

Ανάλυση και Σχεδιασμός Εφαρμογής PyGuru

Φοιτητές:
Βασίλειος Ζωγράφος - Π14050
Παναγιώτης Ευσταθιάδης - Π14042

Για τον σχεδιασμό της εφαρμογής PyGuru χρησιμοποιήθηκε το Android Studio.

Η δομή της εφαρμογής αποτελείται από τα ακόλουθα activities:

MainActivity, MainMenuActivity, RegisterActivity, LessonActivity, LessonsActivity, ProgressActivity, ProfileActivity, ForgotPwdActivity, HelpActivity, QuizActivity, FinalQuizActivity, AnswerActivity, NewTeacherActivity, ResetTeacherPwdActivity, TeacherLoginActivity, TeacherMenuActivity και NewQuestionActivity.

Ο κώδικας της εφαρμογής ειναι μοιρασμένη στα αρχεία:

MainActivity.kt, MainMenuActivity.kt, RegisterActivity.kt, LessonActivity.kt, LessonsActivity.kt, ProgressActivity.kt, ProfileActivity.kt, ForgotPwdActivity.kt, QuizActivity.kt, FinalQuizActivity.kt, AnswerActivity.kt, QuizzesActivity.kt, NewTeacherActivity.kt, ResetTeacherPwdActivity.kt, TeacherLoginActivity.kt, TeacherMenuActivity.kt, HelperActivity.kt, NewQuestionActivity.kt και PyGuruHelper.java .

Εξήγηση Σχεδιασμού Εφαρμογής

Όταν ο χρήστης ανοίγει την εφαρμογή κατευθύνεται στη σελίδα MainActivity η οποία είναι υπεύθυνη για την σύνδεση του χρήστη.

Η σελίδα αυτή περιέχει μια φόρμα στην οποία συμπληρώνει τα στοιχεία του (όνομα χρήστη και κωδικό) και πατά το κουμπί με όνομα submit σε περίπτωση που έχει φτιάξει λογαριασμό. Όλα τα πεδία είναι υποχρεωτικά. Μετά την σύνδεση του ο χρήστης κατευθύνεται στη σελίδα MainMenuActivity όπου εκεί υπάρχει menu με τις ακόλουθες επιλογές: Help, Progress, Reset Password, Logout, Lessons και Tests.

Αν ο χρήστης επιλέξει τον σύνδεσμο Help τότε κατευθύνεται στη σελίδα HelpActivity όπου εκεί υπάρχουν συχνές απορίες σχετικές με τα χαρακτηριστικά της εφαρμογής.

Αν ο χρήστης επιλέξει τον σύνδεσμο Logout τότε αποσυνδέεται και κατευθύνεται στη σελίδα MainActivity.

Αν ο χρήστης επιλέξει τον σύνδεσμο Progress τότε κατευθύνεται στη σελίδα ProgressActivity όπου εκεί βλέπει τους βαθμούς του σε όλα τα τεστ, αν έχει κοπεί σε κάποια από αυτά, ποιες είναι οι αδυναμίες του, αν έχει χαμηλούς βαθμούς αλλά και τι κάνει για να αντιμετωπίσει τις αδυναμίες του.

Αν ο χρήστης επιλέξει τον σύνδεσμο Reset Password τότε κατευθύνεται στη σελίδα ProfileActivity όπου εκεί μπορεί να αλλάξει τον κωδικό πρόσβασης του μέσω της φόρμας που του παρέχει η σελίδα. Τα πεδία της φόρμας είναι όνομα και κωδικός χρήστη και είναι υποχρεωτικά.

Αν ο χρήστης επιλέξει τον σύνδεσμο Lessons τότε κατευθύνεται στη σελίδα LessonsActivity οπού εκεί υπάρχουν όλοι οι τίτλοι των διαθέσιμων μαθήματων, η ημερομηνία και ώρα της τελευταίας επίσκεψης του χρήστη σε κάθε μάθημα(αν υπάρχει). Αν επιλεγεί ένα από αυτά τα μαθήματα τότε ο χρήστης κατευθύνεται στη σελίδα LessonActivity στην οποία βλέπει εκτός από το περιεχόμενο του μαθήματος που επέλεξε και την επισκεψιμότητα του μαθήματος, η οποία αυξάνεται κάθε φορά που το επιλέγει. Επίσης αποθηκεύεται στη βάση δεδομένων η ημερομηνία και ώρα της τελευταίας επίσκεψης του μάθημα που έχει επιλέξει να δει.

