

# *ΕΡΓΑΣΙΑ ΜΑΘΗΜΑΤΟΣ*

---

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

Τμήμα Πληροφορικής



Μάθημα: «ΕΚΠΑΙΔΕΥΤΙΚΟ ΛΟΓΙΣΜΙΚΟ (80 εξ.)»

Π18101 – ΑΝΑΣΤΑΣΙΑ ΙΩΑΝΝΑ ΜΕΞΑ

Π18078 – ΑΘΑΝΑΣΙΑ ΚΟΜΜΑΤΙΔΟΥ

Π18123 – ΒΑΣΙΛΙΚΗ ΠΑΣΙΑ



## Εκφώνηση της άσκησης

Ζητείται να γίνει ένα αλληλεπιδραστικό λογισμικό εκπαίδευσης μαθητών για ένα από τα παρακάτω πεδία διδασκαλίας:

A. Γλώσσα Προγραμματισμού C#, ή Java ή Python.

B. Την Ελληνική Γλώσσα.

Η εργασία αυτή θα περιλαμβάνει τρόπους παρουσίασης του διδακτικού υλικού με στόχο να γίνει το θέμα κατανοητό και να μπορεί να απομνημονευθεί από τους μαθητές καθώς και να εμπεδωθεί η ύλη μέσω ασκήσεων. Ο κύριος σκοπός της εργασίας είναι ο καλός σχεδιασμός και υλοποίηση του εκπαιδευτικού λογισμικού (διδασκαλία – αξιολόγηση του μαθητή) και όχι η εισαγωγή μεγάλου μέρους κεφαλαίων.

Συγκεκριμένα ζητούνται τα παρακάτω:

### ΒΑΣΙΚΕΣ ΛΕΙΤΟΥΡΓΙΕΣ

#### 1. ΠΑΡΟΥΣΙΑΣΗ ΔΙΔΑΣΚΑΛΙΑΣ

#### 2. ΕΡΩΤΗΣΕΙΣ / ΤΕΣΤ ΑΥΤΟΑΞΙΟΛΟΓΗΣΗΣ

##### 2.1 Κατασκευή των τεστ.

- Θα πρέπει να κατασκευάζονται τεστ για την αυτοαξιολόγηση των μαθητών, για κάθε ενότητα διδασκαλίας.
- Στα επαναληπτικά τεστ, θα πρέπει να παρουσιάζονται ασκήσεις απόλειτες τις ενότητες.
- Η μορφή των ασκήσεων μπορεί να είναι πολλαπλών επιλογών ή άλλης μορφής ανάλογα με τη δική σας ανάλυση απαιτήσεων και σχεδιασμό.

##### 2.2 Αποθήκευση στατιστικών στοιχείων προόδου του μαθητή.

Θα πρέπει να υπάρχει μια βάση δεδομένων όπου να αποθηκεύονται στοιχεία για κάθε μαθητή σχετικά με την πρόοδό του. Τα στοιχεία θα βασίζονται στην απόδοση των μαθητών στα τεστ αυτοαξιολόγησης αλλά και στα στατιστικά επισκεψιμότητας της παρουσίασης του θέματος.

##### 2.3 Διάγνωση λαθών του μαθητή και αλληλεπίδραση.

Στη διάγνωση λαθών ζητείται να μπορεί το σύστημα να εντοπίσει αν ο μαθητής έχει πρόβλημα σε κάποια κατηγορία λαθών (π.χ. συντακτικά λάθη/ λάθη λογικής/ τυπογραφικά λάθη). Αν εντοπιστεί κάτι τέτοιο θα πρέπει το σύστημα να παρουσιάζει πάλι τη θεωρία και περισσότερες ερωτήσεις στο συγκεκριμένο θέμα, όπου υπάρχει πρόβλημα. Επίσης θα πρέπει να καταγράφεται η συγκεκριμένη αδυναμία στα στατιστικά προόδου του μαθητή και να σβήνεται όταν ο μαθητής φαίνεται ότι έχει πια μάθει το συγκεκριμένο θέμα

### ΠΡΟΣΘΕΤΕΣ ΛΕΙΤΟΥΡΓΙΕΣ (ΠΡΟΑΙΡΕΤΙΚΕΣ)

#### 1. Διαχείριση από Καθηγητή

#### 2. Λειτουργία στο Web

### B) Συνοδευτικά εγχειρίδια

Η εφαρμογή θα πρέπει να συνοδεύεται από τα εξής εγχειρίδια:

1. Εγχειρίδιο χρήστη (user manual)
2. On-line help
3. Εγχειρίδιο Ανάλυσης και Σχεδιασμού της εφαρμογής (Τεχνικό Εγχειρίδιο).



## ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1	Εισαγωγή.....	4
1.1	Στόχοι της εργασίας.....	4
1.2	Υλοποίηση της εργασίας.....	4
1.3	Ορισμός του προβλήματος προς επίλυση.....	4
2	Ανάλυση απαιτήσεων.....	4
2.1	Σύλληψη απαιτήσεων .....	4
2.2	Ανάλυση και σχεδιασμός.....	6
3	Εγχειρίδιο Ανάλυσης και Σχεδιασμού της εφαρμογής (Τεχνικό Εγχειρίδιο) .....	8
3.1	Βάση δεδομένων .....	8
3.2	C# Project .....	9
3.3	C# ASP.NET Web Project.....	20
4	Εισαγωγή της web εφαρμογής στον IIS (Internet Information Services) .....	26
5	Βιβλιογραφικές Πηγές.....	30



## 1 Εισαγωγή

### 1.1 Στόχοι της εργασίας

Θέλουμε να υλοποιήσουμε ένα αλληλεπιδραστικό λογισμικό εκπαίδευσης μαθητών για τη γλώσσα προγραμματισμού Python.

### 1.2 Υλοποίηση της εργασίας

Η εργασία έχει υλοποιηθεί με τη χρήση δύο τεχνολογιών. Έχουμε υλοποιήσει μία C# desktop εφαρμογή, στην οποία εμφανίζεται μία σειρά μαθημάτων και ασκήσεων εκμάθησης της γλώσσας προγραμματισμού Python, και μία δεύτερη C# ASP.NET Web εφαρμογή που συνδέεται με την πρώτη και εκεί μπορούν οι μαθητές να δουν πληροφορίες σχετικά με την πρόοδο τους, αλλά και οι καθηγητές να ενημερωθούν για την πρόοδο των μαθητών τους. Επιπλέον, η βάση δεδομένων που χρησιμοποιήθηκε είναι η PostgreSQL.

### 1.3 Ορισμός του προβλήματος προς επίλυση

Αρχικά καλούμαστε να ορίσουμε το πρόβλημα που θα επιλύσουμε με την ανάπτυξη του λογισμικού.

- Η εφαρμογή έχει ως στόχο να διδάξει στους μαθητές τις βασικές αρχές του προγραμματισμού με τη γλώσσα Python τόσο με την αναλυτική παρουσίαση της θεωρίας της, όσο και με την επίλυση διαφόρων τεστ.
- Η ηλεκτρονική εφαρμογή/πλατφόρμα αποσκοπεί στην παρακολούθηση της προόδου των μαθητών και την εξερεύνηση των λαθών τους, είτε από τον καθηγητή τους, είτε από τους ίδιους τους μαθητές.

## 2 Ανάλυση απαιτήσεων

### 2.1 Σύλληψη απαιτήσεων

Η ιστοσελίδα υποστηρίζει την προβολή αναλυτικής και με χρήση παραδειγμάτων παρουσίασης της θεωρίας γύρω από τρία βασικά κεφάλαια της γλώσσας προγραμματισμού Python:

1. Μεταβλητές και Τελεστές
2. Δομή Επιλογής
3. Δομή Επανάληψης

Τα τεστ που καλούνται να επιλύσουν οι μαθητές είναι δύο ειδών:

1. Τεστ εξάσκησης, στόχος των οποίων είναι η καλύτερη κατανόηση της θεωρίας μέσω διαβαθμισμένης δυσκολίας ασκήσεων.
2. Τεστ αξιολόγησης, το οποίο μπορούν οι μαθητές να λύσουν αφού ολοκληρώσουν όλα τα προηγούμενα τεστ, και στοχεύει στην τελική αξιολόγηση των γνώσεων που έλαβε ο μαθητής.



Επιπλέον, για κάθε κεφάλαιο υπάρχουν δύο διαθέσιμα τεστ για εξάσκηση, διαβαθμισμένης δυσκολίας, ένα εύκολο και ένα πιο δύσκολο. Η διαφορά στην δυσκολία εστιάζεται τόσο στην ύπαρξη πιο απαιτητικών ασκήσεων στο τεστ, όσο στον τύπο των ασκήσεων κάθε τεστ.

Συγκεκριμένα:

1. Τα εύκολα τεστ εξάσκησης περιλαμβάνουν ερωτήσεις Σωστού ή Λάθους και Πολλαπλής Επιλογής.
2. Τα δύσκολα τεστ εξάσκησης περιλαμβάνουν ερωτήσεις Συμπλήρωσης Κενών και Τοποθέτησης των εντολών στην κατάλληλη σειρά.

Η διάγνωση των λαθών των μαθητών είναι αναλυτική και κατατοπιστική, καθώς οι μαθητές μπορούν να δουν σε ποιες ερωτήσεις έχουν πιθανώς κάνει κάποιο λάθος, καθώς και τον τύπο του λάθους που έχουν κάνει. Αναλυτικότερα, οι τύποι λαθών που έχουν οριστεί είναι τρεις:

1. Συντακτικά λάθη, όπως λάθος στην σύνταξη κάποιων εντολών (πχ. επανάληψης) ή παράληψη κάποιου τελεστή/χαρακτήρα (πχ. ':').
2. Λογικά λάθη, όπως λάθος στην τοποθέτηση στη σωστή σειρά κάποιων εντολών ή χρήση κάποιου λάθος τελεστή (πχ. '>' αντί για '<').
3. Συνδυαστικά λάθη, δηλαδή λάθη που αντιστοιχούν και στις δύο παραπάνω κατηγορίες.

Επίσης, με βάση τα λάθη που πιθανώς υπάρχουν στις απαντήσεις, ένας μαθητής μπορεί να ενημερωθεί σε ποια κεφάλαια υστερεί και θα έπρεπε να ξαναδιαβάσει.

Οι καθηγητές έχουν την δυνατότητα να παρακολουθούν την πρόοδο των μαθητών τους, βλέποντας τις απαντήσεις τους στα τεστ εξάσκησης και αξιολόγησης.

Για την εγγραφή ενός μαθητή ή ενός καθηγητή στην πλατφόρμα θα του ζητούνται να εισάγει μία φορά τα εξής στοιχεία: ονοματεπώνυμο, username και password, και έπειτα αν η εγγραφή είναι επιτυχής ανακατευθύνεται στην κατάλληλη main σελίδα.

