ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Τίτλος Πτυχιακής Εργασίας	Έξυπνο λογισμικό διαχείρισης μάθησης σε φροντιστηριακά μαθήματα Personalized learning management systems for supportive courses
Ονοματεπώνυμο Φοιτητή	Αλεξάντερ Γκινετσι
Πατρώνυμο	Γιώργο
Αριθμός Μητρώου	Π18028
Επιβλέπων	Αναπληρωτής Καθηγητής Ε. Σακκόπουλος

Ημερομηνία Παράδοσης Σεπτέμβριος 2024

Copyright ©

Επιτελική Σύνοψη

Η παρούσα επιστημονική εργασία , πραγματεύεται την υλοποίηση μιας διαδικτυακής εφαρμογής η οποία έχει ως στόχο την ψηφιακή αλληλεπίδραση μεταξύ μαθητών , καθηγητών και γραμματείας αλλά και τον συγκεντρωτισμό των ενεργειών όλων των χρηστών σε μια εφαρμογή (ενημέρωση βαθμολογίας, ανέβασμα αρχείων , δημιουργία τεστ, κλείσιμο ραντεβού). Υπάρχουν διάφορες εφαρμογές , οπού οι χρήστες μπορούν παραδείγματος χαρη να δουν την βαθμολογία τους (Students Unipi, UniStudents), να λάβουν σημειώσεις και να ανεβάζουν εργασίες (Gunet2 , Thales), εν τούτης δεν τους δίνεται μέχρι στιγμής η δυνατότητα να πραγματοποιήσουν αυτές τις ενέργειες σε εάν σύστημα , και επίσης να εξεταστούν στην ιδιά πλατφόρμα.

Η υλοποίηση της συγκεκριμένης εφαρμογής πραγματοποιήθηκε με εάν από τα πιο σύγχρονα αλλά και εάν από τα πιο διαδεδομένα MVC Frameworks , αυτό του .NET Core 8.0 (C#). Το συγκεκριμένο παρέχει πληθώρα δυνατοτήτων αλλά ένα από τα πιο οργανωμένα περιβάλλοντά ανάπτυξης , αφού παρέχει αρκετά εργαλεία αλλά και πλούσιο Documentation. Στην ευκολότερη κατανόηση και απλοποίηση των απαιτήσεων που σχετίζονται με τη δημιουργία του κατείχαν κυρίαρχο ρόλο τόσο ορισμένα διαγράμματα που υλοποιήθηκαν αναφορικά με τη σχεδίαση της εφαρμογής όσο και ο προσδιορισμός των οντοτήτων της βάσης δεδομένων που την αποτελούν, καθώς και των μεταξύ τους σχέσεων.

Τέλος , η διαδικτυακή αυτή πλατφόρμα τηλεκπαίδευσης που θα αναλυθεί εκτενώς στο παρόν έγγραφο προσφέρει αρκετές δυνατότητες και λειτουργίες σε χρήστες που αφορούν τόσο διαχειριστές και καθηγητές όσο και φοιτητές του Πανεπιστημίου Πειραιώς.

Ευχαριστίες

Θα ήθελα να εκφράσω τις ευχαριστίες μου στον αναπληρωτή καθηγητή κ. Ευάγγελο Σακκόπουλο για τη δυνατότητα που μου έδωσε να πραγματοποιήσω την πτυχιακή μου εργασία, την υποστήριξη που μου παρείχε καθώς και την εμπιστοσύνη που μου έδειξε κατά την υλοποίησή της. Τέλος, θα ήθελα να ευχαριστήσω όλους τους καθηγητές του Πανεπιστημίου Πειραιώς για την πληθώρα των πολύτιμων γνώσεων που μου προσέφεραν όλα αυτά τα χρόνια της φοίτησής

Σεπτέμβριος 2024

Αλεξάντερ Γκινέτσι

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Κεφάλαιο 1ο

1 Εισαγωγή

1.1 Περιγραφή του υπό μελέτη προβλήματος.

Η διαδικτυακή πλατφόρμα Student Management System δημιουργήθηκε με σκοπό να υποστηρίξει τον ψηφιακό μετασχηματισμό της μαθησιακής διαδικασίας και να διευκολύνει τη δημιουργία και διεξαγωγή εξετάσεων με σύγχρονα εργαλεία που προσφέρει στους χρήστες της. Από την περίοδο του COVID-19 και έπειτα, υπήρξε μια ραγδαία ανάπτυξη στην ψηφιοποίηση της εκπαιδευτικής δραστηριότητας, με αποτέλεσμα να ενισχυθεί η ανάγκη για συμπλήρωση της δια ζώσης διδασκαλίας από ασύγχρονες διαδικτυακές πλατφόρμες τηλεκπαίδευσης.Επιπλέον, παρατηρήθηκαν δυσκολίες στη διεξαγωγή απομακρυσμένων εξετάσεων, καθώς πολλοί φοιτητές είτε καθυστερούσαν να συνδεθούν είτε αδυνατούσαν να μπουν έγκαιρα σε εξωτερικές πλατφόρμες, με αποτέλεσμα να μην μπορούν να εξεταστούν. Η πλατφόρμα Student Management System αντιμετωπίζει αυτό το πρόβλημα παρέχοντας τη δυνατότητα στους καθηγητές να δημιουργούν και να διεξάγουν εξετάσεις εντός της ίδιας πλατφόρμας. Με αυτόν τον τρόπο επιτυγχάνεται η ομοιομορφία στη διαδικασία εξέτασης και αποφεύγονται οι ασυμφωνίες μεταξύ διαφορετικών εξωτερικών πλατφορμών. Παράλληλα, οι μαθητές έχουν πρόσβαση σε πραγματικό χρόνο στα στατιστικά τους, στις βαθμολογίες τους, στις σημειώσεις των μαθημάτων, στις ανακοινώσεις, καθώς και στη δυνατότητα να κλείσουν ραντεβού με καθηγητές ή τη γραμματεία, όλα σε μία ενιαία πλατφόρμα

1.2 Σκοπός και στόχοι της εργασίας

Ο σκοπός της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη μιας πλατφόρμας που ανταποκρίνεται πλήρως στις σύγχρονες απαιτήσεις της τηλεκπαίδευσης, συγκεντρώνοντας πολλές από τις λειτουργίες της εκπαιδευτικής διαδικασίας σε ένα ενιαίο σύστημα. Αυτή η προσέγγιση αποσκοπεί στη διευκόλυνση τόσο των καθηγητών όσο και των φοιτητών, παρέχοντας λύσεις σε υπάρχοντα προβλήματα της τηλεκπαίδευσης. Το κεντρικό αντικείμενο της παρούσας εργασίας είναι η ανάπτυξη της εφαρμογής Student Management System, η οποία δημιουργήθηκε για να αντιμετωπίσει τα ζητήματα που έχουν εντοπιστεί. Η εφαρμογή Student Management System στοχεύει να προσφέρει λειτουργικότητες και δυνατότητες που θα την καταστήσουν ένα ολοκληρωμένο σύστημα τηλεκπαίδευσης (e-class), προσαρμοσμένο στις σύγχρονες ανάγκες εξατομικευμένης τηλεκπαίδευσης. Η πλατφόρμα παρέχει πρόσβαση σε γραμματεία, καθηγητές και φοιτητές, επιτρέποντας τη διαχείριση πολλών πτυχών της εκπαιδευτικής διαδικασίας. Η γραμματεία έχει τη δυνατότητα να εγγράφει νέους χρήστες (καθηγητές και φοιτητές) στην εφαρμογή, να διαχειρίζεται τους υπάρχοντες χρήστες, να δημιουργεί

κατηγορίες μαθημάτων (Lesson Groups), να αναθέτει μαθήματα σε καθηγητές και φοιτητές, να επικυρώνει βαθμολογίες, να έχει πρόσβαση στα στατιστικά κάθε φοιτητή και να διαχειρίζεται αιτήματα για ραντεβού . Οι καθηγητές μπορούν να ανεβάζουν σημειώσεις για τα μαθήματα που διδάσκουν, να διαμορφώνουν τις εξετάσεις, να αναρτούν ανακοινώσεις και να διαχειρίζονται αιτήματα για ραντεβού από τους φοιτητές. Οι φοιτητές έχουν τη δυνατότητα να εξετάζονται στα μαθήματα στα οποία είναι εγγεγραμμένοι, να βλέπουν τα στατιστικά τους, και να υποβάλλουν αιτήματα για ραντεβού με καθηγητές και τη γραμματεία.

1.3 Παραδοτέα της εργασίας

- Ο κώδικας της εφαρμογής Student Management System.
- Βάση δεδομένων Student Management System.
- Βίντεο της εφαρμογής Student Management System.

1.4 Περιγραφή των κεφαλαίων του παρόντος επιστημονικού εγγράφου

1) Στον κεφάλαιο 1 , γίνεται εισαγωγή στο παρόν επιστημονικό άρθρο
2) Στο κεφάλαιο 2 γίνεται παρουσίαση σχετικών λειτουργιών και συγκριτική παρουσίασή τους. Πιο αναλυτικά γίνεται σύγκριση της εφαρμογής System Management System με τις εξής πλατφόρμες ασύγχρονης τηλεκπαίδευσης:
   - eClass "Αρίσταρχος"
   - *mycourses .ntua.gr*
   - Thales Unipi
   - sis-portal.unipi.gr
3) Στο κεφάλαιο 3, θα εστιάσουμε στο Διάγραμμα Περίπτωσης Χρήσης, που παρουσιάζει τις περιπτώσεις χρήσης και τους διαφορετικούς τύπους χρηστών. Η UML περιλαμβάνει διάφορους τύπους διαγραμμάτων, με το Διάγραμμα Περίπτωσης Χρήσης να απεικονίζει τις αλληλεπιδράσεις χρηστών με το σύστημα.
4) Στο κεφάλαιο 4, θα δούμε πώς υλοποιήθηκε η πλατφόρμα Student Management Studio, χρησιμοποιώντας σύγχρονα εργαλεία όπως το Visual Studio, η γλώσσα C# και το Entity Framework Core. Θα αναλυθούν επίσης τα τεχνολογικά εργαλεία που συνέβαλαν στην επιτυχή ανάπτυξη της πλατφόρμας.
5) Στο κεφάλαιο 5 , περιγράφεται η εμπειρία χρήστη των χρηστών :
   - Admin
   - Professor
   - Student

6) Στο κεφάλαιο 5 , γίνεται μια συμπερασματική αναφορά στην εφαρμογή Student Management System.

Κεφάλαιο 2ο

2) Επισκόπηση του χώρου.

Σε αυτή την ενότητα, θα γίνει σύγκριση της πλατφόρμας Student Management System με ήδη υπάρχουσες πλατφόρμες που καλύπτουν τις μέχρι τώρα ανάγκες της ασύγχρονης τηλεκπαίδευσης. Η σύγκριση αυτή θα συμβάλει στην καλύτερη κατανόηση των αναγκών που υπάρχουν για την ασύγχρονη τηλεκπαίδευση και θα αναδείξει τα πλεονεκτήματα και τις δυνατότητες που μπορεί να προσφέρει η νέα πλατφόρμα σε σχέση με τις υφιστάμενες λύσεις.

2.1 Student Management System & Mycourses .ntua.gr

Η πλατφόρμα **mycourses.ntua.gr** αποτελεί την κύρια πλατφόρμα ασύγχρονης τηλεκπαίδευσης του Τμήματος Πολιτικών Μηχανικών του ΕΜΠ. Η πλατφόρμα προσφέρει βασικές λειτουργίες όπως ο διαχωρισμός χρηστών (καθηγητών και φοιτητών), η δυνατότητα ανάρτησης σημειώσεων από τους καθηγητές, η υποβολή εργασιών από τους φοιτητές σε μορφή .zip εντός προκαθορισμένων προθεσμιών, καθώς και η ανάρτηση και προβολή ανακοινώσεων από καθηγητές και φοιτητές, αντίστοιχα. Επιπλέον, η πλατφόρμα διαθέτει τη δυνατότητα διαχωρισμού μαθημάτων ανά κατηγορία. Ωστόσο, η πλατφόρμα παρουσιάζει σημαντικές ελλείψεις σε σχέση με τις σύγχρονες απαιτήσεις τηλεκπαίδευσης. Δεν παρέχει περιβάλλον ανάπτυξης εξετάσεων, όπως για παράδειγμα τη δημιουργία εξετάσεων πολλαπλών επιλογών, πέρα από την ανάρτηση αρχείων .zip. Επίσης, δεν υπάρχει πρόσβαση στα αναλυτικά στατιστικά ή τις βαθμολογίες των φοιτητών (προσωρινές και οριστικές), ούτε η δυνατότητα για τους φοιτητές να υποβάλλουν αιτήματα για ραντεβού με καθηγητές και τη γραμματεία.

Περισσότερα εδώ : https://mycourses.ntua.gr/

2.2 Student Management System & Thales

Η πλατφόρμα Thales αποτελεί τη νέα κύρια πλατφόρμα ασύγχρονης τηλεκπαίδευσης του τμήματος Πληροφορικής του Πανεπιστημίου Πειραιώς, αντικαθιστώντας την πλατφόρμα Gunet2. Πρόκειται για μια εκπαιδευτική πλατφόρμα που επιτρέπει στους φοιτητές να έχουν πρόσβαση στις σημειώσεις των μαθημάτων στα οποία είναι εγγεγραμμένοι, ενώ οι καθηγητές μπορούν να ανεβάζουν τις σημειώσεις για τα μαθήματά τους. Η πλατφόρμα παρέχει επίσης τη δυνατότητα εμφάνισης της ημερομηνίας εξέτασης κάθε μαθήματος σε μορφή ημερολογίου (calendar) και επιτρέπει στους φοιτητές να υποβάλλουν τις εργασίες τους σε μορφή αρχείου .zip. Ωστόσο, η πλατφόρμα Thales δεν υποστηρίζει τη δημιουργία και διεξαγωγή εξετάσεων εντός της πλατφόρμας, πέρα από την υποβολή εργασιών. Για παράδειγμα, δεν προσφέρει τη δυνατότητα δημιουργίας εξετάσεων πολλαπλών επιλογών. Επιπλέον, δεν παρέχει λειτουργίες για την υποβολή αιτημάτων ραντεβού από τους φοιτητές προς τους καθηγητές ή τη γραμματεία, κάτι που περιορίζει τη διαδραστικότητα και την ευελιξία στην εκπαιδευτική διαδικασία.

Περισσότερες πληροφορίες , εδώ : https://thales.cs.unipi.gr/

2.3 System Management System & «Αρίσταρχος»

Η πλατφόρμα "Αρίσταρχος" αποτελεί την κύρια πλατφόρμα ασύγχρονης τηλεκπαίδευσης του τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς. Είναι ένα εκπαιδευτικό εργαλείο που επιτρέπει στους φοιτητές να έχουν πρόσβαση στις σημειώσεις των μαθημάτων στα οποία είναι εγγεγραμμένοι, ενώ οι καθηγητές μπορούν να ανεβάζουν υλικό για τα μαθήματά τους. Επιπλέον, η πλατφόρμα προβάλλει την ημερομηνία των εξετάσεων κάθε μαθήματος μέσω ημερολογίου (calendar) και επιτρέπει στους φοιτητές να υποβάλλουν τις εργασίες τους σε μορφή αρχείου .zip. Παρά τα οφέλη της, η πλατφόρμα "Αρίσταρχος" δεν προσφέρει τη δυνατότητα δημιουργίας και διεξαγωγής εξετάσεων εντός του συστήματος, εκτός από την υποβολή εργασιών. Λείπουν λειτουργίες όπως η δημιουργία εξετάσεων πολλαπλών επιλογών, γεγονός που περιορίζει τις επιλογές για αξιολόγηση. Επιπλέον, δεν παρέχει τη δυνατότητα υποβολής αιτημάτων για ραντεβού με καθηγητές ή τη γραμματεία, μια λειτουργία που θα μπορούσε να βελτιώσει την επικοινωνία και τη συνεργασία στο πλαίσιο της εκπαιδευτικής διαδικασίας.

Περισσότερες πληροφορίες , εδώ : https://aristarchus.ds.unipi.gr/

2.4 System Management System & Sis-portal.unipi.gr

Η πλατφόρμα **sis-portal.unipi.gr**, πρώην **Student.unipi**, αποτελεί το βασικό σύστημα στο οποίο συνδέονται οι φοιτητές του Πανεπιστημίου Πειραιώς για να έχουν πρόσβαση στα μαθήματά τους, τα οποία ταξινομούνται ανά κατηγορία και έτος σπουδών. Μέσα από την πλατφόρμα, οι φοιτητές μπορούν να δουν τις βαθμολογίες τους, να παρακολουθήσουν τον γενικό μέσο όρο της επίδοσής τους, και να δηλώσουν τα μαθήματα που θα παρακολουθήσουν. Επιπλέον, τους παρέχεται η δυνατότητα να υποβάλουν αιτήσεις για την έκδοση εγγράφων από τη σχολή, όπως βεβαιώσεις σπουδών. Ωστόσο, η πλατφόρμα παρουσιάζει ορισμένες αδυναμίες. Οι φοιτητές πρέπει να δηλώσουν οι ίδιοι τα μαθήματά τους, τα οποία στη συνέχεια πρέπει να επικυρωθούν από τη γραμματεία, διαδικασία που μπορεί να καθυστερήσει, προκαλώντας ενδεχομένως προβλήματα εάν δεν ολοκληρωθεί έγκαιρα. Επιπλέον, η πλατφόρμα δεν υποστηρίζει τη διεξαγωγή εξετάσεων, ούτε δίνει τη δυνατότητα στους καθηγητές να δημιουργούν χώρους για εξετάσεις. Τέλος, δεν παρέχει δυνατότητα προγραμματισμού ραντεβού μεταξύ φοιτητών και καθηγητών ή της γραμματείας, γεγονός που περιορίζει την αλληλεπίδραση μεταξύ των μερών.

