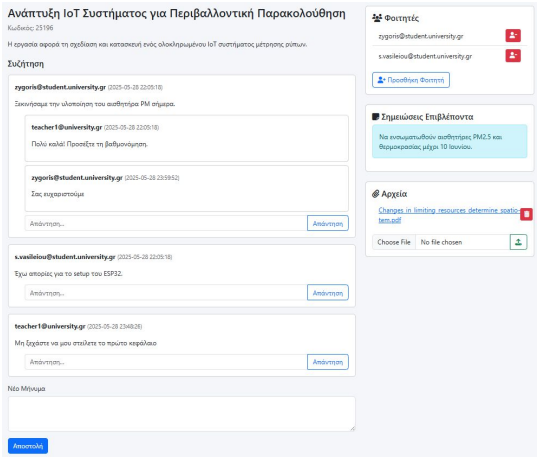
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Σύστημα διαχείρισης και παρακολούθησης εξέλιξης πτυχιακών εργασιών και επικοινωνίας»



Φοιτητής

ΚΩΝΣΤΑΝΤΙΝΟΣ ΖΥΓΟΡΗΣ

518040

Επιβλέπων

Δρ. Κυριάκος Τσιακμάκης

Ιούνιος 2025

Σύστημα διαχείρισης και παρακολούθησης εξέλιξης πτυχιακών εργασιών και επικοινωνίας

Κωδικός: 25156

Φοιτητής: Ζυγόρης Κωνσταντίνος

Εισηγητής: Δρ Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 08-03-2025

Ημερομηνία περάτωσης Π.Ε. 29-05-2025

Περίληψη

Η πτυχιακή εργασία αφορά την ανάπτυξη μιας διαδικτυακής εφαρμογής που διευκολύνει την επικοινωνία και τη συνεργασία μεταξύ φοιτητών και επιβλεπόντων καθηγητών κατά την εκπόνηση πτυχιακής εργασίας. Το σύστημα επιτρέπει στον καθηγητή να δημιουργεί πτυχιακές, να καταχωρεί φοιτητές με βάση το email τους και να διαχειρίζεται τη διαδικασία μέσω σημειώσεων, αρχείων και συνομιλιών. Ο φοιτητής με τη σειρά του έχει πρόσβαση στην πτυχιακή του και μπορεί να συμμετέχει στο σύστημα μηνυμάτων, να ανεβάζει αρχεία και να λαμβάνει καθοδήγηση. Η υλοποίηση έγινε με τη χρήση Python και Flask στο backend και το frontend σχεδιάστηκε με HTML, CSS και Bootstrap. Τα δεδομένα αποθηκεύονται σε βάση MySQL/MariaDB. Η εφαρμογή είναι απλή, λειτουργική και πλήρως εστιασμένη στις ανάγκες της ακαδημαϊκής κοινότητα και παρέχει οργανωμένη συνεργασία και βελτίωση της συνολικής διαδικασίας της πτυχιακής εργασίας.

« System for managing and monitoring the progress of thesis and communication»

Abstract

This thesis concerns the development of a web application that facilitates communication and collaboration between students and supervisors during the preparation of a thesis. The system allows the professor to create theses, register students based on their email and manage the process through notes, files and conversations. The student, in turn, has access to his thesis, can participate in the messaging system, upload files and receive guidance. The implementation was done using Python and Flask in the backend, while the frontend was designed with HTML, CSS and Bootstrap. The data is stored in a MySQL/MariaDB database. The application is simple, functional and fully focused on the needs of the academic community, offering organized collaboration and improvement of the overall thesis process.

Ευχαριστίες

Να ευχαριστήσω τους γονείς μου, τους φίλους μου και τον επιβλέποντα μου για τη βοήθεια τους σε κάθε μου βήμα και δυσκολία.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Στον χώρο της εκπαίδευσης όλο και περισσότερες διαδικασίες γίνονται ψηφιακά, οι εγγραφές, τα μαθήματα, η αξιολόγηση, η επικοινωνία με καθηγητές και συμφοιτητές. Ένα πολύ σημαντικό κομμάτι των σπουδών ειδικά στο πανεπιστήμιο είναι η πτυχιακή εργασία.

Η πτυχιακή εργασία είναι ένα τελικό μεγάλο έργο που κάθε φοιτητής καλείται να ολοκληρώσει πριν πάρει το πτυχίο του. Σε αυτό το έργο ο φοιτητής πρέπει να μελετήσει, να ερευνήσει, να σχεδιάσει ή να υλοποιήσει κάτι σχετικό με το αντικείμενο των σπουδών του. Ένας καθηγητής – επιβλέπων τον καθοδηγεί, του δίνει συμβουλές, διορθώνει κείμενα και τον στηρίζει σε όλη τη διάρκεια της εργασίας. Αυτή η συνεργασία φοιτητή και καθηγητή είναι πολύ σημαντική και διαρκεί αρκετούς μήνες.

Παρόλο που η πτυχιακή είναι ένα σοβαρό και οργανωμένο κομμάτι των σπουδών πολλές φορές δεν υπάρχει ένα εργαλείο που να βοηθά αυτή τη συνεργασία. Οι περισσότεροι φοιτητές και καθηγητές επικοινωνούν με emails, ανεβάζουν αρχεία στο Google Drive ή χρησιμοποιούν πλατφόρμες που δεν είναι φτιαγμένες ειδικά για πτυχιακές. Αυτό δημιουργεί καθυστερήσεις και προβλήματα στη συνεργασία. Δεν υπάρχει μια κοινή σελίδα να βλέπουν και οι δύο την πρόοδο, να σχολιάζουν, να ανταλλάσσουν υλικό και να οργανώνονται.

Για αυτόν τον λόγο δημιουργήθηκε αυτή η εφαρμογή. Είναι μια διαδικτυακή πλατφόρμα που επιτρέπει σε φοιτητές και καθηγητές να διαχειρίζονται μαζί την πτυχιακή εργασία. Το σύστημα αυτό είναι πολύ απλό στη χρήση του και καλύπτει όλα όσα χρειάζονται και οι δύο πλευρές. Ο καθηγητής μπορεί να δημιουργεί νέες πτυχιακές, να προσθέτει φοιτητές, να γράφει σημειώσεις και να βλέπει την εξέλιξη. Ο φοιτητής μπορεί να ανεβάζει τα κείμενα του, να συνομιλεί με τον επιβλέποντα και να παίρνει σχόλια. Όλα γίνονται μέσα από έναν κοινό χώρο με ασφάλεια και καλή οργάνωση.

Η εφαρμογή αυτή φτιάχτηκε με τη γλώσσα προγραμματισμού Python χρησιμοποιώντας το Flask framework για το πίσω μέρος (backend) και HTML/CSS/Bootstrap για το μπροστινό μέρος (frontend). Τα δεδομένα αποθηκεύονται σε μια βάση MySQL/MariaDB και η επικοινωνία των χρηστών γίνεται μέσα από το ίδιο το σύστημα – χωρίς την ανάγκη εξωτερικών εργαλείων. Κάθε χρήστης έχει δικούς του κωδικούς, και η πρόσβαση είναι ελεγχόμενη: ο φοιτητής βλέπει μόνο την πτυχιακή του και ο καθηγητής βλέπει όλες τις δικές του.

Η λύση αυτή δεν προσπαθεί να αντικαταστήσει πλατφόρμες όπως το Microsoft Teams ή το Trello. Είναι σχεδιασμένη ακριβώς για τις ανάγκες της πτυχιακής εργασίας. Είναι απλή, λειτουργική και ακριβώς στοχευμένη σε αυτό που χρειάζονται οι φοιτητές και οι καθηγητές: έναν κοινό, ασφαλή χώρο συνεργασίας. Μέσα από αυτή την εφαρμογή, η διαδικασία της πτυχιακής γίνεται πιο οργανωμένη, λιγότερο αγχωτική και περισσότερο διαφανής.

Η εργασία αυτή υλοποιήθηκε με σκοπό να δώσει λύση σε ένα πραγματικό πρόβλημα που υπάρχει σε πολλά πανεπιστημιακά τμήματα. Μελλοντικά μπορεί να εξελιχθεί ακόμα περισσότερο, προσθέτοντας νέες δυνατότητες όπως είναι αξιολόγηση, χρονοδιαγράμματα. Δημιουργήθηκε ένα απλό και αποτελεσματικό εργαλείο που φέρνει τον φοιτητή και τον καθηγητή πιο κοντά για την επικοινωνία και την πρόοδο της πτυχιακής.

Το Κεφάλαιο 2 αναφέρεται σε "Συστήματα επικοινωνίας μεταξύ φοιτητών και καθηγητών για την πτυχιακή τους", και περιλαμβάνει τρεις υποενότητες: την 2.1 που παρουσιάζει το Asana, την 2.2 για το Trello, και την 2.3 με το εργαλείο Microsoft Teams.

Στο Κεφάλαιο 3 εξετάζεται η τεχνολογική υποδομή που χρησιμοποιήθηκε στην παρούσα εργασία, με τίτλο "Flask+Python+MySQL". Περιλαμβάνει τις ενότητες 3.1 "Γνωριμία με την Python", 3.2 "Γνωριμία με την Flask", και 3.3 "Η MySQL/MariaDB".

Ακολουθεί η ενότητα 4.1 με τίτλο "Bootstrap/CSS/jquery", η οποία αν και δεν έχει αριθμηθεί ως ξεχωριστό κεφάλαιο, αποτελεί αναπόσπαστο μέρος της τεχνικής περιγραφής του frontend της εφαρμογής.

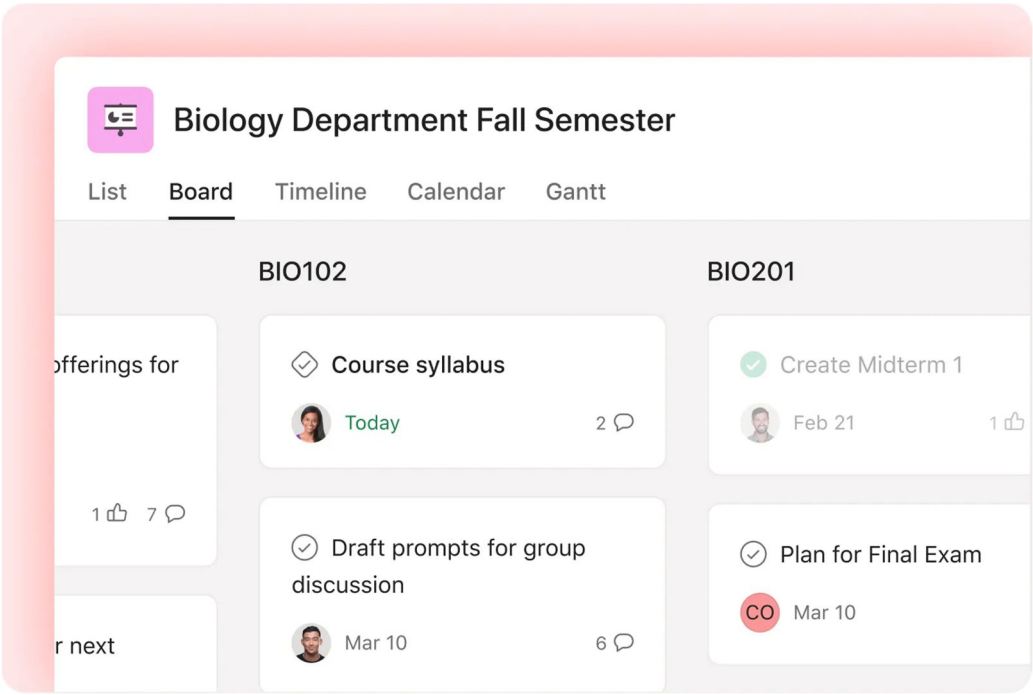
Το Κεφάλαιο 5 φέρει τον τίτλο "Σύστημα Διαχείρισης και Παρακολούθησης Εξέλιξης Πτυχιακών Εργασιών και Επικοινωνίας" και αναλύει την υλοποίηση της εφαρμογής. Περιλαμβάνει την ενότητα 5.1 "Οι ιστοσελίδες της πλατφόρμας", την 5.2 "Περιγραφή του συστήματος μέσω διαγραμμάτων", την 5.3 "Ορισμένες σημαντικές μέθοδοι", και την 5.4 "Λίγα λόγια για την βάση μας".

Το Κεφάλαιο 6 παρουσιάζει τα "Συμπεράσματα – κάποιες βελτιώσεις", ενώ στο τέλος της εργασίας περιλαμβάνονται η Βιβλιογραφία, καθώς και το Παράρτημα Κώδικα Α.

Κεφάλαιο 2ο: Συστήματα επικοινωνίας μεταξύ φοιτητών και καθηγητών για την πτυχιακή τους

2.1 Asana

Η Asana αποτελεί μία από τις πιο διαδεδομένες διαδικτυακές πλατφόρμες διαχείρισης έργων στον χώρο της τεχνολογίας, των επιχειρήσεων και τα τελευταία χρόνια και της εκπαίδευσης. Με την έκδοση Asana for Education, η πλατφόρμα έχει προσαρμοστεί ώστε να εξυπηρετεί τις ανάγκες φοιτητών, καθηγητών και εκπαιδευτικών ιδρυμάτων και είναι ένα ευέλικτο και δυναμικό περιβάλλον συνεργασίας και παρακολούθησης έργων όπως είναι οι πτυχιακές εργασίες [1].



Εικόνα 2.1: Asana for education

[https://assets.asana.biz/transform/e299def3-6bd0-4859-aba3-53cacd85fa51/web-education-FY25-use-case-2-2x-En-us]

Η βασική φιλοσοφία του Asana βασίζεται στην έννοια των έργων (projects), των εργασιών (tasks) και των χρηστών (members) που συνεργάζονται σε αυτά. Στο εκπαιδευτικό χώρο ένα έργο μπορεί να είναι μια πτυχιακή εργασία ή μία συνεργατική ερευνητική διαδικασία όπου φοιτητές και επιβλέποντες συνεργάζονται μέσα από μια κοινή διαδραστική επιφάνεια. Η πλατφόρμα επιτρέπει τη δημιουργία εργασιών με τίτλο, περιγραφή, υπεύθυνο, ημερομηνία παράδοσης, ετικέτες και υπο-εργασίες. Οι φοιτητές μπορούν να αναλαμβάνουν συγκεκριμένα μέρη της πτυχιακής (όπως αναζητήσεις βιβλιογραφίας, ανάπτυξη κώδικα, συγγραφή κεφαλαίων) και ο επιβλέπων έχει τη δυνατότητα να

παρακολουθεί την πρόοδο, να προσθέτει σχόλια ή να επαναπροσδιορίζει προτεραιότητες και προθεσμίες.

Ένα από τα βασικά πλεονεκτήματα του Asana είναι το ευχάριστο και μοντέρνο γραφικό περιβάλλον, το οποίο βοηθάει στην καθημερινή χρήση και μειώνει την αίσθηση βαριάς εργασίας. Μέσα από το γραφικό του περιβάλλον οι χρήστες μπορούν να βλέπουν την πρόοδο με μορφή λίστας, ημερολογίου επιλέγοντας τον τρόπο οργάνωσης που του ταιριάζει καλύτερα. Κάθε εργασία επιτρέπει επισύναψη αρχείων, σχόλια και εσωτερικές ειδοποιήσεις, ώστε να διατηρείται όλο το ιστορικό συνεργασίας συγκεντρωμένο σε ένα σημείο.

Η έκδοση Asana for Education προσφέρει ειδικές εκπτώσεις ή δωρεάν πρόσβαση σε φοιτητές και εκπαιδευτικά ιδρύματα, ενώ υποστηρίζει πλήρως τη συνεργασία σε πραγματικό χρόνο. Η ασύγχρονη επικοινωνία επιτρέπει στους φοιτητές να εργάζονται με ευελιξία, ενημερώνοντας την πρόοδο τους και ανταλλάσσοντας αρχεία ή σχόλια με τον επιβλέποντα χωρίς την ανάγκη φυσικής παρουσίας ή ταυτόχρονου online χρόνου.

Σημαντικό στοιχείο είναι και η δυνατότητα ενσωμάτωσης με άλλα εργαλεία όπως το Google Drive, το Microsoft Teams, το Slack ή το Zoom, επιτρέποντας την ομαλή ένταξη του Asana σε υπάρχοντα ψηφιακά πανεπιστημιακά ιδρυμάτα. Οι φοιτητές μπορούν να συνδέουν αρχεία της εργασίας τους να προγραμματίζουν παρουσιάσεις ή ακόμη και να ενσωματώνουν checkpoints για την προετοιμασία της τελικής παράδοσης.

Από θέμα ασφάλειας η Asana παρέχει ισχυρή προστασία δεδομένων, κάποια εφεδρικά αντίγραφα και έλεγχο πρόσβασης ανά χρήστη. Ο επιβλέπων μπορεί να διαχειρίζεται τα δικαιώματα κάθε φοιτητή και η όλη δραστηριότητα καταγράφεται σε ροές ενεργειών για να υπάρχει διαφάνεια στην κατανομή ευθυνών.

Οι φοιτητές μαθαίνουν να τηρούν προθεσμίες, να αναλύουν το έργο τους σε διαχειρίσιμα τμήματα και να συνεργάζονται με σαφήνεια και συνέπεια. Οι καθηγητές αποκτούν πλήρη εικόνα της προόδου εντοπίζουν καθυστερήσεις ή προβλήματα και προσφέρουν στοχευμένη καθοδήγηση.

Η πλατφόρμα Asana for Education αποτελεί ένα σύγχρονο και λειτουργικό εργαλείο που μπορεί να ενισχύσει σημαντικά την οργάνωση, την επικοινωνία και τη συνολική ποιότητα της διαδικασίας εκπόνησης μιας πτυχιακής εργασίας. Ο συνδυασμός διαφάνειας, ευχρηστίας και δυνατότητας παρακολούθησης κάνουν το Asana έναν ισχυρό υποστηρικτή της ακαδημαϊκής συνεργασίας στην εποχή της ψηφιακής εκπαίδευσης.

Αν και το Asana for Education αποτελεί μια εξελιγμένη λύση διαχείρισης έργων με γενική εφαρμογή και εκτενή χαρακτηριστικά, το σύστημα που αναπτύξαμε επικεντρώνεται αποκλειστικά στην εκπόνηση πτυχιακών εργασιών σε ακαδημαϊκό πλαίσιο. Σε αντίθεση με το Asana, το οποίο βασίζεται σε γενικά tasks και templates, η δική μας πλατφόρμα έχει σχεδιαστεί ειδικά με επίκεντρο τη συνεργασία

επιβλέποντα-φοιτητή, ενσωματώνοντας με απόλυτη απλότητα και σαφήνεια όλα τα στοιχεία που σχετίζονται με μια πτυχιακή: τίτλος, περιγραφή, σημειώσεις επιβλέποντα, συνδεδεμένοι φοιτητές, σύστημα ανταλλαγής μηνυμάτων και αρχεία.