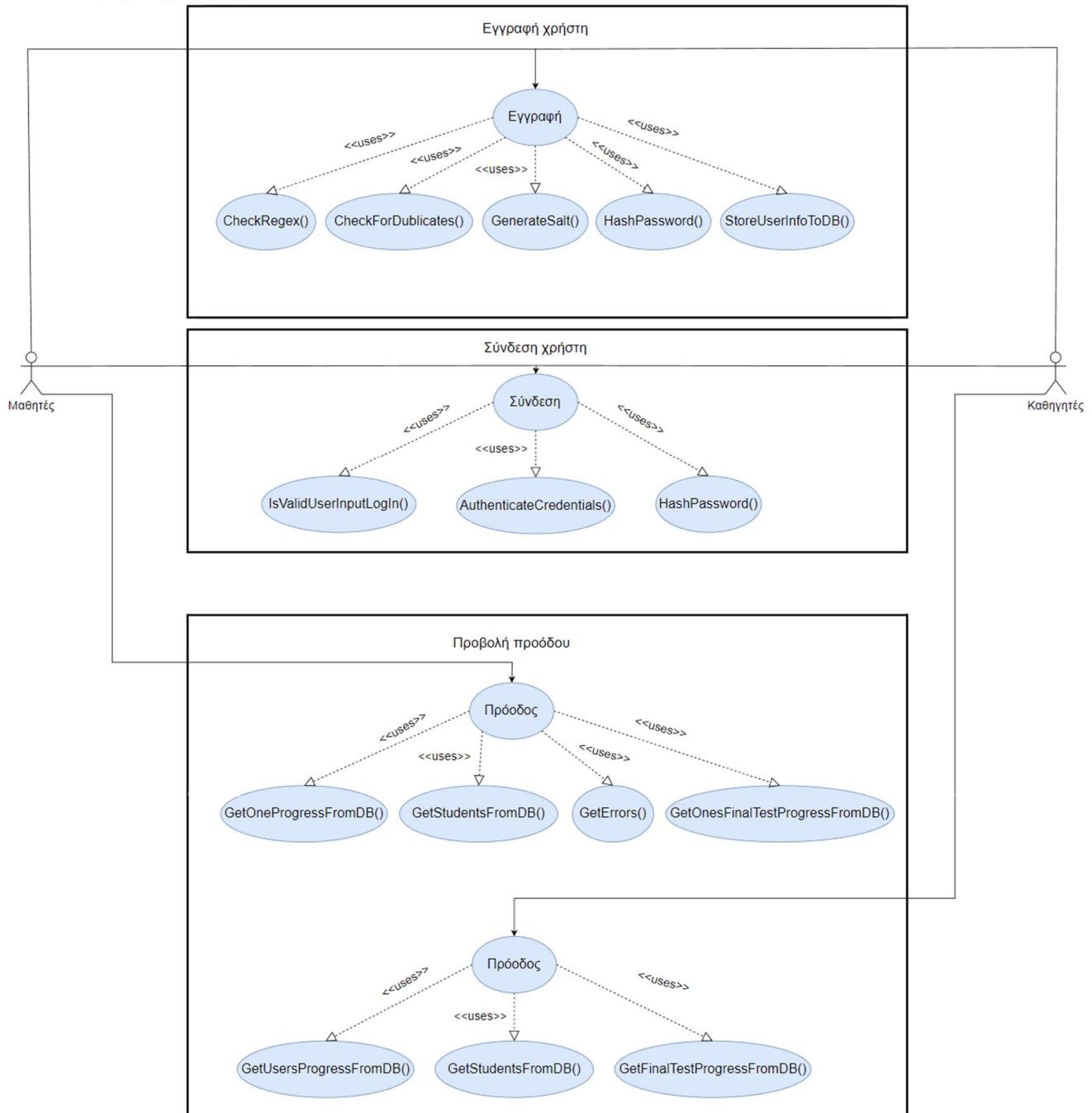


## 2.2 Ανάλυση και σχεδιασμός

### Διαγράμματα Περιπτώσεων Χρήστης

Τα διαγράμματα περιπτώσεων χρήστης περιγράφουν τη συμπεριφορά ενός συστήματος από την οπτική γωνία ενός χρήστη. Επιτρέπουν τον ορισμό των ορίων του συστήματος και του περιβάλλοντος.

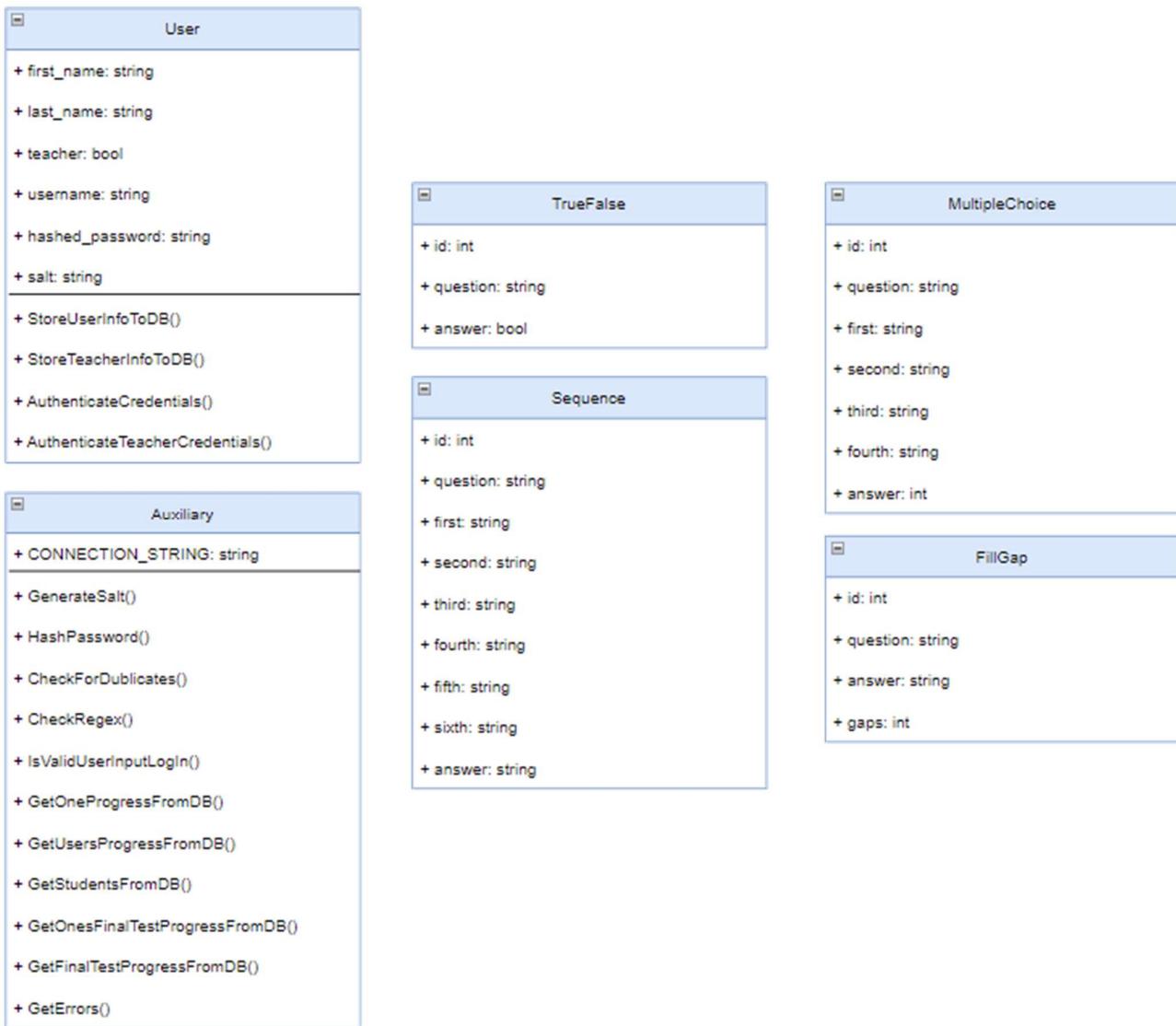
Στο παρακάτω διάγραμμα περιγράφονται οι λειτουργίες που μπορούν να πραγματοποιήσουν οι διάφοροι χρήστες.





## Διαγράμματα Τάξεων (1<sup>η</sup> έκδοση)

Το διάγραμμα των κλάσεων ενός συστήματος είναι ένα διάγραμμα δομής που περιέχει τις κλάσεις μαζί με τους αντίστοιχους δεσμούς εξάρτησης, γενίκευσης και σύνδεσης, που έχουμε δημιουργήσει.

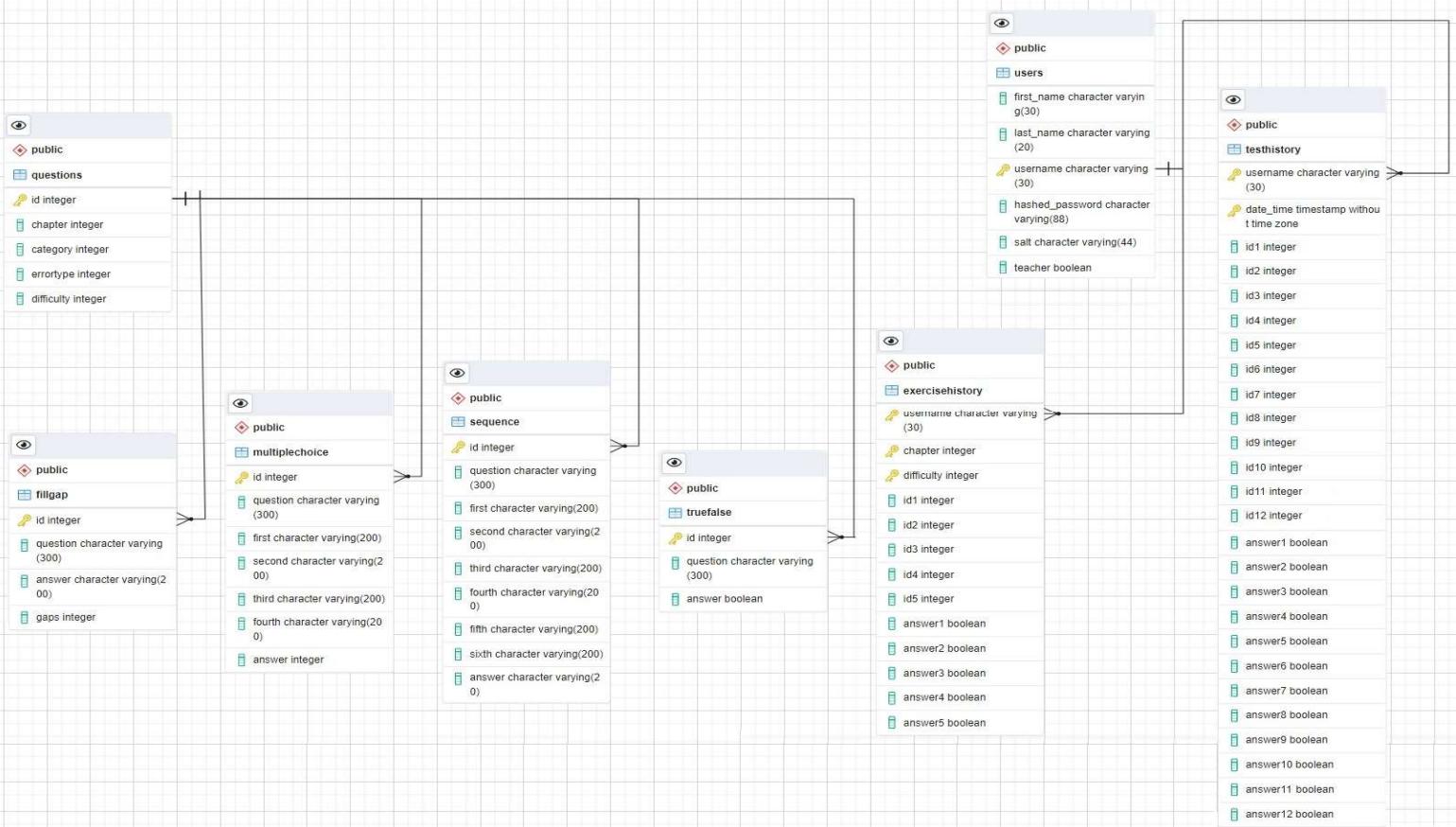




### 3 Εγχειρίδιο Ανάλυσης και Σχεδιασμού της εφαρμογής (Τεχνικό Εγχειρίδιο)

#### 3.1 Βάση δεδομένων

Και τα δύο project συνδέονται πάνω στην ίδια βάση δεδομένων, με όνομα elearning, της οποίας το σχήμα φαίνεται παρακάτω.



