

**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

**ΜΕΤΑΠΤΥΧΙΑΚΟ ΔΙΠΛΩΜΑ ΕΙΔΙΚΕΥΣΗΣ (MSc)**

**στην ΑΝΑΠΤΥΞΗ & ΑΣΦΑΛΕΙΑ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ**

**ΔΙΠΛΩΜΑΤΙKH ΕΡΓΑΣΙΑ**

**“ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΟ ΣΥΣΤΗΜΑ ΑΣΚΗΣΕΩΝ”**

**Αναστάσιος Αντωνόπουλος**

**Ρ3312101**

**ΑΘΗΝΑ, ΣΕΠΤΕΜΒΡΙΟΣ 2023**

# Πίνακας Περιεχομένων (Προσωρινός)

[Ευχαριστίες](#_Ευχαριστίες)  
[Περίληψη](#_Περίληψη)  
[Abstract](#_Abstract)

1. [Εισαγωγή](#_Εισαγωγή)
   1. [Σκοπός της εργασίας](#_Σκοπός_της_εργασίας)
   2. [Παρουσίαση του προβλήματος](#_Παρουσίαση_του_προβλήματος)
   3. [Προδιαγραφές – Ανάλυση Απαιτήσεων](#_Προδιαγραφές_–_Ανάλυση)
   4. [Μεθοδολογία ανάπτυξης](#_Μεθοδολογία_ανάπτυξης)
2. [Ανάλυση της εφαρμογής](#_Ανάλυση_της_εφαρμογής)
   1. [Περιγραφή του συστήματος](#_Περιγραφή_του_συστήματος)
   2. [Απαιτήσεις από το σύστημα](#_Απαιτήσεις_από_το)
   3. Ορισμοί ακρώνυμα και συντομογραφίες
   4. Περιπτώσεις Χρήσης
      1. Επισκόπηση μοντέλου περιπτώσεων χρήσης
      2. Οι ενδιαφερόμενοι και οι ανάγκες τους
      3. Actors του συστήματος
      4. Περιγραφές περιπτώσεων χρήσης
   5. Εννοιολογικό Διάγραμμα Κλάσεων
   6. Συμπληρωματικές προδιαγραφές
      1. Απαιτήσεις διεπαφών
      2. Περιορισμοί σχεδίασης και υλοποίησης
      3. Ποιοτικά χαρακτηριστικά
3. Οθόνες Εφαρμογής

Front-> Components, περιγραφή ανά σελίδα κτλπ,

Back -> Controller, Service, Repository pattern. Πλεονεκτήματα, Μειονεκτήματα. Options Pattern, Dependency Injection Pattern, Πως δουλεύει το gmail API, Πως δουλεύει το JWT Authentication

DB -> Schema, Πίνακες

1. Υλοποίηση
   1. Περιγραφή συστατικών
   2. Βάση δεδομένων
   3. Κλάσεις εφαρμογής
   4. Γενικές Πληροφορίες
2. Μελλοντικές επεκτάσεις
3. Βιβλιογραφία

# Πίνακας Περιεχομένων

[Πίνακας Περιεχομένων (Προσωρινός) 2](#_Toc142343173)

[Πίνακας Περιεχομένων 3](#_Toc142343174)

[Πίνακας Εικόνων 5](#_Toc142343175)

[Ευχαριστίες 6](#_Toc142343176)

[Περίληψη 7](#_Toc142343177)

[Abstract 8](#_Toc142343178)

[1. Εισαγωγή 9](#_Toc142343179)

[1.1. Σκοπός της εργασίας 9](#_Toc142343180)

[1.2. Παρουσίαση του προβλήματος 9](#_Toc142343181)

[1.3. Προδιαγραφές – Ανάλυση απαιτήσεων 10](#_Toc142343182)

[1.4. Μεθοδολογία ανάπτυξης 10](#_Toc142343183)

[2. Ανάλυση της εφαρμογής 12](#_Toc142343184)

[2.1. Περιγραφή του συστήματος 12](#_Toc142343185)

[2.2. Απαιτήσεις από το σύστημα 13](#_Toc142343186)

[2.3. Ορισμοί, ακρώνυμα και συντομογραφίες 14](#_Toc142343187)

[2.4. Περιπτώσεις χρήσης 15](#_Toc142343188)

[2.4.1. Επισκόπηση μοντέλου περιπτώσεων χρήσης 15](#_Toc142343189)

[2.4.2. Οι ενδιαφερόμενοι και οι ανάγκες τους 16](#_Toc142343190)

[2.4.3. Actors του συστήματος 16](#_Toc142343191)

[2.4.4. Περιγραφές περιπτώσεων χρήσης 16](#_Toc142343192)

[ΠΧ1. Διαχείριση προφίλ 16](#_Toc142343193)

[ΠΧ2. Διαχείριση τράπεζας θεμάτων 18](#_Toc142343194)

[ΠΧ3. Διαχείριση Τεστ 19](#_Toc142343195)

[ΠΧ4. Επίλυση τεστ 21](#_Toc142343196)

[2.5. Εννοιολογικό Διάγραμμα Κλάσεων 21](#_Toc142343197)

[2.6. Μη λειτουργικές απαιτήσεις 23](#_Toc142343198)

[2.6.1. Απαιτήσεις διεπαφών 23](#_Toc142343199)

[2.6.2. Περιορισμοί σχεδίασης και υλοποίησης 23](#_Toc142343200)

[2.6.3. Ποιοτικά χαρακτηριστικά 23](#_Toc142343201)

[3. Υλοποίηση 25](#_Toc142343202)

[3.1 Βάση Δεδομένων 25](#_Toc142343203)

[3.1.1 Σχήμα βάσης δεδομένων 25](#_Toc142343204)

[3.1.2 SQL Script για την δημιουργία της βάσης 27](#_Toc142343205)

[3.2 Application Tier 27](#_Toc142343206)

[3.2.1 Οργάνωση κώδικα 27](#_Toc142343207)

[3.2.2 Controller, Service, Repository pattern 27](#_Toc142343208)

[3.2.3 Dependency Injection pattern 27](#_Toc142343209)

[3.2.4 Options pattern 27](#_Toc142343210)

[3.2.4 JWT Authentication 27](#_Toc142343211)

[3.2.5 Gmail API 27](#_Toc142343212)

[Παράρτημα Α – SQL Create script 35](#_Toc142343213)

# Πίνακας Εικόνων

[Εικόνα 1: Σχηματική απεικόνιση της 3-Tier Architecture 10](file:///C:\Users\tasso\Desktop\Master_Thesis\Thesis\Master_Thesis_2022-23_v4.docx#_Toc142343214)

[Εικόνα 2: Σχηματική απεικόνιση των περιπτώσεων χρήσης 15](file:///C:\Users\tasso\Desktop\Master_Thesis\Thesis\Master_Thesis_2022-23_v4.docx#_Toc142343215)

[Εικόνα 3: Εννοιολογικό διάγραμμα κλάσεων 22](#_Toc142343216)

[Εικόνα 4: Σχηματική απεικόνιση βάσης δεδομένων 26](file:///C:\Users\tasso\Desktop\Master_Thesis\Thesis\Master_Thesis_2022-23_v4.docx#_Toc142343217)

# Ευχαριστίες

# Περίληψη

# Abstract

# Εισαγωγή

## Σκοπός της εργασίας

Σκοπός της παρούσας διπλωματικής εργασίας είναι η μελέτη, ο σχεδιασμός και η υλοποίηση μιας διαδικτυακής εφαρμογής, η οποία θα λειτουργήσει ως ένα αυτοματοποιημένο σύστημα ασκήσεων για την αξιολόγηση των γνώσεων σε δυνητικούς υποψηφίους για εργασία ή εκπαιδευόμενους. Τα τελευταία χρόνια η ζήτηση για την κάλυψη νέων θέσεων εργασίας έχει αυξηθεί ραγδαία και ειδικότερα κατά την περίοδο της πανδημίας του COVID-19. Τότε υπήρξε μια μεγάλη μετακίνηση εργαζομένων, αφού πολλοί ήταν εκείνοι που επέλεξαν να κάνουν μια αλλαγή στην καριέρα τους χάρη στην τηλεργασία και το υβριδικό μοντέλο εργασίας. Συνεπώς, γίνεται κατανοητό ότι υπάρχει η ανάγκη για ένα εργαλείο το οποίο μπορεί να αυτοματοποιήσει κάποιες από τις διαδικασίες αξιολόγησης των γνώσεων των υποψηφίων και να κάνει πιο γρήγορη και αξιόπιστη την διαδικασία πρόσληψης. Επιπλέον, η χρήση της εφαρμογής δεν περιορίζεται μόνο στις διαδικασίες πρόσληψης, αλλά μπορεί να χρησιμοποιηθεί και σε άλλες μορφές αξιολόγησης, όπως για παράδειγμα για εκπαιδευτικούς σκοπούς. Συνοπτικά, οι βασικοί στόχοι/λειτουργίες είναι:

* Ανάπτυξη πλατφόρμας με σύγχρονες τεχνολογίες που εξασφαλίζουν ασφάλεια, εγκυρότητα και ευχρηστία.
* Δημιουργία δραστηριοτήτων διαφορετικών βαθμών δυσκολίας και γνωστικών αντικειμένων.
* Σύστημα τράπεζας ασκήσεων/δραστηριοτήτων.
* Ενημέρωση του εξεταστή και του εξεταζόμενου για τις αναθέσεις και τα αποτελέσματα σε πραγματικό χρόνο.

Συμπερασματικά, η σχεδίαση και η υλοποίηση μιας φιλικής και εύχρηστης διαδικτυακής εφαρμογής προς οποιονδήποτε χρήστη και από οποιαδήποτε συσκευή αποτελεί βασικό στόχο.

## Παρουσίαση του προβλήματος

Η ζήτηση για την κάλυψη νέων θέσεων εργασίας έχει γνωρίσει μεγάλη αύξηση σε διάφορα επαγγέλματα. Επιπλέον, η πανδημία του COVID-19 εισήγαγε νέους τρόπους εργασίας και εκπαίδευσης, τόσο εξ αποστάσεως όσο και με υβριδικά μοντέλα. Ο παραδοσιακός τρόπος αξιολογήσεων δεν εξυπηρετεί. Για παράδειγμα οι άνθρωποι που κάνουν τις συνεντεύξεις ή οι εξεταστές έχουν περιορισμένο χρόνο για την κάλυψη των αναγκών και το πέρας των αξιολογήσεων. Επιπλέον, η διαδικασία αυτή είναι αρκετά χρονοβόρα και κοστοβόρα, αφού πολλές φορές πρέπει να επικοινωνήσουν με καθένα από τους εξεταζόμενους ξεχωριστά αλλά και για να διορθώσουν κάθε μια από τις απαντήσεις. Τέλος είναι αρκετά δύσκολο να υπάρχει και να οργανωθεί μια τράπεζα θεμάτων η οποία έχει ταξινομημένα ανά γνωστικό αντικείμενο και δυσκολία θέματα που θα πρέπει να κληθούν να απαντήσουν οι εξεταζόμενοι.

