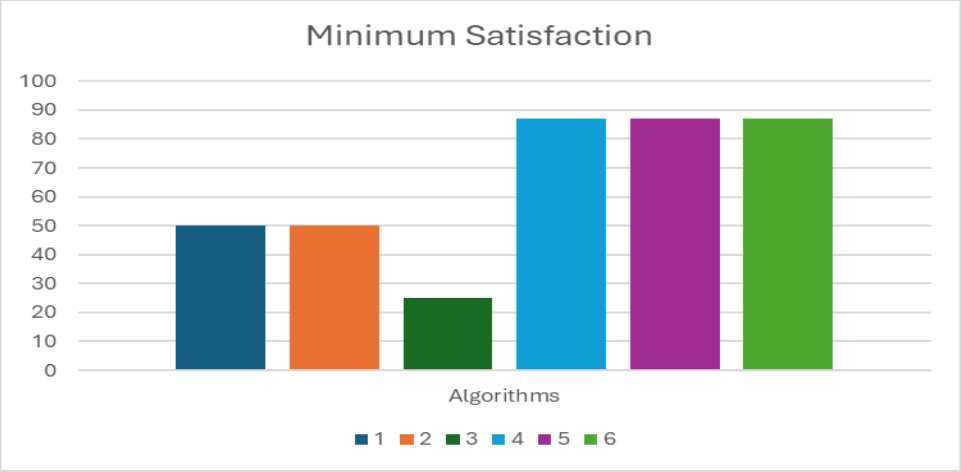
Συμπερασματικά, η αξιοποίηση του πλαισίου xUnit συνέβαλε καθοριστικά στη δομημένη και αξιόπιστη υλοποίηση ελέγχων μονάδας, ενισχύοντας τη σταθερότητα και τη λειτουργική αρτιότητα της εφαρμογής. Μέσω στοχευμένων δοκιμών στη βασική υπηρεσία επίλυσης του CSP, επιβεβαιώθηκε η ορθότητα και πληρότητα των παραγόμενων λύσεων, ενώ με τη χρήση τεχνικών mocking στους controllers επιτεύχθηκε η απομόνωση των μονάδων, διευκολύνοντας την ακριβή αξιολόγηση της συμπεριφοράς τους. Παράλληλα, η εισαγωγή μετρικών για την ποσοτική αξιολόγηση των λύσεων ενίσχυσε τον έλεγχο της ποιότητας από την πλευρά της ικανοποίησης των φοιτητών, αναδεικνύοντας την απόδοση και τη δικαιοσύνη του αλγορίθμου. Η συνδυαστική εφαρμογή ελέγχων και μετρικών ανέδειξε τη σημασία της αυτοματοποίησης και της αξιολόγησης με αντικειμενικά κριτήρια, προσφέροντας αυξημένη αξιοπιστία, ευκολότερη συντήρηση και εμπιστοσύνη στο τελικό προϊόν.

Πιο κάτω ανατίθενται οι σχετικές γραφικές αναπαραστάσεις:

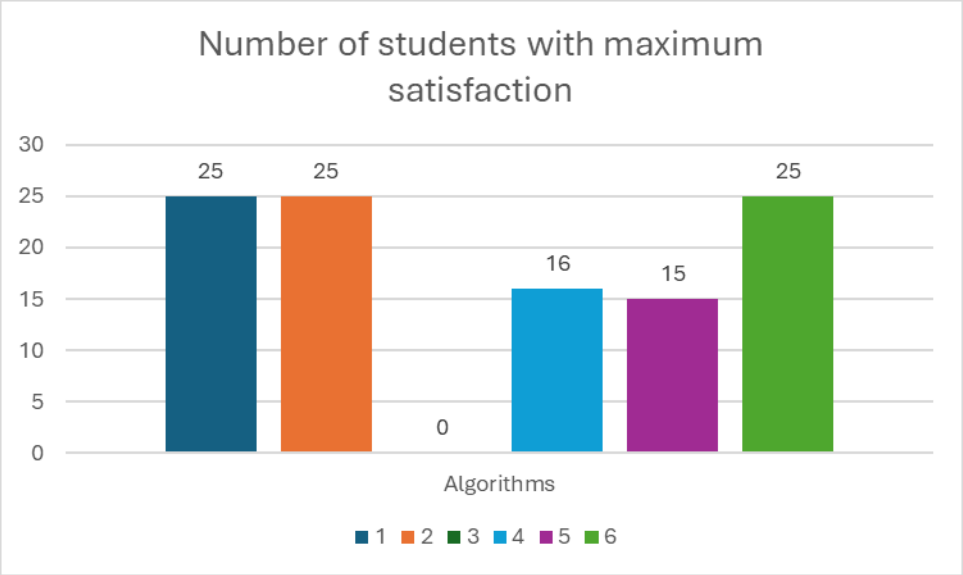
1) Η πιο κάτω γραφική παράσταση, αποδεικνύει την ανωτερότητα του αλγορίθμου 6 (δηλαδή αυτού που έχει υλοποιηθεί), αφού μόνο ένας φοιτητής είχε λιγότερη από τη μισή ικανοποίηση που μπορούσε να έχει (δηλαδή λιγότερη από 125 αφού όπως περιέγραψα προηγουμένως το πεδίο τιμών της ικανοποίησης για κάθε φοιτητή είναι 0 ... 250).



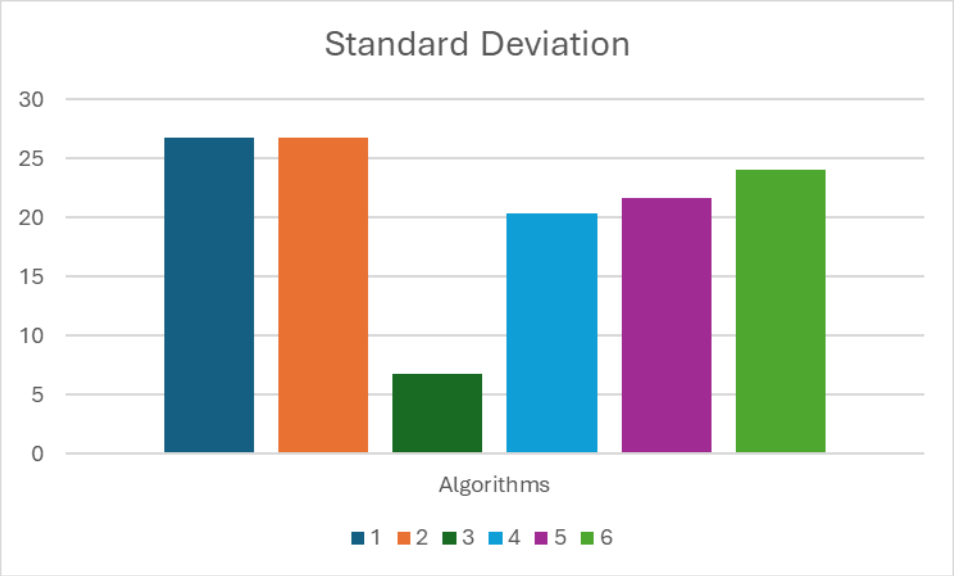
2) Στη πιο κάτω γραφική παράσταση, παρατηρούμε ότι όσον αφορά την τιμή της ελάχιστης ικανοποίησης από όλους τους φοιτητές, οι αλγόριθμοι 4,5,6 είχαν τα καλύτερα αποτελέσματα, αφού η ελάχιστη ικανοποίηση από όλους τους φοιτητές σε όλη την κατανομή μαθημάτων άγγιξε το 90.