Σε επίπεδο λειτουργικότητας, το σύστημα που αναπτύξαμε προσφέρει ένα στοχευμένο και δομημένο περιβάλλον επικοινωνίας, χωρίς να απαιτεί εκπαίδευση ή προσαρμογή σε γενικευμένα εργαλεία διαχείρισης έργων. Ο επιβλέπων έχει αποκλειστικό έλεγχο σε κάθε πτυχιακή, προσθέτει φοιτητές με βάση το email, διαχειρίζεται σημειώσεις ορατές μόνο προς τα μέλη και μπορεί να καθοδηγεί τη διαδικασία με σχόλια, απαντήσεις και ενημερώσεις αρχείων. Παράλληλα ο φοιτητής έχει περιορισμένο αλλά ουσιαστικό ρόλο και με δυνατότητα να συμμετέχει στη συζήτηση και να καταθέτει υλικό-αρχεία.

Το Asana προϋποθέτει προσαρμογή από τον χρήστη και βασίζεται σε abstract μοντέλα εργασιών και δεν διαθέτει ενσωματωμένη την λογική της πτυχιακής. Το δικό μας σύστημα είναι αυτοδύναμο και ανεξάρτητο από τρίτους παρόχους και μπορεί να φιλοξενηθεί εσωτερικά από κάθε τμήμα πανεπιστημίου. Εστιάζει στην ασφάλεια και στην απλότητα με βασικές δυνατότητες χωρίς την ανάγκη ολοκλήρωσης με εξωτερικές υπηρεσίες ή API.

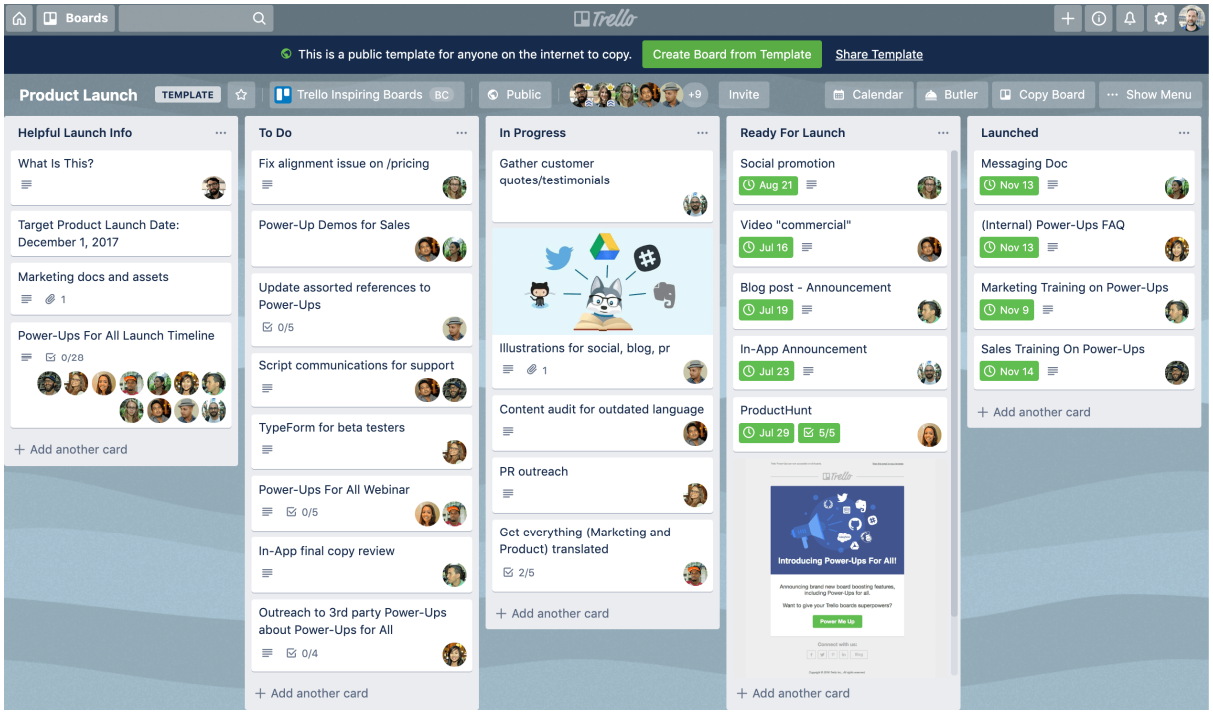
Η δική μας πλατφόρμα δεν πάει να αντικαταστήσει εμπορικές λύσεις όπως το Asana, αλλά λειτουργεί ως μια ειδικά σχεδιασμένη ακαδημαϊκή εφαρμογή η οποία ενσωματώνει ό,τι είναι απαραίτητο για την αποτελεσματική παρακολούθηση, οργάνωση και υποστήριξη των πτυχιακών εργασιών χωρίς περιττή πολυπλοκότητα. Το μεγάλο της πλεονέκτημα είναι ότι γνωρίζει ακριβώς το κοινό που εξυπηρετεί: φοιτητές και καθηγητές με στόχο μια συγκεκριμένη διαδικασία και χρονικό ορίζοντα.

2.2 Trello

Το Trello αποτελεί ένα από τα πιο διαδεδομένα εργαλεία οργάνωσης και διαχείρισης έργων με βάση τη φιλοσοφία Kanban. Αναπτύχθηκε αρχικά για επιχειρησιακή χρήση, ωστόσο έχει βρει ευρεία εφαρμογή και στον εκπαιδευτικό χώρο και προσφέρει έναν απλό και ευέλικτο τρόπο παρακολούθησης της προόδου σε ομαδικές ή ατομικές εργασίες, όπως οι πτυχιακές. Η διεπαφή του Trello βασίζεται σε πίνακες (boards), λίστες (lists) και κάρτες (cards), που επιτρέπουν στους χρήστες να οργανώνουν εργασίες, να προσθέτουν σχόλια, αρχεία, ετικέτες, ημερομηνίες και μέλη, δημιουργώντας έτσι ένα δυναμικό περιβάλλον διαχείρισης [2].

Το Trello μπορεί να χρησιμοποιηθεί για την οργάνωση μιας πτυχιακής εργασίας ως εξής: ο επιβλέπων δημιουργεί έναν πίνακα με τίτλο π.χ. "Πτυχιακή – Μαρία Παπαδοπούλου" και προσκαλεί τον φοιτητή ως μέλος. Οι λίστες μπορούν να αναπαριστούν στάδια της διαδικασίας (π.χ. "Έρευνα", "Συγγραφή", "Διόρθωση", "Υποβολή"), και κάθε εργασία μπορεί να εισαχθεί ως κάρτα, με λεπτομέρειες, προθεσμία,

και συνημμένα αρχεία. Με αυτόν τον τρόπο ο επιβλέπων παρακολουθεί την πρόοδο και ο φοιτητής έχει πάντα ξεκάθαρη εικόνα για το τι απαιτείται να ολοκληρωθεί.



Εικόνα 2.2: Trello

[https://atlassianblog.wpengine.com/wp-content/uploads/2024/01/productlaunch.png]

Ένα από τα μεγαλύτερα πλεονεκτήματα του Trello είναι η αμεσότητα και η απλότητά του. Χωρίς να απαιτεί προηγούμενη τεχνική γνώση, προσφέρει ένα οπτικά ξεκάθαρο περιβάλλον, φιλικό σε χρήστες όλων των επιπέδων. Παράλληλα, υποστηρίζει σχόλια, εσωτερικές ειδοποιήσεις και δυνατότητα επισύναψης αρχείων. Η διασύνδεσή του με εργαλεία όπως Google Drive, Dropbox και Slack προσθέτει λειτουργικότητα σε πιο σύνθετα σενάρια.

Το Trello ως γενικής χρήσης εργαλείο δεν έχει δομές ειδικά προσαρμοσμένες στην ακαδημαϊκή πραγματικότητα. Δεν υπάρχει έννοια πτυχιακής ή φοιτητικού ρόλου και όλες οι λειτουργίες χρειάζονται διαμόρφωση από τον χρήστη. Η ευελιξία που προσφέρει μπορεί να μετατραπεί σε δυσκολία για όσους αναζητούν συγκεκριμένα πρότυπα και αυτοματισμούς για πτυχιακές εργασίες.

Παρά τις δυνατότητές του το Trello ενδείκνυται κυρίως για φοιτητές που διαθέτουν οργανωτικές δεξιότητες και επιθυμούν να διαχειριστούν οι ίδιοι τη δομή του έργου τους. Για τους επιβλέποντες απαιτεί έξτρα προσπάθεια για να ορίσουν διαδικασίες, ειδοποιήσεις και ροή εργασίας και η πλατφόρμα δεν προσφέρει προκαθορισμένα templates για ακαδημαϊκή χρήση.

Το σύστημα που αναπτύξαμε είναι πλήρως προσανατολισμένο στη διαχείριση πτυχιακών εργασιών. Ενσωματώνει τις έννοιες της πτυχιακής, του φοιτητή, του επιβλέποντα και των σχετικών λειτουργιών,

χωρίς να απαιτεί παραμετροποίηση ή δημιουργία custom δομών από τον χρήστη. Όλα έχουν σαφείς ρόλους και περιορισμούς. Ο καθηγητής δημιουργεί την πτυχιακή, προσθέτει φοιτητές, γράφει σημειώσεις, σχολιάζει και διαχειρίζεται αρχεία. Ο φοιτητής συμμετέχει μόνο στις πτυχιακές που του έχουν ανατεθεί και περιορίζεται σε αποστολή αρχείων και μηνυμάτων.

Το Trello δεν έχει έννοιες όπως οριστικοποίηση, ελεγχόμενη πρόσβαση, καταγραφή ιστορικού, οργανωμένη συζήτηση σε μορφή νήματος ή οπτική παρουσίαση πτυχιακής με πληροφορίες και σχετικούς φοιτητές. Το δικό μας σύστημα επιτρέπει αποθήκευση αρχείων στον διακομιστή, απευθείας εμφάνιση μηνυμάτων σε ιεραρχική δομή και πλήρη σύνδεση με τον πίνακα χρηστών.

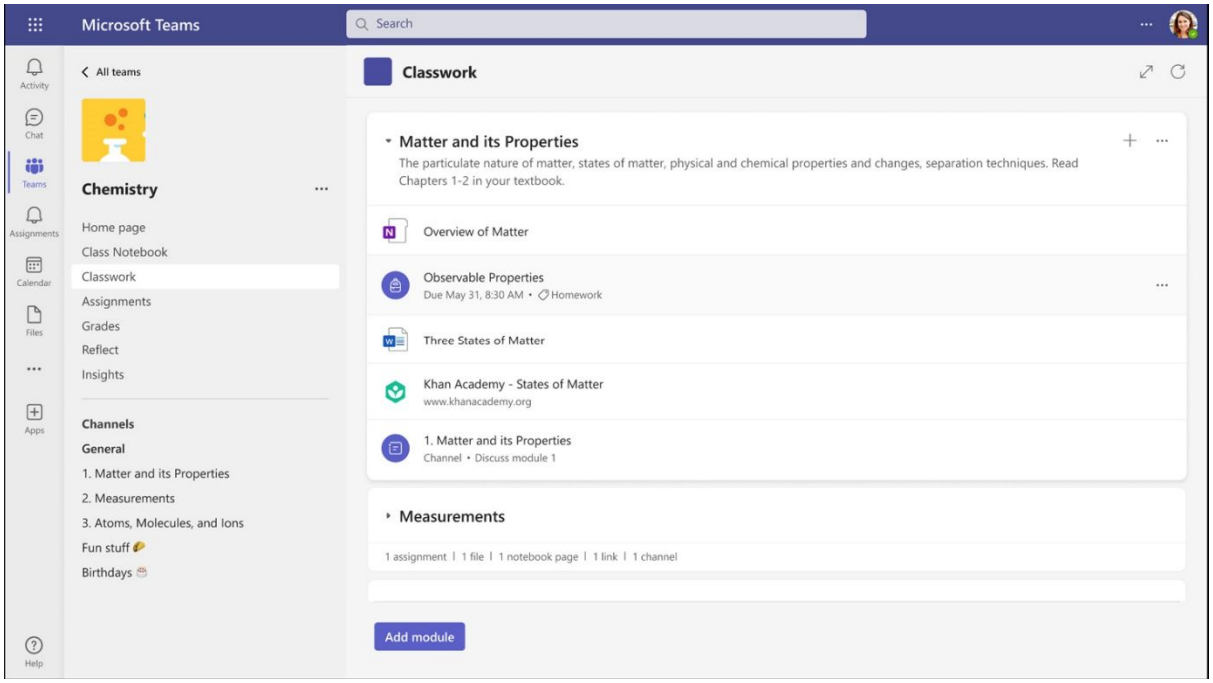
Ένα ακόμη βασικό πλεονέκτημα είναι η αποκλειστική φιλοξενία και διαχείριση από το τμήμα ή το ίδρυμα, που προσφέρει ανεξαρτησία από εξωτερικούς παρόχους, δυνατότητα ενσωμάτωσης σε εσωτερικά συστήματα (όπως ιδρυματικοί λογαριασμοί ή Moodle), και δυνατότητα επέκτασης του συστήματος με βάση τις ανάγκες του εκάστοτε τμήματος.

Η λύση που υλοποιήσαμε υπερτερεί σε προσαρμοστικότητα και καταλληλότητα για το εκπαιδευτικό πλαίσιο της πτυχιακής εργασίας. Δεν απαιτεί μάθηση ή δοκιμές και προσφέρει έτοιμη υποστήριξη για την ακριβή διαδικασία που υπηρετεί.

2.3 Microsoft Teams

Το Microsoft Teams αποτελεί μια από τις πληρέστερες και πιο διαδεδομένες πλατφόρμες επικοινωνίας και συνεργασίας, η οποία έχει υιοθετηθεί σε τεράστια κλίμακα από εκπαιδευτικά ιδρύματα σε όλο τον κόσμο. Ενσωματωμένη πλήρως στο οικοσύστημα του Microsoft 365, προσφέρει ένα σύνολο εργαλείων που καλύπτουν σχεδόν κάθε πτυχή της εκπαιδευτικής διαδικασίας: από απλές συνομιλίες μέχρι κοινή επεξεργασία εγγράφων, διοργάνωση διαδικτυακών μαθημάτων και συνεργασία σε πραγματικό χρόνο [3].

Στο πλαίσιο πτυχιακών εργασιών, το Teams μπορεί να αξιοποιηθεί ως μια ολοκληρωμένη πλατφόρμα για την επικοινωνία μεταξύ επιβλεπόντων και φοιτητών. Κάθε πτυχιακή εργασία μπορεί να αντιστοιχεί σε μια «Ομάδα» (Team) ή ένα ιδιωτικό «Κανάλι» (Channel) μέσα σε μια ομάδα του μαθήματος ή του τμήματος. Μέσα από αυτήν τη δομή, ο φοιτητής και ο καθηγητής μπορούν να συνομιλούν, να ανταλλάσσουν αρχεία, να καταγράφουν σημειώσεις, να δημιουργούν κοινά ημερολόγια και να οργανώνουν συναντήσεις εξ αποστάσεως μέσω βιντεοκλήσεων.



Εικόνα 2.3: Microsoft Teams education

[https://www.microsoft.com/en-us/education/blog/wp-content/uploads/2024/05/Classwork-Hero_web.jpg]

Η ισχύς του Microsoft Teams προέρχεται κυρίως από την ενοποίησή του με τα υπόλοιπα εργαλεία της Microsoft, όπως το Word, Excel, PowerPoint, OneDrive και OneNote. Ο φοιτητής μπορεί να ανεβάσει το κείμενο της πτυχιακής του σε μορφή Word, να το επεξεργαστεί μαζί με τον καθηγητή σε πραγματικό χρόνο και να λαμβάνει σχόλια και διορθώσεις χωρίς την ανάγκη αποστολής email ή συνημμένων. Παράλληλα, μπορεί να προγραμματίσει εβδομαδιαίες συναντήσεις με τον επιβλέποντα μέσω Teams Meetings και να κρατά σημειώσεις στον κοινόχρηστο χώρο OneNote Class Notebook.

Η πλατφόρμα παρέχει ασφάλεια, έλεγχο πρόσβασης και καταγραφή δραστηριοτήτων. Οι καθηγητές έχουν τη δυνατότητα να ελέγχουν ποιοι έχουν πρόσβαση στο υλικό, να περιορίζουν δυνατότητες (π.χ. επεξεργασία ή διαγραφή μηνυμάτων) και να ορίζουν σαφείς ρόλους (ιδιοκτήτης, μέλος, προσκεκλημένος). Το Teams υποστηρίζει ειδοποιήσεις, υπενθυμίσεις και ροές εργασιών μέσω Power Automate, παρέχοντας τη δυνατότητα αυτοματοποίησης απλών διαδικασιών.

Παρά τη δύναμη και τον πλούτο δυνατοτήτων του, το Microsoft Teams δεν είναι εξαρχής δομημένο για να εξυπηρετήσει αποκλειστικά πτυχιακές εργασίες. Απαιτεί παραμετροποίηση, καλή γνώση της πλατφόρμας και εξοικείωση με τις λειτουργίες των καναλιών, αρχείων και αδειών. Η ύπαρξη πολλών εργαλείων μέσα σε ένα περιβάλλον ενδέχεται να προκαλέσει σύγχυση σε λιγότερο έμπειρους φοιτητές ή καθηγητές, ενώ η βασική εμπειρία χρήσης εξαρτάται άμεσα από τη διαθεσιμότητα του Microsoft 365 μέσω του ιδρύματος.

Σε σύγκριση με το Microsoft Teams το δικό μας σύστημα εστιάζει αποκλειστικά στη συνεργασία για πτυχιακές εργασίες, χωρίς την ανάγκη για παραμετροποίηση ή εκπαίδευση. Η διεπαφή του είναι απλή και προσανατολισμένη στη διαδικασία και έχει προβολή τίτλου, περιγραφής, σημειώσεων, αποστολή αρχείων και συζήτηση. Οι χρήστες δεν χρειάζεται να διαχειριστούν ομάδες, ρυθμίσεις καναλιών ή εξωτερικές εφαρμογές. Κάθε πτυχιακή είναι μια αυτόνομη μονάδα με δικό της χώρο και λειτουργίες.

Το σύστημα μας είναι ανοικτού κώδικα και φιλοξενούμενο τοπικά με μεγαλύτερη ανεξαρτησία και προσαρμοστικότητα. Δεν απαιτεί άδεια χρήσης και δεν εξαρτάται από εμπορικές συμφωνίες ή εξωτερικούς servers, και μπορεί να επεκταθεί ή να τροποποιηθεί εσωτερικά ανάλογα με τις ανάγκες του τμήματος. Εστιάζει στην ασφάλεια των δεδομένων και στη σαφήνεια των ρόλων: οι φοιτητές μπορούν να επικοινωνούν μόνο εντός της πτυχιακής τους και οι καθηγητές ελέγχουν την πρόσβαση και την τροποποίηση περιεχομένου.

Το Microsoft Teams παρέχει μεγαλύτερη ευελιξία με περισσότερες δυνατότητες συνεργασίας και υψηλό επίπεδο υποδομής αλλά μπορεί να ξεπερνάει τις ανάγκες μιας απλής πτυχιακής και να εισάγει περιττή πολυπλοκότητα.

