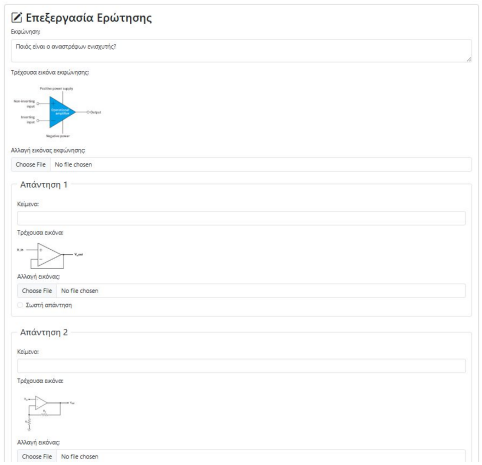
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Σύστημα διαχείρισης εξετάσεων και δημιουργίας διαδικτυακών δοκιμασιών προόδου»



Του φοιτητή

518230

ΕΥΑΓΓΕΛΟΣ ΤΣΑΚΙΡΗΣ

Επιβλέπων

Δρ. Κυριάκος Τσιακμάκης

Ιούνιος 2025

Σύστημα διαχείρισης εξετάσεων και δημιουργίας διαδικτυακών δοκιμασιών προόδου

Κωδικός: 25155

Φοιτητής: Τσακίρης Ευάγγελος

Εισηγητής: Δρ Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 08-03-2025

Ημερομηνία περάτωσης Π.Ε. 29-05-2025

Περίληψη

Αυτή η εργασία αφορά την ανάπτυξη ενός διαδικτυακού συστήματος διαχείρισης εξετάσεων και quiz, σχεδιασμένου για χρήση από καθηγητές και φοιτητές. Το σύστημα αφορά τη δημιουργία μαθημάτων από τον διαχειριστή, την ανάθεση αυτών σε καθηγητές και την κατασκευή εξετάσεων με ερωτήσεις πολλαπλής επιλογής οι οποίες μπορούν να περιλαμβάνουν εικόνες. Οι φοιτητές συμμετέχουν στις εξετάσεις μέσω μοναδικού κωδικού (UID), απαντούν στις ερωτήσεις και υποβάλλουν τις απαντήσεις τους και αυτές καταγράφονται και αξιολογούνται αυτόματα. Το σύστημα αναπτύχθηκε με χρήση Python Flask, MySQL και Bootstrap και έχει φιλικό και responsive περιβάλλον. Υποστηρίζει την εξαγωγή αποτελεσμάτων, την παρακολούθηση στατιστικών και την ασφαλή καταγραφή κάθε υποβολής. Η εφαρμογή μπορεί να φιλοξενηθεί τοπικά ή σε web server και αποτελεί μια καλή και αξιόπιστη λύση για εκπαιδευτικά ιδρύματα που επιθυμούν να ενσωματώσουν ψηφιακά εργαλεία αξιολόγησης.

« Exam Management System and Online Progress Quiz Creation Platform»

Abstract

This work concerns the development of an online exam and quiz management system, designed for use by teachers and students. The system concerns the administrator to create courses, assign them to teachers, and build exams with multiple choice questions which can include images. Students participate in the exams via a unique code (UID), answer the questions and submit their answers, which are automatically recorded and evaluated. The system was developed using Python Flask, MySQL and Bootstrap, offering a friendly and responsive environment. It supports the export of results, monitoring statistics and the secure recording of each submission. The application can be hosted locally or on a web server and is a flexible and reliable solution for educational institutions that wish to integrate digital assessment tools.

Ευχαριστίες

Με την ολοκλήρωση αυτής της εργασίας θα ήθελα να εκφράσω την ευγνωμοσύνη σε όλους όσοι με στήριξαν και με καθοδήγησαν σε κάθε στάδιο αυτής της προσπάθειας, τους γονείς μου και τον επιβλέποντα καθηγητή μου, για την καθοδήγηση, τις διορθώσεις και την υποστήριξη του.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Η διαχείριση εξετάσεων και η δημιουργία διαδικτυακών δοκιμασιών προόδου αποτελούν ένα μέρος της σύγχρονης εκπαιδευτικής διαδικασίας. Η τεχνολογία εξελίσσεται και το διαδίκτυο καθίσταται όλο και πιο προσβάσιμο και υπάρχει ανάγκη για ψηφιοποιημένα, ασφαλή περιβάλλοντα αξιολόγησης και αυτό έχει αναδειχθεί ως κεντρικό ζήτημα σε όλα τα επίπεδα εκπαίδευσης – από τα σχολεία μέχρι τα πανεπιστήμια και τα κέντρα δια βίου μάθησης. Οι παραδοσιακές μορφές γραπτής εξέτασης αντικαθίστανται ή συνδυάζονται πλέον με ηλεκτρονικές μορφές και προσφέρουν δυνατότητες όπως άμεση βαθμολόγηση, αυτοματοποίηση διαδικασιών, διαφάνεια και καταγραφή.

Οι διαδικτυακές δοκιμασίες προόδου δεν λειτουργούν μόνο ως εργαλεία αξιολόγησης αλλά και ως μέσα ανατροφοδότησης τόσο για τους φοιτητές όσο και για τους εκπαιδευτικούς. Επιτρέπουν την καλύτερη παρακολούθηση της πορείας των μαθητών, τη διαπίστωση θεμάτων κατανόησης και τη στοχευμένη παρέμβαση σε σημεία αδυναμίας. Δίνουν τη δυνατότητα για πιο ευέλικτο προγραμματισμό εξετάσεων, προσαρμογή των ερωτήσεων ανά φοιτητή και χρήση διαδραστικού περιεχομένου (όπως εικόνες, βίντεο ή animation) για τη βελτίωση της μαθησιακής εμπειρίας.

Η ευκολία πρόσβασης σε τέτοια συστήματα μέσω υπολογιστών, tablets ή κινητών τηλεφώνων τα κάνει ιδιαίτερα φιλικά για υβριδική ή εξ αποστάσεως εκπαίδευση. Οι τεχνικές απαιτήσεις δεν είναι πλέον τόσο μεγάλες και πολλές πλατφόρμες λειτουργούν μέσα από τον browser χωρίς να χρειάζεται εγκατάσταση ενός λογισμικού. Αυτό προϋποθέτει την ύπαρξη ενός κατάλληλα σχεδιασμένου συστήματος που να καλύπτει τις ανάγκες των εκπαιδευτικών και να είναι απλό στη χρήση από τους φοιτητές και να εξασφαλίζει την σωστή διαδικασία της εξέτασης.

Η ασφάλεια των δεδομένων και η αποτροπή φαινομένων αντιγραφής είναι επίσης κρίσιμα ζητήματα στη σχεδίαση τέτοιων συστημάτων. Πολλές εφαρμογές χρησιμοποιούν μεθόδους όπως μοναδικά κλειδιά εισόδου, χρονόμετρα, τυχαία σειρά ερωτήσεων ή ελεγχόμενη πρόσβαση και άλλες βασίζονται στην εποπτεία ή την αυτοματοποιημένη ανίχνευση συμπεριφορών. Όλα αυτά τα χαρακτηριστικά δεν εξαφανίζουν εντελώς την πιθανότητα παρατυπιών αλλά αυξάνουν σε μεγάλο βαθμό τη διαφάνεια και την αξιοπιστία της διαδικασίας.

Η ανάγκη για ανάπτυξη προσαρμοσμένων λύσεων που βασίζονται σε ανοικτό λογισμικό και καλύπτουν τις ειδικές απαιτήσεις ενός εκπαιδευτικού φορέα είναι αναγκαία. Η δημιουργία ενός δικού μας συστήματος διαχείρισης quiz και εξετάσεων μάς επιτρέπει να ελέγξουμε πλήρως τη λειτουργικότητα, την εμφάνιση, την ασφάλεια και τη δομή του και να προσαρμόζουμε στις ανάγκες των εκπαιδευτών και των φοιτητών ενός συγκεκριμένου ιδρύματος ή περιβάλλοντος.

Το σύστημα που αναπτύξαμε αποτελεί μία λειτουργική διαδικτυακή πλατφόρμα για τη δημιουργία, διαχείριση και υλοποίηση εξετάσεων τύπου quiz. Είναι σχεδιασμένο ώστε να είναι προσβάσιμο από καθηγητές και φοιτητές μέσα από έναν απλό browser και να υποστηρίζει με αποτελεσματικότητα την αμφίδρομη αλληλεπίδραση μεταξύ των δύο πλευρών. Ο σχεδιασμός του είναι δομημένος γύρω από τρεις βασικούς ρόλους: τον διαχειριστή (admin), τον διδάσκοντα (teacher) και τον φοιτητή (student). Κάθε ρόλος έχει πρόσβαση σε διαφορετικές λειτουργίες, μέσω ενός απλού responsive και φιλικού προς το χρήστη περιβάλλοντος.

Ο διαχειριστής του συστήματος έχει τη δυνατότητα να δημιουργεί μαθήματα και να τα αναθέτει σε καθηγητές. Οι καθηγητές, με τη σειρά τους, μπορούν να διαχειρίζονται τα δικά τους μαθήματα, να δημιουργούν quiz για κάθε μάθημα, να προσθέτουν ερωτήσεις (με ή χωρίς εικόνες), να καθορίζουν χρονικά όρια και να δημοσιεύουν τις εξετάσεις στους φοιτητές με έναν μοναδικό πενταψήφιο κωδικό (UID). Η διαδικασία δημιουργίας quiz είναι απλή αλλά δυναμική και επιτρέπει κάθε καθηγητή να ελέγχει πλήρως τη δομή, το περιεχόμενο και το πλαίσιο κάθε αξιολόγησης.

Οι φοιτητές εισέρχονται στο σύστημα και με τη χρήση του UID αποκτούν πρόσβαση σε μία συγκεκριμένη εξέταση. Οι ερωτήσεις παρουσιάζονται σε μορφή πολλαπλής επιλογής, με δυνατότητα ενσωμάτωσης εικόνων τόσο στην εκφώνηση όσο και στις απαντήσεις. Μόλις ολοκληρωθεί η εξέταση και υποβληθούν οι απαντήσεις, το σύστημα αποθηκεύει τα δεδομένα με ακρίβεια και ενημερώνει τον διδάσκοντα. Οι φοιτητές δεν μπορούν να τροποποιήσουν τις απαντήσεις τους μετά την υποβολή, εξασφαλίζοντας έτσι τη διαφάνεια και την εγκυρότητα της αξιολόγησης.

Το σύστημα υποστηρίζει αυτόματη βαθμολόγηση, καταγραφή υποβολών, εξαγωγή συγκεντρωτικών αποτελεσμάτων και παρουσίαση στατιστικών ανά quiz. Οι καθηγητές μπορούν να δουν αναλυτικά ποιοι φοιτητές υπέβαλαν απαντήσεις, ποιο ήταν το σκορ τους και ποιο το ποσοστό επιτυχίας και τα δεδομένα μπορούν να εξαχθούν σε αρχείο CSV για περαιτέρω ανάλυση.

Η υλοποίηση βασίστηκε στη γλώσσα προγραμματισμού Python με το framework Flask για το backend, τη MySQL για τη βάση δεδομένων, και το Bootstrap για τον σχεδιασμό του frontend. Η αρχιτεκτονική είναι απλή και χωρίς εξαρτήσεις από εξωτερικά APIs, με στόχο την ευκολία εγκατάστασης, την αυτονομία και τη δυνατότητα τοπικής λειτουργίας. Όλα τα δεδομένα αποθηκεύονται στη MySQL και η δομή της βάσης έχει σχεδιαστεί με ευελιξία ώστε να υποστηρίζει επέκταση στο μέλλον, όπως τράπεζες ερωτήσεων, κατηγορίες θεματολογίας, τυχαία επιλογή ή επανάληψη quiz.

Το σύστημα μπορεί να φιλοξενηθεί τόσο τοπικά όσο και σε cloud server. Είναι έτοιμο για λειτουργία σε κάθε εκπαιδευτικό περιβάλλον, από μικρές σχολικές τάξεις μέχρι πανεπιστημιακά ιδρύματα. Η δομή του επιτρέπει την εύκολη διαχείριση χρηστών και περιεχομένου και δίνει προσοχή στην ασφάλεια, στους ρόλους και στην απλότητα στη χρήση.

Η πλατφόρμα που δημιουργήσαμε είναι ένα καλό εργαλείο εκπαιδευτικής αξιολόγησης. Με την υποστήριξη πολλαπλών ρόλων, την οργανωμένη διαχείριση quiz, την άμεση καταγραφή

αποτελεσμάτων και το φιλικό περιβάλλον διεπαφής παρέχει τις βασικές απαιτήσεις μιας σύγχρονης εκπαιδευτικής δοκιμασίας και μπορεί να επεκταθεί περαιτέρω στο μέλλον ανάλογα με τις ανάγκες του ιδρύματος ή του εκπαιδευτή.

Η εργασία ξεκινά με το Κεφάλαιο 1 που έχει τίτλο «Εισαγωγή» και περιλαμβάνει δύο υποενότητες: την κύρια εισαγωγή και την παρουσίαση της δομής του κειμένου.
Το Κεφάλαιο 2 εξετάζει Παρόμοια Συστήματα και περιλαμβάνει αναλυτικές παρουσιάσεις για τα Moodle Quiz Module, TCExam και LibreLMS / Open eClass.
Στο Κεφάλαιο 3 αναλύεται η τεχνολογική βάση του έργου. Ξεκινά με την Python για την ανάπτυξη της εφαρμογής, ακολουθεί η MySQL για τη δημιουργία της βάσης δεδομένων και στη συνέχεια η χρήση CSS/Bootstrap για τον αισθητικό σχεδιασμό.
Το Κεφάλαιο 4 εστιάζει στο ίδιο το σύστημα που υλοποιήθηκε. Περιλαμβάνει αναλυτική περιγραφή της εφαρμογής, της βάσης δεδομένων, της ιστοσελίδας, του κώδικα των βασικών λειτουργιών και της ασφάλειας.
Το Κεφάλαιο 5 παρουσιάζει τα Συμπεράσματα και τις Προτάσεις Βελτίωσης του συστήματος.

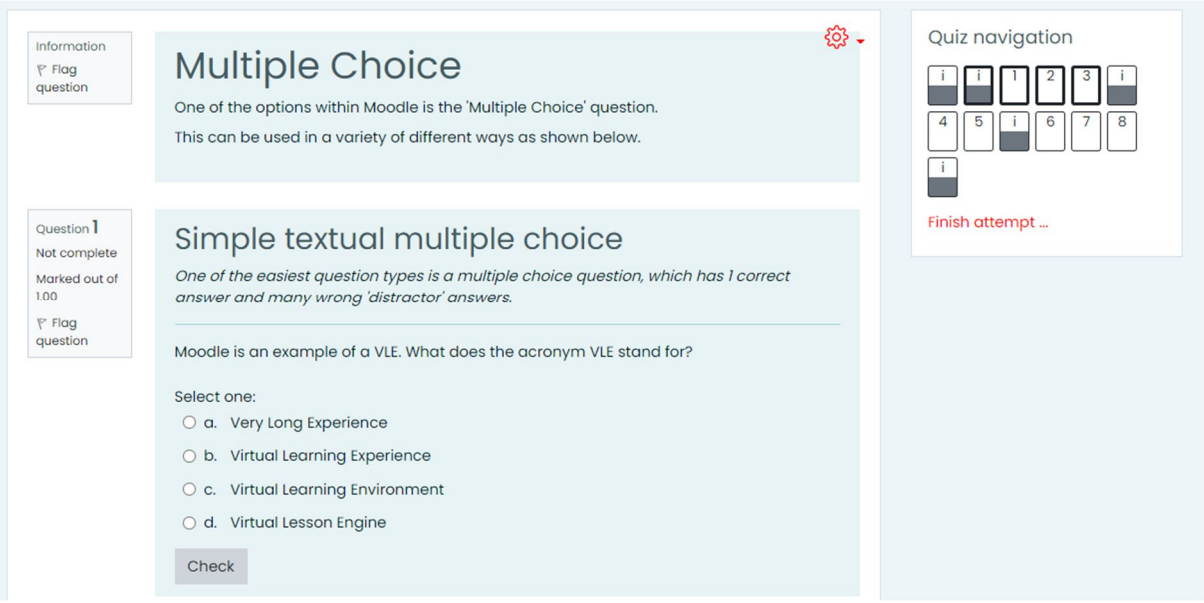
Η εργασία ολοκληρώνεται με τη Βιβλιογραφία και το Παράρτημα Κώδικα.

Κεφάλαιο 2ο: Παρόμοια συστήματα

2.1 Moodle Quiz Module

Το Moodle (Modular Object-Oriented Dynamic Learning Environment) είναι ένα από τα δημοφιλέστερα ανοιχτού κώδικα Συστήματα Διαχείρισης Μάθησης (LMS). Δημιουργήθηκε από τον Martin Dougiamas στις αρχές της δεκαετίας του 2000 με στόχο την υποστήριξη της εκπαιδευτικής διαδικασίας μέσω διαδικτύου. Από τότε εξελίχθηκε σε μια πλήρη πλατφόρμα που χρησιμοποιείται σήμερα από πανεπιστήμια, σχολεία, ιδιωτικούς οργανισμούς και εταιρείες σε περισσότερες από 200 χώρες. [1]

Το Moodle έχει σχεδιαστεί για να υποστηρίζει την ηλεκτρονική εκπαίδευση με ασφάλεια, επεκτασιμότητα και ευελιξία. Είναι εντελώς δωρεάν στη βασική του μορφή, επιτρέποντας την εγκατάσταση σε ιδιωτικούς servers χωρίς περιορισμούς ενώ υποστηρίζεται από μία τεράστια παγκόσμια κοινότητα εκπαιδευτικών, προγραμματιστών και χρηστών. Η πλατφόρμα επιτρέπει τη δημιουργία διαδικτυακών μαθημάτων, με υλικό όπως αρχεία, βίντεο, διαλέξεις, κουίζ, ανακοινώσεις, ημερολόγια, αξιολογήσεις και πολλά ακόμη εργαλεία.



Εικόνα 2.1: Moodle – Quiz

[https://davefoord.wordpress.com/wp-content/uploads/2021/10/moodlequiz.png]

Μία από τις πιο ισχυρές δυνατότητες του Moodle είναι η υποστήριξη ψηφιακών εξετάσεων και quiz, είτε για πρακτικούς λόγους είτε για βαθμολογημένες εξετάσεις. Το υποσύστημα quiz είναι πλήρως ενσωματωμένο και έχει εργαλεία για καθηγητές και φοιτητές που ξεπερνούν τις βασικές λειτουργίες με προηγμένα σενάρια διδασκαλίας, αξιολόγησης και στατιστικής ανάλυσης.

Η λειτουργικότητα του Moodle Quiz Module παρουσιάζει πολλές ομοιότητες με το σύστημα που υλοποιήσαμε στο έργο μας. Και στις δύο περιπτώσεις υπάρχει η δυνατότητα για έναν καθηγητή να δημιουργήσει quiz μέσα από μια εύχρηστη διεπαφή, να προσθέσει ερωτήσεις με ή χωρίς εικόνες, να καθορίσει χρονικά πλαίσια και να παρακολουθήσει τις επιδόσεις των φοιτητών. Το Moodle γενικά έχει μεγαλύτερη κλίμακα και υποστηρίζει περισσότερους τύπους ερωτήσεων, μηχανισμούς για τυχαία ροή και ισχυρά στατιστικά εργαλεία.

Στο έργο μας ο καθηγητής μπορεί να δημιουργήσει quiz με ερωτήσεις πολλαπλής επιλογής και να βάλει προθεσμίες κάτι που είναι άμεσα διαθέσιμο και στο Moodle και με επιπλέον δυνατότητες όπως αρνητική βαθμολογία για τις λάθος απαντήσεις ή ορισμό χρονικών περιορισμών ανά φοιτητή. Η πλατφόρμα μας επιτρέπει την αποθήκευση εικόνων για εκφωνήσεις και απαντήσεις. το ίδιο προσφέρει και το Moodle μέσω του κειμενογράφου του με δυνατότητα μεταφόρτωσης αρχείων πολυμέσων.

Η παρακολούθηση των αποτελεσμάτων και η εμφάνιση βαθμολογίας και η δυνατότητα εξαγωγής σε αρχείο CSV υπάρχει και στις δύο πλατφόρμες αλλά στο Moodle αυτή η λειτουργία είναι πιο μεγάλη σε δυνατότητες με επιλογές για εξαγωγή αναλυτικών αναφορών ανά ερώτηση, ανά φοιτητή ή ανά κατηγορία. Το Moodle ακόμη έχει τράπεζα ερωτήσεων δηλαδή μία συλλογή ερωτήσεων που μπορεί να χρησιμοποιηθεί ξανά σε διαφορετικά quiz κάτι που θα μπορούσε να ενσωματωθεί και στο δικό μας σύστημα ως μελλοντική βελτίωση.

Στον τομέα της ασφάλειας και της πρόσβασης και τα δύο συστήματα περιλαμβάνουν έλεγχο ταυτότητας με ρόλους (π.χ. φοιτητής, καθηγητής, διαχειριστής), ενώ το Moodle προσθέτει δυνατότητες όπως proctoring, χρονόμετρα, ελεγχόμενη πλοήγηση και υποστήριξη για εξωτερικές εφαρμογές εποπτείας.

