

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ



«Εκπαιδευτικό Λογισμικό για επαγγελματικό προσανατολισμό
φοιτητών και αποφοίτων Τμημάτων Πληροφορικής»

Εργασία στο μάθημα «Εκπαιδευτικό Λογισμικό»

Γόμπου Σοφία – Π18031
Ζιώγα Δανάη – Π18197

Πειραιάς 2024

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ	3
ΔΟΜΗ ΤΩΝ FORMS ΚΑΙ ΤΗΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ	4
1. HOME PAGE	5
2. SIGN UP	7
3. Menu & Διδακτικό Κείμενο	8
4. Tests	9
5. Orientation	11
6. Βάση Δεδομένων	12

ΕΙΣΑΓΩΓΗ

Η εφαρμογή είναι ένα αλληλεπιδραστικό λογισμικό εκπαίδευσης που απευθύνεται σε φοιτητές κι απόφοιτους του Τμήματος Πληροφορικής του Πανεπιστημίου Πειραιά.

Οι χρήστες δημιουργούν ένα λογαριασμό και έχουν πρόσβαση σε πληροφορίες που βασίζονται σε μαθήματα του προγράμματος σπουδών του Τμήματος. Έπειτα, περνάνε από τεστ, βασισμένα σε αυτά που διάβασαν. Στο τέλος ο χρήστης περνάει από επαναληπτικά τεστ εφ' όλης της ύλης. Αν η απόδοση δεν είναι ικανοποιητική, πρέπει να ξαναδιαβάσει την ύλη.

Τα σκορ των χρηστών αποθηκεύονται σε βάση δεδομένων, καθώς και η ημερομηνία που τα έκαναν κι η τελευταία σύνδεσή τους.

Η εφαρμογή, με βάση τα σκορ του κάθε τεστ, λαμβάνοντας υπόψιν τα τελευταία τεστ που έχει κάνει και τους βαθμούς που έχει λάβει στο κάθε μάθημα από τον καθηγητή, προτείνει στον χρήστη ποιόν κλάδο να ακολουθήσει και δίνει κάποιες πληροφορίες πάνω στο αντικείμενο.

Οι προτάσεις είναι καθαρά βασισμένες σε στατιστικά και νούμερα και σε καμία περίπτωση δεν είναι απόλυτες. Ο χρήστης μπορεί να πραγματοποιήσει την δικιά του έρευνα και να επιλέξει την μελλοντική του πορεία.

Για την διεκπεραίωση της εργασίας, χρησιμοποιήθηκε το Visual Studio 2019, pgAdmin 4 και έχουν χρησιμοποιηθεί οι γλώσσες C# και SQL.

ΔΟΜΗ ΤΩΝ FORMS ΚΑΙ ΤΗΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

Δίνεται αναλυτικά η λίστα με όλα τα forms που δημιουργήθηκαν για την εργασία.

1. **HomePage**
2. **SignUp**
3. **Menu**
4. **Frontend**
5. **Backend**
6. **Tests**
7. **Javatest**
8. **Htmltest**
9. **Javascripttest**
10. **Cstest**
11. **Frontend**
12. **Backend**
13. **Orientation**