Κεφάλαιο 3ο: Flask+Python+MySQL

3.1 Γνωριμία με την Python

Η Python είναι μία από τις πιο δημοφιλείς, δυναμικές και ευέλικτες γλώσσες προγραμματισμού της σύγχρονης εποχής. Από την αρχική της δημιουργία στα τέλη της δεκαετίας του 1980 από τον Guido van Rossum, η Python εξελίχθηκε σε μια γλώσσα γενικής χρήσης που βρίσκει εφαρμογή σε ένα τεράστιο εύρος τομέων: από την ανάπτυξη εφαρμογών web και desktop μέχρι την επιστήμη δεδομένων, τη μηχανική μάθηση, τη ρομποτική και τον αυτοματισμό. Η φιλοσοφία της βασίζεται στην καθαρή και σύνταξη, την απλότητα και τη δυνατότητα ταχείας ανάπτυξης λογισμικού [4-6].

Η επιλογή της Python για την ανάπτυξη του συστήματος διαχείρισης πτυχιακών εργασιών δεν ήταν τυχαία. Πρόκειται για μια γλώσσα που προσφέρει έναν ιδανικό συνδυασμό παραγωγικότητας και σαφήνειας, στοιχεία απαραίτητα για την υλοποίηση μιας διαδικτυακής εφαρμογής που απαιτεί αλληλεπίδραση με βάση δεδομένων, διαχείριση χρηστών, μηχανισμούς ελέγχου ρόλων και επικοινωνία σε πραγματικό χρόνο. Η Python, με την υποστήριξη του ευρέως οικοσυστήματός της και επιτρέπει στον προγραμματιστή να συγκεντρώνεται περισσότερο στη λογική της εφαρμογής και λιγότερο σε τεχνικές λεπτομέρειες χαμηλού επιπέδου.

Ένα από τα μεγαλύτερα πλεονεκτήματα της Python είναι η τεράστια βιβλιοθήκη έτοιμων πακέτων και εργαλείων. Ειδικά για ανάπτυξη web εφαρμογών, η Python προσφέρει μια σειρά από frameworks, με πιο γνωστά το Django και το Flask. Στην παρούσα εργασία επιλέχθηκε το Flask λόγω της απλότητάς του, της ευκολίας ενσωμάτωσης με HTML πρότυπα και της ευελιξίας στη διαμόρφωση των διαδρομών (routes) και των ροών εργασίας. Οι βιβλιοθήκες όπως το sqlite3 ή mysql.connector επιτρέπουν την άμεση επικοινωνία με σχεσιακές βάσεις δεδομένων, ενώ το werkzeug.security προσφέρει έτοιμες και ασφαλείς μεθόδους για τη διαχείριση και αποθήκευση κωδικών χρηστών.

Σημαντική είναι και η απλότητα της Python στον τομέα της διαχείρισης δεδομένων. Η δυνατότητα χρήσης δομών όπως dictionaries και lists, η ευκολία χρήσης συναρτήσεων υψηλού επιπέδου και η κατανοητή σύνταξη καθιστούν την Python ιδανική για εφαρμογές όπου η λογική ροής είναι πιο σημαντική από την απόδοση σε επίπεδο CPU. Αυτό είναι ιδιαίτερα χρήσιμο σε εφαρμογές όπως η παρούσα, όπου το ζητούμενο είναι η σταθερότητα, η συντηρησιμότητα και η γρήγορη εξέλιξη χωρίς την ανάγκη βελτιστοποίησης κάθε υπολογιστικού κύκλου.

Ένα ακόμη στοιχείο που καθιστά την Python προτιμητέα είναι η τεράστια κοινότητα χρηστών και προγραμματιστών. Αυτό σημαίνει ότι κάθε πρόβλημα ή ανάγκη αντιμετωπίζεται σχεδόν πάντα από κάποιο υπάρχον παράδειγμα ή βιβλιοθήκη. Η τεκμηρίωση είναι πλούσια, και η δυνατότητα εύκολης

ενσωμάτωσης τρίτων εργαλείων (όπως υπηρεσίες email, APIs, authentication tokens) απλοποιεί την ανάπτυξη σύγχρονων και ολοκληρωμένων εφαρμογών.

Στην πλατφόρμα πτυχιακών που αναπτύχθηκε η Python διαχειρίζεται όλη τη λειτουργία της λογικής του συστήματος από την είσοδο χρηστών και την επαλήθευση στοιχείων μέχρι την αποθήκευση αρχείων, την προβολή πτυχιακών, την ανταλλαγή μηνυμάτων και τη διαχείριση δικαιωμάτων. Η χρήση sessions για την αναγνώριση του χρήστη, τα hash στους κωδικούς και η χρήση templates σε συνδυασμό με routes οδηγεί σε μια αρθρωτή, καθαρή και επεκτάσιμη αρχιτεκτονική, ιδανική για μελλοντική συντήρηση και ανάπτυξη.

Η Python αποδεικνύεται όχι μόνο κατάλληλη για εκπαιδευτικά έργα αλλά και εξαιρετικά αποτελεσματική για την υλοποίηση πραγματικών εφαρμογών που εξυπηρετούν ουσιαστικές ανάγκες σε εκπαιδευτικά περιβάλλοντα. Ο συνδυασμός καθαρής σύνταξης, ισχυρών βιβλιοθηκών, άμεσης εκτελέσιμης λογικής και πλήρους ευελιξίας στη σχεδίαση κάνει τη γλώσσα αυτή μια από τις πιο σταθερές επιλογές για κάθε είδους ανάπτυξη λογισμικού ειδικά στο πλαίσιο του web και της ακαδημαϊκής τεχνολογίας.

3.2 Γνωριμία με την Flask

Το Flask αποτελεί ένα από τα πιο ελαφριά και ευέλικτα και δημοφιλή frameworks ανάπτυξης web εφαρμογών στη γλώσσα Python. Αναπτύχθηκε από τον Armin Ronacher και έγινε γρήγορα γνωστό χάρη στην απλότητά του, την καθαρότητα της αρχιτεκτονικής του και την ευκολία με την οποία επιτρέπει τη δημιουργία λειτουργικών διαδικτυακών εφαρμογών με λίγες γραμμές κώδικα. Η φιλοσοφία του Flask βασίζεται στο λεγόμενο "microframework", δηλαδή ένα εργαλείο που παρέχει μόνο τα απολύτως απαραίτητα για τη λειτουργία μιας web εφαρμογής και αφήνει στον προγραμματιστή την ελευθερία να ενσωματώσει μόνο ό,τι χρειάζεται – χωρίς περιττό φορτίο ή επιβαρυντικές εξαρτήσεις [7-9].

Στην εργασία το Flask αποτέλεσε τη βάση για την ανάπτυξη ολόκληρης της λογικής του backend συστήματος. Η επιλογή του έγινε με κριτήρια όπως η ευκολία εκμάθησης, η μεγάλη τεκμηρίωση, η υποστήριξη από την κοινότητα, η ευελιξία στον σχεδιασμό routes και η πλήρης υποστήριξη πρότυπων web τεχνολογιών, όπως HTML, CSS, JavaScript και διαχείριση cookies και sessions.

Η βασική δομή μιας Flask εφαρμογής περιλαμβάνει τον ορισμό διαδρομών (routes), που αντιστοιχούν σε συγκεκριμένες σελίδες ή λειτουργίες της εφαρμογής, την επεξεργασία των αιτημάτων (requests), τη διαχείριση φορμών (forms), και την απόκριση με δυναμικά templates. Το Flask χρησιμοποιεί την

τεχνολογία Jinja2 για την απόδοση HTML templates με μεταβλητά δεδομένα, κάτι που επιτρέπει τη δημιουργία σελίδων με περιεχόμενο που προσαρμόζεται στις ανάγκες του χρήστη και της εφαρμογής.

Στο πλαίσιο του συστήματος διαχείρισης πτυχιακών εργασιών το Flask αξιοποιήθηκε για την υλοποίηση όλων των διαδρομών που σχετίζονται με τη λειτουργικότητα της εφαρμογής. Δημιουργήθηκαν routes για την είσοδο και έξοδο χρηστών, για την προβολή και επεξεργασία πτυχιακών, για την προσθήκη φοιτητών, για την αποστολή και λήψη αρχείων, για τη δημοσίευση σχολίων στο σύστημα συνομιλίας και για την αποστολή email με tokens ορισμού κωδικού. Το Flask επέτρεψε την εύκολη ενσωμάτωση ελέγχου ταυτότητας και ρόλων, δίνοντας τη δυνατότητα να διαφοροποιείται η εμπειρία ανάλογα με το αν ο χρήστης είναι καθηγητής ή φοιτητής.

Μια σημαντική πτυχή του Flask είναι η διαχείριση των sessions. Το Flask παρέχει μηχανισμούς για αποθήκευση πληροφοριών ανάμεσα σε αιτήματα χρηστών μέσω cookies. Αυτή η δυνατότητα αξιοποιήθηκε ώστε να διατηρείται η σύνδεση του χρήστη, να ελέγχεται το δικαίωμα πρόσβασης σε συγκεκριμένες σελίδες και να προσαρμόζεται η διεπαφή με βάση τον ρόλο και την ταυτότητα του χρήστη. Το σύστημα πραγματοποιεί επίσης ελέγχους για την ορθότητα των αιτημάτων και παρέχει κατάλληλα μηνύματα κατάστασης με χρήση της λειτουργίας flash.

Επιλέχθηκε η άμεση χρήση της βιβλιοθήκης mysql.connector για σύνδεση με βάση δεδομένων MySQL, χωρίς ενδιάμεσα μοντέλα, ώστε να διατηρηθεί ο έλεγχος της λογικής και να μειωθεί η πολυπλοκότητα. Η χρήση του Flask παρέμεινε λιτή και στοχευμένη, ώστε η αρχιτεκτονική να είναι εύκολα κατανοητή και συντηρήσιμη από μελλοντικούς χρήστες ή φοιτητές.

Το Flask υποστηρίζει πλήρως τη διαχείριση POST και GET αιτημάτων, τις φορμές HTML με επεξεργασία δεδομένων μέσω request.form, την αποστολή αρχείων μέσω request.files, και την προστασία από επιθέσεις με CSRF ή XSS μέσω εξωτερικών επεκτάσεων ή με χρήση πρακτικών ασφαλούς προγραμματισμού. Η εφαρμογή πτυχιακών ενσωμάτωσε φόρμες με validation, οργάνωσε τα routes ανά λειτουργική ενότητα και χρησιμοποίησε τα πρότυπα templates για τη δυναμική παραγωγή του περιεχομένου.

Ένα από τα πιο σημαντικά πλεονεκτήματα του Flask είναι ότι επιτρέπει σταδιακή υλοποίηση και μπορεί κάποιος να ξεκινήσει από μια απλή εφαρμογή και να την εξελίξει σταδιακά σε πολύπλοκο σύστημα, χωρίς να χρειάζεται να αλλάξει framework ή δομή.

Προσέφερε εργαλεία για την υλοποίηση ενός πλήρους συστήματος διαχείρισης πτυχιακών, με σύγχρονη εμφάνιση, σταθερότητα και προοπτικές επέκτασης. Η χρήση του συνέβαλε σημαντικά στην επίτευξη των στόχων της εφαρμογής και κατέστησε τη διαδικασία ανάπτυξης οργανωμένη, γρήγορη και αποτελεσματική.

Το παρακάτω παράδειγμα αποτελεί την κεντρική ιδέα μιας εφαρμογής Flask για ένα σύστημα πτυχιακών εργασιών. Ξεκινάει με τη σύνδεση με βάση δεδομένων MySQL και την υλοποίηση login.

Κάθε χρήστης αναγνωρίζεται μέσω του session, και μετά την επιτυχή σύνδεση οδηγείται σε διαφορετικό dashboard ανάλογα με τον ρόλο του. Οι καθηγητές βλέπουν όλες τις πτυχιακές που έχουν δημιουργήσει, ενώ οι φοιτητές βλέπουν μόνο αυτή που τους έχει ανατεθεί.

Η χρήση check_password_hash εγγυάται ότι οι κωδικοί δεν αποθηκεύονται σε απλό κείμενο, αλλά σε hashed μορφή. Ο κώδικας είναι δομημένος με τρόπο απλό αλλά ισχυρό, έτοιμος να επεκταθεί με επιπλέον λειτουργίες (π.χ. προβολή λεπτομερειών πτυχιακής, διαχείριση αρχείων, σύστημα συνομιλίας κ.λπ.).

```
# Εισαγωγή απαραίτητων βιβλιοθηκών
from flask import Flask, render_template, request, redirect, url_for, session, flash
import mysql.connector  # Σύνδεση με MySQL
from werkzeug.security import check_password_hash  # Για έλεγχο hashed κωδικού

# Δημιουργία αντικειμένου Flask app
app = Flask(__name__)

# Μυστικό κλειδί για την προστασία των session cookies
app.secret_key = 'your_secret_key_here'

# Συνάρτηση για δημιουργία σύνδεσης με τη βάση MySQL
def get_db_connection():
    return mysql.connector.connect(
        host='localhost',
        user='root',
        password='',
        database='zygoris'
    )

# Αρχική σελίδα - welcome
@app.route('/')
def welcome():
    return render_template('welcome.html')  # Φόρμα login και επαναφοράς κωδικού

# Διαδρομή για επεξεργασία login POST αιτήματος
@app.route('/login', methods=['POST'])
def login():
    # Λήψη στοιχείων από τη φόρμα
    email = request.form['email'].strip().lower()
    password = request.form['password']

    # Σύνδεση στη βάση και αναζήτηση χρήστη
    con = get_db_connection()
    cur = con.cursor()
    cur.execute("SELECT id, password, role FROM users WHERE email = %s", (email,))
    user = cur.fetchone()
    con.close()

    # Έλεγχος αν υπάρχει ο χρήστης και αν ο κωδικός είναι σωστός
    if user and check_password_hash(user[1], password):
        session['user_id'] = user[0]       # Αποθήκευση user_id στο session
        session['role'] = user[2]          # Αποθήκευση ρόλου (teacher ή student)
        flash("Επιτυχής σύνδεση!", "success")
        return redirect(url_for('dashboard'))  # Προώθηση στο dashboard

    else:
        flash("Λάθος email ή κωδικός.", "danger")
        return redirect(url_for('welcome'))  # Επιστροφή στο login

# Διαδρομή για προβολή του dashboard ανάλογα με τον ρόλο
@app.route('/dashboard')
def dashboard():
```

```
    # Έλεγχος αν υπάρχει login χρήστη
    if 'user_id' not in session:
        return redirect(url_for('welcome'))

    user_role = session['role']
    user_id = session['user_id']

    con = get_db_connection()
    cur = con.cursor()

    # Αν είναι καθηγητής: εμφάνιση όλων των πτυχιακών του
    if user_role == 'teacher':
        cur.execute("SELECT id, title FROM theses WHERE teacher_id = %s", (user_id,))
        theses = cur.fetchall()
        con.close()
        return render_template('dashboard_teacher.html', theses=theses)

    # Αν είναι φοιτητής: εμφάνιση μόνο της πτυχιακής στην οποία συμμετέχει
    elif user_role == 'student':
        cur.execute("""
            SELECT t.id, t.title
            FROM theses t
            JOIN thesis_students ts ON t.id = ts.thesis_id
            WHERE ts.student_id = %s
        """, (user_id,))
        thesis = cur.fetchone()
        con.close()
        return render_template('dashboard_student.html', thesis=thesis)

    else:
        con.close()
        return "Μη έγκυρος ρόλος.", 403  # Μη αποδεκτός ρόλος

# Εκκίνηση της Flask εφαρμογής
if __name__ == '__main__':
    app.run(debug=True)
```

3.3 Η MySQL MariaDB

Η επιλογή του κατάλληλου συστήματος διαχείρισης βάσεων δεδομένων (DBMS) είναι κρίσιμη για την επιτυχία κάθε διαδικτυακής εφαρμογής. Στην παρούσα εργασία, επιλέχθηκε η χρήση της MySQL ή εναλλακτικά της πλήρως συμβατής MariaDB, λόγω της σταθερότητας, της απόδοσης, της ευκολίας χρήσης και της ευρείας υποστήριξης σε εκπαιδευτικά και παραγωγικά περιβάλλοντα. Πρόκειται για σχεσιακά συστήματα διαχείρισης βάσεων δεδομένων (RDBMS), τα οποία υλοποιούν το μοντέλο των πινάκων, των σχέσεων και των ερωτημάτων SQL για την αποθήκευση και ανάκτηση δεδομένων.

Η MySQL είναι ανοιχτού κώδικα και ανήκει πλέον στην Oracle, ενώ η MariaDB είναι ένα fork της MySQL που δημιουργήθηκε από τους αρχικούς προγραμματιστές της. Και οι δύο πλατφόρμες υποστηρίζουν το πρότυπο SQL, έχουν παρόμοια σύνταξη, APIs και drivers, γεγονός που επιτρέπει στον προγραμματιστή να αλλάξει από τη μία στην άλλη χωρίς να τροποποιήσει τον κώδικα της εφαρμογής [10-12].

Στο πλαίσιο της πλατφόρμας διαχείρισης πτυχιακών εργασιών η βάση δεδομένων αποτελεί το κεντρικό σημείο αποθήκευσης όλων των βασικών στοιχείων του συστήματος. Μέσα σε αυτή καταγράφονται οι χρήστες (καθηγητές και φοιτητές), οι πτυχιακές εργασίες, οι σχέσεις ανάμεσα σε πτυχιακές και φοιτητές, τα μηνύματα που ανταλλάσσονται μέσω του ενσωματωμένου συστήματος συνομιλίας, καθώς και τα αρχεία που ανεβάζουν οι συμμετέχοντες. Κάθε λειτουργία της εφαρμογής, όπως η σύνδεση, η προσθήκη φοιτητή ή η αποστολή ενός αρχείου, μεταφράζεται σε SQL εντολές που εκτελούνται στη βάση.

Η σχεδίαση της βάσης δεδομένων έγινε με γνώμονα την απλότητα, την κανονικοποίηση των δεδομένων και την ευελιξία στην προσθήκη μελλοντικών λειτουργιών. Οι πίνακες της βάσης έχουν δημιουργηθεί χωρίς τη χρήση FOREIGN KEY περιορισμών, ώστε να προσφέρεται μεγαλύτερη ανεξαρτησία στη διαχείριση και μεταφορά των δεδομένων, ενώ οι σχέσεις υλοποιούνται λογικά μέσω των IDs και της κατάλληλης SQL λογικής.

Η βάση περιλαμβάνει μεταξύ άλλων τους εξής βασικούς πίνακες:

- users: περιλαμβάνει όλους τους χρήστες με στοιχεία email, κωδικό, ρόλο (φοιτητής ή καθηγητής), και χρονικές σφραγίδες (created_at, updated_at).
- theses: αποθηκεύει τις πτυχιακές εργασίες με τίτλο, περιγραφή, σημειώσεις του επιβλέποντα και τον δημιουργό (teacher_id).
- thesis_students: συνδέει φοιτητές με πτυχιακές μέσω των πεδίων thesis_id και student_id.
- messages: περιλαμβάνει τα μηνύματα των χρηστών με δυνατότητα απάντησης μέσω parent_id.
- files: καταγράφει τα αρχεία που ανεβαίνουν, με αναφορά στο ποιος τα ανέβασε και σε ποια πτυχιακή ανήκουν.