- Ο πίνακας questions περιέχει πληροφορίες για την κάθε ερώτηση. Πιο συγκεκριμένα, περιέχει το id της ερώτησης, σε ποιο κεφάλαιο ανήκει, σε περίπτωση που απαντηθεί λάθος αν είναι συντακτικό ή λογικό ή συνδυαστικό λάθος, καθώς και το επίπεδο της δυσκολίας της (ευκολό, δύσκολο).
- Ο πίνακας users περιέχει πληροφορίες για τον χρήστη, όπως το ονοματεπώνυμο του, το username, τον κρυπτογραφημένο (hashed) κωδικό του, αλλά και το salt που χρησιμοποιήθηκε κατά την κρυπτογράφηση και την πληροφορία αν είναι καθηγητής ή μαθητής.
- Ο πίνακας fillgap περιέχει πληροφορίες για τις ερωτήσεις συμπλήρωσης κενών. Στην βάση είναι αποθηκευμένα το περιεχόμενο της ερώτησης, η σωστή απάντηση και το πλήθος των κενών που ζητούνται να συμπληρωθούν.
- Ο πίνακας multiplechoice περιέχει πληροφορίες για τις ερωτήσεις πολλαπλών επιλογών. Στην βάση είναι αποθηκευμένα η εκφώνηση της ερώτησης, οι πιθανές απαντήσεις και η σωστή απάντηση.
- Ο πίνακας sequence περιέχει πληροφορίες για τις ερωτήσεις τοποθέτησης εντολών σε σωστή σειρά. Στην βάση είναι αποθηκευμένα η εκφώνηση της ερώτησης, οι εντολές και η σωστή απάντηση.
- Ο πίνακας truefalse περιέχει πληροφορίες για τις ερωτήσεις σωστού λάθους. Στην βάση είναι αποθηκευμένα η εκφώνηση της ερώτησης και η σωστή απάντηση.
- Στους πίνακες fillgap, multiplechoice, sequence, truefalse το πεδίο id αποτελεί ξένο κλειδί στον πίνακα



questions.

- Ο πίνακας exercisehistory περιέχει πληροφορίες για τις απαντήσεις του χρήστη στις ερωτήσεις που του έπεσαν στα τεστ εξάσκησης. Συγκεκριμένα, περιέχονται το username του χρήστη που έδωσε το τεστ, για ποιο κεφάλαιο αφορά, το επίπεδο δυσκολίας, τα id των ερωτήσεων που έπεσαν, καθώς και η πληροφορία αν απάντησε σωστά στις ερωτήσεις αυτές.
- Ο πίνακας testhistory περιέχει πληροφορίες για τις απαντήσεις του χρήστη στις ερωτήσεις που του έπεσαν στο τελικό τεστ. Συγκεκριμένα, περιέχονται το username του χρήστη που έδωσε το τεστ, η ημερομηνία καταχώρησης του τεστ, τα id των ερωτήσεων που έπεσαν, καθώς και η πληροφορία αν απάντησε σωστά στις ερωτήσεις αυτές.
- Στον πίνακα exercisehistory οι εγγραφές ανά χρήστη ανανεώνονται κάθε φορά που ο χρήστης υποβάλει ένα τεστ εξάσκησης, καθώς σκοπός είναι η συλλογή των πιο πρόσφατων δεδομένων που αντανακλούν το επίπεδο της γνώσης του χρήστη, και έτσι προσαρμόζονται ανάλογα τα τεστ.
- Στον πίνακα testhistory η εγγραφή ανά χρήστη εισάγεται μία μόνο φορά όταν ο χρήστης υποβάλει το τελικό τεστ αξιολόγησης, καθώς δεν είναι δυνατή η επανυποβολή του.
- Στους πίνακες exercisehistory, testhistory το πεδίο username αποτελεί ξένο κλειδί στον πίνακα users.

## 3.2 C# Project

Έχουμε υλοποιήσει μερικές κλάσεις για την ανάπτυξη της εφαρμογής μας.

Οι κλάσεις που χρησιμοποιούμε για την δημιουργία αντικειμένων των διαφορετικών τύπων ερωτήσεων είναι οι εξής:

- **TrueFalse**

```
namespace EkpaideutikoLogismiko
{
    class TrueFalse
    {
        public int id;
        public string question;
        public bool answer;

        //----Constructors----
        public TrueFalse(int id, string question, bool answer)
        {
            this.id = id;
            this.question = question;
            this.answer = answer;
        }
    }
}
```



- **MultipleChoice**

```
namespace Ekpaeideutikologismiko
{
    class MultipleChoice
    {
        public int id;
        public string question;
        public string first;
        public string second;
        public string third;
        public string fourth;
        public int answer;

        //----Constructors----
        public MultipleChoice(int id, string question, string first, string second, string third, string fourth, int answer)
        {
            this.id = id;
            this.question = question;
            this.first = first;
            this.second = second;
            this.third = third;
            this.fourth = fourth;
            this.answer = answer;
        }
    }
}
```

- **Sequence**

```
namespace Ekpaeideutikologismiko
{
    class Sequence
    {
        public int id;
        public string question;
        public string first;
        public string second;
        public string third;
        public string fourth;
        public string fifth;
        public string sixth;
        public string answer;

        //----Constructors----
        public Sequence(int id, string question, string first, string second, string third, string fourth, string fifth, string sixth, string answer)
        {
            this.id = id;
            this.question = question;
            this.first = first;
            this.second = second;
            this.third = third;
            this.fourth = fourth;
            this.fifth = fifth;
            this.sixth = sixth;
            this.answer = answer;
        }
    }
}
```



- FillGap

```
namespace EkipaideutikoLogismiko
{
    class FillGap
    {
        public int id;
        public string question;
        public string answer;
        public int gaps;

        //----Constructors----
        public FillGap(int id, string question, string answer, int gaps)
        {
            this.id = id;
            this.question = question;
            this.answer = answer;
            this.gaps = gaps;
        }
    }
}
```

Η κλάση που χρησιμοποιούμε για την δημιουργία αντικειμένων των χρηστών καθώς και την διαχείριση μερικών λειτουργιών όπως για παράδειγμα τις λειτουργίες της σύνδεσης και της εγγραφής είναι η κλάση User.

```
namespace EkipaideutikoLogismiko
{
    public class User
    {
        //----Fields----
        public string first_name;
        public string last_name;
        public bool teacher;
        public string username;
        public string hashed_password;
        public string salt;

        //----Constructors----
        public User(string first_name, string last_name, string username, string hashed_password, string salt, bool teacher)
        {
            this.first_name = first_name;
            this.last_name = last_name;
            this.teacher = teacher;
            this.username = username;
            this.hashed_password = hashed_password;
            this.salt = salt;
        }

        public User(string username)
        {
            this.username = username;
        }
    }
}
```



Οι μέθοδοι της κλάσης User αναλύονται παρακάτω.

- StoreUserInfoToDB() και StoreTeacherInfoToDB()

Αυτές οι μέθοδοι χρησιμοποιούνται για την διαδικασία εγγραφής των χρηστών.

```
-----Methods-----
=====
//Stores user's info(firstname, lastname, email, username,hashedpassword,salt) in DB when he signs up.
public string StoreUserInfoToDB()
{
    string query = "insert into users values (@first_name, @last_name, @username, @hashed_password, @salt, @teacher)";
    int rowsAffected = -1; //false value
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);
        connection.Open();
        //define query's parameters
        NpgsqlCommand command = new NpgsqlCommand(query, connection);
        command.Parameters.AddWithValue("first_name", first_name);
        command.Parameters.AddWithValue("last_name", last_name);
        command.Parameters.AddWithValue("username", username);
        command.Parameters.AddWithValue("hashed_password", hashed_password);
        command.Parameters.AddWithValue("salt", salt);
        command.Parameters.AddWithValue("teacher", teacher);
        rowsAffected = command.ExecuteNonQuery(); //run query
        connection.Close();
    }
    catch
    {
        return "Υπήρξε κάποιο πρόβλημα.";
    }
    if (rowsAffected == -1)
        return "Υπήρξε κάποιο πρόβλημα με την βάση δεδομένων, ξαναπροσπαθήστε αργότερα.";
    else
        return "done";
}

//Stores user's info(firstname, lastname, email, username,hashedpassword,salt) in DB when he signs up.
public string StoreTeacherInfoToDB()
{
    string query = "insert into users(first_name, last_name, username, hashed_password, salt, teacher) values (@first_name, @last_name, @username, @hashed_password, @salt, @teacher)";
    int rowsAffected = -1; //false value
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);
        connection.Open();
        //define query's parameters
        NpgsqlCommand command = new NpgsqlCommand(query, connection);
        command.Parameters.AddWithValue("first_name", first_name);
        command.Parameters.AddWithValue("last_name", last_name);
        command.Parameters.AddWithValue("username", username);
        command.Parameters.AddWithValue("hashed_password", hashed_password);
        command.Parameters.AddWithValue("salt", salt);
        command.Parameters.AddWithValue("teacher", teacher);
        rowsAffected = command.ExecuteNonQuery(); //run query
        connection.Close();
    }
    catch
    {
        return "Υπήρξε κάποιο πρόβλημα.";
    }
    if (rowsAffected == -1)
        return "Υπήρξε κάποιο πρόβλημα με την βάση δεδομένων, ξαναπροσπαθήστε αργότερα.";
    else
        return "done";
}
```

```
//Stores user's info(firstname, lastname, email, username,hashedpassword,salt) in DB when he signs up.
public string StoreUserInfoToDB()
{
    string query = "insert into users values (@first_name, @last_name, @username, @hashed_password, @salt, @teacher)";
    int rowsAffected = -1; //false value
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);
        connection.Open();
        //define query's parameters
        NpgsqlCommand command = new NpgsqlCommand(query, connection);
        command.Parameters.AddWithValue("first_name", first_name);
        command.Parameters.AddWithValue("last_name", last_name);
        command.Parameters.AddWithValue("username", username);
        command.Parameters.AddWithValue("hashed_password", hashed_password);
        command.Parameters.AddWithValue("salt", salt);
        command.Parameters.AddWithValue("teacher", teacher);
        rowsAffected = command.ExecuteNonQuery(); //run query
        connection.Close();
    }
    catch
    {
        return "Υπήρξε κάποιο πρόβλημα.";
    }
    if (rowsAffected == -1)
        return "Υπήρξε κάποιο πρόβλημα με την βάση δεδομένων, ξαναπροσπαθήστε αργότερα.";
    else
        return "done";
}

//Stores user's info(firstname, lastname, email, username,hashedpassword,salt) in DB when he signs up.
public string StoreTeacherInfoToDB()
{
    string query = "insert into users(first_name, last_name, username, hashed_password, salt, teacher) values (@first_name, @last_name, @username, @hashed_password, @salt, @teacher)";
    int rowsAffected = -1; //false value
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);
        connection.Open();
        //define query's parameters
        NpgsqlCommand command = new NpgsqlCommand(query, connection);
        command.Parameters.AddWithValue("first_name", first_name);
        command.Parameters.AddWithValue("last_name", last_name);
        command.Parameters.AddWithValue("username", username);
        command.Parameters.AddWithValue("hashed_password", hashed_password);
        command.Parameters.AddWithValue("salt", salt);
        command.Parameters.AddWithValue("teacher", teacher);
        rowsAffected = command.ExecuteNonQuery(); //run query
        connection.Close();
    }
    catch
    {
        return "Υπήρξε κάποιο πρόβλημα.";
    }
    if (rowsAffected == -1)
        return "Υπήρξε κάποιο πρόβλημα με την βάση δεδομένων, ξαναπροσπαθήστε αργότερα.";
    else
        return "done";
}
```