Αν ο χρήστης επιλέξει τον σύνδεσμο Tests τότε κατευθύνεται στη σελίδα QuizzesActivity οπού εκεί βλέπει όλα τα διαθέσιμα τεστ και μπορεί να επιλέξει ένα από αυτά. Όταν επιλέξει κάποιο από αυτά τότε κατευθύνεται στη σελίδα QuizActivity αν το τεστ δεν είναι το επαναληπτικό αλλιώς μεταφέρεται στη σελίδα FinalQuizActivity όπου και στις δυο σελίδες ο χρήστης βλέπει όλες τις ερωτήσεις του τεστ που επέλεξε (7 ερωτήσεις για τη σελίδα QuizActivity και 14 για τη σελίδα FinalQuizActivity). Όταν απαντήσει όλες τις ερωτήσεις (οι οποίες είναι τυχαίες κάθε φορά που επιλεγεί ένα τεστ) και πατήσει το κουμπί done τότε βλέπει σε ένα παράθυρο το βαθμό του και σε ποιες ερωτήσεις έκανε λάθος.

Για να απαντήσει σε μια ερώτηση απλά πατάει το κείμενο της ερώτησης και τότε μεταφέρεται στη σελίδα AnswerActivity. Εκεί βλέπει όλες τις διαθέσιμες απαντήσεις οι οποίες είναι πάντα τρεις σε πλήθος και μια μόνο είναι η σωστή.

Αν ο χρήστης δεν έχει λογαριασμό τότε πατά τον σύνδεσμο “Not a member yet? Register now!” του MainActivity για να

τον κατευθύνει η εφαρμογή στη σελίδα RegisterActivity. Όλα τα πεδία είναι υποχρεωτικά. Όταν συμπληρώσει τη φόρμα τότε κατευθύνεται στη σελίδα MainActivity για να συνδεθεί. Μπορεί να υπάρχει σε κάθε εφαρμογή το πολύ ένας μαθητής.

Αν ο χρήστης δεν είναι μαθητής αλλά εξεταστής τότε πατά το σύνδεσμο “Are you a teacher? Login here” του MainActivity για να τον κατευθύνει στη σελίδα TeacherLoginActivity. Εκεί γραφεί μέσα σε μια φόρμα το όνομα και τον κωδικό του και πατά το κουμπί με όνομα change password τα οποία είναι υποχρεωτικά. Όταν συμπληρώσει τη φόρμα του ο χρήστης επιτυχώς μεταφέρεται στη σελίδα TeacherMenuActivity. Εκεί έχει τρεις επιλογές: αλλαγή κωδικού, αποσύνδεση και πρόσθεση νέας ερώτησης. Η εφαρμογή μας μπορεί να αποθηκεύσει παραπάνω από έναν εξεταστή.

Αν ο εξεταστής επιλέξει την επιλογή αποσύνδεση (Logout) αποσυνδέεται και μεταφέρεται στο TeacherLoginActivity.

Αν ο εξεταστής επιλέξει την επιλογή αλλαγή κωδικού (τίτλος: Reset Password) τότε μεταφέρεται στη σελίδα ResetTeacherPwdActivity η οποία λειτουργεί όπως και η σελίδα ProfileActivity.

Αν ο εξεταστής επιλέξει την επιλογή πρόσθεση ερώτησης (τίτλος: Add Question) τότε μεταφέρεται στη σελίδα NewQuestionActivity. Εκεί υπάρχει μια μεγάλη φόρμα στην οποία πρέπει να γράψει το περιεχόμενο της ερώτησης, της απαντήσεις της, ποια απάντηση είναι η σωστή και σε ποιο τεστ θα προστεθεί η ερώτηση. Όλα τα πεδία της φόρμας αυτής είναι υποχρεωτικά.

Αν δεν έχει φτιάξει ακόμη λογαριασμό ένας χρήστης και είναι κάποιος εξεταστής τότε πατά τον σύνδεσμο με κείμενο “Not a member yet? Register now!” του MainActivity για να τον κατευθύνει η εφαρμογή στη σελίδα NewTeacherActivity. Σε αυτή τη σελίδα συμπληρώνει τη φόρμα με το όνομα και τον κωδικό του. Όλα τα πεδία είναι υποχρεωτικά.

Αν ο χρήστης είναι μαθητής και δε θυμάται τον κωδικό πρόσβασης του τότε πατά τον σύνδεσμο “Forgot Password” του MainActivity για να τον κατευθύνει στη σελίδα ForgotPwdActivity. Εκεί γραφεί μέσα σε μια φόρμα το όνομα και τον νέο κωδικό του και πατά το κουμπί με όνομα change password. Όλα τα πεδία είναι υποχρεωτικά.