Συμπερασματικά, η υλοποίηση μιας εφαρμογής η οποία θα λειτουργήσει ως ένα αυτοματοποιημένο σύστημα ασκήσεων μπορεί να βελτιώσει πολλά από τα παραπάνω προβλήματα. Αρχικά, με την άνοδο της εξ αποστάσεως εκπαίδευσης και εργασίας, μια εφαρμογή που δημιουργεί τεστ γνώσης πολλαπλών επιλογών επιτρέπει στους εξεταστές να διαχειρίζονται εύκολα τις αξιολογήσεις και στους μαθητές και στους εργαζόμενους να έχουν πρόσβαση και να τις ολοκληρώσουν εξ αποστάσεως ασύγχρονα, ανάλογα με τα ωράρια τους. Επίσης, μια τέτοια εφαρμογή μπορεί να χρησιμοποιηθεί για τη δημιουργία εξατομικευμένων αξιολογήσεων για μαθητές ή υπαλλήλους με βάση τις ατομικές μαθησιακές ανάγκες και την πρόοδό τους. Επιπλέον, αυτοματοποιεί τη διαδικασία βαθμολόγησης των αποτελεσμάτων, εξοικονομώντας χρόνο και προσπάθεια από τους εξεταστές. Ακόμη, προσφέρει ανατροφοδότηση στους εξεταζόμενους σε πραγματικό χρόνο, ώστε να γνωρίζουν την βαθμολογία τους, και παράλληλα να αντιλαμβάνονται ελλείψεις τους, ώστε να μπορούν να βελτιωθούν. Τέλος, όλα τα παραπάνω μπορούν να εξοικονομήσουν και χρήματα σε σύγκριση με τη δημιουργία και τη βαθμολόγηση με μη αυτοματοποιημένες μεθόδους.

## Προδιαγραφές – Ανάλυση απαιτήσεων

Το πρώτο στάδιο της σχεδίασης ενός συστήματος είναι η καταγραφή και η ανάλυση των απαιτήσεων. Αυτές διακρίνονται στις Λειτουργικές και Μη-Λειτουργικές απαιτήσεις.

Οι Λειτουργικές απαιτήσεις αφορούν την περιγραφή των υπηρεσιών/λειτουργιών που θα παρέχει το σύστημα.

Οι Μη-Λειτουργικές απαιτήσεις σχετίζονται με τους περιορισμούς στην σχεδίαση και την υλοποίηση του συστήματος, δηλαδή έχουν να κάνουν με ποιοτικά χαρακτηριστικά όπως είναι η Ασφάλεια, η Απόδοση, η Διαθεσιμότητα, η Ευελιξία κ.λπ.

## Μεθοδολογία ανάπτυξης

Diagram

Description automatically generatedΓια το πρώτο στάδιο υλοποίησης του συστήματος που είναι αυτό της ανάλυσης και του σχεδιασμού θα χρησιμοποιηθεί η UML που είναι μια πρότυπη γλώσσα σχεδίασης και αποτύπωσης προτύπων. Η UML αποτελεί έναν τρόπο για την οπτικοποίηση της συμπεριφοράς και της δομής ενός συστήματος και είναι ο καλύτερος τρόπος για την επικοινωνία των ατόμων που συμμετέχουν στην ανάπτυξη ενός συστήματος.   
  
Στόχος για την υλοποίηση της εφαρμογής είναι η χρήση εργαλείων ανοιχτού κώδικα. Επιπλέον η ανάπτυξη της εφαρμογής θα βασιστεί στην αρχιτεκτονική 3-tier. Δηλαδή, θα αποτελείται από τo επίπεδο της Παρουσίασης (Presentation tier), το επίπεδο της Εφαρμογής (Application tier) και το επίπεδο των Δεδομένων (Data tier). Στο επίπεδο της Παρουσίασης, ένας front-end web server παρουσιάζει στον τελικό χρήστη στατικό ή δυναμικό περιεχόμενο και πιθανώς κάποιο περιεχόμενο αποθηκευμένο σε μία cache μνήμη. Στο επίπεδο της Εφαρμογής γίνεται όλη η διαδικασία της επεξεργασίας και της εφαρμογής των κανόνων που ορίζει η επιχειρηματική ανάλυση και εκτελείται σε έναν application server. Τέλος στο επίπεδο των Δεδομένων περιλαμβάνεται η αποθήκευση των δεδομένων αλλά και το σύστημα διαχείρισης της βάσης δεδομένων που μας παρέχει πρόσβαση σε αυτά.

Εικόνα : Σχηματική απεικόνιση της 3-Tier Architecture

Για την ανάπτυξη του επιπέδου Presentation tier θα χρησιμοποιηθεί η χρήση της JavaScript βιβλιοθήκης React η οποία δημιουργήθηκε και συντηρείται από την Meta (πρώην Facebook) και μια μεγάλη κοινότητα που αποτελείται από ανεξάρτητους προγραμματιστές και εταιρείες. Επιπλέον, για την συγγραφή της θα γίνει χρήση του ολοκληρωμένου περιβάλλοντος ανάπτυξης (IDE) Visual Studio Code της Microsoft. Κάποιοι από τους λόγους που επιλέχθηκε η βιβλιοθήκη React για την ανάπτυξη του Presentation tier είναι οι παρακάτω:

* Είναι δημοφιλής και την εμπιστεύονται μεγάλες εταιρείες όπως η Meta, η Airbnb, το Netflix, το Uber κλπ. Αυτό έχει σαν αποτέλεσμα να υπάρχει μια μεγάλη κοινότητα προγραμματιστών που βοηθούν με το υλικό τους την ευκολότερη μάθηση.
* Επιτρέπει την δημιουργία components τα οποία μπορούν να επαναχρησιμοποιηθούν κάνοντας ευκολότερη την ανάπτυξη πολύπλοκων εφαρμογών.
* Κάνει χρήση ενός εικονικού DOM το οποίο βελτιώνει την απόδοση αφού κάνει re-rendering μόνο τα components τα οποία έχουν αλλαγές.

Για την ανάπτυξη του επιπέδου Application tier θα χρησιμοποιηθεί το software framework του .ΝΕΤ 6 με χρήση της γλώσσας προγραμματισμού C# καθώς και το ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) Visual Studio 2022 της Microsoft. Κάποιοι από τους λόγους που οδήγησαν στην επιλογή του .NET 6 είναι οι παρακάτω:

* Προσφέρει μεγάλη ευελιξία αφού υποστηρίζει την εκτέλεση σε πολλές πλατφόρμες συμπεριλαμβανομένων των Windows, macOS και Linux.
* Έχει βελτιωμένη απόδοση αφού προσφέρει γρήγορους χρόνους εκτέλεσης, βελτιωμένο garbage collection και λιγότερες απαιτήσεις σε μνήμη.

Τέλος για την ανάπτυξη του Data tier επιλέχθηκε ο SQL Server 2022 όπου είναι μια σχεσιακή βάση δεδομένων η οποία έχει αναπτυχθεί από τη Microsoft. Τα πλεονεκτήματα της χρήσης του SQL Server είναι τα παρακάτω:

* Πλήρης συμβατότητα με το .NET 6. Το γεγονός αυτό κάνει ευκολότερη την αλληλεπίδραση και την διαχείριση των δεδομένων, που είναι αποθηκευμένα στην βάση δεδομένων, από την εφαρμογή .NET.
* Προσφέρει σε άριστο βαθμό Απόδοση(Performance), Επεκτασιμότητα(Scalability) και Ασφάλεια(Security) που είναι βασικά στοιχεία για την ανάπτυξη της εφαρμογής που περιγράφεται.
* Έχει μειωμένο κόστος αφού προσφέρει πολλές διαφορετικές εκδόσεις, συμπεριλαμβανόμενης μιας δωρεάν, open-sourced έκδοσης, η οποία και θα χρησιμοποιηθεί.

# Ανάλυση της εφαρμογής

## Περιγραφή του συστήματος

Το σύστημα που θα υλοποιήσουμε θα στοχεύει στην χρήση του από δύο κατηγορίες χρηστών. Η μία κατηγορία είναι αυτή του εξεταστή και η άλλη είναι αυτή του εξεταζόμενου/εκπαιδευόμενου. Η παραπάνω διάκριση γίνεται κατά την δημιουργία ενός λογαριασμού στην εφαρμογή. Από τον νέο χρήστη θα του ζητείται να συμπληρώσει το ονοματεπώνυμο του, το email του, τον προσωπικό του κωδικό, την εταιρεία ή το ίδρυμα όπου εργάζεται και τον τύπο του προφίλ που επιθυμεί να δημιουργήσει. Μετά την εγγραφή του, ο χρήστης θα μπορεί να συνδεθεί στη εφαρμογή συμπληρώνοντας το email του και τον κωδικό που έχει ορίσει κατά την διάρκεια της εγγραφής. Παρακάτω περιγράφονται αναλυτικά οι λειτουργίες που θα προσφέρονται από το σύστημα για τις δύο κατηγορίες χρηστών.

Στους εξεταστές το σύστημα θα προσφέρει την δυνατότητα να αποθηκεύσουν ερωτήσεις με το πρότυπο των ερωτήσεων πολλαπλής επιλογής. Εκτός από την ερώτηση και τις πιθανές απαντήσεις, θα μπορεί να σημειώνει το αντικείμενο που αφορά η ερώτηση καθώς και την δυσκολία της ανάμεσα σε 4 επίπεδα (Εύκολο, Μεσαίο, Δύσκολο, Πολύ Δύσκολο). Στην συνέχεια, θα μπορεί να αποθηκεύσει τις ερωτήσεις σε τράπεζες θεμάτων, που έχουν ήδη δημιουργηθεί για να μπορεί μελλοντικά να τις αναζητήσει και να τις επεξεργαστεί. Επιπλέον, ο εξεταστής, θα μπορεί να ζητά από το σύστημα να του δημιουργήσει Τεστ που περιλαμβάνουν σετ ερωτήσεων με αυτοματοποιημένο τρόπο. Για την δημιουργία των τεστ το σύστημα θα ζητά από τον εξεταστή να επιλέξει κάποια κριτήρια όπου με βάση αυτά θα γίνει η δημιουργία των τεστ. Τα κριτήρια είναι το θέμα, η δυσκολία και το πλήθος των ερωτήσεων. Τα τεστ θα τα αναθέτει σε εξεταζόμενους προσθέτοντας το email του εξεταζόμενου σε αντίστοιχο πεδίο. Επιπλέον, θα έχει την δυνατότητα να αποθηκεύσει τα τεστ, έτσι ώστε να μπορεί να τα αναθέσει εκ νέου σε κάποιον εξεταζόμενο στο μέλλον. Τέλος, ο εξεταστής θα ενημερώνεται σε πραγματικό χρόνο μέσω email για την ολοκλήρωση του τεστ από κάποιον εξεταζόμενο και θα μπορεί να βλέπει για κάθε επιλεγμένο τεστ συγκεντρωτικά και αναλυτικά την επίδοση αυτών που το έχουν απαντήσει.