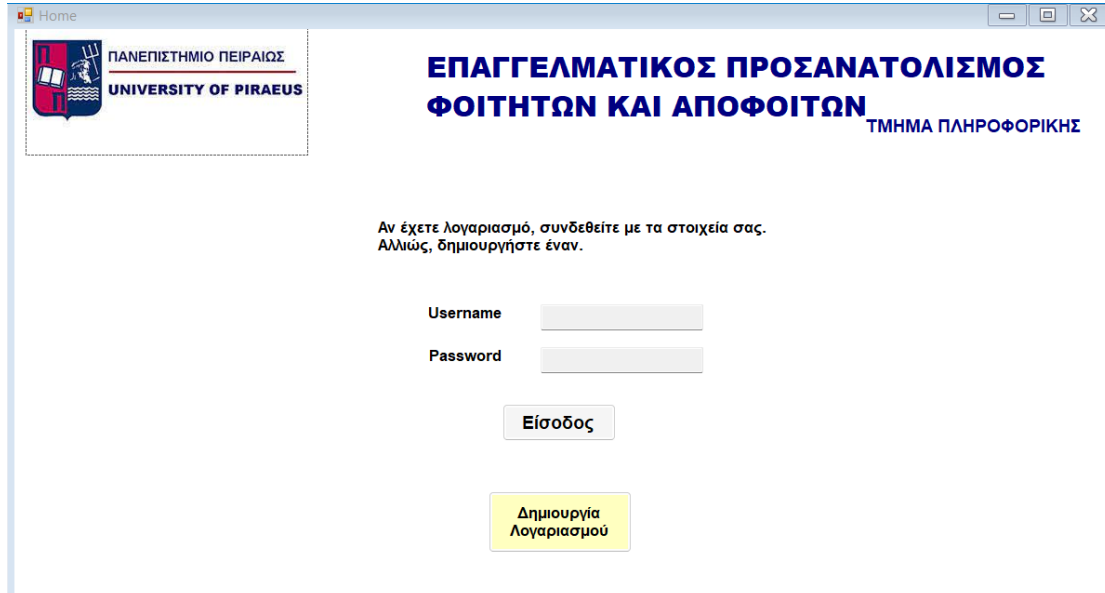
Η βάση δεδομένων, που εκτελείται στο pgadmin 4, αποτελείται από τους παρακάτω πίνακες με τις εξής στήλες:

1. **Users**
 - 1.1. **Userid (PK)**: Ένας αριθμός που αριθμεί τον κάθε εγγεγραμμένο χρήστη
 - 1.2. **Username**: Το username που εισάγει ο χρήστης για να εισέρχεται στην πλατφόρμα
 - 1.3. **Password**: Ο κωδικός πρόσβασης του κάθε χρήστη
 - 1.4. **Lastlogin**: Η ημερομηνία και ώρα που έκανε login ο χρήστης
2. **Usertests**
 - 2.1. **Usertestid (PK)**: Ένας αριθμός που αριθμεί κάθε τεστ του κάθε χρήστη
 - 2.2. **Userid (Foreign Key from table USERS)**
 - 2.3. **Test**: Η ονομασία του τεστ που κάνει ο χρήστης
 - 2.4. **Score**: Το σκορ του χρήστη στο τεστ που έχει κάνει
 - 2.5. **Testdate**: Η ημερομηνία και ώρα που έκανε το τεστ ο χρήστης

Για να μπορέσουμε να χρησιμοποιήσουμε την SQL στην C#, χρησιμοποιήσαμε την βιβλιοθήκη Npgsql.

1. HOME PAGE

Η αρχική σελίδα περιλαμβάνει 2 textbox για την εισαγωγή του username και του password αντίστοιχα, σε περίπτωση που ο χρήστης έχει ήδη εγγραφτεί στην πλατφόρμα.



Ξεκινώντας το connection με την βάση πατώντας το κουμπί «Είσοδος» γίνονται οι εξής διεργασίες

```
private void login_Click(object sender, EventArgs e)
{
    string un = username.Text;
    string pw = password.Text;

    using (NpgsqlConnection connection = new NpgsqlConnection(connectionString))
    {
        connection.Open();

        string sql = "SELECT COUNT(*) FROM users WHERE username = @username AND password = @password";
        using (NpgsqlCommand command = new NpgsqlCommand(sql, connection))
        {
            command.Parameters.AddWithValue("@username", un);
            command.Parameters.AddWithValue("@password", pw);

            int count = Convert.ToInt32(command.ExecuteScalar());
            if (count > 0)
            {
                string updateSql = "UPDATE users SET lastlogin = @lastlogin WHERE username = @username";
                using (NpgsqlCommand updateCommand = new NpgsqlCommand(updateSql, connection))
                {
                    updateCommand.Parameters.AddWithValue("@lastlogin", DateTime.Now);
                    updateCommand.Parameters.AddWithValue("@username", un);

                    int rowsAffected = updateCommand.ExecuteNonQuery();
                    if (rowsAffected == 0)
                    {

```

1. . Ορίζονται οι παράμετροι @username και @password της εντολής με τις αντίστοιχες τιμές (un και pw).

2. Ορίζεται η μεταβλητή **sql**, η οποία περιέχει μια SQL εντολή **SELECT COUNT(*) FROM users WHERE username = @username AND password = @password**. Αυτή η εντολή επιστρέφει τον αριθμό των εγγραφών στον πίνακα 'users' που έχουν το συγκεκριμένο όνομα

χρήστη και κωδικό πρόσβασης που παρέχονται ως παράμετροι (@username και @password).