Εξήγηση Κώδικα Εφαρμογής

MainActivity.kt:

```
12 class MainActivity : AppCompatActivity() {
13
14     override fun onCreate(savedInstanceState: Bundle?) {
15         super.onCreate(savedInstanceState)
16         setContentView(R.layout.activity_main)
17
18         val dbHelper = PyGuruHelper(this)
19
20         var loginBtn : Button = findViewById(R.id.loginButton)
21
22         loginBtn.setOnClickListener {
23             // get username and password
24             var username : TextView = findViewById(R.id.editText3)
25             var password : TextView = findViewById(R.id.editText4)
26
27             if (!username.text.isEmpty() && !password.text.isEmpty()) {
28                 // user validation here with sqlite
29                 val res : Boolean = dbHelper.findUser(username.text.toString(), password.text.toString())
30                 if (res) {
31                     // save user's username
32                     val sharedPref: SharedPreferences = this.getSharedPreferences("PyGuruStudent", Context.MODE_PRIVATE)
33                     val editor : SharedPreferences.Editor = sharedPref.edit()
34                     editor.putString("PyGuruUser", username.text.toString())
35                     editor.commit()
36
37                     // go to main menu
38                     val intent = Intent(this, MainMenuActivity::class.java)
39                     startActivity(intent) // start main menu activity
40                 } else {
41                     // some field is wrong
42                     Toast.makeText(this, "Username/Password is wrong.", Toast.LENGTH_LONG).show()
43                 }
44
45             } else {
46                 // form's fields are required
47                 Toast.makeText(this, "You must fill the form.", Toast.LENGTH_LONG).show()
48             }
49         }
50
51         val registerText : TextView = findViewById(R.id.textView2)
52         registerText.setOnClickListener {
53             val intent = Intent(this, RegisterActivity::class.java)
54             startActivity(intent) // start register activity
55         }
56
57         val teacherText : TextView = findViewById(R.id.teacherLogin)
58         teacherText.setOnClickListener {
59             val intent = Intent(this, TeacherLoginActivity::class.java)
60             startActivity(intent) // start teacher login activity
61         }
62
63         val forgotPwdText : TextView = findViewById(R.id.forgotPwd)
64         forgotPwdText.setOnClickListener {
65             val intent = Intent(this, ForgotPwdActivity::class.java)
66             startActivity(intent) // start forgot password login activity
67         }
68 }
```

Στη γραμμή 18 δημιουργούμε ένα αντικείμενο της κλάσης PyGuruHelper το οποίο είναι υπεύθυνο για την επικοινωνία με τη βάση δεδομένων και για την παροχή βοηθητικών συναρτήσεων. Η κλάση PyGuruHelper θα εξηγηθεί αργότερα.

Στις γραμμές 20-67 προσθέτουμε συναρτήσεις που διαχειρίζονται το click event του κουμπιού Login και των TextViews που είναι υπεύθυνα για την εγγραφή μαθητή, την σύνδεση εξεταστή και αλλαγή κωδικού μαθητή.

Στο click event του κουμπιού Login βλέπουμε αν όλα τα πεδία της φόρμας Login είναι συμπληρωμένα και αν τα στοιχεία που μας παρέχονται από τον χρήστη είναι σωστά. Αν δεν υπάρχει μαθητής με τα στοιχεία που δίνονται από τον χρήστη ή δεν είναι όλα τα στοιχεία συμπληρωμένα ή αν έγινε κάποιο σφάλμα κατά την επικοινωνία με τη βάση δεδομένων τότε εμφανίζουμε το κατάλληλο μνήμα στο χρήστη. Αν υπάρχει χρήστης με τα στοιχεία που δοθήκανε από την φόρμα Login τότε μεταφέρουμε τον χρήστη στη σελίδα MainMenuActivity και αποθηκεύουμε το όνομα του στη συσκευή για μελλοντική χρήση.

Στο click event των TextViews για την εγγραφή μαθητή, την σύνδεση εξεταστή και αλλαγή κωδικού μαθητή μεταφέρουμε τον χρήστη στις σελίδες RegisterActivity, TeacherLoginActivity και ForgotPwdActivity αντίστοιχα.