Κάθε πίνακας περιλαμβάνει επίσης τα πεδία created_at και updated_at, που προσφέρουν δυνατότητα παρακολούθησης της δραστηριότητας και εύκολης ταξινόμησης.

Η σύνδεση της εφαρμογής Flask με τη βάση MySQL/MariaDB γίνεται μέσω της βιβλιοθήκης mysql.connector. Η επικοινωνία περιλαμβάνει τα εξής βασικά στάδια:

1. Άνοιγμα σύνδεσης με get_db_connection() που επιστρέφει ενεργή σύνδεση.
2. Δημιουργία cursor για εκτέλεση εντολών SQL.
3. Εκτέλεση ερωτημάτων (SELECT, INSERT, UPDATE, DELETE).
4. Commit των αλλαγών, όπου απαιτείται.
5. Κλείσιμο σύνδεσης για αποφυγή διαρροών πόρων.

Η χρήση παραμέτρων με cur.execute(sql, values) προσφέρει ασφάλεια έναντι SQL injection, ενώ οι εντολές είναι απλές και κατανοητές, ακόμα και σε φοιτητές με βασικές γνώσεις βάσεων δεδομένων.

Η MySQL/MariaDB προσφέρει πολλά πλεονεκτήματα:

- Ταχύτητα και σταθερότητα σε μικρές και μεσαίες εφαρμογές.
- Συμβατότητα με τα περισσότερα λειτουργικά συστήματα και περιβάλλοντα.
- Ευκολία στη δημιουργία και επεξεργασία βάσεων μέσω εργαλείων όπως phpMyAdmin, DBeaver ή MySQL Workbench.
- Εκτενής τεκμηρίωση και υποστήριξη από κοινότητα.
- Ασφάλεια, με δυνατότητες δημιουργίας χρηστών, ορισμού δικαιωμάτων και κρυπτογράφησης κωδικών.

Η επιλογή MySQL/MariaDB ήταν στρατηγική: πρόκειται για τεχνολογίες που χρησιμοποιούνται ευρέως σε πανεπιστήμια και είναι πλήρως συμβατές με open-source stacks όπως Flask, προσφέροντας ιδανική ισορροπία μεταξύ ισχύος και απλότητας.

Η χρήση της MySQL/MariaDB στην παρούσα εργασία προσέφερε μια αξιόπιστη υποδομή για την αποθήκευση και ανάκτηση όλων των κρίσιμων δεδομένων της εφαρμογής. Η ευκολία ενσωμάτωσης με Python και Flask, η δυνατότητα επέκτασης και ο χαμηλός προγραμματιστικός φόρτος, καθιστούν την τεχνολογία αυτή ιδανική επιλογή για εφαρμογές όπως αυτή της διαχείρισης πτυχιακών. Η απλή αλλά ισχυρή σχεδίαση της βάσης εξυπηρετεί με σαφήνεια το μοντέλο χρήσης της πλατφόρμας, επιτρέποντας μελλοντική συντήρηση και εξέλιξη.

3.4 Bootstrap/CSS/jquery

Ο σχεδιασμός της διεπαφής χρήστη (frontend) μιας web εφαρμογής είναι σημαντικός με τη λογική που υλοποιείται στο backend. Μια καθαρή και λειτουργική διεπαφή μπορεί να βοηθήσει σημαντικά την εμπειρία του τελικού χρήστη και να διευκολύνει τη χρήση και να ενισχύσει την αλληλεπίδραση με το σύστημα. Ο σχεδιασμός του frontend βασίστηκε στον συνδυασμό των τεχνολογιών Bootstrap, CSS και jQuery οι οποίες συνεργάζονται άψογα για να παρέχουν responsive σχεδίαση, όμορφη αισθητική και διαδραστικά στοιχεία.

Το Bootstrap είναι το πιο δημοφιλές frontend framework και χρησιμοποιήθηκε εκτενώς για τη διάταξη και αισθητική όλων των σελίδων της εφαρμογής. Παρέχει έτοιμες κλάσεις CSS και συστατικά (components) που επιτρέπουν τη δημιουργία responsive διατάξεων, κουμπιών, φορμών, καρτών, πλαισίων, alerts, πινάκων και navigation bars, χωρίς να απαιτείται από τον προγραμματιστή να γράψει μεγάλο όγκο custom CSS.

Το πλεονέκτημα του Bootstrap είναι ότι λειτουργεί "έξω από το κουτί". Η εφαρμογή κάνει χρήση του grid system του Bootstrap για διάταξη των στηλών, των tabs και των panels, εξασφαλίζοντας ότι κάθε

σελίδα εμφανίζεται σωστά σε desktop, tablet και κινητές συσκευές. Ενσωματώθηκαν στοιχεία όπως navbar, modals, cards και alerts για να δημιουργηθεί μια συνεπής, επαγγελματική και ευχάριστη εμπειρία χρήσης.

Οι φόρμες σύνδεσης, δημιουργίας πτυχιακής, αποστολής αρχείων και προσθήκης φοιτητή στηρίζονται εξολοκλήρου σε Bootstrap components, διασφαλίζοντας σωστή εμφάνιση και προσβασιμότητα. Οι ειδοποιήσεις (flash) εμφανίζονται ως Bootstrap alerts με διακριτικά χρώματα ανάλογα με το μήνυμα (success, danger, info).

Αν και το Bootstrap καλύπτει τις βασικές σχεδιαστικές ανάγκες, η χρήση custom CSS είναι απαραίτητη για την προσαρμογή της διεπαφής στις ιδιαιτερότητες της εφαρμογής. Μέσω αρχείου style.css, έγινε επέκταση ή τροποποίηση βασικών χαρακτηριστικών του Bootstrap, ώστε να προστεθεί ομορφιά στο σύστημα.

Η CSS συνέβαλε στη δημιουργία πιο "καμπύλων" αισθητικών δίνοντας στην εφαρμογή μια πιο μοντέρνα και ευχάριστη εικόνα. Η χρήση CSS έγινε με φιλικό και οργανωμένο τρόπο για να είναι εύκολα συντηρήσιμη και επεκτάσιμη στο μέλλον.

Η χρήση της βιβλιοθήκης jQuery προσέθεσε επίπεδο διαδραστικότητας στο frontend. Παρόλο που το Flask λειτουργεί κυρίως με server-side rendering, υπήρχαν περιπτώσεις όπου η χρήση client-side JavaScript ήταν απαραίτητη για βελτίωση της εμπειρίας χρήσης.

Με το jQuery υλοποιήθηκαν λειτουργίες όπως:

- Εμφάνιση/απόκρυψη περιοχών ανάλογα με την ενέργεια του χρήστη (π.χ. εμφάνιση πεδίου απάντησης στο chat).
- Επαλήθευση δεδομένων σε φόρμες πριν την αποστολή (π.χ. έλεγχος για κενά πεδία ή επιβεβαίωση διαγραφής).
- Αυτόματη εναλλαγή tabs και scroll στο τέλος του chat.
- Ενεργοποίηση modal παραθύρων με επιβεβαίωση πριν κρίσιμες ενέργειες (όπως διαγραφή αρχείων ή φοιτητή).

Το jQuery επέτρεψε τη δημιουργία μιας πιο ζωντανής και ευέλικτης διεπαφής, χωρίς να μεταφέρεται περιττή λογική στον server. Σε συνδυασμό με τις AJAX δυνατότητές του, μπορεί στο μέλλον να αξιοποιηθεί για δυναμική φόρτωση στοιχείων χωρίς επαναφόρτωση της σελίδας.

Η δομή του frontend βασίζεται σε templates HTML με Jinja2 (για Flask), τα οποία "χτίζονται" πάνω στο base.html. Το αρχείο αυτό περιλαμβάνει:

- την κοινή navbar (menu πλοήγησης),
- τη σύνδεση των CSS αρχείων,
- την εισαγωγή των βιβλιοθηκών Bootstrap, jQuery και FontAwesome,
- και την υποδοχή του δυναμικού περιεχομένου με το {% block content %}.

Από εκεί και πέρα κάθε επιμέρους σελίδα (welcome.html, dashboard_teacher.html, thesis_view.html, κ.λπ.) επεκτείνει το base.html και εισάγει το δικό της περιεχόμενο για ενιαία σχεδίαση σε όλη την εφαρμογή.

Ο συνδυασμός Bootstrap, CSS και jQuery αποτέλεσε έναν αποδοτικό και ισορροπημένο τρόπο για τη δημιουργία ενός λειτουργικού, καλαίσθητου και φιλικού frontend για την πλατφόρμα πτυχιακών εργασιών. Το Bootstrap προσέφερε άμεση υποστήριξη για responsive σχεδίαση και έτοιμα στοιχεία, η CSS έδωσε την απαραίτητη προσαρμογή και προσωπικότητα, ενώ το jQuery εμπλούτισε την εμπειρία με διαδραστικότητα και ευχρηστία. Το αποτέλεσμα είναι ένα σύστημα που μπορεί να χρησιμοποιηθεί με άνεση από φοιτητές και καθηγητές, είτε από υπολογιστή είτε από κινητή συσκευή.

Κεφάλαιο 4ο: Το Σύστημα μας

Το σύστημα που αναπτύχθηκε αποτελεί μία λειτουργική διαδικτυακή πλατφόρμα διαχείρισης πτυχιακών εργασιών για να καλύπτει τις ανάγκες συνεργασίας μεταξύ επιβλεπόντων καθηγητών και φοιτητών. Βασίζεται στη οριοθέτηση των ρόλων, την απλότητα στη χρήση και τη διαφάνεια στην επικοινωνία. Υπάρχουν μόνο δύο ρόλοι χρηστών: ο Επιβλέπων και ο Φοιτητής της Πτυχιακής.

Ο επιβλέπω, μετά τη σύνδεσή του στο σύστημα, έχει τη δυνατότητα να δημιουργεί νέες πτυχιακές εργασίες δίνοντας έναν τίτλο, μία περιγραφή και έναν μοναδικό κωδικό αναγνώρισης. Επιπλέον, μπορεί να ορίζει σημειώσεις προς τους φοιτητές, οι οποίες είναι ορατές σε εκείνους αλλά επεξεργάσιμες μόνο από τον ίδιο. Σε κάθε πτυχιακή έχει τη δυνατότητα να προσθέσει έναν ή περισσότερους φοιτητές με βάση το email τους. Αν ο φοιτητής δεν υπάρχει ήδη στη βάση, δημιουργείται αυτόματα ένας λογαριασμός με ρόλο 'student', χωρίς κωδικό, και του αποστέλλεται email για να ορίσει το δικό του.

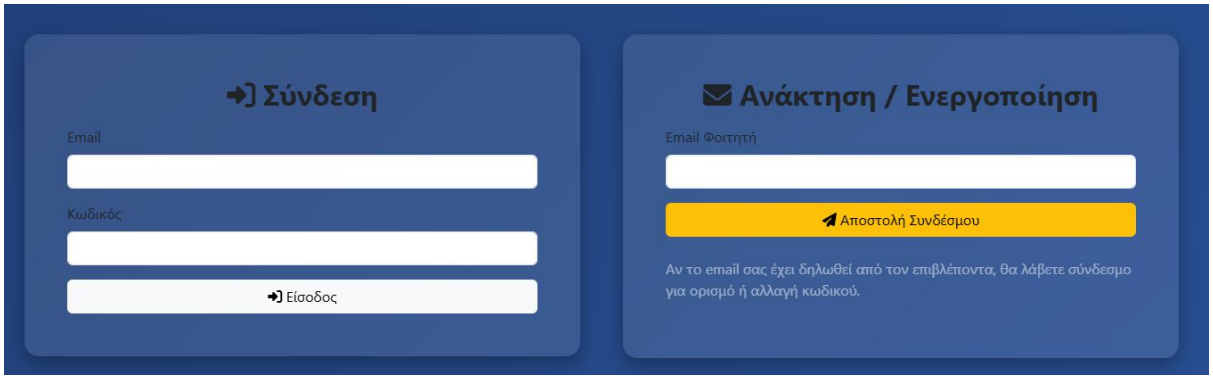
Κάθε πτυχιακή περιλαμβάνει σύστημα ενσωματωμένης συζήτησης (chat), το οποίο επιτρέπει την αποστολή μηνυμάτων και απαντήσεων σε επίπεδο νήματος. Η επικοινωνία είναι αμφίδρομη, απλή και καταγεγραμμένη, με χρονοσφραγίδες και ονόματα αποστολέων, προσφέροντας δομημένη και συνεχή ανατροφοδότηση μεταξύ επιβλέποντα και φοιτητών. Επιπλέον, κάθε χρήστης μπορεί να ανεβάζει αρχεία σχετικά με την πτυχιακή, τα οποία εμφανίζονται σε λίστα με δυνατότητα προεπισκόπησης ή λήψης. Η διαγραφή των αρχείων είναι επιτρεπτή μόνο στον χρήστη που τα ανέβασε ή στον καθηγητή.

Οι φοιτητές έχουν περιορισμένη αλλά ουσιαστική, πρόσβαση. Μπορούν να δουν τα βασικά στοιχεία της πτυχιακής (τίτλο, περιγραφή, σημειώσεις), να συμμετέχουν στη συζήτηση, να ανεβάζουν αρχεία και να ακολουθούν την πρόοδο. Δεν έχουν πρόσβαση στη διαχείριση μελών ή στην επεξεργασία των στοιχείων της εργασίας, διατηρώντας έτσι την ακεραιότητα της οργάνωσης.

Η πλατφόρμα είναι σχεδιασμένη με βάση το Flask framework και χρησιμοποιεί MySQL για την αποθήκευση δεδομένων. Το frontend υλοποιείται με Bootstrap και FontAwesome για μοντέρνα και responsive διεπαφή. Η αρχιτεκτονική διατηρεί διαχωρισμό ανάμεσα σε παρουσίαση και λογική, με χρήση templates και blueprints όπου χρειάζεται. Έχουν ληφθεί υπόψη θέματα ασφάλειας, όπως η διαχείριση session, η προστασία routes και η χρήση hash για τους κωδικούς.

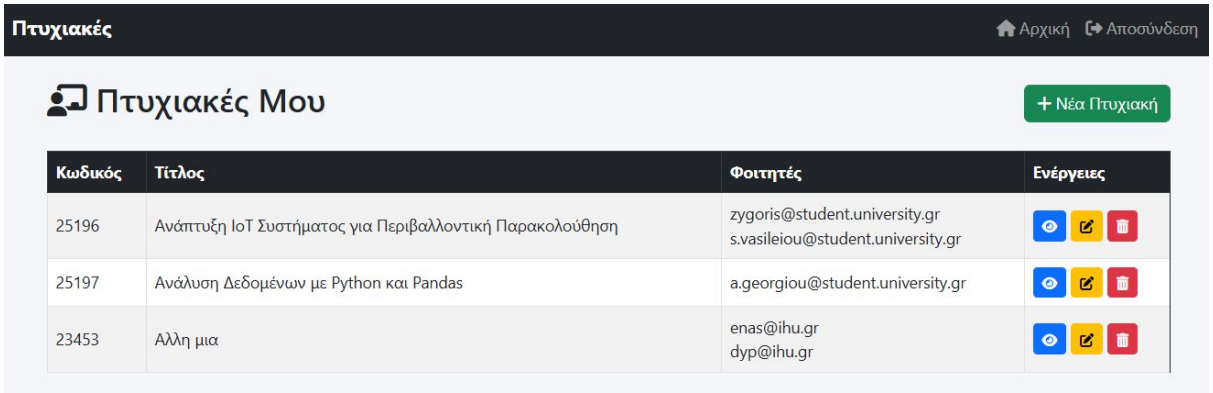
4.1 Οι ιστοσελίδες της πλατφόρμας

Στην Εικόνα 4.1 φαίνεται η αρχική σελίδα της εφαρμογής, η οποία παρέχει στους χρήστες δύο βασικές δυνατότητες: την είσοδο με υπάρχοντα λογαριασμό και την ενεργοποίηση ή ανάκτηση πρόσβασης με βάση το email. Η διάταξη είναι χωρισμένη σε δύο κάρτες — στα αριστερά βρίσκεται η φόρμα σύνδεσης με πεδία email και κωδικού, ενώ στα δεξιά υπάρχει η φόρμα ενεργοποίησης/ανάκτησης, στην οποία ο φοιτητής εισάγει το email του ώστε να του σταλεί σύνδεσμος επαναφοράς. Το περιβάλλον είναι οπτικά εντυπωσιακό, με gradient φόντο, εικονίδια FontAwesome και διακριτικά animations κατά την είσοδο των φορμών.



Εικόνα 4.1: Σελίδα σύνδεσης και ανάκτησης ή ενεργοποίησης λογαριασμού

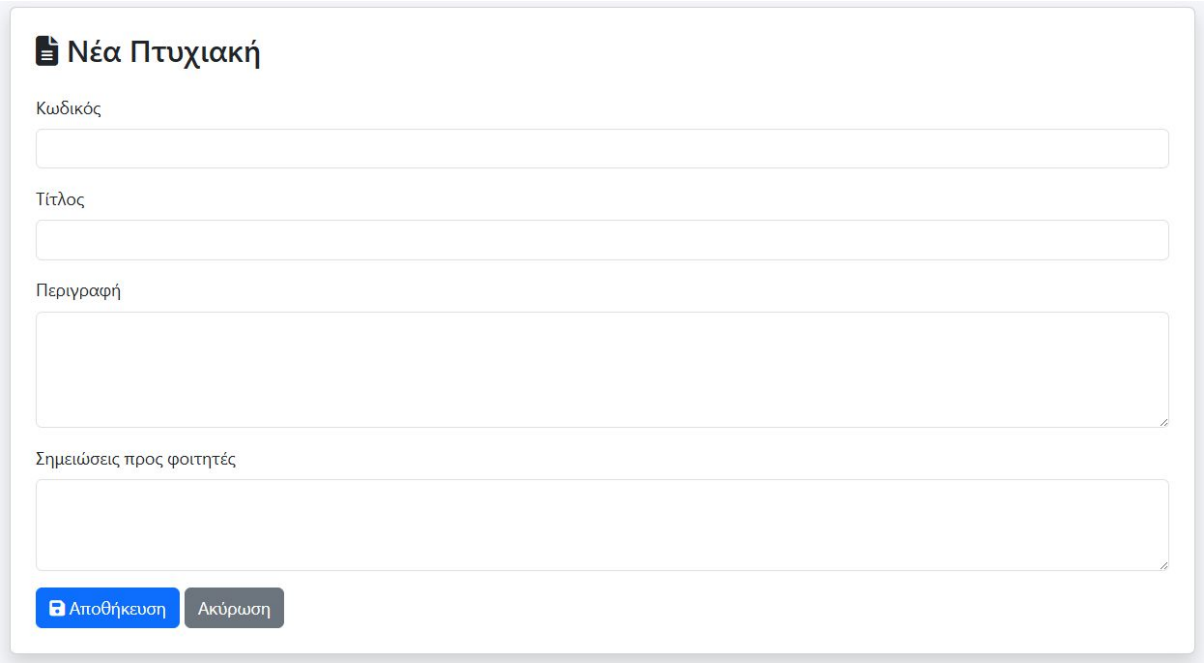
Η Εικόνα 4.2 παρουσιάζει τη σελίδα όπου ο καθηγητής βλέπει συγκεντρωμένες όλες τις πτυχιακές εργασίες που έχει δημιουργήσει ή διαχειρίζεται. Η λίστα είναι οργανωμένη σε πίνακα, με στήλες για τον τίτλο, τον μοναδικό κωδικό, τα email των φοιτητών που συμμετέχουν, καθώς και κουμπιά για ενέργειες όπως προβολή, επεξεργασία ή διαγραφή κάθε πτυχιακής. Το περιβάλλον είναι λειτουργικά λιτό και αποδοτικό, επιτρέποντας στον καθηγητή να έχει πλήρη επισκόπηση και έλεγχο των εργασιών του.