Περισσότερες πληροφορίες , εδώ : https://sso.unipi.gr/login?service=https%3A%2F%2Fsis-portal.unipi.gr%2Flogin%2Fcas

2.5 Πίνακας συγκρίσεων μεταξύ των πλατφορμων

Πλατφόρμα	**Διαχείριση Ραντεβού**	**Δημιουργία Εξετάσεων**	**Χώρος Ανακοινώσεων**	**Προσωπικοί Λογαριασμοί**	**Ανέβασμα Αρχείων (από μαθητές)**	**Πρόσβαση σε βαθμολογία**
mycourses .ntua.gr	✘	✘	✓	✓	✓	✘
Thales	✘	✘	✓	✓	✓	✘
«Αρίσταρχος»	✘	✘	✓	✓	✓	✘
sis-portal	✘	✘	✓	✓	✘	✓
School Management System	✓	✓	✓	✓	✓	✓

Εικόνα - πίνακας σύγκρισης εφαρμογών με την εφαρμογή Student Management System

Κεφάλαιο 3ο

3 UML διαγράμματα εφαρμογής

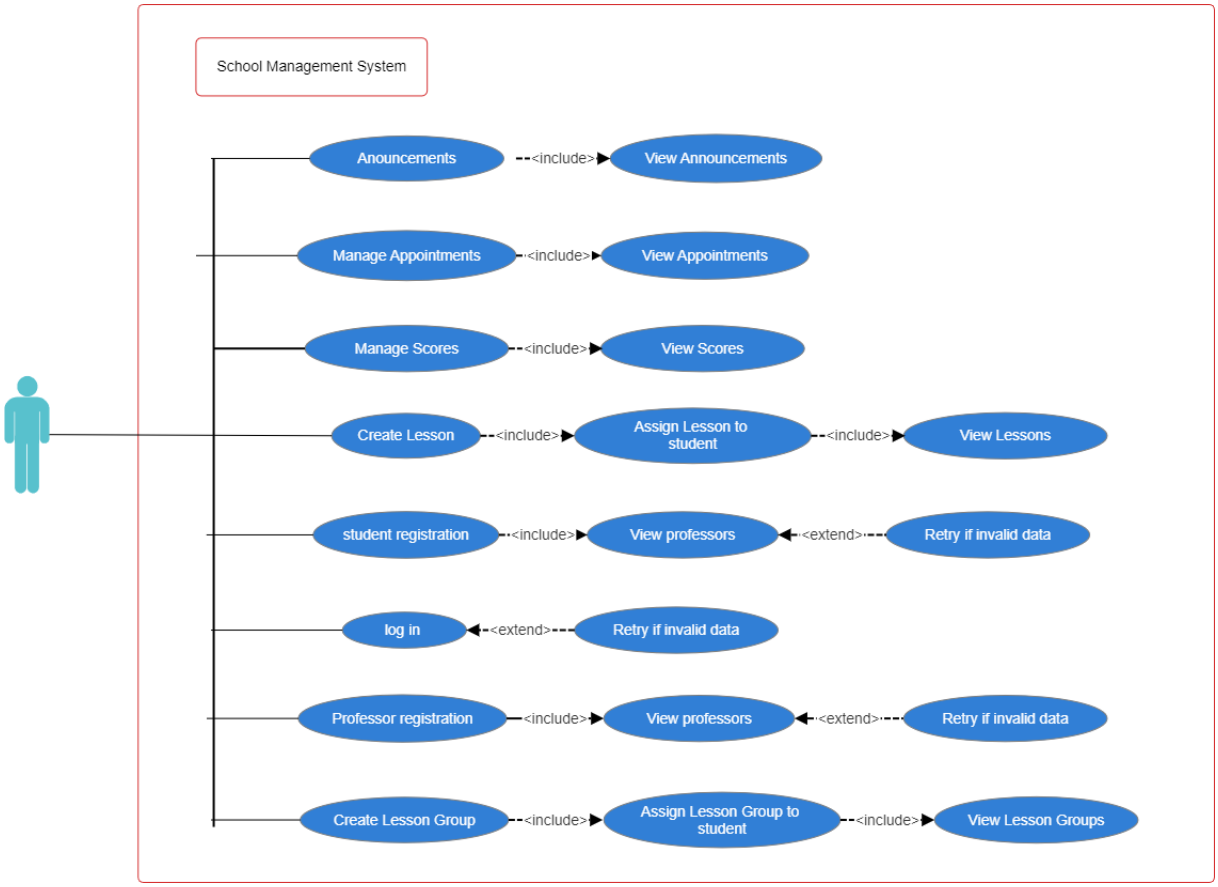
Σε αυτό το κεφάλαιο του παρόντος επιστημονικού εγγράφου, εξετάζεται η ανάλυση και ο σχεδιασμός της πλατφόρμας τηλεκπαίδευσης Student Management System, εστιάζοντας στα αρχικά στάδια υλοποίησής της, τα οποία διαδραματίζουν κρίσιμο ρόλο στην επιτυχημένη ανάπτυξη ενός λογισμικού στον τομέα της Πληροφορικής. Η σωστή θεμελίωση και σχεδίαση στα αρχικά στάδια εξασφαλίζει τη βιωσιμότητα, την επεκτασιμότητα και την αποτελεσματικότητα του τελικού προϊόντος. Για να διευκολυνθεί η διαδικασία ανάλυσης και να καταγραφούν με ακρίβεια οι λειτουργικές και μη λειτουργικές απαιτήσεις της εφαρμογής, είναι απαραίτητο να αποτυπωθεί η ροή εργασίας και οι σχέσεις μεταξύ των διαφόρων στοιχείων του συστήματος μέσω διαγραμμάτων. Τέτοιου είδους διαγράμματα, όπως τα διαγράμματα UML (Unified Modeling Language), επιτρέπουν μια λεπτομερή και τεκμηριωμένη παρουσίαση της δομής και της λειτουργικότητας του συστήματος, προσφέροντας μια σφαιρική κατανόηση του σχεδιασμού του λογισμικού τόσο στους προγραμματιστές όσο και στους ενδιαφερόμενους. Η χρήση αυτών των διαγραμμάτων είναι ουσιαστική για την επικοινωνία της πολυπλοκότητας του συστήματος και την εξασφάλιση μιας ομαλής πορείας υλοποίησης. Εξίσου σημαντικός είναι ο προσεκτικός σχεδιασμός της βάσης δεδομένων, ο οποίος αποτελεί τη ραχοκοκαλιά κάθε σύγχρονης εφαρμογής. Ένα σωστά δομημένο σύστημα βάσεων δεδομένων εξασφαλίζει την αποδοτική αποθήκευση, ανάκτηση και διαχείριση των δεδομένων, ενώ διευκολύνει τη μελλοντική ανάπτυξη και προσαρμογή της πλατφόρμας σε νέες απαιτήσεις. Επιπλέον, ένας καλός σχεδιασμός βάσεων δεδομένων συμβάλλει στη βελτίωση της απόδοσης του συστήματος και στη μείωση της πολυπλοκότητας κατά τη διάρκεια της ανάπτυξης, της συντήρησης και της διαχείρισης του λογισμικού.

3.1 Χρήση διαγραμμάτων UML

Η UML περιλαμβάνει 9 διαφορετικούς τύπους διαγραμμάτων, τα οποία χρησιμοποιούνται για την απεικόνιση διάφορων πτυχών της μοντελοποίησης. Στο παρόν επιστημονικό έγγραφο θα εστιάσουμε στο διάγραμμα UML τύπου Use Case Diagram η αλλιώς Διάγραμμα Περίπτωσης Χρήσης. Το Διάγραμμα Περίπτωσης Χρήσης, είναι μια γραφική απεικόνιση των πιθανών αλληλεπιδράσεων ενός χρήστη με το σύστημα. Ένα τέτοιο διάγραμμα παρουσιάζει διάφορες περιπτώσεις χρήσης και τους διαφορετικούς τύπους χρηστών που αλληλοεπιδρούν με το σύστημα. Συχνά συνοδεύεται και από άλλους τύπους διαγραμμάτων.

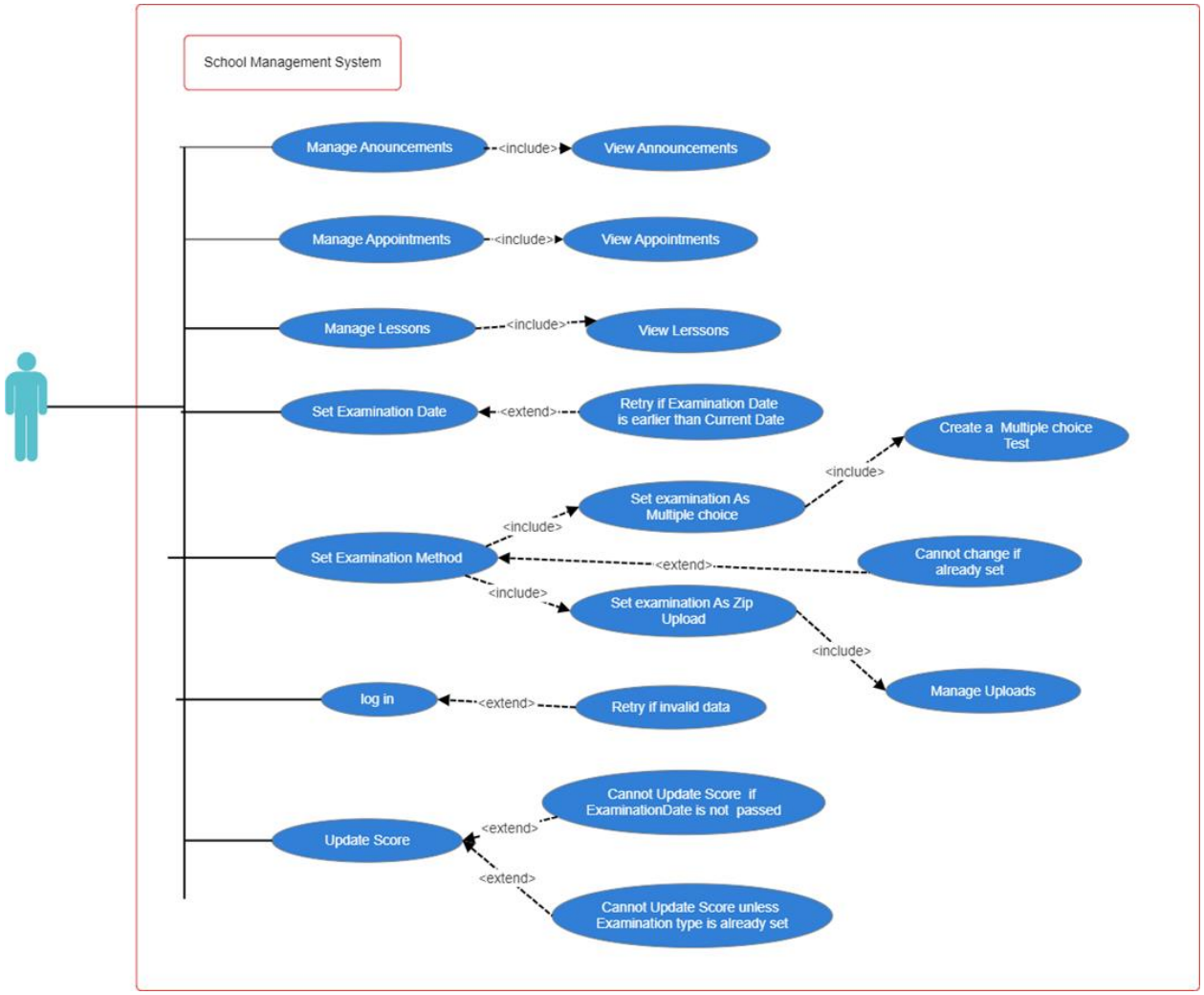
3.2 Student Management System (Use Case Diagrams)

3.2.1 Admin (Γραμματεία).



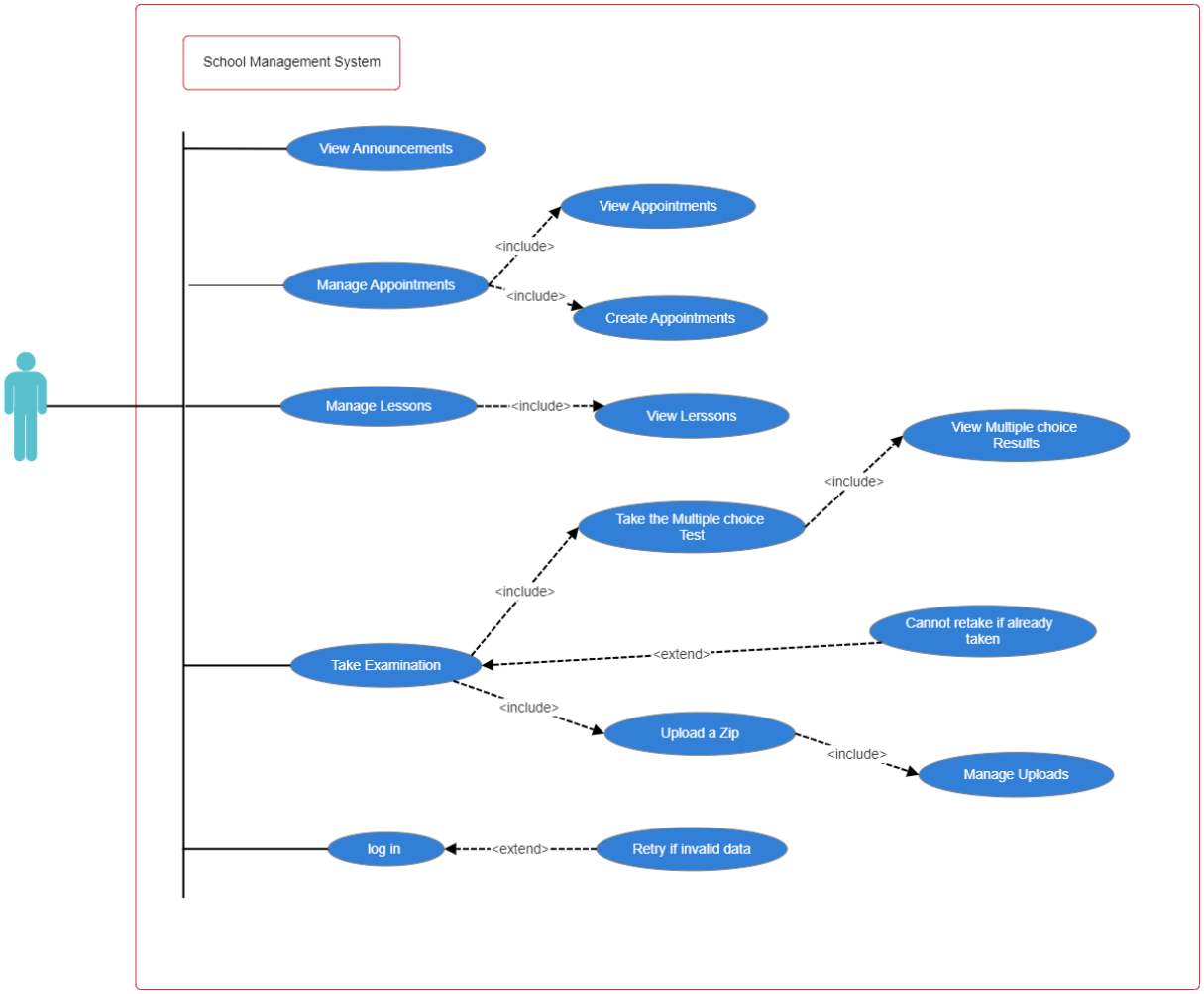
Εικόνα 1 - Διάγραμμα περιπτώσεων χρήσης admin για Student Management System

3.2.2 Professor (Καθηγητής).



Εικόνα 2 - Διάγραμμα περιπτώσεων χρήσης professor για Student Management System

3.2.3 Student (Μαθητής).



Εικόνα 3 - Διάγραμμα περιπτώσεων χρήσης student για Student Management System

Κεφάλαιο 4ο

4 Εργαλεία – Λογισμικό

Η πλατφόρμα τηλεκπαίδευσης Student Management Studio υλοποιήθηκε με τη βοήθεια και τη χρήση διαφόρων σύγχρονων εργαλείων λογισμικού, τα οποία έπαιξαν σημαντικό ρόλο στην επιτυχή ανάπτυξή της. Η δημιουργία της πραγματοποιήθηκε στο Ολοκληρωμένο Περιβάλλον Ανάπτυξης (Integrated Development Environment - IDE), το Visual Studio, που αποτελεί ένα από τα πιο διαδεδομένα εργαλεία ανάπτυξης λογισμικού. Για τον προγραμματισμό της πλατφόρμας χρησιμοποιήθηκε η γλώσσα **C#**, σε συνδυασμό με το Entity Framework Core και την πλατφόρμα .NET Core 8.0. Το Visual Studio προσφέρει ένα ισχυρό περιβάλλον ανάπτυξης, το οποίο επιτρέπει τη δημιουργία διαδικτυακών εφαρμογών, προγραμμάτων για υπολογιστές, διαδικτυακών υπηρεσιών, ιστοσελίδων. Επιπλέον, υποστηρίζει πολλαπλές γλώσσες προγραμματισμού, μία εκ των οποίων είναι η C#, με την οποία υλοποιήθηκε η συγκεκριμένη εφαρμογή.

Η γλώσσα προγραμματισμού C# είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού, αναπτυγμένη από τη Microsoft, που χρησιμοποιείται κυρίως στην πλατφόρμα .NET για την ανάπτυξη διαφόρων τύπων εφαρμογών. Οι εφαρμογές αυτές περιλαμβάνουν λογισμικό για υπολογιστές, ιστοσελίδες, υπηρεσίες cloud και παιχνίδια.

Στη συνέχεια, θα γίνει εκτενής αναφορά στα εργαλεία και τις τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής, περιγράφοντας πώς κάθε ένα από αυτά συνέβαλε στην ολοκλήρωση της πλατφόρμας Student Management System

4.1 Εργαλεία

4.1.1 Entity Framework Core (ref 21)

Το Entity Framework Core ή EF Core σε συντομία, που χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής, είναι ένα ισχυρό εργαλείο για τη διαχείριση των βάσεων δεδομένων, επιτρέποντας την απλοποιημένη διαχείριση δεδομένων με αντικειμενοστραφή τρόπο. Παρέχει τη δυνατότητα για άμεση αλληλεπίδραση με τη βάση δεδομένων χωρίς την ανάγκη γραφής πολύπλοκου SQL κώδικα, επιταχύνοντας έτσι την ανάπτυξη και διευκολύνοντας τη διαχείριση δεδομένων.

4.1.2 .NET Core 8.0 (ref 22)

Η .NET Core 8.0 είναι η πλατφόρμα ανάπτυξης της Microsoft που επιτρέπει την ανάπτυξη λογισμικού σε διαφορετικά λειτουργικά συστήματα, όπως Windows, macOS και Linux. Παρέχει υψηλή απόδοση και ευελιξία, καθιστώντας την ιδανική

επιλογή για την ανάπτυξη σύγχρονων εφαρμογών. Η έκδοση 8.0 είναι η τελευταία έκδοση που προσφέρει Long Term Support.

4.1.3 MVC Αρχιτεκτονική (ref 1)

Το Model-View-Controller (MVC) είναι ένα από τα πιο δημοφιλή πρότυπα σχεδίασης λογισμικού, ιδιαίτερα στην ανάπτυξη εφαρμογών με διεσπάρη χρήστη (user interface). Χρησιμοποιείται για την οργάνωση και τη δομή του κώδικα, κάνοντας τη διαδικασία ανάπτυξης πιο αποδοτική και ευέλικτη. Το MVC χωρίζει τη λογική μιας εφαρμογής σε τρία κύρια στοιχεία, τα οποία είναι:

- **Model** (Μοντέλο): Το Model αναπαριστά τα δεδομένα και τους κανόνες διαχείρισής τους. Είναι υπεύθυνο για την αποθήκευση, ανάκτηση και ενημέρωση των δεδομένων από τη βάση. Δεν ασχολείται με την εμφάνιση και επικοινωνεί με τον Controller για την ενημέρωση των δεδομένων. Παραδείγματα: βάσεις δεδομένων, αντικείμενα αποθήκευσης δεδομένων.

- **View** (Προβολή): Το View εμφανίζει τα δεδομένα στο χρήστη και αποτελεί τη διεπαφή χρήστη (UI). Δεν επεμβαίνει στη λογική των δεδομένων, μόνο στην εμφάνισή τους. Παραδείγματα: HTML, CSS, πίνακες δεδομένων.

- **Controller** (Ελεγκτής): Ο Controller συνδέει το Model με το View. Λαμβάνει την είσοδο από τον χρήστη, την επεξεργάζεται και αλληλεπιδρά με το Model για να διαχειριστεί τα δεδομένα και να κατευθύνει το View. Παραδείγματα: διαχείριση εισόδου από φόρμες, εντολές για αποθήκευση δεδομένων.

4.1.4 Microsoft.AspNetCore.Identity(ref 3)

Η Microsoft.AspNetCore.Identity είναι μια βιβλιοθήκη στο πλαίσιο της πλατφόρμας ASP.NET Core που παρέχει συστήματα ταυτοποίησης και διαχείρισης χρηστών. Αυτή η βιβλιοθήκη επιτρέπει στους προγραμματιστές να προσθέσουν εύκολα λειτουργίες όπως εγγραφή, σύνδεση, διαχείριση ρόλων, έλεγχο ταυτότητας και εξουσιοδότηση χρηστών στις εφαρμογές τους. Είναι ιδιαίτερα χρήσιμη για την ανάπτυξη ασφαλών εφαρμογών με δυνατότητες διαχείρισης χρηστών και δικαιωμάτων.