Όσον αφορά στον εξεταζόμενο, το σύστημα θα τον ενημερώνει με email, αλλά και μέσω της εφαρμογής ότι του έχει ανατεθεί κάποιο τεστ το οποίο πρέπει να απαντήσει. Ο εξεταζόμενος θα συνδέεται με το χρήστη του στην εφαρμογή και θα του εμφανίζονται όλα τα τεστ τα οποία του έχουν ανατεθεί, καθώς και από ποιον εξεταστή. Επιλέγοντας το τεστ που επιθυμεί θα μπορεί να το ολοκληρώσει απαντώντας στις ερωτήσεις. Με το πέρας της αξιολόγησης θα του εμφανίζονται αυτόματα τα αποτελέσματα της επίδοσης του τα οποία θα περιλαμβάνουν την βαθμολογία του, την κατάταξή του ανάμεσα στους υπόλοιπους που έχουν ολοκληρώσει το συγκεκριμένο τεστ σε μορφή ποσοστού και θα μπορεί να βλέπει αναλυτικά τις ερωτήσεις που απάντησε λανθασμένα.

Από τη παραπάνω περιγραφή, διακρίνονται εννοιολογικά οι εξής οντότητες:

1. Ο χρήστης του συστήματος, που διακρίνεται σε δύο κατηγορίες τον **Εξεταστή** και τον **Εξεταζόμενο**
2. Οι **Ερωτήσεις πολλαπλής επιλογής**, οι οποίες διακρίνονται από τα χαρακτηριστικά: Θεματολογία, Δυσκολία (Εύκολο, Μέτριο, Δύσκολο, Πολύ Δύσκολο), Περιγραφή, Πιθανές απαντήσεις, Σωστή απάντηση.
3. Το **Τεστ**, το οποίο το δημιουργεί ο εξεταστής αυτόματα με βάση κάποια χαρακτηριστικά και το επιλύει ο εξεταζόμενος.
4. Οι **Τράπεζες θεμάτων** που είναι λίστες από αποθηκευμένες **Κατηγορίες** μαζί με τις **Υποκατηγορίες** τους.
5. Την **Αξιολόγηση**, όπου περιέχει τα αποτελέσματα των απαντήσεων από τους **Εξεταζόμενους**

## Απαιτήσεις από το σύστημα

Με βάση την παραπάνω περιγραφή του συστήματος θα ακολουθήσει η αναλυτική καταγραφή των απαιτήσεων από το σύστημα.

|  |  |
| --- | --- |
| **R1** | Το σύστημα θα περιλαμβάνει μενού με τις παρακάτω επιλογές:   1. Αρχική 2. Τεστ (ορατή μόνο όταν έχει γίνει σύνδεση με λογαριασμό) 3. Τράπεζες Θεμάτων (ορατή μόνο όταν έχει γίνει σύνδεση με λογαριασμό Εξεταστή)    1. Κατηγορίες    2. Υποκατηγορίες    3. Ερωτήσεις 4. Λογαριασμός    1. Σύνδεση    2. Εγγραφή    3. Πληροφορίες λογαριασμού (ορατή μόνο όταν έχει γίνει σύνδεση με λογαριασμό)    4. Αποσύνδεση (ορατή μόνο όταν έχει γίνει σύνδεση με λογαριασμό) |
| **R2** | H οθόνη «**Αρχική»** θα περιλαμβάνει γενικές πληροφορίες σχετικά με το τι προσφέρει η εφαρμογή ανάλογα τον τύπο χρήστη. Θα είναι η πρώτη οθόνη η οποία θα βλέπει ο χρήστης όταν θα εισέρχεται στην εφαρμογή. |
| **R3** | Η οθόνη **«Τεστ»** στους εξεταζόμενους θα εμφανίζει την λίστα των τεστ τα οποία τους έχουν ανατεθεί. Σε όσα έχουν ήδη απαντηθεί θα εμφανίζει επιπλέον την βαθμολογία καθώς και όταν το επιλέγουν θα τους εμφανίζονται οι λύσεις. Τα Τεστ τα οποία είναι προς επίλυση θα μπορεί να τα επιλέξει και να ξεκινήσει τη διαδικασία επίλυσης τους.  Για τους εξεταστές, η οθόνη, θα εμφανίζει τη λίστα των τεστ που έχουν δημιουργήσει. Όταν θα επιλέγουν κάποιο τεστ θα μπορούν να δουν λίστα των εξεταζόμενων που έχουν αναθέσει το τεστ μαζί με την βαθμολογία τους αλλά και τη δυνατότητα για εκ νέου ανάθεση σε κάποιον νέο εξεταζόμενο. Επιπλέον, θα δίνεται η επιλογή να μπορούν να ακυρώσουν την ανάθεση ενός τεστ, αλλά και να μπορούν να δημιουργήσουν και να αναθέσουν κάποιο νέο τεστ. |
| **R4** | Οι οθόνες που ανήκουν στην κατηγορία **«Τράπεζες θεμάτων»** θα είναι ορατές μόνο όταν κάποιος έχει συνδεθεί με λογαριασμό Εξεταστή. Ο Εξεταστής θα μπορεί να βλέπει σε λίστες τις αποθηκευμένες Κατηγορίες, Υποκατηγορίες και Ερωτήσεις. Θα περιλαμβάνονται κατάλληλα κουμπιά έτσι ώστε αν θελήσει να μπορεί να επεξεργαστεί ή να διαγράψει κάποια ερώτηση αλλά και να φιλτράρει τις αποθηκευμένες λίστες. |
| **R5** | Η οθόνη **«Κατηγορίες»** θα είναι ορατή μόνο σε λογαριασμούς Εξεταστή. Σε αυτή την οθόνη ο χρήστης θα μπορεί να προσθέσει στο σύστημα μια νέα Κατηγορία συμπληρώνοντας μια φόρμα που περιέχει τα χαρακτηριστικά του τίτλου και της περιγραφής. Επιπλέον θα μπορεί να διαγράψει ή να επεξεργαστεί κάποια ήδη υπάρχουσα κατηγορία. |
| **R6** | Η οθόνη **«Υποκατηγορίες»** περιλαμβάνει τις υποκατηγορίες όπου έχει αποθηκεύσει κάποιος χρήστης. Ο χρήστης θα μπορεί να προσθέσει στο σύστημα μια νέα Υποκατηγορία συμπληρώνοντας μια φόρμα που περιέχει τα χαρακτηριστικά του τίτλου, της περιγραφής και της κατηγορίας που ανήκει. Επιπλέον θα μπορεί να διαγράψει ή να επεξεργαστεί κάποια ήδη υπάρχουσα υποκατηγορία. Τέλος θα μπορεί να φιλτράρει τα αποτελέσματα με βάση το φίλτρο της κατηγορίας που ανήκουν οι υποκατηγορίες. |
| **R7** | Η οθόνη **«Ερωτήσεις»** θα περιλαμβάνει μια λίστα με τις αποθηκευμένες ερωτήσεις στο σύστημα. Θα υπάρχει η δυνατότητα φιλτραρίσματος με βάση την κατηγορία και την υποκατηγορία. Επιπλέον, θα υπάρχει η δυνατότητα προσθήκης νέας ερώτησης όπου ο χρήστης θα συμπληρώνει μια φόρμα με όλα τα χαρακτηριστικά. Τέλος, σε κάθε ερώτηση θα δίνεται η δυνατότητα επεξεργασίας ή διαγραφής. |
| **R8** | Η οθόνη **«Σύνδεση»** θα παρέχει στον χρήστη την δυνατότητα να κάνει σύνδεση στην εφαρμογή συμπληρώνοντας μια φόρμα με βάση το email του ή το username του και τον κωδικό πρόσβασης του. |
| **R9** | Η οθόνη «Εγγραφή» θα παρέχει στον χρήστη την δυνατότητα να συμπληρώσει την φόρμα εγγραφής στην εφαρμογή και να δημιουργεί τον λογαριασμό του. |
| **R10** | Η οθόνη «Πληροφορίες λογαριασμού» επιτρέπει στον χρήστη να κάνει μεταβολή κάποιων στοιχείων του. |

## Ορισμοί, ακρώνυμα και συντομογραφίες

Παρακάτω ακολουθεί η αναλυτική περιγραφή και καταγραφή των εννοιών που περιλαμβάνονται στο σύστημα.

|  |  |
| --- | --- |
| Έννοια | Επεξήγηση |
| Εξεταστής | Ο εξεταστής περιγράφεται από τα εξής: Ονοματεπώνυμο, email, τηλέφωνο επικοινωνίας, ίδρυμα στο οποίο ανήκει. |
| Εξεταζόμενος | Ο εξεταζόμενος περιγράφεται από τα εξής: Ονοματεπώνυμο, email, τηλέφωνο επικοινωνίας |
| Ερώτηση | Ερώτηση του τύπου πολλαπλής επιλογής. Είναι αποθηκευμένη στην Αποθήκη θεμάτων και την έχει εισάγει κάποιος Εξεταστής. Χαρακτηρίζεται από τα εξής: Θεματολογία, Δυσκολία (Εύκολο, Μέτριο, Δύσκολο, Πολύ Δύσκολο), Περιγραφή, Πιθανές απαντήσεις, Σωστή απάντηση, Αποθήκες θεμάτων που ανήκει. |
| Τράπεζα θεμάτων | Περιέχει αποθηκευμένες ερωτήσεις. Περιγράφεται από τα εξής: Τίτλο, Περιγραφή, Ερωτήσεις |
| Τεστ | Περιέχει πολλές ερωτήσεις. Ανατίθεται σε Εξεταζόμενους για να τα επιλύσουν και δημιουργείται αυτόματα από τους Εξεταστές.  Περιγράφεται από τα εξής: Τίτλος, Περιγραφή, Δυσκολία, Πλήθος ερωτήσεων, Θεματολογία |
| Αξιολόγηση/Ανάθεση | Περιγράφει την συσχέτιση της ανάθεσης και επίλυσης ενός Τεστ από έναν Εξεταστή σε έναν εξεταζόμενο. Περιγράφεται από τα εξής: Βαθμολογία, Ημερομηνία Επίλυσης |
| Ίδρυμα/Εταιρεία | Είναι το εκπαιδευτικό Ίδρυμα ή η Εταιρεία στην οποία ανήκει ένας Εξεταστής. |
| Προφίλ | Περιέχει τα στοιχεία του προφίλ ενός χρήστη. Περιγράφεται από τα εξής: Όνομα, Επώνυμο, Email, Τηλέφωνο επικοινωνίας, Κωδικός πρόσβασης, Εταιρεία/Ίδρυμα. |

## Περιπτώσεις χρήσης

### Εικόνα που περιέχει διάγραμμα, κείμενο, γραμμή Περιγραφή που δημιουργήθηκε αυτόματαΕπισκόπηση μοντέλου περιπτώσεων χρήσης