3. Χρησιμοποιείται η μέθοδος **"ExecuteScalar()"** για να εκτελεστεί η SQL εντολή και να ανακτηθεί το αποτέλεσμα, δηλαδή ο αριθμός των εγγραφών που ικανοποιούν τα κριτήρια αναζήτησης.

Εάν ο αριθμός αυτός είναι μεγαλύτερος από το μηδέν, σημαίνει ότι βρέθηκε μια εγγραφή με το συγκεκριμένο όνομα χρήστη και κωδικό πρόσβασης.

4. Στη συνέχεια, αν ο χρήστης είναι εγγεγραμμένος, εκτελείται μια δεύτερη SQL εντολή με τη χρήση της συμβολοσειράς `updateSql`, η οποία ενημερώνει το πεδίο `lastlogin` του πίνακα `USERS` με την τρέχουσα ημερομηνία και ώρα.
5. Τέλος, εμφανίζεται ένα μήνυμα επιτυχίας και ανοίγει η επόμενη φόρμα «Menu», ενώ αποκρύπτεται η τρέχουσα φόρμα.

```

int rowsAffected = updateCommand.ExecuteNonQuery();
if (rowsAffected == 0)
{
    MessageBox.Show("Failed to update last login timestamp.");
}
else
{
    int loggedInUserID = RetrieveLoggedInUserID(un);
    if (loggedInUserID != -1)
    {
        HomePage.userid = loggedInUserID;
        MessageBox.Show("Επιτυχής Σύνδεση");
        var nextform = new Menu();
        nextform.Show();
        this.Hide();
    }
}
}
else
{
    MessageBox.Show("Λάθος username ή/και password!");
}
}

```

Με την μέθοδο `RetrieveLoggedInUserID(string username)`, σε περίπτωση που το login είναι επιτυχές, αποθηκεύεται στην μεταβλητή **`public static int userid { get; set; }`** το `userid`, διότι κληρονομείται από τις υπόλοιπες φόρμες για να γίνεται η επί τόπου αποθήκευση των αποτελεσμάτων για τον τρέχων συνδεδεμένο χρήστη, αλλά και την δυνατότητα εμφάνισης του ιστορικού σύνδεσής του.

```

string connectionString = "Host=localhost;Port=5432;Username=postgres;Password=1234admin;Database=ekp_log;";
public static int userid { get; set; }

private int RetrieveLoggedInUserID(string username)
{
    int userid = -1;

    using (NpgsqlConnection connection = new NpgsqlConnection(connectionString))
    {
        connection.Open();

        string sql = "SELECT userid FROM users WHERE username = @username";
        using (NpgsqlCommand command = new NpgsqlCommand(sql, connection))
        {
            command.Parameters.AddWithValue("@username", username);

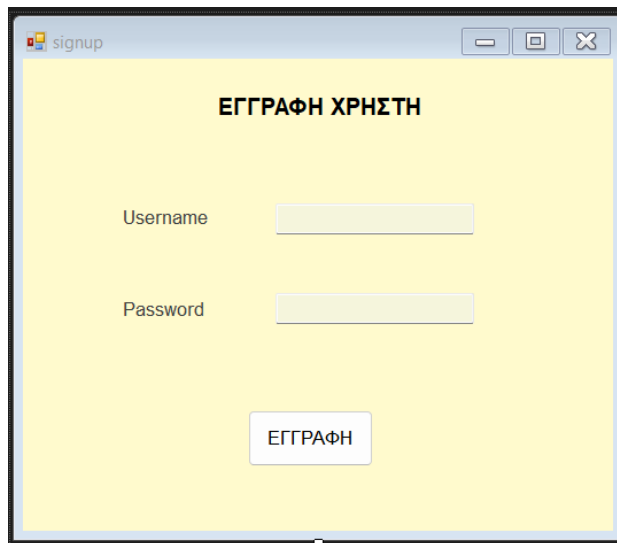
            object result = command.ExecuteScalar();
            if (result != null)
            {
                userid = Convert.ToInt32(result);
            }
        }
    }

    return userid;
}

```

Σε περίπτωση που ο χρήστης δεν έχει λογαριασμό, πατάει το κουμπί «Δημιουργία Λογαριασμού» και μεταφέρεται αμέσως στην φόρμα «Signup».