- AuthenticateCredentials() και AuthenticateTeacherCredentials()

Αυτές οι μέθοδοι χρησιμοποιούνται για την διαδικασία σύνδεσης των χρηστών.

```
//Used for logging in. Check if given username exists and if it does, then creates the hashed password anew and compares it with the one at the DB.  
//If username and password match, then he logs in.  
public string AuthenticateCredentials(string rawPassword)  
{  
    string query = "select username, hashed_password, salt from users where teacher = false";  
    List<string> userData = new List<string>(); //store user's username, hashed password and salt  
    try  
    {  
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);  
        connection.Open();  
        NpgsqlCommand command = new NpgsqlCommand(query, connection);  
        NpgsqlDataReader dataReader = command.ExecuteReader(); //run query  
        while (dataReader.Read())  
            userData.Add(dataReader[0].ToString() + " | " + dataReader[1].ToString() + " | " + dataReader[2].ToString());  
        connection.Close();  
    }  
    catch (Exception e)  
    {  
        return e.Message;  
    }  
    foreach (string dataRow in userData)  
    {  
        string[] dataCols = dataRow.Split('|');  
        if (dataCols[0].Equals(username))  
            if (dataCols[1].Equals(Auxiliary.HashPassword(rawPassword, Convert.FromBase64String(dataCols[2]))))  
                return null;  
            else  
                return "Λάθος κωδικός.";  
    }  
    return "Ο χρήστης δεν υπάρχει.";  
}  
  
//Used for logging in for teachers. Check if given username exists and if it does, then creates the hashed password anew and compares it with the one at the DB.  
//If username and password match, then he logs in.  
public string AuthenticateTeacherCredentials(string rawPassword)  
{  
    string query = "select username, hashed_password, salt from users where teacher = true";  
    List<string> userData = new List<string>(); //store user's username, hashed password and salt  
    try  
    {  
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);  
        connection.Open();  
        NpgsqlCommand command = new NpgsqlCommand(query, connection);  
        NpgsqlDataReader dataReader = command.ExecuteReader(); //run query  
        while (dataReader.Read())  
            userData.Add(dataReader[0].ToString() + " | " + dataReader[1].ToString() + " | " + dataReader[2].ToString());  
        connection.Close();  
    }  
    catch (Exception e)  
    {  
        return e.Message;  
    }  
    foreach (string dataRow in userData)  
    {  
        string[] dataCols = dataRow.Split('|');  
        if (dataCols[0].Equals(username))  
            if (dataCols[1].Equals(Auxiliary.HashPassword(rawPassword, Convert.FromBase64String(dataCols[2]))))  
                return null;  
            else  
                return "Λάθος κωδικός.";  
    }  
    return "Ο χρήστης δεν υπάρχει.";  
}=====
```



Τέλος, έχουμε υλοποιήσει μια βοηθητική κλάση Auxiliary που περιέχει μερικές βοηθητικές μεθόδους, καθώς και το connection string για την σύνδεση με τη βάση.

```
namespace EkipaideutikoLogismiko
{
    public class Auxiliary
    {
        public static readonly string CONNECTION_STRING = "Host=127.0.0.1;Port=5432;User ID=postgres;Password=maxrouso2;Database=elearning;"
```

Οι μέθοδοι της κλάσης Auxiliary αναλύονται παρακάτω.

- GenerateSalt() και HashPassword()

Αυτές οι μέθοδοι χρησιμοποιούνται για την κρυπτογράφηση του κωδικού των χρηστών.

```
//---Hash Password---
//-----
//Generate a salt to hash the user's password.
public static byte[] GenerateSalt()
{
    byte[] salt = new byte[32]; //salt length is 32(can be changed)
    using (var random = new RNGCryptoServiceProvider()) //"RNGCryptoServiceProvider" object is disposed, there are no resource leaks
    {
        random.GetNonZeroBytes(salt); //salt
    }
    return salt;
}

//Hashes the user's password along with the salt.
public static string HashPassword(string rawPassword, byte[] salt)
{
    byte[] password = Encoding.UTF8.GetBytes(rawPassword); //convert to bytes
    byte[] passwordWithSalt = new byte[password.Length + salt.Length];
    for (int i = 0; i < password.Length; i++) //copy password bytes
    {
        passwordWithSalt[i] = password[i];
    }
    for (int i = 0; i < salt.Length; i++) //copy salt bytes
    {
        passwordWithSalt[i + password.Length] = salt[i];
    }
    using (SHA512 shaM = new SHA512Managed()) //"SHA512Managed" object is disposed, there are no resource leaks
    {
        return Convert.ToBase64String(shaM.ComputeHash(passwordWithSalt)); //hashed password (with salt)
    }
}
//-----
```

- CheckForDuplicates() και CheckRegex()

Αυτές οι μέθοδοι χρησιμοποιούνται κατά την διαδικασία σύνδεσης των χρηστών, ελέγχοντας για διπλότυπες εγγραφές στην βάση και ελέγχοντας το input του χρήστη να είναι στην κατάλληλη μορφή.



```
-----Sign Up-----
=====
// Username is pk
public static string CheckForDuplicates(string username)
{
    string query = "select username from users";
    List<string> usernamesDB = new List<string>(); //store usernames
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(CONNECTION_STRING);
        connection.Open();
        NpgsqlCommand command = new NpgsqlCommand(query, connection);
        NpgsqlDataReader dataReader = command.ExecuteReader(); //run query
        while (dataReader.Read())
        {
            usernamesDB.Add(dataReader[0].ToString());
        }
        connection.Close();
    }
    catch
    {
        return "Υπήρξε κάποιο πρόβλημα.";
    }

    foreach (string usernameDB in usernamesDB) //check for duplicates (username)
    {
        if (usernameDB.Equals(username))
            return "Ο χρήστης υπάρχει ήδη.";
    }
    return null;
}

// Check user's input
public static string CheckRegex(string firstname, string lastname, string username, string password)
{
    if (!Regex.IsMatch(firstname, @"^[a-zA-Z]+$") || !Regex.IsMatch(lastname, @"^[a-zA-Z]+$"))
    {
        return "Το όνομα πρέπει να περιέχει μόνο γράμματα.";
    }
    if (password.Trim().Length < 6)
    {
        return "Ο κωδικός πρέπει να είναι μεγαλύτερος ή ίσος με 6 χαρακτήρες.";
    }
    if (Regex.IsMatch(username, @"^[0-9]+$"))
    {
        return "Το όνομα χρήστη δεν μπορεί να περιέχει μόνο αριθμούς.";
    }
    return null;
}
```



- IsValidUserInputLogin()

Η συγκεκριμένη μέθοδος χρησιμοποιείται κατά τη διαδικασία σύνδεσης των χρηστών και συγκεκριμένα ελέγχει την καταλληλότητα του user input.

```
//----Log In----  
//-----  
  
//Validate user's input.  
public static string IsValidUserInputLogIn(string username, string password)  
{  
    if (username.Trim().Length == 0)  
        return "Παρακαλώ συμπληρώστε όλα τα πεδία.";  
    if (password.Trim().Length < 6)  
        return "Ο κωδικός πρέπει να είναι μεγαλύτερος ή ίσος με 6 χαρακτήρες.";  
    return null;  
}  
//-----
```

**Σημείωση:** Λόγω του μεγάλου όγκου των κώδικα καθώς και για λόγους εξοικονόμησης χώρου, για τα επόμενα παραδείγματα, σας παραθέτουμε σε screenshot μόνο τον κώδικα για τον υπολογισμό της βαθμολογίας και εγγραφής της στην βάση. Ο πλήρες κώδικας μαζί με επεξηγηματικά σχόλια βρίσκεται στο παραδοτέο αρχείο.

## Επεξηγηση κώδικα για το τεστ εύκολης εξάσκησης

Τα τεστ εύκολης εξάσκησης αποτελούνται από 5 τυχαίες ερωτήσεις σωστού –λάθους και πολλαπλής επιλογής.

Αρχικά, επιλέγεται ένας τυχαίος αριθμός μεταξύ του 1 και του 4, ο οποίος αντιστοιχεί στο τυχαίο πλήθος των ερωτήσεων τύπου σωστού λάθους. Ο υπολειπόμενος αριθμός ερωτήσεων αντιστοιχεί στο τυχαίο πλήθος των ερωτήσεων τύπου πολλαπλής επιλογής.

Στη συνέχεια, φορτώνουμε όλες τις ερωτήσεις τύπου σωστού –λάθους και πολλαπλής επιλογής από την βάση δεδομένων σε δύο λίστες τύπου TrueFalse και MultipleChoice αντίστοιχα. Το περιεχόμενο των λιστών αυτών ταξινομείται με τυχαίο τρόπο και διαλέγουμε τις πρώτες n τυχαίες ερωτήσεις από κάθε λίστα (όπου n ο τυχαίος αριθμός που ορίστηκε προηγουμένως για το πλήθος του κάθε τύπου ερωτήσεων).

Έπειτα, οι 5 τυχαία επιλεγμένες ερωτήσεις εισάγονται και ταξινομούνται τυχαία σε μία καινούρια λίστα. Κατ' αυτό τον τρόπο, η λίστα αυτή περιέχει τις ερωτήσεις που έχουν επιλεχτεί και θα εμφανιστούν τελικά στον χρήστη.

Για κάθε ερώτηση που απαντάει ο χρήστης, πρώτα γίνεται έλεγχος για το αν απάντησε σωστά ή όχι, ενημερώνοντάς τον κατάλληλα για το αν απάντησε σωστά ή όχι, και μετά προχωράει στην επόμενη ερώτηση.

Μετά την απάντηση και των 5 ερωτήσεων από τον χρήστη, το πρόγραμμα υπολογίζει την βαθμολογία του. Ελέγχουμε για το αν η απάντηση του χρήστη ταυτίζεται με τη σωστή απάντηση που έχει ανακτηθεί από την βάση δεδομένων για κάθε μία ερώτηση του τεστ. Αν οι απαντήσεις ταυτίζονται, σημαίνει ότι έχει απαντήσει σωστά, διαφορετικά έχει απαντήσει λανθασμένα.