Εικόνα : Σχηματική απεικόνιση των περιπτώσεων χρήσης

|  |  |
| --- | --- |
| Περίπτωση χρήσης | Περιγραφή |
| ΠΧ1 Διαχείριση προφίλ | Η περίπτωση χρήσης κατά την οποία οι χρήστες (Εξεταστής, Εξεταζόμενος) του συστήματος διαχειρίζονται το προφίλ τους. |
| ΠΧ2 Διαχείριση τράπεζας θεμάτων | Η περίπτωση χρήσης στην οποία ο Εξεταστής δημιουργεί ή διαγράφει μια ερώτηση στο σύστημα. |
| ΠΧ3 Διαχείριση τεστ | Η περίπτωση χρήσης στην οποία ο Εξεταστής δημιουργεί, διαγράφει και αναθέτει ένα τεστ σε Εξεταζόμενους. |
| ΠΧ4 Επίλυση τεστ | Η περίπτωση χρήσης όπου ο Εξεταζόμενος επιλύει το Τεστ που του έχει ανατεθεί |

### 2.4.2. Οι ενδιαφερόμενοι και οι ανάγκες τους

|  |  |
| --- | --- |
| Ενδιαφερόμενος | Ανάγκες |
| Εξεταστής | * Να δημιουργεί και να αποθηκεύει ερωτήσεις στο σύστημα σε συγκεκριμένες τράπεζες θεμάτων. * Να δημιουργεί τεστ με αυτόματο τρόπο και να τα αναθέτει σε Εξεταζόμενους. * Να γνωρίζει την βαθμολογία που έχουν επιτύχει οι Εξεταζόμενοι που έχουν επιλύσει τα τεστ. |
| Εξεταζόμενος | * Επίλυση των τεστ που του έχουν ανατεθεί. * Να γνωρίζει τις σωστές απαντήσεις και τα λάθη του. * Να γνωρίζει την βαθμολογία που πέτυχε με την επίλυση του τεστ. |

### 2.4.3. Actors του συστήματος

|  |  |  |
| --- | --- | --- |
| Actor | Στόχοι | Ενδιαφερόμενοι |
| Εξεταστής | * Δημιουργία ερωτήσεων * Δημιουργία τεστ * Διαχείριση τράπεζας θεμάτων * Διαχείριση τεστ * Διαχείριση προφίλ * Προβολή αποτελεσμάτων στις επιλύσεις των εξεταζόμενων | Εξεταζόμενος |
| Εξεταζόμενος | * Επίλυση τεστ * Διαχείριση προφίλ | Εξεταστής |

### 2.4.4. Περιγραφές περιπτώσεων χρήσης

#### ΠΧ1. Διαχείριση προφίλ

**Πρωτεύων Actor:** Εξεταστής, Εξεταζόμενος

**Ενδιαφερόμενοι:** Εξεταστής, Εξεταζόμενος

**Προϋποθέσεις**: Ο Εξεταστής/Εξεταζόμενος έχει εγγραφεί και έχει ταυτοποιηθεί στο σύστημα.

**Βασική Ροή:**

Α) Επεξεργασία στοιχείων προφίλ

1. Ο χρήστης επιλέγει από το κεντρικό μενού το κουμπί «Λογαριασμός».
2. Στην συνέχεια το σύστημα εμφανίζει μια λίστα με επιλογές από την οποία ο χρήστης επιλέγει την επιλογή «Πληροφορίες λογαριασμού».
3. Το σύστημα του εμφανίζει τις πληροφορίες σχετικά με το προφίλ του προ-συμπληρωμένες στην φόρμα επεξεργασίας. Σε αυτή θα μπορεί μόνο να επεξεργαστεί το email του, το τηλέφωνο επικοινωνίας καθώς και τον κωδικό πρόσβασης.
4. Ο χρήστης συμπληρώνει τα στοιχεία που θέλει να μεταβάλει και πατάει το κουμπί «Ενημέρωση Στοιχείων».
5. Το σύστημα ενημερώνει τα στοιχεία του και εμφανίζει στον χρήστη μήνυμα επιτυχίας.

Β) Σύνδεση

1. Ο χρήστης επιλέγει από το κεντρικό μενού το κουμπί «Λογαριασμός».
2. Στην συνέχεια το σύστημα εμφανίζει μια λίστα με επιλογές από την οποία ο χρήστης επιλέγει την επιλογή «Σύνδεση».
3. Το σύστημα του εμφανίζει μια φόρμα για να συμπληρώσει τα απαραίτητα στοιχεία ώστε να συνδεθεί.
4. Ο χρήστης συμπληρώνει το username ή το email του καθώς και το κωδικό πρόσβασης και στην συνέχεια πατάει το κουμπί «Σύνδεση»
5. Το σύστημα τον ενημερώνει με μήνυμα ότι έχει συνδεθεί επιτυχώς.

5α. Ο χρήστης έχει εισάγει κάποιο από τα στοιχεία του λάθος.  
5β. Το σύστημα τον ενημερώνει με αντίστοιχο μήνυμα.

Γ) Αποσύνδεση

1. Ο χρήστης επιλέγει από το κεντρικό μενού το κουμπί «Λογαριασμός».
2. Στην συνέχεια το σύστημα εμφανίζει μια λίστα με επιλογές από την οποία ο χρήστης επιλέγει την επιλογή «Αποσύνδεση».
3. Το σύστημα του εμφανίζει μια οθόνη που ενημερώνει τον χρήστη ότι πρόκειται να κάνει αποσύνδεση από την εφαρμογή.
4. Ο χρήστης πατάει το κουμπί «Αποσύνδεση»
5. Το σύστημα αποσυνδέει τον χρήστη από την εφαρμογή και τον μεταφέρει αυτόματα στην αρχική οθόνη.

Δ) Εγγραφή

1. Ο χρήστης επιλέγει από το κεντρικό μενού το κουμπί «Λογαριασμός».
2. Στην συνέχεια το σύστημα εμφανίζει μια λίστα με επιλογές από την οποία ο χρήστης επιλέγει την επιλογή «Εγγραφή».
3. Το σύστημα εμφανίζει μια φόρμα έτσι ώστε ο χρήστης να συμπληρώσει το email, το username, το όνομα, το επώνυμο, το κινητό τηλέφωνο, τον τύπο χρήστη που επιθυμεί, την εταιρεία/ίδρυμα που ανήκει καθώς και τον κωδικό πρόσβασης.
4. Ο χρήστης συμπληρώνει τα απαραίτητα στοιχεία και επιλέγει το κουμπί «Εγγραφή»  
   4α. Ο χρήστης δεν συμπληρώνει κάποιο από τα απαραίτητα στοιχεία.  
   4β. Το σύστημα τον ενημερώνει με κατάλληλο μήνυμα σφάλματος.
5. Το σύστημα ενημερώνει τον χρήστη ότι έχει εγγραφεί επιτυχώς.

#### ΠΧ2. Διαχείριση τράπεζας θεμάτων

**Πρωτεύων Actor:** Εξεταστής

**Ενδιαφερόμενοι:** Εξεταζόμενος

**Προϋποθέσεις**: Ο Εξεταστής έχει εγγραφεί στο σύστημα και έχει ταυτοποιηθεί.

**Βασική Ροή:**

Α) Δημιουργία ερώτησης

* 1. Ο Εξεταστής συμπληρώνει την φόρμα εισαγωγής με όλα τα απαραίτητα πεδία (Θεματολογία, Δυσκολία, Περιγραφή, Πιθανές απαντήσεις, Σωστή απάντηση, Αποθήκες θεμάτων που ανήκει).
  2. Το σύστημα αποθηκεύει την αίτηση και ενημερώνει με κατάλληλο μήνυμα τον Εξεταστή.

**Εναλλακτική Ροή:**

**1α.** Δεν έχει γίνει εισαγωγή κάποιου από τα υποχρεωτικά στοιχεία.

1. Το σύστημα εμφανίζει κατάλληλο μήνυμα σφάλματος.
2. Η ΠΧ παραμένει στο βήμα 1 της βασικής ροής.

Β) Διαγραφή ερώτησης

1. Ο Εξεταστής επιλέγει μια ερώτηση από την λίστα και πατάει την επιλογή διαγραφή.
2. Το σύστημα διαγράφει την ερώτηση και ενημερώνει με κατάλληλο μήνυμα τον Εξεταστή.

Γ) Δημιουργία τράπεζας θεμάτων

1. Ο Εξεταστής επιλέγει το κουμπί προσθήκης νέας τράπεζας θεμάτων.
2. Το σύστημα του ζητά να συμπληρώσει τον Τίτλο, την Περιγραφή καθώς και να επιλέξει τις ερωτήσεις που θα περιέχονται στην τράπεζα θεμάτων.
3. Ο Εξεταστής συμπληρώνει τα απαραίτητα στοιχεία και πατάει το κουμπί «Αποθήκευση».

*3α. Δεν έχουν συμπληρωθεί τα απαραίτητα στοιχεία*

* 1. Το σύστημα εμφανίζει κατάλληλο μήνυμα σφάλματος.
  2. Η ΠΧ επιστρέφει στο βήμα 2 της βασικής ροής.

1. Το σύστημα δημιουργεί την τράπεζα θεμάτων και ενημερώνει με κατάλληλο μήνυμα τον Εξεταστή.

Δ) Ενημέρωση τράπεζας θεμάτων

1. Ο Εξεταστής επιλέγει μια από τις αποθηκευμένες τράπεζες θεμάτων.
2. Το σύστημα εμφανίζει τις λεπτομέρειες της επιλεγμένης τράπεζας θεμάτων (Τίτλο, Περιγραφή και Ερωτήσεις που περιέχονται).
3. Ο Εξεταστής επιλέγει το κουμπί «Επεξεργασία».
4. Το σύστημα τον αφήνει να επεξεργαστεί τον Τίτλο, την Περιγραφή καθώς και να μπορεί να αφαιρέσει κάποια από τις Ερωτήσεις. Επίσης, του εμφανίζει ένα κουμπί «Προσθήκη Ερώτησης» για να μπορέσει να προσθέσει κάποια ερώτηση από κάποια λίστα.
5. Ο Εξεταστής κάνει τις ενέργειες που επιθυμεί και πατάει το κουμπί «Ενημέρωση».

*5α.**Δεν έχουν συμπληρωθεί τα απαραίτητα στοιχεία*

* 1. Το σύστημα εμφανίζει κατάλληλο μήνυμα σφάλματος.
  2. Η ΠΧ επιστρέφει στο βήμα 2 της βασικής ροής.

1. Το σύστημα αποθηκεύει τις αλλαγές και ενημερώνει με κατάλληλο μήνυμα τον Εξεταστή.

Ε) Διαγραφή τράπεζας θεμάτων

1. Ο Εξεταστής επιλέγει το κουμπί «Επιλογή» από την κεντρική οθόνη.
2. Το σύστημα του εμφανίζει την δυνατότητα επιλογής της τράπεζας θεμάτων και το κουμπί «Διαγραφή» ως ανενεργό.
3. Ο Εξεταστής επιλέγει τις τράπεζα θεμάτων που επιθυμεί να διαγράψει.
4. Το σύστημα μετατρέπει το κουμπί σε ενεργό.
5. Ο Εξεταστής πατάει το κουμπί «Διαγραφή»
6. Το σύστημα διαγράφει την επιλεγμένη τράπεζα θεμάτων και ενημερώνει με κατάλληλο μήνυμα τον Εξεταστή.