4.1.5 System.Security.Claims (ref 6)

Το System.Security.Claims είναι ένα namespace στην πλατφόρμα .NET που χρησιμοποιείται για τη διαχείριση ταυτοτήτων και πληροφοριών (claims) σχετικά με τους χρήστες ενός συστήματος. Τα claims είναι δηλώσεις ή πληροφορίες που περιγράφουν χαρακτηριστικά ή ιδιότητες ενός χρήστη, όπως η ταυτότητα, ο ρόλος, ή τα δικαιώματα πρόσβασης

4.1.6 System.IO.Compression (ref 4)

Το System.IO.Compression είναι ένα namespace στη πλατφόρμα .NET που παρέχει κλάσεις και μεθόδους για τη συμπίεση και αποσυμπιεστή αρχείων και δεδομένων. Χρησιμοποιείται για να χειρίζεται αρχεία ZIP και gzip, προσφέροντας λειτουργίες όπως δημιουργία, ανάγνωση και εξαγωγή συμπιεσμένων αρχείων.

4.1.7 Microsoft.EntityFrameworkCore.SqlServer (ref 2)

Το Microsoft.EntityFrameworkCore.SqlServer είναι ένα πακέτο επέκτασης (NuGet package) που επιτρέπει στο Entity Framework Core (EF Core) να συνδεθεί και να διαχειριστεί βάσεις δεδομένων SQL Server. Το Entity Framework Core είναι ένα δημοφιλές Object-Relational Mapper (ORM) στην πλατφόρμα .NET, που επιτρέπει στους προγραμματιστές να εργάζονται με βάσεις δεδομένων χρησιμοποιώντας αντικείμενα C# αντί να γράφουν απευθείας SQL. Το πακέτο Microsoft.EntityFrameworkCore.SqlServer παρέχει όλες τις απαραίτητες λειτουργίες για την αλληλεπίδραση με μια βάση δεδομένων Microsoft SQL Server

4.1.8 Microsoft.EntityFrameworkCore.Tools (ref 18)

Το Microsoft.EntityFrameworkCore.Tools είναι ένα πακέτο επέκτασης (NuGet package) για το Entity Framework Core (EF Core) που προσθέτει εργαλεία για την ανάπτυξη και τη διαχείριση της βάσης δεδομένων κατά τη διάρκεια της ανάπτυξης του λογισμικού. Αυτά τα εργαλεία διευκολύνουν την εκτέλεση εργασιών, όπως η δημιουργία και διαχείριση των migrations(ref 15), η δημιουργία του DbContext, και η αναπαραγωγή αλλαγών στη βάση δεδομένων μέσω του Command Line Interface (CLI) ή του Package Manager Console στο Visual Studio.

4.2 Ανάλυση σημείων κώδικα

4.2.1 Registration

Το .NET, μέσω της βιβλιοθήκης Identity, παρέχει ένα πλήρες σύστημα αυθεντικοποίησης (Authentication System), το οποίο περιλαμβάνει λειτουργίες όπως Login, Logout, Registration, Forgot Password, και Reset Password. Σε αυτή την ενότητα θα εστιάσουμε στη διαδικασία της εγγραφής χρήστη (Registration).Αρχικά, πραγματοποιείται η επικύρωση (validation) των δεδομένων που υποβάλλει ο χρήστης. Όπως φαίνεται στην παρακάτω εικόνα, υπάρχουν διάφοροι περιορισμοί που πρέπει να πληρούνται. Συγκεκριμένα:

- Το όνομα (**FirstName**) και το επώνυμο(**LastName**) είναι υποχρεωτικά πεδία. Πρέπει να είναι τύπου string, αρχικοποιούνται ως κενές συμβολοσειρές και όχι ως null.

- Το **Email** είναι επίσης υποχρεωτικό. Πρέπει να είναι τύπου string, να ακολουθεί τη μορφή έγκυρης διεύθυνσης email (π.χ., test@gmail.com), να μην ξεπερνά τους 255 χαρακτήρες, και να είναι μοναδικό στον πίνακα χρηστών (users table), δηλαδή να μην υπάρχει άλλος χρήστης με την ίδια διεύθυνση email.

- Για το ΑΦΜ (**AFM**), υπάρχουν οι εξής περιορισμοί: πρέπει να είναι τύπου ακέραιος αριθμός (int) και να έχει ακριβώς 9 χαρακτήρες.

- Ο αριθμός τηλεφώνου (**Phone**) πρέπει να είναι τύπου int και να έχει ακριβώς 10 χαρακτήρες.

- Τέλος, για τον κωδικό πρόσβασης (**Password**) ισχύουν οι εξής κανόνες: είναι υποχρεωτικό πεδίο, πρέπει να είναι τύπου string, να περιέχει τουλάχιστον 6 χαρακτήρες, και να γίνεται επιβεβαίωση του κωδικού, δηλαδή ο χρήστης πρέπει να τον εισάγει σε δύο πεδία και οι τιμές να ταιριάζουν.

```
public class InputModel
{

    [Required]
    [EmailAddress]
    [Display(Name = "Email")]
    7 references
    public string Email { get; set; }

    [Required]
    [Display(Name = "FirstName")]
    4 references
    public string FirstName { get; set; } = string.Empty;

    [Required]
    4 references
    public string LastName { get; set; } = string.Empty;

    [Required]
    4 references
    public string Address { get; set; } = string.Empty;

    [Required]
    [RegularExpression("^[0-9]{9}$", ErrorMessage = "AFM must be  9 digits.")]
    4 references
    public string Afm { get; set; } = string.Empty;

    [Required]
    [Phone]
    [RegularExpression("^[0-9]{10}$", ErrorMessage = "Phone must be 10 digits.")]

    [Display(Name = "Phone")]
    4 references
    public string Phone { get; set; } = string.Empty;
```

Εικόνα – Το model InputModel

```
    [Required]
    [Display(Name = "Status")]
    4 references
    public string Status { get; set; } = string.Empty;


    [Required]
    [StringLength(100, ErrorMessage = "The {0} must be at least {2} and at max {1} characters long.", MinimumLength = 6)]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    4 references
    public string Password { get; set; }


    [DataType(DataType.Password)]
    [Display(Name = "Confirm password")]
    [Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
    3 references
    public string ConfirmPassword { get; set; }
}
```

Εικόνα – Το model InputModel (συνέχεια)

Προκειμένου να δούμε την σελίδα του registration , καλούμε την μέθοδο **OnGetAsync** (GET Request):

```
public async Task OnGetAsync(string returnUrl = null)
{
    ReturnUrl = returnUrl;
    ExternalLogins = (await _signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
}
```

Εικόνα – η μέθοδος OnGetSync(string returnUrl = null)

Προκειμένου να καταχωρηθεί η νέα εγγραφή χρήστη , καλούμε την μέθοδο **OnPostRequest** (POST Request)

```
0 references
public async Task<IActionResult> OnPostAsync(string returnUrl = null)
{
    returnUrl ??= Url.Content("~/");
    ExternalLogins = (await _signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
    if (ModelState.IsValid)
    {
        var user = CreateUser();
        user.RoleId = _context.Roles.Where(x => x.Name == "Professor").Select(x => x.Id).SingleOrDefault();
        user.FirstName = Input.FirstName;
        user.LastName = Input.LastName;
        user.Address = Input.Address;
        user.Phone = Input.Phone;
        user.Afm = Input.Afm;
        user.Status = Input.Status;
        await _userStore.SetUserNameAsync(user, Input.Email, CancellationToken.None);
        await _emailStore.SetEmailAsync(user, Input.Email, CancellationToken.None);
        var result = await _userManager.CreateAsync(user, Input.Password);

        if (result.Succeeded)
        {
            await _userManager.AddToRoleAsync(user, "Professor");
            _logger.LogInformation("User created a new account with password.");

            var userId = await _userManager.GetUserIdAsync(user);
            var code = await _userManager.GenerateEmailConfirmationTokenAsync(user);
            code = WebEncoders.Base64UrlEncode(Encoding.UTF8.GetBytes(code));
            var callbackUrl = Url.Page(
                "/Account/ConfirmEmail",
                pageHandler: null,
                values: new { area = "Identity", userId = userId, code = code, returnUrl = returnUrl },
                protocol: Request.Scheme);

            await _emailSender.SendEmailAsync(Input.Email, "Confirm your email",
                $"Please confirm your account by <a href='{HtmlEncoder.Default.Encode(callbackUrl)}'>clicking here</a>.");

            if (_userManager.Options.SignIn.RequireConfirmedAccount)
```

Με την μέθοδο CreateUser , δημιουργούμε ένα νέο instance της κλάσης ApplicationUser , που αντιπροσωπεύει την νέα εγγραφή στο πίνακα ApplicationUsers , αρά έναν νέο χρήστη.

```
1 reference
private ApplicationUser CreateUser()
{
    try
    {
        return Activator.CreateInstance<ApplicationUser>();
    }
    catch
    {
        throw new InvalidOperationException($"Can't create an instance of '{nameof(ApplicationUser)}'. " +
            $"Ensure that '{nameof(ApplicationUser)}' is not an abstract class and has a parameterless constructor, or alternati
            $"override the register page in /Areas/Identity/Pages/Account/Register.cshtml");
    }
}
```

Εικόνα – Η μέθοδος CreateUser()

4.2.2 Entity Relationships

Εδώ θα γίνει εκτενής αναφορά σε εάν από τα πιο σημαντικά κομμάτια του .NET Core , οι σχέσεις μεταξύ των μοντέλων (ref 7). Παραδείγματος χάρη , μια κατηγορία μαθήματος (Lesson Group) περιέχει πολλά μαθήματα (Lessons) . Χρησιμοποιώντας το Fluent API configuration του Entity Framework Core, μπορούμε πολύ ευκολά να δημιουργήσουμε την παρακάτω σχέση.

```
builder.Entity<Lesson>()
    .HasOne(l => l.LessonGroup)
    .WithMany(lg => lg.Lessons)
    .HasForeignKey(l => l.LessonGroupId)
    ;
```

Εικόνα – Η σχέση μεταξύ Lesson & LessonGroup

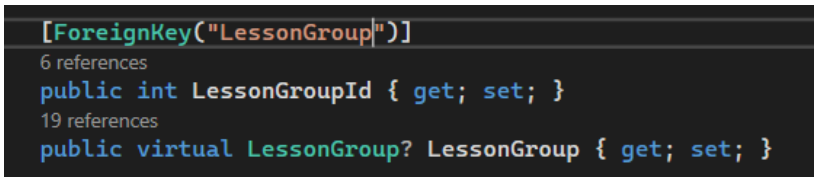
- Εδώ δηλώνουμε ότι το Entity Lesson έχει αντιστοιχία ένα-προς-πολλά με το Entity LessonGroup και ότι συνδέονται με το Foreign key LessonGroupId που ανήκει στο Lesson

- Παρακάτω δηλώνουμε τα Entities στην κλάση ApplicationDbContaxt.cs

```
5 references
public DbSet<SchoolManagementSystem.Areas.Identity.Data.LessonGroup> LessonGroups { get; set; } = default;
32 references
public DbSet<SchoolManagementSystem.Areas.Identity.Data.Lesson> Lessons { get; set; } = default;
6 references
```

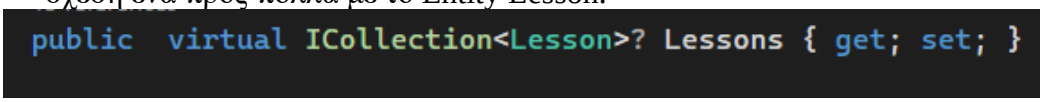
Εικόνα – Τα Entities Lesson & LessonGroup στο ApplicatioDbContext

- Στην συνέχεια δημιουργούμε σύνδεση μεταξύ των Entities Lesson & LessonGroup , δημιουργώντας ένα εξωτερικό κλειδί (Foreign Key) , στο Lesson ονομαι LessonGroupId .



Εικόνα – Το foreign key LessonGroupId

- Έπειτα πηγαίνουμε στο Entity LessonGroup για δημιουργούμε μια virtual σχέση ένα-προς-πολλά με το Entity Lesson.



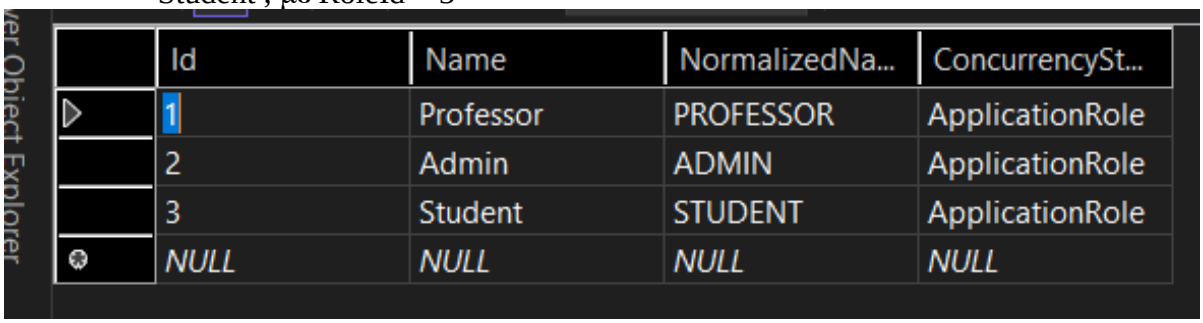
Εικόνα – Η virtual σχέση

4.2.3 Restricted User Access.

Εδώ θα γίνει αναφορά στην δυνατότητα που μας προσφέρει η βιβλιοικη Intendity προκειμένου να συγκεκραμένες προσβάσεις , σε ορισμένους χρήστες για συγκεκριμένες ενέργειες.

Στον πινακα ApplicationRoles , έχουν προ σεταριστει ο παρακάτω ρολόι:

- Professor , με RoleId = 1
- Admin, με RoleId = 2
- Student , με RoleId = 3



	Id	Name	NormalizedNa...	ConcurrencySt...
▷	1	Professor	PROFESSOR	ApplicationRole
	2	Admin	ADMIN	ApplicationRole
	3	Student	STUDENT	ApplicationRole
	NULL	*NULL*	*NULL*	*NULL*

Εικόνα – Ο πίνακας ApplicationRole

Στην συνέχεια, δηλώνουμε στο Entity **ApplicationRole** , μια virtual σχεση με το Entity **ApplicationUser :**

Εικόνα – Η virtual σχέση

Έπειτα δηλώνουμε μια σχέση ένα-προς-πολλά μεταξύ των Entities **ApplicationRole** και **ApplicationRole** (σε ένα χρήστη αντιστοιχεί ένας ρόλος και πολλοί χρήστες μπορούν έχουν τον ίδιο ρολό) στην κλάση **ApplicationDbContext.**

```
builder.Entity<ApplicationUser>()
    .HasOne(x => x.Role)
    .WithMany(x => x.Users)
    .OnDelete(DeleteBehavior.NoAction);
```

Εικόνα – Η σχέση μεταξύ ApplicationUser

Με αυτόν τον τρόπο έχουμε τους τρεις ρόλους στην εφαρμογή.
Με βάση αυτούς τους ρόλους επιτυγχάνεται η περιορισμένη πρόσβαση σε σημεία της εφαρμογής. Μερικά παραδείγματα είναι τα παρακάτω. :

Περιορισμένη πρόσβαση του χρήστη Student στην σελίδα **StudentScoreOverview :**

```
@if (User.IsInRole("Student")) {
    <a class="nav-link text-light" asp-controller="Student" asp-action="StudentScoreOverview">Overview</a>

}
```

Εικόνα – Περιορισμένη πρόσβαση (1)

Περιορισμένη πρόσβαση του Admin στις ενεργείς του **AdminController :**

```
[Authorize(Roles = "Admin")]
1 reference
public class AdminController : Controller
{
    private readonly UserManager<ApplicationU
```

Εικόνα – Περιορισμένη πρόσβαση (2)

4.3 Lesson Group and Lesson Assigment

Μια από τις βασικότερες λειτουργίες της εφαρμογής είναι η δυνατότητα της γραμματείας να αναθέτει στους καθηγητές κατηγορίες μαθήματων (Lesson Groups) η μαθήματα (Lessons) στους μαθητές.

4.3.1 Lesson Group Assignment

Ο χρήστης Admin πατώντας το action CreateLessonGroup() , μπορεί να δημιουργήσει ένα καινούργιο Lesson Group (Κατηγορία Μαθήματος) και μέσα από ένα viewbag να επιλέξει ένα από τους ήδη εγγεγραμμένους χρήστες Καθηγητές προκειμένου να γίνει assign στο συγκεκριμένο Lesson Group.

Η GET μέθοδος **CreateLessonGroup()**, επιστρέφει μια φόρμα συμπλήρωσης στοιχείων με τα πεδία : Όνομα κατηγορίας και ένα viewbag από καθηγητές

```
[HttpGet]
0 references
public IActionResult CreateLessonGroup()
{
    var professors = _context.Users.Where(x => x.RoleId == "1").ToList();
    ViewBag.Professors = professors.Select(p => new SelectListItem { Value = p.Id, Text = p.UserName }).ToList();
    return View();
}
```

Εικόνα – η GET μέθοδος CreateLessonGroup()

Η POST μέθοδος **CreateLessonGroup(string name, string professorId)**, δέχεται ως ορίσματα ένα string name , οπού είναι το όνομα της κατηγορίας και το professorId οπού είναι το UserId του χρήστη που επιλέχθηκε από το παραπάνω viewbag, και δημιουργεί ένα καινούργια instance της κλάσης LessonGroup και το αποθήκευση στην βάση, έπειτα μας γυρνάει πίσω στην σελίδα Index.

```
[HttpPost]
0 references
public IActionResult CreateLessonGroup(string name, string professorId)
{
    var lessonGroup = new LessonGroup { Name = name, UserId = professorId };
    _context.LessonGroups.Add(lessonGroup);
    _context.SaveChanges();
    return RedirectToAction("Index");
}
```

Εικόνα – η POST μέθοδος CreateLessonGroup()

4.3.2 Lesson Assignment

Ο χρήστης Admin πατώντας το action **AssingLessonToStudent()**, μπορεί να επιλέξει ένα από τα ήδη επιλεγμένα Lesson Groups , έπειτα με βάση την επιλογή αυτή του εμφανίζονται τα μαθήματα που ανήκουν στο συγκεκριμένο Lesson Group και τέλος εμφανίζεται ένα viewbag με τους χρήστες μαθητές.

Η GET μέθοδος **AssignLessonToStudent()** μας επιστρέφει ,μια φόρμα συμπλήρωσης με τα πεδία : Lesson Group (viewbag από υπάρχοντες κατηγορίες) , Lesson(viewbag από μαθήματα) , Name (viewbag από χρήστες μαθητές)

```
[HttpGet]
0 references
public IActionResult AssignLessonToStudent()
{
    var lessonGroups = _context.LessonGroups.ToList();
    var students = _context.Users.Where(u => u.RoleId == "3").ToList(); /

    ViewBag.LessonGroups = new SelectList(lessonGroups, "Id", "Name");
    ViewBag.Lessons = new SelectList(new List<Lesson>(), "Id", "Title");
    ViewBag.Students = new SelectList(students, "Id", "UserName");

    return View();
}
```

Εικόνα – η GET μέθοδος AssignLessonToStudent()

Η POST μέθοδος **AssignLessonToStudent()** δέχεται δύο παραμέτρους, το lessonId (ID μαθήματος) και το selectedStudentId (ID φοιτητή). Ελέγχει αν τα δεδομένα είναι έγκυρα, και αν δεν είναι, επιστρέφει ένα σφάλμα. Στη συνέχεια, ανακτά το μάθημα και τον φοιτητή από τη βάση δεδομένων. Αν δεν βρεθούν, επιστρέφει μηνύματα λάθους. Αν ο φοιτητής δεν είναι ήδη εγγεγραμμένος στο μάθημα, προστίθεται, αποθηκεύονται οι αλλαγές και ο χρήστης ανακατευθύνεται στη σελίδα με τα μαθήματα. Τέλος , εφόσον τοποθετήθηκαν έγκυρα στοιχεία στην παραπάνω φόρμα , γίνεται assign ένα μάθημα σε ένα μαθητή.

```
[HttpPost]
0 references
public IActionResult AssignLessonToStudent(int lessonId, string selectedStudentId)
{
    if (string.IsNullOrEmpty(selectedStudentId) || lessonId == 0)
    {
        ModelState.AddModelError("", "You must select both a lesson and a student.");
        return RedirectToAction("AssignLessonToStudent");
    }

    var lesson = _context.Lessons.Include(l => l.Users).SingleOrDefault(l => l.Id == lessonId);
    if (lesson == null) return NotFound("Lesson not found");

    var student = _context.Users.SingleOrDefault(u => u.Id == selectedStudentId);
    if (student == null) return NotFound("Student not found");

    if (!lesson.Users.Any(u => u.Id == student.Id))
    {
        lesson.Users.Add(student);
        _context.SaveChanges();
    }

    return RedirectToAction("IndexLesson");
}
```

Εικόνα – η Postμέθοδος AssignLessonToStudent()

4.4 Appointment Creation & Management

Μια από της βασικότερες λειτουργίες αυτής της εφαρμογής είναι η δημιουργία αιτημάτων ραντεβού από τους χρήστες μαθητές και η επικύρωση των αιτημάτων από τους χρήστες καθηγητές και μαθητές.