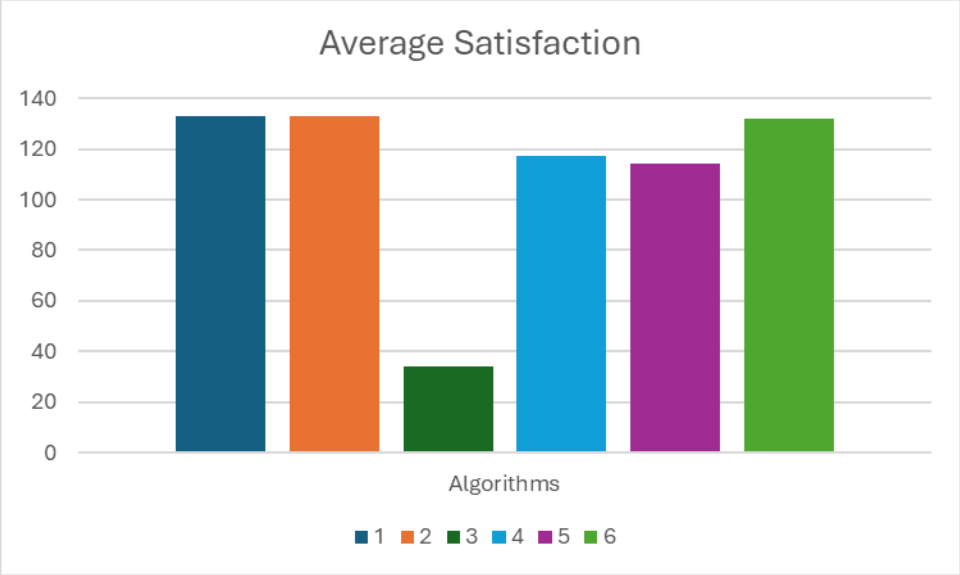
3) Στη πιο κάτω γραφική παράσταση, παρατηρούμε ότι οι αλγόριθμοι 1,2 και 6 είχαν τα καλύτερα αποτελέσματα, έχοντας το μεγαλύτερο αριθμό φοιτητών που είχαν το μέγιστο βαθμό ικανοποίησης.όταν λέμε μέγιστο βαθμό ικανοποιήσης εννοούμε ότι πήραν ακριβώς τις καλύτερες επιλογές που ήθελαν. Για παράδειγμα, φοιτητές που δήλωσαν να εγγραφούν σε ένα μάθημα πήραν τη πρώτη τους επιλογή, φοιτητές που δήλωσαν να εγγραφούν σε δύο μαθήματα πήραν τη πρώτη και δεύτερη τους επιλογή και φοιτητές που δήλωσαν να εγγραφούν σε τρία μαθήματα πήραν τη πρώτη, δεύτερη και τρίτη τους επιλογή.



4) Στη πιο κάτω γραφική παράσταση, αναπαριστάται η τυπική απόκλιση ως προς την ικανοποίηση των όλων των φοιτητών, παρατηρώντας ότι οι αλγόριθμοι 1,2 έχουν την υψηλότερη τυπική απόκλιση, ενώ ο αλγόριθμος 3 παρουσιάζει αρκετά πιο μειωμένη τυπική απόκλιση σε σχέση με τους υπόλοιπους αλγόριθμους.