Τέλος, ενημερώνονται οι εγγραφές του πίνακα exercisehistory καταλλήλως.



```
if (i == 5) // User answered all the questions go back to main menu
{
    // Calculate the results
    int score = 0;
    List<bool> scores = new List<bool>();
    for (int i = 0; i < 5; i++)
    {
        if (answers.ElementAt(i).Equals(rightAnswers.ElementAt(i)))
        {
            score++;
            scores.Add(true);
        }
        else
        {
            scores.Add(false);
        }
    }

    // Write the results of the test to database
    int rowsAffected = -1; // False value
    string query = "Insert into exercisehistory values (@username, @chapter, @difficulty, @id1, @id2, @id3, @id4, @id5, @answer1, @answer2, @answer3, @answer4, @answer5)" +
    "on conflict (username, chapter, difficulty) do update set id1 = @id1, id2 = @id2, id3 = @id3, id4 = @id4, id5 = @id5, answer1 = @answer1, answer2 = @answer2, answer3 = @answer3, answer4 = @answer4, answer5 = @answer5";
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);
        connection.Open();
        //define query's parameters
        NpgsqlCommand command = new NpgsqlCommand(query, connection);
        command.Parameters.AddWithValue("username", username);
        command.Parameters.AddWithValue("chapter", 1);
        command.Parameters.AddWithValue("difficulty", 0);
        command.Parameters.AddWithValue("id1", qids.ElementAt(0));
        command.Parameters.AddWithValue("id2", qids.ElementAt(1));
        command.Parameters.AddWithValue("id3", qids.ElementAt(2));
        command.Parameters.AddWithValue("id4", qids.ElementAt(3));
        command.Parameters.AddWithValue("id5", qids.ElementAt(4));
        command.Parameters.AddWithValue("answer1", scores.ElementAt(0));
        command.Parameters.AddWithValue("answer2", scores.ElementAt(1));
        command.Parameters.AddWithValue("answer3", scores.ElementAt(2));
        command.Parameters.AddWithValue("answer4", scores.ElementAt(3));
        command.Parameters.AddWithValue("answer5", scores.ElementAt(4));
        rowsAffected = command.ExecuteNonQuery(); //run query
        connection.Close();
    }
    catch
    {
        MessageBox.Show("Κάτι πήγε λάθος", "Αποτυχία", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    if (rowsAffected == -1)
    {
        MessageBox.Show("Κάτι πήγε λάθος, ξαναπροσάθηκε αργότερα.", "Αποτυχία", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    else
    {
        MessageBox.Show("Η απάντησή σου καταχωρίθηκε!", "Επιτυχία", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        MainStudentMenu form = new MainStudentMenu(username);
        form.Show();
        this.Hide();
    }
}
```

## Επεξήγηση κώδικα για το τεστ δύσκολης εξάσκησης

Τα τεστ δύσκολης εξάσκησης αποτελούνται από 5 τυχαίες ερωτήσεις συμπλήρωσης κενών και τοποθέτησης των εντολών/ερωτημάτων σε σωστή σειρά.

Αρχικά, επιλέγεται ένας τυχαίος αριθμός μεταξύ του 1 και του 4, ο οποίος αντιστοιχεί στο τυχαίο πλήθος των ερωτήσεων τύπου τοποθέτησης των εντολών/ερωτημάτων σε σωστή σειρά. Ο υπολειπόμενος αριθμός ερωτήσεων αντιστοιχεί στο τυχαίο πλήθος των ερωτήσεων τύπου συμπλήρωσης κενών.

Στη συνέχεια, φορτώνουμε όλες τις ερωτήσεις τύπου συμπλήρωσης κενών και τοποθέτησης των εντολών/ερωτημάτων σε σωστή σειρά από την βάση δεδομένων σε δύο λίστες τύπου FillGap και Sequence αντίστοιχα. Το περιεχόμενο των λιστών αυτών ταξινομείται με τυχαίο τρόπο και διαλέγουμε τις πρώτες n τυχαίες ερωτήσεις από κάθε λίστα (όπου n ο τυχαίος αριθμός που ορίστηκε προηγουμένως για το πλήθος του κάθε τύπου ερωτήσεων).

Έπειτα, οι 5 τυχαία επιλεγμένες ερωτήσεις εισάγονται και ταξινομούνται τυχαία σε μία καινούρια λίστα. Κατ' αυτό τον τρόπο, η λίστα αυτή περιέχει τις ερωτήσεις που έχουν επιλεχτεί και θα εμφανιστούν τελικά στον χρήστη.

Για κάθε ερώτηση που απαντάει ο χρήστης, πρώτα γίνεται έλεγχος για το αν απάντησε σωστά ή όχι, ενημερώνοντάς τον κατάλληλα για το αν απάντησε σωστά ή όχι, και μετά προχωράει στην επόμενη ερώτηση.

Μετά την απάντηση και των 5 ερωτήσεων από τον χρήστη, το πρόγραμμα υπολογίζει την βαθμολογία του. Ελέγχουμε για το αν η απάντηση του χρήστη ταυτίζεται με τη σωστή απάντηση που έχει ανακτηθεί από την βάση δεδομένων για κάθε μία ερώτηση του τεστ. Αν οι απαντήσεις ταυτίζονται, σημαίνει ότι έχει απαντήσει σωστά, διαφορετικά έχει απαντήσει λανθασμένα.

Τέλος, ενημερώνονται οι εγγραφές του πίνακα exercisehistory καταλλήλως.



```
if (i == 5) // User answered all the questions go back to main menu
{
    // Calculate the results
    int score = 0;
    List<bool> scores = new List<bool>();
    for (int i = 0; i < 5; i++)
    {
        if (answers.ElementAt(i).Equals(rightAnswers.ElementAt(i)))
        {
            score++;
            scores.Add(true);
        }
        else
        {
            scores.Add(false);
        }
    }

    // Write the results of the test to database
    int rowsAffected = -1; // false value
    string query = "Insert into exercisehistory values (@username, @chapter, @difficulty, @id1, @id2, @id3, @id4, @id5, @answer1, @answer2, @answer3, @answer4, @answer5)" +
        "on conflict (username, chapter, difficulty) do update set id1 = @id1, id2 = @id2, id3 = @id3, id4 = @id4, id5 = @id5, answer1 = @answer1, answer2 = @answer2, answer3 = @answer3, answer4 = @answer4, answer5 = @answer5";
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);
        connection.Open();
        //define query's parameters
        NpgsqlCommand command = new NpgsqlCommand(query, connection);
        command.Parameters.AddWithValue("username", username);
        command.Parameters.AddWithValue("chapter", 1);
        command.Parameters.AddWithValue("difficulty", 1);
        command.Parameters.AddWithValue("id1", qids.ElementAt(0));
        command.Parameters.AddWithValue("id2", qids.ElementAt(1));
        command.Parameters.AddWithValue("id3", qids.ElementAt(2));
        command.Parameters.AddWithValue("id4", qids.ElementAt(3));
        command.Parameters.AddWithValue("id5", qids.ElementAt(4));
        command.Parameters.AddWithValue("answer1", scores.ElementAt(0));
        command.Parameters.AddWithValue("answer2", scores.ElementAt(1));
        command.Parameters.AddWithValue("answer3", scores.ElementAt(2));
        command.Parameters.AddWithValue("answer4", scores.ElementAt(3));
        command.Parameters.AddWithValue("answer5", scores.ElementAt(4));
        rowsAffected = command.ExecuteNonQuery(); //run query
        connection.Close();
    }
    catch
    {
        MessageBox.Show("Κάτι πήγε λάθος", "Ανοτυχία", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    if (rowsAffected == -1)
    {
        MessageBox.Show("Κάτι πήγε λάθος, ξαναπροσάρθρωσε αργότερα.", "Ανοτυχία", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    else
    {
        MessageBox.Show("Η απάντηση σου καταχωρίθηκε!", "Επιτυχία", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        MainStudentMenu form = new MainStudentMenu(username);
        form.Show();
        this.Hide();
    }
}
```

## Επεξήγηση κώδικα για το τελικό τεστ αξιολόγησης

Το τελικό τεστ αξιολόγησης αποτελούνται από 12 τυχαίες ερωτήσεις όλων των ειδών.

Αρχικά, φορτώνουμε όλες τις ερωτήσεις από την βάση δεδομένων σε τέσσερεις λίστες τύπου TrueFalse, MultipleChoice, FillGap και Sequence. Το περιεχόμενο των λιστών αυτών ταξινομείται με τυχαίο τρόπο και διαλέγουμε τις πρώτες 3 τυχαίες ερωτήσεις από κάθε λίστα.

Έπειτα, οι 12 τυχαία επιλεγμένες ερωτήσεις εισάγονται και ταξινομούνται τυχαία σε μία καινούρια λίστα. Κατ' αυτό τον τρόπο, η λίστα αυτή περιέχει τις ερωτήσεις που έχουν επιλεχτεί και θα εμφανιστούν τελικά στον χρήστη.

Μετά την απάντηση και των 12 ερωτήσεων από τον χρήστη, το πρόγραμμα υπολογίζει την βαθμολογία του. Ελέγχουμε για το αν η απάντηση του χρήστη ταυτίζεται με τη σωστή απάντηση που έχει ανακτηθεί από την βάση δεδομένων για κάθε μία ερώτηση του τεστ. Αν οι απαντήσεις ταυτίζονται, σημαίνει ότι έχει απαντήσει σωστά, διαφορετικά έχει απαντήσει λανθασμένα.