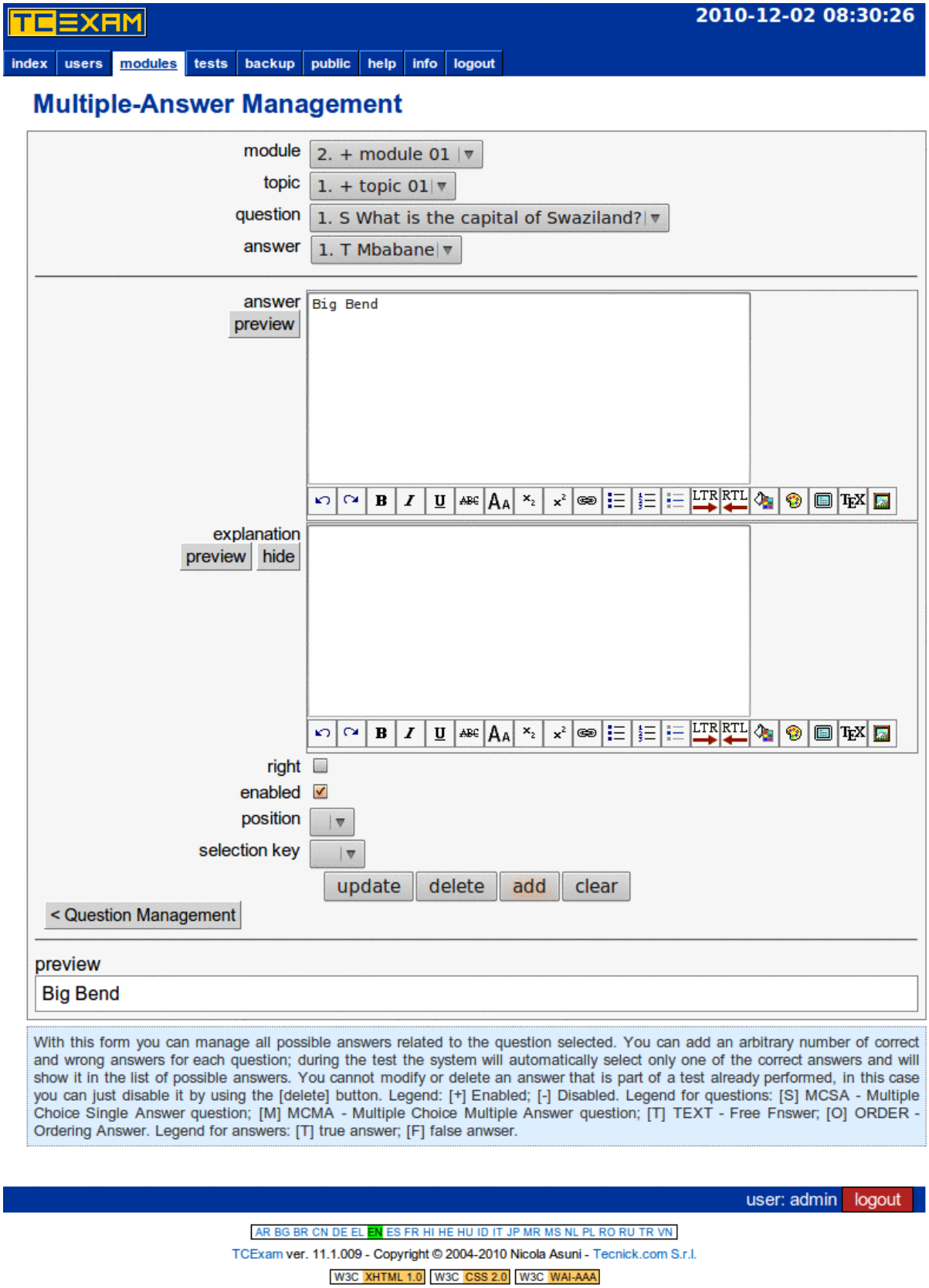
Σε ό,τι αφορά την τεχνολογία το Moodle είναι γραμμένο σε PHP και χρησιμοποιεί MySQL ή PostgreSQL για βάση δεδομένων. Το σύστημα μας που βασίζεται σε Python και MySQL με τη δυνατότητα μελλοντικής διασύνδεσης ή μεταφοράς περιεχομένου με μετατροπή δεδομένων.

Το Moodle αποτελεί ένα καλό και πλήρως λειτουργικό σύστημα που καλύπτει τις ανάγκες εκπαιδευτικών quiz και μπορεί να θεωρηθεί ως ένα ισχυρό σημείο σύγκρισης με το δικό μας έργο

το οποίο διατηρεί την απλότητα στις βασικές λειτουργίες με πιο ελαφριά και ελεγχόμενη εμπειρία χρήστη.

2.2 TCExam

Το TCExam είναι ένα πλήρως ανοιχτού κώδικα λογισμικό για τη δημιουργία, διαχείριση και διενέργεια εξετάσεων μέσω διαδικτύου. Αναπτύχθηκε με στόχο να παρέχει ένα ανεξάρτητο και επεκτάσιμο περιβάλλον για εκπαιδευτικά ιδρύματα, δημόσιες υπηρεσίες και ιδιώτες που επιθυμούν να αξιολογήσουν χρήστες μέσω ηλεκτρονικών τεστ. Το σύστημα είναι γραμμένο σε PHP και χρησιμοποιεί MySQL ή PostgreSQL ως βάση δεδομένων, γεγονός που το καθιστά εύκολα εγκαταστάσιμο σε κάθε περιβάλλον με web server, όπως Apache.[2]



Εικόνα 2.2: TCExam quiz

[https://tcexam.org/img/quick_guide/en/tcexam_ftguide_022.png]

Σε αντίθεση με το Moodle που είναι ένα πλήρες LMS (Learning Management System) το TCExam εστιάζει αποκλειστικά στην ψηφιακή αξιολόγηση και είναι κατάλληλο για όσους χρειάζονται ένα σύστημα επικεντρωμένο στις εξετάσεις, χωρίς περιττές λειτουργίες μαθημάτων ή διαλέξεων. Η πλατφόρμα προσφέρει web-based περιβάλλον για τη διαχείριση ερωτήσεων, τη δημιουργία τεστ, τον προγραμματισμό εξετάσεων, την καταγραφή υποβολών και την εξαγωγή αναφορών και στατιστικών.

Το TCExam δεν απαιτεί ειδική γνώση για να χρησιμοποιηθεί. Ο administrator μπορεί να δημιουργήσει γρήγορα θεματολόγια, να ορίσει ομάδες ερωτήσεων, να κατηγοριοποιήσει υλικό, να αναθέσει χρήστες σε εξετάσεις και να επιτρέψει την εκτέλεση των τεστ μέσω οποιουδήποτε browser. Διαθέτει υποστήριξη για διαφορετικές γλώσσες, τύπους ερωτήσεων (πολλαπλής επιλογής, αντιστοίχισης, αριθμητικές κ.λπ.), τυχαία σειρά απαντήσεων και χρονικά όρια, ενώ ταυτόχρονα μπορεί να χρησιμοποιηθεί ακόμα και offline σε τοπικά δίκτυα.

Το TCExam παρουσιάζει πολλές ομοιότητες με το έργο μας αφού και τα δύο επικεντρώνονται στη διαχείριση και διενέργεια quiz και εξετάσεων μέσω web. Στο σύστημά μας οι καθηγητές έχουν τη δυνατότητα να δημιουργούν quiz με πολλαπλές ερωτήσεις και απαντήσεις, να ορίζουν προθεσμίες, να παρακολουθούν αποτελέσματα και να εξάγουν δεδομένα. Το TCExam έχει όλες αυτές τις δυνατότητες με περισσότερες επιλογές εξατομίκευσης.

Και στις δύο πλατφόρμες οι χρήστες χωρίζονται σε ρόλους — στο δικό μας σύστημα υπάρχουν οι admin, καθηγητές και φοιτητές, ενώ στο TCExam υπάρχουν administrators, συγγραφείς θεμάτων, υποψήφιοι και επόπτες. Η φιλοσοφία είναι κοινή, ο καθηγητής (ή administrator) δημιουργεί τα τεστ και οι εξεταζόμενοι απαντούν σε αυτά μέσα από μια ασφαλή, online διεπαφή.

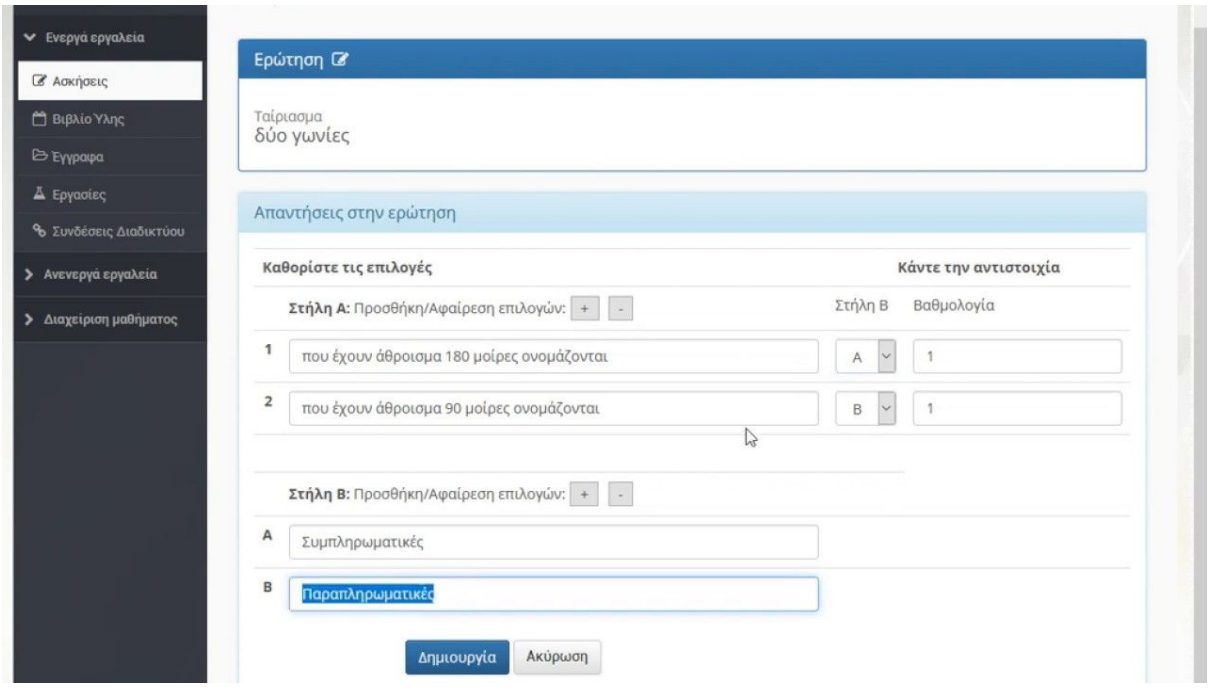
Στο TCExam υπάρχει δυνατότητα ομαδοποίησης ερωτήσεων και θεμάτων κάτι που αντιστοιχεί σε μελλοντική επέκταση του δικού μας συστήματος με θεματικές ενότητες ή βάσεις ερωτήσεων. Το TCExam υποστηρίζει βαθμολόγηση με ποσοστά, ορισμό δυσκολίας ανά ερώτηση, χρονόμετρα και τυχαία επιλογή ερωτήσεων ανά συμμετοχή, χαρακτηριστικά που επίσης μπορούν να ενσωματωθούν στο δικό μας έργο.

Ένα άλλο κοινό στοιχείο είναι η εξαγωγή αποτελεσμάτων. Στην πλατφόρμα μας ο καθηγητής μπορεί να δει τα αποτελέσματα φοιτητών και να τα εξάγει σε CSV. Το TCExam υποστηρίζει εξαγωγή σε HTML, XML, PDF και CSV με δυνατότητα οπτικής απεικόνισης μέσω γραφημάτων.

Τεχνολογικά και τα δύο συστήματα βασίζονται σε MySQL, πράγμα που σημαίνει ότι η δομή της βάσης είναι παρόμοια και κατανοητή, με πίνακες για χρήστες, εξετάσεις, ερωτήσεις, απαντήσεις και υποβολές. Αν και το TCExam είναι γραμμένο σε PHP και όχι σε Python, το μοντέλο λειτουργίας του είναι συμβατό με αυτό που υλοποιήσαμε.

2.3 LibreLMS / Open eClass

Το Open eClass είναι ένα ελληνικής κατασκευής σύστημα διαχείρισης ηλεκτρονικών μαθημάτων (LMS), το οποίο αναπτύχθηκε αρχικά από το Ακαδημαϊκό Διαδίκτυο GUnet και χρησιμοποιείται ευρέως στα ελληνικά πανεπιστήμια, ΤΕΙ, σχολεία και ιδιωτικούς φορείς. Είναι ανοιχτού κώδικα, πλήρως δωρεάν και παρέχει στους διδάσκοντες τη δυνατότητα να δημιουργούν διαδικτυακά μαθήματα με διαλέξεις, έγγραφα, επικοινωνία, δραστηριότητες και φυσικά quiz και ασκήσεις αξιολόγησης.[3]



Εικόνα 2.3: Open eClass quiz

[https://i.ytimg.com/vi/7GYL7AA7_4g/maxresdefault.jpg]

Η πλατφόρμα υποστηρίζει ελληνικά πλήρως και είναι φιλική προς τον τελικό χρήστη και αυτό την κάνει ιδανική για το ελληνικό ακαδημαϊκό χώρο. Η εγκατάστασή της μπορεί να γίνει τοπικά σε server ενός ιδρύματος, επιτρέποντας τον πλήρη έλεγχο των δεδομένων, χωρίς να απαιτείται

σύνδεση με εξωτερικές υπηρεσίες. Το Open eClass ακολουθεί τις προδιαγραφές SCORM για την ενσωμάτωση εκπαιδευτικού υλικού, διαθέτει εργαλεία διαχείρισης ομάδων, φόρουμ, ανταλλαγής αρχείων, ημερολογίων, wiki και πολλά ακόμη.

Ένα από τα πιο χρήσιμα υποσυστήματα του Open eClass είναι εκείνο της διαχείρισης ασκήσεων/quiz, το οποίο χρησιμοποιείται για τη δημιουργία εξετάσεων, τεστ αυτοαξιολόγησης ή εβδομαδιαίων δραστηριοτήτων. Οι διδάσκοντες μπορούν να δημιουργούν ερωτήσεις, να ορίζουν σωστές απαντήσεις, να επιλέγουν αν θα υπάρχει χρονόμετρο ή όχι και να διαχειρίζονται τις προσπάθειες των φοιτητών.

Η λειτουργία των quiz στο Open eClass είναι σε μεγάλο βαθμό παρόμοια με αυτή του έργου μας. Και στις δύο πλατφόρμες, ο καθηγητής μπορεί να δημιουργήσει εξετάσεις, να προσθέσει ερωτήσεις πολλαπλής επιλογής και να παρακολουθεί τη συμμετοχή των φοιτητών. Στο Open eClass όπως και στη δική μας πλατφόρμα, οι εξετάσεις είναι προσβάσιμες μέσω ενός συγκεκριμένου περιβάλλοντος και απαιτούν αυθεντικοποίηση του χρήστη για να καταγράφονται σωστά τα δεδομένα της υποβολής.

Στο έργο μας κάθε quiz συνοδεύεται από ένα μοναδικό UID, το οποίο δίνεται στους φοιτητές για να ξεκινήσουν την εξέταση. Το Open eClass υλοποιεί κάτι αντίστοιχο μέσω της επιλογής «ορατότητα μαθήματος» ή «ενεργοποίηση δραστηριότητας», χωρίς όμως να χρησιμοποιεί ρητά UID. Οι προθεσμίες, οι απόπειρες και ο αυτόματος υπολογισμός βαθμολογίας υπάρχουν και στις δύο πλατφόρμες. Στην υλοποίησή μας, το quiz βαθμολογείται αυτόματα με βάση τις σωστές απαντήσεις, και αυτό συμβαίνει και στο Open eClass, που υποστηρίζει επίσης το φιλτράρισμα αποτελεσμάτων ανά φοιτητή ή ερώτηση.

Και τα δύο συστήματα υποστηρίζουν αποθήκευση αρχείων όπως εικόνες σε ερωτήσεις ή απαντήσεις. Στο έργο μας, κάθε ερώτηση μπορεί να έχει εικόνα, και το ίδιο υποστηρίζεται στο eClass μέσω του WYSIWYG editor και της αποθήκευσης πολυμέσων. Η διαφορά είναι ότι το έργο μας ενσωματώνει αυτή τη δυνατότητα πιο απλά και άμεσα, μέσα από αρχεία τύπου `upload`, χωρίς να βασίζεται σε HTML σύνταξη.

Σε σχέση με τους ρόλους, και οι δύο πλατφόρμες διακρίνουν τον διαχειριστή, τον διδάσκοντα και τον φοιτητή. Στο έργο μας, αυτή η διάκριση ενσωματώνεται στη βάση και ελέγχεται μέσω του session. Στο eClass, ο διαχειριστής μπορεί να καθορίσει δικαιώματα με πιο περίπλοκο σύστημα ρυθμίσεων.

Η διαχείριση των αποτελεσμάτων στο Open eClass περιλαμβάνει στατιστικά, εξαγωγή σε Excel και αναλυτικές αναφορές. Στο έργο μας ο καθηγητής μπορεί να δει τη λίστα των συμμετοχών, τη βαθμολογία και να εξάγει αποτελέσματα σε CSV αρχείο.

Η πλατφόρμα μας υλοποιεί με απλό τρόπο όλα τα βασικά που προσφέρει το eClass στο κομμάτι της εξέτασης, χωρίς όμως τη βαριά υποδομή ενός πλήρους LMS. Αυτό κάνει τη δική μας λύση πιο ευέλικτη, πιο εύκολη στην προσαρμογή και πιο φιλική για μικρές ομάδες ή ανεξάρτητους καθηγητές.

Κεφάλαιο 3ο: Python και άλλα

3.1 Python για τη δημιουργία του έργου μας

Η γλώσσα προγραμματισμού Python αποτέλεσε τη βάση για την υλοποίηση ολόκληρου του backend της πλατφόρμας δημιουργίας και διαχείρισης quiz που αναπτύχθηκε στο πλαίσιο του παρόντος έργου. Η Python είναι μια από τις πιο διαδεδομένες και ευέλικτες γλώσσες προγραμματισμού στον κόσμο, με ιστορία που ξεκινά στις αρχές της δεκαετίας του 1990. Δημιουργήθηκε από τον Guido van Rossum και σχεδιάστηκε με στόχο να είναι απλή, αναγνώσιμη και ταυτόχρονα ισχυρή για να μπορεί να αξιοποιηθεί από αρχάριους αλλά και από έμπειρους προγραμματιστές σε σύνθετα έργα λογισμικού. Με την πάροδο των ετών η Python εξελίχθηκε σε μία από τις βασικές επιλογές για ανάπτυξη διαδικτυακών εφαρμογών, επιστημονικό υπολογισμό, μηχανική μάθηση, αυτοματισμούς συστημάτων και πολλά ακόμη. [4-5]

Ένα από τα σημαντικότερα πλεονεκτήματα της Python είναι η καθαρή της σύνταξη η οποία βοηθά στην επίλυση του προβλήματος και όχι στη σύνταξη του κώδικα. Οι εντολές είναι κατανοητές ακόμη και σε μη ειδικούς κάτι που διευκολύνει την ομαδική εργασία και την τεκμηρίωση. Η Python διαθέτει επίσης ένα τεράστιο πλήθος από έτοιμες βιβλιοθήκες και εργαλεία που αυτά καλύπτουν όλες τις ανάγκες ενός σύγχρονου έργου πληροφορικής, από διασύνδεση με βάσεις δεδομένων μέχρι τη δημιουργία web εφαρμογών, API, διαγράμματα, διαχείριση χρηστών και πολλά ακόμη. Στο συγκεκριμένο έργο αξιοποιήθηκε η βιβλιοθήκη Flask για το web backend που επιτρέπει την εύκολη δημιουργία διαδρομών (routes), την αποστολή δεδομένων από και προς HTML σελίδες και τη διαχείριση της συνεδρίας κάθε χρήστη.

Η Python χρησιμοποιήθηκε στο έργο μας ως το βασικό εργαλείο για την επικοινωνία του frontend με τη βάση δεδομένων. Μέσα από απλό και οργανωμένο κώδικα, διαχειριζόμαστε λειτουργίες όπως η δημιουργία quiz, η αποθήκευση ερωτήσεων και απαντήσεων, η υποβολή συμμετοχών από φοιτητές και η προβολή των αποτελεσμάτων από τους καθηγητές. Η Python χρησιμοποιείται και για λειτουργίες διαχειριστικής φύσης όπως η ανάθεση μαθημάτων σε καθηγητές από τον admin, ή η εξαγωγή των βαθμολογιών σε αρχείο CSV για περαιτέρω αξιολόγηση.

Η ευκολία με την οποία η Python διαχειρίζεται τη σύνδεση με βάσεις δεδομένων MySQL αποτέλεσε σημαντικό παράγοντα στην επιλογή της. Χρησιμοποιώντας τον connector της MySQL είναι δυνατή η εκτέλεση queries απευθείας από Python κώδικα και η μετατροπή των αποτελεσμάτων σε λεξικά ή λίστες για εύκολη επεξεργασία. Ο χειρισμός δεδομένων, η καταγραφή συμμετοχών, η αποθήκευση αρχείων εικόνας, η αντιστοίχιση φοιτητή με την υποβολή του και ο υπολογισμός ποσοστών επιτυχίας, γίνονται με ασφάλεια και αξιοπιστία στην Python.

Οσο αφορά την ασφάλεια η Python μας προσέφερε πολλά έτοιμα εργαλεία για τη διαχείριση sessions, την κρυπτογράφηση κωδικών, την αποτροπή επιθέσεων μέσω φορμών (π.χ. CSRF ή injection), και τη διαχείριση των αρχείων που ανεβαίνουν από τους χρήστες. Μπορούμε να ορίσουμε ποιος χρήστης έχει πρόσβαση σε ποιο τμήμα του συστήματος (π.χ. φοιτητής, καθηγητής, διαχειριστής) και αυτό κάνει τη γλώσσα ιδανική για την υλοποίηση πολύ - επίπεδων συστημάτων με ξεχωριστούς ρόλους και λειτουργίες.

Όσον αφορά το διαδίκτυο η Python υποστηρίζει την κατασκευή web εφαρμογών σε όλα τα επίπεδα. Είτε πρόκειται για ένα μικρό σύστημα ερωταπαντήσεων, είτε για μια πλήρη εφαρμογή με σύνδεση σε πολλαπλούς χρήστες, αρχεία, δικαιώματα, ασφάλεια και δυναμικές διεπαφές η Python μπορεί να το υλοποιήσει. Η παρούσα εφαρμογή μπορεί να επεκταθεί για χρήση μέσω cloud, με υποστήριξη για πολλαπλούς server και σύνδεση σε online βάσεις δεδομένων αφού το backend που έχει γραφεί είναι ευέλικτο και εύκολα παραμετροποιήσιμο.

Στη συγκεκριμένη πλατφόρμα η Python αποτέλεσε το βασικό στοιχείο πάνω στο οποίο χτίστηκε όλο το σύστημα. Οι HTML σελίδες συνεργάζονται άψογα με το backend, μέσω της τεχνολογίας Jinja2, η οποία είναι ενσωματωμένη στο Flask. Αυτό μας επέτρεψε να δημιουργήσουμε δυναμικές ιστοσελίδες που εμφανίζουν δεδομένα από τη βάση όπως τα quiz, τις ερωτήσεις, τις απαντήσεις, τις συμμετοχές των φοιτητών και τα τελικά αποτελέσματα. Κάθε διαδρομή στον κώδικα Python αντιστοιχεί σε μια συγκεκριμένη λειτουργία της εφαρμογής, δημιουργώντας έτσι έναν σαφή και κατανοητό χάρτη λειτουργιών για τον προγραμματιστή και τον χρήστη.

Η Python, επομένως, δεν ήταν απλώς μια επιλογή για την υλοποίηση του έργου αλλά καθοριστική. Έχει απλότητα, ευελιξία, ασφάλεια και επεκτασιμότητα και μα παρείχε τη δημιουργία ενός πλήρως λειτουργικού και αξιόπιστου συστήματος ψηφιακής αξιολόγησης.

3.2 MySQL για τη δημιουργία της βάσης μας

Η βάση δεδομένων είναι η "καρδιά" κάθε συστήματος που χρειάζεται να αποθηκεύει και να διαχειρίζεται πληροφορίες. Στο δικό μας έργο χρησιμοποιήσαμε τη MySQL, ένα από τα πιο γνωστά και διαδεδομένα συστήματα βάσεων δεδομένων στον κόσμο. Η MySQL είναι δωρεάν, γρήγορη και σταθερή, και χρησιμοποιείται ευρέως τόσο σε μικρές ιστοσελίδες όσο και σε πολύ μεγάλες εφαρμογές όπως το Facebook και το YouTube. Ο λόγος που την επιλέξαμε είναι επειδή είναι εύκολη στη χρήση, έχει πολύ καλή τεκμηρίωση και λειτουργεί άψογα με την Python. [6-7]

Η MySQL μάς βοήθησε να οργανώσουμε όλες τις πληροφορίες του συστήματος σε πίνακες. Κάθε πίνακας μοιάζει με ένα Excel, όπου αποθηκεύουμε συγκεκριμένα δεδομένα. Για παράδειγμα, έχουμε

πίνακα για τους χρήστες (users), για τα μαθήματα (courses), για τα quiz, τις ερωτήσεις, τις απαντήσεις και τις υποβολές των φοιτητών. Όλα αυτά τα δεδομένα βρίσκονται συγκεντρωμένα μέσα στη MySQL, και από εκεί τα διαβάζει η εφαρμογή μας ώστε να παρουσιάσει τη σωστή πληροφορία σε κάθε χρήστη.