2. SIGN UP



Στην φόρμα αυτή, ο χρήστης που δεν έχει λογαριασμό, μπορεί να εγγραφτεί εισάγοντας το username και τον κωδικό πρόσβασης που επιθυμεί. Τα στοιχεία αυτά αποθηκεύονται αυτόματα στην βάση δεδομένων.

```
string connectionString = "Host=localhost;Port=5432;Username=postgres;Password=1234admin;Database=ekp_log;";

private void signup_button_Click(object sender, EventArgs e)
{
    string un = username.Text;
    string pw = password.Text;

    using (NpgsqlConnection connection = new NpgsqlConnection(connectionString))
    {
        connection.Open();
        string checkUsernameSql = "SELECT COUNT(*) FROM users WHERE username = @username";
        using (NpgsqlCommand checkUsernameCommand = new NpgsqlCommand(checkUsernameSql, connection))
        {
            checkUsernameCommand.Parameters.AddWithValue("@username", un);
            int existingUserCount = Convert.ToInt32(checkUsernameCommand.ExecuteScalar());

            if (existingUserCount > 0)
            {
                MessageBox.Show("Το username χρησιμοποιείται ήδη! Δοκιμάστε ένα διαφορετικό.");
                return;
            }
        }
    }
}
```

Γίνεται έλεγχος αν υπάρχει ήδη χρήστης με το ίδιο όνομα χρήστη (username) που προσπαθεί να δημιουργήσει ο χρήστης. Αυτό γίνεται με την εκτέλεση μιας εντολής SQL που μετρά τον αριθμό των χρηστών με το συγκεκριμένο όνομα χρήστη. Αν υπάρχει τουλάχιστον ένας χρήστης, εμφανίζεται ένα μήνυμα λάθους και η διαδικασία διακόπτεται.

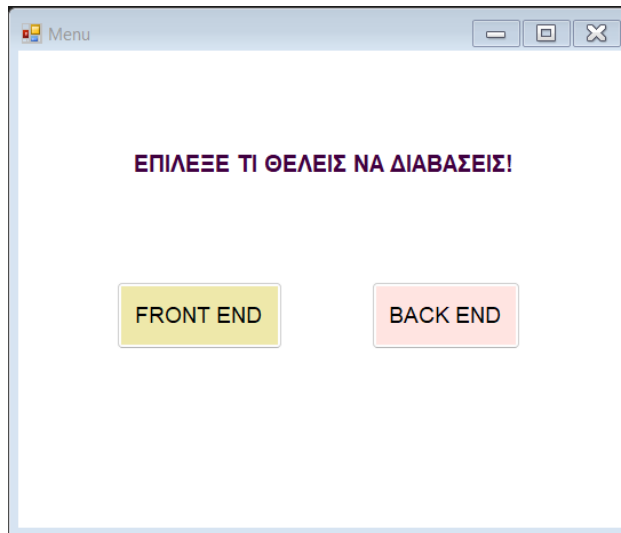
Αν δεν υπάρχει χρήστης με αυτό το όνομα χρήστη, τότε εκτελείται η εντολή SQL για να εισαχθεί ο νέος χρήστης στη βάση δεδομένων. Η εντολή SQL αυτή εισάγει το username, τον κωδικό πρόσβασης και την τρέχουσα ημερομηνία και ώρα σύνδεσης (lastlogin) του χρήστη.

Αν η εισαγωγή είναι επιτυχής, εμφανίζεται ένα μήνυμα επιτυχίας και ο χρήστης μεταφέρεται στην αρχική σελίδα της εφαρμογής, όπου μπορεί να συνδεθεί.