MainMenuActivity.kt:

```
11 class MainActivity : AppCompatActivity() {
12
13     override fun onCreate(savedInstanceState: Bundle?) {
14         super.onCreate(savedInstanceState)
15         setContentView(R.layout.activity_main_menu)
16
17         val welcomeTxt : TextView = findViewById<TextView>(R.id.welcomeText)
18         var txt : String = welcomeTxt.text.toString()
19
20         // get user's name
21         val sharedPref: SharedPreferences = this.getSharedPreferences("PyGuruStudent", Context.MODE_PRIVATE)
22         txt = txt + " " + sharedPref.getString("PyGuruUser", "") + " !"
23
24         // update text
25         welcomeTxt.text = txt
26
27
28         // setup listeners
29         val progressBtn : Button = findViewById(R.id.progressBar)
30         progressBtn.setOnClickListener {
31             // go to progress
32             val intent = Intent(this, ProgressActivity::class.java)
33             startActivity(intent) // start progress activity
34         }
35
36         val profileBtn : Button = findViewById(R.id.profileButton)
37         profileBtn.setOnClickListener {
38             // go to profile
39             val intent = Intent(this, ProfileActivity::class.java)
40             startActivity(intent) // start profile activity
41         }
42
43         val testsBtn : Button = findViewById(R.id.testsButton)
44         testsBtn.setOnClickListener {
45             // go to quizzes
46             val intent = Intent(this, QuizzesActivity::class.java)
47             startActivity(intent) // start quizzes activity
48         }
49
50         val lessonsBtn : Button = findViewById(R.id.lessonsButton)
51         lessonsBtn.setOnClickListener {
52             // go to lessons
53             val intent = Intent(this, LessonsActivity::class.java)
54             startActivity(intent) // start lessons activity
55         }
56
57         val helpBtn : Button = findViewById(R.id.helpBtn)
58         helpBtn.setOnClickListener {
59             // go to lessons
60             val intent = Intent(this, HelpActivity::class.java)
61             startActivity(intent) // start help activity
62         }
63
64     val logoutBtn : Button = findViewById(R.id.logout_button)
65     logoutBtn.setOnClickListener {
66         // remove current stored user
67         val sharedPref: SharedPreferences = this.getSharedPreferences("PyGuruStudent", Context.MODE_PRIVATE)
68         val editor : SharedPreferences.Editor = sharedPref.edit()
69         editor.putString("PyGuruUser", "")
70         editor.commit()
71
72         // go to login view again
73         val intent = Intent(this, MainActivity::class.java)
74         startActivity(intent) // start main(login) activity
75     }
76 }
77 }
```

```
64     val logoutBtn : Button = findViewById(R.id.logout_button)
65     logoutBtn.setOnClickListener {
66         // remove current stored user
67         val sharedPref: SharedPreferences = this.getSharedPreferences("PyGuruStudent", Context.MODE_PRIVATE)
68         val editor : SharedPreferences.Editor = sharedPref.edit()
69         editor.putString("PyGuruUser", "")
70         editor.commit()
71
72         // go to login view again
73         val intent = Intent(this, MainActivity::class.java)
74         startActivity(intent) // start main(login) activity
75     }
76 }
77 }
```

Στις γραμμές 17-25 δημιουργούμε το κείμενο “Welcome <username>!” όπου το username έχει αποθηκευτεί στη συσκευή από τη στιγμή που έχει κάνει σύνδεση ο χρήστης ως μαθητής.

Στις γραμμές 28-78 προσθέτουμε συναρτήσεις που διαχειρίζονται το click event των κουμπιών Progress, Reset Password, Lessons, Tests, Help και Logout. Κάθε ένα από αυτά τα κουμπιά μεταφέρει τον χρήστη στις σελίδες ProgressActivity, ProfileActivity, LessonsActivity, TestsActivity, HelpActivity και MainActivity αντίστοιχα.