Το ωραίο με τη MySQL είναι ότι μπορούμε να "μιλάμε" μαζί της χρησιμοποιώντας απλές εντολές SQL. Αυτές οι εντολές μας επιτρέπουν να αποθηκεύουμε νέες πληροφορίες (INSERT), να τις βλέπουμε (SELECT), να τις αλλάζουμε (UPDATE) ή να τις διαγράφουμε (DELETE). Μέσα από την Python, γράψαμε κώδικα που εκτελεί τέτοιες εντολές, έτσι ώστε κάθε φορά που ένας καθηγητής δημιουργεί quiz ή ένας φοιτητής απαντά σε ερωτήσεις όλα να αποθηκεύονται αυτόματα στη βάση.

Για λόγους απλότητας και ελευθερίας δεν χρησιμοποιήσαμε περιορισμούς τύπου foreign keys. Αυτό σημαίνει ότι εμείς ελέγχαμε στον κώδικα αν κάτι συνδέεται σωστά, π.χ. ότι μια ερώτηση ανήκει σε ένα υπαρκτό quiz. Αν και οι foreign keys μπορούν να προσφέρουν επιπλέον ασφάλεια, στο δικό μας σύστημα προτιμήσαμε να διαχειριζόμαστε εμείς αυτή τη λογική για να έχουμε περισσότερη ευελιξία.

Η MySQL υποστήριξε πολύ καλά όλες τις ανάγκες μας. Η βάση μας είναι δομημένη έτσι ώστε να επιτρέπει στους καθηγητές να βλέπουν μόνο τα μαθήματά τους, να δημιουργούν quiz για κάθε μάθημα, να προσθέτουν ερωτήσεις και απαντήσεις και στους φοιτητές να συμμετέχουν στις εξετάσεις και να αποθηκεύονται οι υποβολές τους. Όλα τα δεδομένα καταγράφονται με ακρίβεια και με τη βοήθεια SQL εντολών μπορούμε να τα φιλτράρουμε, να τα ταξινομούμε, να εξάγουμε αποτελέσματα και να τα παρουσιάζουμε στην ιστοσελίδα με εύκολο τρόπο.

Η MySQL μάς έδωσε ένα πολύ δυνατό και αξιόπιστο περιβάλλον αποθήκευσης για όλες τις λειτουργίες του έργου. Είναι εύκολη να εγκατασταθεί, συνεργάζεται άψογα με τον server μας και είναι έτοιμη να υποστηρίξει την πλατφόρμα ακόμα και αν μεγαλώσει πολύ στο μέλλον. Για όποιον ξεκινά ένα έργο που χρειάζεται βάση δεδομένων, η MySQL είναι μια εξαιρετική επιλογή που συνδυάζει απλότητα, ταχύτητα και δύναμη.

3.3 CSS – Bootstrap για την ομορφιά

Ο σχεδιασμός μιας ιστοσελίδας δεν αφορά μόνο το τι λειτουργίες προσφέρει, αλλά και το πώς φαίνεται και πώς αισθάνεται ο χρήστης όταν τη χρησιμοποιεί. Στο έργο μας επιλέξαμε να χρησιμοποιήσουμε την τεχνολογία Bootstrap, η οποία μας βοήθησε να κάνουμε το σύστημα πιο όμορφο, πιο ευανάγνωστο και πιο εύκολο στη χρήση, τόσο από φοιτητές όσο και από καθηγητές. Το Bootstrap είναι ένα έτοιμο σύνολο εργαλείων για σχεδιασμό ιστοσελίδων, βασισμένο στη γλώσσα CSS και μας επέτρεψε να δημιουργήσουμε γρήγορα και με απλό τρόπο μοντέρνες, καθαρές και responsive σελίδες. [8-9]

Ένα από τα μεγάλα πλεονεκτήματα του Bootstrap είναι ότι δίνει δυνατότητα να οργανώσουμε σωστά το περιεχόμενο για να προσαρμόζεται αυτόματα ανάλογα με το μέγεθος της οθόνης. Αυτό σημαίνει πως η πλατφόρμα μας φαίνεται και λειτουργεί καλά όχι μόνο σε υπολογιστή αλλά και σε κινητά ή tablets. Τα κουμπιά, τα μενού, οι λίστες, τα πλαίσια, τα χρώματα και τα εικονίδια γίνονται πιο ευχάριστα στο μάτι και πιο λειτουργικά χάρη στα έτοιμα components του Bootstrap όπως οι κάρτες, οι alert ειδοποιήσεις και τα navigation bars που χρησιμοποιήσαμε.

Με το Bootstrap μπορέσαμε να κρατήσουμε έναν σταθερό και επαγγελματικό σχεδιασμό σε όλες τις σελίδες. Οι φοιτητές βλέπουν ξεκάθαρα πού να βάλουν τον κωδικό για να μπουν σε μια εξέταση, πώς να απαντήσουν τις ερωτήσεις και πώς να υποβάλουν τις απαντήσεις τους. Οι καθηγητές έχουν άνετη πλοήγηση στα μαθήματά τους, μπορούν εύκολα να δημιουργήσουν quiz, να προσθέσουν ερωτήσεις και να δουν αποτελέσματα σε καλοσχεδιασμένους πίνακες. Χρησιμοποιήσαμε επίσης Bootstrap Icons για να κάνουμε πιο ευχάριστη την εμπειρία, με μικρά αλλά χρήσιμα σύμβολα δίπλα στα κουμπιά.

Το Bootstrap παρέχει έτοιμες κλάσεις CSS τις οποίες απλώς προσθέτουμε στα HTML στοιχεία μας. Έτσι χωρίς να χρειάζεται να γράφουμε πολλές γραμμές CSS έχουμε εντυπωσιακά αποτελέσματα. Η πλατφόρμα μας απέκτησε ενιαία αισθητική, με επαγγελματική παρουσίαση και λειτουργική δομή κάτι που θα ήταν πολύ πιο δύσκολο χωρίς ένα framework όπως το Bootstrap.

Το CSS και το Bootstrap έπαιξαν πολύ σημαντικό ρόλο στη δημιουργία ενός φιλικού και μοντέρνου περιβάλλοντος. Με την ενσωμάτωση του Bootstrap, το σύστημα έγινε πιο επαγγελματικό, ευχάριστο και εύκολο στη χρήση από όλους.

Κεφάλαιο 4ο: Το σύστημα

4.1 Το σύστημα - εισαγωγή

Το σύστημα αποτελεί μια πλήρως λειτουργική διαδικτυακή πλατφόρμα διαχείρισης εξετάσεων με τη μορφή quiz, προσαρμοσμένη στις ανάγκες εκπαιδευτικών φορέων και ιδρυμάτων. Υλοποιήθηκε με χρήση Python (Flask framework) και MySQL, χωρίς εξαρτήσεις από foreign keys, και συνοδεύεται από responsive διεπαφή χρήστη με Bootstrap και Jinja2 templates.

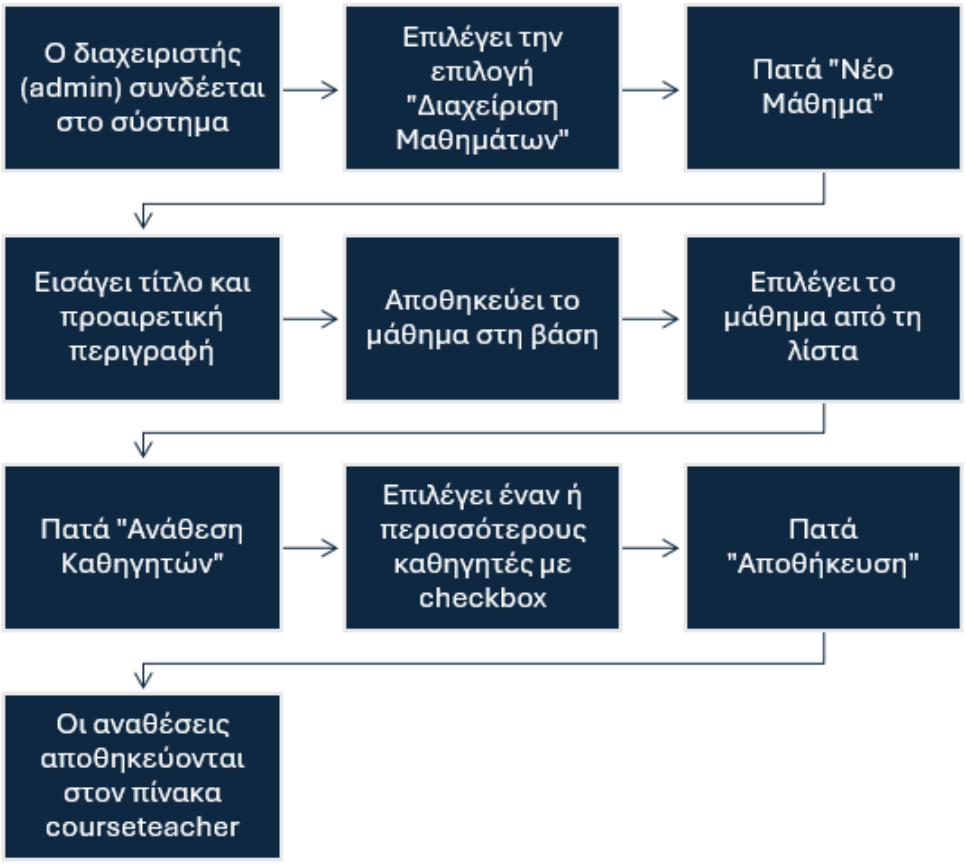
Χαρακτηριστικά:

- Ρόλοι Χρηστών:
    - Διαχειριστής (Admin): διαχειρίζεται τα μαθήματα και αναθέτει καθηγητές.
    - Καθηγητής (Teacher): δημιουργεί quizzes, θέτει ερωτήσεις και παρακολουθεί αποτελέσματα.
    - Φοιτητής (Student): συμμετέχει σε εξετάσεις εισάγοντας τον 5ψήφιο μοναδικό κωδικό κάθε quiz.
- Διαχείριση Μαθημάτων:
    - Ο admin προσθέτει μαθήματα και ορίζει ποιοι καθηγητές μπορούν να διαχειριστούν το καθένα.
- Δημιουργία και Οργάνωση Quiz:
    - Κάθε καθηγητής μπορεί να δημιουργεί quizzes σε κάθε μάθημα που του έχει ανατεθεί.
    - Ορίζει τίτλο, χρονικά όρια (έναρξη και λήξη), και παράγει αυτόματα μοναδικό 5ψήφιο UID.
- Υποσύστημα Ερωτήσεων:
    - Κάθε quiz περιλαμβάνει πολλαπλές ερωτήσεις τύπου πολλαπλής επιλογής.
    - Οι ερωτήσεις και οι απαντήσεις μπορούν να συνοδεύονται από εικόνες.
- Συμμετοχή Φοιτητών:
    - Οι φοιτητές εισέρχονται στο quiz με τον UID κωδικό και ολοκληρώνουν τις ερωτήσεις.
    - Η υποβολή είναι τελική και μη επεξεργάσιμη.
- Παρακολούθηση Αποτελεσμάτων:

- Οι καθηγητές έχουν πρόσβαση στα στατιστικά κάθε quiz με συνολική απεικόνιση σωστών/λανθασμένων απαντήσεων.
- Υπάρχει δυνατότητα εξαγωγής των αποτελεσμάτων σε αρχείο .csv.

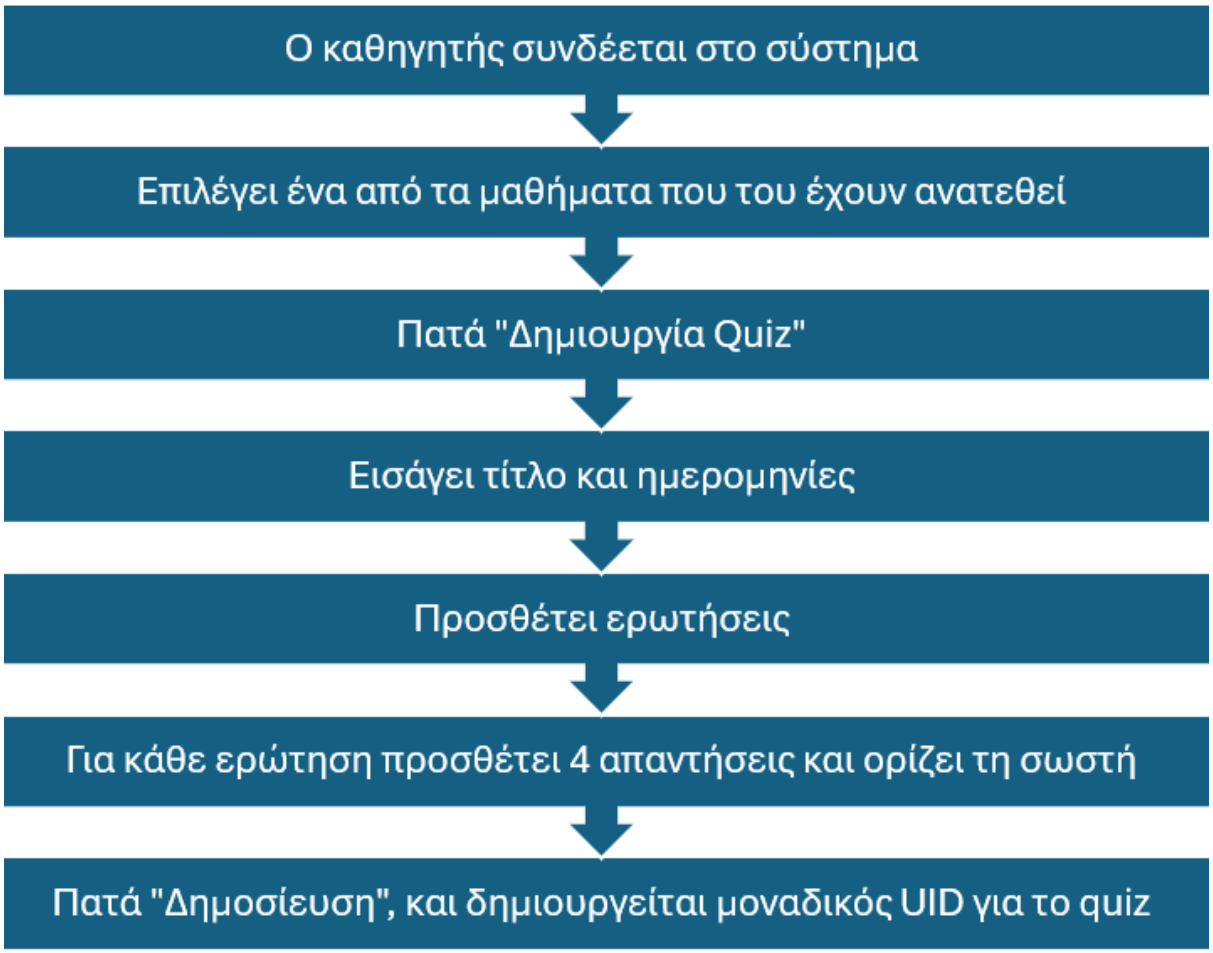
- Responsive UI:
  - Όλες οι σελίδες είναι πλήρως responsive και φιλικές προς κινητά/tablet.
  - Περιλαμβάνονται μενού πλοήγησης, εικονίδια και σύγχρονη αισθητική με χρήση Bootstrap και Bootstrap Icons.

Το σύστημα είναι πλήρως επεκτάσιμο και μπορεί να προσαρμοστεί σε πλήθος σεναρίων αξιολόγησης (π.χ. διαγωνίσματα, αυτοαξιολογήσεις, επαναληπτικά τεστ).



Εικόνα 4.1: Διαχείριση Μαθημάτων από Admin

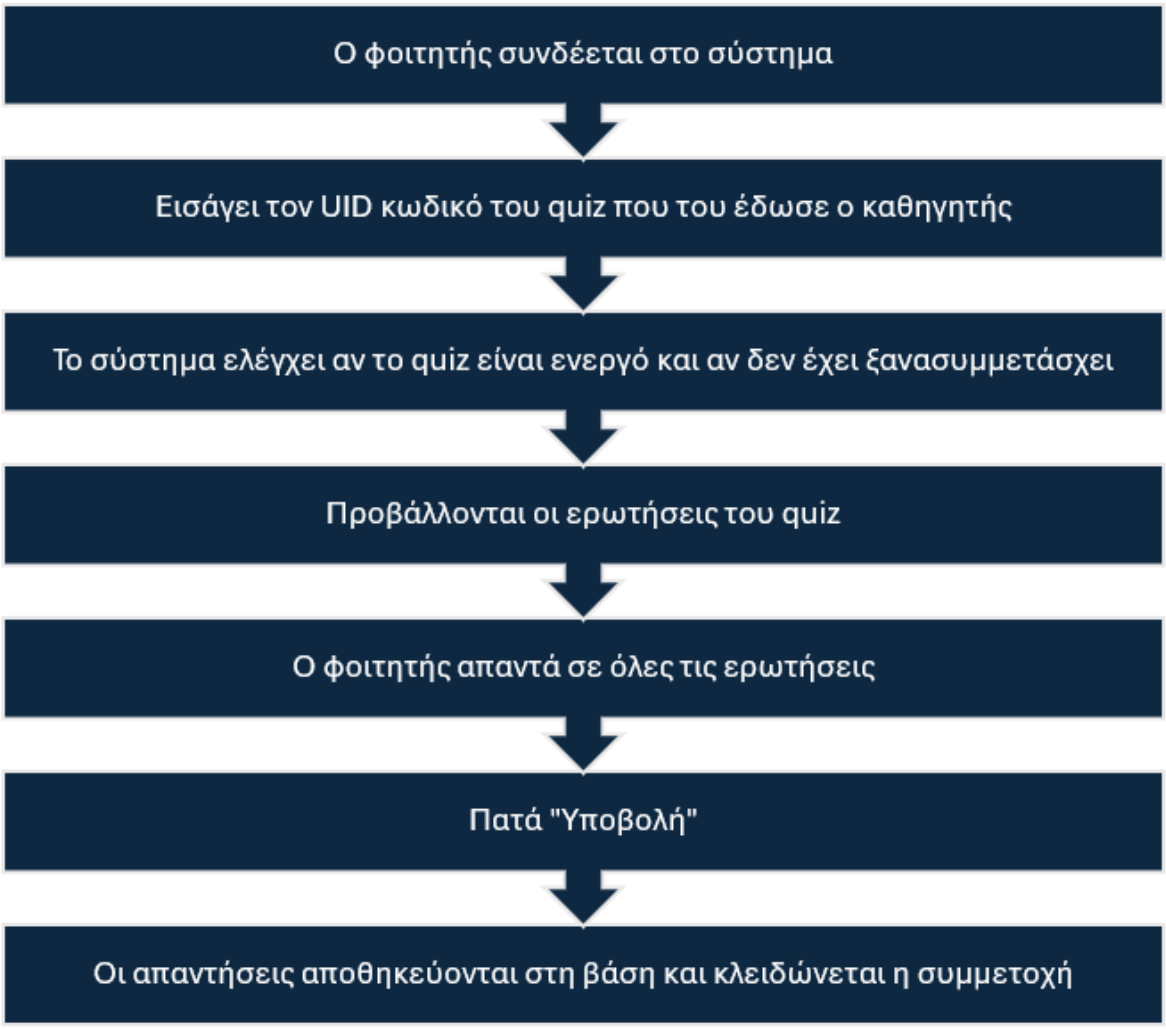
Στην Εικόνα 4.1 παρουσιάζεται η ροή ενεργειών του ρόλου *Διαχειριστής (Admin)* σχετικά με τη διαχείριση των μαθημάτων. Αρχικά, ο διαχειριστής συνδέεται στο σύστημα και έχει πρόσβαση σε ειδικό πάνελ από το οποίο μπορεί να δει τη λίστα των ήδη καταχωρημένων μαθημάτων. Μέσω της επιλογής "Νέο Μάθημα", έχει τη δυνατότητα να δημιουργήσει νέο μάθημα εισάγοντας τον τίτλο και την περιγραφή του. Μπορεί να επιλέξει ένα μάθημα από τη λίστα και να προχωρήσει στην ανάθεση καθηγητών σε αυτό, μέσω μιας φόρμας με checkbox όπου επιλέγει ποιοι χρήστες (με ρόλο καθηγητή) θα έχουν πρόσβαση στο συγκεκριμένο μάθημα. Όλες οι αλλαγές καταχωρούνται αυτόματα στη βάση δεδομένων. Το συγκεκριμένο διάγραμμα οπτικοποιεί τον τρόπο με τον οποίο ο admin έχει τον πλήρη έλεγχο του εκπαιδευτικού περιεχομένου και της ανάθεσης ρόλων στο σύστημα.