4.4.1 Appointment Creation

Η GET μέθοδος Create στον AppointmentsController, μας επιτρέπει να έχουμε πρόσβαση στην φόρμα συμπλήρωσης (Create.cshtml View) και γεμίζει ένα viewbag φιλτραρισμένους χρήστες , με βάση το RoleId τους (στο παράδειγμα με βάση το RoleId=2 , Admin)

```
// GET: Appointments/Create
[Authorize(Roles = "Student")]
0 references
public IActionResult Create()
{
    // Fetch users with RoleId = 2 (e.g., Teachers)
    var usersWithRoleId2 = _context.Users
        .Where(u => _context.UserRoles.Any(ur => ur.UserId == u.Id && ur.RoleId == "2"))
        .ToList();

    // Pass the list to the view using ViewBag
    ViewBag.UsersWithRoleId2 = new SelectList(usersWithRoleId2, "Id", "Email");

    return View();
}
```

Εικόνα – η GET μέθοδος Create()

Η POST μέθοδος Create , επιτρέπει να κατοχυρώσουμε το αίτημα ραντεβού προσθέτοντας ένα επιπλέον πεδίο , την σημερινή ημερομηνία . Σε περίπτωση που αποτύχει η κατοχύρωση του αιτήματος , μας επιστέφει ξανά την φόρμα κενή αλλά ξαναγεμίζει το viewbag

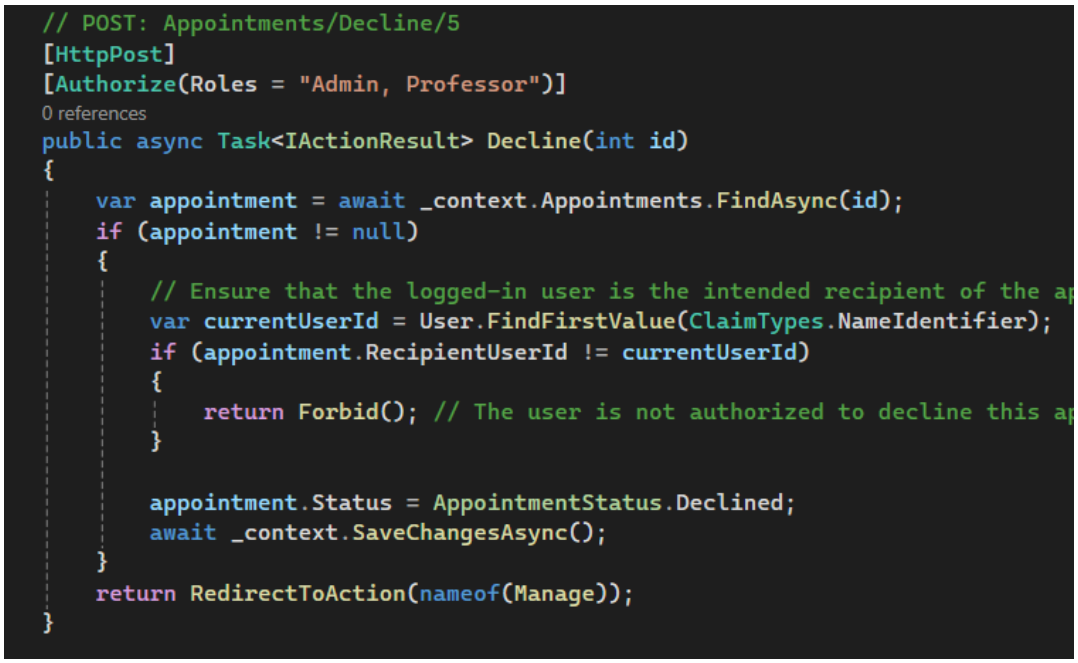
```
// POST: Appointments/Create
[HttpPost]
[ValidateAntiForgeryToken]
[Authorize(Roles = "Student")]
0 references
public async Task<IActionResult> Create([Bind("Title,Description,AppointmentDate,UserId")] Appointment appointment, string recipientUse
{
    if (ModelState.IsValid)
    {
        // Set the logged-in user's ID as the creator of the appointment
        var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);
        appointment.UserId = userId; // The creator's ID
        appointment.CreatedAt = DateTime.Now;
        appointment.Status = AppointmentStatus.Pending;

        // Set the recipient (the person the appointment is intended for)
        appointment.RecipientUserId = recipientUserId; // This is the selected user from the dropdown

        _context.Add(appointment);
        await _context.SaveChangesAsync();
        return RedirectToAction("Index2");
    }

    // If something fails, we need to repopulate the ViewBag for the dropdown list
    var usersWithRoleId2 = _context.Users
        .Where(u => _context.UserRoles.Any(ur => ur.UserId == u.Id && ur.RoleId == "2"))
        .ToList();

    ViewBag.UsersWithRoleId2 = new SelectList(usersWithRoleId2, "Id", "Email");

    return View(appointment);
}
```

4.4.2 Appointment Management

Με τις POST μεθόδους Accept & Decline , ο χρήστης Admin , μπορεί είτε να δεχτεί (Accept) ένα αίτημα για ραντεβού , είτε να αρνηθεί ένα αίτημα για ραντεβού , δηλαδή αλλάζουν το status των ραντεβού στην βάση από pending σε είτε Accepted , είτε Declined.

```
// POST: Appointments/Accept/5
[HttpPost]
[Authorize(Roles = "Admin, Professor")]
0 references
public async Task<IActionResult> Accept(int id)
{
    var appointment = await _context.Appointments.FindAsync(id);
    if (appointment != null)
    {
        // Ensure that the logged-in user is the intended recipient of the a
        var currentUserId = User.FindFirstValue(ClaimTypes.NameIdentifier);
        if (appointment.RecipientUserId != currentUserId)
        {
            return Forbid(); // The user is not authorized to accept this ap
        }

        appointment.Status = AppointmentStatus.Accepted;
        appointment.AcceptedByUserId = currentUserId;

        await _context.SaveChangesAsync();
    }
    return RedirectToAction(nameof(Manage));
}
```

Εικόνα – η POST μέθοδος Accept()



```
// POST: Appointments/Decline/5
[HttpPost]
[Authorize(Roles = "Admin, Professor")]
0 references
public async Task<IActionResult> Decline(int id)
{
    var appointment = await _context.Appointments.FindAsync(id);
    if (appointment != null)
    {
        // Ensure that the logged-in user is the intended recipient of the ap
        var currentUserId = User.FindFirstValue(ClaimTypes.NameIdentifier);
        if (appointment.RecipientUserId != currentUserId)
        {
            return Forbid(); // The user is not authorized to decline this ap
        }

        appointment.Status = AppointmentStatus.Declined;
        await _context.SaveChangesAsync();
    }
    return RedirectToAction(nameof(Manage));
}
```

Εικόνα – η POST μέθοδος Decline()

4.5 Multiple Choice Quiz Creation & Examination

4.5.1 Multiple Choice Quiz Creation

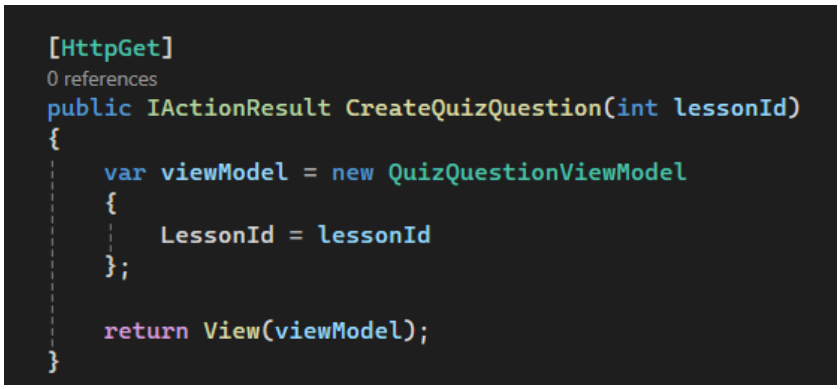
Η μέθοδος ManageQuizQuestion λαμβάνει ως είσοδο το lessonId και αναζητά το μάθημα μαζί με τις ερωτήσεις κουίζ του. Εάν το μάθημα δεν βρεθεί, επιστρέφει NotFound. Αν υπάρχουν περισσότερες από 10 ερωτήσεις, επιλέγει τυχαία 10 και ενημερώνει τον καθηγητή μέσω TempData. Στη συνέχεια, δημιουργεί ένα ViewModel (ref 10) με τα δεδομένα του μαθήματος και τις επιλεγμένες ερωτήσεις κουίζ και το αποστέλλει στην αντίστοιχη θέα (View).

```
[HttpGet]
0 references
public IActionResult ManageQuizQuestion(int lessonId)
{
    var lesson = _context.Lessons
        .Include(l => l.QuizQuestions)
        .SingleOrDefault(l => l.Id == lessonId);

    if (lesson == null)
    {
        return NotFound();
    }

    var quizQuestions = lesson.QuizQuestions.ToList();

    // If there are more than 10 questions, randomly select 10
    if (quizQuestions.Count > 10)
    {
        quizQuestions = quizQuestions.OrderBy(q => Guid.NewGuid()).Take(10).ToList();
        // Inform the professor that only 10 questions will be used
        TempData["Message"] = "Only 10 questions will be used for the quiz. The selecti
    }

    var viewModel = new ManageQuizQuestionViewModel
    {
        LessonId = lessonId,
        LessonTitle = lesson.Title,
        QuizQuestions = quizQuestions
    };

    return View(viewModel);
}
```

Εικόνα – η GET μέθοδος ManageQuizQuestion()

4.5.2 Multiple Choice Quiz Creation

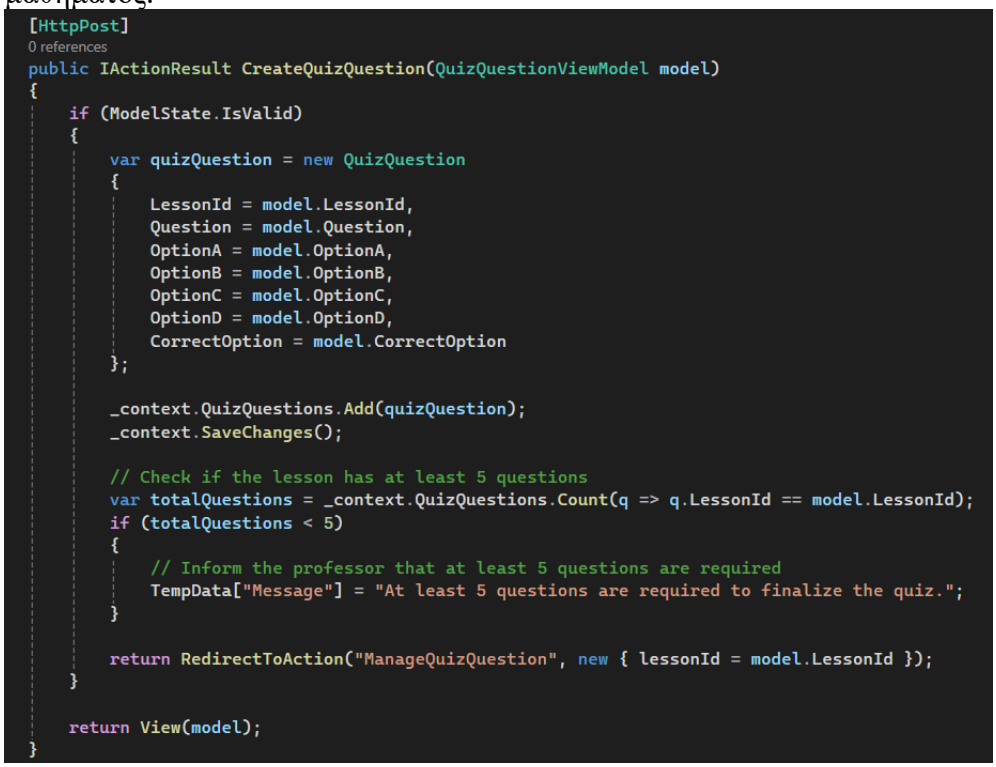
Η GET μέθοδος CreateQuizQuestion δημιουργεί ένα ViewModel για να επιτρέψει στον καθηγητή να δημιουργήσει μια νέα ερώτηση κουίζ για ένα συγκεκριμένο μάθημα, βασισμένο στο lessonId, και επιστρέφει τη θέα (View) για την εισαγωγή της ερώτησης.



```
[HttpGet]
0 references
public IActionResult CreateQuizQuestion(int lessonId)
{
    var viewModel = new QuizQuestionViewModel
    {
        LessonId = lessonId
    };

    return View(viewModel);
}
```

Εικόνα – η GET μέθοδος CreateQuizQuestion()

Η μέθοδος **CreateQuizQuestion** (POST) ελέγχει αν το model που υποβάλλει ο χρήστης είναι έγκυρο μέσω του **ModelState**. Εάν είναι έγκυρο, δημιουργεί μια νέα ερώτηση κουίζ και την αποθηκεύει στη βάση δεδομένων. Μετά την αποθήκευση, ελέγχει αν το μάθημα έχει τουλάχιστον 5 ερωτήσεις και, αν όχι, ενημερώνει τον καθηγητή μέσω **TempData** ότι απαιτούνται τουλάχιστον 5 ερωτήσεις για την ολοκλήρωση του κουίζ. Τέλος, ανακατευθύνει τον χρήστη στη σελίδα **ManageQuizQuestion** για τη διαχείριση των ερωτήσεων του μαθήματος.



```
[HttpPost]
0 references
public IActionResult CreateQuizQuestion(QuizQuestionViewModel model)
{
    if (ModelState.IsValid)
    {
        var quizQuestion = new QuizQuestion
        {
            LessonId = model.LessonId,
            Question = model.Question,
            OptionA = model.OptionA,
            OptionB = model.OptionB,
            OptionC = model.OptionC,
            OptionD = model.OptionD,
            CorrectOption = model.CorrectOption
        };

        _context.QuizQuestions.Add(quizQuestion);
        _context.SaveChanges();

        // Check if the lesson has at least 5 questions
        var totalQuestions = _context.QuizQuestions.Count(q => q.LessonId == model.LessonId);
        if (totalQuestions < 5)
        {
            // Inform the professor that at least 5 questions are required
            TempData["Message"] = "At least 5 questions are required to finalize the quiz.";
        }

        return RedirectToAction("ManageQuizQuestion", new { lessonId = model.LessonId });
    }

    return View(model);
}
```

Εικόνα – η POST μέθοδος CreateQuizQuestion()

4.5.3 Multiple Choice Quiz Examination

Η μέθοδος **TakeQuiz** λαμβάνει ως είσοδο το lessonId και ελέγχει αν ο χρήστης έχει ήδη συμμετάσχει στο κουίζ για το συγκεκριμένο μάθημα, ανακτώντας το userId και τα αποτελέσματα από τη βάση δεδομένων. Αν ο χρήστης έχει ήδη συμμετάσχει, εμφανίζεται μήνυμα σφάλματος μέσω **TempData** και γίνεται ανακατεύθυνση στη σελίδα αποτελεσμάτων του κουίζ. Αν δεν έχει συμμετάσχει, ανακτά το μάθημα, τις ερωτήσεις κουίζ και τις πληροφορίες εξέτασης από τη βάση δεδομένων. Ελέγχει επίσης αν υπάρχει προθεσμία και αν έχει λήξει. Αν η προθεσμία έχει περάσει, εμφανίζεται μήνυμα σφάλματος και γίνεται ανακατεύθυνση στη σελίδα λεπτομερειών του μαθήματος. Εάν όλα είναι έγκυρα, επιστρέφεται η προβολή του κουίζ.

```
public IActionResult TakeQuiz(int lessonId)
{
    var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);

    // Check if the user has already taken the quiz for this lesson
    bool hasTakenQuiz = _context.QuizResults
        .Any(qr => qr.LessonId == lessonId && qr.UserId == userId);

    if (hasTakenQuiz)
    {
        TempData["ErrorMessage"] = "You have already taken this quiz.";
        return RedirectToAction("QuizResults", new { lessonId = lessonId });
    }

    var lesson = _context.Lessons
        .Include(l => l.QuizQuestions)
        .Include(l => l.Examination) // Include the examination to check the due date
        .SingleOrDefault(l => l.Id == lessonId);

    if (lesson == null)
    {
        return NotFound();
    }

    // Check if the examination exists and if the due date has passed
    if (lesson.Examination != null && lesson.Examination.ExaminationDate.HasValue
        && lesson.Examination.ExaminationDate.Value < DateTime.Now)
    {
        TempData["ErrorMessage"] = "The due date for this quiz has passed. You can no longer take the quiz.";
        return RedirectToAction("LessonDetails", new { id = lessonId }); // Redirect to a lesson detail page
    }

    return View(lesson);
}
```

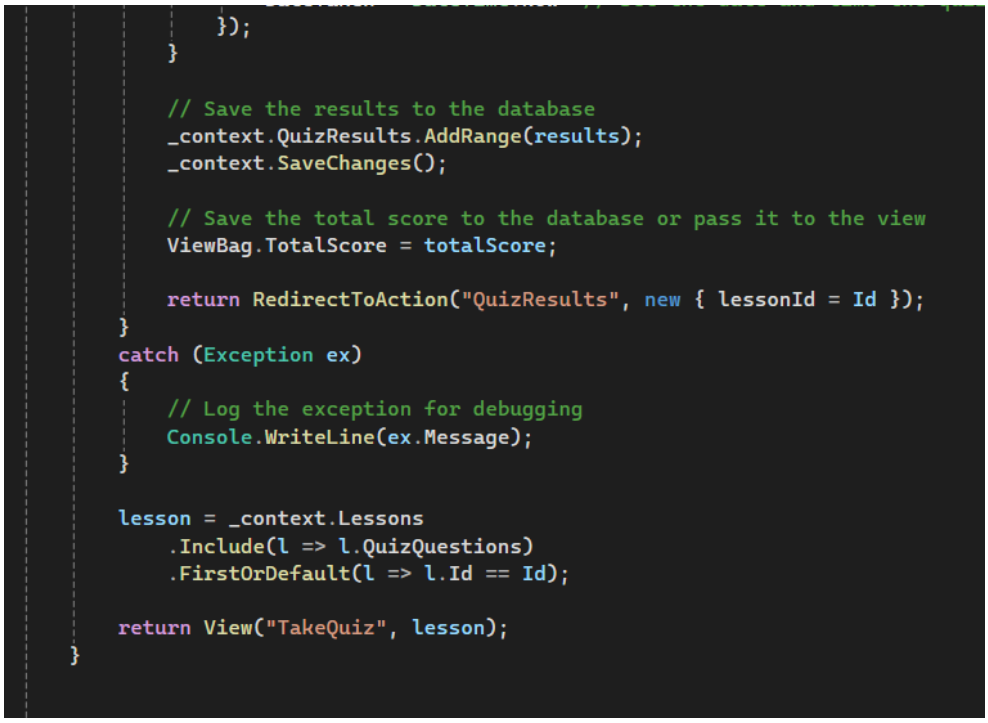
Εικόνα – η GET μέθοδος TakeQUiz()