PyGuruHelper.java:

```
16 public class PyGuruHelper extends SQLiteOpenHelper {
17     public static final String DB_NAME = "PyGuru.db";
18
19     public static String[] bad = {
20         "You don't know the basics!\n See Lesson 1 and try again!",
21         "You aren't very good at list and tuple processing!\n See Lesson 2 and try again!",
22         "You aren't very good at dictionaries!\n See Lesson 3 and try again!",
23         "You aren't very good at while loops!\n See Lesson 4 and try again!",
24         "You don't know about conditional statements and booleans!\n See Lesson 5 and try again!",
25         "You are not good at string processing! See Lesson 6 and try again!",
26         "You aren't good at algorithmic thinking in Python!\n See all the lessons and try again when you're able to at least pass every test"
27     };
28     public static String good = "Good job!";
29     public static String veryGood = "Very good! Keep it up!";
30     public static String excellent = "Excellent!";
31
32     public PyGuruHelper(Context context) {
33         super(context, DB_NAME,null,1);
34     }
35
36     @Override
37     public void onCreate(SQLiteDatabase db) {
38         db.execSQL(
39             "CREATE TABLE IF NOT EXISTS Students(id INTEGER PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR, marks TEXT)"
40         );
41         db.execSQL(
42             "CREATE TABLE IF NOT EXISTS Lessons(id INTEGER PRIMARY KEY AUTOINCREMENT, body TEXT, title TEXT, visits INTEGER DEFAULT 0)"
43         );
44         db.execSQL(
45             "CREATE TABLE IF NOT EXISTS Questions(id INTEGER PRIMARY KEY AUTOINCREMENT, body TEXT, quizNo INTEGER, answers TEXT)"
46         );
47         db.execSQL(
48             "CREATE TABLE IF NOT EXISTS Teachers(id INTEGER PRIMARY KEY AUTOINCREMENT, username VARCHAR, password VARCHAR)"
49         );
50
51     String lesson1 = "Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.\n"
52         "\n" +
53         "Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different"
54         "\n" +
55         "The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable.";
56
57     String lesson2 = "The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. I\n"
58         "\n" +
59         "Creating a list is as simple as putting different comma-separated values between square brackets.\nA tuple is a sequence of immutable Python objects. Tuples are"
56
58         "\n" +
59         "Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also.";
60
61     String lesson3 = "Each key is separated from its value by a colon ':', the items are separated by commas, and the whole thing is enclosed in curly braces. An empty dict"
62         "\n" +
63         "Keys are unique within a dictionary while values may not be. The values of a dictionary can be of any type, but the keys must be of an immutable data type such as"
64
65
66     String lesson4 = "A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.\n\nHere, statement"
67         "\n" +
68         "When the condition becomes false, program control passes to the line immediately following the loop.";
69
70     String lesson5 = "Decision making is anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions.\n"
71         "\n" +
72         "Decision structures evaluate multiple expressions which produce TRUE or FALSE as outcome. You need to determine which action to take and which statements to execute based on the outcome.";
73
74     String lesson6 = "Strings are amongst the most popular types in Python. We can create them simply by enclosing characters in quotes. Python treats single quotes the same as double quotes."
75         "\n" +
76         "To access substrings, use the square brackets for slicing along with the index or indices to obtain your substring.";
77
78     db.execSQL(
79         "INSERT INTO Lessons(id, body, title)" +
80             "VALUES (1, '"+lesson1+"', \"Python Basics\")" +
81             "(2, '"+lesson2+"', \"Lists and Tuples\")" +
82             "(3, '"+lesson3+"', \"Dictionaries in Python\")" +
83             "(4, '"+lesson4+"', \"While loops\")" +
84             "(5, '"+lesson5+"', \"Conditional statements & Booleans\")" +
85             "(6, '"+lesson6+"', \"String joining, indexing and many more...\")"
86     );
87
88     db.execSQL(
89         "INSERT INTO Questions(id, body, quizNo, answers)" +
90             "// quiz 1
91             "VALUES (1, \"What Python built-in function returns the unique number assigned to an object\", 1, \"id,y;ref(),n;refnum(),n\"),
92             "(2, \"Consider the following sequence of statements: 'n = 100; m = n'. How many objects and how many references we have?\", 1, \"2 obj - 1 ref,y;1 obj\"),