Εικόνα 4.2: Διαχείριση Quiz από Καθηγητή

Στην Εικόνα 4.2 παρουσιάζεται η λειτουργική ροή που ακολουθεί ένας *Καθηγητής* κατά τη δημιουργία και διαχείριση quiz για τα μαθήματά του. Ο καθηγητής, αφού συνδεθεί στο σύστημα μεταβαίνει στην ενότητα των μαθημάτων του και επιλέγει κάποιο από αυτά. Στη συνέχεια έχει τη δυνατότητα να

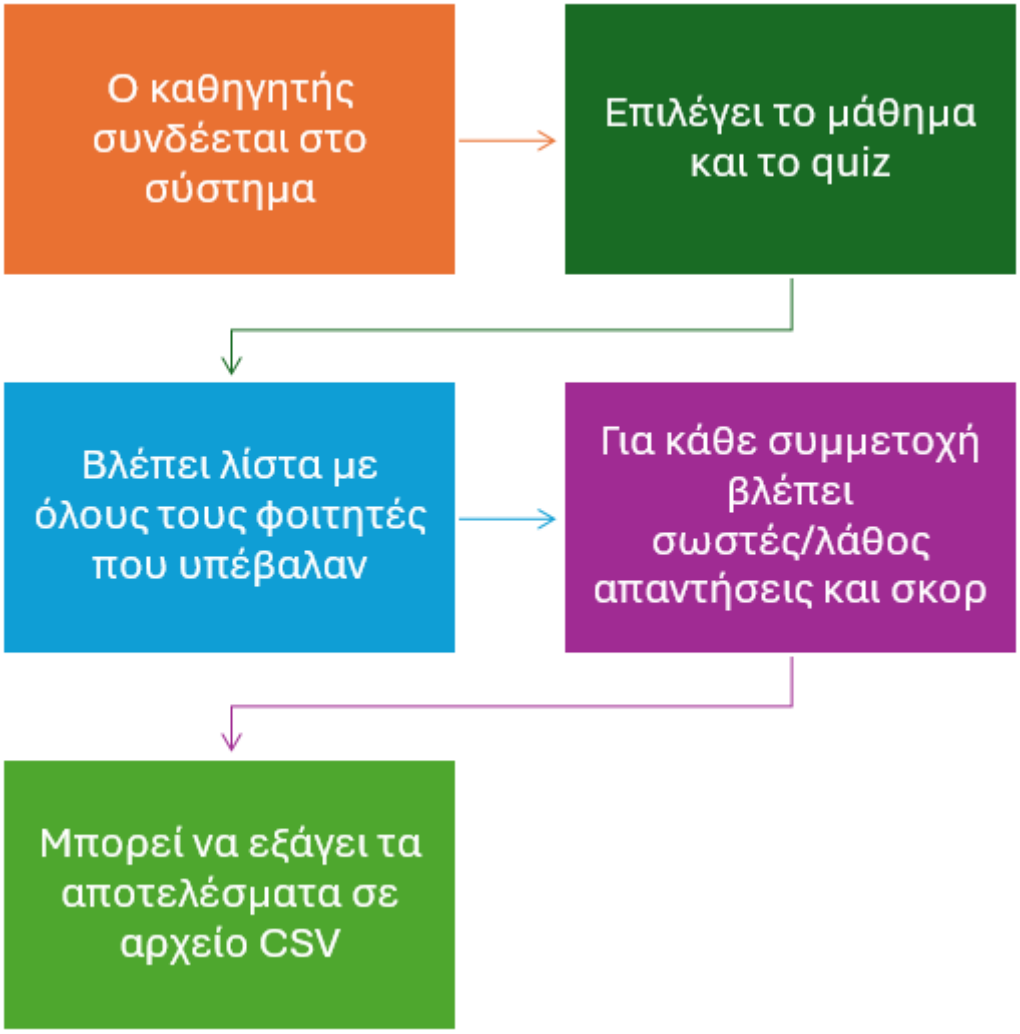
δημιουργήσει νέο quiz για το επιλεγμένο μάθημα, εισάγοντας τον τίτλο και τις ημερομηνίες ενεργοποίησης και λήξης του. Έπειτα περνά στη φάση προσθήκης ερωτήσεων όπου για κάθε ερώτηση μπορεί να καθορίσει εκφώνηση (κειμενική ή με εικόνα) και τέσσερις πιθανές απαντήσεις επιλέγοντας μία ως σωστή. Αφού ολοκληρωθεί η σύνθεση το quiz δημοσιεύεται και αποκτά έναν μοναδικό πενταψήφιο κωδικό (UID), τον οποίο μπορεί να δώσει στους φοιτητές του για να συμμετάσχουν στην εξέταση. Το διάγραμμα αναδεικνύει την απλή αλλά ισχυρή διαδικασία δημιουργίας δομημένων quiz από τον εκπαιδευτικό.



Εικόνα 4.3: Συμμετοχή Φοιτητή

Η **Εικόνα 4.3** αποτυπώνει την ροή διαδικασίας συμμετοχής ενός *Φοιτητή* σε ένα quiz μέσω της πλατφόρμας. Ο φοιτητής ξεκινά με τη σύνδεσή του και στην αρχική σελίδα καλείται να εισάγει τον πενταψήφιο κωδικό (UID) που του έχει δοθεί από τον καθηγητή. Το σύστημα ελέγχει την εγκυρότητα του quiz, το αν είναι εντός χρονικού πλαισίου και εάν έχει ήδη συμμετάσχει. Αν όλα είναι έγκυρα, εμφανίζονται οι ερωτήσεις του quiz, καθεμία με τις τέσσερις επιλογές απάντησης. Ο φοιτητής απαντά και στο τέλος πατά "Υποβολή". Με την υποβολή, οι απαντήσεις αποθηκεύονται στη βάση δεδομένων

και η συμμετοχή του κλειδώνεται ώστε να μην μπορεί να ξαναλάβει μέρος στο ίδιο quiz. Το διάγραμμα καταγράφει με ακρίβεια τον τρόπο με τον οποίο διασφαλίζεται η ακεραιότητα της συμμετοχής.



Εικόνα 4.4: Προβολή Αποτελεσμάτων από Καθηγητή

Στην **Εικόνα 4.4** φαίνεται η διαδικασία με την οποία ένας *Καθηγητής* μπορεί να ελέγξει και να αξιολογήσει τις υποβολές των φοιτητών του σε ένα quiz. Ο καθηγητής αρχικά επιλέγει το σχετικό μάθημα και στη συνέχεια το συγκεκριμένο quiz. Το σύστημα εμφανίζει λίστα με όλους τους φοιτητές που υπέβαλαν τις απαντήσεις τους, μαζί με βασικά στοιχεία όπως το όνομα, τον αριθμό μητρώου, τον χρόνο υποβολής και το ποσοστό επιτυχίας. Ο καθηγητής μπορεί να δει λεπτομέρειες για κάθε συμμετοχή, συμπεριλαμβανομένων των απαντήσεων και της ορθότητας κάθε επιλογής. Ακόμη του δίνεται η δυνατότητα εξαγωγής όλων των αποτελεσμάτων σε αρχείο CSV για περαιτέρω επεξεργασία ή αρχειοθέτηση. Το διάγραμμα αυτό αναδεικνύει την πληρότητα της εποπτείας και αξιολόγησης από πλευράς εκπαιδευτικού.

4.2 Η βάση δεδομένων μας MySQL

Για την υλοποίηση επιλέχθηκε η MySQL, λόγω της ευρείας αποδοχής της, της σταθερότητάς της και της συμβατότητάς της με εφαρμογές Flask.

Η βάση δεδομένων είναι δομημένη γύρω από τους βασικούς ρόλους (Admin, Teacher, Student) και τις λειτουργικές ενότητες της εφαρμογής (Μαθήματα, Quizzes, Ερωτήσεις, Απαντήσεις, Υποβολές). Για κάθε ξεχωριστό αντικείμενο της εφαρμογής (entity) δημιουργήθηκε ξεχωριστός πίνακας. Παρόλο που δεν χρησιμοποιούνται ξένες κλείδες (foreign keys), η σύνδεση μεταξύ πινάκων επιτυγχάνεται μέσω id πεδίων και της λογικής συνέπειας στην εφαρμογή.

Περιγραφή Πινάκων

Ο πίνακας users περιέχει όλους τους χρήστες του συστήματος: διαχειριστές, καθηγητές και φοιτητές. Διαφοροποίηση των ρόλων γίνεται με το πεδίο role.

Πεδία:

- id (INT, PK)
- name (VARCHAR)
- email (VARCHAR)
- password (VARCHAR) – αποθηκευμένο με hash
- role (ENUM: 'admin', 'teacher', 'student')
- student_number (VARCHAR, optional για φοιτητές)

Ο πίνακας αυτός περιέχει όλα τα διαθέσιμα μαθήματα.

Πεδία:

- id (INT, PK)
- title (VARCHAR)
- description (TEXT)

Αυτός ο πίνακας συνδέει καθηγητές με μαθήματα. Ένας καθηγητής μπορεί να διδάσκει πολλά μαθήματα, και ένα μάθημα μπορεί να έχει πολλούς καθηγητές.

Πεδία:

- id (INT, PK)
- user_id (INT, καθηγητής)
- course_id (INT, μάθημα)

Ο πίνακας quizzes αποθηκεύει τις εξετάσεις τύπου quiz που έχουν δημιουργηθεί για κάθε μάθημα.

Πεδία:

- id (INT, PK)
- course_id (INT)
- title (VARCHAR)
- uid (VARCHAR(5), μοναδικός κωδικός)
- deadline_start (DATETIME)
- deadline_end (DATETIME)

questions

Κάθε εγγραφή αντιπροσωπεύει μια ερώτηση που ανήκει σε συγκεκριμένο quiz.

Πεδία:

- id (INT, PK)
- quiz_id (INT)
- question_text (TEXT)
- image_path (VARCHAR, optional)

Ο πίνακας answers περιέχει τις απαντήσεις για κάθε ερώτηση, μαζί με την ένδειξη για το ποια είναι η σωστή.

Πεδία:

- id (INT, PK)
- question_id (INT)
- answer_text (TEXT)
- image_path (VARCHAR, optional)

- is_correct (BOOLEAN)

student_submissions

Καταγράφει κάθε συμμετοχή φοιτητή σε quiz. Μια υποβολή είναι μοναδική για κάθε φοιτητή ανά quiz.

Πεδία:

- id (INT, PK)
- quiz_id (INT)
- user_id (INT)
- submission_time (DATETIME)
- is_submitted (BOOLEAN)

student_answers

Περιέχει τις απαντήσεις που έδωσε κάθε φοιτητής σε συγκεκριμένη υποβολή.

Πεδία:

- id (INT, PK)
- submission_id (INT)
- question_id (INT)
- answer_id (INT)

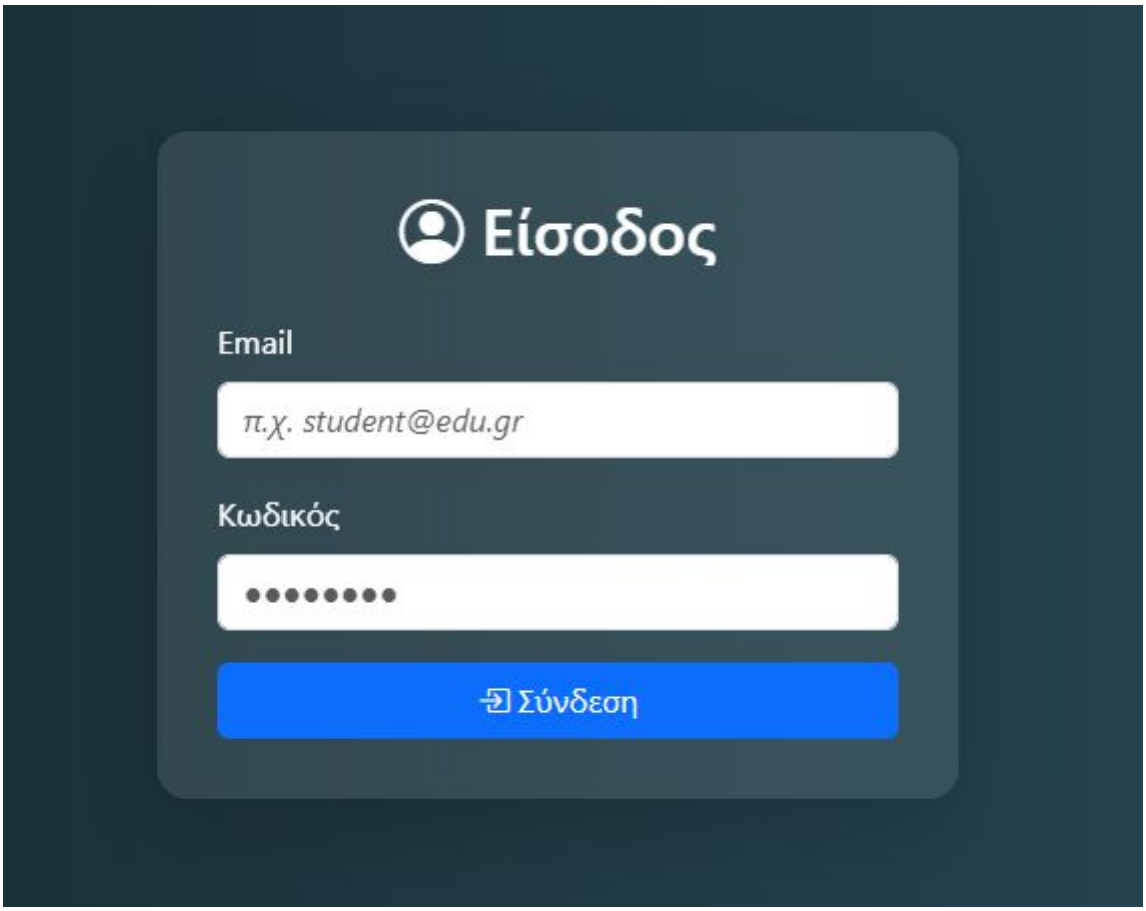
Παρότι οι πίνακες δεν συνδέονται με foreign keys, οι σχέσεις τους είναι απολύτως καθορισμένες:

- Κάθε quiz ανήκει σε ένα course
- Κάθε question ανήκει σε ένα quiz
- Κάθε answer ανήκει σε μία question
- Κάθε student_submission συνδέει έναν user με ένα quiz
- Κάθε student_answer συνδέει μια απάντηση με μια υποβολή και ερώτηση

Πλεονεκτήματα Σχεδίασης

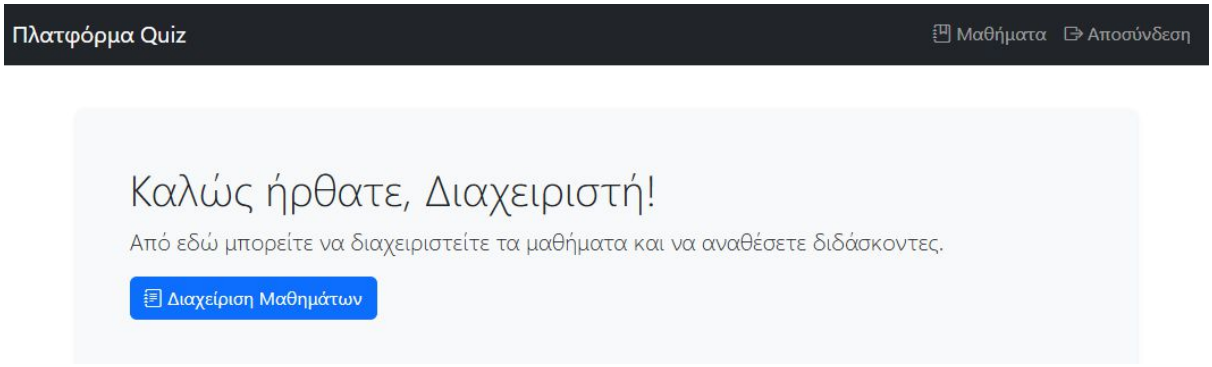
- Επεκτασιμότητα: Μπορούν εύκολα να προστεθούν νέοι τύποι χρηστών ή λειτουργίες.
- Ευελιξία: Η απουσία foreign keys επιτρέπει ελευθερία σε μελλοντικές επεμβάσεις ή μετεγκαταστάσεις της βάσης.
- Καθαρή Δομή: Η οργάνωση της πληροφορίας είναι ευανάγνωστη και ξεκάθαρη.

4.3 Η ιστοσελίδα μας



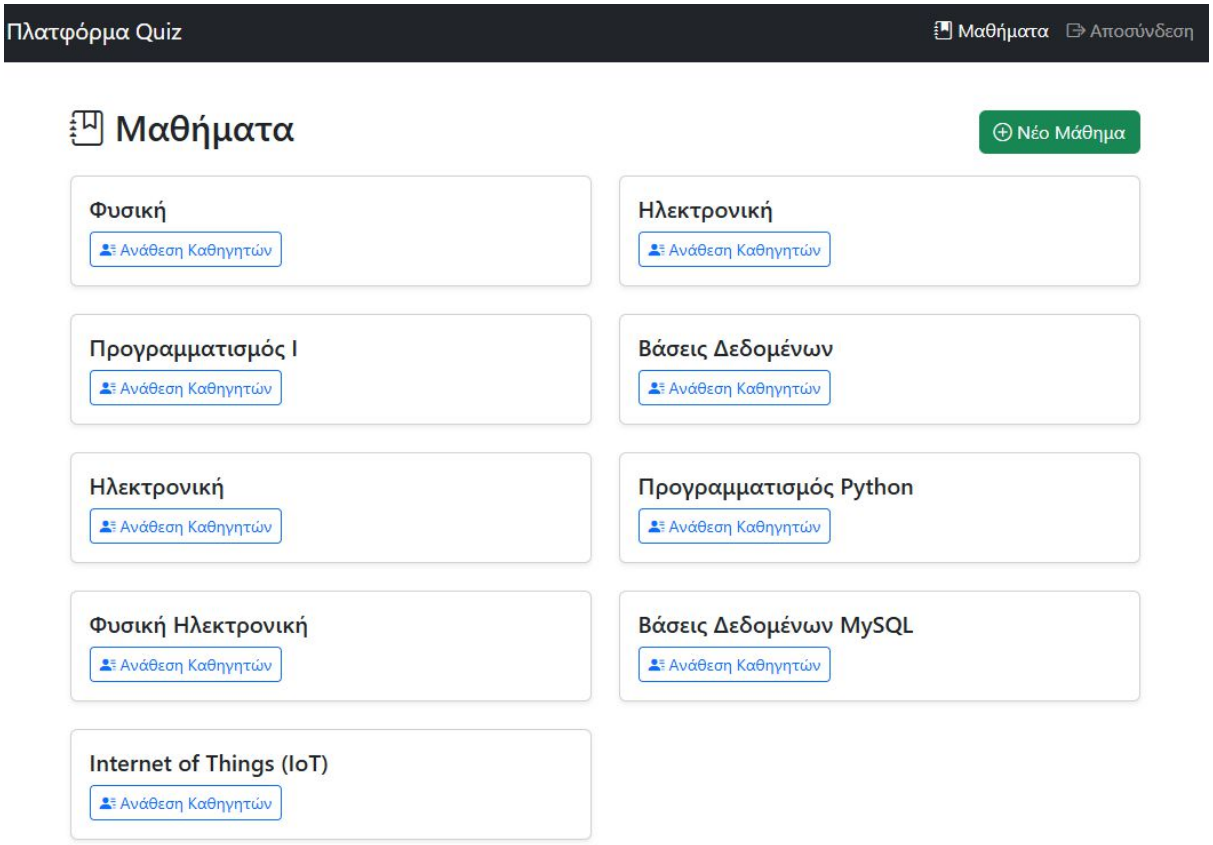
Εικόνα 4.5: Σύνδεση του χρήστη

Η εικόνα 4.5 παρουσιάζει τη βασική φόρμα σύνδεσης χρηστών στο σύστημα. Ο κάθε χρήστης εισάγει το email και τον κωδικό του για να αποκτήσει πρόσβαση. Ανάλογα με τον ρόλο του (admin, καθηγητής ή φοιτητής), ανακατευθύνεται στο αντίστοιχο dashboard. Το περιβάλλον είναι καθαρό και φιλικό προς τον χρήστη ενώ σε περίπτωση λανθασμένων στοιχείων εμφανίζεται κατάλληλο μήνυμα σφάλματος.



Εικόνα 4.6: Εισαγωγή στο διαχειριστικό για τον Admin

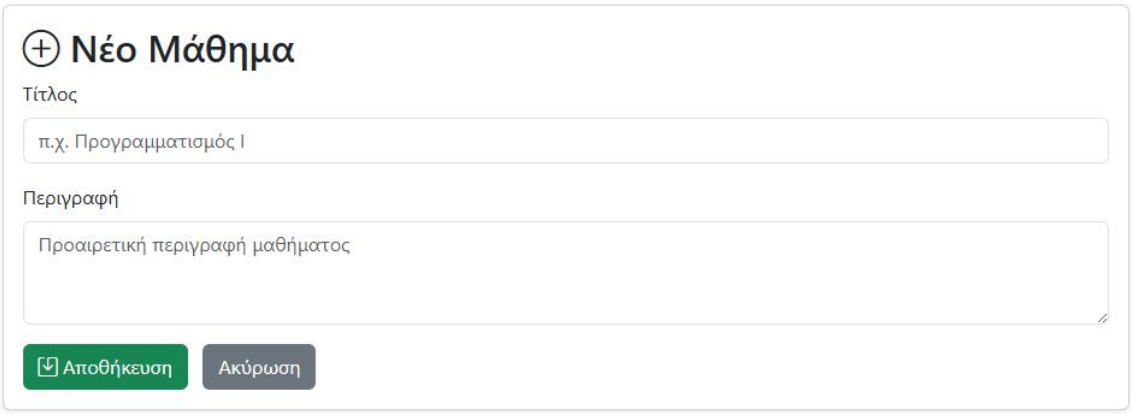
Στην εικόνα 4.6 αποτυπώνεται η αρχική οθόνη που βλέπει ο διαχειριστής μετά τη σύνδεση. Περιλαμβάνει βασικές ενέργειες όπως η δημιουργία μαθημάτων και η διαχείριση των καθηγητών. Παρέχεται γρήγορη πλοήγηση στις λειτουργίες του συστήματος με ξεκάθαρη διάταξη και χρήση εικονιδίων για διευκόλυνση.



Εικόνα 4.7: Τα μαθήματα που βλέπει ο admin

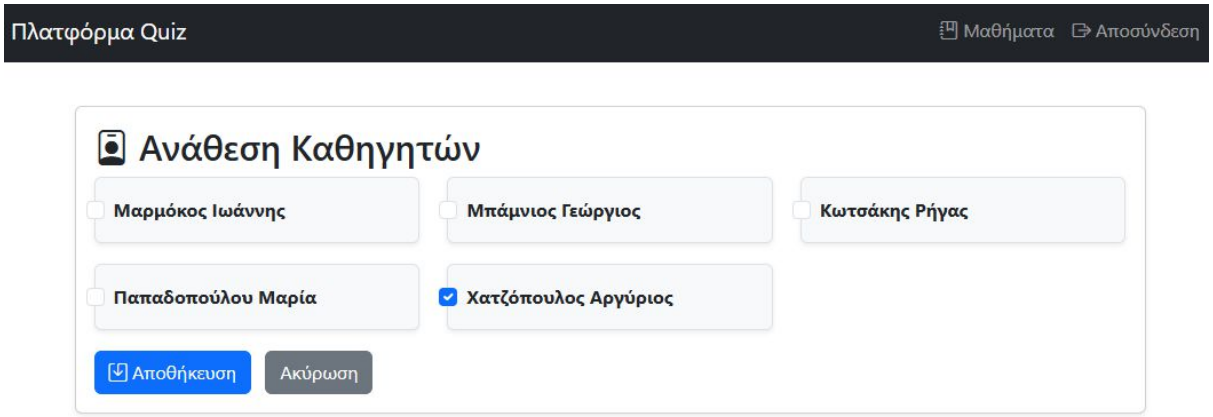
Ο διαχειριστής έχει τη δυνατότητα να βλέπει τη λίστα με όλα τα διαθέσιμα μαθήματα του συστήματος (Εικόνα 4.7). Κάθε μάθημα εμφανίζεται με τίτλο και περιγραφή, ενώ παρέχονται επιλογές για ανάθεση καθηγητών ή τροποποίηση στοιχείων. Η διάταξη επιτρέπει εύκολη διαχείριση ακόμη και σε περιβάλλον με πολλά μαθήματα.

Η εικόνα 4.8 τη φόρμα εισαγωγής νέου μαθήματος. Ο admin εισάγει τον τίτλο και την περιγραφ ενώ το περιβάλλον είναι απλό και χρηστικό. Η καταχώρηση πραγματοποιείται με το πάτημα ενός κουμπιού και το μάθημα γίνεται άμεσα διαθέσιμο για ανάθεση σε καθηγητές.