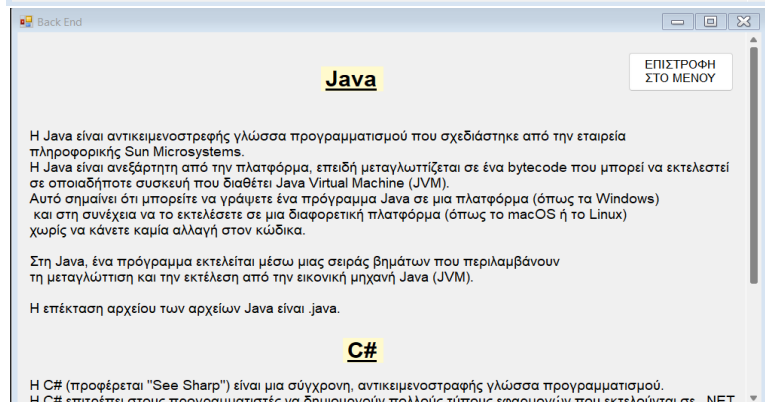
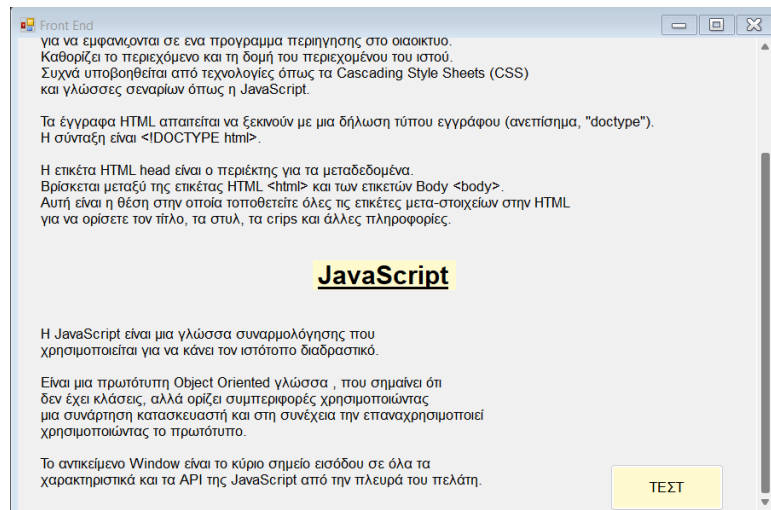
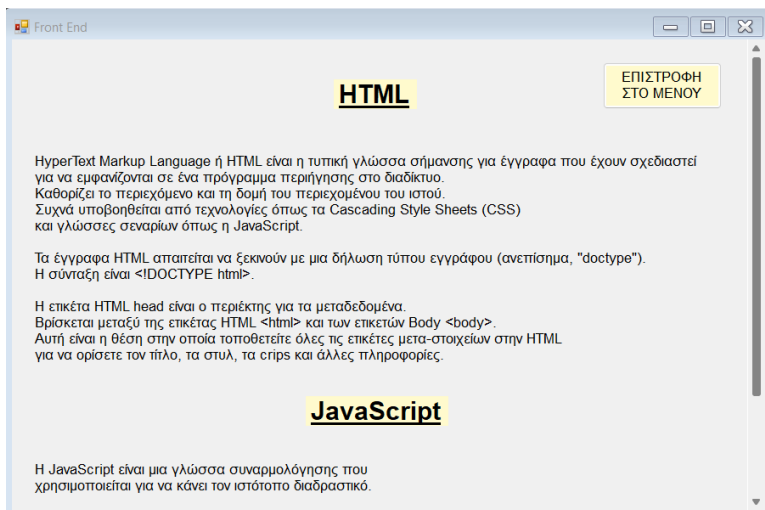
Αν η εισαγωγή αποτύχει, εμφανίζεται ένα μήνυμα λάθους.

3. Menu & Διδακτικό Κείμενο

Μετά την επιτυχή σύνδεση, ο χρήστης μεταφέρεται σε μια φόρμα όπου μπορεί να επιλέξει ποιο αντικείμενο θέλει να διαβάσει.