Η μέθοδος **SubmitQuiz** είναι μια HTTP POST μέθοδος που λαμβάνει το Id (ID του μαθήματος), έναν πίνακα με ερωτήσεις κουίζ και έναν πίνακα με τις επιλεγμένες απαντήσεις του χρήστη. Αρχικά, ελέγχει αν ο χρήστης έχει ήδη υποβάλει το κουίζ χρησιμοποιώντας το userId και αν ναι, εμφανίζει μήνυμα σφάλματος και ανακατευθύνει στη σελίδα αποτελεσμάτων (**QuizResults**). Στη συνέχεια, ελέγχει αν υπάρχει προθεσμία για την εξέταση και αν έχει λήξει, επιστρέφει μήνυμα σφάλματος. Υπολογίζει το συνολικό σκορ, συγκρίνοντας τις επιλεγμένες απαντήσεις με τις σωστές, δημιουργεί μια **λίστα με τα** αποτελέσματα (**QuizResults**), και τα αποθηκεύει στη βάση δεδομένων μαζί με την ημερομηνία υποβολής. Αν όλα πάνε καλά, ανακατευθύνει στη σελίδα

QuizResults. Σε περίπτωση σφάλματος, καταγράφει το λάθος και επιστρέφει την προβολή του κουίζ.

```
[HttpPost]
0 references
public IActionResult SubmitQuiz(int Id, int[] QuizQuestions, string[] SelectedOptions)
{
    var userId = User.FindFirstValue(ClaimTypes.NameIdentifier);

    // Check if the user has already taken the quiz for this lesson
    bool hasTakenQuiz = _context.QuizResults
        .Any(qr => qr.LessonId == Id && qr.UserId == userId);

    if (hasTakenQuiz)
    {
        TempData["ErrorMessage"] = "You have already submitted this quiz.";
        return RedirectToAction("QuizResults", new { lessonId = Id });
    }

    var lesson = _context.Lessons
        .Include(l => l.Examination) // Include examination to check the due date
        .FirstOrDefault(l => l.Id == Id);

    if (lesson == null)
    {
        return NotFound();
    }

    // Check if the examination exists and if the due date has passed
    if (lesson.Examination != null && lesson.Examination.ExaminationDate.HasValue
        && lesson.Examination.ExaminationDate.Value < DateTime.Now)
    {
        TempData["ErrorMessage"] = "The due date for this quiz has passed. You can no longer submit the quiz.";
        return RedirectToAction("LessonDetails", new { id = Id }); // Redirect to a lesson detail page or any ot
    }
```

```
        try
        {
            if (SelectedOptions == null)
            {
                SelectedOptions = new string[0];
            }

            double totalScore = 0;
            var results = new List<QuizResult>();

            for (int i = 0; i < QuizQuestions.Length; i++)
            {
                int questionId = QuizQuestions[i];
                string selectedOption = SelectedOptions.Length > i ? SelectedOptions[i] : null;

                // Retrieve the correct option from the database
                var correctOption = _context.QuizQuestions
                    .Where(q => q.Id == questionId)
                    .Select(q => q.CorrectOption)
                    .FirstOrDefault();

                // Determine if the selected option is correct
                var isCorrect = !string.IsNullOrEmpty(selectedOption) && correctOption == selectedOption;

                // Increment total score for each correct answer
                if (isCorrect)
                {
                    totalScore += 1;
                }

                // Create a new QuizResult instance
                results.Add(new QuizResult
                {
                    UserId = userId,  // Set the student ID
                    LessonId = Id,  // Set the associated lesson ID
                    QuizQuestionId = questionId,  // Set the quiz question ID
                    IsCorrect = isCorrect,  // Store whether the answer was correct
```



```
            });
        }

        // Save the results to the database
        _context.QuizResults.AddRange(results);
        _context.SaveChanges();

        // Save the total score to the database or pass it to the view
        ViewBag.TotalScore = totalScore;

        return RedirectToAction("QuizResults", new { lessonId = Id });
    }
    catch (Exception ex)
    {
        // Log the exception for debugging
        Console.WriteLine(ex.Message);
    }

    lesson = _context.Lessons
        .Include(l => l.QuizQuestions)
        .FirstOrDefault(l => l.Id == Id);

    return View("TakeQuiz", lesson);
}
```

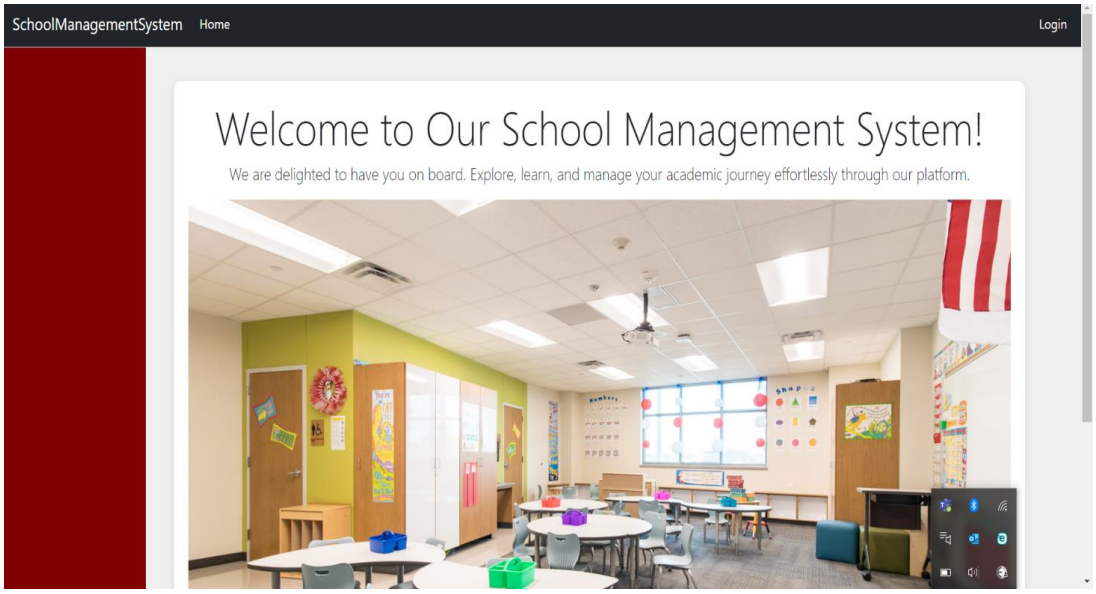
Εικόνα – η POST μέθοδος SubmitQuiz()

Κεφάλαιο 5ο

User Manual

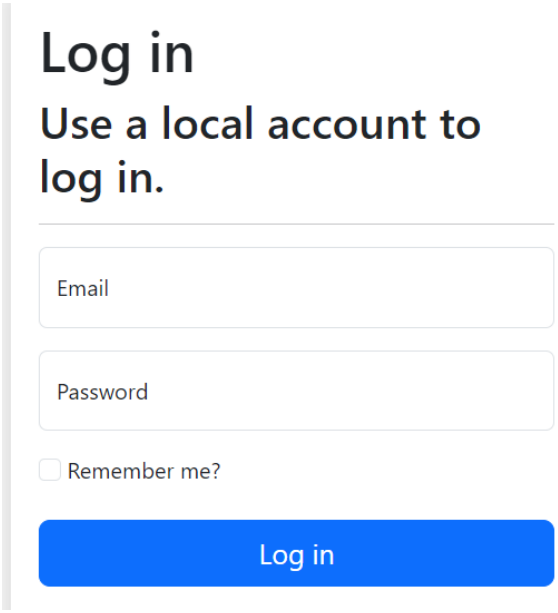
5.1 Admin

1) **Home page :** Η Σελίδα που καλωσορίζει τον χρήστη.



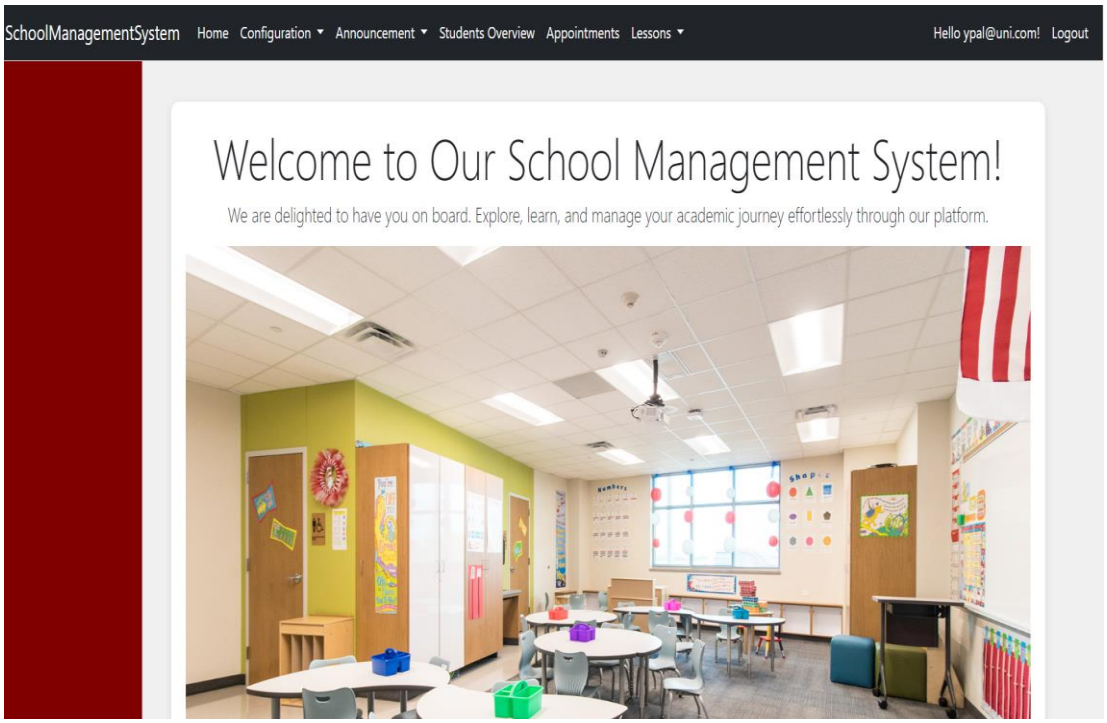
Εικόνα – General Home Page

2) **Login page :** Πατώντας το λινκ Login , ο χρήστης βλέπει αυτήν την φόρμα συμπλήρωσης στοιχείων :



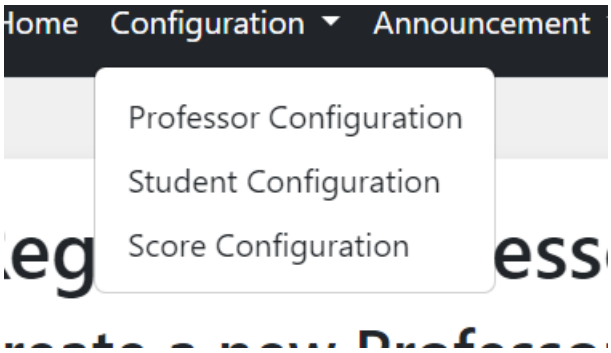
Εικόνα – Φόρμα συμπλήρωσης στοιχείων για είσοδο στην εφαρμογή

3) **Admin Home page :** Αφού ο χρήστης συμπληρώσει σωστά τα στοιχεία του , βλέπει αυτήν την σελίδα .



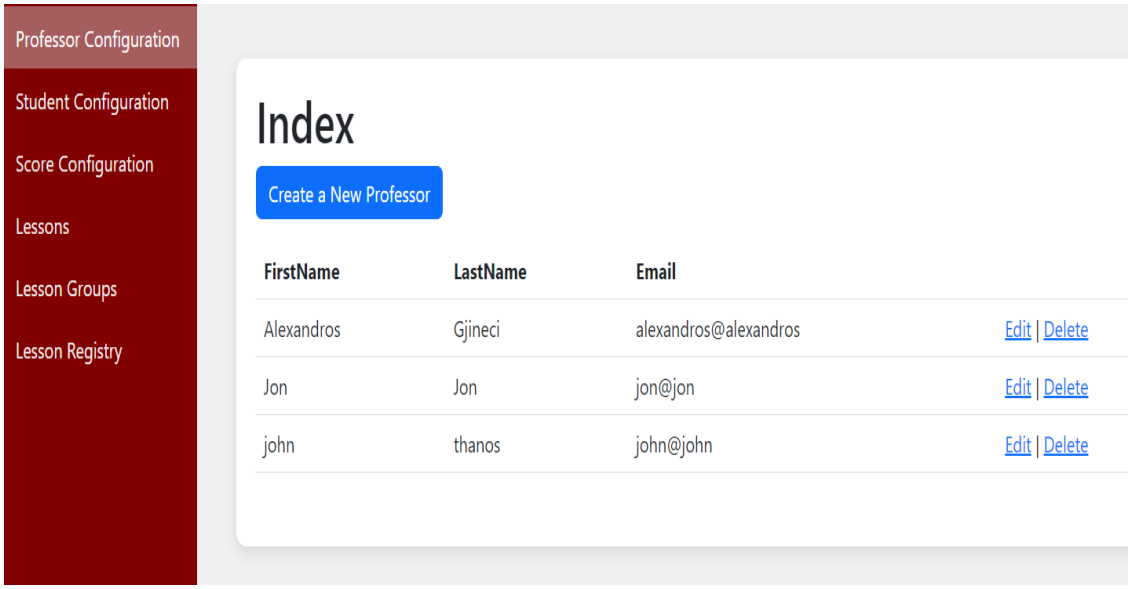
Εικόνα – Admin Home Page

4) **Configuration Tab :** Πατώντας το tab Configuration , ο χρήστης , ο χρήστης βλέπει τρία links , Professor Configuration , Student Configuration , Score Configuration.



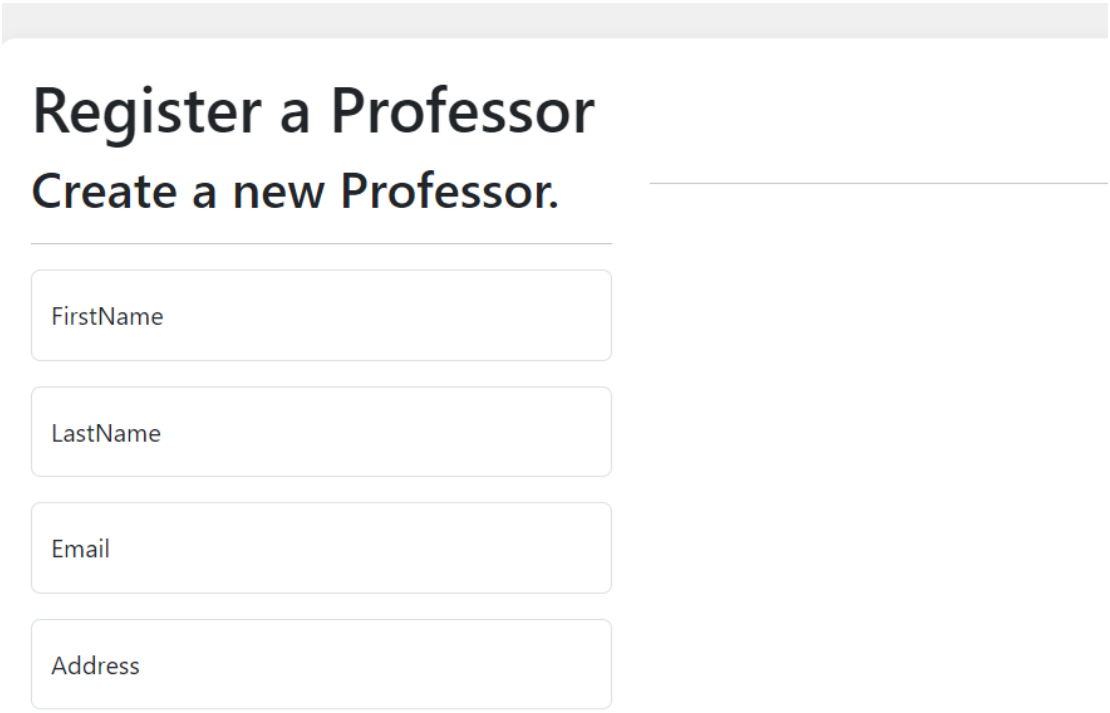
Εικόνα – Dropdown Configuration

5) **Professor Configuration :** Πατώντας το tab **Professor Configuration ,** ο χρήστης μπορεί να επεξεργαστεί τους χρήστες-**Professors**(Καθηγητές). Εδώ βλέπουμε τους ήδη εγγεγραμμένους χρηστες-Καθηγητες, οπού μπορούμε να επεξεργαστούμε τα στοιχεία τους **(Edit**) , να τους διαγράψουμε (**Delete**) αλλά και να δημιουργήσουμε νέους χρηστες-Καθηγητες (**Create a New Professor**).



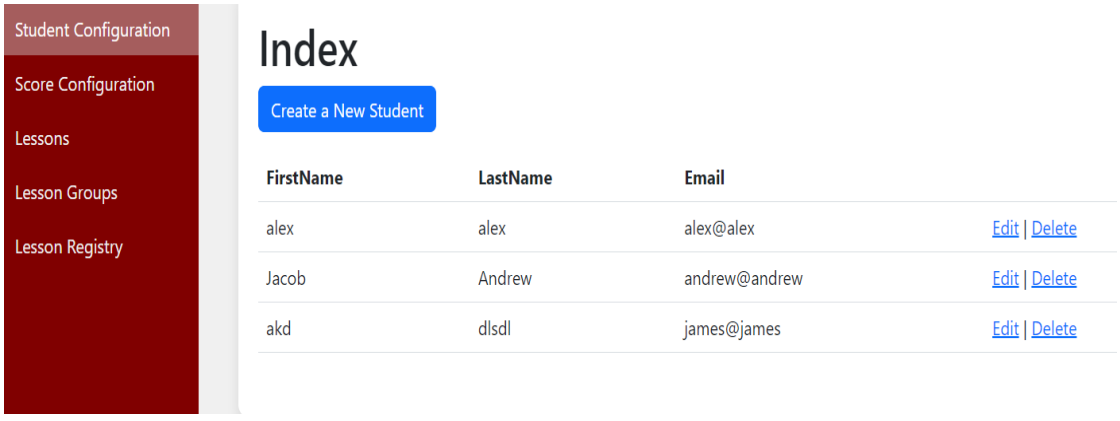
Εικόνα – Πίνακας εγγεγραμμένων χρηστών καθηγητών

6) **Create a New Professor :** Πατώντας το πλήκτρο **Create a New Porfessor** , εμφανίζεται η φόρμα συμπλήρωσης στοιχεία εγγραφή χρηστών καθηγητών. Ο Admin απαιτείται να συμπληρώσει το Όνομα και το επίθετο το καθηγητή, το email, το ΑΦΜ το Status του καθώς και τον κωδικό πρόσβασης του.



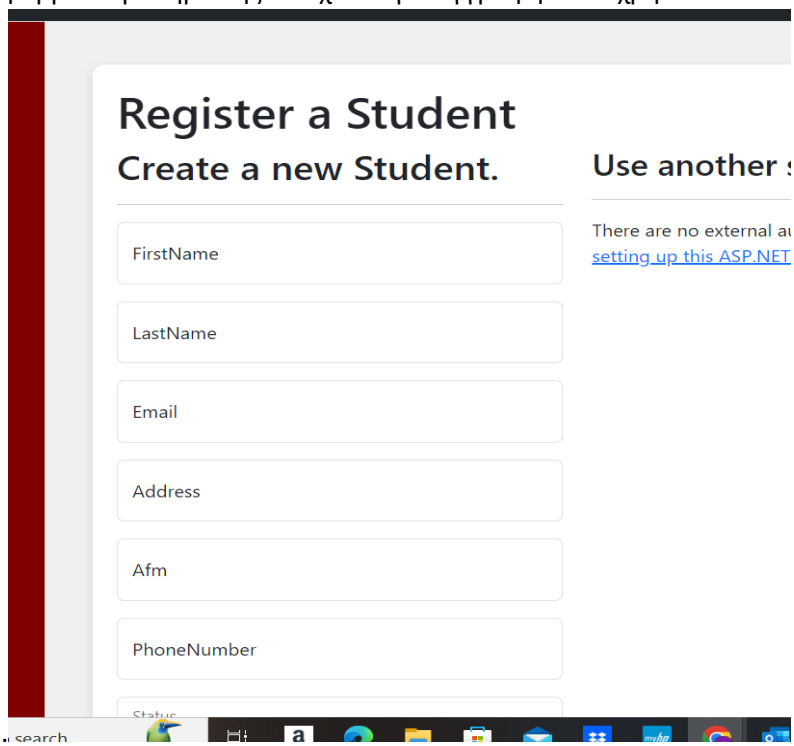
Εικόνα – Φόρμα καταχώρησης νέου χρήστη καθηγητη

7) **Student Configuration :** Πατώντας το λινκ Student Configuration , ο χρήστης βλέπει ένα πίνακα με τους ήδη εγγεγραμμένους χρηστες-Μαθητες(Students).Μπορεί να δημιουργήσει νέους χρηστες-Μαθητες (Create New Student), να τους επεξεργαστεί (Edit), να τους διαγράψει (Delete).