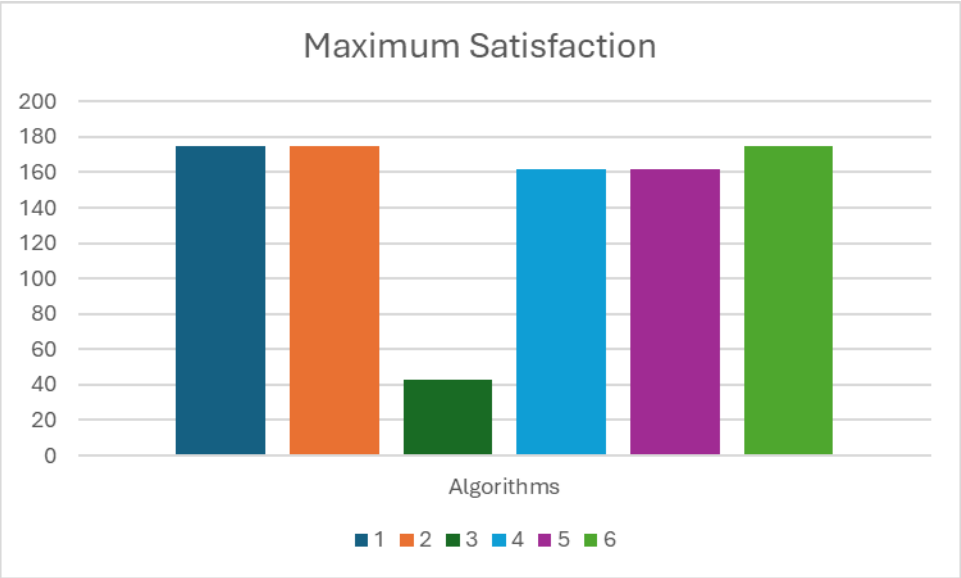


5)Στη πιο κάτω γραφική παράσταση, αναπαριστάται αναπαριστάται η μέση ικανοποίηση των όλων των φοιτητών, παρατηρόντας ότι οι αλγόριθμοι 1,2 και 6 είχαν τις ψηλότερες τιμές.



6)Η πιο κάτω γραφική παράσταση, παρουσιάζει τον μέγιστο βαθμό ικανοποίησης από όλο το σύνολο των φοιτητών για τους 6 αλγόριθμους. Παρατηρούμε ότι όλοι οι αλγόριθμοι είχαν πάρα πολύ καλά αποτελέσματα, εκτός από τον αλγόριθμο 3.Ο αλγόριθμος 3 που χρησιμοποιεί τη τυπική απόκλιση ως συνάρτηση στόχου επιδιώκει την ισορροπία στην

ικανοποίηση μεταξύ των φοιτητών, μειώνοντας τη διαφορά στις βαθμολογίες προτίμησης. Ωστόσο, αυτό δεν οδηγεί απαραίτητα στη μέγιστη συνολική ικανοποίηση, καθώς μπορεί να «θυσιάσει» υψηλές προτιμήσεις ορισμένων φοιτητών μόνο και μόνο για να εξισορροπήσει τις κατανομές. Έτσι, σε πολλές περιπτώσεις, το αποτέλεσμα είναι μια γενικά μέτρια ικανοποίηση για όλους, χωρίς να αξιοποιείται πλήρως η δυνατότητα να ικανοποιηθούν στο έπακρο οι προτιμήσεις όσο το δυνατόν περισσότερων φοιτητών.



Να σημειώσουμε ότι και οι 6 αλγόριθμοι που παρουσιάστηκαν πιο πάνω είχαν βέλτιστα αποτελέσματα ως προς τον χρόνο εκτέλεσης, αφού για όλους ήταν λιγότερο από ένα δευτερόλεπτο.

Κεφάλαιο 7

Συμπεράσματα και Μελλοντικές Εργασίες

7.1 Συμπεράσματα

Η παρούσα ατομική διπλωματική εργασία πραγματεύεται τον σχεδιασμό και την υλοποίηση μιας διαδικτυακής εφαρμογής για την υποστήριξη της διαδικασίας προεγγραφών σε μαθήματα περιορισμένης επιλογής, που απευθύνεται στους φοιτητές του Τμήματος Πληροφορικής. Το πρόβλημα της διαχείρισης προτιμήσεων και διαθέσιμων θέσεων στα μαθήματα αυτής της κατηγορίας, όταν αντιμετωπίζεται χειροκίνητα ή με μη αυτοματοποιημένες μεθόδους, δημιουργεί σημαντικό φόρτο εργασίας για το διοικητικό προσωπικό και τους συντονιστές, ενώ παράλληλα προκαλεί δυσφορία στους φοιτητές, καθώς συχνά βιώνουν έλλειψη διαφάνειας και ασυνέπειες στη διαδικασία. Η εφαρμογή που αναπτύχθηκε απαντά σε αυτές τις προκλήσεις, προσφέροντας ένα εργαλείο αυτοματοποίησης και βελτιστοποίησης της διαδικασίας εγγραφών.

Το σύστημα διαχωρίζει σαφώς τους ρόλους των χρηστών – φοιτητών και γραμματείας – προσφέροντας αντίστοιχα εξατομικευμένες λειτουργίες. Οι φοιτητές έχουν τη δυνατότητα να δηλώσουν προτιμήσεις μαθημάτων με σειρά προτεραιότητας και να επιλέξουν, αν επιθυμούν, και μάθημα διπλωματικής εργασίας. Η γραμματεία μπορεί να διαχειρίζεται τα διαθέσιμα μαθήματα, να δημοσιεύει ανακοινώσεις, να παρακολουθεί στατιστικά, να εκτελεί τον αλγόριθμο κατανομής ή να κάνει χειροκίνητες παρεμβάσεις, όταν απαιτείται. Η όλη διαδικασία καθίσταται πιο οργανωμένη, δίκαιη και αποδοτική.