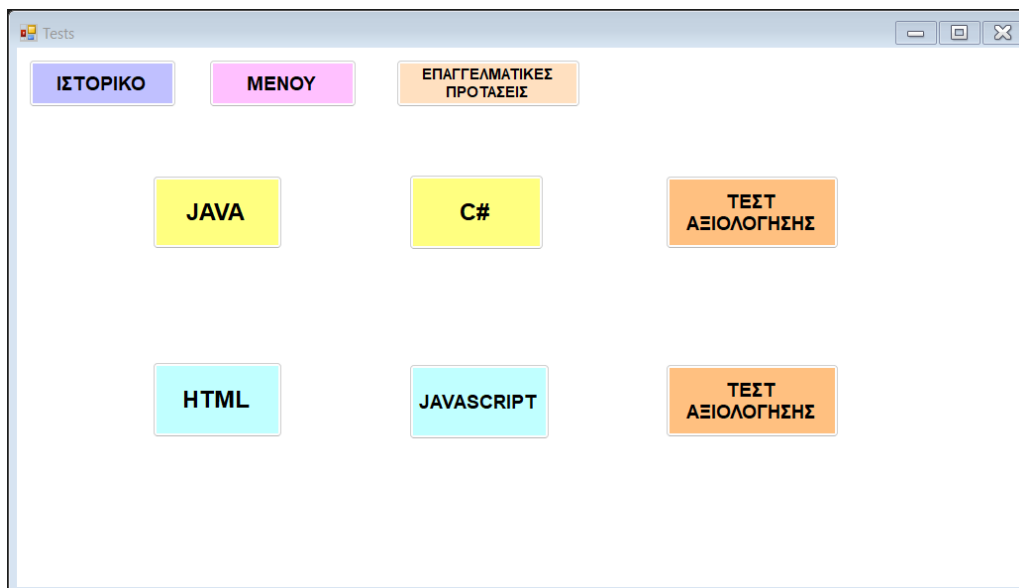
Όταν επιλέξει μια από τις δύο κατηγορίες, μεταβαίνει στη αντίστοιχη φόρμα που μπορεί να διαβάσει τα κείμενα για το αντικείμενο αυτό.



Πατώντας το κουμπί «ΕΠΙΣΤΡΟΦΗ ΣΤΟ ΜΕΝΟΥ», επιστρέφει στην προηγούμενη φόρμα «Menu».

Πατώντας το κουμπί «ΤΕΣΤ», μεταβαίνει στην φόρμα που περιέχει όλες τις κατηγορίες των τεστ.

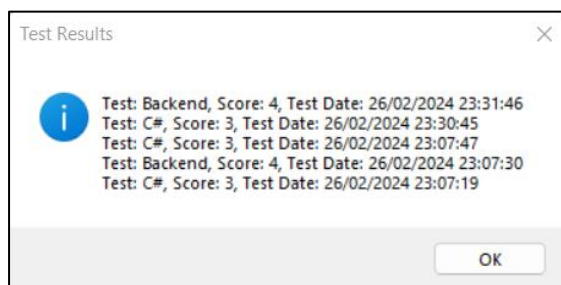
4. Tests



Ο χρήστης επιλέγει ποιο τεστ θέλει να κάνει, με τα τεστ αξιολόγησης να είναι επαναληπτικά βασισμένα στα προηγούμενα τεστ της ίδιας κατηγορίας.

Με το κουμπί «ΜΕΝΟΥ», επιστρέφει στην αντίστοιχη φόρμα.

Με το κουμπί «ΙΣΤΟΡΙΚΟ», εμφανίζει τα σκορ του χρήστη από τα προηγούμενα τεστ που είχε κάνει.



```
private void history_Click(object sender, EventArgs e)
{
    string connectionString = "Host=localhost;Port=5432;Username=postgres;Password=1234admin;Database=ekp_log;";
    int userid = HomePage.userid;

    string sql = @"
    SELECT test, score, testdate
    FROM public.usertests
    WHERE userid = @userid
    ORDER BY testdate DESC;";

    try
    {
        using (NpgsqlConnection connection = new NpgsqlConnection(connectionString))
        {
            connection.Open();

            using (NpgsqlCommand command = new NpgsqlCommand(sql, connection))
            {
                command.Parameters.AddWithValue("@userid", userid);

                using (NpgsqlDataReader reader = command.ExecuteReader())
                {
                    StringBuilder sb = new StringBuilder();
                    while (reader.Read())
                    {
                        string test = reader.GetString(0);
                        int score = reader.GetInt32(1);
                        DateTime testDate = reader.GetDateTime(2);

                        sb.AppendLine($"Test: {test}, Score: {score}, Test Date: {testDate}");
                    }

                    if (sb.Length > 0)
                    {
                        MessageBox.Show(sb.ToString(), "Test Results", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                    else
                    {
                        MessageBox.Show("No test results found for the user.", "Test Results", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                }
            }
        }
    }
}
```

Συγκεντρώνει όλα τα τεστ του χρήστη με βάση την ημερομηνία που έκανε το τεστ.