Τέλος, εισάγεται μία εγγραφή στον πίνακα testhistory καταλλήλως.



```
if (i == 12) // User answered all the questions go back to main menu
{
    // Calculate the results
    int score = 0;
    List<bool> scores = new List<bool>();
    for (int i = 0; i < 12; i++)
    {
        if (answers.ElementAt(i).Equals(rightAnswers.ElementAt(i)))
        {
            score++;
            scores.Add(true);
        }
        else
        {
            scores.Add(false);
        }
    }
    // Write the results of the test to database
    int rowsAffected = -1; // false value
    string query = "insert into testhistory values (@username, @date_time, @id1, @id2, @id3, @id4, @id5, @id6, @id7, @id8, @id9, @id10, @id11, @id12, " +
        "@answer1, @answer2, @answer3, @answer4, @answer5, @answer6, @answer7, @answer8, @answer9, @answer10, @answer11, @answer12)";
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);
        connection.Open();
        //define query's parameters
        NpgsqlCommand command = new NpgsqlCommand(query, connection);
        command.Parameters.AddWithValue("username", username);
        command.Parameters.AddWithValue("date_time", DateTime.Now.ToString("dd-MM-yyyy HH:mm:ss"));
        command.Parameters.AddWithValue("id1", qids.ElementAt(0));
        command.Parameters.AddWithValue("id2", qids.ElementAt(1));
        command.Parameters.AddWithValue("id3", qids.ElementAt(2));
        command.Parameters.AddWithValue("id4", qids.ElementAt(3));
        command.Parameters.AddWithValue("id5", qids.ElementAt(4));
        command.Parameters.AddWithValue("id6", qids.ElementAt(5));
        command.Parameters.AddWithValue("id7", qids.ElementAt(6));
        command.Parameters.AddWithValue("id8", qids.ElementAt(7));
        command.Parameters.AddWithValue("id9", qids.ElementAt(8));
        command.Parameters.AddWithValue("id10", qids.ElementAt(9));
        command.Parameters.AddWithValue("id11", qids.ElementAt(10));
        command.Parameters.AddWithValue("id12", qids.ElementAt(11));
        command.Parameters.AddWithValue("answer1", scores.ElementAt(0));
        command.Parameters.AddWithValue("answer2", scores.ElementAt(1));
        command.Parameters.AddWithValue("answer3", scores.ElementAt(2));
        command.Parameters.AddWithValue("answer4", scores.ElementAt(3));
        command.Parameters.AddWithValue("answer5", scores.ElementAt(4));
        command.Parameters.AddWithValue("answer6", scores.ElementAt(5));
        command.Parameters.AddWithValue("answer7", scores.ElementAt(6));
        command.Parameters.AddWithValue("answer8", scores.ElementAt(7));
        command.Parameters.AddWithValue("answer9", scores.ElementAt(8));
        command.Parameters.AddWithValue("answer10", scores.ElementAt(9));
        command.Parameters.AddWithValue("answer11", scores.ElementAt(10));
        command.Parameters.AddWithValue("answer12", scores.ElementAt(11));
        rowsAffected = command.ExecuteNonQuery(); //run query
        connection.Close();
    }
    catch
    {
        MessageBox.Show("Κάτι πήγε λάθος", "Αποτυχία", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    if (rowsAffected == -1)
    {
        MessageBox.Show("Κάτι πήγε λάθος, ξαναπροσάθησε αργότερα.", "Αποτυχία", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    else
    {
        MessageBox.Show("Η απάντησή σου καταχωρήθηκε!", "Επιτυχία", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        MainStudentMenu form = new MainStudentMenu(username);
        form.Show();
        this.Hide();
    }
}
```



### 3.3 C# ASP.NET Web Project

Έχουμε υλοποιήσει μερικές κλάσεις για την ανάπτυξη της web εφαρμογής μας.

Η κλάση που χρησιμοποιούμε για την δημιουργία αντικειμένων των χρηστών καθώς και την διαχείριση μερικών λειτουργιών όπως για παράδειγμα τις λειτουργίες της σύνδεσης και της εγγραφής είναι η κλάση User.

```
public class User
{
    public string first_name;
    public string last_name;
    public bool teacher;
    public string username;
    public string hashed_password;
    public string salt;

    //----Constructors----
    public User(string first_name, string last_name, string username, string hashed_password, string salt, bool teacher)
    {
        this.first_name = first_name;
        this.last_name = last_name;
        this.teacher = teacher;
        this.username = username;
        this.hashed_password = hashed_password;
        this.salt = salt;
    }

    public User(string username)
    {
        this.username = username;
    }
}
```

Οι μέθοδοι της κλάσης User αναλύονται παρακάτω.

- AuthenticateCredentials() και IsAdmin()

Χρησιμοποιούνται για την διαδικασία σύνδεσης των χρηστών.

```
//----Methods----
//-----
//Used for logging in. Check if given username exists and if it does,then creates the hashed password anew and compares it with the one at the DB.
//If username and password match, then he logs in.
public string AuthenticateCredentials(string rawPassword)
{
    string query = "select username, hashed_password, salt from users";
    List<string> userData = new List<string>(); //store user's username, hashed password and salt
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);
        connection.Open();
        NpgsqlCommand command = new NpgsqlCommand(query, connection);
        NpgsqlDataReader dataReader = command.ExecuteReader(); //run query
        while (dataReader.Read())
            userData.Add(dataReader[0].ToString() + " | " + dataReader[1].ToString() + " | " + dataReader[2].ToString());
        connection.Close();
    }
    catch (Exception e)
    {
        return e.Message;
    }
    foreach (string dataRow in userData)
    {
        string[] dataCols = dataRow.Split('|');
        if (dataCols[0].Equals(username))
            if (dataCols[1].Equals(Auxiliary.HashPassword(rawPassword, Convert.FromBase64String(dataCols[2]))))
                return null;
            else
                return "Incorrect password.";
    }
    return "Username does not exist.";
}
```



```
//Check if user is admin.
public string IsAdmin()
{
    string isAdmin = null;
    string query = "select teacher from users where username = @username";
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);
        connection.Open();
        NpgsqlCommand command = new NpgsqlCommand(query, connection);
        command.Parameters.AddWithValue("username", username);
        NpgsqlDataReader dataReader = command.ExecuteReader(); //run query
        while (dataReader.Read())
            isAdmin = dataReader[0].ToString(); //get result
        connection.Close();
        return isAdmin;
    }
    catch
    {
        return null;
    }
}
```

Επιπλέον, έχουμε υλοποιήσει μια βοηθητική κλάση Auxiliary που περιέχει μερικές βοηθητικές μεθόδους, καθώς και το connection string για την σύνδεση με τη βάση.

```
namespace PyLearn
{
    public class Auxiliary
    {
        public static readonly string CONNECTION_STRING = "Host=127.0.0.1;Port=5432;User ID=postgres;Password=maxrouso2;Database=elearning;";
```

Οι μέθοδοι της κλάσης Auxiliary αναλύονται παρακάτω.

- GenerateSalt() και HashPassword()

Αυτές οι μέθοδοι χρησιμοποιούνται για την κρυπτογράφηση του κωδικού των χρηστών.

```
-----Hash Password-----
//-----
//Generate a salt to hash the user's password.
public static byte[] GenerateSalt()
{
    byte[] salt = new byte[32]; //salt length is 32(can be changed)
    using (var random = new RNGCryptoServiceProvider()) // "RNGCryptoServiceProvider" object is disposed, there are no resource leaks
    {
        random.GetNonZeroBytes(salt); //salt
    }
    return salt;
}

//Hashes the user's password along with the salt.
public static string HashPassword(string rawPassword, byte[] salt)
{
    byte[] password = Encoding.UTF8.GetBytes(rawPassword); //convert to bytes
    byte[] passwordWithSalt = new byte[password.Length + salt.Length];
    for (int i = 0; i < password.Length; i++) //copy password bytes
    {
        passwordWithSalt[i] = password[i];
    }
    for (int i = 0; i < salt.Length; i++) //copy salt bytes
    {
        passwordWithSalt[i + password.Length] = salt[i];
    }
    using (SHA512 shaM = new SHA512Managed()) // "SHA512Managed" object is disposed, there are no resource leaks
    {
        return Convert.ToBase64String(shaM.ComputeHash(passwordWithSalt)); //hashed password (with salt)
    }
}
-----
```



- IsValidUserInputLogin()

Η συγκεκριμένη μέθοδος χρησιμοποιείται κατά τη διαδικασία σύνδεσης των χρηστών και συγκεκριμένα ελέγχει την καταλληλότητα του user input.

```
//----Log In----  
//=====  
  
//Validate user's input.  
public static string IsValidUserInputLogIn(string username, string password)  
{  
    if (username.Trim().Length == 0)  
        return "Παρακαλώ συμπληρώστε όλα τα πεδία.";  
    if (password.Trim().Length < 6)  
        return "Ο κωδικός πρέπει να είναι μεγαλύτερος ή ίσος με 6 χαρακτήρες.";  
    return null;  
}  
//=====
```

- GetOneProgressFromDB() και GetUsersProgressFromDB()

Αυτές οι μέθοδοι χρησιμοποιούνται για την ανάκτηση της προόδου των χρηστών ή ενός χρήστη, από την βάση δεδομένων.

```
//----Get Progress---  
//=====  
  
public static List<string> GetOneProgressFromDB()  
{  
    string query = "select * from exercisehistory where username='" + (string)HttpContext.Current.Session["Username"] + "'";  
    List<string> progressList = new List<string>(); //store plate_numberIDs  
    try  
    {  
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);  
        connection.Open();  
        NpgsqlCommand command = new NpgsqlCommand(query, connection);  
        NpgsqlDataReader dataReader = command.ExecuteReader(); //run query  
        while (dataReader.Read())  
        {  
            progressList.Add(dataReader[0].ToString());  
            progressList.Add(dataReader[1].ToString());  
            if (dataReader[2].ToString().Equals("0"))  
            {  
                progressList.Add("Εύκολο");  
            }  
            else  
            {  
                progressList.Add("Δύσκολο");  
            }  
            progressList.Add(dataReader[3].ToString());  
            progressList.Add(dataReader[4].ToString());  
            progressList.Add(dataReader[5].ToString());  
            progressList.Add(dataReader[6].ToString());  
            progressList.Add(dataReader[7].ToString());  
            progressList.Add(dataReader[8].ToString());  
            progressList.Add(dataReader[9].ToString());  
            progressList.Add(dataReader[10].ToString());  
            progressList.Add(dataReader[11].ToString());  
            progressList.Add(dataReader[12].ToString());  
        }  
        connection.Close();  
        return progressList;  
    }  
    catch  
    {  
        return null;  
    }  
}
```



```
public static List<string> GetUsersProgressFromDB()
{
    string query = "select * from exercisehistory";
    List<string> progressList = new List<string>(); //store plate_numberIDs
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);
        connection.Open();
        NpgsqlCommand command = new NpgsqlCommand(query, connection);
        NpgsqlDataReader dataReader = command.ExecuteReader(); //run query
        while (dataReader.Read())
        {
            progressList.Add(dataReader[0].ToString());
            progressList.Add(dataReader[1].ToString());
            if (dataReader[2].ToString().Equals("0"))
            {
                progressList.Add("Εύκολο");
            }
            else
            {
                progressList.Add("Δύσκολο");
            }
            progressList.Add(dataReader[3].ToString());
            progressList.Add(dataReader[4].ToString());
            progressList.Add(dataReader[5].ToString());
            progressList.Add(dataReader[6].ToString());
            progressList.Add(dataReader[7].ToString());
            progressList.Add(dataReader[8].ToString());
            progressList.Add(dataReader[9].ToString());
            progressList.Add(dataReader[10].ToString());
            progressList.Add(dataReader[11].ToString());
            progressList.Add(dataReader[12].ToString());
        }
        connection.Close();
        return progressList;
    }
    catch
    {
        return null;
    }
}
```

- GetStudentsFromDB()

Η μέθοδος χρησιμοποιείται για την ανάκτηση των μαθητών από την βάση δεδομένων.

```
public static List<string> GetStudentsFromDB(string username)
{
    string query = "select first_name, last_name, username from users where teacher=false";
    List<string> progressList = new List<string>(); //store plate_numberIDs
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);
        connection.Open();
        NpgsqlCommand command = new NpgsqlCommand(query, connection);
        NpgsqlDataReader dataReader = command.ExecuteReader(); //run query
        while (dataReader.Read())
        {
            // Don't return as a classmate the same user
            if (!username.Equals(dataReader[2].ToString()))
            {
                progressList.Add(dataReader[0].ToString());
                progressList.Add(dataReader[1].ToString());
            }
        }
        connection.Close();
        return progressList;
    }
    catch
    {
        return null;
    }
}
```



- GetOnesFinalTestProgressFromDB() και GetFinalTestProgressFromDB()