Εικόνα – Πίνακας εγγεγραμμένων χρηστών μαθητών

8) **Create a New Student :** Πατώντας το πλήκτρο Create a new Student , εμφανίζεται η φόρμα συμπλήρωσης στοιχείων για εγγραφή νέων χρηστων-Μαθητων



Εικόνα – Φόρμα καταχώρησης νέου χρήστη μαθητή

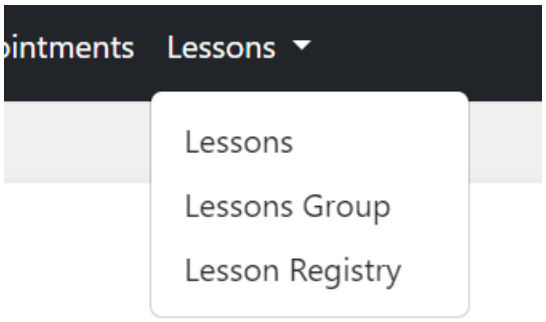
9) **Score Configuration :** Στο tab **Score Configuration,** ο χρήστης **Admin** μπορεί δει ήδη επικυρωμένες βαθμολογίες μαθητών η αιτήματα (requests) για επικύρωση βαθμολογίας.

Εικόνα – Πίνακας βαθμολογίας μαθητών

10) Ο Χρήστης έχει την δυνατότητα να επικύρωση βαθμολογίες δηλαδή να αλλάξει το status τους από Pending σε Finalized.

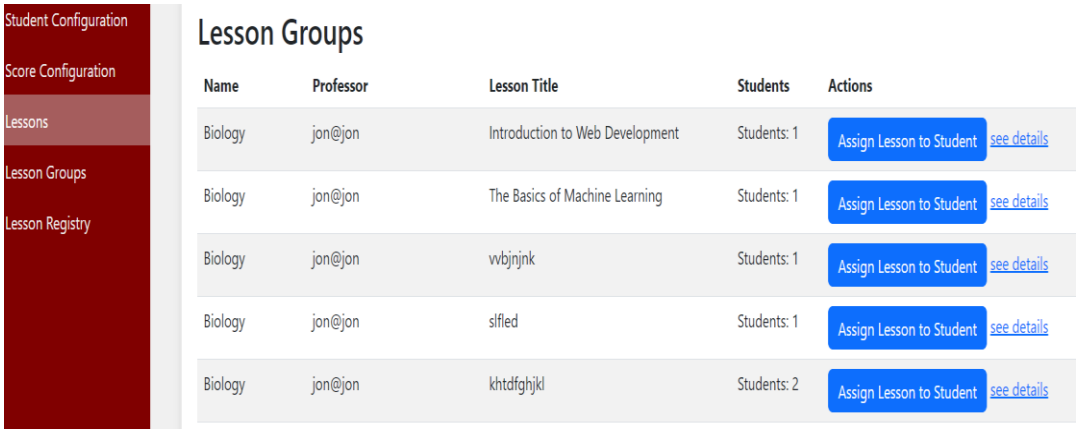
Εικόνα – Αίτημα για οριστικοποίηση βαθμολογίας

11) Lessons : Πατώντας το λινκ Lessons , φαίνονται τα εξής λινκς :



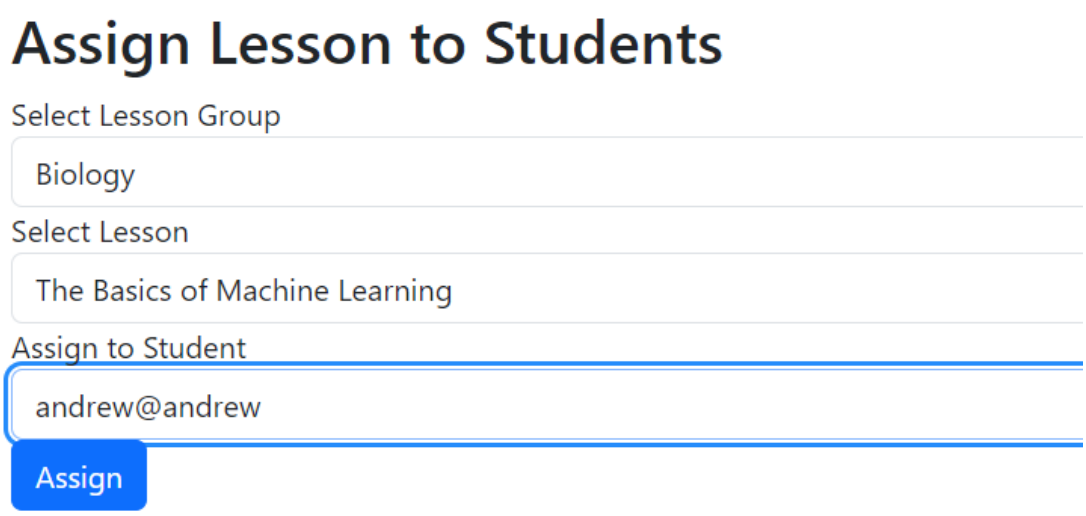
Εικόνα – Dropdown Lessons

12) Lessons : Επιλέγοντας το πρώτο λινκ , Lessons , ο χρήστης βλέπει ένα πίνακα με τα μαθήματα και τις κατηγόριες μαθήματω**ν (Lesson Groups)** τα οποία ανήκουν καθώς και πόσοι μαθητές είναι εγγεγραμμένοι σε αυτά.



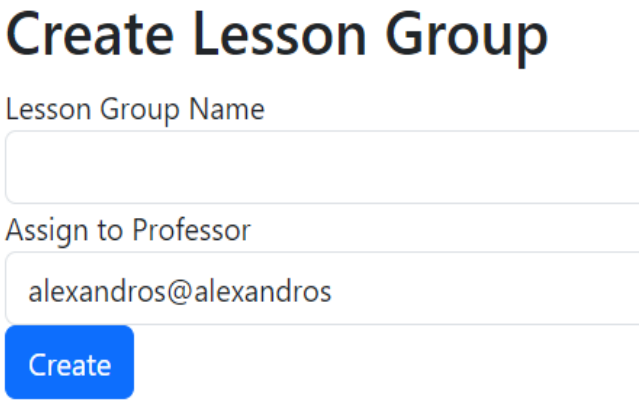
Εικόνα – Μαθήματα ανα Lesson Group

13) Assing Lesson to Student : Πατώντας το λινκ **Assign Lesson to Students ,** ο χρήστης μπορεί να αναθέσει το συγκεκριμένο μάθημα σε μαθητή**, οποίος όμως δεν έχει ήδη εγγράφει σε αυτό .**



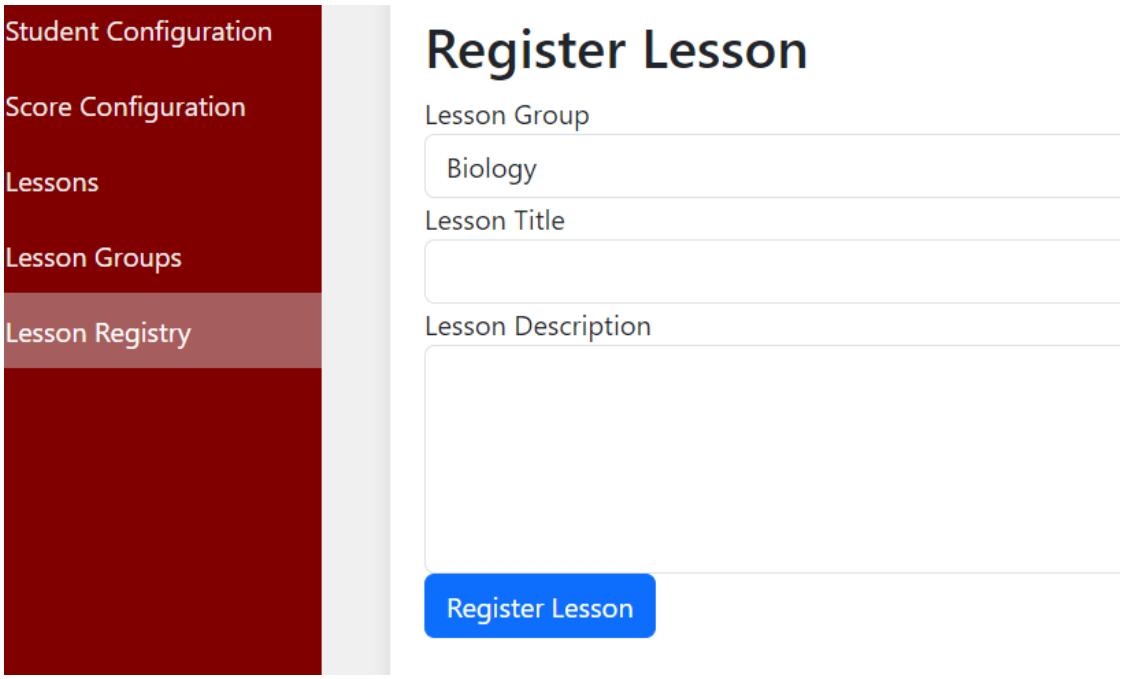
Εικόνα – Assignment μαθήματος σε μαθητή

14) **Lesson Groups :** Επιλέγοντας από αριστερά το λινκ **Lesson Groups ,** ο χρήστης μπορεί να δημιουργήσει νέα Lesson Groups , καθώς και να τα αναθέσει σε ήδη υπάρχοντες καθηγητές.



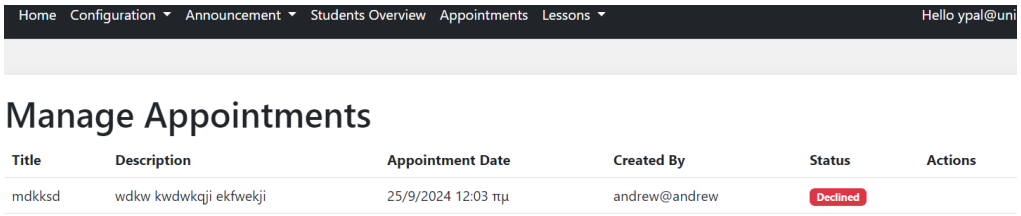
Εικόνα – Assignment Lesson Group σε καθηγητές

15) **Lesson Registry** : Επιλέγοντας από αριστερά το link **Lesson Registry**, ο χρήστης μπορεί να δημιουργήσει μαθήματα που υπάγονται σε ήδη υπάρχουσες κατηγορίες μαθημάτων (**Lesson Groups**).



Εικόνα – Καταχώρηση νέου μαθήματος

16) **Appointments :** Στο tab Appointments , ο χρήστης μπορεί να δει τα υπάρχοντα ραντεβού του καθώς και να διαχειριστεί αιτήματα για ραντεβού από χρηστες-Μαθητες.

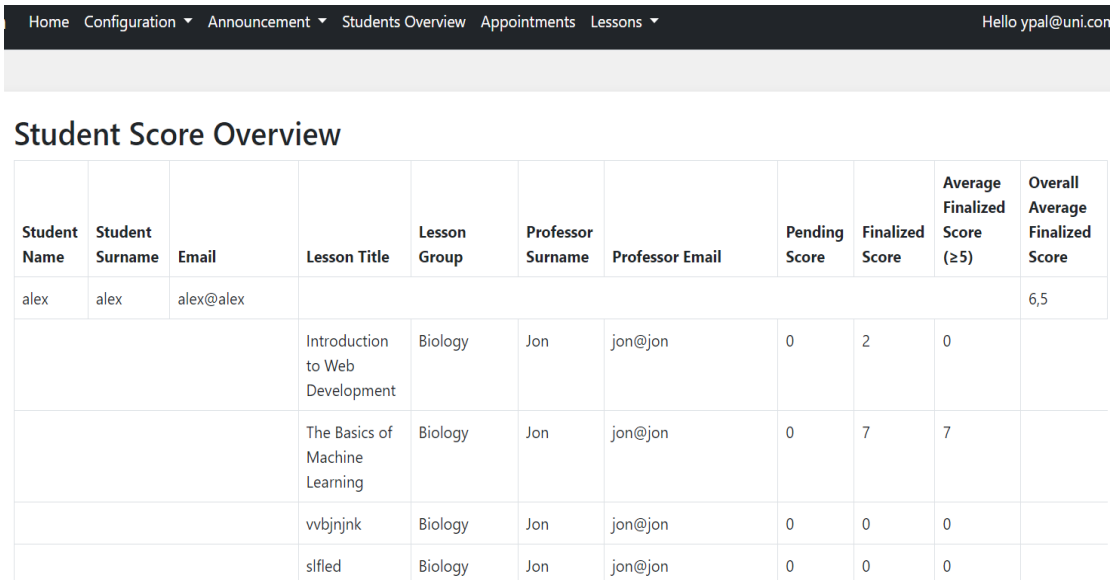


Εικόνα – Πίνακας αιτημάτων ραντεβού

17) Ο Χρήστης μπορεί να αποδεκτει(**Accept**) η να αρνηθεί **(Decline**) αιτήματα για ραντεβού.

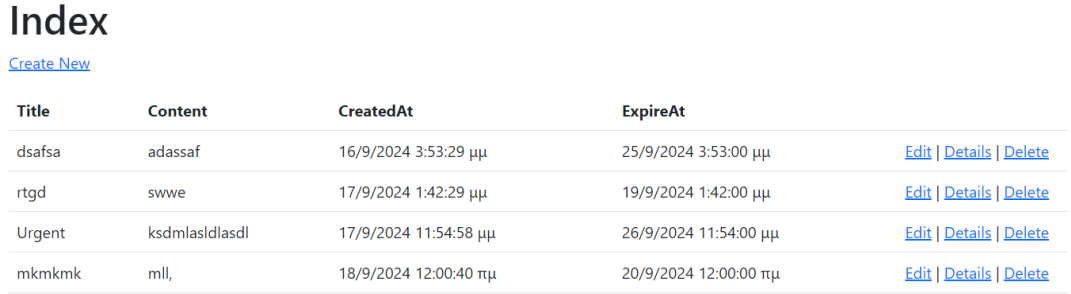
Εικόνα – Αιτήματα για ραντεβού

18) **Student Overview**: Πατώντας το tab **Student Overview** , ο χρήστης μπορεί να δει την πλήρη επίδοση του κάθε μαθητή.



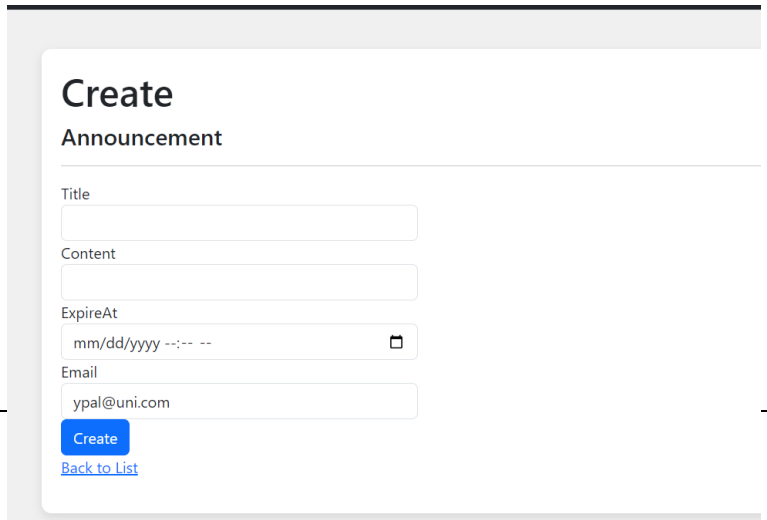
Εικόνα – Βαθμολογία μαθητών

19) Announcements : Επιλέγοντας το tab Announcements , ο χρήστης μπορεί να δει τις ήδη υπάρχουσες ανακοινώσεις ¨.



Εικόνα – Index πίνακας Ανακοινώσεων

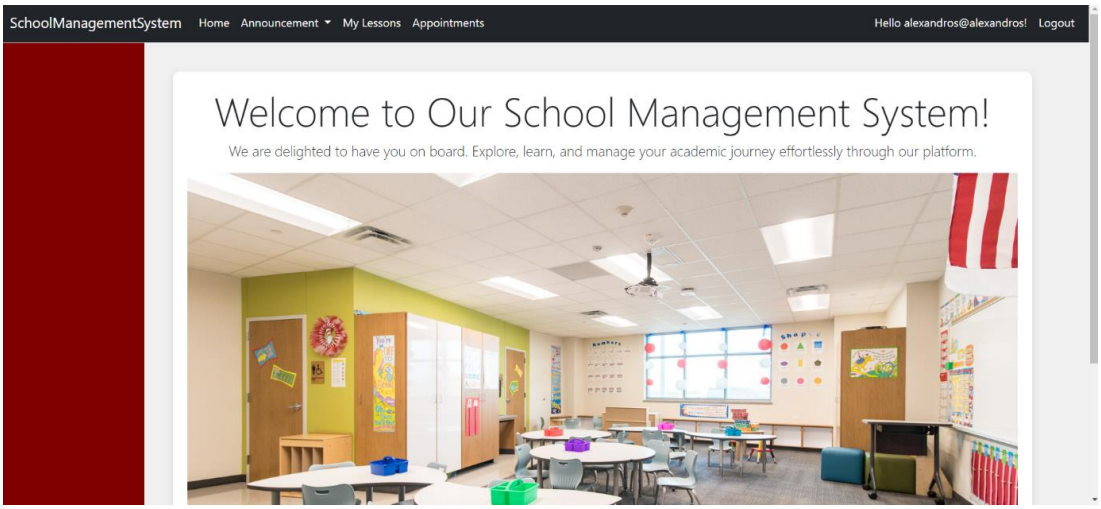
20) Announcements Configuration : Επιλέγοντας το tab Announcements Configuration , ο χρήστης μπορεί να δημιουργήσει ανακοινώσεις Γραμματείας .



Εικόνα – Φόρμα υποβολής Ανακοίνωσης

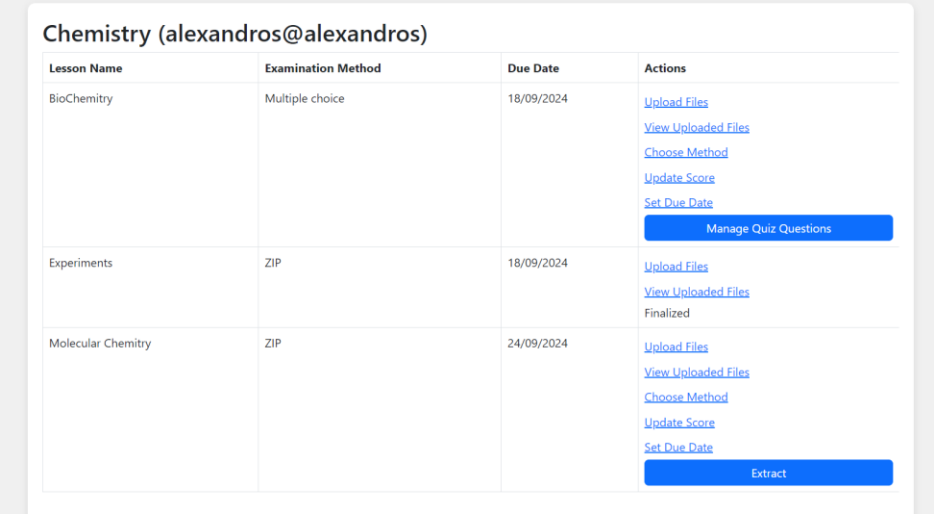
5.2 Professor

1) Κάνοντας **Login** στην εφαρμογή με τους κωδικούς ως χρήστης-καθηγητής, χρήστης βλέπει την παρακάτω οθόνη.