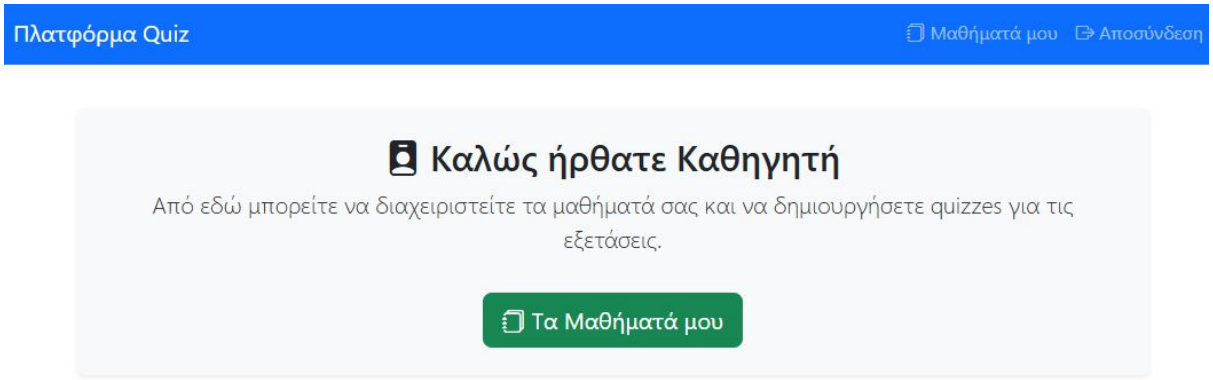
Εικόνα 4.8: Δημιουργία νέου μαθήματος από τον Admin

Σε αυτό το στάδιο (Εικόνα 4.9), ο admin επιλέγει ποιοι καθηγητές θα έχουν πρόσβαση στο μάθημα μέσω checkbox. Η εικόνα δείχνει τη φόρμα ανάθεσης όπου οι διαθέσιμοι καθηγητές εμφανίζονται με ονόματα και οι ήδη ανατεθειμένοι είναι προεπιλεγμένοι. Η αποθήκευση των επιλογών ολοκληρώνει τη διαδικασία.



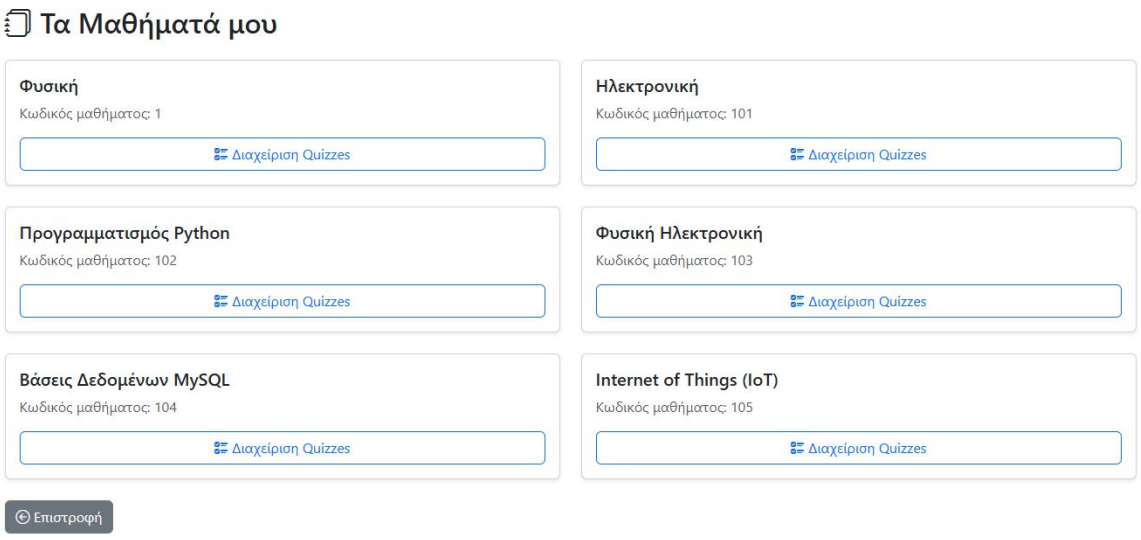
Εικόνα 4.9: Επιλογή ποιοι από τους καθηγητές μπορούν να έχουν πρόσβαση στη διαχείριση του μαθήματος

Ο καθηγητής μόλις συνδεθεί οδηγείται σε αρχική σελίδα καλωσορίσματος. Από εδώ (Εικόνα 4.10) έχει πρόσβαση στα μαθήματά του και στις λειτουργίες δημιουργίας quiz. Το περιβάλλον είναι προσαρμοσμένο στις ανάγκες του εκπαιδευτικού, με εικονίδια και κουμπιά για άμεση πλοήγηση.



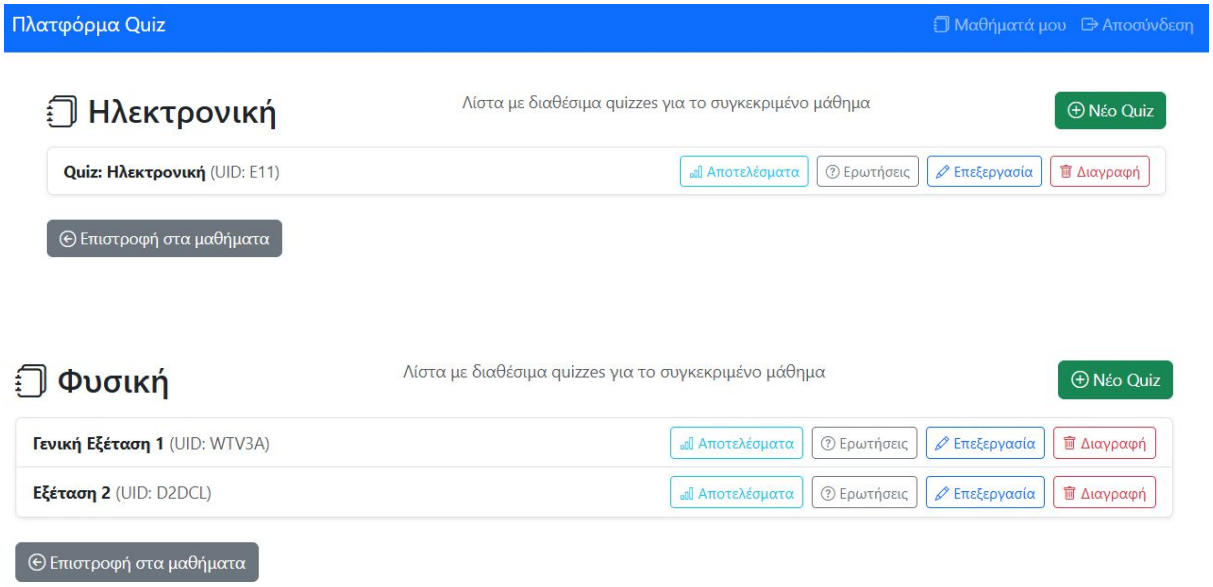
Εικόνα 4.10: Προβολή Αρχικής Οθόνης για Teacher

Η εικόνα 4.11 παρουσιάζει την προσωπική λίστα μαθημάτων του καθηγητή. Κάθε μάθημα εμφανίζεται σε κάρτα με τίτλο και μοναδικό κωδικό, ενώ υπάρχει κουμπί για μετάβαση στη διαχείριση των quiz του μαθήματος. Ο σχεδιασμός επιτρέπει καθαρή προβολή.



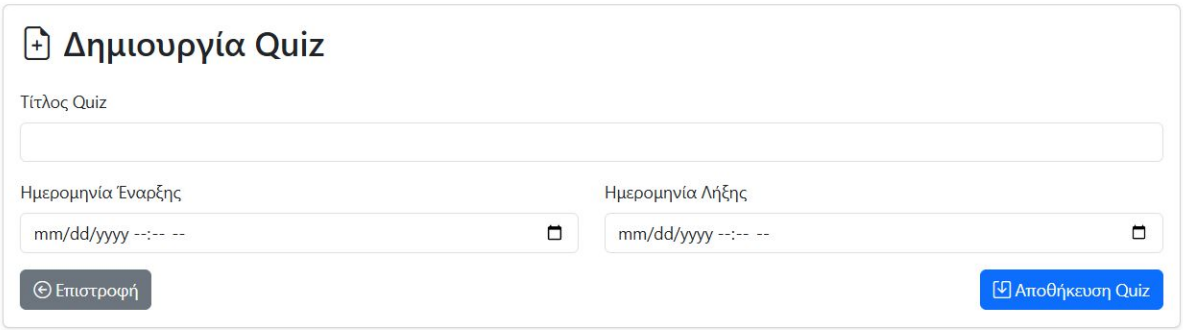
Εικόνα 4.11: Τα μαθήματα μου ως Teacher

Εδώ εμφανίζονται όλα τα quizzes που έχουν δημιουργηθεί για συγκεκριμένο μάθημα (εικόνα 4.12). Κάθε quiz συνοδεύεται από τίτλο, μοναδικό UID και κουμπιά ενεργειών: αποτελέσματα, ερωτήσεις, επεξεργασία και διαγραφή. Ο καθηγητής έχει πλήρη έλεγχο των διαθέσιμων quiz με ένα κλικ.



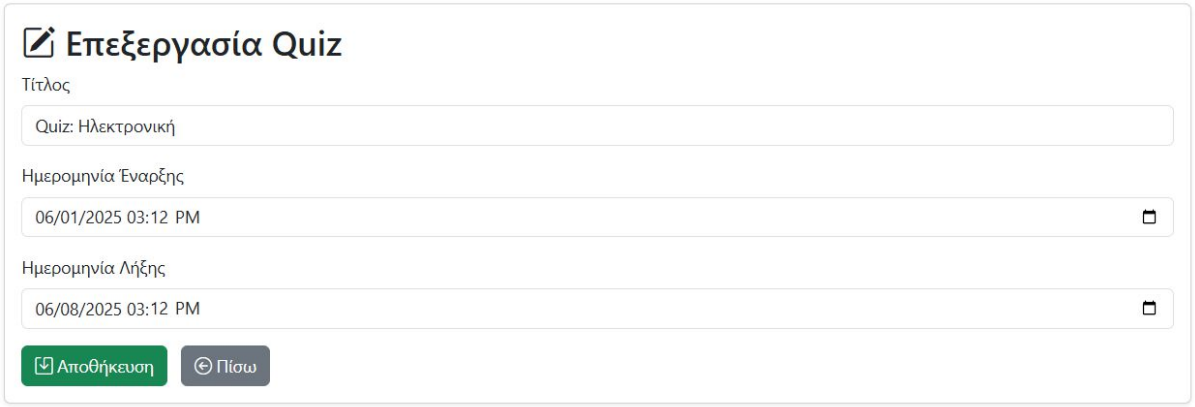
Εικόνα 4.12: Λίστα με τα Quizzes που έχει το μάθημα Φυσική και Ηλεκτρονική - Teacher

Η φόρμα δημιουργίας quiz επιτρέπει στον καθηγητή να καθορίσει τον τίτλο του quiz και τις ημερομηνίες έναρξης και λήξης (Εικόνα 4.13). Μετά την αποθήκευση, δημιουργείται αυτόματα μοναδικός UID, ο οποίος θα δοθεί στους φοιτητές. Η διαδικασία είναι απλή και κατανοητή.



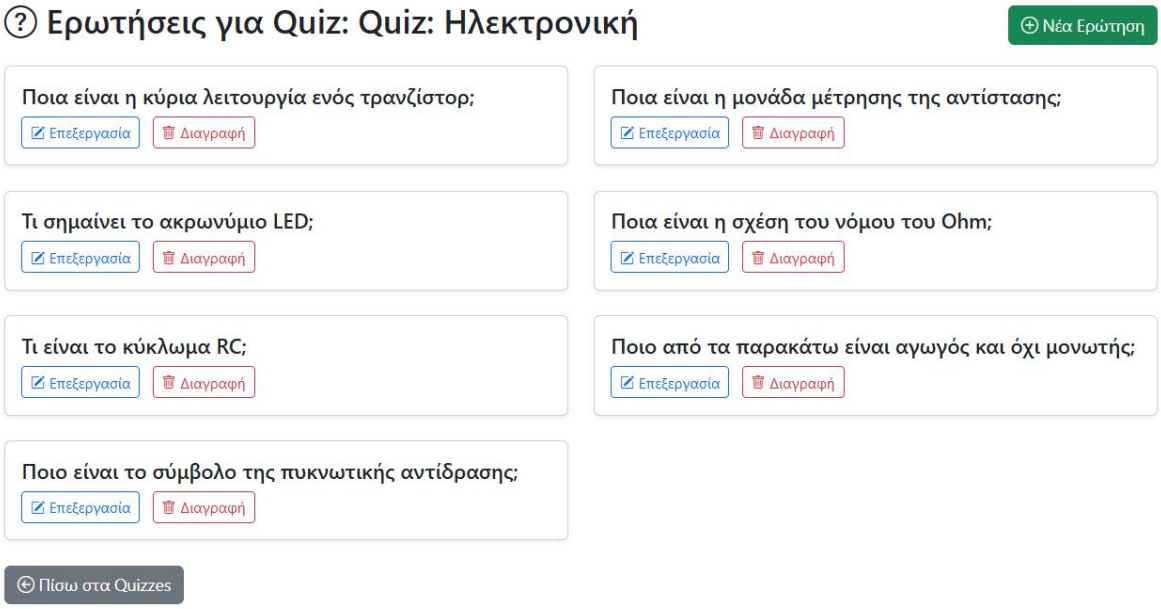
Εικόνα 4.13: Σελίδα για δημιουργία νέου Quiz - Teacher

Ο καθηγητής μπορεί να επεξεργαστεί τις πληροφορίες του quiz ακόμα και μετά τη δημιουργία του (Εικόνα 4.14). Η εικόνα δείχνει τη φόρμα όπου μπορεί να αλλάξει τον τίτλο και τις ημερομηνίες. Δεν επηρεάζονται οι ήδη καταχωρημένες ερωτήσεις, διατηρώντας την ακεραιότητα της εξέτασης.



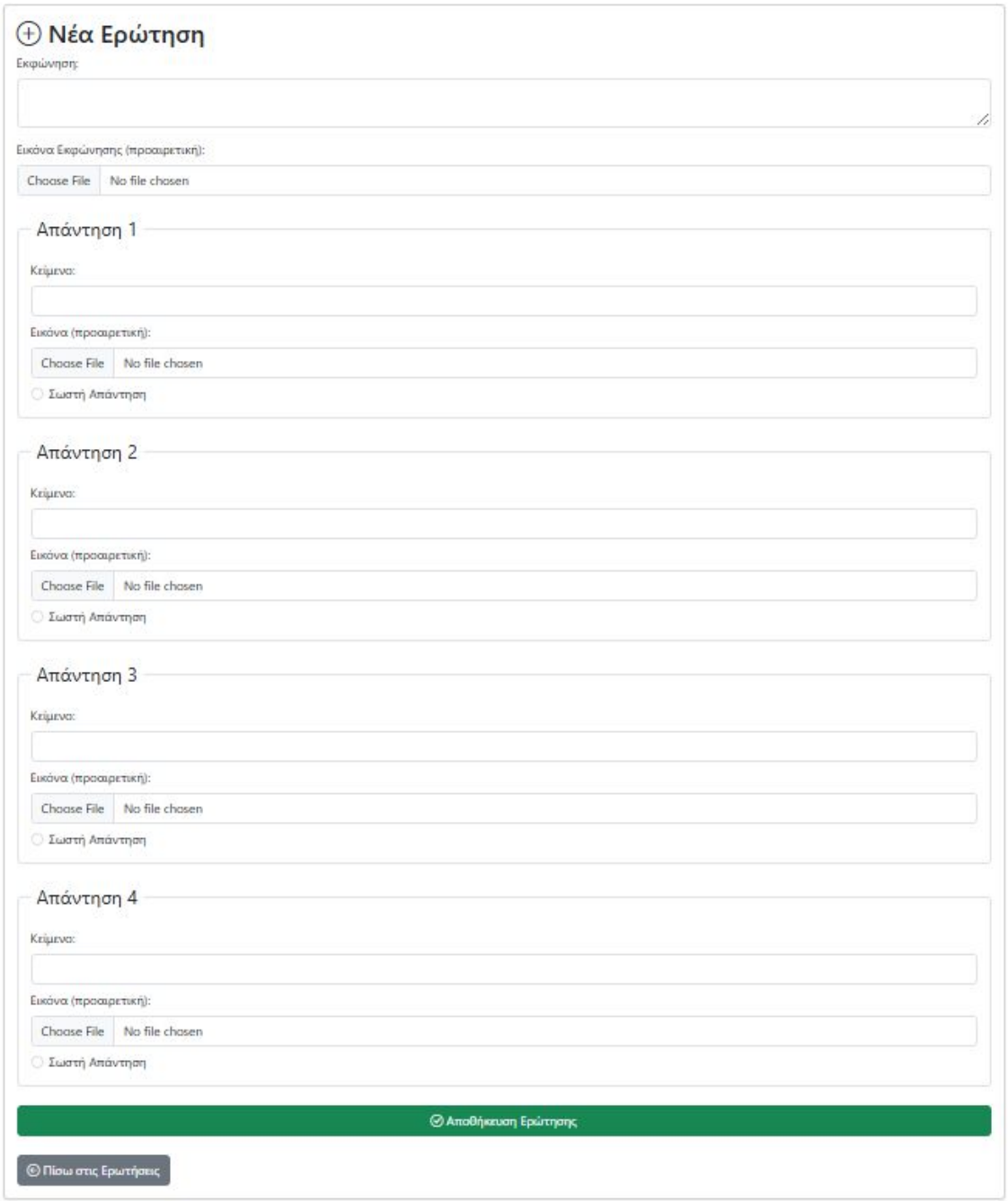
Εικόνα 4.14: Σελίδα για επεξεργασία Quiz - Teacher

Η εικόνα 4.15 παρουσιάζει τη σελίδα διαχείρισης ερωτήσεων για ένα quiz. Εμφανίζεται η λίστα των ερωτήσεων, με δυνατότητα επεξεργασίας ή διαγραφής. Οι εκφωνήσεις και οι εικόνες προβάλλονται και η δομή είναι φιλική και σαφής για τον καθηγητή.



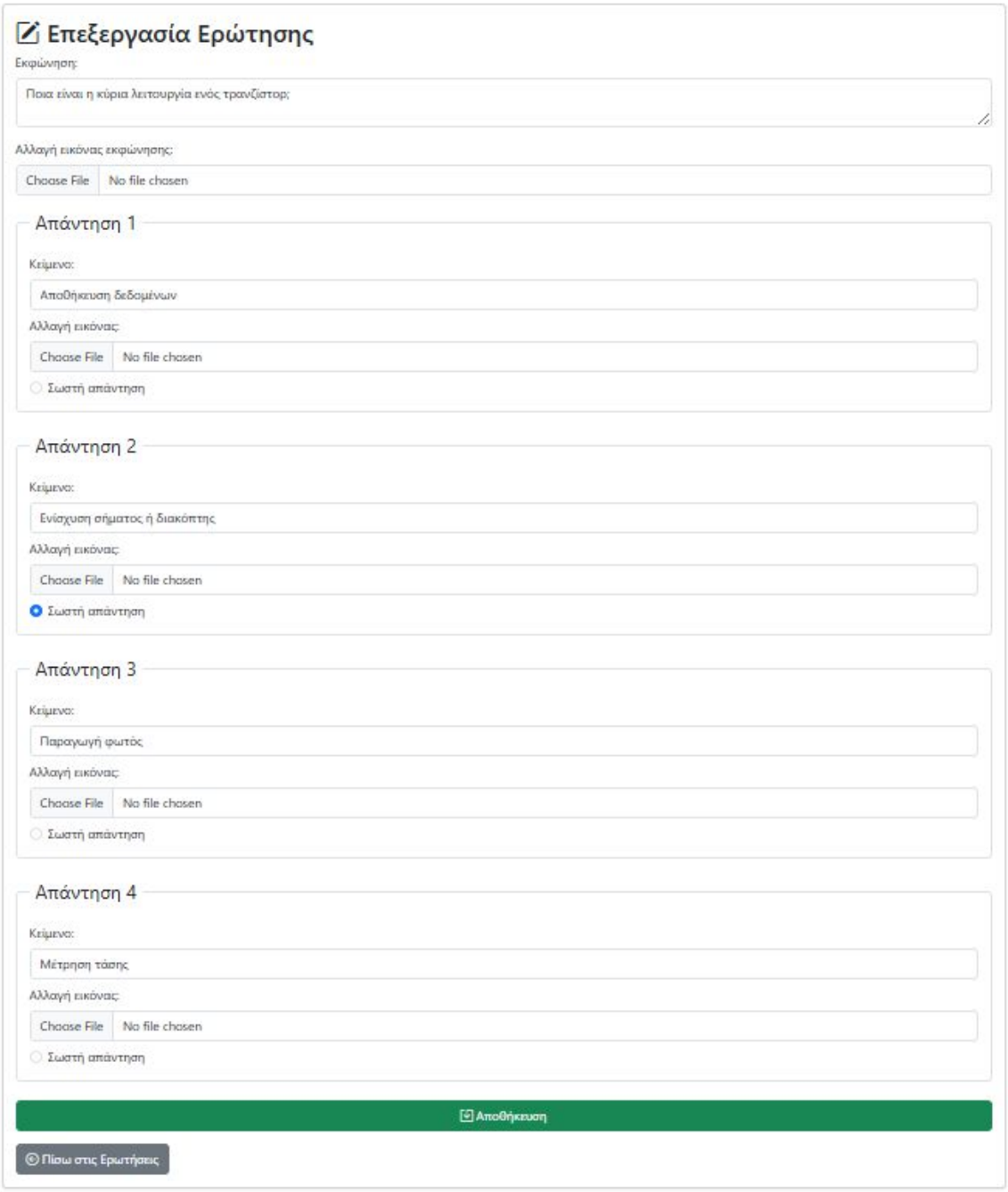
Εικόνα 4.15: Ερωτήσεις για Quiz: Ηλεκτρονική - Teacher

Ο καθηγητής μπορεί να προσθέσει νέα ερώτηση στο quiz, εισάγοντας εκφώνηση, εικόνα και τέσσερις πιθανές απαντήσεις (Εικόνα 4.16). Η μία επιλέγεται ως σωστή. Η διαδικασία επιτρέπει γρήγορη προσθήκη περιεχομένου χωρίς περιττή πολυπλοκότητα.



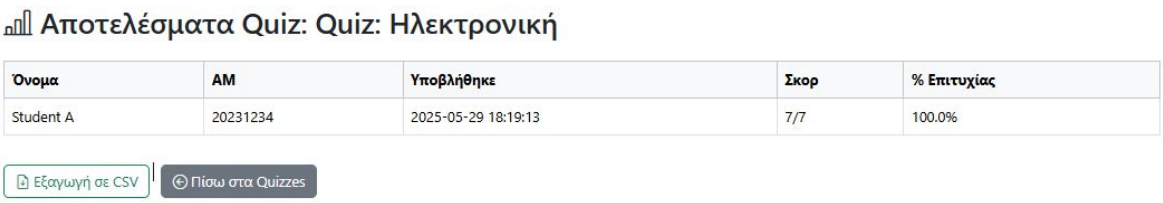
Εικόνα 4.16: Νέα Ερώτηση - Teacher

Η εικόνα 4.17 παρουσιάζει τη φόρμα επεξεργασίας μιας ήδη υπάρχουσας ερώτησης μαζί με τις τέσσερις απαντήσεις της. Ο καθηγητής μπορεί να αλλάξει το κείμενο ή τις εικόνες των απαντήσεων και να ορίσει εκ νέου ποια είναι σωστή.