Για κάθε γραμμή αποτελέσματος, λαμβάνονται τα δεδομένα του τεστ (όνομα τεστ, σκορ και ημερομηνία τεστ) και προστίθενται σε ένα StringBuilder.

Με το κουμπί «ΕΠΑΓΓΕΛΜΑΤΙΚΕΣ ΠΡΟΤΑΣΕΙΣ», μεταβαίνει στην φόρμα που προτείνει τον επαγγελματικό κλάδο για να ακολουθήσει ο χρήστης με βάση τα πιο πρόσφατα σκορ του και τις βαθμολογίες που έχει πάρει στο κάθε μάθημα.

Όλα τα τεστ έχουν την ίδια διάταξη ερωτήσεων, όπως το παρακάτω:

HTML σημαίνει __

☒ HyperText Markup Language

☐ HyperText Marking Language

☐ HighText Marking Language

Ποια είναι η σωστή σύνταξη του doctype στην HTML5;

☐ <doctype html> ☐ </DOCTYPE html> ☐ <IDOCYPE html>

Σε ποιο μέρος της HTML περιέχονται τα μεταδεδομένα;

☐ head tag ☐ html tag ☐ body tag

ΤΕΛΟΣ

Για κάθε ερώτηση, οι επιλογές είναι radioButtons και για κάθε ερώτηση βρίσκονται μέσα σε groupBox για να μπορεί να γίνεται η επιλογή της απάντησης.

Πατώντας το κουμπί «ΤΕΛΟΣ», εμφανίζεται ένα μήνυμα με το σκορ του χρήστη. Αν είναι >2, ζητάει από το χρήστη να ξαναδιαβάσει τις πληροφορίες και επιστρέφει στην φόρμα με το αντίστοιχο κείμενο-μάθημα.

Αν είναι =<2, επιστρέφει τον χρήστη στην φόρμα Tests.

```
public void SaveTestScore(int score)
{
    string connectionString = "Host=localhost;Port=5432;Username=postgres;Password=1234admin;Database=ekp_log;";

    int userid = HomePage.userid;
    string test = "HTML";
    DateTime testDate = DateTime.Now;

    string sql = @"
INSERT INTO public.usertests (userid, test, score, testdate)
VALUES (@userid, @test, @score, @testdate);
";

    try
    {
        using (NpgsqlConnection connection = new NpgsqlConnection(connectionString))
        {
            connection.Open();

            using (NpgsqlCommand command = new NpgsqlCommand(sql, connection))
            {
                command.Parameters.AddWithValue("@userid", userid);
                command.Parameters.AddWithValue("@test", test);
                command.Parameters.AddWithValue("@score", score);
                command.Parameters.AddWithValue("@testdate", testDate);

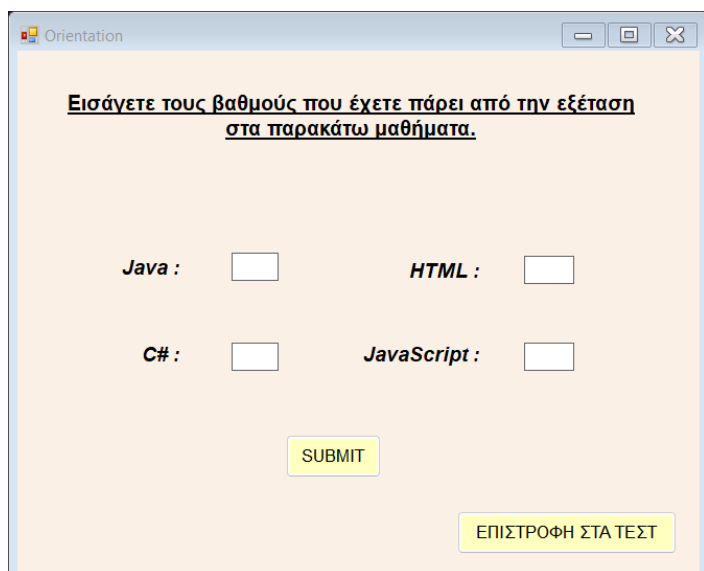
                command.ExecuteNonQuery();
            }
        }
    }
}
```

Σε κάθε περίπτωση, το σκορ αποθηκεύεται στην βάση δεδομένων.