93             "(3, \"In Python, a variable may be assigned a value of one type, and then later assigned a value of a different type!\", 1, \"True,y;False,n;None of the above,n\"),
94             "(4, \"Which of the following statements assigns the value 100 to the variable x!\", 1, \"x = 100,y;x := 100,n;x <> 100,n\"),
95             "(5, \"In Python, a variable must be declared before it is assigned a value!\", 1, \"False,y;True,n;None of the above,n\"),
96             "(6, \"What is the result of a = 5; a += 4!\", 1, \"9,y;0,n;-2,n\"),
97             "(7, \"What is a correct syntax to output 'Hello World'?\", 1, \"print('Hello World'),y;echo 'Hello world',n;puts 'Hello World',n\"),
98             "(8, \"Which of these is not a core data type?\", 1, \"Class,y;List,n;Tuple,n\"),
99             "(9, \"What data type is the object below? L = [1, 23, 'Hello', 1]\", 1, \"List,y;Array,n;Dictionary,n\"),
100            "(10, \"Which of the following function convert a string to a float in python?\", 1, \"float(x),y;str(x),n;int(x, [base]),n\"),
101            "(11, \"Which of the following statement(s) is TRUE?\", 1, \"A hash function takes a message of arbitrary length and generates a fixed length code,y;
102            "(12, \"What is the output of print(9/2)\", 1, \"4,y;4.5,n;4.0,n\"),
103            "(13, \"Which function overloads the >> operator?\", 1, \"more(),y;gt(),n;ge(),n\"),
104            "(14, \"Which operator is overloaded by the or() function?\", 1, \"||,y;,n;n//,n\"),
105
106
107             // quiz 2
108             //id body           quizNo   answers
109             "(15, \"Which of the following are true of Python lists?\", 2, \"A list can hold any type,y;A list may contain any type of object except another list,n;
110             "(16, \"Assume the following list definition: a = ['foo', 'bar', 'baz']. What is a[-1]?\", 2, \"'baz',y;error,n;'foo',n\"),
111             "(17, \"If x = [10, [3.14], 20, [30, 'baz', 2.718]], 'foo' what expr gives '2' from 'baz'\", 2, '\"x[1][2][1][2],y;x[1][2][2],n;x[2][1][1][2],n\"),
112             "(18, \"If x = [10, [3.14], 20, [30, 'baz', 2.718]], 'foo' what expr gives '2.218'\", 2, '\"x[1][2][1]:,y;x[1][1]:,n;x[1][2][0:2],n\"),
113             "(19, \"If a = [1, 2, 3, 4, 5] what expr deletes elem 2?\", 2, '\"del a[1],y;a.remove(2),n;a[2]=[],n\"),
114             "(20, \"If a = ['a', 'b', 'c'] what expr adds 'd'?\", 2, '\"a.append('d'),y;a[3]='d',n;a += 'd',n\"),
115             "(21, \"If t = ('foo', 'bar', 'baz') what expr gives 'baz'?\", 2, '\"t[0],y;t(0),n;None of the above,n\"),
116             "(22, \"If a, b, c = (1, 2, 3, 4, 5, 6, 7, 8, 9)[1::3] then b is '\", 2, '\"3,y;9,n;\n\"),
117             "(23, \" Which of the following is a Python tuple?\", 2, '\"(a, 4, 0.44),y;{a': 1, 'b': 2},n;[1, 3, 6],n\"),
118             "(24, \"If t = (1, 2, 4, 3), which of the following is incorrect?\", 2, '\"print(t[3]),y;max(t),n;t[3]=45,n\"),
119             "(25, \"If t = (1, 2, 4, 3) then what is t[1:3]?\", 2, '\"(2, 4),y;(1, 2, 4),n;(2, 4, 3),n\"),
120             "(26, \"If t = (1, 2, 4, 3), then what is t[1:-1]?\", 2, '\"(2, 4),y;(1, 2),n;(2, 4, 3),n\"),
121             "(27, \"If t = (1, 2, 4, 3, 8, 9) then what is [t[i] for i in range(0, len(t), 2)]?\", 2, '\"[1, 4, 8],y;(1, 4, 8),n;[2, 3, 9],n\"),
122             "(28, \"If a = [14, 52, 7], b = a.copy() then is b == a?\", 2, '\"False,y;True,n;None of the above,n\")
123 }
```

```

124 // quiz 3
125 //id body quizNo answers
126 "(29, \"Which of the following is a dictionary?\", 3, \"\{1:3 2:4,y:(4 6),n;['a' 'b'],n\\\"),\" +
127 "(30, \"Which of the following statements create a dictionary?\", 3, \"All mentioned below,y;d={"john":40 "peter":45},n;d={},n\\\"),\" +
128 "(31, \"If d = {'john':40, 'peter':45} the d's keys are:\", 3, '\"john' 'peter',y:40 45,n;None of the above,n\"),\" +
129 "(32, \"If d = {'john':40, 'peter':45} the d's values are:\", 3, '\"40 45,y:'john' 'peter',n;None of the above,n\"),\" +
130 "(33, \"What will be the output of ('john' in d) if d = {'john':40, 'peter':45}\", 3, '\"True,y;False,n;None of the above,n\"),\" +
131 "(34, \"What will be the output of d1 == d2 if: d1 = {'john':40, 'peter':45}, d2 = {'john':466, 'peter':45}\", 3, '\"False,y;True,n;Error,n\"),\" +
132 "(35, \"What will be the output of d1 > d2 if: d1 = {'john':40, 'peter':45}, d2 = {'john':466, 'peter':45}\", 3, '\"Error,y;True,n;False,n\"),\" +
133 "(36, \"If d = {'john':40, 'peter':45} then d['john'] is\", 3, '\"40,y:45,n;None,n\"),\" +
134 "(37, \"If d = {'john':40, 'peter':45} then to delete 'john' we do\", 3, '\"del d['john'],y;del d['john']: 40),n;d.delete('john'),n\"),\" +