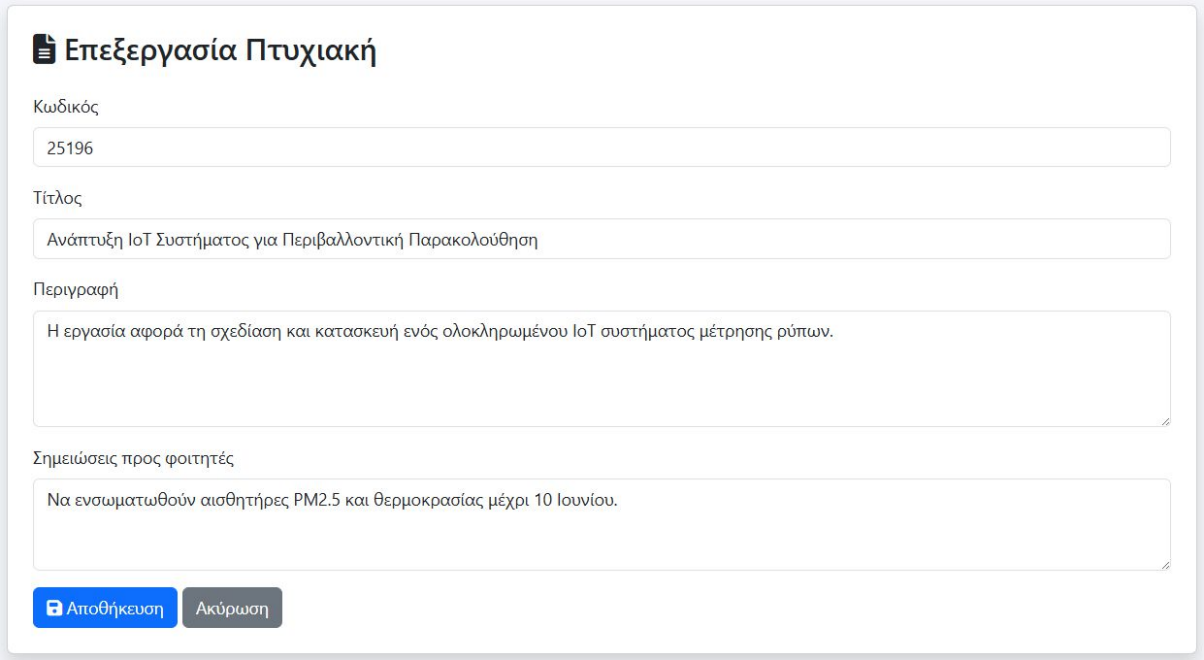
Εικόνα 4.2: Η σελίδα με τη λίστα των Πτυχιακών - Teacher

Στην Εικόνα 4.3 αποτυπώνεται η φόρμα δημιουργίας νέας πτυχιακής από τον επιβλέποντα καθηγητή. Ο χρήστης έχει τη δυνατότητα να εισάγει τον τίτλο, την περιγραφή, τον μοναδικό κωδικό της πτυχιακής καθώς και σημειώσεις που θα είναι ορατές στους φοιτητές. Η φόρμα είναι απλή, με ξεκάθαρα πεδία και κουμπί αποθήκευσης, ενώ υπάρχει και επιλογή ακύρωσης που οδηγεί πίσω στο dashboard.

Η Εικόνα 4.4 δείχνει την οθόνη επεξεργασίας μιας ήδη υπάρχουσας πτυχιακής εργασίας. Η δομή είναι παρόμοια με τη δημιουργία, με τα πεδία προσυμπληρωμένα ώστε να μπορούν να τροποποιηθούν εύκολα. Ο καθηγητής μπορεί να αλλάξει τα βασικά στοιχεία της πτυχιακής χωρίς να επηρεάζονται οι συνδεδεμένοι φοιτητές ή τα μηνύματα.



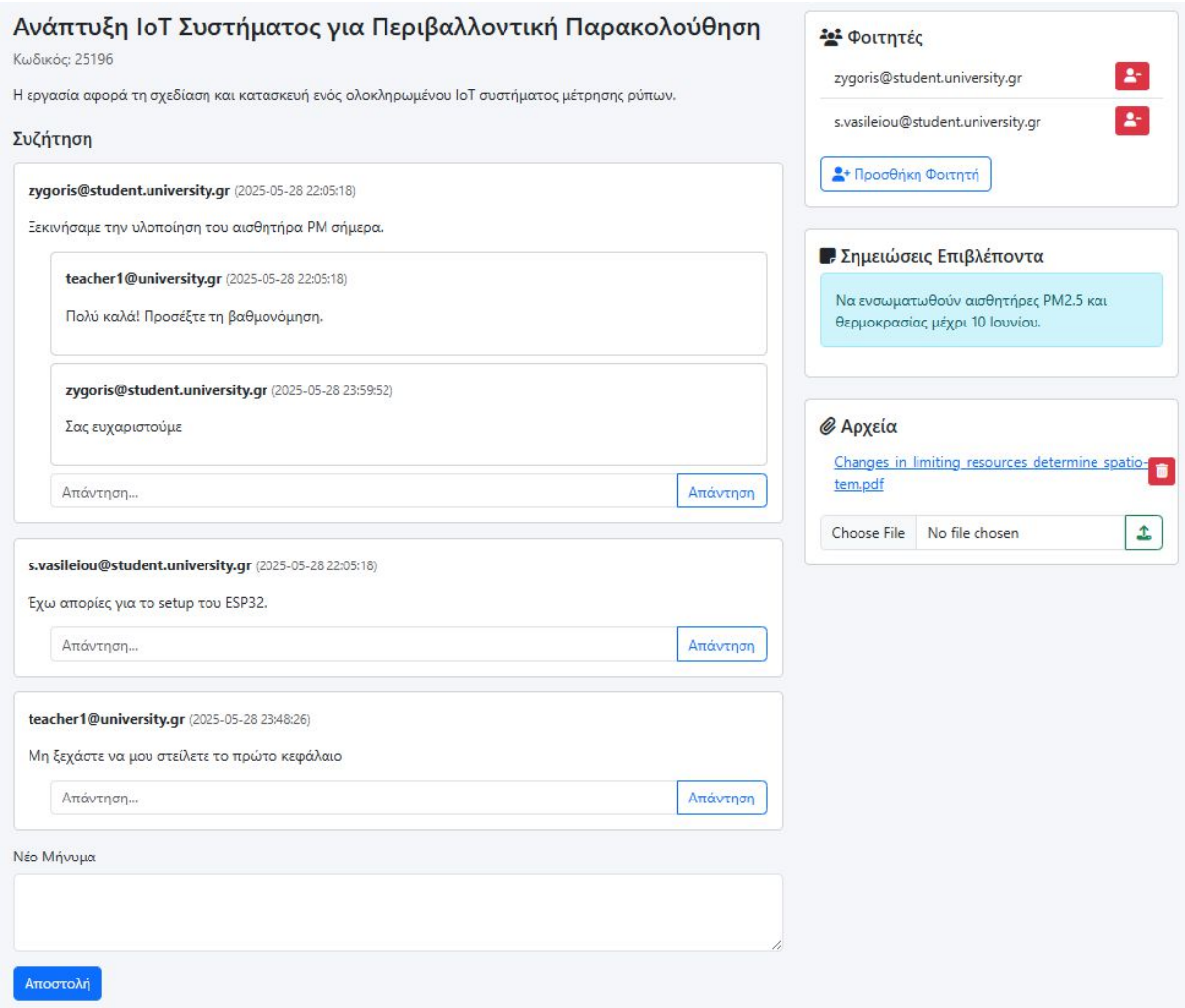
Εικόνα 4.3: Δημιουργία Νέας Πτυχιακής - Teacher



Εικόνα 4.4: Επεξεργασία Πτυχιακής - Teacher

Η Εικόνα 4.4 δείχνει την οθόνη επεξεργασίας μιας ήδη υπάρχουσας πτυχιακής εργασίας. Η δομή είναι παρόμοια με τη δημιουργία, με τα πεδία προσυμπληρωμένα ώστε να μπορούν να τροποποιηθούν εύκολα. Ο καθηγητής μπορεί να αλλάξει τα βασικά στοιχεία της πτυχιακής χωρίς να επηρεάζονται οι συνδεδεμένοι φοιτητές ή τα μηνύματα.

Η Εικόνα 4.5 καταγράφει τη βασική σελίδα προβολής μιας πτυχιακής όπως την βλέπει ο επιβλέπων. Στο κύριο τμήμα της σελίδας δεσπόζει το σύστημα συζήτησης (chat) όπου οι φοιτητές και ο καθηγητής μπορούν να ανταλλάσσουν μηνύματα και απαντήσεις. Κάθε μήνυμα εμφανίζεται με όνομα αποστολέα και χρονική σήμανση.

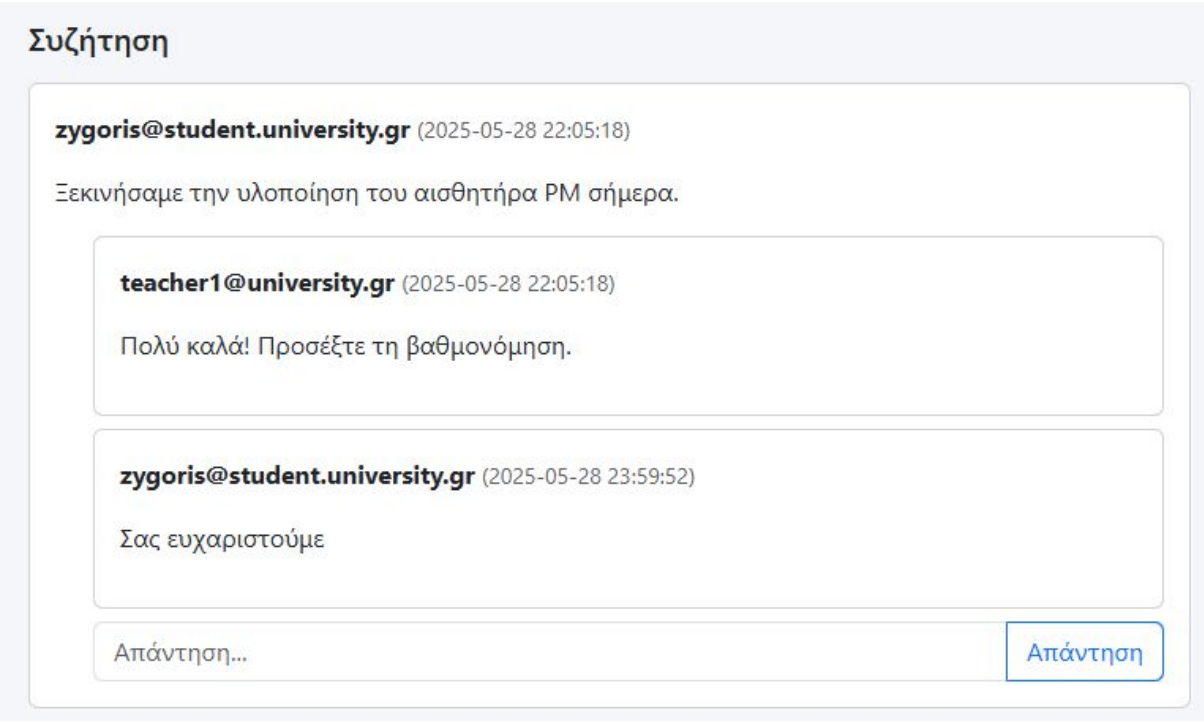


Εικόνα 4.5: Η κεντρική σελίδα που βλέπει ο Teacher – Στο κεντρικό το chat για επικοινωνία με τα μέλη φοιτητές της πτυχιακής

Στην Εικόνα 4.6 εμφανίζεται το πάνω μέρος της σελίδας πτυχιακής, όπου διακρίνονται ο τίτλος, ο κωδικός της εργασίας και η περιγραφή που έχει εισαχθεί κατά τη δημιουργία ή την επεξεργασία της. Αυτές οι πληροφορίες δίνουν το γενικό πλαίσιο του έργου τόσο στον επιβλέποντα όσο και στους φοιτητές.

Η Εικόνα 4.7 επικεντρώνεται στο σύστημα μηνυμάτων που χρησιμοποιείται για την επικοινωνία μεταξύ καθηγητή και φοιτητών. Υποστηρίζονται απαντήσεις ενός επιπέδου (nested), ώστε να διατηρείται η συνοχή στη συζήτηση και να γίνεται σαφές ποιος απαντά σε ποιον.

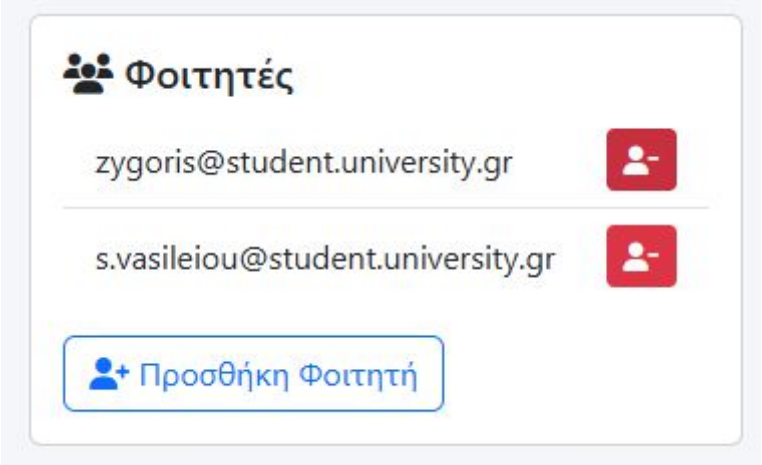
Εικόνα 4.6: Στην κεντρική σελίδα που βλέπει ο Teacher φαίνεται ο τίτλος, ο κωδικός και η περιγραφή



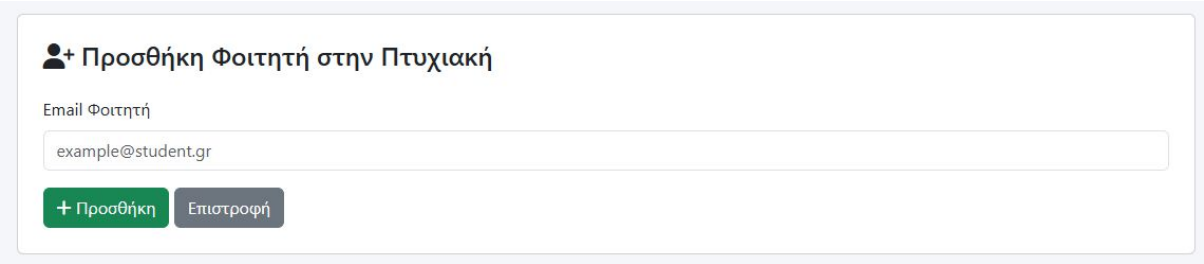
Εικόνα 4.7: Στην κεντρική σελίδα που βλέπει ο Teacher φαίνεται το chat για επικοινωνία με τα μέλη φοιτητές της πτυχιακής

Στην Εικόνα 4.8 προβάλλεται το δεξί τμήμα της σελίδας της πτυχιακής, όπου καταγράφονται οι φοιτητές που συμμετέχουν. Για κάθε φοιτητή υπάρχει δυνατότητα αφαίρεσης από την πτυχιακή μέσω σχετικού κουμπιού. Επιπλέον, στο κάτω μέρος υπάρχει επιλογή για προσθήκη νέου μέλους με βάση το email του.

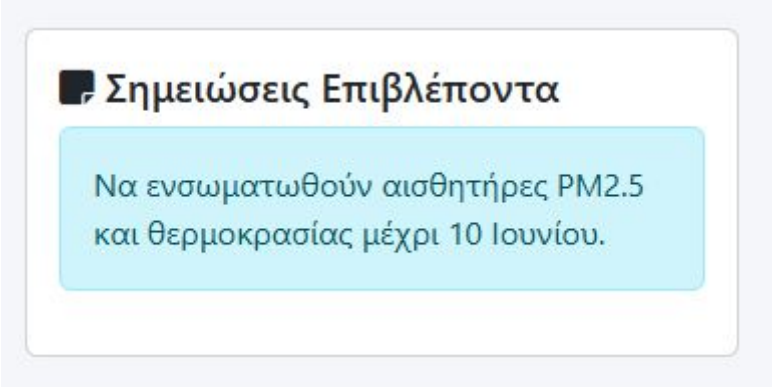
Η Εικόνα 4.9 δείχνει την ειδική φόρμα προσθήκης φοιτητή (ή καθηγητή) σε μια πτυχιακή εργασία. Ο επιβλέπων εισάγει το email και, αν δεν υπάρχει ήδη ο φοιτητής στη βάση, δημιουργείται αυτόματα. Αν είναι ήδη γραμμένος, προστίθεται στην πτυχιακή εφόσον δεν συμμετέχει ήδη.



Εικόνα 4.8: Στην κεντρική σελίδα που βλέπει ο Teacher δεξιά φαίνονται τα μέλη φοιτητές, η δυνατότητα αφαίρεσης τους και το κουμπί για να μεταβεί στη σελίδα προσθήκης μελών.