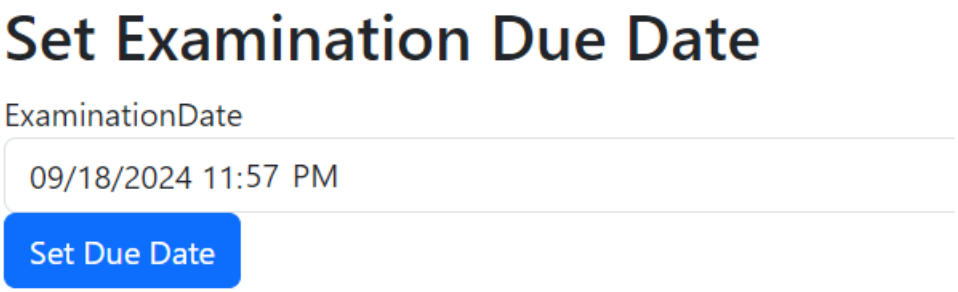
Εικόνα – Professor Home Page

2) **My Lessons :** Πατώντας το link **My Lessons**, ο χρήστης βλέπει τα υπάρχοντα μαθήματα για τα οποία είναι υπεύθυνος, μέσω της κατηγορίας μαθημάτων στην οποία είναι εγγεγραμμένος..



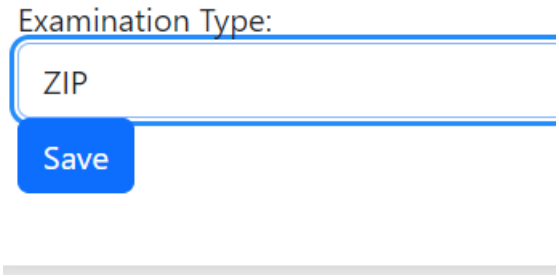
Εικόνα – Κατάλογος μαθήματων

3) **Set Due Date :** Επιλέγοντας το λινκ **Set Due Date ,** ο χρήστης μπορεί να θέσει Deadline , δηλαδή τελική ημερομηνία εξέτασης του μαθήματος. Προσοχή , ο χρήστης δεν μπορεί να επιλέξει ξανά ημερομηνία εξέτασης μαθήματος , εφόσον έχει ήδη επιλεγμένη ημερομηνία.



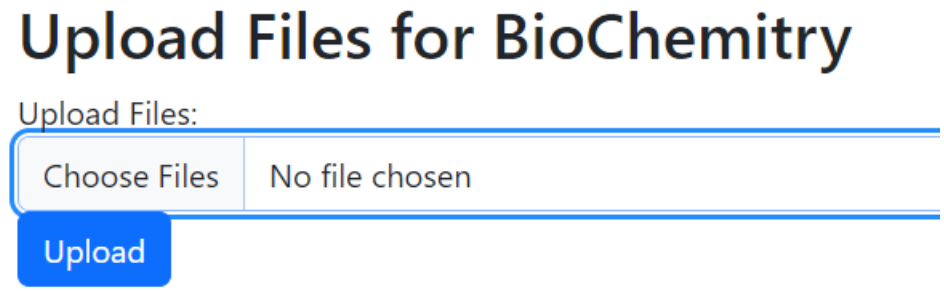
Εικόνα – Ενημέρωση Deadline υποβολής εξέτασης

4) **Set Examination Date :** Επιλέγοντας το link **Set Examination Date**, ο χρήστης μπορεί να επιλέξει τον τρόπο εξέτασης του μαθήματος ανάμεσα στις επιλογές: **Multiple choice** test (Τεστ πολλαπλών επιλογών) και ανέβασμα εργασίας σε μορφή .zip (**Upload a file**). Προσοχή: ο χρήστης δεν μπορεί να επιλέξει μέθοδο εξέτασης εφόσον έχει ήδη επιλεγεί κάποια μέθοδος εξέτασης.



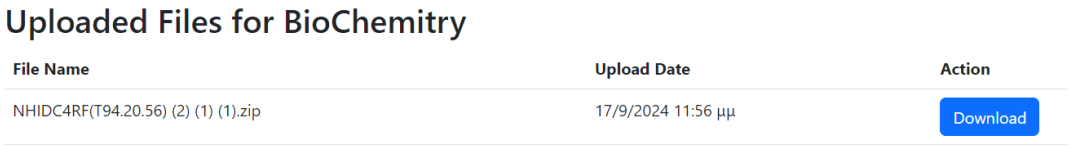
Εικόνα – Επιλογή τρόπου εξέτασης

5) **Upload Files :** Επιλέγοντας το λινκ upload files , ο χρήστης μπορεί να ανεβασει σημειωσεις για τους χρηστες μαθητες που είναι εγγεγραμενοι στα μαθηματα του.



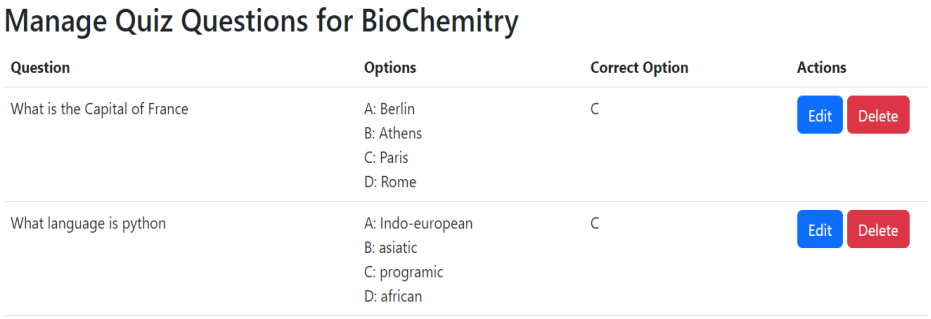
Εικόνα – Ανέβασμα Αρχείων

6) **View Upload Files** : Επιλέγοντας το λινκ **View Upload Files** , ο χρήστης μπορεί να δει τα αρχεία που έχει ανεβάσει**.**

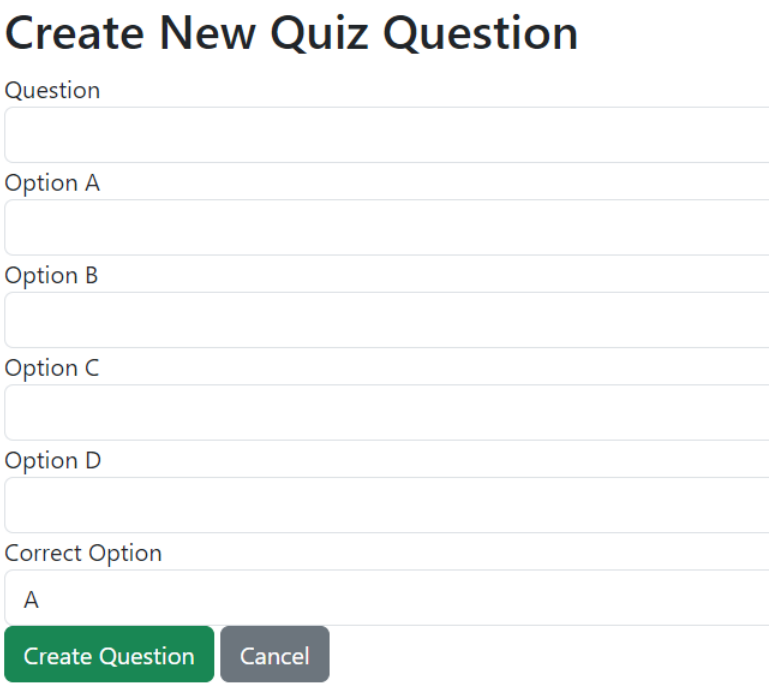


Εικόνα – Ανεβασμένα αρχεία

7) **Manage quiz Questions :** Σε περιπτωση που η μεθοδος εξετασης του μαθηματος είναι Multiple choice (Πολλαπλες επιλογες), εμφανιζεται το λινκ Manage Quiz Questions . Πατωντας το, ο χρηστης μπορει να δημιουργησεις ερωτησεις και πιθανες απαντησεις της συγκεκριμενης ερωτησης . Εχει την δυνατοτητα να θεσει μια από τις 4 απαντησεις ως σωστη απαντηση .

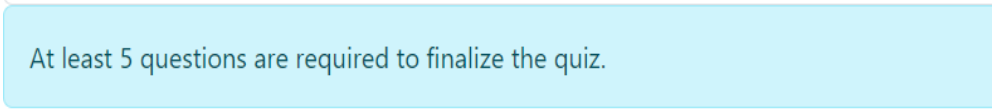


Εικόνα – Manage Questions



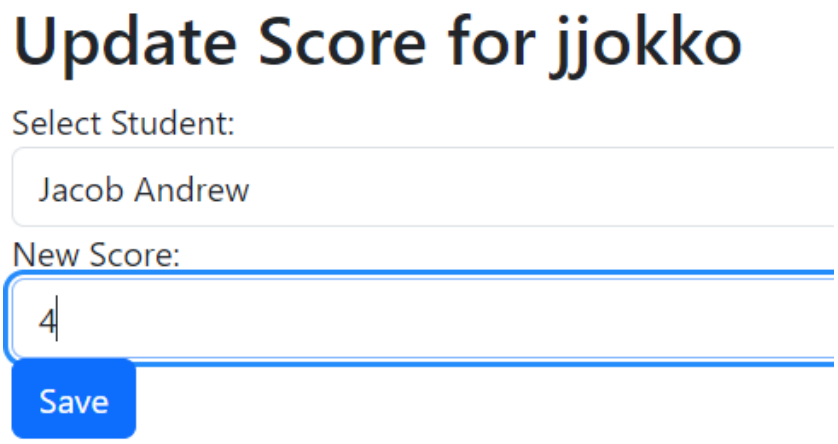
Εικόνα – Φόρμα συμπλήρωσης Ερώτησης

8) Προκειμένου το τεστ να είναι έγκυρο , εμφανίζεται η προειδοποίηση ότι πρέπει να έχουν δημιουργηθεί τουλάχιστον 5 ερωτήσεις για κάθε μάθημα.



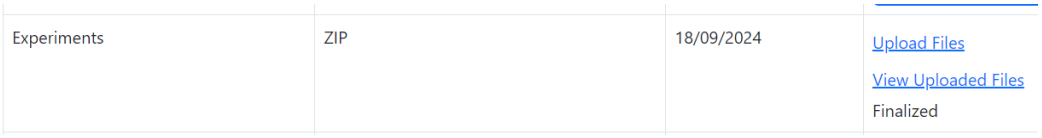
Εικόνα – Προειδοποίηση

9) **Update score :** Επιλέγοντας το link **Update Score**, ο χρήστης μπορεί να καταχωρίσει μια προσωρινή (**Pending**) βαθμολογία για κάθε μαθητή σε κάθε μάθημα στο οποίο είναι εγγεγραμμένος.
Προσοχή: η βαθμολογία μπορεί να καταχωρηθεί μόνο εφόσον έχει οριστεί ημερομηνία εξέτασης, έχει παρέλθει η ημερομηνία αυτή, και έχει ήδη καθοριστεί η μέθοδος εξέτασης. Επιπλέον, η βαθμολογία μπορεί να καταχωρηθεί μόνο μία φορά, και εφόσον έχει οριστικοποιηθεί από τη γραμματεία, δεν υπάρχει δυνατότητα επεξεργασίας.



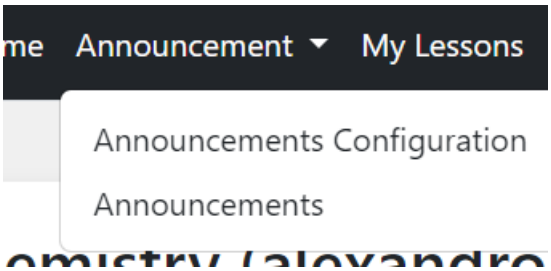
Εικόνα – Update Score

10) Όταν έχει οριστικοποιηθεί η βαθμολογία από τη γραμματεία, τα links "Update Score", "Set Due Date", "Choose Examination Type" γίνονται μη προσβάσιμα και στη θέση τους εμφανίζεται η ένδειξη "Finalized", δείχνοντας έτσι ότι το μάθημα έχει οριστικοποιημένο status.



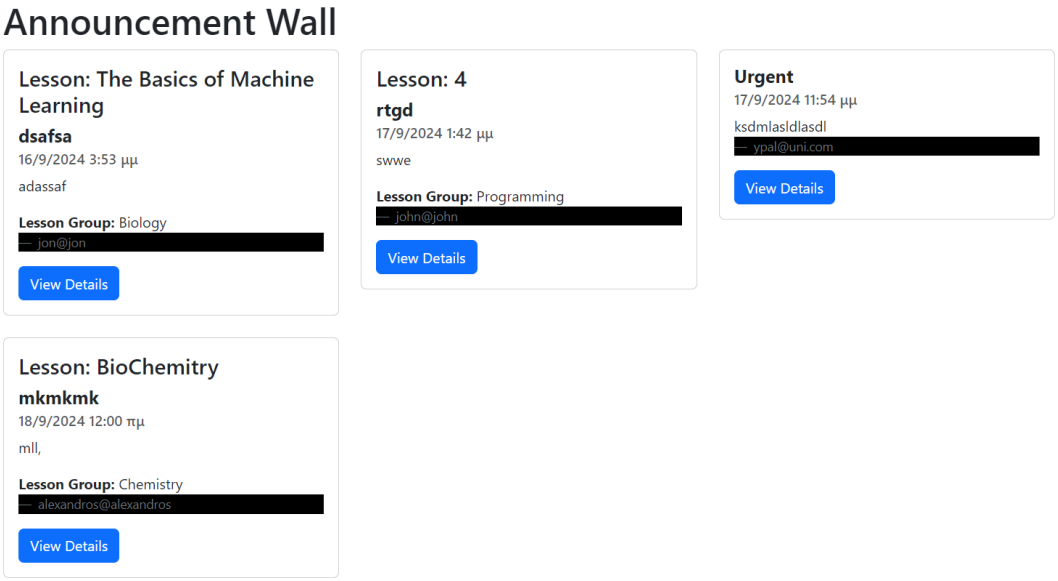
Εικόνα – Τα λινκ μετα την οριστικοποίηση

11) **Announcements :** Επιλέγοντας το tab , Announcements , ο χρήστης μπορεί να επιλέξει τα λινκς : Announcements & Announcements Configuration.



Dropdown Announcements

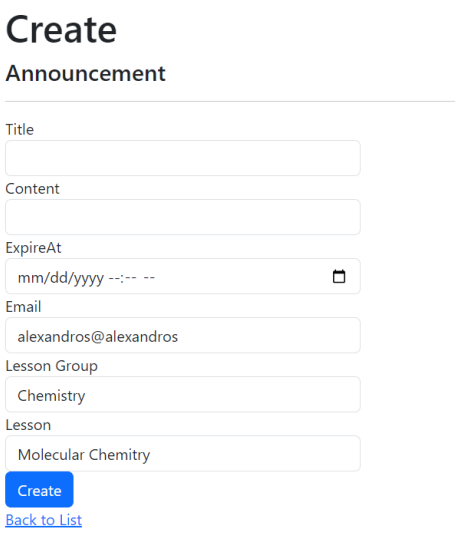
12) Επιλέγοντας το λινκ Announcements από το drop down , ο χρήστης μπορεί να δει το γενικό πίνακα ανακοινώσεων .



Εικόνα – Πίνακας Ανακοινώσεων.

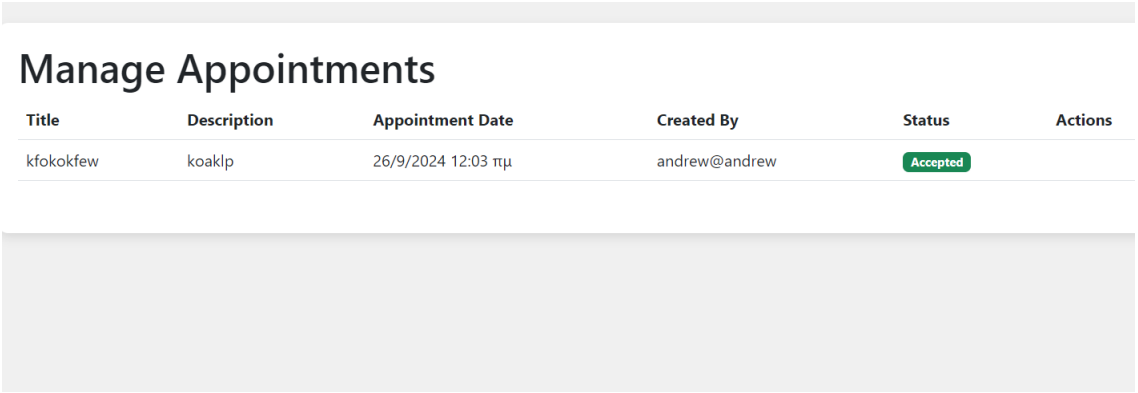
13) **Announcements Configuration :** Επιλέγοντας το λινκ Announcements Configuration , ο χρήστης μπορεί δημιουργήσει ανακοινώσεις χρηστών Καθηγητών , οπού η κάθε μια περιέχει τα εξής :

- Τίτλος.
- Περιεχόμενο.
- Ημερομηνία λήξης της ανακοίνωσης (λήγει μετα την παρέλευση της ημερομηνίας.)
- Email του χρήστη που δημιουργεί την ανακοίνωση.
- Κατηγορία μαθήματος.
- Μάθημα



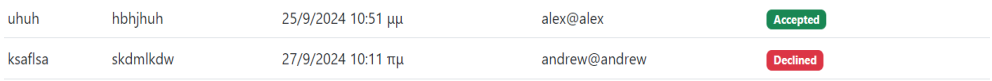
Εικόνα – Φόρμα Create Announcement

14) **Appointments** : Επιλέγοντας το tab Appointments, ο χρήστης μπορεί να δει τα υπάρχοντα κλεισμένα ραντεβού του, καθώς και αιτήματα για ραντεβού από χρήστες-Μαθητές. Ο χρήστης μπορεί είτε να δεχτεί (Accept) είτε να αρνηθεί (Decline) τα αιτήματα για ραντεβού.