135 "(38, \"If d = {'john':40, 'peter':45} then to get the d's size we do\", 3, '\"len(d),y;size(d),n;d.len(),n\"),\" +
136 "(39, \"What will be the output of print(list(d.keys())) if d = {'john':40, 'peter':45}\?", 3, '\"['john' 'peter'],y:('john' 'peter'),n;('john':40 'peter':45)\",\" +
137 "(40, \"If d = {'john':40, 'peter':45} then d['bob'] is\", 3, '\"None,y;Error,n;Something else,n\"),\" +
138 "(41, \"If a = {1:'a', 2:'b', 'c':3} then prin(a.get(1,4)) is\", 3, '\"a',y:1,n;4,n\"),\" +
139 "(42, \"If a = {1:'a', 2:'b', 'c':3} then prin(a.get(5,4)) is\", 3, '\"5,y;Error,n;4,n\"),\" +
140
141 // quiz 4
142 //id body quizNo answers
143 "(43, \"What is the output of the following?\ni = 1\\n\" +
144 "x = ['ab', 'cd']\"+
145 "while True:\\n\" +
146 "    if i%3 == 0:\\n\" +
147 "        break\\n\" +
148 "        print(i)\\n\" +
149 "    \\n\" +
150 "    i += 1\\n\", 4, '\"Error,y;1 2 3,n;1 2,n\"),\" +
151 "(44, \"What is the output of the following?\ni = 1\\n\" +
152 "while True:\\n\" +
153 "    if i%007 == 0:\\n\" +
154 "        break\\n\" +
155 "        print(i)\\n\" +
156 "    i += 1\\n\", 4, '\"1 2 3 4 5 6,y;1 2 3 4 5 6 7,n;Error,n\"),\" +
157 "(45, \"What is the output of the following?\ni = 1\\n\" +
158 "while True:\\n\" +
159 "    if i%0011 == 0:\\n\" +
160 "        break\\n\" +
161 "        print(i)\\n\" +
162 "    i += 1\\n\", 4, '\"5 6 7 8,y;Error,n;5 6 7 8 9 10,n\"),\" +
163 "(46, \"What is the output of the following?\ni = 5\\n\" +
164 "while True:\\n\" +
165 "    if i%009 == 0:\\n\" +
166 "        break\\n\" +
167 "        print(i)\\n\" +
168 "    i += 1\\n\", 4, '\"Error,y;5 6 7 8,n;5 6 7 8 9,n\"),\" +
169 "(47, \"What is the output of the following?\ni = 1\\n\" +
170 "while True:\\n\" +
171 "    if i%2 == 0:\\n\" +
172 "        break\\n\" +
173 "        print(i)\\n\" +
174 "    i += 2\\n\", 4, '\"1 3 5 6 9 11..,y;1 2,n;1,n\"),\" +
175 "(48, \"What is the output of the following?\ni = 2\\n\" +
176 "while True:\\n\" +
177 "    if i%3 == 0:\\n\" +
178 "        break\\n\" +
179 "        print(i)\\n\" +
180 "    i += 2\\n\", 4, '\"2 4,y;2 3,n;2 4 6 8 10..,n\"),\" +
181 "(49, \"What is the output of the following?\ni = 1\\n\" +
182 "while False:\\n\" +
183 "    if i%2 == 0:\\n\" +
184 "        break\\n\" +
185 "        print(i)\\n\" +
186 "    i += 2\\n\", 4, '\"None of the below,y;1,n;1 3 5 7..,n\"),\" +
187 "(50, \"What is the output of the following?\nTrue = False\\n\" +
188 "while True:\\n\" +
189 "    print(True)\\n\" +
190 "    break\\n\", 4, '\"None of the below,y;None,n;True,n\"),\" +
191 "(51, \"What is the output of the following?\ni = 0\\n\" +
192 "while i < 5:\\n\" +
193 "    print(i)\\n\" +
194 "    i += 1\\n\" +
195 "    if i == 3:\\n\" +
196 "        break\\n\" +
197 "else:\\n\" +
198 "    print(0)\\n\", 4, '\"0 1 2,y;Error,n;0 1 2 ,n\"),\" +
199 "(52, \"What is the output of the following?\ni = 0\\n\" +
200 "while i < 3:\\n\" +
201 "    print(i)\\n\" +
202 "    i += 1\\n\" +
203 "else:\\n\" +
204 "    print(0)\\n\", 4, '\"0 1 2 0,y;0 1 2 3,n;Error,n\"),\" +
205 "(53, \"What is the output of the following?\nx='abcdef'\nwhile i in x:\\n\" +
206 "    print(i, end= ' ')\\n\", 4, '\"i i i i ..,y;abcdef,n;a b c d e f,n\"),\" +
207 "(54, \"What is the output of the following?\nx = 'abcdef'\\n\" +
208 "i = 'i'\\n\" +
209 "while i in x:\\n\" +
210 "    print(i, end= ' ')\\n\", 4, '\"Nothing,y;abcdef,n;i i i i ..,n\"),\" +
211 "(55, \"What is the output of the following?\nx = 'abcdef'\\n\" +
212 "i = 'a'\\n\" +
213 "while i in x:\\n\" +
214 "    print(i, end = ' ')\\n\", 4, '\"a a a a ..,y;Nothing,n;i i i i ..,n\"),\" +
215 "(56, \"What is the output of the following?\nx = 'abcdef'\\n\" +
216 "i = 'a'\\n\" +
217 "while i in x:\\n\" +
218 "    print('i', end = ' ')\\n\", 4, '\"i i i i ..,y;a b c d e f,n;abcdef,n\"),\" +
219
220 // quiz 5
221 //id body quizNo answers
222 "(57, \"In a Python program, a control structure\", 5, \"Directs the order of execution of the statements in the program,y;Dictates what happens before
223 "(58, \"Which one of the following if statements will not execute\", 5, \"if (1, 2):print('foo'),y;if (1, 2): print('foo'),n;if (1, 2):\n\tprint('foo')
224 "(59, \"What signifies the end of a statement block\", 5, \"A line that is indented less than the previous line,y:end,n;n\\\"),\" +
225 "(60, \"What is the output of the following?\nif 'bar' in {'foo': 1, 'bar': 2, 'baz': 3}:\n    \" +
226 "    print(1)\\n\" +
227 "    print(2)\\n\" +
228 "    if 'a' in 'qux':\\n\" +
229 "        print(3)\\n\" +
230 "print(4)\\n\", 5, '\"1 2 4,y;4,n;Nothing,n\"),\" +
231 "(61, \"What is the output of the following?\nbool('False')\\n\" +
232 "bool()\\n\", 5, '\"True False,y;False True,n;Error,n\"),\" +