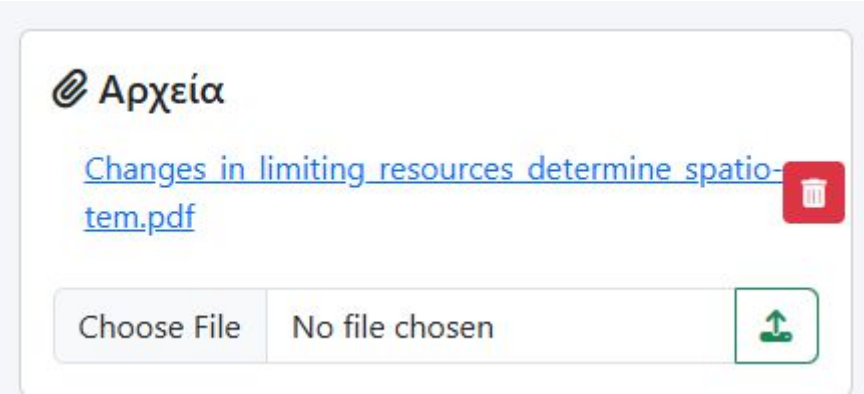
Εικόνα 4.9: Προσθήκη μελών (φοιτητών ή καθηγητών)



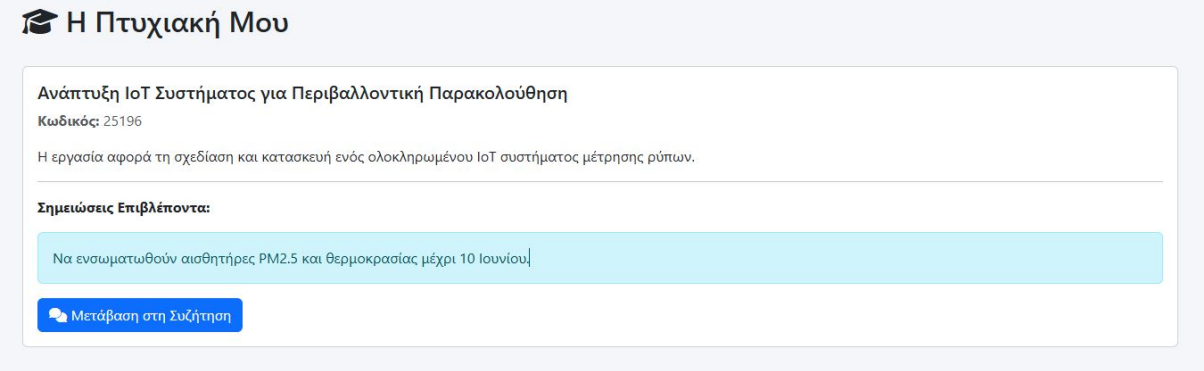
Εικόνα 4.10: Στην κεντρική σελίδα που βλέπει ο Teacher δεξιά φαίνονται οι Σημειώσεις του Επιβλέποντα

Στην Εικόνα 4.10 εμφανίζεται το πλαίσιο που περιλαμβάνει τις σημειώσεις του επιβλέποντα. Οι σημειώσεις είναι ορατές μόνο στους φοιτητές της πτυχιακής και μπορούν να τροποποιούνται αποκλειστικά από τον καθηγητή.

Η Εικόνα 4.11 παρουσιάζει το τμήμα διαχείρισης αρχείων. Εκεί προβάλλονται όλα τα αρχεία που έχουν ανεβεί από φοιτητές ή τον επιβλέποντα. Για κάθε αρχείο υπάρχει σύνδεσμος λήψης, ενώ οι χρήστες που το έχουν ανεβάσει μπορούν να το διαγράψουν. Οι καθηγητές έχουν πλήρη έλεγχο στα αρχεία όλων των μελών.



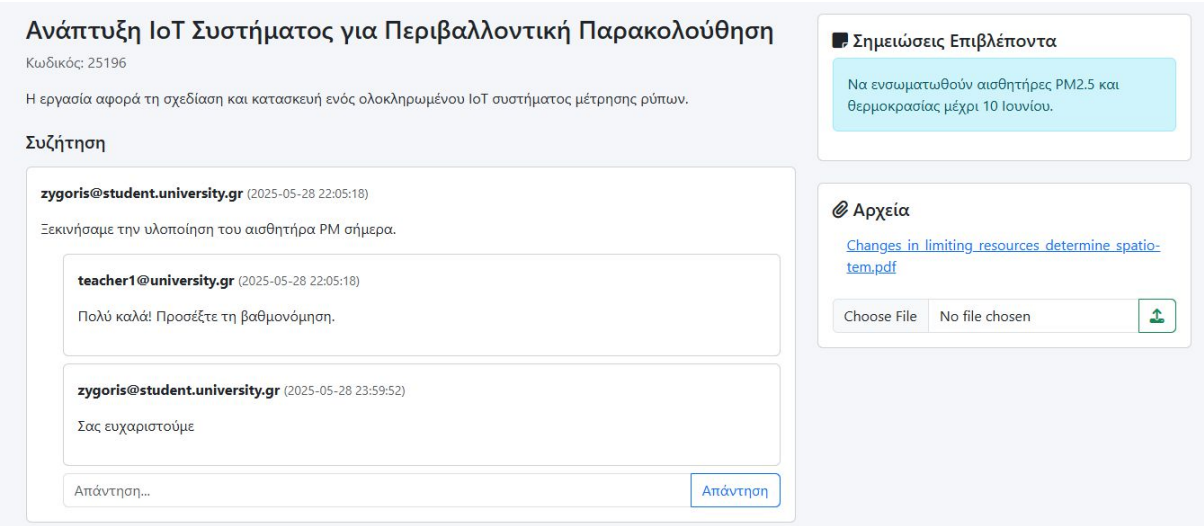
Εικόνα 4.11: Στην κεντρική σελίδα δεξιά φαίνεται η διαχείριση των αρχείων από τον Teacher



Εικόνα 4.12: Στην κεντρική σελίδα που βλέπει ο Student δεξιά φαίνονται τίτλος, περιγραφή και οι Σημειώσεις του Επιβλέποντα

Στην Εικόνα 4.12 φαίνεται η αντίστοιχη σελίδα όπως την βλέπει ένας φοιτητής. Δεξιά προβάλλεται ο τίτλος, η περιγραφή και οι σημειώσεις του επιβλέποντα, ώστε να έχει πλήρη εικόνα για το αντικείμενο και τις κατευθύνσεις της πτυχιακής.

Η Εικόνα 4.13 δείχνει τη συνολική σελίδα του φοιτητή, η οποία μοιάζει με εκείνη του καθηγητή αλλά δεν περιλαμβάνει τις ενότητες διαχείρισης μελών και αρχείων. Έτσι διασφαλίζεται ότι ο φοιτητής μπορεί να δει και να συμμετάσχει χωρίς να επηρεάζει την οργάνωση της πτυχιακής.



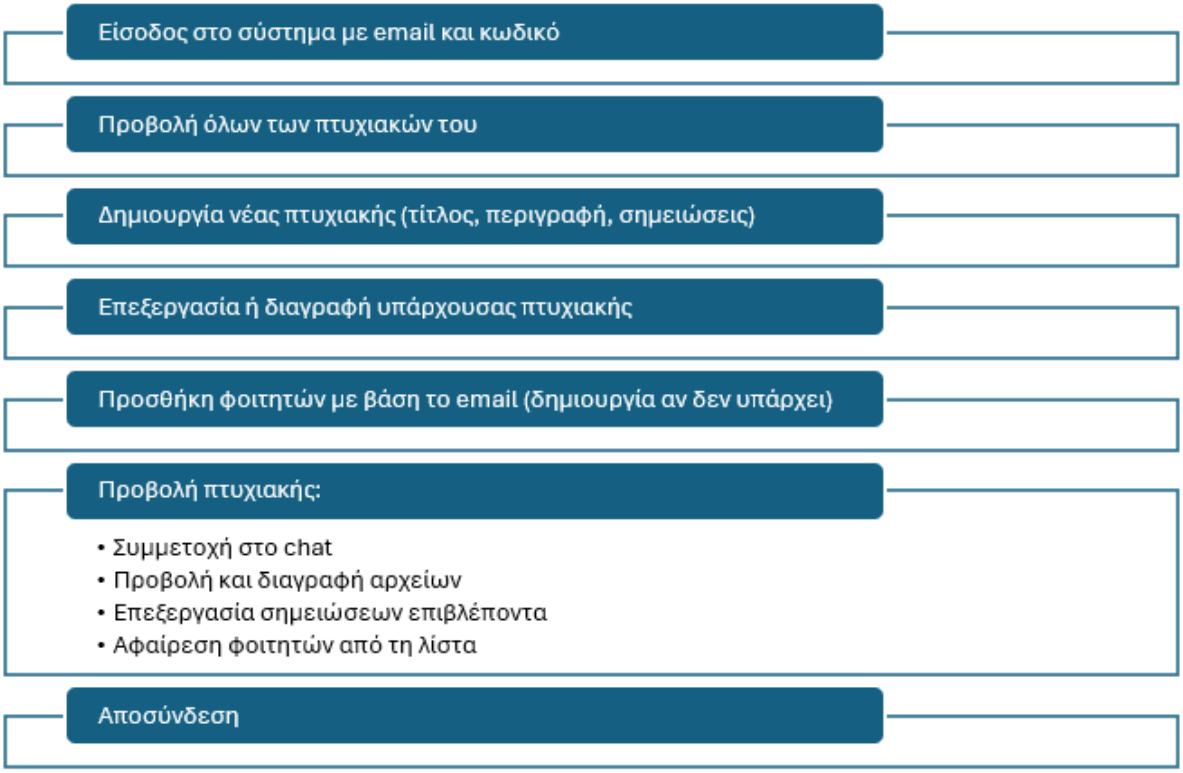
Εικόνα 4.13: Στην κεντρική σελίδα που βλέπει ο Student φαίνονται όσα βλέπει και ο Teacher εκτός διαχείριση μελών και αρχείων.

4.2 Περιγραφή του συστήματος μέσω διαγραμμάτων

Το πρώτο διάγραμμα απεικονίζει αναλυτικά τη λειτουργική ροή του επιβλέποντα καθηγητή στο σύστημα. Ξεκινώντας με την είσοδο στην πλατφόρμα μέσω email και κωδικού, ο χρήστης μεταβαίνει στο dashboard όπου μπορεί να δει συγκεντρωμένες όλες τις πτυχιακές που διαχειρίζεται. Από εκεί έχει τη δυνατότητα να δημιουργήσει νέα πτυχιακή εργασία εισάγοντας τίτλο, περιγραφή και σημειώσεις προς τους φοιτητές ή να επεξεργαστεί/διαγράψει μια υπάρχουσα.

Ένα από τα πιο βασικά σημεία είναι η προσθήκη φοιτητών σε κάθε πτυχιακή, όπου η εφαρμογή αναλαμβάνει να δημιουργήσει λογαριασμό αν το email δεν υπάρχει. Στη συνέχεια, η προβολή της πτυχιακής δίνει πρόσβαση σε κρίσιμες λειτουργίες: συμμετοχή στη συζήτηση (chat), διαχείριση αρχείων, επεξεργασία των σημειώσεων και αφαίρεση φοιτητών από τη λίστα.Ο κύκλος χρήσης ολοκληρώνεται με την αποσύνδεση.

Το διάγραμμα τονίζει με σαφήνεια τον ρόλο του καθηγητή ως διοικητή της ροής πληροφορίας και υπεύθυνου για τον ορισμό της πτυχιακής και των μελών της.

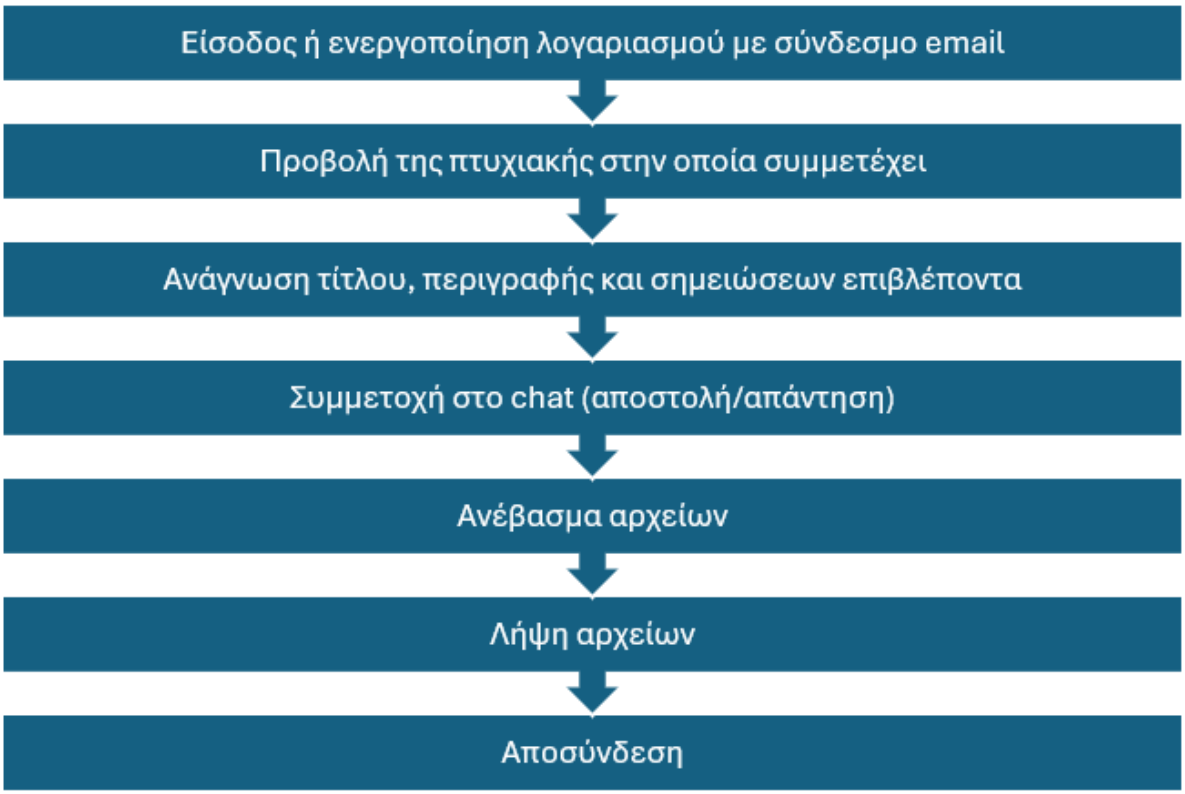


Εικόνα 4.14: Διάγραμμα Ροής Χρήστη - Teacher

Το δεύτερο διάγραμμα περιγράφει τη διαδρομή ενός φοιτητή μέσα στο σύστημα, εστιάζοντας στην απλότητα και την καθοδηγούμενη συμμετοχή του. Ο φοιτητής είτε ενεργοποιεί λογαριασμό μέσω συνδέσμου που έλαβε στο email είτε συνδέεται αν έχει ήδη θέσει κωδικό. Μετά την επιτυχή είσοδο, μεταφέρεται αυτόματα στην πτυχιακή στην οποία έχει ανατεθεί.

Η αρχική προβολή περιλαμβάνει τον τίτλο, την περιγραφή και τις σημειώσεις του επιβλέποντα, προσφέροντας το γενικό πλαίσιο του έργου. Στη συνέχεια, ο φοιτητής έχει ενεργή συμμετοχή στο σύστημα συζήτησης (chat), όπου μπορεί να απαντά ή να δημοσιεύει νέα μηνύματα. Μπορεί επίσης να ανεβάζει αρχεία και να κατεβάζει αυτά που έχουν αναρτηθεί από τον καθηγητή ή άλλα μέλη. Η διαδικασία είναι κυκλική και ολοκληρώνεται με την αποσύνδεση.

Το διάγραμμα αναδεικνύει τον ρόλο του φοιτητή ως ενεργού μέλους της ομάδας με έμφαση στη συμμετοχή και την πληροφόρηση και όχι στη διαχείριση.



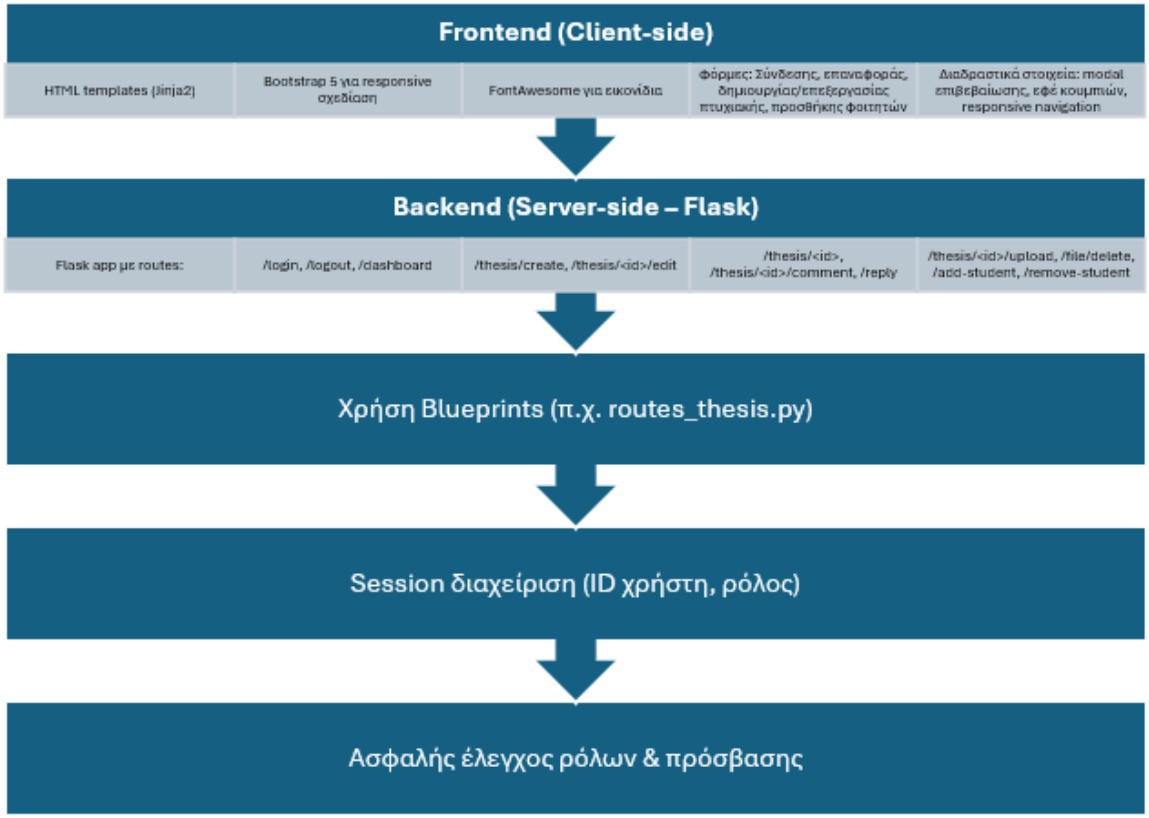
Εικόνα 4.15: Διάγραμμα Ροής Χρήστη - Student

Το τρίτο διάγραμμα στην Εικόνα 4.16 καταγράφει την αρχιτεκτονική της πλατφόρμας σε επίπεδο λογικών μονάδων, ακολουθώντας την προσέγγιση ενός layered system. Στην κορυφή βρίσκεται το frontend (client-side), το οποίο βασίζεται σε HTML templates με Jinja2 και αξιοποιεί Bootstrap 5 και FontAwesome για responsive σχεδίαση και εικονίδια. Περιλαμβάνει φόρμες για όλες τις βασικές ενέργειες και δυναμικά στοιχεία όπως επιβεβαιώσεις και πλοήγηση.

Το backend είναι δομημένο με Flask, με routes που διαχειρίζονται τη ροή δεδομένων ανάμεσα στον client και τη βάση. Χωρίζεται σε κομμάτια που περιλαμβάνουν login, dashboard, CRUD για πτυχιακές, chat, αρχεία και διαχείριση μελών.

Ακολουθεί η χρήση blueprints (π.χ. routes_thesis.py) για καλύτερη οργάνωση, και στη συνέχεια το session layer που διαχειρίζεται την ταυτότητα και τον ρόλο κάθε χρήστη. Στη βάση αυτής της λογικής ιεραρχίας βρίσκεται η ασφάλεια: με ελέγχους πρόσβασης σε κάθε route και με session isolation, διασφαλίζεται ότι κάθε χρήστης βλέπει και κάνει μόνο ό,τι του επιτρέπεται.