Ραντεβού(Accepted)

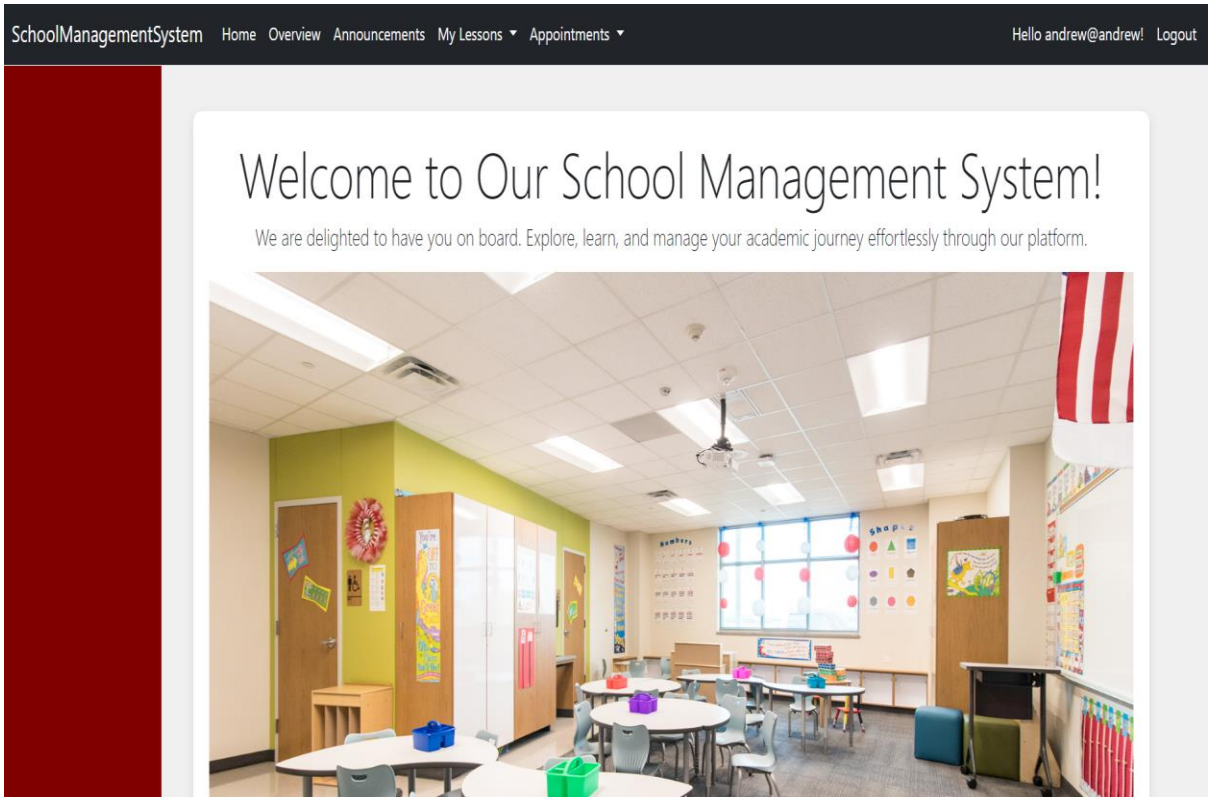
Εικόνα – Αίτημα για ραντεβού



Εικόνα – Ραντεβού(Accepted & Declined)

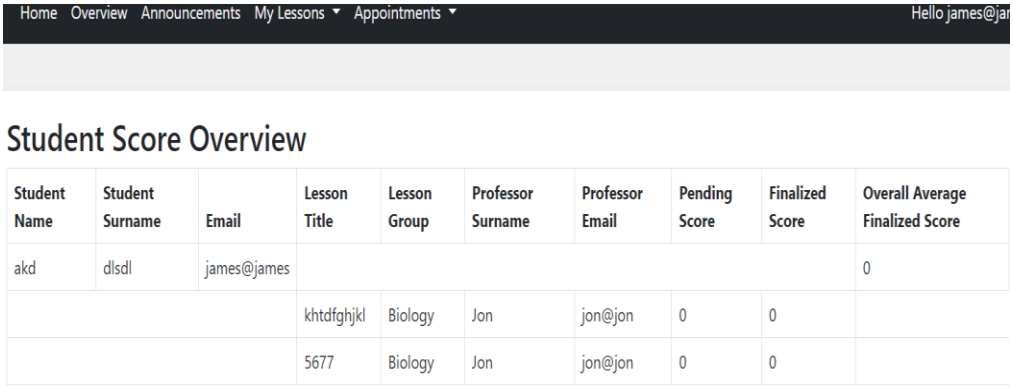
5.3 Students

1) Κάνοντας **Login** ως χρήστης – Μαθητής , με τα αντίστοιχα credentials , βλέπει αυτήν την σελίδα.



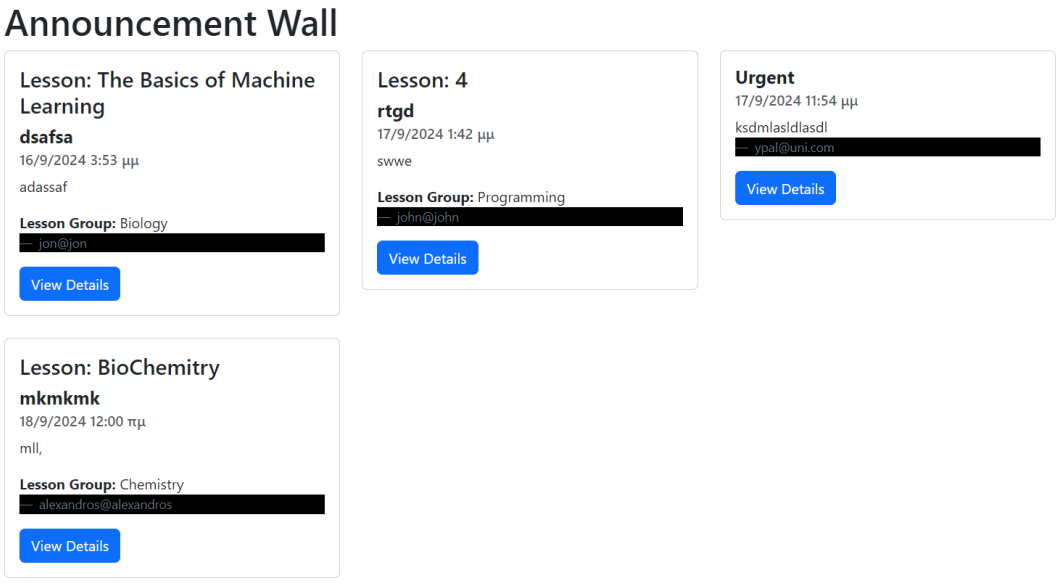
Εικόνα – Student Home Page

2) **Overview :** Πατώντας το tab **Overview**, ο χρήστης μπορεί να δει τη συνολική βαθμολογία ανά μάθημα στο οποίο είναι εγγεγραμμένος, είτε προσωρινή (**Pending**) είτε τελική (**Finalized**), τον μέσο όρο του στα μαθήματα με τελική βαθμολογία, καθώς και τα στοιχεία επικοινωνίας των καθηγητών που είναι υπεύθυνοι για αυτά.



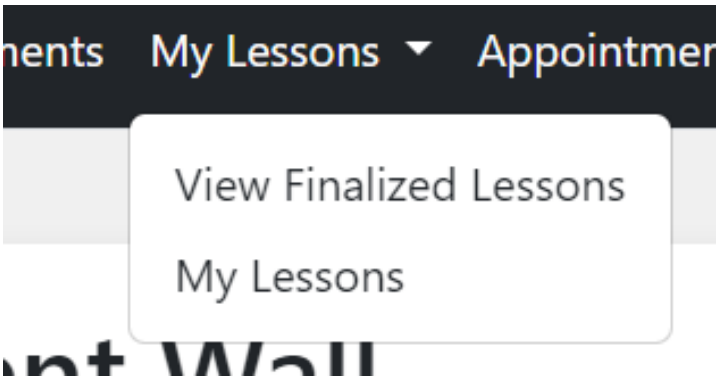
Εικόνα – Πληροφορίες βαθμολογίας

3) **Announcements :** Επιλέγοντας το tab **Announcements**, ο χρήστης μπορεί να δει τις ανακοινώσεις που έχουν δημιουργήσει οι χρήστες Admin και Professors.



Εικόνα – Πίνακας ανακοινώσεων

4) **Lessons :** Επιλέγοντας το tab **Lessons** , μας εμφανίζονται τα λινκ **My Lessons** & **View Finalized Lessons.**



Εικόνα – Dropdown My Lessons

5) **My Lessons :** Επιλέγοντας ο χρήστης το λινκ **My Lessons** , βλέπει την παρακάτω σελίδα.

Εικόνα – Διαθέσιμα μαθήματα

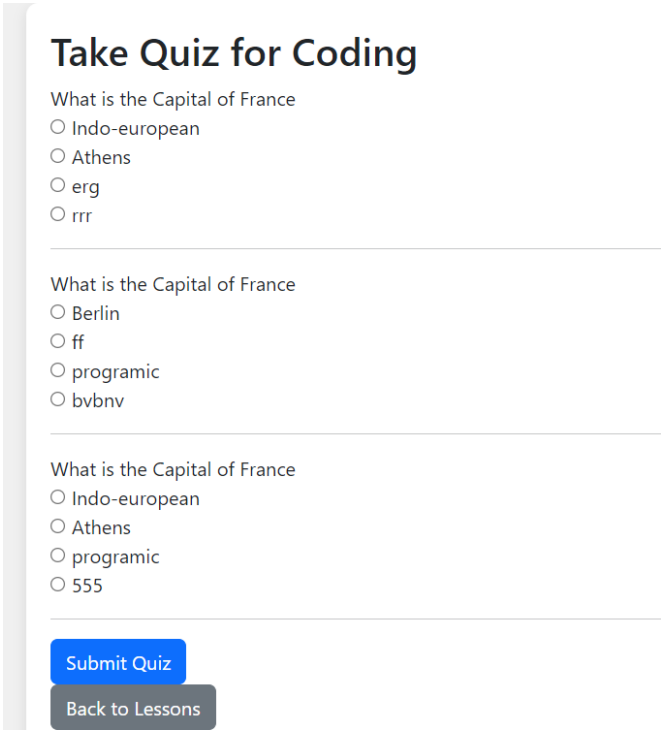
6) **Details :** Πατώντας το λινκ Details , ο χρηστής μαθητής μπορεί να δει γενικές πληροφορίες για το μάθημα που επέλεξε .

Εικόνα – Πληροφορίες Μαθήματος

7) **View Files :** Πατώντας το link **View Files**, ο χρήστης μπορεί να δει τις σημειώσεις που έχει αναρτήσει ο καθηγητής και να τις εξαγάγει (**Extract**).

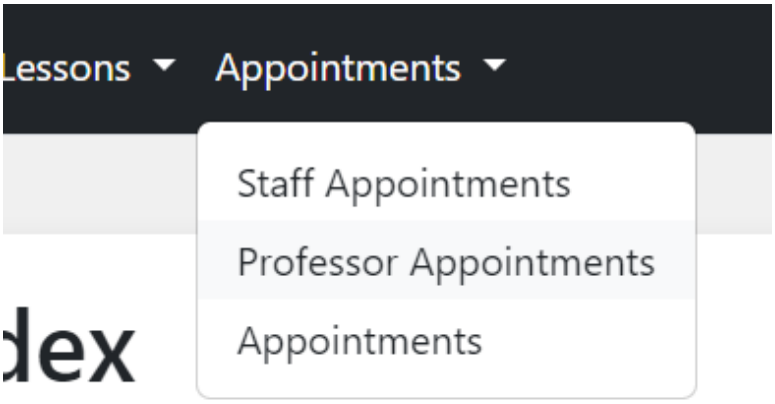
Εικόνα – Ανεβασμένα αρχεία

8) **Start Examination :** Πατώντας το link **Start Examination**, ο μαθητής μπορεί να εξεταστεί στο μάθημά του. Στο συγκεκριμένο παράδειγμα, η εξέταση περιλαμβάνει ερωτήσεις πολλαπλής επιλογής.



Εικόνα – Quiz

9) Appointments : Επιλέγοντας το tab Appointments , εμφανίζεται το παρακάτω dropdown .



Εικόνα – Dropdown Appoinments

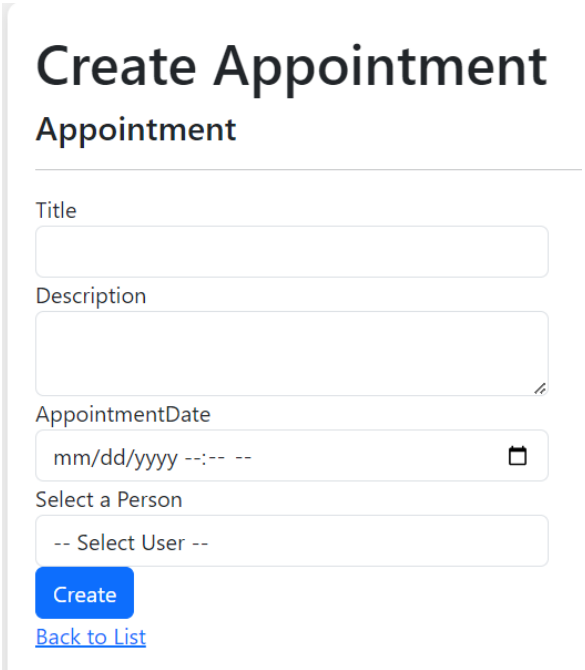
10) **Appoinments :** Επιλέγοντας το λινκ **Appointments** , ο χρήστης μπορεί να δει το status όλων των ραντεβού του (Pending , Accepted , Declined).

Appointments Index

Title	Description	AppointmentDate	CreatedAt	Status	Email	Email
lkl	kmkml	18/9/2024 10:52:00 μμ	16/9/2024 10:52:09 μμ	Accepted	alex@alex	ypal@uni.com
sffs	sfagdhfng	25/9/2024 12:21:00 μμ	17/9/2024 12:21:18 μμ	Declined	alex@alex	
sffs	sfvsd	25/9/2024 12:27:00 μμ	17/9/2024 12:27:13 μμ	Pending	alex@alex	
eee	fffff	25/9/2024 1:44:00 μμ	17/9/2024 1:44:55 μμ	Pending	alex@alex	
uhuh	hbhjhuh	25/9/2024 10:51:00 μμ	17/9/2024 10:51:21 μμ	Accepted	alex@alex	john@john
mdkksd	wdkw kwdwkqji ekfwekji	25/9/2024 12:03:00 πμ	18/9/2024 12:03:26 πμ	Declined	andrew@andrew	
kfokokfew	koaklp	26/9/2024 12:03:00 πμ	18/9/2024 12:03:55 πμ	Accepted	andrew@andrew	alexandros@alexandros
kwdmlkdwmlwqd	mqlkdkwoq wdakdf wkjdkwqkdkqmk	25/9/2024 10:10:00 πμ	23/9/2024 10:11:21 πμ	Accepted	andrew@andrew	ypal@uni.com
ksaflsa	skdmlkdw	27/9/2024 10:11:00 πμ	23/9/2024 10:11:51 πμ	Declined	andrew@andrew	

Εικόνα – Πίνακας ραντεβού

11) Staff & Professor Appointment : Επιλέγοντας ένα από τα λινκ Staff Appointment η Professor Appointment , ο χρήστης μπορεί να συμπληρώσει την φόρμα συμπλήρωσης ραντεβού.



Εικόνα – Φόρμα Create Appointment

Κεφάλαιο 6ο

6 Συμπεράσματα

Από τα δεδομένα που παρουσιάστηκαν σε όλη την έκταση της παρούσας επιστημονικής εργασίας, γίνεται κατανοητό ότι η διαδικτυακή πλατφόρμα ασύγχρονης τηλεκπαίδευσης School Management System αποτελεί μια ολοκληρωμένη λύση για τη διαχείριση της μαθησιακής διαδικασίας. Η ιδέα για την υλοποίησή της προέκυψε από την ανάγκη προώθησης του ψηφιακού μετασχηματισμού στη μαθησιακή διαδικασία και τη βελτίωση της αλληλεπίδρασης μεταξύ καθηγητών και μαθητών. Αν και υπάρχουν ήδη αρκετές πλατφόρμες τηλεκπαίδευσης, οι δυνατότητες και οι λειτουργίες που προσφέρουν συχνά δεν είναι επαρκείς για να καλύψουν τις σύγχρονες ανάγκες, ενώ η παραμετροποίησή τους μπορεί να είναι αρκετά περίπλοκη. Για τον λόγο αυτό, κρίθηκε απαραίτητη η δημιουργία της online εφαρμογής School Management System, η οποία αντιμετωπίζει όλες τις δυσκολίες που αναφέρθηκαν, παρέχοντας πρόσβαση και δυνατότητες τόσο σε διαχειριστές και καθηγητές όσο και σε μαθητές. Όπως φαίνεται στο εγχειρίδιο χρήσης, το περιβάλλον της πλατφόρμας είναι φιλικό και εύκολο στη χρήση. Συνολικά, τα στοιχεία που παρουσιάστηκαν καθιστούν το School Management System μια ολοκληρωμένη και εύχρηστη διαδικτυακή πλατφόρμα για την υποστήριξη της σύγχρονης εκπαίδευσης.

6.1 Οδηγίες εγκατάστασης της εφαρμογής.

1) Απαιτείται η προεγκατάσταση του IDE Visual Studio. Περισσότερα εδώ : https://visualstudio.microsoft.com/downloads/
2) Ανοίγουμε το project μέσω του Visual Studio
3) Στο tab Tools , επιλέγουμε την επιλογή Package Manager Console
4) Αφού ανοίξει η γραμμή εντολών , πληκτρολογούμε την εντολή : update-database
5) Στην εφαρμογή , μέσω της μεθόδου Seed(), έχει γίνει Configure ο χρήστης – Admin με τα εξής credentials :
    - Username : ypal@uni.com
    - Password : Pass123$
6) Κάνουμε είσοδο στην εφαρμογή κάνοντας χρήση των κωδίκων του Admin

7 Βιβλιογραφικές Πηγές

1. **MVC Tutorial**, Διαθέσιμο στην διεύθυνση : https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm

2. **Microsoft.EntityFrameworkCore.SqlServer,** Διαθέσιμο στην διεύθυνση: https://www.nuget.org/packages/Microsoft.EntityFrameworkCore.SqlServer/

3. **UserManager<TUser> Class ,** Διαθέσιμο στην διεύθυνση : https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.identity.usermanager-1?view=aspnetcore-8.0

4. **IFormFile Interface,** Διαθέσιμο στην διεύθυνση : https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.http.iformfile?view=aspnetcore-8.0

5. **Introduction to Identity on ASP.NET Core,** Διαθέσιμο στην διεύθυνση : https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-8.0&tabs=visual-studio

6. **System.Security.Claims ,** Διαθέσιμο στην διεύθυνση : https://learn.microsoft.com/en-us/dotnet/api/system.security.claims?view=net-8.0

7. **Add a model to an ASP.NET Core MVC app,** Διαθέσιμο στην διεύθυνση : https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/adding-model?view=aspnetcore-8.0&tabs=visual-studio

8. **ASP.NET MVC Controller Overview (C#) ,** Διαθέσιμο στην διεύθυνση : https://learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/controllers-and-routing/aspnet-mvc-controllers-overview-cs

9. **Views in ASP.NET Core MVC ,** Διαθέσιμο στην διεύθυνση : https://learn.microsoft.com/en-us/aspnet/core/mvc/views/overview?view=aspnetcore-8.0

10. **Use ViewData and Implement ViewModel Classes ,** Διαθέσιμο στην διεύθυνση : https://learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions-1/nerddinner/use-viewdata-and-implement-viewmodel-classes

11. **Model-View-ViewModel (MVVM),** Διαθέσιμο στην διεύθυνση : https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm

12. **RoleManager<TRole> Class,** Διαθέσιμο στην διεύθυνση : https://learn.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.identity.rolemanager-1?view=aspnetcore-8.0

13. **DbContext Lifetime, Configuration, and Initialization,** Διαθέσιμο στην διεύθυνση : https://learn.microsoft.com/en-us/ef/core/dbcontext-configuration/

14. **Migrations Overview ,** Διαθέσιμο στην διεύθυνση: https://learn.microsoft.com/en-us/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli

15. **.NET migration documentation,** Διαθέσιμο στην διεύθυνση: https://learn.microsoft.com/en-us/dotnet/navigate/migration-guide/

16. **HTML Tutorial ,** Διαθέσιμο στην διεύθυνση: https://www.w3schools.com/html/default.asp

17. **CSS Tutorial ,** Διαθέσιμο στην διεύθυνση: https://www.w3schools.com/css/default.asp

18. **Microsoft.EntityFrameworkCore.Tools,** Διαθέσιμο στην διεύθυνση: https://www.nuget.org/packages/Microsoft.EntityFrameworkCore.Tools

19. **Language Integrated Query (LINQ),** Διαθέσιμο στην διεύθυνση: https://learn.microsoft.com/en-us/dotnet/csharp/linq/

20. **Entity Framework Core,** Διαθέσιμο στην διεύθυνση: https://learn.microsoft.com/en-us/ef/core/

21. **.NET Core 8.0,** Διαθέσιμο στην διεύθυνση: https://dotnet.microsoft.com/en-us/download/dotnet/8.0