#### ΠΧ3. Διαχείριση Τεστ

**Πρωτεύων Actor:** Εξεταστής

**Ενδιαφερόμενοι:** Εξεταζόμενος

**Προϋποθέσεις**: Ο Εξεταστής έχει εγγραφεί στο σύστημα και έχει ταυτοποιηθεί.

**Βασική Ροή:**

Α) Δημιουργία τεστ

1. Ο Εξεταστής επιλέγει το κουμπί «Δημιουργία νέου τεστ» από την κεντρική οθόνη «Τεστ».
2. Το σύστημα του ζητά να συμπληρώσει τα χαρακτηριστικά (Τίτλος, Περιγραφή, Πλήθος ερωτήσεων, Θεματολογία) με βάση τα οποία θα παραχθεί το Τεστ.
3. Ο Εξεταστής συμπληρώνει τα απαραίτητα στοιχεία και πατάει το κουμπί «Δημιουργία».

*3α. Δεν έχουν συμπληρωθεί τα απαραίτητα στοιχεία*

* 1. Το σύστημα εμφανίζει κατάλληλο μήνυμα σφάλματος.
  2. Η ΠΧ επιστρέφει στο βήμα 2 της βασικής ροής.

1. Το σύστημα δημιουργεί και αποθηκεύει το Τεστ που δημιούργησε.
2. Το σύστημα ζητά από τον Εξεταστή αν επιθυμεί εκείνη την στιγμή να αναθέσει σε κάποιον το Τεστ που δημιουργήθηκε.
3. Ο Εξεταστής προαιρετικά επιλέγει σε ποια άτομα θα αναθέσει το Τεστ που δημιούργησε είτε επιλέγοντας από τις Επαφές του είτε συμπληρώνοντας το email τους και πατάει το κουμπί «Ανάθεση».
4. Το σύστημα αναθέτει προς επίλυση το Τεστ και στέλνει ειδοποίηση μέσω email στους Εξεταζόμενους.

Β) Ανάθεση τεστ

1. Ο Εξεταστής επιλέγει από την λίστα κάποιο από τα Τεστ που έχει δημιουργήσει από την κεντρική οθόνη «Τεστ».
2. Το σύστημα του εμφανίζει τις πληροφορίες σχετικά με το συγκεκριμένο Τεστ.
3. Ο Εξεταστής επιλέγει το κουμπί «Ανάθεση».
4. Το σύστημα ζητά από τον Εξεταστή να αναθέσει σε κάποιον το Τεστ που έχει επιλέξει.
5. Ο Εξεταστής επιλέγει σε ποια άτομα θα αναθέσει το Τεστ που επέλεξε είτε επιλέγοντας από τις Επαφές του είτε συμπληρώνοντας το email τους και πατάει το κουμπί «Ανάθεση».
6. Το σύστημα αναθέτει προς επίλυση το Τεστ και στέλνει ειδοποίηση μέσω email στους Εξεταζόμενους.

Γ) Προβολή αποτελεσμάτων τεστ

1. Ο Εξεταστής επιλέγει από την λίστα κάποιο από τα Τεστ που έχει δημιουργήσει από την κεντρική οθόνη «Τεστ».
2. Το σύστημα του εμφανίζει τις πληροφορίες σχετικά με το συγκεκριμένο Τεστ.
3. Ο Εξεταστής επιλέγει το κουμπί «Προβολή αποτελεσμάτων».
4. Το σύστημα του εμφανίζει αναλυτικά τις επιδόσεις του κάθε Εξεταζόμενου καθώς και ποιοι είναι ακόμα σε κατάσταση αναμονής αφού δεν το έχουν επιλύσει ακόμα. Επιπλέον του εμφανίζει τα αποτελέσματα σε μορφή γραφήματος.

Δ) Ενημέρωση τεστ

1. Ο Εξεταστής επιλέγει από την λίστα κάποιο από τα Τεστ που έχει δημιουργήσει από την κεντρική οθόνη «Τεστ».
2. Το σύστημα του εμφανίζει τις πληροφορίες σχετικά με το συγκεκριμένο Τεστ.
3. Ο Εξεταστής επιλέγει το κουμπί «Επεξεργασία».
4. Το σύστημα τον αφήνει να επεξεργαστεί τον Τίτλο, την Περιγραφή καθώς και να μπορεί να αφαιρέσει κάποιον Εξεταζόμενο από την ανάθεση του Τεστ.
5. Ο Εξεταστής πατάει το κουμπί «Αποθήκευση».
6. Το σύστημα αποθηκεύει τις αλλαγές και ενημερώνει με κατάλληλο μήνυμα τον Εξεταστή.

Ε) Διαγραφή Τεστ

1. Ο Εξεταστής επιλέγει από την λίστα κάποιο από τα Τεστ που έχει δημιουργήσει από την κεντρική οθόνη «Τεστ».
2. Το σύστημα του εμφανίζει τις πληροφορίες σχετικά με το συγκεκριμένο Τεστ.
3. Ο Εξεταστής επιλέγει το κουμπί «Διαγραφή».
4. Το σύστημα τον ενημερώνει για την ενέργεια που πρόκειται να κάνει και ζητά για επιβεβαίωση.
5. Ο Εξεταστής επιλέγει το κουμπί «Επιβεβαίωση».

*5α. Ο Εξεταστής επιλέγει το κουμπί «Ακύρωση».*

* 1. Η ΠΧ επιστρέφει στο βήμα 2 της βασικής ροής.

1. Το σύστημα διαγράφει το συγκεκριμένο Τεστ και ενημερώνει με κατάλληλο μήνυμα τον Εξεταστή.

#### ΠΧ4. Επίλυση τεστ

**Πρωτεύων Actor:** Εξεταζόμενος

**Ενδιαφερόμενοι:** Εξεταστής

**Προϋποθέσεις**: Ο Εξεταζόμενος έχει εγγραφεί και έχει ταυτοποιηθεί στο σύστημα. Επιπλέον του έχει ανατεθεί κάποιο Τεστ προς επίλυση

**Βασική Ροή:**

Α) Επίλυση τεστ

1. Ο Εξεταζόμενος επιλέγει το κουμπί «Τεστ» από το κεντρικό μενού.
2. Το σύστημα του εμφανίζει την λίστα των Τεστ που του έχουν ανατεθεί.
3. Ο Εξεταζόμενος επιλέγει από την λίστα ποιο Τεστ θέλει να επιλύσει.
4. Το σύστημα του εμφανίζει τις πληροφορίες του Τεστ και το κουμπί «Επίλυση» ως ανενεργό.
5. Ο Εξεταζόμενος επιλέγει το κουμπί «Επίλυση».
6. Το σύστημα του εμφανίζει κάθε μια από τις ερωτήσεις, τις πιθανές απαντήσεις για να επιλέξει την σωστή και το κουμπί «Επόμενο» ως ανενεργό.
7. Ο Εξεταζόμενος επιλέγει την απάντηση στην ερώτηση.
8. Το σύστημα εμφανίζει το κουμπί «Επόμενο» ως ενεργό.
9. Ο Εξεταζόμενος επιλέγει το κουμπί «Επόμενο».
10. Το σύστημα τον μεταφέρει στην επόμενη ερώτηση. Μόλις απαντηθούν όλες οι ερωτήσεις το σύστημα εμφανίζει το κουμπί «Υποβολή».
11. Ο Εξεταζόμενος επιλέγει το κουμπί «Υποβολή».
12. Το σύστημα αποθηκεύει τις απαντήσεις του χρήστη. Ενημερώνει με email τον Εξεταστή. Εμφανίζει στον Εξεταζόμενο την βαθμολογία που συγκέντρωσε.

Β) Προβολή αποτελεσμάτων

1. Ο Εξεταζόμενος επιλέγει το κουμπί «Τεστ» από το κεντρικό μενού.
2. Το σύστημα του εμφανίζει την λίστα των Τεστ που του έχουν ανατεθεί.
3. Ο Εξεταζόμενος επιλέγει από την λίστα ποιο Τεστ θέλει να δει τα αποτελέσματα του.
4. Το σύστημα του εμφανίζει την βαθμολογία που συγκέντρωσε και τις απαντήσεις του μαζί με τα λάθη του.

## Εννοιολογικό Διάγραμμα Κλάσεων

Τα διαγράμματα Μοντέλου Οντοτήτων - Συσχετίσεων χρησιμοποιούνται στην ανάπτυξη λογισμικού με σκοπό την γραφική αποτύπωση των βασικών εννοιών και των σχέσεων τους σε ένα σύστημα. Είναι κάτι που βοηθά στην κατανόηση των απαιτήσεων ενός συστήματος και στο σχεδιασμό μιας λύσης η οποία θα πληροί όλες τις προδιαγραφές που έχουν καταγραφεί.

Στα διαγράμματα αυτά, οι κύριες οντότητες ή έννοιες του πεδίου αντιπροσωπεύονται ως κλάσεις, με τα χαρακτηριστικά και τις λειτουργίες τους. Οι σχέσεις μεταξύ αυτών των κλάσεων απεικονίζονται με συσχετισμούς. Παρακάτω ακολουθεί το διάγραμμα Μοντέλου Οντοτήτων – Συσχετίσεων για το σύστημα που έχουμε σχεδιάσει.

Εικόνα που περιέχει κείμενο, διάγραμμα, παράλληλα, Σχέδιο

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα : Εννοιολογικό διάγραμμα κλάσεων

## Μη λειτουργικές απαιτήσεις

### Απαιτήσεις διεπαφών

|  |  |
| --- | --- |
| Διεπαφή | Λεπτομέρειες |
| Χρήστη | * Όλες οι διεπαφές χρήστη θα είναι διαδικτυακές. |
| Υλικού | * Το σύστημα δεν χρησιμοποιεί διεπαφές υλικού. |
| Επικοινωνίας | * Το σύστημα θα χρησιμοποιεί το Gmail API της Google για την αποστολή email. |
| Λογισμικού | * Το σύστημα δεν χρησιμοποιεί διεπαφές λογισμικού. |

### 2.6.2. Περιορισμοί σχεδίασης και υλοποίησης

Οι περιορισμοί που θα πρέπει να ληφθούν υπόψιν κατά την διαδικασία σχεδίασης και υλοποίησης είναι οι εξής:

1. Το σύστημα θα πρέπει να υποστηρίζει τα σχεσιακά συστήματα διαχείρισης βάσεων δεδομένων του SQL Server.
2. Η υλοποίηση του front-end συστήματος θα γίνει με χρήση της JavaScript βιβλιοθήκης React.
3. Η υλοποίηση του back-end συστήματος θα γίνει με χρήση του software framework του .ΝΕΤ 6 με χρήση της γλώσσας προγραμματισμού C#.
4. Η αυθεντικοποίηση των χρηστών θα γίνει με χρήση του εργαλείου αυθεντικοποίησης JWT Security Token της Microsoft.
5. Η αποστολή των ενημερωτικών email θα γίνεται με την χρήση της βιβλιοθήκης Gmail API της Google.
6. Το σύστημα θα μπορεί να εκτελείται σε υπολογιστές και κινητές συσκευές.