Αυτές οι μέθοδοι χρησιμοποιούνται για την ανάκτηση της προόδου των χρηστών ή ενός χρήστη, από την βάση δεδομένων, για το τελικό τεστ αξιολόγησης.

```
public static List<string> GetOnesFinalTestProgressFromDB()
{
    string query = "select username, date_time, answer1, answer2, answer3, answer4, answer5, answer6, answer7, answer8, answer9, answer10, answer11, answer12 from testhistory where username='"
    + (string)HttpContext.Current.Session["Username"] + "'";
    List<string> progressList = new List<string>(); //store plate_numberIDs
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);
        connection.Open();
        NpgsqlCommand command = new NpgsqlCommand(query, connection);
        NpgsqlDataReader dataReader = command.ExecuteReader(); //run query
        while (dataReader.Read())
        {
            progressList.Add(dataReader[0].ToString());
            progressList.Add(dataReader[1].ToString());
            progressList.Add(dataReader[2].ToString());
            progressList.Add(dataReader[3].ToString());
            progressList.Add(dataReader[4].ToString());
            progressList.Add(dataReader[5].ToString());
            progressList.Add(dataReader[6].ToString());
            progressList.Add(dataReader[7].ToString());
            progressList.Add(dataReader[8].ToString());
            progressList.Add(dataReader[9].ToString());
            progressList.Add(dataReader[10].ToString());
            progressList.Add(dataReader[11].ToString());
            progressList.Add(dataReader[12].ToString());
            progressList.Add(dataReader[13].ToString());
        }
        connection.Close();
        return progressList;
    }
    catch
    {
        return null;
    }
}

public static List<string> GetFinalTestProgressFromDB()
{
    string query = "select username, date_time, answer1, answer2, answer3, answer4, answer5, answer6, answer7, answer8, answer9, answer10, answer11, answer12 from testhistory";
    List<string> progressList = new List<string>(); //store plate_numberIDs
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);
        connection.Open();
        NpgsqlCommand command = new NpgsqlCommand(query, connection);
        NpgsqlDataReader dataReader = command.ExecuteReader(); //run query
        while (dataReader.Read())
        {
            progressList.Add(dataReader[0].ToString());
            progressList.Add(dataReader[1].ToString());
            progressList.Add(dataReader[2].ToString());
            progressList.Add(dataReader[3].ToString());
            progressList.Add(dataReader[4].ToString());
            progressList.Add(dataReader[5].ToString());
            progressList.Add(dataReader[6].ToString());
            progressList.Add(dataReader[7].ToString());
            progressList.Add(dataReader[8].ToString());
            progressList.Add(dataReader[9].ToString());
            progressList.Add(dataReader[10].ToString());
            progressList.Add(dataReader[11].ToString());
            progressList.Add(dataReader[12].ToString());
            progressList.Add(dataReader[13].ToString());
        }
        connection.Close();
        return progressList;
    }
    catch
    {
        return null;
    }
}
```



- GetErrors()

Η μέθοδος χρησιμοποιείται για την ανάκτηση των σφαλμάτων των χρηστών από την βάση δεδομένων.

```
public static List<string> GetErrors()
{
    string query = "select id, errortype from questions";
    List<string> idsList = new List<string>(); //store ids
    List<string> errorTypeList = new List<string>(); //store error types
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);
        connection.Open();
        NpgsqlCommand command = new NpgsqlCommand(query, connection);
        NpgsqlDataReader dataReader = command.ExecuteReader(); //run query
        while (dataReader.Read())
        {
            idsList.Add(dataReader[0].ToString());
            errorTypeList.Add(dataReader[1].ToString());
        }
        connection.Close();
    }
    catch
    {
        return null;
    }

    string query2 = "select id1, id2, id3, id4, id5, answer1, answer2, answer3, answer4, answer5, chapter from exercisehistory where username='"
        + (string)HttpContext.Current.Session["Username"] + "'";
    int syntaxErrors = 0;
    int logicalErrors = 0;
    int combErrors = 0;
    int chapter1Errors = 0;
    int chapter2Errors = 0;
    int chapter3Errors = 0;
    try
    {
        NpgsqlConnection connection = new NpgsqlConnection(Auxiliary.CONNECTION_STRING);
        connection.Open();
        NpgsqlCommand command = new NpgsqlCommand(query2, connection);
        NpgsqlDataReader dataReader = command.ExecuteReader(); //run query
        while (dataReader.Read())
        {
            System.Diagnostics.Debug.WriteLine("WHILE");
            for (int i=0; i < 5; i++)
            {
                System.Diagnostics.Debug.WriteLine("FOR");
                if (dataReader[i + 5].ToString().Equals("False"))
                {
                    //int position = idsList.FindIndex(a => a == dataReader[i].ToString());
                    int position = idsList.IndexOf(dataReader[i].ToString());
                    System.Diagnostics.Debug.WriteLine(position.ToString());
                    if (errorTypeList.ElementAt(position).Equals("0"))
                    {
                        syntaxErrors++;
                    }
                    else if (errorTypeList.ElementAt(position).Equals("1"))
                    {
                        logicalErrors++;
                    }
                    else
                    {
                        combErrors++;
                    }

                    if (dataReader[10].ToString().Equals("1"))
                    {
                        chapter1Errors++;
                    }
                    else if (dataReader[10].ToString().Equals("2"))
                    {
                        chapter2Errors++;
                    }
                    else
                    {
                        chapter3Errors++;
                    }
                }
            }
        }
        connection.Close();
    }
}
```

```
System.Diagnostics.Debug.WriteLine("WHILE");
for (int i=0; i < 5; i++)
{
    System.Diagnostics.Debug.WriteLine("FOR");
    if (dataReader[i + 5].ToString().Equals("False"))
    {
        //int position = idsList.FindIndex(a => a == dataReader[i].ToString());
        int position = idsList.IndexOf(dataReader[i].ToString());
        System.Diagnostics.Debug.WriteLine(position.ToString());
        if (errorTypeList.ElementAt(position).Equals("0"))
        {
            syntaxErrors++;
        }
        else if (errorTypeList.ElementAt(position).Equals("1"))
        {
            logicalErrors++;
        }
        else
        {
            combErrors++;
        }

        if (dataReader[10].ToString().Equals("1"))
        {
            chapter1Errors++;
        }
        else if (dataReader[10].ToString().Equals("2"))
        {
            chapter2Errors++;
        }
        else
        {
            chapter3Errors++;
        }
    }
}
connection.Close();
```

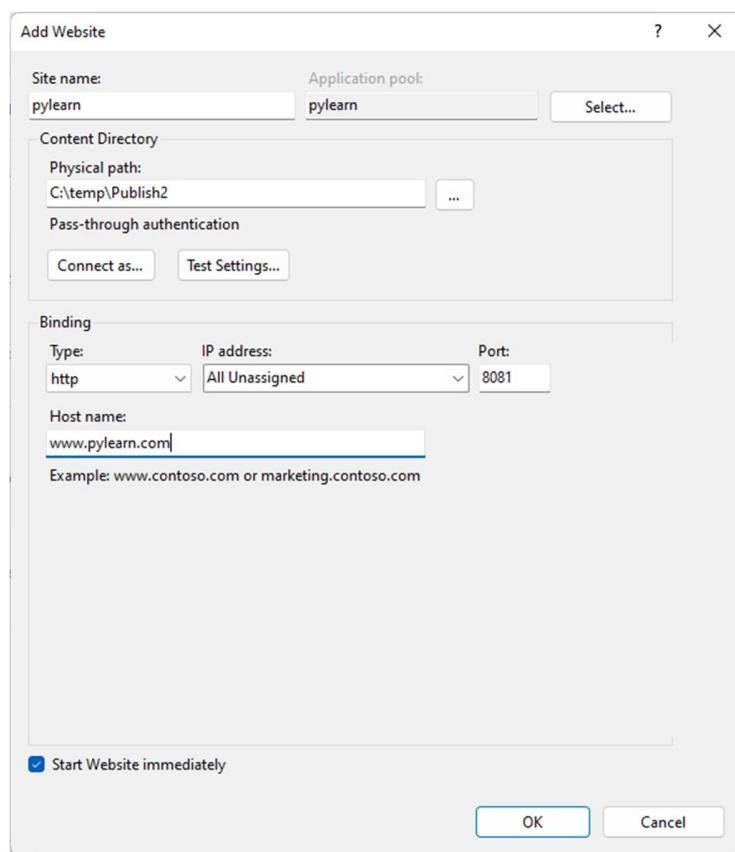


```
        }
        catch
        {
            return null;
        }

        List<string> results = new List<string>();
        results.Add(syntaxErrors.ToString());
        results.Add(logicalErrors.ToString());
        results.Add(combErrors.ToString());
        results.Add(chapter1Errors.ToString());
        results.Add(chapter2Errors.ToString());
        results.Add(chapter3Errors.ToString());
        return results;
    }
```

## 4 Εισαγωγή της web εφαρμογής στον IIS (Internet Information Services)

Καταρχάς, για να εγκαταστήσουμε την εφαρμογή μας στον IIS την κάναμε Publish μέσα από το περιβάλλον του Visual Studio. Στη συνέχεια, κάναμε προσθήκη της ιστοσελίδας μας μέσω του IIS Manager με τις παρακάτω ρυθμίσεις.





Με σκοπό να πληκτρολογούμε για την είσοδο στην εφαρμογή το domain name που ορίσαμε αντί για «localhost», τροποποιήσαμε το αρχείο hosts που βρίσκεται στο path C:\Windows\System32\drivers\etc\, όπως φαίνεται στο παρακάτω screenshot.

```
1 # Copyright (c) 1993-2009 Microsoft Corp.
2 #
3 # This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
4 #
5 # This file contains the mappings of IP addresses to host names. Each
6 # entry should be kept on an individual line. The IP address should
7 # be placed in the first column followed by the corresponding host name.
8 # The IP address and the host name should be separated by at least one
9 # space.
10 #
11 # Additionally, comments (such as these) may be inserted on individual
12 # lines or following the machine name denoted by a '#' symbol.
13 #
14 # For example:
15 #
16 #      102.54.94.97      rhino.acme.com          # source server
17 #      38.25.63.10      x.acme.com              # x client host
18 #
19 # localhost name resolution is handled within DNS itself.
20 #      127.0.0.1      localhost
21 #      ::1            localhost
22 #
23 [REDACTED]
24 127.0.0.1      www.pylearn.com
```

Για την δημιουργία ενός SSL πιστοποιητικού τρέξαμε την παρακάτω εντολή στο PowerShell με δικαιώματα διαχειριστή.

```
[+] Administrator: Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> New-SelfSignedCertificate -DnsName "www.pylearn.com" -CertStoreLocation "cert:\LocalMachine\My"

PSParentPath: Microsoft.PowerShell.Security\Certificate::LocalMachine\My

Thumbprint                               Subject
-----                               -----
78D5C898661E8710BEF477A9F0C6C4C2C7C7EEAF CN=www.pylearn.com

PS C:\WINDOWS\system32>
```



Το αποτέλεσμα της εντολής ήταν η δημιουργία ενός αυτό-υπογεγραμμένου πιστοποιητικού, το οποίο μπορούμε να το εντοπίσουμε μέσω της εφαρμογής mmc.exe. Πιο συγκεκριμένα βρίσκεται στον υπο-φάκελο «Πιστοποιητικά», του φακέλου «Προσωπικός χώρος αποθήκευσης».