Καθοριστικό στοιχείο της εφαρμογής είναι η ενσωμάτωση αλγορίθμου κατανομής που βασίζεται σε τεχνικές Προγραμματισμού Ικανοποίησης Περιορισμών (Constraint Satisfaction Programming). Ο αλγόριθμος επιχειρεί να επιτύχει τη μέγιστη ικανοποίηση των φοιτητών, διασφαλίζοντας παράλληλα τη δίκαιη κατανομή των περιορισμένων θέσεων. Η εφαρμογή αυτών των τεχνικών προσέφερε πρακτική εξάσκηση σε μεθόδους που διδάχθηκαν στο μάθημα ΕΠΛ433 και μου επέτρεψε να κατανοήσω σε βάθος τη δύναμη της αναπαράστασης προβλημάτων με περιορισμούς και της αυτόματης επίλυσής τους. Είναι ένας τομέας που με ενθουσίασε ιδιαίτερα και σκοπεύω να συνεχίσω να ασχολούμαι με αυτόν στο μέλλον, είτε σε επίπεδο ερευνητικό είτε επαγγελματικό, καθώς συνδέεται στενά και με την ανάπτυξη συστημάτων τεχνητής νοημοσύνης, η οποία με ελκύει όλο και περισσότερο.

Η εμπειρία από αυτή τη διπλωματική δεν περιορίστηκε μόνο στο τεχνικό κομμάτι. Ιδιαίτερα σημαντικό ήταν το γεγονός ότι είχα τη χαρά να συνεργαστώ στενά με τον συμφοιτητή μου, Στυλιανό, με τον οποίο λειτουργήσαμε σαν ομάδα έργου σε ένα επαγγελματικό περιβάλλον. Η συνεργασία μας ήταν άψογη και αποτελεσματική, θυμίζοντας συνθήκες πραγματικού εταιρικού project. Μέσα από την κατανομή ρόλων, την επικοινωνία, την επίλυση προβλημάτων και την κοινή λήψη αποφάσεων, αποκομίσαμε πολύτιμη εργασιακή εμπειρία που μας προετοιμάζει για την είσοδό μας στην αγορά εργασίας.

Τέλος, η εξοικείωσή μου με τεχνολογίες ανάπτυξης διαδικτυακών εφαρμογών και σχεδιασμού βάσεων δεδομένων, καθώς και η πρακτική εφαρμογή τεχνικών ικανοποίησης περιορισμών και ανάπτυξης συστημάτων με νοημοσύνη, ενίσχυσαν σημαντικά το γνωστικό μου υπόβαθρο. Οι δεξιότητες που απέκτησα και οι εμπειρίες που αποκόμισα από αυτή τη διαδικασία αποτελούν ισχυρό εφόδιο για την επαγγελματική μου πορεία.

7.2 Μελλοντικές Εργασίες

Η διαδικτυακή εφαρμογή που αναπτύχθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας αποτελεί ένα πλήρες και λειτουργικό σύστημα υποστήριξης της διαδικασίας προεγγραφών σε μαθήματα περιορισμένης επιλογής για τους φοιτητές του Τμήματος Πληροφορικής. Ωστόσο, όπως συμβαίνει με κάθε πληροφοριακό σύστημα που στοχεύει στην

πρακτική εφαρμογή του σε πραγματικά περιβάλλοντα, υπάρχουν πολλές προοπτικές βελτίωσης, επέκτασης και περαιτέρω αξιοποίησής του στο μέλλον.