Το διάγραμμα καταδεικνύει την καθαρή διαστρωμάτωση του συστήματος και επιβεβαιώνει τη λειτουργικότητα και επεκτασιμότητα της πλατφόρμας.



Εικόνα 4.16: Διάγραμμα Λογικής Δομής – Συστατικά Συστήματος

4.3 Ορισμένες σημαντικές μέθοδοι

Η view_thesis είναι μια από τις πιο βασικές συναρτήσεις του συστήματος, καθώς φορτώνει την πλήρη σελίδα πτυχιακής τόσο για τον καθηγητή όσο και για τον φοιτητή. Περιλαμβάνει:

- Έλεγχο δικαιώματος πρόσβασης (teacher ή student)
- Ανάκτηση όλων των βασικών δεδομένων της πτυχιακής: τίτλος, περιγραφή, σημειώσεις
- Ανάκτηση όλων των μηνυμάτων και των απαντήσεών τους
- Φόρτωση όλων των συνδεδεμένων αρχείων
- Αν ο χρήστης είναι καθηγητής, εμφανίζει και τη λίστα των φοιτητών που συμμετέχουν

Αποτελεί τη βασική σελίδα εργασίας και συνεργασίας μεταξύ μελών.

```
@app.route('/thesis/<int:id>')
def view_thesis(id):
    # Αν δεν είναι συνδεδεμένος ο χρήστης, ανακατεύθυνση
    if 'user_id' not in session:
        return redirect(url_for('welcome'))

    user_id = session['user_id']
    role = session['role']

    con = get_db_connection()
    cur = con.cursor(dictionary=True)

    # Ανάλογα με τον ρόλο, επιλέγουμε την πτυχιακή μόνο αν έχει πρόσβαση
    if role == 'teacher':
        cur.execute("SELECT * FROM theses WHERE id = %s AND teacher_id = %s", (id, user_id))
    elif role == 'student':
        cur.execute("""
            SELECT theses.* FROM theses
            JOIN thesis_students ON thesis_students.thesis_id = theses.id
            WHERE theses.id = %s AND thesis_students.student_id = %s
        """, (id, user_id))
    else:
        con.close()
        return "Μη έγκυρος ρόλος.", 403

    thesis = cur.fetchone()
    if not thesis:
        con.close()
        flash("Δεν έχετε πρόσβαση σε αυτή την πτυχιακή.", "danger")
        return redirect(url_for('dashboard'))

    # Ανάκτηση κύριων μηνυμάτων (parent_id IS NULL)
    cur.execute("""
        SELECT m.*, u.email AS user_email
        FROM messages m
        JOIN users u ON m.user_id = u.id
        WHERE m.thesis_id = %s AND m.parent_id IS NULL
        ORDER BY m.timestamp ASC
    """, (id,))
    messages = cur.fetchall()

    # Για κάθε κύριο μήνυμα, φέρνουμε τις απαντήσεις του
    for msg in messages:
        cur.execute("""
            SELECT m.*, u.email AS user_email
            FROM messages m
            JOIN users u ON m.user_id = u.id
            WHERE m.parent_id = %s
            ORDER BY m.timestamp ASC
        """, (msg['id'],))
        msg['replies'] = cur.fetchall()

    # Ανάκτηση αρχείων που έχουν συνδεθεί με την πτυχιακή
    cur.execute("SELECT id, filename, filepath FROM files WHERE thesis_id = %s", (id,))
    raw_files = cur.fetchall()
    files = []
    for f in raw_files:
        files.append({
            'id': f['id'],
            'filename': f['filename'],
            'filepath': f['filepath'],
            'uploaded_by': f.get('uploaded_by'),
            'url': url_for('static', filename='uploads/' + f['filepath'])
        })

    # Αν ο χρήστης είναι καθηγητής, φέρνουμε και τη λίστα φοιτητών
    students = []
```

```
    if role == 'teacher':
        cur.execute("""
            SELECT users.id, users.email FROM users
            JOIN thesis_students ON thesis_students.student_id = users.id
            WHERE thesis_students.thesis_id = %s
        """, (id,))
        students = cur.fetchall()

    con.close()

    # Απόδοση τελικού template
    return render_template('thesis_view.html',
        thesis=thesis, messages=messages, files=files, students=students)
```

Η add_student είναι καθοριστική για τη λειτουργικότητα της πλατφόρμας, καθώς επιτρέπει στον επιβλέποντα καθηγητή να προσθέσει έναν φοιτητή σε πτυχιακή εργασία με βάση το email του. Αν το email δεν αντιστοιχεί σε ήδη καταχωρημένο φοιτητή, τότε:

- Δημιουργείται αυτόματα νέος λογαριασμός χρήστη με ρόλο student
- Ο φοιτητής συνδέεται με την πτυχιακή μέσω του πίνακα thesis_students

Η συνάρτηση αυτή είναι ιδιαίτερα φιλική προς τον καθηγητή και δεν απαιτεί προεγγραφή των φοιτητών από διαχειριστή.

```
@app.route('/thesis/<int:id>/add-student', methods=['GET', 'POST'])
def add_student(id):
    # Εάν δεν είναι συνδεδεμένος ή δεν είναι teacher, απορρίπτεται
    if 'user_id' not in session or session['role'] != 'teacher':
        return redirect(url_for('welcome'))

    con = get_db_connection()
    cur = con.cursor(dictionary=True)

    # Έλεγχος αν η πτυχιακή ανήκει στον τρέχοντα καθηγητή
    cur.execute("SELECT * FROM theses WHERE id = %s AND teacher_id = %s", (id, session['user_id']))
    thesis = cur.fetchone()
    if not thesis:
        con.close()
        flash("Δεν έχετε πρόσβαση σε αυτή την πτυχιακή.", "danger")
        return redirect(url_for('dashboard'))

    if request.method == 'POST':
        email = request.form['email'].strip().lower()

        # Ψάχνουμε αν υπάρχει ήδη φοιτητής με το email
        cur.execute("SELECT id FROM users WHERE email = %s", (email,))
        student = cur.fetchone()

        if not student:
            # Αν δεν υπάρχει, δημιουργείται νέος χρήστης (φοιτητής)
            cur.execute("""
                INSERT INTO users (email, role, activated)
                VALUES (%s, 'student', 0)
            """, (email,))
            con.commit()
            student_id = cur.lastrowid
            flash("Ο φοιτητής δημιουργήθηκε και προστέθηκε στην πτυχιακή.", "success")
        else:
            student_id = student['id']
            # Έλεγχος αν είναι ήδη στην ίδια πτυχιακή
```

```
            cur.execute("SELECT * FROM thesis_students WHERE thesis_id = %s AND student_id = %s", (id, student_id))
            exists = cur.fetchone()
            if exists:
                flash("Ο φοιτητής είναι ήδη καταχωρημένος.", "warning")
                con.close()
                return redirect(url_for('view_thesis', id=id))

        # Σύνδεση φοιτητή με την πτυχιακή
        cur.execute("""
            INSERT INTO thesis_students (thesis_id, student_id, created_at, updated_at)
            VALUES (%s, %s, NOW(), NOW())
        """, (id, student_id))
        con.commit()
        con.close()

        return redirect(url_for('view_thesis', id=id))

    # GET request → εμφάνιση φόρμας
    con.close()
    return render_template('add_student_form.html', thesis_id=id)
```

Αυτή η συνάρτηση συνδυάζει λειτουργίες δημιουργίας χρήστη, έλεγχου ύπαρξης, συσχέτισης με την πτυχιακή και τελική ενημέρωση της βάσης.

Το σύστημα συνομιλίας αποτελεί έναν από τους πιο βασικούς πυλώνες της πλατφόρμας, επιτρέποντας την συνεχή, καταγεγραμμένη και οργανωμένη επικοινωνία ανάμεσα στους φοιτητές και τον επιβλέποντα καθηγητή. Η υλοποίηση είναι τύπου forum-thread, με:

- Κεντρικά μηνύματα (χωρίς parent)
- Απαντήσεις έως και 1 επίπεδο βάθους
- Εμφάνιση αποστολέα, ημερομηνίας και ώρας για κάθε μήνυμα
- Δυνατότητα απάντησης άμεσα κάτω από κάθε μήνυμα
- Αποστολή νέου σχολίου (μη σχετιζόμενο με άλλο)

Κάθε μήνυμα αποθηκεύεται σε βάση δεδομένων στον πίνακα messages ο οποίος περιλαμβάνει πεδία όπως content, user_id, timestamp, parent_id και thesis_id.

```
@app.route('/thesis/<int:id>/comment', methods=['POST'])
def add_comment(id):
    if 'user_id' not in session:
        return redirect(url_for('welcome'))

    content = request.form['content'].strip()
    user_id = session['user_id']

    if content:
        con = get_db_connection()
        cur = con.cursor()
        cur.execute("""
            INSERT INTO messages (thesis_id, user_id, content, timestamp)
            VALUES (%s, %s, %s, NOW())
        """, (id, user_id, content))
```

```
    con.commit()
    con.close()

return redirect(url_for('view_thesis', id=id))
```

Κεφάλαιο 5ο: Τα συμπεράσματα – κάποιες βελτιώσεις

Η εργασία αποτέλεσε ένα ουσιαστικό παράδειγμα εφαρμογής τεχνολογιών στον χώρο της εκπαιδευτικής πληροφορικής. Μέσα από την υλοποίηση της εφαρμογής επιτεύχθηκε ο βασικός στόχος: η δημιουργία ενός φιλικού, επεκτάσιμου και αποδοτικού περιβάλλοντος για να επιτρέπει την αλληλεπίδραση φοιτητών και επιβλεπόντων κατά τη διάρκεια της συνεργασίας τους για την εκπόνηση πτυχιακής εργασίας.

Το σύστημα κατάφερε να συνδυάσει με επιτυχία τη λειτουργικότητα με τη χρηστικότητα. Η διαχείριση των πτυχιακών από τους καθηγητές πραγματοποιείται με εύκολο και άμεσο τρόπο ενώ η συμμετοχή των φοιτητών στην επικοινωνία και στην ανταλλαγή αρχείων γίνεται σε πραγματικό χρόνο με απόλυτη διαφάνεια. Ο σχεδιασμός του γραφικού περιβάλλοντος βασισμένος στο Bootstrap framework, συνέβαλε σημαντικά στη δημιουργία ενός σύγχρονου και οπτικά ελκυστικού περιβάλλοντος εργασίας.

Σε επίπεδο τεχνολογιών η χρήση του Flask για τον backend αποδείχθηκε ιδιαίτερα αποτελεσματική, καθώς προσέφερε την απαραίτητη ευελιξία για την ταχεία υλοποίηση των βασικών λειτουργιών του συστήματος. Η απουσία foreign key περιορισμών στην βάση δεδομένων εξυπηρέτησε την απλοποίηση της ανάπτυξης, αν και σε μελλοντικές εκδόσεις αυτό μπορεί να ενισχυθεί με επιπλέον ελέγχους ακεραιότητας. Να σημειωθεί ότι χρησιμοποιήθηκε το chat ai για τη διόρθωση κειμένου και να μας προτείνει διάφορα στοιχεία για την εργασία.

Παρακάτω θα δούμε κάποιες προτάσεις βελτίωσης.

Επαλήθευση email και ενεργοποίηση λογαριασμού

Η υλοποίηση βασίζεται σε σύνδεσμο επαναφοράς κωδικού μέσω email. Αν και λειτουργική μια πιο ολοκληρωμένη ροή εγγραφής, με σύστημα επιβεβαίωσης και επιβεβαιωμένο activation, θα πρόσθετε ένα επίπεδο ασφάλειας και αξιοπιστίας.

Πολλαπλοί επιβλέποντες

Αυτή τη στιγμή, κάθε πτυχιακή εργασία μπορεί να έχει μόνο έναν επιβλέποντα. Η επέκταση της βάσης ώστε να υποστηρίζει και συν-επιβλέποντες ή reviewers θα μπορούσε να καλύψει πληρέστερα τις ανάγκες πραγματικών εκπαιδευτικών προγραμμάτων.

Αναφορές – Export Δεδομένων

Η δυνατότητα εξαγωγής δεδομένων (π.χ. ιστορικό επικοινωνίας, λίστα αρχείων, λίστα φοιτητών) σε μορφή PDF ή Excel θα διευκόλυνε την τεκμηρίωση και αρχειοθέτηση των πτυχιακών.

Αρχειοθέτηση / Ολοκλήρωση Πτυχιακής

Δεν υπάρχει ακόμη κάποιος μηχανισμός που να δηλώνει ότι μια πτυχιακή έχει ολοκληρωθεί. Μια κατάσταση "Ολοκληρωμένη" ή "Υποβληθείσα" θα βοηθούσε στην οργάνωση των ενεργών και παλαιών εργασιών.

Πιο εξελιγμένο Chat

Αν και το υπάρχον σύστημα συνομιλίας καλύπτει βασικές ανάγκες, θα μπορούσε να εμπλουτιστεί με:

- Υποστήριξη επισύναψης αρχείων μέσα στο chat
- Ειδοποιήσεις σε πραγματικό χρόνο
- Επιλογή επεξεργασίας ή διαγραφής μηνυμάτων από τον συγγραφέα ή τον επιβλέποντα

Responsive Notifications / Δραστηριότητα

Ένα σύστημα ειδοποιήσεων που θα εμφανίζει νέα αρχεία, νέα σχόλια ή αλλαγές σε πτυχιακές εργασίες (π.χ. νέος φοιτητής) θα βελτίωνε την ενημέρωση των χρηστών χωρίς να απαιτείται συνεχής παρακολούθηση της σελίδας.

Η πλατφόρμα αποτελεί ένα έτοιμο προς χρήση εργαλείο συνεργασίας για πτυχιακές εργασίες. Δίνει όλα τα απαραίτητα εργαλεία, είναι φιλική προς τον χρήστη και θέτει τη βάση για περαιτέρω βελτιώσεις. Οι επεκτάσεις που προτάθηκαν μπορούν να υλοποιηθούν σταδιακά χωρίς να αλλάξουν τη βασική λειτουργία της εφαρμογής.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] https://asana.com/industry/education

[2] https://trello.com/

[3] https://www.microsoft.com/en-us/education/products/teams

[4] https://www.python.org/about/gettingstarted/

[5] https://realpython.com/

[6] https://www.programiz.com/python-programming/online-compiler/

[7] https://en.wikipedia.org/wiki/Flask_(web_framework)

[8] https://www.tutorialspoint.com/flask/index.htm

[9] https://dev.to/detimo/python-flask-pros-and-cons-1mlo

[10] https://en.wikipedia.org/wiki/MariaDB

[11] https://aws.amazon.com/compare/the-difference-between-mariadb-vs-mysql/

[12] https://mariadb.com/database-topics/mariadb-vs-mysql/

ΠΑΡΑΡΤΗΜΑ ΚΩΔΙΚΑ Α

Κώδικες που χρησιμοποιήθηκαν.