### 2.6.3. Ποιοτικά χαρακτηριστικά

|  |  |
| --- | --- |
| Χαρακτηριστικά | Λεπτομέρειες |
| Απόδοση | * Το σύστημα θα πρέπει να υποστηρίζει έως και 500 ταυτόχρονους χρήστες. * Η απόκριση της βάσης δεδομένων δεν θα πρέπει να ξεπερνά τα 5 δευτερόλεπτα. |
| Διαθεσιμότητα | * Το σύστημα θα είναι διαθέσιμο όλο το 24ωρο. * Ο χρόνος της μη διαθεσιμότητας δεν θα πρέπει να ξεπερνά το 5%. |
| Ασφάλεια | * Όλες οι λειτουργίες του συστήματος θα πραγματοποιούνται με διαδικασία αυθεντικοποίησης. * Το σύστημα δεν θα αποθηκεύει κωδικούς λογαριασμών χρηστών. * Θα εγγυάται η ασφάλεια και ακεραιότητα των δεδομένων των χρηστών. |
| Ευελιξία | * Ενδεχόμενη προσαρμογή του συστήματος σε ένα νέο σύστημα διαχείρισης βάσης δεδομένων δεν θα πρέπει να ξεπερνά τις 20 μέρες. |
| Ευχρηστία | * Προϋπόθεση για τον τελικό χρήστη είναι να γνωρίζει τη χρήση των Web Βrowsers. * Ο χρήστης θα πρέπει να είναι σε θέση να μπορεί να χρησιμοποιεί το σύστημα χωρίς την ανάγκη κάποιας εκπαίδευσης. |

# Υλοποίηση

## 3.1 Βάση Δεδομένων

### 3.1.1 Σχήμα βάσης δεδομένων

Με βάση τα παραπάνω που έχουμε περιγράψει στο στάδιο της ανάλυσης της εφαρμογής μας προχωράμε στον σχεδιασμό της βάσης δεδομένων μας. Στην βάση δεδομένων θα αποθηκεύονται όλα τα χρήσιμα χαρακτηριστικά του συστήματος και πάνω σε αυτή θα βασιστούν στην συνέχεια οι υλοποιήσεις του Application Tier και του Presentation Tier. Στην βάση δεδομένων θα αποθηκεύονται όλα τα δεδομένα που θα παράγονται από τους χρήστες του συστήματος και στην συνέχεια θα αντλούνται από αυτή.

Παρακάτω παρουσιάζεται το σχήμα της βάσης δεδομένων μας το οποίο φυσικά έχει βασιστεί στο εννοιολογικό διάγραμμα κλάσεων που περιγράψαμε στην ενότητα 2.5. Σε αυτό μπορούμε να διακρίνουμε τους πίνακες της βάσης μας μαζί με τις στήλες τους. Επιπλέον βλέπουμε τις συσχετίσεις μεταξύ αυτών των πινάκων και πως συνδέονται μεταξύ τους και τέλος βλέπουμε τα πρωτεύοντα και δευτερεύοντα κλειδιά.

Επιπλέον ιδιαίτερα χρήσιμο είναι και το SQL script της δημιουργίας της βάσης δεδομένων μας. Εκτελώντας το στο διαχειριστικό περιβάλλον του SQL Server μπορούμε να δημιουργήσουμε εκ νέου την βάση δεδομένων μας. Επιπλέον αν το επιθυμούμε μπορούμε να συμπεριλάβουμε σε αυτό και τα δεδομένα που υπάρχουν σε αυτή. Παρακάτω ακολουθεί το αντίστοιχο SQL script για την βάση μας μαζί με τα δεδομένα που περιλαμβάνει.

Εικόνα που περιέχει κείμενο, διάγραμμα, Σχέδιο, παράλληλα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα : Σχηματική απεικόνιση βάσης δεδομένων

### 3.1.2 SQL Script για την δημιουργία της βάσης

Απαραίτητο συστατικό για την λειτουργία της εφαρμογής μας είναι η εγκατάσταση της βάσης στον SQL Server. Για να γίνει αυτό χρειάζεται να τρέξουμε το SQL Script το οποίο βρίσκεται στο Παράρτημα Α.

## 3.2 Application Tier

### 3.2.1 Οργάνωση κώδικα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματαΤο solution της Application Tier εφαρμογής έχει οργανωθεί σε ξεχωριστά project ανάλογα με την έννοια τους έτσι ώστε να μπορεί η εφαρμογή να είναι εύκολα διαχειρισίμη και επεκτάσιμη στην πάροδο του χρόνου.

Συγκεκριμένα βλέπουμε ότι έχει οργανωθεί στα εξής projects.

|  |  |
| --- | --- |
| Project | Περιεχόμενα |
| ApplicationService | Περιέχει την υλοποίηση των διαφόρων services τα οποία υλοποιούν την επιχειρησιακή λογική (business logic) της εφαρμογής μας. |
| Controllers | Περιέχει την υλοποίηση των κλάσεων που είναι υπεύθυνες για να χειρίζονται τα εισερχόμενα αιτήματα από την εφαρμογή του Presentation Tier. Δηλαδή λειτουργεί σαν μεσάζων ανάμεσα στον client και το service layer της εφαρμογής. |
| Interfaces | Περιέχει τις διεπαφές των application services και των repository. |
| Mapper | Περιέχει την αντιστοίχιση των κλάσεων του Database Context σε αντικείμενα DTO έτσι ώστε να γίνεται αυτόματα το mapping. |
| Repositories | Περιέχει την υλοποίηση των service που είναι υπεύθυνα για την επικοινωνία με την βάση δεδομένων. |
| Security | Περιέχει την υλοποίηση για τους μηχανισμούς της κρυπτογράφησης του κωδικού πρόσβασης του χρήστη και του ελέγχου αν αυτός ο κωδικός είναι έγκυρος. |
| Types | Περιέχει όλες τις κλάσεις που αφορούν τους τύπους δεδομένων της εφαρμογής. Αυτές περιλαμβάνουν και το database context, τους τύπους DTO που επιστρέφονται στην εφαρμογή του client αλλά και τους τύπους δεδομένων που έρχονται στα εισερχόμενα αιτήματα από τον client (requests). |

### 3.2.2 Controller, Service, Repository pattern

Here's a brief overview of each component in the pattern:

1. Controller: The controller is responsible for handling incoming requests from clients (e.g., web browsers or mobile apps) and invoking the appropriate business logic. It acts as an intermediary between the client and the service layer. The controller is often associated with a specific route or endpoint and is responsible for parsing request parameters, validating input, and returning responses to the client.
2. Service: The service layer contains the business logic of the application. It encapsulates the core functionality of the application, such as data manipulation, business rules, and computations. The service layer provides a higher-level, more abstract interface to the controllers, enabling them to perform complex operations without getting into low-level details.
3. Repository: The repository layer is responsible for data access and persistence. It provides an abstraction over the data storage mechanisms (e.g., databases, external APIs) and allows the service layer to interact with data without being tightly coupled to the underlying storage implementation. This separation makes it easier to switch between different data sources or data storage technologies without affecting the service layer.

Usage of the Controller-Service-Repository pattern:

1. Decoupling and Maintainability: By separating concerns into distinct layers, the Controller-Service-Repository pattern reduces dependencies between different parts of the application. This makes it easier to maintain and extend the application as each layer can be modified independently.
2. Testability: The pattern promotes testability by facilitating the creation of unit tests for each layer. Controllers can be tested using mock requests and responses, services can be tested independently from data access, and repositories can be tested against various data storage scenarios.
3. Code Reusability: With a clear separation of concerns, the code becomes more modular, allowing for better code reusability. The service layer, containing business logic, can be used across multiple controllers, and the repository layer can be utilized with different services or applications.
4. Scalability: The pattern makes it easier to scale the application by allowing for the introduction of additional services or instances to handle increased traffic or processing load.
5. Security: The separation of concerns helps enforce security measures effectively. For example, input validation and access control can be implemented in the controller, while data access restrictions can be enforced within the repository layer.

Overall, the Controller-Service-Repository pattern helps in building well-organized and maintainable applications by promoting separation of concerns, which is crucial for the long-term success of software projects.

The Controller-Service-Repository pattern is a widely used architectural pattern that aims to separate concerns and improve the maintainability and scalability of software applications. Below are the advantages and disadvantages of using the Controller-Service-Repository pattern:

Advantages:

1. Separation of Concerns: The pattern promotes a clear separation of concerns, allowing developers to isolate and manage different functionalities of the application independently. Controllers handle the user interface and routing, services encapsulate business logic, and repositories manage data access, making it easier to understand and modify specific parts of the system.
2. Modularity: By breaking the application into distinct layers, each with its specific responsibilities, the code becomes more modular. This modular design facilitates code reuse and allows developers to work on individual components without affecting others, leading to better maintainability.
3. Testability: The separation of concerns makes it easier to write unit tests for each layer independently. Controllers can be tested for handling different types of requests, services can be tested for their business logic, and repositories can be tested for their data access capabilities.
4. Scalability: The pattern enables scalability by allowing different layers to be scaled independently. For example, if an application experiences increased traffic, additional instances of the service layer can be deployed without affecting the controller or repository layers.
5. Flexibility in Data Storage: The repository layer provides an abstraction over data storage, allowing the application to work with various data sources without altering the service layer. This flexibility makes it simpler to switch between different data storage solutions or databases.
6. Encapsulation of Business Logic: With the business logic contained within the service layer, the controllers become lean and focused on handling user input and invoking appropriate service methods. This promotes cleaner and more maintainable code.

Disadvantages:

1. Complexity: Introducing multiple layers and abstractions can add complexity to the application, especially for smaller projects or simple applications. Overengineering with the pattern may not be beneficial in such cases.
2. Overhead: The pattern may introduce some additional overhead due to the need for communication between the layers. This could affect the application's performance slightly, though the benefits of modularity and maintainability often outweigh this drawback.
3. Learning Curve: Developers who are new to the pattern may require some time to understand the structure and relationships between the different layers. This learning curve could be a challenge for teams transitioning from different architectural approaches.
4. Indirection: The pattern introduces indirection between layers, which may make it harder to trace the flow of data and logic through the application. This can be mitigated by proper documentation and code comments.
5. Over-Reliance on the Pattern: Blindly applying the pattern to all scenarios might not be suitable for every application. It's essential to evaluate the specific requirements of the project and decide whether the pattern's overhead justifies its usage.
6. Code Duplication: In some cases, the pattern might lead to code duplication, especially if the application requires similar logic in different service methods. Proper refactoring and abstraction are essential to avoid this issue.

Overall, while the Controller-Service-Repository pattern offers numerous advantages in terms of maintainability, scalability, and testability, its implementation should be considered thoughtfully, especially for smaller projects or simple applications where the added complexity might not be necessary.