5. Orientation

Η πλατφόρμα έχει την δυνατότητα να προτείνει επαγγελματική κατεύθυνση στον χρήστη με βάση τα σκορ του και τους βαθμούς που έχει πάρει στο κάθε μάθημα από τον διδάσκοντα.

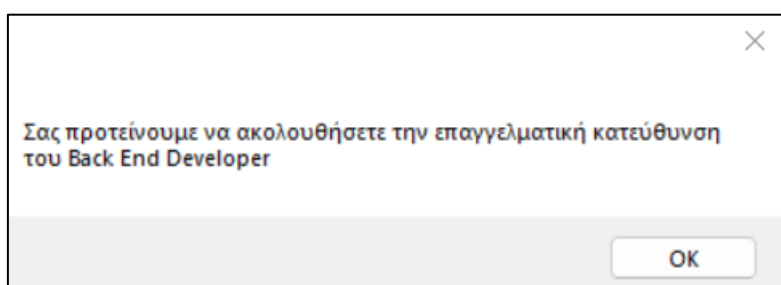
Ο χρήστης εισάγει στα κουτιά τους βαθμούς του αντίστοιχου μαθήματος.



The screenshot shows a web application window titled "Orientation". Inside, there is a heading: **Εισάγετε τους βαθμούς που έχετε πάρει από την εξέταση στα παρακάτω μαθήματα.** Below this, there are four input fields arranged in a 2x2 grid. The labels are: **Java :**, **HTML :**, **C# :**, and **JavaScript :**. Each label is followed by a small, empty rectangular input box. At the bottom center of the form is a yellow button labeled "SUBMIT". At the bottom right is a yellow button labeled "ΕΠΙΣΤΡΟΦΗ ΣΤΑ ΤΕΣΤ".

Έπειτα, με βάση αυτούς τους βαθμούς και τα πιο πρόσφατα σκορ από τα τελευταία τεστ, βγαίνει το εξής αποτέλεσμα:

- Αν το άθροισμα του Back End είναι το μεγαλύτερο, τότε προτείνει να ακολουθήσει καριέρα Back End Developer.
- Αν το άθροισμα του Front End είναι το μεγαλύτερο, τότε προτείνει να ακολουθήσει καριέρα Front End Developer.
- Αν είναι ίσα, τότε προτείνει να ακολουθήσει καριέρα Full Stack Developer.



6. Βάση Δεδομένων

Όπως αναφέρθηκε κι αρχικά, η βάση δεδομένων έχει γίνει στο περιβάλλον pgAdmin χρησιμοποιώντας postgresql.

	userid [PK] integer	username character varying (50)	password character varying (100)	lastlogin timestamp without time zone
1	1	sofia	1234	2024-03-06 15:13:56.669145
2	2	eleni	1234	2024-02-27 01:11:41.788258
3	3	popi	1234	2024-02-25 19:55:05.071846
4	4	merkouris	1234	2024-02-26 23:06:06.700468
5	5	marika	1234	2024-02-26 18:43:11.957954
6	6	danai	1234	2024-02-26 20:21:47.876654

	usertestid [PK] integer	userid integer	test character varying (255)	score integer	testdate timestamp without time zone
1	1	1	HTML	3	2024-02-26 20:20:41.860871
2	2	6	HTML	2	2024-02-26 20:21:23.212761
3	3	6	HTML	3	2024-02-26 20:21:58.124158
4	4	4	JavaScript	2	2024-02-26 23:06:19.896511
5	5	4	Frontend	4	2024-02-26 23:06:30.921885
6	6	2	C#	3	2024-02-26 23:07:19.678505
7	7	2	Backend	4	2024-02-26 23:07:30.311902
8	8	2	C#	3	2024-02-26 23:07:47.645628
9	9	2	C#	3	2024-02-26 23:30:45.75723
10	10	2	Backend	4	2024-02-26 23:31:46.692289

Τα παραπάνω στιγμιότυπα, δείχνουν την δομή της βάσης και τις μέχρι τώρα εισαγωγές που έχουν γίνει.