Εικόνα 4.17: Διαχείριση υποβολών των φοιτητών - Teacher

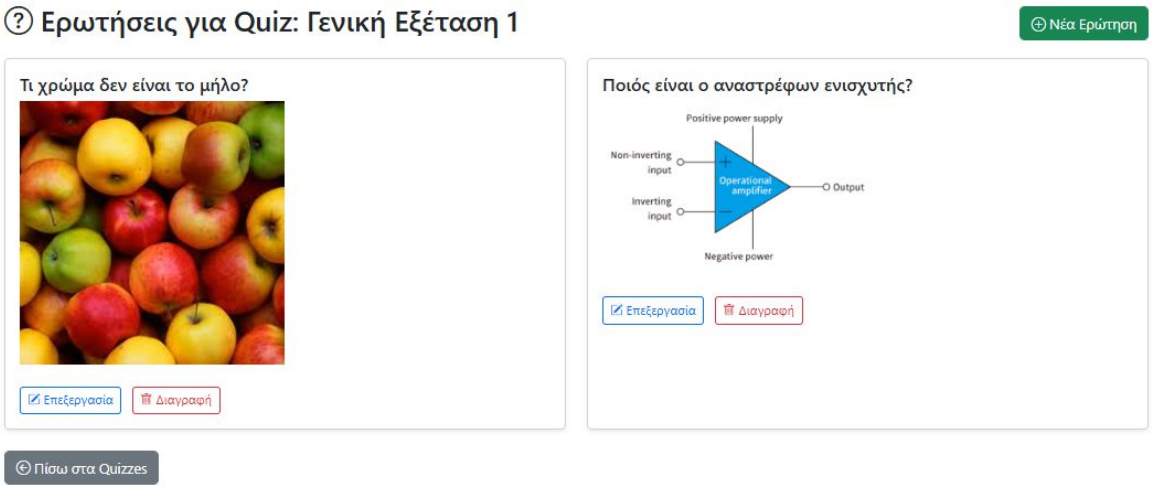
Εδώ εμφανίζονται οι υποβολές των φοιτητών για συγκεκριμένο quiz (Εικόνα 4.18). Κάθε γραμμή περιλαμβάνει το όνομα, τον αριθμό μητρώου, την ώρα υποβολής, το σκορ και το ποσοστό επιτυχίας. Ο καθηγητής αποκτά γρήγορη εικόνα της απόδοσης των συμμετεχόντων.



Εικόνα 4.18: Αποτελέσματα Quiz: Ηλεκτρονική - Teacher

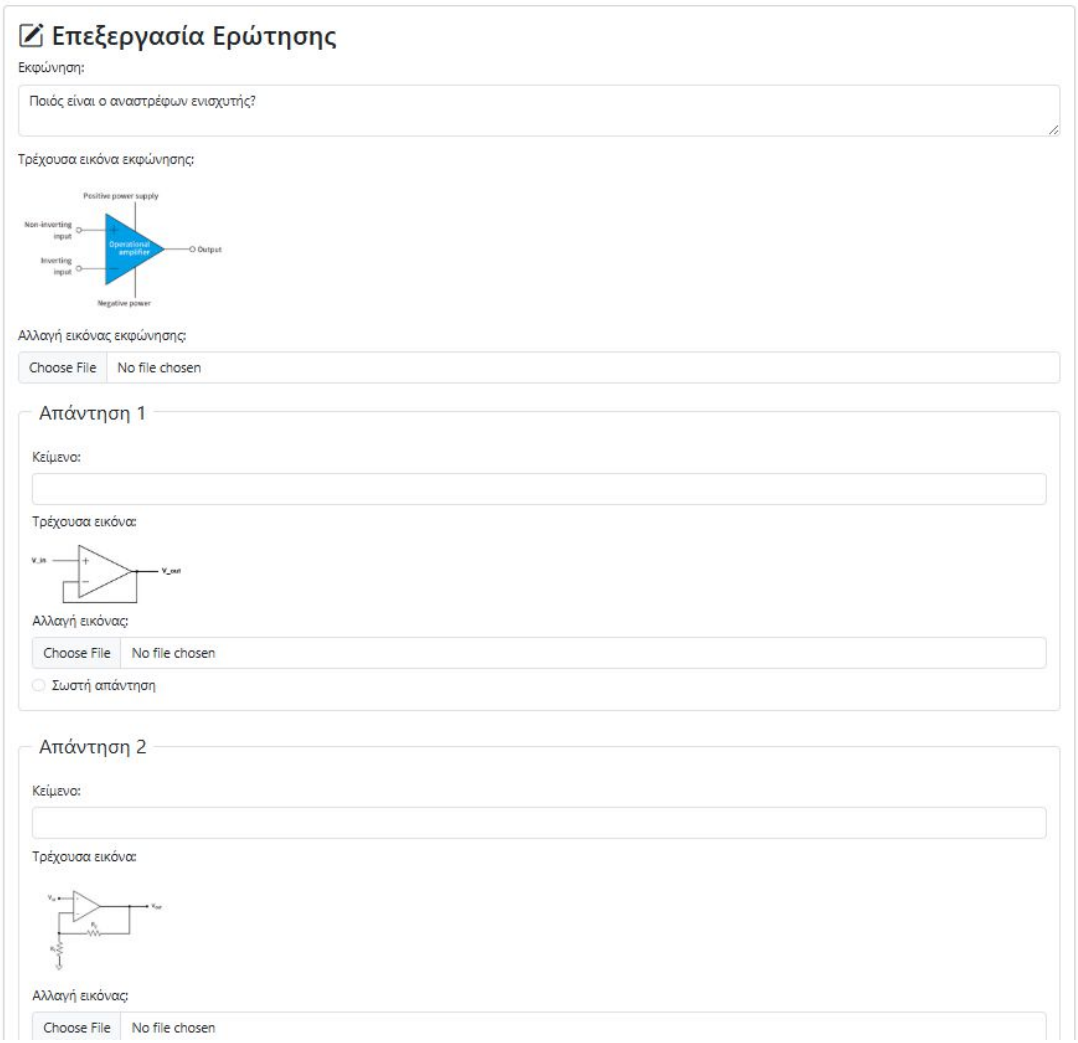
Ο καθηγητής μπορεί να εξάγει τα αποτελέσματα σε αρχείο CSV για να τα επεξεργαστεί σε Excel ή να τα αρχειοθετήσει. Η λειτουργία αυτή είναι χρήσιμη για αναφορές, στατιστική ανάλυση ή αποστολή στοιχείων σε τρίτους.

Εικόνα 4.19: Εξαγωγή σε CVS - Teacher



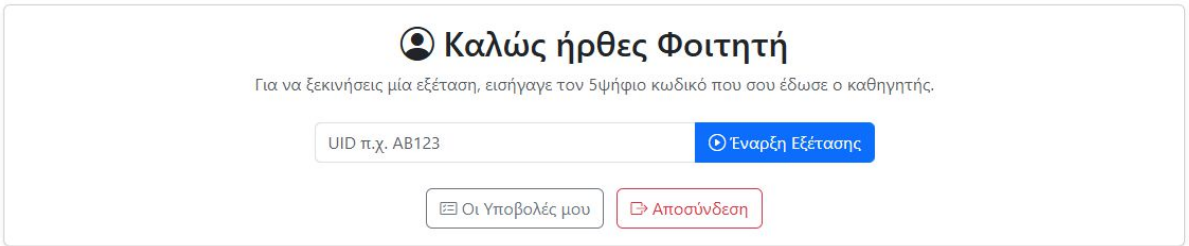
Εικόνα 4.20: Ερωτήσεις με εικόνα στην εκφώνηση - Teacher

Στην Εικόνα 4.20 παρουσιάζεται ερώτηση στην οποία έχει επισυναφθεί εικόνα στην εκφώνηση. Το σύστημα υποστηρίζει την εμφάνιση γραφικών ή διαγραμμάτων κάτι που είναι απαραίτητο σε επιστημονικά αντικείμενα.



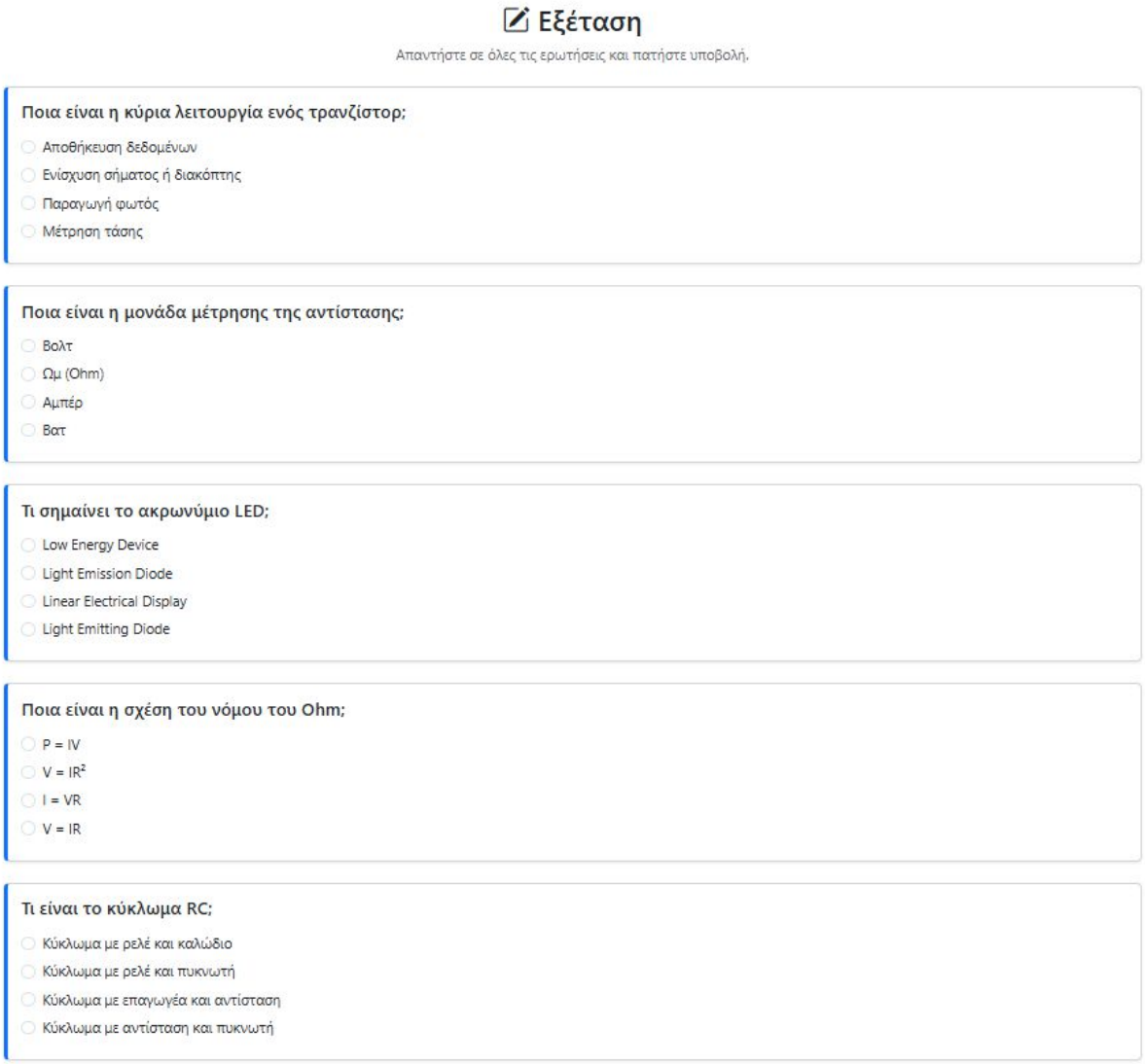
Εικόνα 4.21: Απαντήσεις με Εικόνες στην εκφώνηση κάθε απάντησης - Teacher

Η εικόνα 4.21 παρουσιάζει ερώτηση στην οποία κάθε απάντηση συνοδεύεται από εικόνα. Αυτή η δυνατότητα είναι ιδανική για ερωτήσεις επιλογής αντικειμένου, εικόνας ή διαγράμματος.



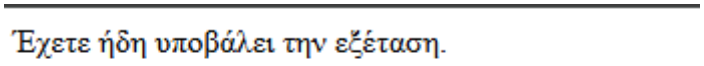
Εικόνα 4.22: Εισαγωγή κωδικού μαθήματος-Quiz για να ξεκινήσει το Quiz - Student

Ο φοιτητής εισάγει τον πενταψήφιο UID που του έχει δοθεί για να ξεκινήσει την εξέταση. Το σύστημα επαληθεύει αν το quiz είναι ενεργό και διαθέσιμο για συμμετοχή.



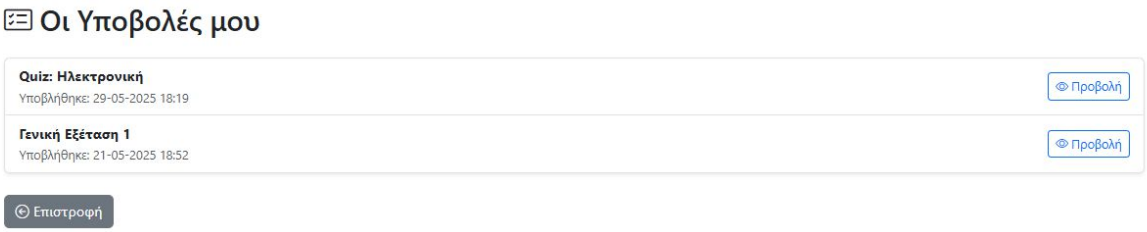
Εικόνα 4.23: Εξέταση στο Quiz - Student

Την Εικόνα 4.23 βλέπουμε: Ο φοιτητής βλέπει τις ερωτήσεις του quiz επιλέγει τις απαντήσεις και υποβάλλει τη συμμετοχή του. Μετά την υποβολή δεν μπορεί να επιστρέψει ή να επεξεργαστεί τις απαντήσεις.



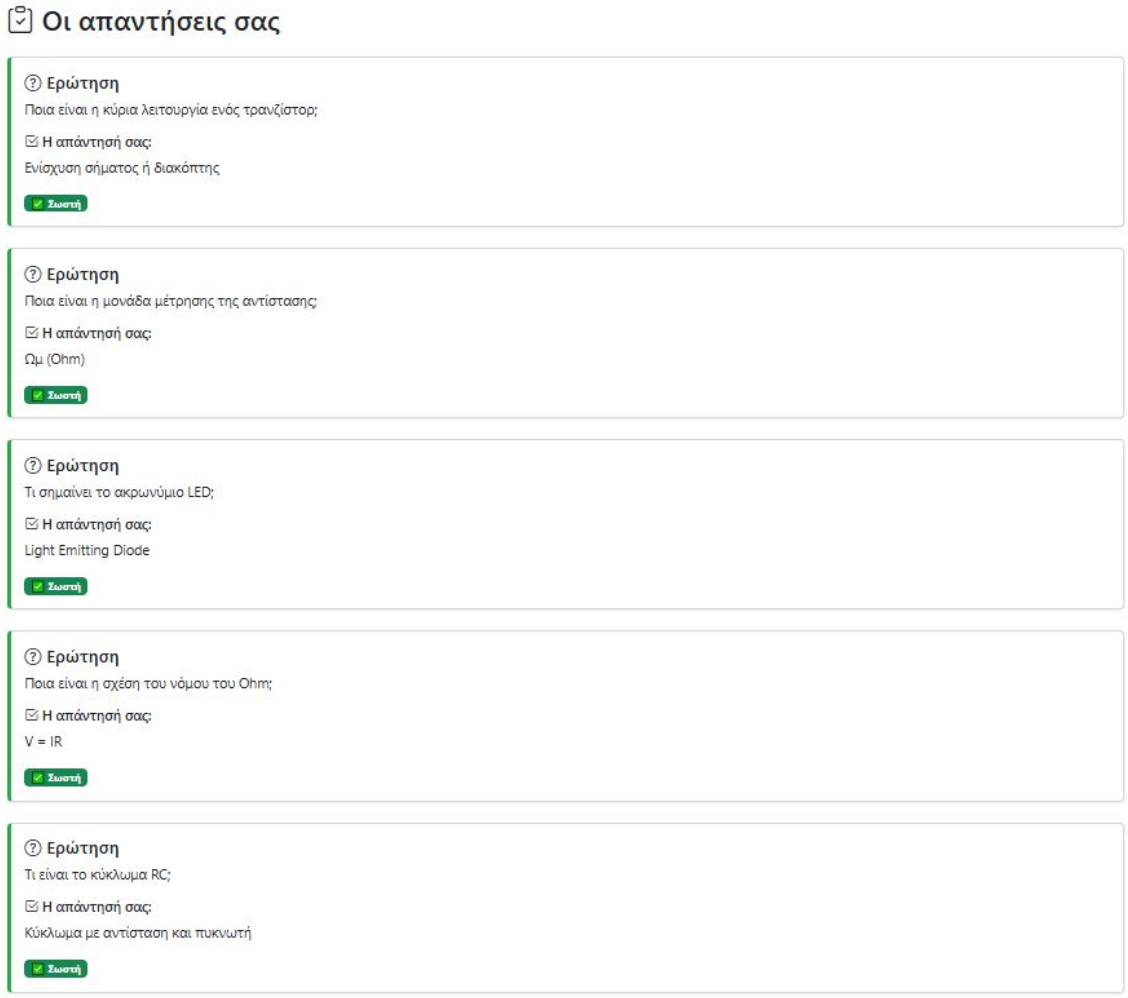
Εικόνα 4.24: Μήνυμα σε περίπτωση επανάληψης - Student

Αν ο φοιτητής προσπαθήσει να συμμετάσχει ξανά στο ίδιο quiz το σύστημα εμφανίζει προειδοποιητικό μήνυμα και μπλοκάρει την ενέργεια (Εικόνα 4.24). Έτσι διασφαλίζεται η μοναδικότητα κάθε συμμετοχής.



Εικόνα 4.25: Οι υποβολές του φοιτητή - Student

Ο φοιτητής μπορεί να δει όλες τις συμμετοχές του σε quiz, με τίτλο, ημερομηνία και κουμπί για προβολή των απαντήσεων (Εικόνα 4.25). Εχει εικόνα του ιστορικού εξετάσεων.



Εικόνα 4.26: Οι απαντήσεις σε Quiz που έχει υποβάλλει - Student

Στην Εικόνα 4.26 παρουσιάζεται η ανάλυση των απαντήσεων του φοιτητή με αναφορά στο αν ήταν σωστές ή λανθασμένες. Εμφανίζονται οι σχετικές εικόνεςαν υπήρχαν. Ο φοιτητής μπορεί έτσι να δει την επίδοσή του αναλυτικά.

4.4 Κώδικας για την υλοποίηση των σημαντικών μερών

Η συνάρτηση submit_quiz(quiz_id) είναι υπεύθυνη για την πιο κρίσιμη λειτουργία του συστήματος: την υποβολή απαντήσεων από φοιτητή. Όταν ο φοιτητής μπαίνει με έναν μοναδικό κωδικό (UID), οδηγείται σε αυτή τη σελίδα όπου εμφανίζονται οι ερωτήσεις του quiz. Αν έχει ήδη υποβάλει εμποδίζεται να ξανασυμμετάσχει για να διασφαλίζεται η ακεραιότητα της εξέτασης.

Η συνάρτηση κάνει διάκριση μεταξύ GET και POST:

- Στο GET, προετοιμάζει τις ερωτήσεις με τις αντίστοιχες απαντήσεις (και εικόνες), τις οργανώνει σε dictionary, και τις στέλνει στο HTML για προβολή.
- Στο POST, διαβάζει τα στοιχεία της φόρμας, αποθηκεύει την κάθε απάντηση στη βάση και οριστικοποιεί την υποβολή.

Είναι σχεδιασμένη ώστε:

- Να αποτρέπει διπλές υποβολές
- Να επιτρέπει πρόχειρη αποθήκευση σε περίπτωση που γίνει διακοπή
- Να υποστηρίζει ερωτήσεις και απαντήσεις με εικόνες
- Να παρέχει ασφαλή εγγραφή στη βάση δεδομένων

Αποτελεί την καρδιά του συστήματος για την πλευρά των φοιτητών, καθώς από αυτήν ξεκινά η διαδικασία αξιολόγησης που θα καταλήξει στην τελική βαθμολόγηση.

```
@app.route('/student/quiz/<int:quiz_id>', methods=['GET', 'POST'])
def submit_quiz(quiz_id):
    # Αν δεν είναι φοιτητής, κάνε redirect
    if session.get('role') != 'student':
        return redirect(url_for('dashboard'))

    user_id = session['user_id']
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Έλεγχος αν έχει ήδη υποβάλει αυτό το quiz
    cursor.execute("""
        SELECT * FROM student_submissions
        WHERE user_id = %s AND quiz_id = %s AND is_submitted = 1
    """, (user_id, quiz_id))
    if cursor.fetchone():
        conn.close()
        return "Έχετε ήδη υποβάλει την εξέταση."

    # Έλεγχος για προσωρινή αποθηκευμένη υποβολή
    cursor.execute("""
        SELECT * FROM student_submissions
        WHERE user_id = %s AND quiz_id = %s
    """, (user_id, quiz_id))
    submission = cursor.fetchone()

    # Αν δεν υπάρχει καν υποβολή, δημιουργείται νέα εγγραφή
    if not submission:
        cursor.execute("""
            INSERT INTO student_submissions (quiz_id, user_id, submission_time, is_submitted)
            VALUES (%s, %s, NOW(), 0)
        """, (quiz_id, user_id))
        submission_id = cursor.lastrowid
    else:
        submission_id = submission['id']

    # Αν πρόκειται για POST, αποθηκεύουμε τις απαντήσεις
```

```
    if request.method == 'POST':
        for key in request.form:
            if key.startswith("question_"):
                qid = key.split("_")[1]
                answer_id = request.form[key]
                cursor.execute("""
                    INSERT INTO student_answers (submission_id, question_id, answer_id)
                    VALUES (%s, %s, %s)
                """, (submission_id, qid, answer_id))

        # Οριστικοποίηση υποβολής
        cursor.execute("""
            UPDATE student_submissions SET is_submitted = 1, submission_time = NOW()
            WHERE id = %s
        """, (submission_id,))
        conn.commit()
        conn.close()
        return "Η υποβολή σας καταχωρήθηκε με επιτυχία."

    # Αν πρόκειται για GET, ετοιμάζουμε τις ερωτήσεις για εμφάνιση
    cursor.execute("""
        SELECT q.id AS qid, q.question_text, q.image_path AS qimg,
               a.id AS aid, a.answer_text, a.image_path AS aimg
        FROM questions q
        JOIN answers a ON a.question_id = q.id
        WHERE q.quiz_id = %s
    """, (quiz_id,))
    rows = cursor.fetchall()
    conn.close()

    if not rows:
        return "Το quiz δεν έχει ερωτήσεις."

    # Ομαδοποίηση ερωτήσεων και απαντήσεων για εμφάνιση στο template
    questions = {}
    for row in rows:
        qid = row['qid']
        if qid not in questions:
            questions[qid] = {
                'text': row['question_text'],
                'image': row['qimg'],
                'answers': []
            }
        questions[qid]['answers'].append({
            'id': row['aid'],
            'text': row['answer_text'],
            'image': row['aimg']
        })

    return render_template('student_quiz.html', quiz_id=quiz_id, questions=questions)
```

Η quiz_results(quiz_id) είναι η βασική συνάρτηση με την οποία ο καθηγητής μπορεί να παρακολουθεί τις επιδόσεις των φοιτητών του σε συγκεκριμένο quiz. Πρόκειται για ένα εργαλείο ανατροφοδότησης και αξιολόγησης και παρέχει ολοκληρωμένη εικόνα των αποτελεσμάτων με πλήρη στατιστικά ανά συμμετοχή.