### 3.2.3 UnitOfWork pattern

The Unit of Work (UnitOfWork) pattern is a behavioral design pattern commonly used in software development, particularly when working with data persistence, such as databases or ORMs (Object-Relational Mapping). The pattern is designed to manage transactions and ensure data integrity within a single logical unit of work.

Description of the Unit of Work Pattern:

The Unit of Work pattern essentially groups multiple related data operations into a single unit, ensuring that all these operations either succeed together as a whole or fail together as a whole. It acts as an abstraction over the data access layer and is responsible for tracking changes to entities, managing transactions, and coordinating with the underlying data storage mechanism.

The primary components of the Unit of Work pattern are as follows:

1. Unit of Work: This component is responsible for tracking changes to entities and orchestrating transactions. It exposes methods to start, commit, or rollback a transaction. The Unit of Work keeps track of all entities that have been modified, added, or deleted within the current unit of work.
2. Repository: The Unit of Work pattern often works in conjunction with the Repository pattern, which provides an abstraction over data access. The Repository pattern is responsible for CRUD (Create, Read, Update, Delete) operations on specific entity types. The Unit of Work coordinates with repositories to persist changes to the data store.

Advantages of the Unit of Work Pattern:

1. Transaction Management: The Unit of Work pattern ensures that all related data operations are treated as a single transaction. This means that if any part of the unit of work fails, the entire unit is rolled back, preserving data integrity and consistency.
2. Reduces Database Calls: By grouping related data operations into a single unit, the pattern reduces the number of database calls. This can lead to improved performance and reduced overhead in database interactions.
3. Simplifies Business Logic: The pattern allows business logic to be decoupled from data access concerns. Business logic can be encapsulated within the Unit of Work and repositories, making it easier to manage and maintain.
4. Enhances Testability: The Unit of Work pattern facilitates better testability by allowing the creation of mock or in-memory implementations of the data access layer during testing. This enables developers to write unit tests that do not depend on a specific database instance.

Disadvantages of the Unit of Work Pattern:

1. Complexity: Implementing the Unit of Work pattern can introduce some complexity to the application, particularly when dealing with large and complex data models. The overhead of managing transactions and tracking changes might not be suitable for simpler applications.
2. Learning Curve: Developers who are new to the Unit of Work pattern may require some time to understand its concepts and integration with repositories and data access layers.
3. Performance Overhead: While the pattern reduces the number of database calls, it may lead to increased memory consumption and processing overhead, especially if the unit of work becomes too large.
4. Potential for Large Units of Work: If not carefully managed, the unit of work can become large, including too many data operations. This can impact the overall efficiency and performance of the application.

In conclusion, the Unit of Work pattern provides an effective solution for managing transactions and data integrity in applications dealing with data persistence. However, its implementation should be carefully considered, especially for projects with specific performance requirements or relatively simple data access needs.

### 3.2.3 Dependency Injection pattern

Dependency Injection (DI) is a software design pattern used in .NET 6 and other modern programming frameworks. It allows developers to implement loose coupling between classes and components by providing dependencies to a class from an external source, rather than the class creating its dependencies. In .NET 6, DI is an integral part of the framework and is available through the built-in Dependency Injection container.

Description of Dependency Injection Pattern in .NET 6:

In the Dependency Injection pattern, classes do not instantiate their dependencies directly but instead rely on an external entity (Dependency Injection container) to provide them. The container manages the instantiation and lifetime of dependencies, ensuring that they are created and managed efficiently. .NET 6 provides a built-in DI container, making it easy to utilize DI in your applications.

Advantages of Dependency Injection Pattern in .NET 6:

1. Loose Coupling: DI promotes loose coupling between components, making the codebase more maintainable and flexible. Classes depend on abstractions (interfaces) rather than concrete implementations, allowing you to change or extend the behavior of the application without modifying existing code.
2. Testability: DI enhances testability by enabling the injection of mock or fake dependencies during unit testing. This allows you to isolate the code being tested and focus on testing individual components in isolation.
3. Separation of Concerns: By moving the responsibility of creating and managing dependencies outside of the class, the DI pattern improves the separation of concerns. Classes focus on their primary responsibilities, making the codebase more organized and easier to maintain.
4. Modularity: The DI pattern encourages modularity in your application, making it easier to compose different parts of the system. This facilitates code reuse and promotes better project organization.
5. Scalability: The DI pattern simplifies the process of adding or replacing components, making the application more scalable and adaptable to changing requirements.
6. Centralized Configuration: .NET 6's DI container provides centralized configuration for managing dependencies, making it easier to understand and modify the composition of the application.

Disadvantages of Dependency Injection Pattern in .NET 6:

1. Learning Curve: Implementing DI in .NET 6 may require developers to learn and understand the concepts of the DI pattern and the specific syntax and usage of the DI container.
2. Overhead: While the DI pattern improves maintainability, it may introduce some overhead due to the additional complexity of dependency resolution and instantiation.
3. Configuration Complexity: In larger applications, managing dependency configurations in the DI container may become complex and challenging to maintain.
4. Debugging Complexity: Troubleshooting issues related to dependency resolution or configuration errors may be more complex than direct instantiation of dependencies.

In summary, the Dependency Injection pattern in .NET 6 provides numerous advantages, such as loose coupling, testability, and modularity, contributing to better software design and development. However, developers should be aware of potential challenges and complexities in configuration and debugging when working with DI in larger and more complex applications.

### 3.2.4 JWT Authentication

Microsoft JWT (JSON Web Token) Authentication in .NET 6 is a built-in authentication mechanism provided by the .NET framework for securing web applications and APIs. It allows developers to authenticate and authorize users by issuing and validating JWTs, which are digitally signed tokens containing user information and claims.

How JWT Authentication Works in .NET 6:

1. Token Issuance: When a user successfully logs in, the server generates a JWT containing the user's claims (e.g., username, role) and signs it using a secret key. The token is then returned to the client.
2. Token Storage: The client typically stores the received JWT, either in memory or a persistent storage mechanism like cookies or local storage.
3. Token Inclusion in Requests: For each subsequent request to protected resources (e.g., API endpoints), the client includes the JWT in the request header, typically as a "Authorization: Bearer {token}" header.
4. Token Validation: On the server-side, the JWT middleware in .NET 6 validates the token's authenticity, ensuring it has not been tampered with and is still valid. If the token is valid, the server allows access to the requested resource; otherwise, it returns a 401 Unauthorized status code.

Advantages of Microsoft JWT Authentication in .NET 6:

1. Stateless: JWT authentication is stateless, meaning the server does not need to store session information, making it more scalable and suitable for distributed applications.
2. Improved Security: JWTs are digitally signed, ensuring the integrity of user information and preventing tampering. Additionally, sensitive information like passwords is not included in the token, enhancing security.
3. Decentralized Authentication: JWTs enable a decentralized authentication approach, allowing microservices and APIs to validate tokens independently, reducing the need for centralized authentication servers.
4. Performance: JWT authentication is lightweight and can be more performant than traditional session-based authentication, as it eliminates the need for frequent database queries to check session state.
5. Cross-Origin Authentication: JWTs can be used for cross-origin authentication, allowing a user authenticated in one application to access resources in another application without re-authenticating.

Disadvantages of Microsoft JWT Authentication in .NET 6:

1. Token Size: JWTs can become large if they contain many claims, leading to increased network overhead. It is crucial to keep the token size manageable to maintain performance.
2. Token Expiry Management: JWTs have an expiration time, and managing token expiration and renewal can be challenging, especially in distributed systems.
3. Single Sign-On Limitations: While JWTs facilitate cross-origin authentication, they might have limitations in more complex single sign-on (SSO) scenarios, which may require additional mechanisms like OAuth.
4. Key Management: Proper key management is essential for JWT security. The secret key used for signing the tokens must be securely stored and rotated regularly to prevent unauthorized access.

In conclusion, Microsoft JWT Authentication in .NET 6 offers a secure and scalable authentication mechanism for web applications and APIs. Its stateless nature and decentralized approach make it suitable for modern distributed systems. However, developers should carefully manage token size, expiration, and key security to ensure the best possible implementation.

### 3.2.5 Gmail API

Πως δουλεύει το gmail API

## Παράρτημα Α – SQL Create script

USE [master]

GO

/\*\*\*\*\*\* Object:  Database [AutoQuizzer]    Script Date: 7/8/2023 11:13:15 μμ \*\*\*\*\*\*/

CREATE DATABASE [AutoQuizzer]

 CONTAINMENT = NONE

 ON  PRIMARY

( NAME = N'AutoQuizzer', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\DATA\AutoQuizzer.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )

 LOG ON

( NAME = N'AutoQuizzer\_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\DATA\AutoQuizzer\_log.ldf' , SIZE = 8192KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )

 WITH CATALOG\_COLLATION = DATABASE\_DEFAULT

GO

ALTER DATABASE [AutoQuizzer] SET COMPATIBILITY\_LEVEL = 150

GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))

begin

EXEC [AutoQuizzer].[dbo].[sp\_fulltext\_database] @action = 'enable'

end

GO

ALTER DATABASE [AutoQuizzer] SET ANSI\_NULL\_DEFAULT OFF

GO

ALTER DATABASE [AutoQuizzer] SET ANSI\_NULLS OFF

GO

ALTER DATABASE [AutoQuizzer] SET ANSI\_PADDING OFF

GO

ALTER DATABASE [AutoQuizzer] SET ANSI\_WARNINGS OFF

GO

ALTER DATABASE [AutoQuizzer] SET ARITHABORT OFF

GO

ALTER DATABASE [AutoQuizzer] SET AUTO\_CLOSE OFF

GO

ALTER DATABASE [AutoQuizzer] SET AUTO\_SHRINK OFF

GO

ALTER DATABASE [AutoQuizzer] SET AUTO\_UPDATE\_STATISTICS ON

GO

ALTER DATABASE [AutoQuizzer] SET CURSOR\_CLOSE\_ON\_COMMIT OFF

GO

ALTER DATABASE [AutoQuizzer] SET CURSOR\_DEFAULT  GLOBAL

GO

ALTER DATABASE [AutoQuizzer] SET CONCAT\_NULL\_YIELDS\_NULL OFF

GO

ALTER DATABASE [AutoQuizzer] SET NUMERIC\_ROUNDABORT OFF

GO

ALTER DATABASE [AutoQuizzer] SET QUOTED\_IDENTIFIER OFF

GO

ALTER DATABASE [AutoQuizzer] SET RECURSIVE\_TRIGGERS OFF

GO

ALTER DATABASE [AutoQuizzer] SET  DISABLE\_BROKER

GO

ALTER DATABASE [AutoQuizzer] SET AUTO\_UPDATE\_STATISTICS\_ASYNC OFF

GO

ALTER DATABASE [AutoQuizzer] SET DATE\_CORRELATION\_OPTIMIZATION OFF

GO

ALTER DATABASE [AutoQuizzer] SET TRUSTWORTHY OFF

GO

ALTER DATABASE [AutoQuizzer] SET ALLOW\_SNAPSHOT\_ISOLATION OFF

GO

ALTER DATABASE [AutoQuizzer] SET PARAMETERIZATION SIMPLE

GO

ALTER DATABASE [AutoQuizzer] SET READ\_COMMITTED\_SNAPSHOT OFF

GO

ALTER DATABASE [AutoQuizzer] SET HONOR\_BROKER\_PRIORITY OFF

GO

ALTER DATABASE [AutoQuizzer] SET RECOVERY SIMPLE

GO

ALTER DATABASE [AutoQuizzer] SET  MULTI\_USER

GO

ALTER DATABASE [AutoQuizzer] SET PAGE\_VERIFY CHECKSUM

GO

ALTER DATABASE [AutoQuizzer] SET DB\_CHAINING OFF

GO

ALTER DATABASE [AutoQuizzer] SET FILESTREAM( NON\_TRANSACTED\_ACCESS = OFF )