Μία από τις σημαντικότερες μελλοντικές εργασίες είναι η διασύνδεση της εφαρμογής με το υφιστάμενο κεντρικό πληροφοριακό σύστημα του Πανεπιστημίου, το Banner Web. Η ενοποίηση με το Banner θα μπορούσε να επιφέρει σημαντικά πλεονεκτήματα, τόσο σε επίπεδο χρηστικότητας όσο και λειτουργικότητας. Συγκεκριμένα, όταν ολοκληρώνεται η ανάθεση των φοιτητών στα μαθήματα μέσω του αλγορίθμου της εφαρμογής μας, τα δεδομένα αυτά θα μπορούν να μεταφέρονται αυτόματα στο Banner, πραγματοποιώντας την επίσημη εγγραφή στα μαθήματα χωρίς περαιτέρω ενέργεια από πλευράς του φοιτητή ή της γραμματείας.

Αυτό σημαίνει ότι κατά την έναρξη της περιόδου των εγγραφών, κάθε φοιτητής θα είναι ήδη εγγεγραμμένος στα μαθήματα περιορισμένης επιλογής που του έχουν ανατεθεί μέσω του συστήματος, και θα του απομένει να εγγραφεί μόνο στα υποχρεωτικά του μαθήματα, σε μαθήματα ελεύθερης επιλογής, ή στη διπλωματική εργασία. Επιπλέον, στην περίπτωση που υπάρχουν διαθέσιμες θέσεις σε κάποια μαθήματα περιορισμένης επιλογής, θα μπορούσε να δίνεται η δυνατότητα στους φοιτητές να προβούν σε προσθαφαιρέσεις μαθημάτων, με δική τους ευθύνη και σύμφωνα με την προσωπική τους κρίση.

Μια ακόμη σημαντική προοπτική που καθίσταται εφικτή μέσω αυτής της ενσωμάτωσης είναι η αυτόματη επαλήθευση των προαπαιτούμενων μαθημάτων. Το σύστημα, αντλώντας στοιχεία από τη βάση δεδομένων του Banner, θα μπορεί να διαπιστώνει σε πραγματικό χρόνο αν κάθε φοιτητής πληροί τις προϋποθέσεις για να εγγραφεί σε ένα μάθημα. Αυτή η λειτουργία είναι ιδιαίτερα κρίσιμη, καθώς διασφαλίζει την ορθή εφαρμογή του προγράμματος σπουδών, μειώνει τα λάθη και ελαχιστοποιεί τις περιπτώσεις εσφαλμένων δηλώσεων που οδηγούν σε ακαδημαϊκά ή διοικητικά προβλήματα στο μέλλον.

Ένα ακόμη στοιχείο που θα μπορούσε να προστεθεί στο σύστημα είναι η υλοποίηση ενιαίας πιστοποίησης (Single Sign-On), ώστε όλοι οι χρήστες —φοιτητές, γραμματεία, συντονιστές— να εισέρχονται στην εφαρμογή με τα ιδρυματικά τους διαπιστευτήρια. Αυτό καταργεί την ανάγκη για ξεχωριστή εγγραφή και δημιουργία νέου λογαριασμού στο δικό μας σύστημα, ενισχύοντας τόσο την ασφάλεια όσο και την εμπειρία χρήσης.

Σε μελλοντικές εκδόσεις, το σύστημα θα μπορούσε να αποκτήσει και επιπλέον υποστηρικτικές λειτουργίες βασισμένες σε τεχνολογίες Τεχνητής Νοημοσύνης και Προγραμματισμού Ικανοποίησης Περιορισμών (Constraint Programming). Ενδεικτικά:

- Αυτόματη Δημιουργία Ωρολογίων Προγραμμάτων: Ανάπτυξη ενός εργαλείου που δημιουργεί προγράμματα μαθημάτων που ικανοποιούν ταυτόχρονα πλήθος περιορισμών, όπως διαθεσιμότητα διδασκόντων, σύγκρουση μαθημάτων, χρήση αιθουσών κ.ά.

- Αναφορές και Υποστήριξη Απόφασης για τη Γραμματεία: Παροχή προτάσεων για αύξηση/μείωση θέσεων σε μαθήματα με υψηλή ή χαμηλή ζήτηση, πρόβλεψη φόρτου εργασίας ανά περίοδο, και εντοπισμό «στενώσεων» στο πρόγραμμα σπουδών.