Η συνάρτηση αρχικά ελέγχει την ταυτότητα του χρήστη ώστε μόνο καθηγητές να έχουν πρόσβαση. Στη συνέχεια αναζητά τον τίτλο του quiz και όλες τις έγκυρες (υποβεβλημένες) συμμετοχές των

φοιτητών. Για κάθε συμμετοχή παίρνουμε το πλήθος των σωστών και λανθασμένων απαντήσεων και υπολογίζεται ποσοστό επιτυχίας με ακρίβεια μίας δεκαδικής.

Τα δεδομένα μορφοποιούνται σε κατάλληλη λίστα dictionaries η οποία αποστέλλεται στο template quiz_results.html για να παρουσιαστεί σε πίνακα με όνομα φοιτητή, ΑΜ, ημερομηνία υποβολής, σκορ και ποσοστό επιτυχίας. Η σχεδίαση επιτρέπει στον εκπαιδευτικό να εντοπίζει αδύνατα σημεία ή αριστούχους εύκολα.

Η συνάρτηση υποστηρίζει επίσης εξαγωγή σε CSV μέσω ξεχωριστού route. Μπορεί μελλοντικά να επεκταθεί για προβολή γραφημάτων, ιστορικών συμμετοχών, ή αναλύσεων / ερώτηση.

Πρόκειται για μια από τις σημαντικότερες λειτουργίες της πλευράς του καθηγητή και επιτρέπει ουσιαστικό έλεγχο και παρακολούθηση της απόδοσης των φοιτητών.

```
@app.route('/teacher/quiz/<int:quiz_id>/results')
def quiz_results(quiz_id):
    # Έλεγχος αν είναι καθηγητής
    if session.get('role') != 'teacher':
        return redirect(url_for('dashboard'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Λήψη τίτλου του quiz
    cursor.execute("SELECT title FROM quizzes WHERE id = %s", (quiz_id,))
    quiz = cursor.fetchone()
    if not quiz:
        conn.close()
        return "Quiz δεν βρέθηκε."

    # Λήψη όλων των υποβολών με πληροφορίες φοιτητών
    cursor.execute("""
        SELECT ss.id AS submission_id, u.name, u.student_number, ss.submission_time
        FROM student_submissions ss
        JOIN users u ON u.id = ss.user_id
        WHERE ss.quiz_id = %s AND ss.is_submitted = 1
    """, (quiz_id,))
    submissions = cursor.fetchall()

    # Υπολογισμός σκορ για κάθε φοιτητή
    results = []
    for s in submissions:
        # Παίρνουμε τις απαντήσεις του φοιτητή
        cursor.execute("""
            SELECT sa.answer_id, a.is_correct
            FROM student_answers sa
            JOIN answers a ON a.id = sa.answer_id
            WHERE sa.submission_id = %s
        """, (s['submission_id'],))
        answers = cursor.fetchall()

        total = len(answers)
        correct = sum(1 for a in answers if a['is_correct'] == 1)
        percent = round((correct / total) * 100, 1) if total else 0

        # Προσθήκη αποτελέσματος στη λίστα
        results.append({
            'name': s['name'],
```

```
            'student_number': s['student_number'],
            'time': s['submission_time'],
            'score': f"{correct}/{total}",
            'percent': percent
        })

    conn.close()
    return render_template('quiz_results.html', quiz=quiz, results=results)
```

Η συνάρτηση quiz_questions(quiz_id) εξυπηρετεί την ανάγκη των καθηγητών να προβάλουν και να διαχειρίζονται τις ερωτήσεις ενός συγκεκριμένου quiz. Είναι μία από τις βασικότερες οθόνες στην καθηγητική διεπαφή καθώς μέσω αυτής ο χρήστης μπορεί να δει με οργανωμένο τρόπο όλες τις ερωτήσεις που έχουν δημιουργηθεί, να προσθέσει νέες, να επεξεργαστεί τις υπάρχουσες ή να τις διαγράψει.

Η συνάρτηση αρχικά ελέγχει ότι ο τρέχων χρήστης είναι καθηγητής για να προστατεύεται από μη εξουσιοδοτημένη πρόσβαση. Μετά γίνεται σύνδεση με τη βάση και ανακτάται ο τίτλος του quiz, ο οποίος χρησιμεύει για την επικεφαλίδα της σελίδας και όλες οι ερωτήσεις που έχουν καταχωρηθεί στο συγκεκριμένο quiz μέσω της quiz_id.

Τα δεδομένα αυτά αποστέλλονται στο αρχείο quiz_questions.html το οποίο παρουσιάζει τις ερωτήσεις σε λίστα. Για κάθε μία υπάρχει δυνατότητα εμφάνισης εικόνας (αν έχει επισυναφθεί), καθώς και κουμπιά επεξεργασίας και διαγραφής. Ο καθηγητής μπορεί επίσης να πατήσει «Νέα Ερώτηση» ώστε να προσθέσει περιεχόμενο στο quiz.

Η συνάρτηση είναι σχεδιασμένη ώστε να είναι επεκτάσιμη — π.χ. μπορεί εύκολα να προστεθεί προβολή αριθμού απαντήσεων ανά ερώτηση, ποσοστά επιτυχίας ή στατιστικά αξιολόγησης. Ακόμη μέσω του session['last_course_id'] μπορεί να γίνει σύνδεση με το μάθημα που περιέχει το quiz, προσφέροντας καλύτερη εμπειρία πλοήγησης.

Αποτελεί ένα κρίσιμο σημείο αλληλεπίδρασης του εκπαιδευτικού με το περιεχόμενο της εξέτασης, προσφέροντας πλήρη εποπτεία και διαχείριση της δομής του quiz.

```
@app.route('/teacher/quiz/<int:quiz_id>/questions')
def quiz_questions(quiz_id):
    # Έλεγχος ότι ο χρήστης είναι καθηγητής
    if session.get('role') != 'teacher':
        return redirect(url_for('dashboard'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Παίρνουμε τον τίτλο του quiz
    cursor.execute("SELECT title FROM quizzes WHERE id = %s", (quiz_id,))
    quiz = cursor.fetchone()
    if not quiz:
        conn.close()
```

```
    return "Το quiz δεν βρέθηκε."

# Παίρνουμε όλες τις ερωτήσεις για το συγκεκριμένο quiz
cursor.execute("SELECT * FROM questions WHERE quiz_id = %s", (quiz_id,))
questions = cursor.fetchall()

conn.close()

# Στέλνουμε τα δεδομένα στο αντίστοιχο template για προβολή
return render_template(
    'quiz_questions.html',
    quiz_id=quiz_id,
    course_id=session.get('last_course_id', 1),
    quiz_title=quiz['title'],
    questions=questions
)
```

Η add_question(quiz_id) αποτελεί το βασικό εργαλείο δημιουργίας περιεχομένου στο σύστημα, καθώς επιτρέπει στους καθηγητές να προσθέσουν νέες ερωτήσεις σε ένα quiz, μαζί με τις αντίστοιχες απαντήσεις. Είναι μία από τις πιο πολύπλοκες και λειτουργικές ροές του έργου, διότι περιλαμβάνει φόρμα με πολλαπλά πεδία, μεταφόρτωση αρχείων εικόνας και εισαγωγές σε δύο πίνακες της βάσης δεδομένων (questions, answers).

Η ροή χωρίζεται σε δύο μέρη:

- Στο GET: επιστρέφεται η φόρμα συμπλήρωσης ερώτησης και απαντήσεων (μέσω template).
- Στο POST: διαβάζονται τα δεδομένα της φόρμας, αποθηκεύεται η εκφώνηση (και η εικόνα αν υπάρχει), δημιουργείται η εγγραφή στον πίνακα questions και ακολουθεί η καταχώρηση των 4 απαντήσεων στον πίνακα answers, με μία από αυτές να σημειώνεται ως σωστή.

Κάθε εικόνα που ανεβαίνει αποθηκεύεται στον φάκελο static/uploads, με ασφαλή όνομα αρχείου ώστε να αποτραπεί πιθανή εκτέλεση κακόβουλου περιεχομένου. Το secure_filename() διασφαλίζει ότι το όνομα είναι αποδεκτό και ασφαλές για αποθήκευση στο server.

Η συνάρτηση είναι δομημένη έτσι ώστε να υποστηρίζει:

- Ερωτήσεις με ή χωρίς εικόνα
- Απαντήσεις με ή χωρίς εικόνα
- Δυνατότητα επιλογής της σωστής απάντησης με radio button

Με την ολοκλήρωση ο καθηγητής μεταφέρεται πίσω στη λίστα ερωτήσεων για το συγκεκριμένο quiz, ώστε να συνεχίσει την προσθήκη ή να τις διαχειριστεί.

Αυτή η λειτουργία βρίσκεται στη δημιουργική διαδικασία του quiz και είναι υπεύθυνη για την εισαγωγή του βασικού εκπαιδευτικού υλικού. Είναι ασφαλής και πλήρως επεκτάσιμη (π.χ. για υποστήριξη ανοικτών ερωτήσεων ή χρονικών περιορισμών μελλοντικά).

```
@app.route('/teacher/quiz/<int:quiz_id>/questions/add', methods=['GET', 'POST'])
def add_question(quiz_id):
    # Έλεγχος ρόλου: μόνο καθηγητές επιτρέπεται να προσθέτουν ερωτήσεις
    if session.get('role') != 'teacher':
        return redirect(url_for('dashboard'))

    if request.method == 'POST':
        # Λήψη κειμένου εκφώνησης
        question_text = request.form['question_text']

        # Αποθήκευση εικόνας εκφώνησης (αν υπάρχει)
        question_image = request.files['question_image']
        qimg_filename = None
        if question_image and question_image.filename:
            qimg_filename = secure_filename(question_image.filename)
            question_image.save(os.path.join('static/uploads', qimg_filename))

        # Σύνδεση με βάση και καταχώρηση ερώτησης
        conn = get_db_connection()
        cursor = conn.cursor()
        cursor.execute("""
            INSERT INTO questions (quiz_id, question_text, image_path)
            VALUES (%s, %s, %s)
        """, (quiz_id, question_text, qimg_filename))
        question_id = cursor.lastrowid

        # Λήψη αριθμού σωστής απάντησης από radio button
        correct_index = int(request.form['correct_answer'])

        # Καταχώρηση 4 απαντήσεων
        for i in range(1, 5):
            answer_text = request.form.get(f'answer_text_{i}')
            answer_image = request.files.get(f'answer_image_{i}')
            aimg_filename = None
            if answer_image and answer_image.filename:
                aimg_filename = secure_filename(answer_image.filename)
                answer_image.save(os.path.join('static/uploads', aimg_filename))

            is_correct = 1 if i == correct_index else 0

            cursor.execute("""
                INSERT INTO answers (question_id, answer_text, image_path, is_correct)
                VALUES (%s, %s, %s, %s)
            """, (question_id, answer_text, aimg_filename, is_correct))

        conn.commit()
        conn.close()

        # Μετά την προσθήκη, redirect πίσω στη λίστα ερωτήσεων
        return redirect(url_for('quiz_questions', quiz_id=quiz_id))

    # GET μέθοδος: εμφάνιση φόρμας
    return render_template('teacher_add_question.html', quiz_id=quiz_id)
```

Η teacher_courses() αποτελεί το σημείο εισόδου του καθηγητή στο σύστημα και είναι η πρώτη οθόνη που βλέπει μετά την επιτυχημένη σύνδεση. Ο σκοπός της είναι να προβάλλει όλα τα μαθήματα στα οποία έχει ανατεθεί ο συγκεκριμένος καθηγητής από τον διαχειριστή του συστήματος. Είναι μια συνάρτηση απλή και ουσιαστική σε λειτουργία και καθορίζει την πλοήγηση του εκπαιδευτικού προς τα quiz και τις ερωτήσεις κάθε μαθήματος.

Στην αρχή η συνάρτηση ελέγχει αν ο χρήστης είναι καθηγητής (μέσω του session['role']). Αν δεν είναι γίνεται ανακατεύθυνση στο dashboard για αποτροπή μη εξουσιοδοτημένης πρόσβασης. Μετά ανακτάται το user_id από το session και πραγματοποιείται ερώτημα προς τη βάση για να ληφθούν όλα τα μαθήματα που σχετίζονται με τον καθηγητή.

Η σχέση των μαθημάτων με τον χρήστη αποθηκεύεται στον πίνακα courseteacher, επομένως το query κάνει join με τον πίνακα courses για να ανακτήσει τα πλήρη στοιχεία (τίτλο, περιγραφή, ID κ.λπ.) του κάθε μαθήματος.

Η λίστα που προκύπτει αποστέλλεται στο template teacher_courses.html, το οποίο προβάλλει κάθε μάθημα μέσα σε Bootstrap κάρτες, παρέχοντας κουμπί για «Διαχείριση Quizzes» ανά μάθημα. Αυτό δίνει μια φιλική εμπειρία στον καθηγητή και τον διευκολύνει να πλοηγηθεί γρήγορα στην ενότητα του κάθε quiz.

Αυτή η συνάρτηση είναι ουσιαστικά το "κέντρο ελέγχου" του εκπαιδευτικού αφού από εδώ ξεκινάει η διαχείριση του εκπαιδευτικού περιεχομένου του.

```
@app.route('/teacher/courses')
def teacher_courses():
    # Έλεγχος ρόλου: επιτρέπεται μόνο σε καθηγητές
    if session.get('role') != 'teacher':
        return redirect(url_for('dashboard'))

    # Λήψη του ID του καθηγητή από το session
    user_id = session['user_id']

    # Σύνδεση με τη βάση
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Λήψη μαθημάτων στα οποία έχει ανατεθεί ο χρήστης
    cursor.execute("""
        SELECT courses.* FROM courses
        JOIN courseteacher ON courseteacher.course_id = courses.id
        WHERE courseteacher.user_id = %s
    """, (user_id,))
    courses = cursor.fetchall()

    # Κλείσιμο σύνδεσης
    conn.close()

    # Απόδοση template με τα μαθήματα
    return render_template('teacher_courses.html', courses=courses)
```

Οι υπόλοιπες συναρτήσεις βρίσκονται στο Παράρτημα στο τέλος.

4.5 Ασφάλεια

Η ασφάλεια αποτελεί θεμελιώδη πυλώνα για την επιτυχημένη λειτουργία κάθε καλής διαδικτυακής εφαρμογής ειδικά όταν αυτή διαχειρίζεται ευαίσθητα δεδομένα όπως προσωπικά στοιχεία χρηστών, εκπαιδευτικό υλικό και εξεταστικά αποτελέσματα. Το σύστημα διαχείρισης quiz που αναπτύχθηκε σχεδιάστηκε εξαρχής με σκοπό να εξασφαλίζει την προστασία των δεδομένων, την ακεραιότητα των εξεταστικών διαδικασιών και τη σωστή διαχείριση των δικαιωμάτων κάθε ρόλου.

Αρχικά η πρόσβαση στο σύστημα ελέγχεται μέσω μηχανισμού ταυτοποίησης χρηστών. Κάθε χρήστης — είτε είναι διαχειριστής, καθηγητής ή φοιτητής — πρέπει να συνδεθεί με προσωπικό email και κωδικό πρόσβασης. Οι κωδικοί αποθηκεύονται στη βάση δεδομένων σε μορφή κρυπτογραφημένη (hashed), χρησιμοποιώντας τον αλγόριθμο bcrypt για να μην είναι ανακτήσιμοι ούτε από διαχειριστές του συστήματος. Με αυτόν τον τρόπο ακόμα και σε περίπτωση χακαρίσματος της βάσης οι κωδικοί δεν μπορούν να αποκαλυφθούν. Για κάθε σύνδεση γίνεται έλεγχος του ρόλου του χρήστη για να έχει πρόσβαση μόνο στις σελίδες που του αναλογούν.

Το σύστημα διαχωρίζει ξεκάθαρα τους ρόλους και τα δικαιώματα κάθε χρήστη. Ο διαχειριστής έχει πλήρη πρόσβαση στα δεδομένα των μαθημάτων και των καθηγητών χωρίς να μπορεί να παρέμβει στις υποβολές των φοιτητών. Ο καθηγητής έχει πρόσβαση μόνο στα quizzes και τα μαθήματα στα οποία του έχουν ανατεθεί και δεν έχει πρόσβαση στις λειτουργίες διαχείρισης χρηστών ή δημιουργίας μαθημάτων. Ο φοιτητής με τη σειρά του μπορεί να συμμετέχει μόνο μία φορά σε κάθε quiz δεν έχει δυνατότητα επεξεργασίας μετά την υποβολή και δεν μπορεί να δει τα αποτελέσματα άλλων συμμετεχόντων. Αυτοί οι περιορισμοί εφαρμόζονται τόσο στο backend όσο και στην εμφάνιση του frontend.

Όσον αφορά τις εξετάσεις το σύστημα έχει αυστηρό έλεγχο μοναδικότητας. Κάθε quiz συνοδεύεται από έναν μοναδικό πενταψήφιο κωδικό (UID) τον οποίο δίνει ο καθηγητής στους φοιτητές. Ο κωδικός αυτός είναι τυχαία παραγόμενος και δεν επαναλαμβάνεται. Ο φοιτητής αφού τον εισάγει μπορεί να συμμετάσχει στην εξέταση μόνο εφόσον το quiz είναι ενεργό και δεν έχει ήδη συμμετάσχει. Μετά την υποβολή η συμμετοχή κλειδώνει ενώ οι απαντήσεις αποθηκεύονται με ασφάλεια στη βάση και δεν επιδέχονται αλλαγές. Με αυτόν τον τρόπο εξασφαλίζεται η ακεραιότητα της εξέτασης και αποφεύγονται φαινόμενα απάτης ή διπλών συμμετοχών.

Η εφαρμογή προστατεύεται από επιθέσεις τύπου CSRF και SQL Injection. Όλα τα δεδομένα που εισάγονται από φόρμες περνάνε έλεγχο και αποφυγή επικίνδυνων εντολών. Ακόμη χρησιμοποιείται ασφαλής τρόπος σύνταξης ερωτημάτων στη βάση (με χρήση prepared statements και parameterized queries) για να αποτραπεί η εκτέλεση κακόβουλων SQL εντολών. Τα uploads εικόνων για τις ερωτήσεις

και τις απαντήσεις φιλτράρονται και αποθηκεύονται μόνο εφόσον πληρούν συγκεκριμένα κριτήρια τύπου και μεγέθους αρχείου.

Σε επίπεδο προστασίας από bots και spam, εφαρμόζονται περιορισμοί σε κρίσιμες λειτουργίες, όπως η υποβολή εξετάσεων ή η σύνδεση. Προαιρετικά μπορεί να ενσωματωθεί honeypot πεδίο ή ακόμα και μηχανισμός reCAPTCHA για ενισχυμένη προστασία. Επίσης η πρόσβαση σε routes γίνεται αποκλειστικά μέσω κατάλληλου session και ελέγχου ταυτότητας ώστε χρήστες που επιχειρούν να εισάγουν URL απευθείας χωρίς σύνδεση να ανακατευθύνονται στην αρχική σελίδα.

Η ασφάλεια του συστήματος αποτελεί βασική προτεραιότητα. Με τον διαχωρισμό ρόλων, την προστασία των δεδομένων σύνδεσης, τη λογική κλειδώματος των υποβολών, τη χρήση ασφαλών τεχνικών επικοινωνίας με τη βάση και την αποφυγή επιθέσεων injection, διαμορφώνεται ένα περιβάλλον αξιόπιστο, λειτουργικό και κατάλληλο για αξιολόγηση φοιτητών με ψηφιακό τρόπο.

Κεφάλαιο 5ο: Συμπεράσματα και προτάσεις βελτίωσης

Η ανάπτυξη της πλατφόρμας δημιουργίας και διαχείρισης quiz για εκπαιδευτικά ιδρύματα αποτέλεσε ένα έργο που συνδύασε τεχνολογίες backend, frontend, βάσεις δεδομένων και ασφαλής ανάπτυξη. Η υλοποίηση βασίστηκε σε πραγματικές ανάγκες του εκπαιδευτικού περιβάλλοντος και καλύπτει επαρκώς τη διαδικασία δημιουργίας εξετάσεων από καθηγητές, συμμετοχής φοιτητών με μοναδικό κωδικό, αποθήκευσης και επεξεργασίας αποτελεσμάτω και διαχείρισης των μαθημάτων από διαχειριστή. Η λειτουργικότητα που επιτεύχθηκε εξασφαλίζει έναν οργανωμένο και αποτελεσματικό τρόπο αξιολόγησης απλοποιώντας παράλληλα τη διαχείριση των εμπλεκόμενων ρόλων.

Κατά την πορεία του έργου διαπιστώθηκε ότι ο σχεδιασμός που βασίστηκε στην απλότητα και την ταχύτητα ανταπόκρισης του χρήστη είχε καλά αποτελέσματα. Το σύστημα είναι λειτουργικό, αξιόπιστο και ασφαλές, προσφέροντας σαφή διαχωρισμό ρόλων, ευχρηστία στη διεπαφή και αξιόπιστη αποθήκευση των δεδομένων. Οι λειτουργίες που παρέχονται καλύπτουν το σύνολο των βασικών απαιτήσεων μιας ψηφιακής πλατφόρμας αξιολόγησης. Το γεγονός ότι ο κάθε φοιτητής μπορεί να συμμετάσχει μόνο μία φορά σε κάθε quiz, ότι οι ερωτήσεις υποστηρίζουν και πολυμέσα (εικόνες) και ότι όλα τα αποτελέσματα είναι διαθέσιμα για εξαγωγή και περαιτέρω επεξεργασία από τον καθηγητή, συμβάλλει ουσιαστικά στη διαφάνεια και την αξιοπιστία των εξετάσεων.

Όπως σε κάθε σύστημα πληροφορικής υπάρχει πάντα περιθώριο για περαιτέρω βελτίωση. Μία από τις βασικές προτάσεις αφορά την ενσωμάτωση ενός συστήματος ειδοποιήσεων ώστε καθηγητές και φοιτητές να ενημερώνονται σε πραγματικό χρόνο για νέες εξετάσεις, αλλαγές σε quiz ή νέες υποβολές. Η δυνατότητα μαζικής εισαγωγής ερωτήσεων από αρχείο Excel ή CSV θα διευκόλυνε σημαντικά τη διαδικασία για τον εκπαιδευτικό ειδικά σε περιπτώσεις μεγάλου όγκου περιεχομένου. Άλλη χρήσιμη επέκταση θα ήταν η προσθήκη στατιστικής ανάλυσης για τα αποτελέσματα όπως ο μέσος όρος ανά ερώτηση, ποσοστά σωστών/λανθασμένων απαντήσεων και αναφορές με τη μορφή διαγραμμάτων. Η δυνατότητα ανωνυμοποίησης των υποβολών για τις ανάγκες κρυφής αξιολόγησης ή η εισαγωγή χρονικού περιορισμού ανά ερώτηση είναι καλές ιδεες που θα μπορούσαν να βελτιώσουν τη εφαρμογή.