GO

ALTER DATABASE [AutoQuizzer] SET TARGET\_RECOVERY\_TIME = 60 SECONDS

GO

ALTER DATABASE [AutoQuizzer] SET DELAYED\_DURABILITY = DISABLED

GO

ALTER DATABASE [AutoQuizzer] SET ACCELERATED\_DATABASE\_RECOVERY = OFF

GO

ALTER DATABASE [AutoQuizzer] SET QUERY\_STORE = OFF

GO

USE [AutoQuizzer]

GO

/\*\*\*\*\*\* Object:  User [aantonopoulos]    Script Date: 7/8/2023 11:13:15 μμ \*\*\*\*\*\*/

CREATE USER [aantonopoulos] FOR LOGIN [aantonopoulos] WITH DEFAULT\_SCHEMA=[dbo]

GO

ALTER ROLE [db\_owner] ADD MEMBER [aantonopoulos]

GO

ALTER ROLE [db\_accessadmin] ADD MEMBER [aantonopoulos]

GO

ALTER ROLE [db\_securityadmin] ADD MEMBER [aantonopoulos]

GO

ALTER ROLE [db\_ddladmin] ADD MEMBER [aantonopoulos]

GO

ALTER ROLE [db\_backupoperator] ADD MEMBER [aantonopoulos]

GO

ALTER ROLE [db\_datareader] ADD MEMBER [aantonopoulos]

GO

ALTER ROLE [db\_datawriter] ADD MEMBER [aantonopoulos]

GO

ALTER ROLE [db\_denydatareader] ADD MEMBER [aantonopoulos]

GO

ALTER ROLE [db\_denydatawriter] ADD MEMBER [aantonopoulos]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[Category]    Script Date: 7/8/2023 11:13:15 μμ \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Category](

    [CategoryID] [int] IDENTITY(1,1) NOT NULL,

    [Title] [nvarchar](50) NOT NULL,

    [Description] [nvarchar](max) NOT NULL,

    [UserID] [int] NOT NULL,

 CONSTRAINT [PK\_Category\_1] PRIMARY KEY CLUSTERED

(

    [CategoryID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[Exam]    Script Date: 7/8/2023 11:13:15 μμ \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Exam](

    [ExamID] [int] IDENTITY(1,1) NOT NULL,

    [AssignmentDate] [datetime] NOT NULL,

    [ResolvedDate] [datetime] NULL,

    [Grade] [int] NULL,

    [UserID] [int] NULL,

    [TestID] [int] NULL,

 CONSTRAINT [PK\_Exam] PRIMARY KEY CLUSTERED

(

    [ExamID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[ExamDetail]    Script Date: 7/8/2023 11:13:15 μμ \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[ExamDetail](

    [ExamDetailID] [int] IDENTITY(1,1) NOT NULL,

    [ExamID] [int] NOT NULL,

    [TestQuestionID] [int] NOT NULL,

    [QuestionGrade] [int] NOT NULL,

 CONSTRAINT [PK\_ExamDetails] PRIMARY KEY CLUSTERED

(

    [ExamDetailID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[Question]    Script Date: 7/8/2023 11:13:15 μμ \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Question](

    [QuestionID] [int] IDENTITY(1,1) NOT NULL,

    [Difficulty] [nvarchar](20) NOT NULL,

    [Description] [nvarchar](max) NOT NULL,

    [SubcategoryID] [int] NOT NULL,

 CONSTRAINT [PK\_Questions] PRIMARY KEY CLUSTERED

(

    [QuestionID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[QuestionAnswer]    Script Date: 7/8/2023 11:13:15 μμ \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[QuestionAnswer](

    [AnswerID] [int] IDENTITY(1,1) NOT NULL,

    [QuestionID] [int] NOT NULL,

    [Description] [nvarchar](max) NOT NULL,

    [IsCorrect] [smallint] NOT NULL,

 CONSTRAINT [PK\_QuestionAnswer] PRIMARY KEY CLUSTERED

(

    [AnswerID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[Subcategory]    Script Date: 7/8/2023 11:13:15 μμ \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Subcategory](

    [SubcategoryID] [int] IDENTITY(1,1) NOT NULL,

    [CategoryID] [int] NOT NULL,

    [Title] [nvarchar](50) NOT NULL,

    [Description] [nvarchar](max) NOT NULL,

 CONSTRAINT [PK\_Category] PRIMARY KEY CLUSTERED

(

    [SubcategoryID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[Test]    Script Date: 7/8/2023 11:13:15 μμ \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Test](

    [TestID] [int] IDENTITY(1,1) NOT NULL,

    [Title] [nvarchar](50) NOT NULL,

    [Subject] [nvarchar](50) NOT NULL,

    [Difficulty] [nvarchar](20) NOT NULL,

    [QuestionsNumber] [int] NOT NULL,

    [Categories] [nvarchar](50) NOT NULL,

    [Subcategories] [nvarchar](50) NOT NULL,

    [ExaminerID] [int] NOT NULL,

 CONSTRAINT [PK\_Tests] PRIMARY KEY CLUSTERED

(

    [TestID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[TestQuestion]    Script Date: 7/8/2023 11:13:15 μμ \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[TestQuestion](

    [TestQuestionID] [int] IDENTITY(1,1) NOT NULL,

    [TestID] [int] NULL,

    [QuestionID] [int] NULL,

 CONSTRAINT [PK\_TestQuestions\_1] PRIMARY KEY CLUSTERED

(

    [TestQuestionID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object:  Table [dbo].[User]    Script Date: 7/8/2023 11:13:15 μμ \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[User](

    [UserID] [int] IDENTITY(1,1) NOT NULL,

    [Username] [nvarchar](50) NOT NULL,

    [FirstName] [nvarchar](250) NOT NULL,

    [LastName] [nvarchar](250) NOT NULL,

    [Email] [nvarchar](100) NOT NULL,

    [Password] [nvarchar](250) NOT NULL,

    [PhoneNumber] [nvarchar](15) NOT NULL,

    [Institution] [nvarchar](250) NOT NULL,

    [UserRole] [nvarchar](10) NOT NULL,

 CONSTRAINT [PK\_Users] PRIMARY KEY CLUSTERED

(

    [UserID] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

 CONSTRAINT [IX\_Users] UNIQUE NONCLUSTERED

(

    [Email] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Category]  WITH CHECK ADD  CONSTRAINT [FK\_Category\_User] FOREIGN KEY([UserID])

REFERENCES [dbo].[User] ([UserID])

GO

ALTER TABLE [dbo].[Category] CHECK CONSTRAINT [FK\_Category\_User]

GO

ALTER TABLE [dbo].[Exam]  WITH CHECK ADD  CONSTRAINT [FK\_Assignment\_Test] FOREIGN KEY([TestID])

REFERENCES [dbo].[Test] ([TestID])

GO

ALTER TABLE [dbo].[Exam] CHECK CONSTRAINT [FK\_Assignment\_Test]

GO

ALTER TABLE [dbo].[Exam]  WITH CHECK ADD  CONSTRAINT [FK\_Exam\_User] FOREIGN KEY([UserID])

REFERENCES [dbo].[User] ([UserID])

GO

ALTER TABLE [dbo].[Exam] CHECK CONSTRAINT [FK\_Exam\_User]

GO

ALTER TABLE [dbo].[ExamDetail]  WITH CHECK ADD  CONSTRAINT [FK\_ExamDetail\_Exam] FOREIGN KEY([ExamID])

REFERENCES [dbo].[Exam] ([ExamID])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[ExamDetail] CHECK CONSTRAINT [FK\_ExamDetail\_Exam]

GO

ALTER TABLE [dbo].[ExamDetail]  WITH CHECK ADD  CONSTRAINT [FK\_ExamDetail\_TestQuestion] FOREIGN KEY([TestQuestionID])

REFERENCES [dbo].[TestQuestion] ([TestQuestionID])

GO

ALTER TABLE [dbo].[ExamDetail] CHECK CONSTRAINT [FK\_ExamDetail\_TestQuestion]

GO

ALTER TABLE [dbo].[Question]  WITH CHECK ADD  CONSTRAINT [FK\_Question\_Subcategory] FOREIGN KEY([SubcategoryID])

REFERENCES [dbo].[Subcategory] ([SubcategoryID])

GO

ALTER TABLE [dbo].[Question] CHECK CONSTRAINT [FK\_Question\_Subcategory]

GO

ALTER TABLE [dbo].[QuestionAnswer]  WITH CHECK ADD  CONSTRAINT [FK\_QuestionAnswer\_Question] FOREIGN KEY([QuestionID])

REFERENCES [dbo].[Question] ([QuestionID])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[QuestionAnswer] CHECK CONSTRAINT [FK\_QuestionAnswer\_Question]

GO

ALTER TABLE [dbo].[Subcategory]  WITH CHECK ADD  CONSTRAINT [FK\_Subcategory\_Category] FOREIGN KEY([CategoryID])

REFERENCES [dbo].[Category] ([CategoryID])

GO

ALTER TABLE [dbo].[Subcategory] CHECK CONSTRAINT [FK\_Subcategory\_Category]

GO

ALTER TABLE [dbo].[Test]  WITH CHECK ADD  CONSTRAINT [FK\_Test\_User] FOREIGN KEY([ExaminerID])

REFERENCES [dbo].[User] ([UserID])

GO

ALTER TABLE [dbo].[Test] CHECK CONSTRAINT [FK\_Test\_User]

GO

ALTER TABLE [dbo].[TestQuestion]  WITH CHECK ADD  CONSTRAINT [FK\_TestQuestions\_Questions] FOREIGN KEY([QuestionID])

REFERENCES [dbo].[Question] ([QuestionID])

ON UPDATE CASCADE

ON DELETE SET NULL

GO

ALTER TABLE [dbo].[TestQuestion] CHECK CONSTRAINT [FK\_TestQuestions\_Questions]

GO

ALTER TABLE [dbo].[TestQuestion]  WITH CHECK ADD  CONSTRAINT [FK\_TestQuestions\_Tests] FOREIGN KEY([TestID])

REFERENCES [dbo].[Test] ([TestID])

ON UPDATE CASCADE

ON DELETE SET NULL

GO

ALTER TABLE [dbo].[TestQuestion] CHECK CONSTRAINT [FK\_TestQuestions\_Tests]

GO

USE [master]

GO

ALTER DATABASE [AutoQuizzer] SET  READ\_WRITE

GO