The screenshot shows the Windows Certificate Store ( mmc.exe ) with the following details:

**Κονσόλα1 - [Κονσόλα ρίζα\Πιστοποιητικά (τοπικός υπολογιστή)\Προσωπικός χώρος αποθήκευσης\Πιστοποιητικά]**

**Αρχείο Ενέργεια Προβολή Αγορημένα Παράθυρο Βοήθεια**

**Κάτοχος** ^ Εκδόθηκε από Ημερομηνία λ... Συγκεκριμένες χρή... Φύλικό όνομα

Κάτοχος	Εκδόθηκε από	Ημερομηνία λ...	Συγκεκριμένες χρή...	Φύλικό όνομα
localhost	localhost	25/1/2023	Έλεγχος ταυτότη...	<None>
localhost	localhost	8/6/2026	Έλεγχος ταυτότη...	IIS Express Develop...
www.pylearn.com	www.pylearn.com	2/6/2023	Έλεγχος ταυτότη...	<None>
www.pylearn.com	www.pylearn.com	2/7/2023	Έλεγχος ταυτότη...	<None>

**Ενέργειες**

Πιστοποιητικά

Περισσότερες ενέργειες

**Πιστοποιητικό**

**Γενικά Λεπτομέρειες Διαδρομή πιστοποίησης**

**Πληροφορίες για το πιστοποιητικό**

**Το πιστοποιητικό προορίζεται για τους εξής σκοπούς:**

- Εγγύηση της ταυτότητάς σας σε ένα απομακρυσμένο υπολογιστή
- Εγγύηση της ταυτότητας ενός απομακρυσμένου υπολογιστή
- Όλες οι πολιτικές έκδοσης

**Κάτοχος:** www.pylearn.com

**Εκδόθηκε από:** www.pylearn.com

**Έγκυρο από:** 2/7/2022 έως 2/7/2023

**Ειδικά:** Έχετε ένα ιδιωτικό κλειδί που αντιστοιχεί σε αυτό το πιστοποιητικό.

Ένα πιστοποιητικό με το αντίστοιχο ίδιο Δήλωση εκδότη

OK

**Πιστοποιητικό**

**Γενικά Λεπτομέρεις Διαδρομή πιστοποίησης**

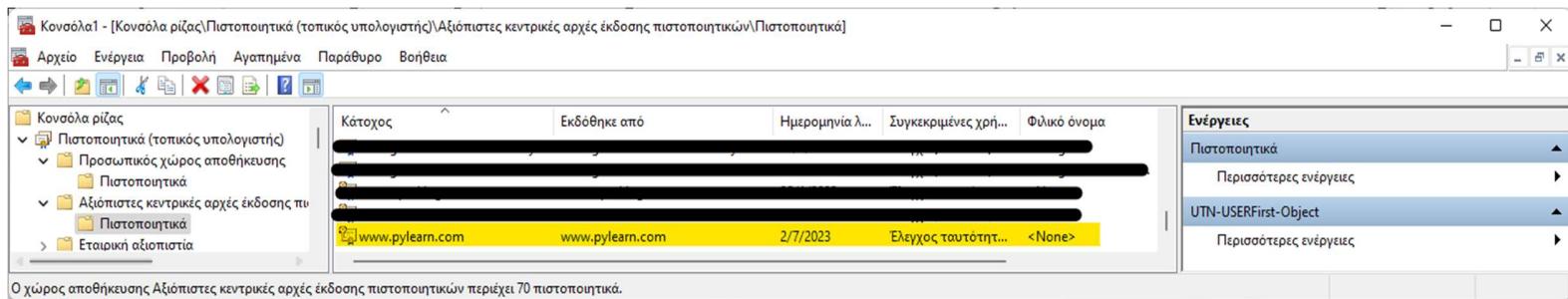
Εμφάνιση: <Όλα>

Πεδίο	Τιμή
Έκδοση	V3
Αριθμός σειράς	4fac69b5cacac4a644118bfa3...
Αλγόριθμος υπογραφής	sha256RSA
Αλγόριθμος κατακερματισμού	sha256
Εκδότης	www.pylearn.com
Έγκυρο από	Σάββατο, 2 Ιουλίου 2022 1:36...
Έγκυρο μέχρι	Κυριακή, 2 Ιουλίου 2023 1:56:...
Αρίθμ.	www.pylearn.com

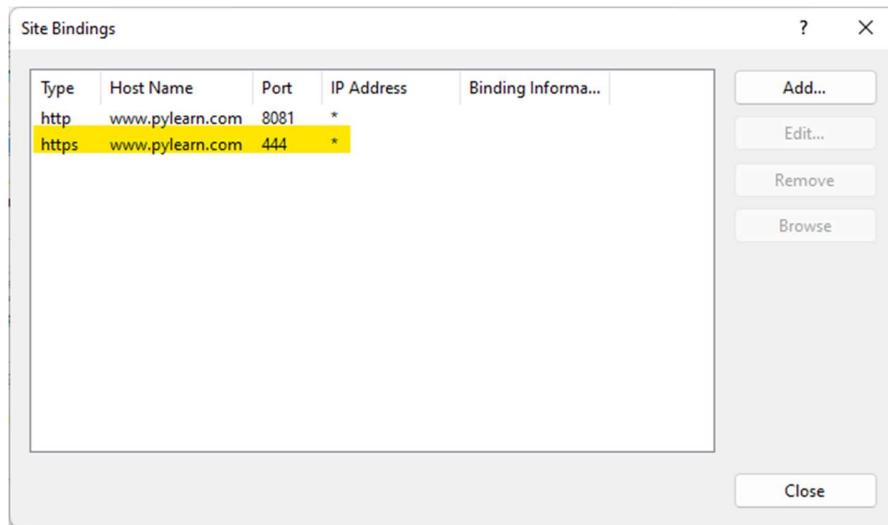
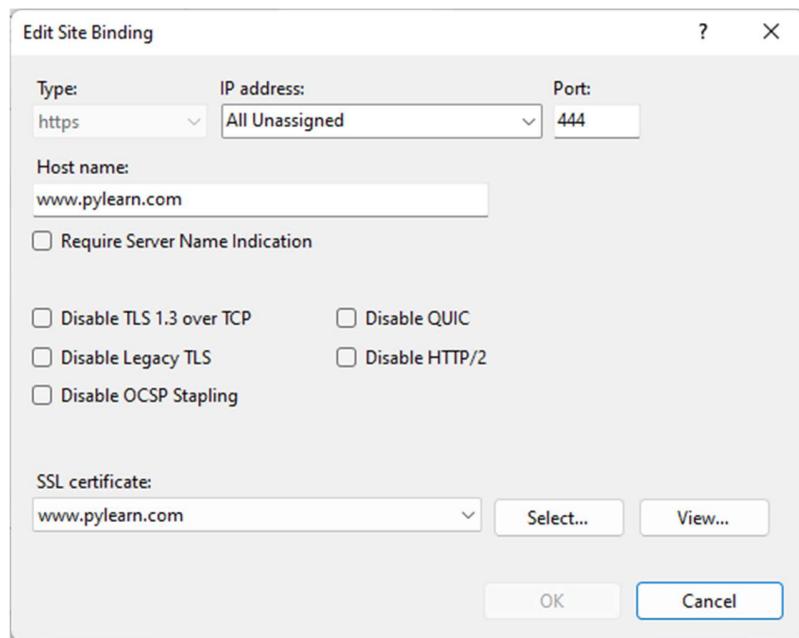
Επεξεργασία ιδιοτήτων... Αντηγραφή σε αρχείο... OK



Παρόλα αυτά, το πιστοποιητικό μας δεν θεωρείται έγκυρο αν δεν εισαχθεί στην λίστα «Αξιόπιστες κεντρικές αρχές έκδοσης πιστοποιητικών» στο mmc.exe. Αφού εισαχθεί, θα θεωρείται πλέον έγκυρο από τον browser.

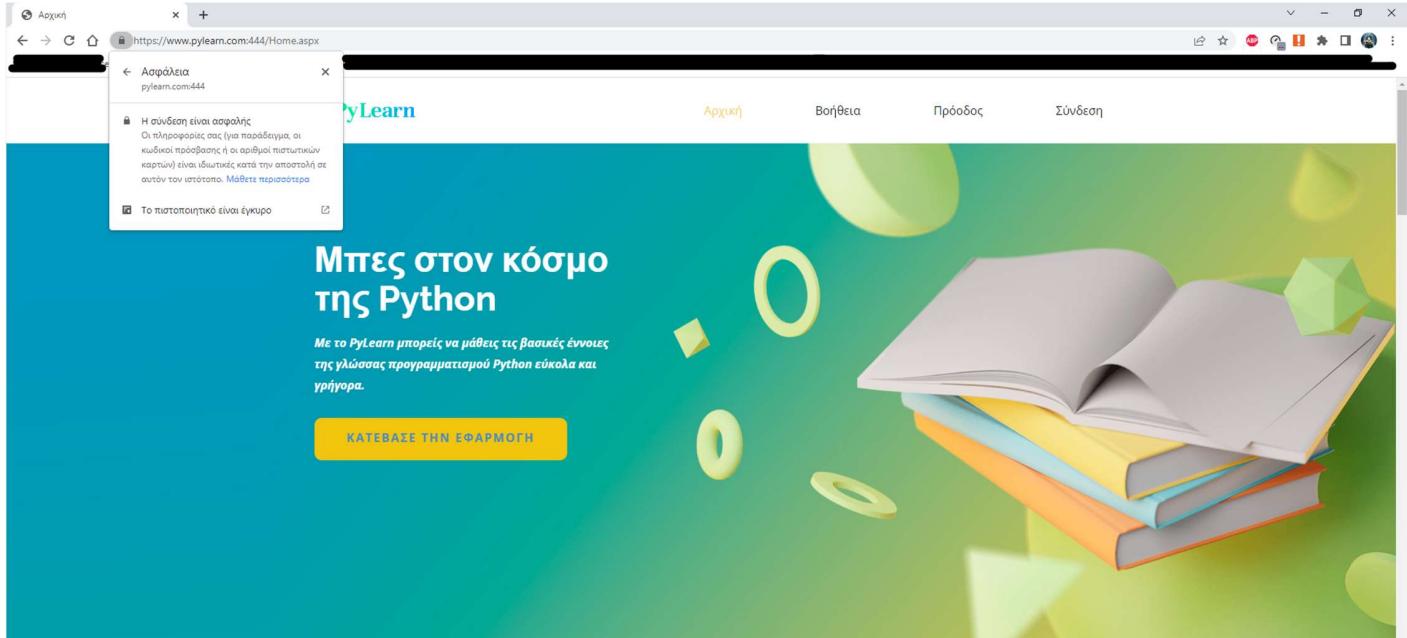


Το τελικό βήμα είναι να προσθέσουμε ένα https binding στην ιστοσελίδα μας μέσω του IIS.





Το αποτέλεσμα όλων των παραπάνω είναι όταν πληκτρολογούμε στον browser την διεύθυνση [www.pylearn.com](https://www.pylearn.com), να συνδεόμαστε στην ιστοσελίδα μας με ασφάλεια, όπως φαίνεται παρακάτω.



## 5 Βιβλιογραφικές Πηγές

### 1. Όλο το υλικό των εργαστηριακών διαλέξεων.