App.py

```
from flask import Flask, request, redirect, render_template, flash, url_for, session
from db_config import get_db_connection
import secrets
from datetime import datetime, timedelta
import smtplib
from email.message import EmailMessage
from werkzeug.security import generate_password_hash, check_password_hash
import os
from werkzeug.utils import secure_filename

app = Flask(__name__)
app.secret_key = 'your_secret_key_here'

UPLOAD_FOLDER = os.path.join('static', 'uploads')
ALLOWED_EXTENSIONS = {'pdf', 'docx', 'txt', 'png', 'jpg', 'jpeg'}
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

reset_tokens = {}

def send_reset_email(to_email, link):
    msg = EmailMessage()
    msg['Subject'] = 'Πλατφόρμα Πτυχιακών - Επαναφορά Κωδικού'
    msg['From'] = 'your_email@gmail.com'
    msg['To'] = to_email
    msg.set_content(f'Πατήστε στον παρακάτω σύνδεσμο για να ορίσετε νέο κωδικό:\n{link}')
    try:
        with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:
            smtp.login('your_email@gmail.com', 'your_app_password')
            smtp.send_message(msg)
        print(f"[OK] Email στάλθηκε στο {to_email}")
    except Exception as e:
        print(f"[ΣΦΑΛΜΑ] Αποστολής Email: {e}")

@app.route('/')
def welcome():

    #print(f"pass: {generate_password_hash('1234')}")
    return render_template('welcome.html')

@app.route('/request-reset', methods=['POST'])
def request_reset():
    email = request.form['email'].strip().lower()
    con = get_db_connection()
    cur = con.cursor()
    cur.execute("""
        SELECT users.id FROM users
        JOIN thesis_students ON thesis_students.student_id = users.id
        WHERE users.email = %s AND users.role = 'student'
```

```
    """, (email,))
    result = cur.fetchone()
    con.close()

    if result:
        token = secrets.token_urlsafe(32)
        expiry = datetime.utcnow() + timedelta(hours=1)
        reset_tokens[token] = (email, expiry)
        reset_link = url_for('reset_password', token=token, _external=True)
        send_reset_email(email, reset_link)
        flash("Αν το email είναι έγκυρο, στάλθηκε σύνδεσμος για ορισμό κωδικού.", "success")
    else:
        flash("Δεν βρέθηκε ενεργή πτυχιακή για αυτό το email ή δεν έχει δηλωθεί ακόμα.", "danger")

    return redirect(url_for('welcome'))

@app.route('/reset/<token>', methods=['GET', 'POST'])
def reset_password(token):
    data = reset_tokens.get(token)
    if not data:
        return "Ο σύνδεσμος δεν είναι έγκυρος ή έχει λήξει.", 400

    email, expiry = data
    if datetime.utcnow() > expiry:
        del reset_tokens[token]
        return "Ο σύνδεσμος έχει λήξει.", 400

    if request.method == 'POST':
        password = request.form['password']
        hashed_pw = generate_password_hash(password)
        con = get_db_connection()
        cur = con.cursor()
            cur.execute("UPDATE users SET password = %s, activated = 1 WHERE email = %s",
(hashed_pw, email))
        con.commit()
        con.close()

        del reset_tokens[token]
        flash("Ορίστηκε νέος κωδικός με επιτυχία. Μπορείτε να συνδεθείτε.", "success")
        return redirect(url_for('welcome'))

    return render_template('reset.html')

@app.route('/login', methods=['POST'])
def login():
    email = request.form['email'].strip().lower()
    password = request.form['password']
    print(f"[DEBUG] Προσπάθεια σύνδεσης με email: {email}")

    con = get_db_connection()
    cur = con.cursor()
    cur.execute("SELECT id, password, role FROM users WHERE email = %s", (email,))
    row = cur.fetchone()
    con.close()
```

```
    if row:
        print(f"[DEBUG] Χρήστης βρέθηκε: id={row[0]}, role={row[2]}")
        if 1: #check_password_hash(row[1], password):
            session['user_id'] = row[0]
            session['role'] = row[2]
            flash("Επιτυχής σύνδεση!", "success")
            return redirect(url_for('dashboard'))
        else:
            print("[DEBUG] Μη έγκυρος κωδικός.")
    else:
        print("[DEBUG] Δεν βρέθηκε χρήστης με αυτό το email.")

    flash("Λάθος email ή κωδικός.", "danger")
    return redirect(url_for('welcome'))

@app.route('/logout')
def logout():
    session.clear()
    flash("Αποσυνδεθήκατε με επιτυχία.", "info")
    return redirect(url_for('welcome'))

@app.route('/dashboard')
def dashboard():
    if 'user_id' not in session:
        return redirect(url_for('welcome'))

    user_id = session['user_id']
    role = session['role']

    con = get_db_connection()
    cur = con.cursor(dictionary=True)

    if role == 'teacher':
        cur.execute("SELECT * FROM theses WHERE teacher_id = %s", (user_id,))
        theses = cur.fetchall()

        for thesis in theses:
            cur.execute("""
                SELECT users.email FROM users
                JOIN thesis_students ON thesis_students.student_id = users.id
                WHERE thesis_students.thesis_id = %s
            """, (thesis['id'],))
            thesis['students'] = cur.fetchall()

        con.close()
        return render_template('dashboard_teacher.html', theses=theses)

    elif role == 'student':
        cur.execute("""
            SELECT theses.* FROM theses
            JOIN thesis_students ON thesis_students.thesis_id = theses.id
            WHERE thesis_students.student_id = %s
            LIMIT 1
        """, (user_id,))
        thesis = cur.fetchone()
```

```
        con.close()
        return render_template('dashboard_student.html', thesis=thesis)

    else:
        return "Μη έγκυρος ρόλος.", 403

@app.route('/thesis/create', methods=['GET', 'POST'])
def create_thesis():
    if 'user_id' not in session or session['role'] != 'teacher':
        return redirect(url_for('welcome'))

    if request.method == 'POST':
        title = request.form['title']
        description = request.form['description']
        code = request.form['code']
        notes = request.form['notes']
        teacher_id = session['user_id']

        con = get_db_connection()
        cur = con.cursor()
        cur.execute("""
            INSERT INTO theses (code, title, description, teacher_id, notes, created_at, updated_at)
            VALUES (%s, %s, %s, %s, %s, NOW(), NOW())
        """, (code, title, description, teacher_id, notes))
        con.commit()
        con.close()

        flash("Η πτυχιακή δημιουργήθηκε με επιτυχία.", "success")
        return redirect(url_for('dashboard'))

    return render_template('thesis_form.html', mode='create')

@app.route('/thesis/<int:id>/edit', methods=['GET', 'POST'])
def edit_thesis(id):
    if 'user_id' not in session or session['role'] != 'teacher':
        return redirect(url_for('welcome'))

    con = get_db_connection()
    cur = con.cursor(dictionary=True)
      cur.execute("SELECT * FROM theses WHERE id = %s AND teacher_id = %s", (id,
session['user_id']))
    thesis = cur.fetchone()

    if not thesis:
        con.close()
        flash("Δεν βρέθηκε η πτυχιακή.", "danger")
        return redirect(url_for('dashboard'))

    if request.method == 'POST':
        title = request.form['title']
        description = request.form['description']
        code = request.form['code']
        notes = request.form['notes']

        cur.execute("""
```

```
            UPDATE theses
            SET title = %s, description = %s, code = %s, notes = %s, updated_at = NOW()
            WHERE id = %s
        """, (title, description, code, notes, id))
        con.commit()
        con.close()

        flash("Η πτυχιακή ενημερώθηκε.", "success")
        return redirect(url_for('dashboard'))

    con.close()
    return render_template('thesis_form.html', mode='edit', thesis=thesis)

@app.route('/thesis/<int:id>')
def view_thesis(id):
    if 'user_id' not in session:
        return redirect(url_for('welcome'))

    user_id = session['user_id']
    role = session['role']

    con = get_db_connection()
    cur = con.cursor(dictionary=True)

    # Βεβαιώσου ότι ο χρήστης έχει πρόσβαση στη συγκεκριμένη πτυχιακή
    if role == 'teacher':
        cur.execute("SELECT * FROM theses WHERE id = %s AND teacher_id = %s", (id, user_id))
    elif role == 'student':
        cur.execute("""
            SELECT theses.* FROM theses
            JOIN thesis_students ON thesis_students.thesis_id = theses.id
            WHERE theses.id = %s AND thesis_students.student_id = %s
        """, (id, user_id))
    else:
        con.close()
        return "Μη έγκυρος ρόλος.", 403

    thesis = cur.fetchone()
    if not thesis:
        con.close()
        flash("Δεν έχετε πρόσβαση σε αυτή την πτυχιακή.", "danger")
        return redirect(url_for('dashboard'))

    # Φέρε τα κύρια μηνύματα
    cur.execute("""
        SELECT m.*, u.email AS user_email
        FROM messages m
        JOIN users u ON m.user_id = u.id
        WHERE m.thesis_id = %s AND m.parent_id IS NULL
        ORDER BY m.timestamp ASC
    """, (id,))
    messages = cur.fetchall()

    # Φέρε τις απαντήσεις για κάθε μήνυμα
    for msg in messages:
```

```
        cur.execute("""
            SELECT m.*, u.email AS user_email
            FROM messages m
            JOIN users u ON m.user_id = u.id
            WHERE m.parent_id = %s
            ORDER BY m.timestamp ASC
        """, (msg['id'],))
        msg['replies'] = cur.fetchall()

    # Φέρε τα αρχεία
    cur.execute("SELECT id, filename, filepath FROM files WHERE thesis_id = %s", (id,))
    raw_files = cur.fetchall()
    files = []
    for f in raw_files:
        files.append({
            'id': f['id'],
            'filename': f['filename'],
            'filepath': f['filepath'],
            'uploaded_by': f.get('uploaded_by'),
            'url': url_for('static', filename='uploads/' + f['filepath'])
        })

    students = []
    if role == 'teacher':
        cur.execute("""
            SELECT users.id, users.email FROM users
            JOIN thesis_students ON thesis_students.student_id = users.id
            WHERE thesis_students.thesis_id = %s
        """, (id,))
        students = cur.fetchall()

    con.close()
        return render_template('thesis_view.html', thesis=thesis, messages=messages, files=files,
students=students)

@app.route('/thesis/<int:thesis_id>/student/<int:student_id>/remove')
def remove_student(thesis_id, student_id):
    if 'user_id' not in session or session['role'] != 'teacher':
        return redirect(url_for('welcome'))

    con = get_db_connection()
    cur = con.cursor()
    # Επιβεβαίωση ότι η πτυχιακή ανήκει στον χρήστη
     cur.execute("SELECT id FROM theses WHERE id = %s AND teacher_id = %s", (thesis_id,
session['user_id']))
    if not cur.fetchone():
        con.close()
        flash("Δεν έχετε δικαίωμα να αλλάξετε αυτή την πτυχιακή.", "danger")
        return redirect(url_for('dashboard'))

     cur.execute("DELETE FROM thesis_students WHERE thesis_id = %s AND student_id = %s",
(thesis_id, student_id))
    con.commit()
    con.close()
```

```
    flash("Ο φοιτητής αφαιρέθηκε.", "info")
    return redirect(url_for('view_thesis', id=thesis_id))

@app.route('/thesis/<int:id>/comment', methods=['POST'])
def comment_thesis(id):
    if 'user_id' not in session:
        return redirect(url_for('welcome'))

    user_id = session['user_id']
    content = request.form['content'].strip()

    if not content:
        flash("Το περιεχόμενο δεν μπορεί να είναι κενό.", "warning")
        return redirect(url_for('view_thesis', id=id))

    con = get_db_connection()
    cur = con.cursor()
    cur.execute("""
            INSERT INTO messages (thesis_id, user_id, content, parent_id, timestamp, created_at,
updated_at)
        VALUES (%s, %s, %s, NULL, NOW(), NOW(), NOW())
    """, (id, user_id, content))
    con.commit()
    con.close()

    flash("Το μήνυμα προστέθηκε.", "success")
    return redirect(url_for('view_thesis', id=id))

@app.route('/thesis/<int:id>/reply', methods=['POST'])
def reply_to_message(id):
    if 'user_id' not in session:
        return redirect(url_for('welcome'))

    user_id = session['user_id']
    content = request.form['content'].strip()
    parent_id = request.form.get('parent_id')

    if not content or not parent_id:
        flash("Το μήνυμα δεν μπορεί να είναι κενό.", "warning")
        return redirect(url_for('view_thesis', id=id))

    con = get_db_connection()
    cur = con.cursor()
    cur.execute("""
            INSERT INTO messages (thesis_id, user_id, content, parent_id, timestamp, created_at,
updated_at)
        VALUES (%s, %s, %s, %s, NOW(), NOW(), NOW())
    """, (id, user_id, content, parent_id))
    con.commit()
    con.close()

    flash("Η απάντηση προστέθηκε.", "success")
    return redirect(url_for('view_thesis', id=id))

def allowed_file(filename):
```

```
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS

@app.route('/thesis/<int:id>/upload', methods=['POST'])
def upload_file(id):
    if 'user_id' not in session:
        return redirect(url_for('welcome'))

    file = request.files.get('file')
    if not file or file.filename == '':
        flash("Δεν επιλέχθηκε αρχείο.", "warning")
        return redirect(url_for('view_thesis', id=id))

    if allowed_file(file.filename):
        filename = secure_filename(file.filename)
        filepath = f"{id}_{filename}"
        full_path = os.path.join(app.config['UPLOAD_FOLDER'], filepath)
        file.save(full_path)

        con = get_db_connection()
        cur = con.cursor()
        cur.execute("""
            INSERT INTO files (thesis_id, filename, filepath, uploaded_by, timestamp, created_at,
updated_at)
            VALUES (%s, %s, %s, %s, NOW(), NOW(), NOW())
        """, (id, filename, filepath, session['user_id']))
        con.commit()
        con.close()

        flash("Το αρχείο αποθηκεύτηκε.", "success")
    else:
        flash("Μη επιτρεπτός τύπος αρχείου.", "danger")

    return redirect(url_for('view_thesis', id=id))

@app.route('/thesis/<int:thesis_id>/file/<int:file_id>/delete')
def delete_file(thesis_id, file_id):
    if 'user_id' not in session:
        return redirect(url_for('welcome'))

    user_id = session['user_id']
    role = session['role']

    con = get_db_connection()
    cur = con.cursor(dictionary=True)
    cur.execute("SELECT * FROM files WHERE id = %s AND thesis_id = %s", (file_id, thesis_id))
    file = cur.fetchone()

    if not file:
        con.close()
        flash("Το αρχείο δεν βρέθηκε.", "danger")
        return redirect(url_for('view_thesis', id=thesis_id))

    if file['uploaded_by'] != user_id and role != 'teacher':
        con.close()
        flash("Δεν έχετε δικαίωμα διαγραφής του αρχείου.", "danger")
```

```
        return redirect(url_for('view_thesis', id=thesis_id))

    # Διαγραφή αρχείου από δίσκο
    try:
        full_path = os.path.join(app.root_path, app.config['UPLOAD_FOLDER'], file['filepath'])
        if os.path.exists(full_path):
            os.remove(full_path)
        else:
            print(f"[WARN] Αρχείο δεν βρέθηκε στο σύστημα: {full_path}")
    except Exception as e:
        print(f"[WARN] Αποτυχία διαγραφής αρχείου: {e}")

    # Διαγραφή από βάση
    cur.execute("DELETE FROM files WHERE id = %s", (file_id,))
    con.commit()
    con.close()

    flash("Το αρχείο διαγράφηκε.", "success")
    return redirect(url_for('view_thesis', id=thesis_id))

@app.route('/thesis/<int:id>/add-student', methods=['GET', 'POST'])
def add_student(id):
    if 'user_id' not in session or session['role'] != 'teacher':
        return redirect(url_for('welcome'))

    con = get_db_connection()
    cur = con.cursor(dictionary=True)

    # Επιβεβαίωση ότι ο χρήστης είναι ιδιοκτήτης της πτυχιακής
    cur.execute("SELECT * FROM theses WHERE id = %s AND teacher_id = %s", (id,
session['user_id']))
    thesis = cur.fetchone()
    if not thesis:
        con.close()
        flash("Δεν έχετε πρόσβαση σε αυτή την πτυχιακή.", "danger")
        return redirect(url_for('dashboard'))

    if request.method == 'POST':
        email = request.form['email'].strip().lower()

        # Ψάχνουμε τον φοιτητή στη βάση
        cur.execute("SELECT id FROM users WHERE email = %s", (email,))
        student = cur.fetchone()

        if not student:
            # Δημιουργία νέου φοιτητή
            cur.execute("""
                INSERT INTO users (email, role, activated)
                VALUES (%s, 'student', 0)
            """, (email,))
            con.commit()
            student_id = cur.lastrowid
            flash("Ο φοιτητής δημιουργήθηκε και προστέθηκε στην πτυχιακή.", "success")
        else:
            student_id = student['id']
```

```
            # Έλεγχος αν είναι ήδη στην πτυχιακή
              cur.execute("SELECT * FROM thesis_students WHERE thesis_id = %s AND student_id =
%s", (id, student_id))
            exists = cur.fetchone()
            if exists:
                flash("Ο φοιτητής είναι ήδη καταχωρημένος.", "warning")
                con.close()
                return redirect(url_for('view_thesis', id=id))

        # Σύνδεση φοιτητή με την πτυχιακή
        cur.execute("""
            INSERT INTO thesis_students (thesis_id, student_id, created_at, updated_at)
            VALUES (%s, %s, NOW(), NOW())
        """, (id, student_id))
        con.commit()
        con.close()

        return redirect(url_for('view_thesis', id=id))

    con.close()
    return render_template('add_student_form.html', thesis_id=id)

@app.route('/thesis/<int:id>/delete')
def delete_thesis(id):
    if 'user_id' not in session or session['role'] != 'teacher':
        return redirect(url_for('welcome'))

    con = get_db_connection()
    cur = con.cursor()

    # Επιβεβαίωση ότι η πτυχιακή ανήκει στον χρήστη
      cur.execute("SELECT * FROM theses WHERE id = %s AND teacher_id = %s", (id,
session['user_id']))
    thesis = cur.fetchone()

    if not thesis:
        con.close()
        flash("Δεν βρέθηκε η πτυχιακή ή δεν έχετε δικαίωμα.", "danger")
        return redirect(url_for('dashboard'))

    # Φέρνουμε τα αρχεία για να τα διαγράψουμε από τον δίσκο
    cur.execute("SELECT filepath FROM files WHERE thesis_id = %s", (id,))
    files = cur.fetchall()

    for f in files:
        full_path = os.path.join(app.root_path, app.config['UPLOAD_FOLDER'], f[0])
        if os.path.exists(full_path):
            os.remove(full_path)

    # Διαγραφές βάσης
    cur.execute("DELETE FROM thesis_students WHERE thesis_id = %s", (id,))
    cur.execute("DELETE FROM messages WHERE thesis_id = %s", (id,))
    cur.execute("DELETE FROM files WHERE thesis_id = %s", (id,))
    cur.execute("DELETE FROM theses WHERE id = %s", (id,))
    con.commit()
```

```
    con.close()

    flash("Η πτυχιακή διαγράφηκε οριστικά.", "info")
    return redirect(url_for('dashboard'))

if __name__ == '__main__':
    app.run(debug=True)
```

thesis_view.py

```
{% extends "base.html" %}

{% block title %}Πτυχιακή{% endblock %}

{% block content %}

<div class="row">

    <div class="col-lg-8">

        <div class="mb-4">

            <h3>{{ thesis.title }}</h3>

            <p class="text-muted">Κωδικός: {{ thesis.code }}</p>

            <p>{{ thesis.description }}</p>

        </div>


        <h5 class="mb-3">Συζήτηση</h5>

        {% for msg in messages %}

        <div class="card mb-3">

            <div class="card-body">

                 <p><strong>{{ msg.user_email }}</strong> <small class="text-muted">({{ msg.timestamp
}})</small></p>

                <p>{{ msg.content }}</p>


                {% for reply in msg.replies %}

                <div class="card mt-2 ms-4">

                    <div class="card-body">

                            <p><strong>{{ reply.user_email }}</strong> <small class="text-muted">({{
reply.timestamp }})</small></p>

                        <p>{{ reply.content }}</p>

                    </div>

                </div>

                {% endfor %}


                <form method="POST" action="/thesis/{{ thesis.id }}/reply" class="mt-2 ms-4">
```

```
                    <input type="hidden" name="parent_id" value="{{ msg.id }}">
                    <div class="input-group">
                       <input type="text" name="content" class="form-control" placeholder="Απάντηση..."
required>
                        <button class="btn btn-outline-primary" type="submit">Απάντηση</button>
                    </div>
                </form>
            </div>
        </div>
        {% endfor %}

        <form method="POST" action="/thesis/{{ thesis.id }}/comment">
            <div class="mb-3">
                <label for="content" class="form-label">Νέο Μήνυμα</label>
                <textarea class="form-control" name="content" rows="3" required></textarea>
            </div>
            <button type="submit" class="btn btn-primary">Αποστολή</button>
        </form>
    </div>
    <div class="col-lg-4">
        {% if session['role'] == 'teacher' %}
<div class="card mb-4">
    <div class="card-body">
        <h5><i class="fas fa-users"></i> Φοιτητές</h5>
        <ul class="list-group list-group-flush">
            {% for s in students %}
            <li class="list-group-item d-flex justify-content-between align-items-center">
                {{ s.email }}
                <a href="{{ url_for('remove_student', thesis_id=thesis.id, student_id=s.id) }}"
                   class="btn btn-sm btn-danger"
                   onclick="return confirm('Διαγραφή φοιτητή;');">
                    <i class="fas fa-user-minus"></i>
                </a>
            </li>
            {% endfor %}
        </ul>
        <a href="{{ url_for('add_student', id=thesis.id) }}" class="btn btn-outline-primary mt-3">
            <i class="fas fa-user-plus"></i> Προσθήκη Φοιτητή
        </a>
    </div>
</div>
```

```
{% endif %}


        <div class="card mb-4">
            <div class="card-body">
                <h5><i class="fas fa-sticky-note"></i> Σημειώσεις Επιβλέποντα</h5>
                <div class="alert alert-info">{{ thesis.notes }}</div>
            </div>
        </div>


        <div class="card mb-4">
            <div class="card-body">
                <h5><i class="fas fa-paperclip"></i> Αρχεία</h5>
                <ul class="list-group list-group-flush">
                    {% for file in files %}
                    <li class="list-group-item d-flex justify-content-between align-items-center">
                        <a href="{{ file.url }}" target="_blank">{{ file.filename }}</a>
                        {% if session['user_id'] == file.uploaded_by or session['role'] == 'teacher' %}
                         <a href="/thesis/{{ thesis.id }}/file/{{ file.id }}/delete" class="btn btn-sm
btn-danger" onclick="return confirm('Διαγραφή αρχείου;')">
                            <i class="fas fa-trash-alt"></i>
                        </a>
                        {% endif %}
                    </li>
                    {% else %}
                    <li class="list-group-item text-muted">Δεν υπάρχουν αρχεία.</li>
                    {% endfor %}
                </ul>
                  <form method="POST" action="/thesis/{{ thesis.id }}/upload" enctype="multipart/form-
data" class="mt-3">
                    <div class="input-group">
                        <input type="file" name="file" class="form-control" required>
                             <button type="submit" class="btn btn-outline-success"><i class="fas fa-
upload"></i></button>
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
{% endblock %}
```