Ακόμη θα ήταν καλή η υλοποίηση mobile-friendly έκδοσης με responsive σχεδιασμό που θα επιτρέπει στους φοιτητές να συμμετέχουν σε εξετάσεις και από φορητές συσκευές. Η διεπαφή έχει ήδη υλοποιηθεί με Bootstrap και περαιτέρω βελτιστοποίηση θα προσφέρει καλύτερη εμπειρία χρήσης. Επίσης η δημιουργία native mobile εφαρμογής με notifications, offline λειτουργία και συγχρονισμό απαντήσεων θα μπορούσε να είναι το επόμενο βήμα για πλήρη χρήση των δυνατοτήτων της πλατφόρμας.

Σε μελλοντική έκδοση θα μπορεί να ενσωματωθεί υποσύστημα βαθμολόγησης ανοικτών ερωτήσεων, όπου ο καθηγητής θα μπορεί να αξιολογεί προσωπικά τις απαντήσεις φοιτητών που απαιτούν ελεύθερο κείμενο. Αυτό σε συνδυασμό με σύστημα σχολιασμού ανά απάντηση θα έκανε το σύστημα καλύτερο. Να σημειωθεί ότι για την συγγραφή της εργασίας έχει χρησιμοποιηθεί ai μηχανή για τη διόρθωση συντακτικού στο κείμενο.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] https://moodle.org/

[2] https://tcexam.org/

[3] https://www.openeclass.org/

[4] https://www.learnpython.org/

[5] https://www.squareboat.com/blog/advantages-and-disadvantages-of-python

[6] https://www.w3schools.com/MySQL/default.asp

[7] https://medium.com/@josipvojak/exploring-the-pros-and-cons-of-sql-databases-mysql-postgres-oracle-microsoft-sql-and-amazon-3c8de880b8d4

[8] https://www.w3schools.com/html/html_css.asp

[9] https://www.tutorialrepublic.com/twitter-bootstrap-tutorial/

ΠΑΡΑΡΤΗΜΑ ΚΩΔΙΚΑ

```
from flask import Flask, render_template, request, redirect, session, url_for
from db_config import get_db_connection
import hashlib
import random
import string
import os
import io
from werkzeug.utils import secure_filename

import csv
from flask import make_response

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

@app.route('/', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = hashlib.sha256(request.form['password'].encode()).hexdigest()

        conn = get_db_connection()
        cursor = conn.cursor(dictionary=True)
        cursor.execute("SELECT * FROM users WHERE email = %s AND password = %s", (email, password))
        user = cursor.fetchone()
        conn.close()

        if user:
            session['user_id'] = user['id']
            session['role'] = user['role']
            return redirect(url_for('dashboard'))
        else:
            return render_template('login.html', error="Λάθος στοιχεία!")

    return render_template('login.html')

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

    role = session['role']
    if role == 'admin':
        return render_template('dashboard_admin.html')
    elif role == 'teacher':
        return render_template('dashboard_teacher.html')
    elif role == 'student':
        return render_template('dashboard_student.html')
    else:
        return "Άγνωστος ρόλος."

@app.route('/logout')
def logout():
    session.clear()
```

```
    return redirect(url_for('login'))

@app.route('/admin/courses')
def admin_courses():
    if session.get('role') != 'admin':
        return redirect(url_for('dashboard'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)
    cursor.execute("SELECT * FROM courses")
    courses = cursor.fetchall()
    conn.close()
    return render_template('admin_courses.html', courses=courses)

@app.route('/admin/courses/create', methods=['GET', 'POST'])
def create_course():
    if session.get('role') != 'admin':
        return redirect(url_for('dashboard'))

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

        conn = get_db_connection()
        cursor = conn.cursor()
        cursor.execute("INSERT INTO courses (title, description) VALUES (%s, %s)", (title, description))
        conn.commit()
        conn.close()
        return redirect(url_for('admin_courses'))

    return render_template('admin_create_course.html')

@app.route('/admin/courses/<int:course_id>/teachers', methods=['GET', 'POST'])
def assign_teachers(course_id):
    if session.get('role') != 'admin':
        return redirect(url_for('dashboard'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Όλοι οι καθηγητές
    cursor.execute("SELECT * FROM users WHERE role = 'teacher'")
    all_teachers = cursor.fetchall()

    # Οι ανατεθειμένοι
    cursor.execute("SELECT user_id FROM courseteacher WHERE course_id = %s", (course_id,))
    assigned_ids = {row['user_id'] for row in cursor.fetchall()}

    if request.method == 'POST':
        selected_ids = request.form.getlist('teacher_ids')
        selected_ids = set(map(int, selected_ids))

        # Διαγραφή παλιών
        cursor.execute("DELETE FROM courseteacher WHERE course_id = %s", (course_id,))

        # Προσθήκη νέων
        for uid in selected_ids:
            cursor.execute("INSERT INTO courseteacher (course_id, user_id) VALUES (%s, %s)", (course_id, uid))
```

```
            conn.commit()
            conn.close()
            return redirect(url_for('admin_courses'))

        conn.close()
        return render_template('admin_assign_teachers.html', course_id=course_id, teachers=all_teachers, assigned=assigned_ids)

@app.route('/teacher/courses')
def teacher_courses():
    if session.get('role') != 'teacher':
        return redirect(url_for('dashboard'))

    user_id = session['user_id']
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)
    cursor.execute("""
        SELECT courses.* FROM courses
        JOIN courseteacher ON courseteacher.course_id = courses.id
        WHERE courseteacher.user_id = %s
    """, (user_id,))
    courses = cursor.fetchall()
    conn.close()
    return render_template('teacher_courses.html', courses=courses)

@app.route('/teacher/courses/<int:course_id>/quizzes')
def course_quizzes(course_id):
    if session.get('role') != 'teacher':
        return redirect(url_for('dashboard'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Check if this course belongs to the teacher
    cursor.execute("""
        SELECT * FROM courseteacher
        WHERE course_id = %s AND user_id = %s
    """, (course_id, session['user_id']))
    if not cursor.fetchone():
        conn.close()
        return "Δεν έχετε πρόσβαση σε αυτό το μάθημα."

    # Get course title
    cursor.execute("SELECT title FROM courses WHERE id = %s", (course_id,))
    course_row = cursor.fetchone()
    course_title = course_row['title'] if course_row else 'Άγνωστο Μάθημα'

    # Get the quizzes
    cursor.execute("SELECT * FROM quizzes WHERE course_id = %s", (course_id,))
    quizzes = cursor.fetchall()
    conn.close()

    return render_template('teacher_quizzes.html', course_id=course_id, course_title=course_title, quizzes=quizzes)

def generate_uid():
    return ''.join(random.choices(string.ascii_uppercase + string.digits, k=5))

@app.route('/teacher/courses/<int:course_id>/quizzes/create', methods=['GET', 'POST'])
```

```
def create_quiz(course_id):
    if session.get('role') != 'teacher':
        return redirect(url_for('dashboard'))

    if request.method == 'POST':
        title = request.form['title']
        deadline_start = request.form['deadline_start']
        deadline_end = request.form['deadline_end']
        uid = generate_uid()

        conn = get_db_connection()
        cursor = conn.cursor()
        cursor.execute("""
            INSERT INTO quizzes (course_id, title, uid, deadline_start, deadline_end)
            VALUES (%s, %s, %s, %s, %s)
        """, (course_id, title, uid, deadline_start, deadline_end))
        conn.commit()
        conn.close()
        return redirect(url_for('course_quizzes', course_id=course_id))

    return render_template('teacher_create_quiz.html', course_id=course_id)

@app.route('/teacher/quiz/<int:quiz_id>/questions')
def quiz_questions(quiz_id):
    if session.get('role') != 'teacher':
        return redirect(url_for('dashboard'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Get quiz details
    cursor.execute("SELECT title, course_id FROM quizzes WHERE id = %s", (quiz_id,))
    quiz = cursor.fetchone()
    if not quiz:
        conn.close()
        return "Το quiz δεν βρέθηκε."

    course_id = quiz['course_id']

    # Get questions
    cursor.execute("SELECT * FROM questions WHERE quiz_id = %s", (quiz_id,))
    questions = cursor.fetchall()

    conn.close()
    return render_template(
        'quiz_questions.html',
        quiz_id=quiz_id,
        course_id=course_id,
        quiz_title=quiz['title'],
        questions=questions
    )

UPLOAD_FOLDER = 'static/uploads'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

@app.route('/teacher/quiz/<int:quiz_id>/questions/add', methods=['GET', 'POST'])
def add_question(quiz_id):
```

```
    if session.get('role') != 'teacher':
        return redirect(url_for('dashboard'))

    if request.method == 'POST':
        text = request.form['question_text']
        image = request.files.get('question_image')
        image_path = None

        if image and image.filename != '':
            filename = secure_filename(image.filename)
            image.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
            image_path = filename

        conn = get_db_connection()
        cursor = conn.cursor()
        cursor.execute("INSERT INTO questions (quiz_id, question_text, image_path) VALUES (%s, %s, %s)",
                       (quiz_id, text, image_path))
        question_id = cursor.lastrowid

        # Προσθήκη 4 απαντήσεων
        for i in range(1, 5):
            ans_text = request.form.get(f'answer_text_{i}')
            ans_img = request.files.get(f'answer_image_{i}')
            ans_img_path = None
            if ans_img and ans_img.filename != '':
                ans_filename = secure_filename(ans_img.filename)
                ans_img.save(os.path.join(app.config['UPLOAD_FOLDER'], ans_filename))
                ans_img_path = ans_filename
            is_correct = 1 if request.form.get('correct_answer') == str(i) else 0
            cursor.execute("""
                INSERT INTO answers (question_id, answer_text, image_path, is_correct)
                VALUES (%s, %s, %s, %s)
            """, (question_id, ans_text, ans_img_path, is_correct))

        conn.commit()
        conn.close()
        return redirect(url_for('quiz_questions', quiz_id=quiz_id))

    return render_template('add_question.html', quiz_id=quiz_id)

@app.route('/teacher/question/<int:question_id>/edit', methods=['GET', 'POST'])
def edit_question(question_id):
    if session.get('role') != 'teacher':
        return redirect(url_for('dashboard'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Πάρε την ερώτηση και τις απαντήσεις
    cursor.execute("SELECT * FROM questions WHERE id = %s", (question_id,))
    question = cursor.fetchone()
    if not question:
        conn.close()
        return "Η ερώτηση δεν βρέθηκε."

    quiz_id = question['quiz_id']

    cursor.execute("SELECT * FROM answers WHERE question_id = %s ORDER BY id ASC", (question_id,))
```

```
    answers = cursor.fetchall()

    if request.method == 'POST':
        # Εκφώνηση
        question_text = request.form['question_text']
        qimg = request.files.get('question_image')
        qimg_path = question['image_path']

        if qimg and qimg.filename:
            filename = secure_filename(qimg.filename)
            qimg.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
            qimg_path = filename

        cursor.execute("""
            UPDATE questions SET question_text = %s, image_path = %s WHERE id = %s
        """, (question_text, qimg_path, question_id))

        # Απαντήσεις
        for i, ans in enumerate(answers, start=1):
            aid = ans['id']
            new_text = request.form.get(f'answer_text_{i}')
            new_img = request.files.get(f'answer_image_{i}')
            img_path = ans['image_path']

            if new_img and new_img.filename:
                filename = secure_filename(new_img.filename)
                new_img.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
                img_path = filename

            is_correct = 1 if request.form.get('correct_answer') == str(i) else 0

            cursor.execute("""
                UPDATE answers
                SET answer_text = %s, image_path = %s, is_correct = %s
                WHERE id = %s
            """, (new_text, img_path, is_correct, aid))

        conn.commit()
        conn.close()
        return redirect(url_for('quiz_questions', quiz_id=quiz_id))

    conn.close()
    return render_template('edit_question.html', question=question, answers=answers)

@app.route('/student/start', methods=['GET', 'POST'])
def start_quiz():
    if session.get('role') != 'student':
        return redirect(url_for('dashboard'))

    if request.method == 'POST':
        uid = request.form['uid'].strip().upper()

        conn = get_db_connection()
        cursor = conn.cursor(dictionary=True)
        cursor.execute("""
            SELECT * FROM quizzes
            WHERE uid = %s AND published = 1 AND NOW() BETWEEN deadline_start AND deadline_end
        """, (uid,))
```

```
            quiz = cursor.fetchone()
            conn.close()

            if quiz:
                return redirect(url_for('submit_quiz', quiz_id=quiz['id']))
            else:
                return render_template('student_start_quiz.html', error="Μη έγκυρος ή μη διαθέσιμος κωδικός.")

        return render_template('student_start_quiz.html')

    @app.route('/student/quiz/<int:quiz_id>', methods=['GET', 'POST'])
    def submit_quiz(quiz_id):
        if session.get('role') != 'student':
            return redirect(url_for('dashboard'))

        user_id = session['user_id']
        conn = get_db_connection()
        cursor = conn.cursor(dictionary=True)

        # Έχει ήδη υποβάλει;
        cursor.execute("""
            SELECT * FROM student_submissions
            WHERE user_id = %s AND quiz_id = %s AND is_submitted = 1
        """, (user_id, quiz_id))
        if cursor.fetchone():
            conn.close()
            return "Έχετε ήδη υποβάλει την εξέταση."

        # Υπάρχει προσωρινή;
        cursor.execute("""
            SELECT * FROM student_submissions
            WHERE user_id = %s AND quiz_id = %s
        """, (user_id, quiz_id))
        submission = cursor.fetchone()

        if not submission:
            cursor.execute("""
                INSERT INTO student_submissions (quiz_id, user_id, submission_time, is_submitted)
                VALUES (%s, %s, NOW(), 0)
            """, (quiz_id, user_id))
            submission_id = cursor.lastrowid
        else:
            submission_id = submission['id']

        if request.method == 'POST':
            # Καταγραφή απαντήσεων
            for key in request.form:
                if key.startswith("question_"):
                    qid = key.split("_")[1]
                    answer_id = request.form[key]
                    cursor.execute("""
                        INSERT INTO student_answers (submission_id, question_id, answer_id)
                        VALUES (%s, %s, %s)
                    """, (submission_id, qid, answer_id))

            # Οριστικοποίηση
            cursor.execute("""
                UPDATE student_submissions SET is_submitted = 1, submission_time = NOW()
```

```
            WHERE id = %s
        """, (submission_id,))
        conn.commit()
        conn.close()
        return "Η υποβολή σας καταχωρήθηκε με επιτυχία."

    # GET → εμφάνιση ερωτήσεων
    cursor.execute("""
        SELECT q.id AS qid, q.question_text, q.image_path AS qimg,
               a.id AS aid, a.answer_text, a.image_path AS aimg
        FROM questions q
        JOIN answers a ON a.question_id = q.id
        WHERE q.quiz_id = %s
    """, (quiz_id,))
    rows = cursor.fetchall()
    conn.close()

    if not rows:
        return "Το quiz δεν έχει ερωτήσεις."

    # Ομαδοποίηση ερωτήσεων
    questions = {}
    for row in rows:
        qid = row['qid']
        if qid not in questions:
            questions[qid] = {
                'text': row['question_text'],
                'image': row['qimg'],
                'answers': []
            }
        questions[qid]['answers'].append({
            'id': row['aid'],
            'text': row['answer_text'],
            'image': row['aimg']
        })

    return render_template('student_quiz.html', quiz_id=quiz_id, questions=questions)

    # POST: αποθήκευση απαντήσεων

@app.route('/teacher/quiz/<int:quiz_id>/results')
def quiz_results(quiz_id):
    if session.get('role') != 'teacher':
        return redirect(url_for('dashboard'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Πάρε τίτλο quiz
    cursor.execute("SELECT id, title FROM quizzes WHERE id = %s", (quiz_id,))
    quiz = cursor.fetchone()
    if not quiz:
        conn.close()
        return "Quiz δεν βρέθηκε."

    # Πάρε όλες τις υποβολές
    cursor.execute("""
        SELECT ss.id AS submission_id, u.name, u.student_number, ss.submission_time
```

```
        FROM student_submissions ss
        JOIN users u ON u.id = ss.user_id
        WHERE ss.quiz_id = %s AND ss.is_submitted = 1
    """, (quiz_id,))
    submissions = cursor.fetchall()

    # Για κάθε υποβολή, υπολόγισε σκορ
    results = []
    for s in submissions:
        cursor.execute("""
            SELECT sa.answer_id, a.is_correct
            FROM student_answers sa
            JOIN answers a ON a.id = sa.answer_id
            WHERE sa.submission_id = %s
        """, (s['submission_id'],))
        answers = cursor.fetchall()
        total = len(answers)
        correct = sum(1 for a in answers if a['is_correct'] == 1)
        percent = round((correct / total) * 100, 1) if total else 0

        results.append({
            'name': s['name'],
            'student_number': s['student_number'],
            'time': s['submission_time'],
            'score': f"{correct}/{total}",
            'percent': percent
        })

    conn.close()
    return render_template('quiz_results.html', quiz=quiz, results=results)

@app.route('/teacher/quiz/<int:quiz_id>/edit', methods=['GET', 'POST'])
def edit_quiz(quiz_id):
    if session.get('role') != 'teacher':
        return redirect(url_for('dashboard'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # GET ή POST
    if request.method == 'POST':
        title = request.form['title']
        deadline_start = request.form['deadline_start']
        deadline_end = request.form['deadline_end']

        cursor.execute("""
            UPDATE quizzes
            SET title = %s, deadline_start = %s, deadline_end = %s
            WHERE id = %s
        """, (title, deadline_start, deadline_end, quiz_id))

        conn.commit()
        conn.close()
        return redirect(url_for('course_quizzes', course_id=session.get('last_course_id', 1)))

    # GET: φόρτωσε τα δεδομένα του quiz
    cursor.execute("SELECT * FROM quizzes WHERE id = %s", (quiz_id,))
    quiz = cursor.fetchone()
```

```
    conn.close()

    if not quiz:
        return "Το quiz δεν βρέθηκε."

    return render_template('edit_quiz.html', quiz=quiz)

@app.route('/teacher/quiz/<int:quiz_id>/delete')
def delete_quiz(quiz_id):
    if session.get('role') != 'teacher':
        return redirect(url_for('dashboard'))

    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("DELETE FROM quizzes WHERE id = %s", (quiz_id,))
    conn.commit()
    conn.close()
    return redirect(url_for('teacher_courses'))

@app.route('/teacher/question/<int:question_id>/delete')
def delete_question(question_id):
    if session.get('role') != 'teacher':
        return redirect(url_for('dashboard'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Πάρε quiz_id για redirect
    cursor.execute("SELECT quiz_id FROM questions WHERE id = %s", (question_id,))
    question = cursor.fetchone()
    if not question:
        conn.close()
        return "Η ερώτηση δεν βρέθηκε."

    quiz_id = question['quiz_id']

    # Διαγραφή απαντήσεων
    cursor.execute("DELETE FROM answers WHERE question_id = %s", (question_id,))
    # Διαγραφή ερώτησης
    cursor.execute("DELETE FROM questions WHERE id = %s", (question_id,))
    conn.commit()
    conn.close()

    return redirect(url_for('quiz_questions', quiz_id=quiz_id))

@app.route('/teacher/quiz/<int:quiz_id>/export_csv')
def export_quiz_csv(quiz_id):
    if session.get('role') != 'teacher':
        return redirect(url_for('dashboard'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Quiz title
    cursor.execute("SELECT title FROM quizzes WHERE id = %s", (quiz_id,))
    quiz = cursor.fetchone()
    if not quiz:
        conn.close()
```

```
        return "Το quiz δεν βρέθηκε."

    # Submissions
    cursor.execute("""
        SELECT ss.id AS submission_id, u.name, u.student_number, ss.submission_time
        FROM student_submissions ss
        JOIN users u ON u.id = ss.user_id
        WHERE ss.quiz_id = %s AND ss.is_submitted = 1
    """, (quiz_id,))
    submissions = cursor.fetchall()

    # Δημιουργία αρχείου CSV σε μνήμη
    output = io.StringIO()
    writer = csv.writer(output)
    writer.writerow(['Ονοματεπώνυμο', 'ΑΜ', 'Ημερομηνία', 'Σκορ', 'Ποσοστό'])

    for s in submissions:
        cursor.execute("""
            SELECT sa.answer_id, a.is_correct
            FROM student_answers sa
            JOIN answers a ON a.id = sa.answer_id
            WHERE sa.submission_id = %s
        """, (s['submission_id'],))
        answers = cursor.fetchall()
        total = len(answers)
        correct = sum(1 for a in answers if a['is_correct'] == 1)
        percent = round((correct / total) * 100, 1) if total else 0

        writer.writerow([
            s['name'],
            s['student_number'],
            s['submission_time'].strftime('%d-%m-%Y %H:%M'),
            f"{correct}/{total}",
            f"{percent}%"
        ])

    conn.close()

    response = make_response(output.getvalue())
    response.headers["Content-Disposition"] = f"attachment; filename=quiz_{quiz_id}_results.csv"
    response.headers["Content-Type"] = "text/csv"
    return response

@app.route('/student/my_submissions')
def my_submissions():
    if session.get('role') != 'student':
        return redirect(url_for('dashboard'))

    user_id = session['user_id']
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Πάρε όλα τα quizzes που έχει υποβάλει
    cursor.execute("""
        SELECT q.id AS quiz_id, q.title, s.submission_time
        FROM student_submissions s
        JOIN quizzes q ON q.id = s.quiz_id
        WHERE s.user_id = %s AND s.is_submitted = 1
```

```
        ORDER BY s.submission_time DESC
    """, (user_id,))
    quizzes = cursor.fetchall()
    conn.close()

    return render_template('student_submissions.html', quizzes=quizzes)

@app.route('/student/submission/<int:quiz_id>')
def view_submission(quiz_id):
    if session.get('role') != 'student':
        return redirect(url_for('dashboard'))

    user_id = session['user_id']
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Πάρε submission ID
    cursor.execute("""
        SELECT id FROM student_submissions
        WHERE quiz_id = %s AND user_id = %s AND is_submitted = 1
    """, (quiz_id, user_id))
    sub = cursor.fetchone()
    if not sub:
        conn.close()
        return "Δεν βρέθηκε υποβολή."

    submission_id = sub['id']

    # Πάρε ερωτήσεις και απαντήσεις
    cursor.execute("""
        SELECT q.question_text, q.image_path AS qimg,
               a.answer_text AS chosen_text, a.image_path AS chosen_img,
               a.is_correct
        FROM student_answers sa
        JOIN questions q ON sa.question_id = q.id
        JOIN answers a ON sa.answer_id = a.id
        WHERE sa.submission_id = %s
    """, (submission_id,))
    answers = cursor.fetchall()
    conn.close()

    return render_template('view_submission.html', answers=answers)


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