Τέλος, η εμπειρία ανάπτυξης της παρούσας εφαρμογής φανερώνει ότι η ενσωμάτωση έξυπνων τεχνικών από τον τομέα της Τεχνητής Νοημοσύνης, και ειδικότερα του Προγραμματισμού Ικανοποίησης Περιορισμών, μπορεί να βρει ουσιαστικές εφαρμογές στον ακαδημαϊκό χώρο. Η δυνατότητα έκφρασης πολύπλοκων κανόνων και προτιμήσεων σε μορφή περιορισμών, και η επίλυσή τους με μαθηματικά ακριβείς και υπολογιστικά αποδοτικές μεθόδους, μπορεί να συμβάλει ουσιαστικά στην αυτοματοποίηση και εξορθολογισμό διαδικασιών, από τις πιο απλές ως τις πιο σύνθετες.

Η ανάπτυξη τέτοιων συστημάτων στο μέλλον δεν αποτελεί μόνο τεχνική πρόκληση, αλλά και ευκαιρία μετασχηματισμού του τρόπου με τον οποίο λειτουργεί η ανώτατη εκπαίδευση, με γνώμονα την τεχνολογία, την καινοτομία και την ακαδημαϊκή ποιότητα.

Βιβλιογραφία

[1] Freeman, A., & Sanderson, P. (2021). Pro ASP.NET Core MVC 5. Apress.

[2] Esposito, D. (2020). Modern Web Development with ASP.NET Core 5. Microsoft Press.

[3] Microsoft Learn Documentation for ASP.NET Core: https://learn.microsoft.com/en-us/aspnet/core

[4] **Rossi, F., van Beek, P., & Walsh, T. (2006).** *Handbook of Constraint Programming*. Elsevier.

[5] **Guns, T. (2018).** *Essentials of Constraint Programming*. Springer.

[6] **Delaney, K. (2018).** *Inside Microsoft SQL Server 2019*. Microsoft Press.

[7] Microsoft_SQL_Server_Documentation https://learn.microsoft.com/en-us/sql/sql-server/

[8] **Price, J. (2021).** *Entity Framework Core in Action (2nd Edition)*. Manning Publications.

Παράρτημα Α

Παρατίθονται τα αποτελέσματα της έρευνας που έγινε στους φοιτητές του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου.

Ερώτηση 1)

Εικόνα 1 – Αποτελέσματα ερώτησης 1

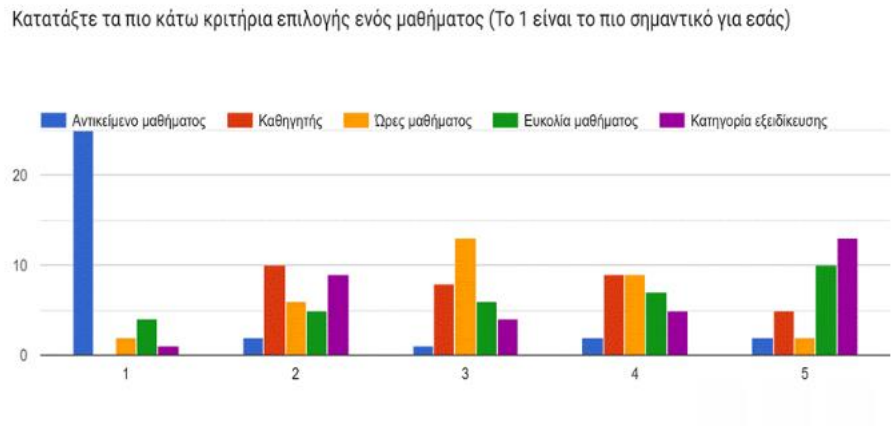
Από αυτή την ερώτηση βλέπουμε ότι οι περισσότεροι φοιτητές που παίρνουν μαθήματα περιορισμένης επιλογής επιλέγουν είτε 1 είτε 2 ενώ ελάχιστοι είναι αυτοί που θα επιλέξουν 3 μαθήματα.

Ερώτηση 2)

Εικόνα 2 – Αποτελέσματα ερώτησης 2

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

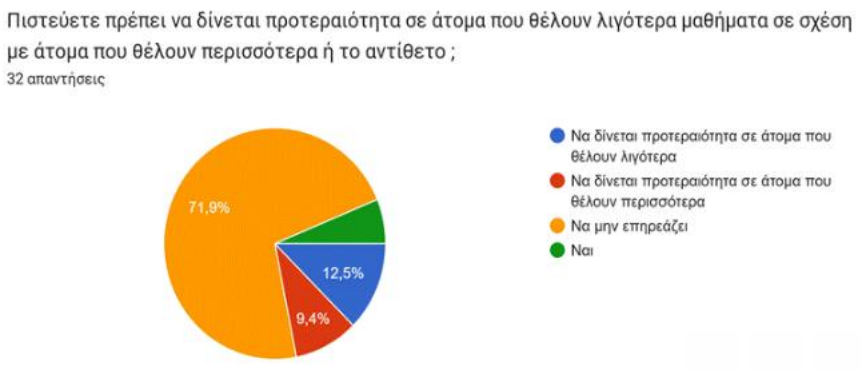
Ερώτηση 3)



Εικόνα 3 – Αποτελέσματα ερώτησης 3

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

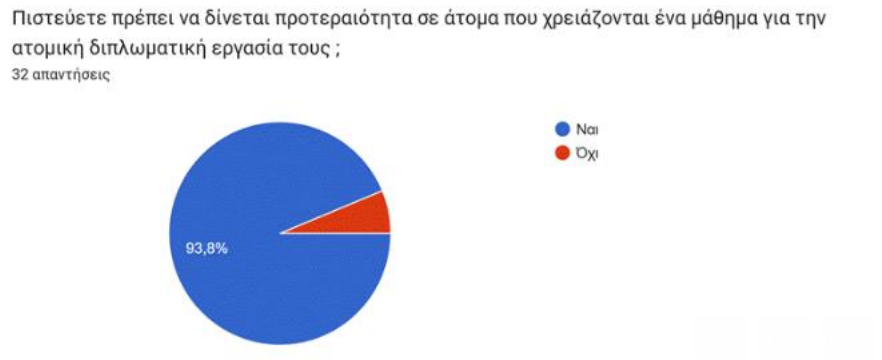
Ερώτηση 4)



Εικόνα 4 – Αποτελέσματα ερώτησης 4

Η πλειοψηφία των φοιτητών δεν θέλει ο αριθμός των μαθημάτων περιορισμένης επιλογής που θέλει ένας φοιτητής να επηρεάζει τη προτεραιότητα του.

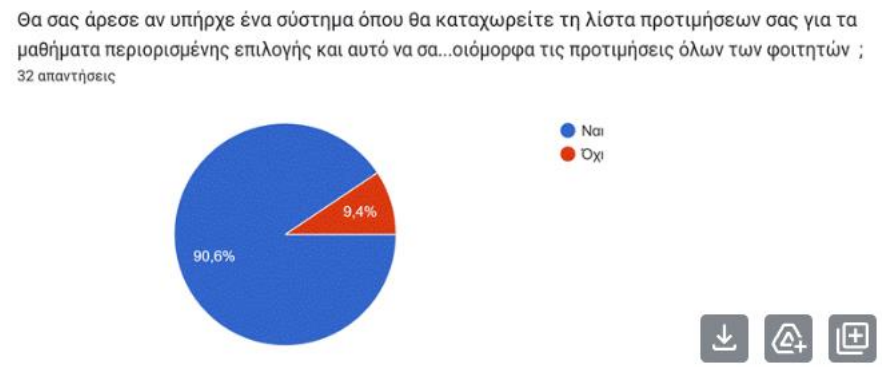
Ερώτηση 5)



Εικόνα 5 – Αποτελέσματα ερώτησης 5

Οι φοιτητές θέλουν να έχουν προτεραιότητα εάν χρειάζονται ένα μάθημα για την εκπλήρωση της διπλωματικής τους εργασίας.

Ερώτηση 6)



Εικόνα 6 – Αποτελέσματα ερώτησης 6

Οι φοιτητές θέλουν 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.”