```

Στις γραμμές 19-30 ορίζουμε static μεταβλητές οι οποίες χρησιμοποιούνται κατά τον σχολιασμό των βαθμών ενός μαθητή.

Στις γραμμές 38-49 δημιουργούμε μια τοπική βάση δεδομένων η οποία περιέχει τους πίνακες: Students, Lessons, Teachers και Questions.

Στις γραμμές 79-87 εισάγουμε δεδομένα στη βάση Lessons.

Στις γραμμές 89-317 εισάγουμε δεδομένα στη βάση Questions.

```
319     @Override
320     public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
321         db.execSQL("DROP TABLE IF EXISTS Students");
322         db.execSQL("DROP TABLE IF EXISTS Lessons");
323         db.execSQL("DROP TABLE IF EXISTS Questions");
324         db.execSQL("DROP TABLE IF EXISTS Teachers");
325         onCreate(db);
326     }
327
328     public boolean insertTeacher (String username, String password) {
329         SQLiteDatabase db = this.getWritableDatabase();
330         ContentValues contentValues = new ContentValues();
331         contentValues.put("username", username);
332         contentValues.put("password", password);
333         long res = db.insert("Teachers", null, contentValues);
334         return res != -1; // if res == -1 then something gone wrong
335     }
336
337     public boolean findTeacher (String username, String password) {
338         SQLiteDatabase db = this.getWritableDatabase();
339         String[] data = {username, password};
340         Cursor res;
341         try {
342             res = db.rawQuery("SELECT * FROM Teachers WHERE username = ? AND password = ?", data);
343             res.moveToFirst();
344             boolean b = res.getCount() == 1; // check if teacher exists
345
346             if (!res.isClosed()) {
347                 res.close();
348             }
349
350             return b;
351         } catch (Exception e) {
352             Log.e("Silly-Err: ", e.getMessage());
353             return false;
354         }
355     }
356
357     public boolean changeTeachersPassword(String username, String newPassword) {
358         SQLiteDatabase db = this.getWritableDatabase();
359
360         // update password
361         String[] data = {newPassword, username};
362         Cursor res;
363
364         try {
365             res = db.rawQuery("UPDATE Teachers SET password = ? WHERE username = ? ", data);
366             res.moveToFirst();
367
368             boolean b = res.getPosition() != -1; // did something gone wrong
369
370             if (!res.isClosed()) {
371                 res.close();
372             }
373         }
```

```
372         }
373
374         return b;
375     } catch (Exception e) {
376         Log.e("Silly-Err: ", e.getMessage());
377         return false;
378     }
379 }
380
381 public boolean insertQuestion(String question, int quiz, String answers) {
382     SQLiteDatabase db = this.getWritableDatabase();
383     ContentValues contentValues = new ContentValues();
384     contentValues.put("body", question);
385     contentValues.put("quizNo", quiz);
386     contentValues.put("answers", answers);
387     long res = db.insert("Questions", null, contentValues);
388     return res != -1; // if res == -1 then something gone wrong
389 }
390
391 public boolean insertUser (String username, String password) {
392     SQLiteDatabase db = this.getWritableDatabase();
393     ContentValues contentValues = new ContentValues();
394     contentValues.put("username", username);
395     contentValues.put("password", password);
396     contentValues.put("marks", "0,0,0,0,0,0,0"); // 7 quizzes total, the seventh one is the final
397     long res = db.insert("Students", null, contentValues);
398     return res != -1;
399 }
400
401 public boolean userExists(){
402     SQLiteDatabase db = this.getReadableDatabase();
403     int numOfRows = (int) DatabaseUtils.queryNumEntries(db, "Students");
404     return numOfRows == 1; // check if user already registered in the local database
405 }
406
407 public boolean findUsername (String username) {
408     SQLiteDatabase db = this.getWritableDatabase();
409     String[] data = {username};
410     Cursor res;
411     try {
412         res = db.rawQuery("SELECT * FROM Students WHERE username = ?", data);
413         res.moveToFirst();
414         boolean b = res.getCount() == 1; // check if user exists
415
416         if (!res.isClosed()) {
417             res.close();
418         }
419
420         return b;
421     } catch (Exception e) {
422         Log.e("Silly-Err: ", e.getMessage());
423         return false;
424     }
425 }
```

Στις γραμμές 320-326 διαγράφουμε τους πίνακες της τοπικής βάσης δεδομένων σε περίπτωση που έχει αλλάξει κάτι στη δομή της βάσης.

Στις γραμμές 328-335 εισάγουμε νέο εξεταστή στον πίνακα Teachers και ελέγχουμε εάν η ενέργεια αυτή είναι επιτυχής.

Στις γραμμές 337-355 ψάχνουμε στον πίνακα Teachers για κάποια γραμμή που έχει κάποιο συγκεκριμένο username και password και ελέγχουμε εάν η ενέργεια αυτή είναι επιτυχής.

Στις γραμμές 357-379 αλλάζουμε στον πίνακα Teachers σε κάποια γραμμή με συγκεκριμένο username το πεδίο password και ελέγχουμε εάν η ενέργεια αυτή είναι επιτυχής.

Στις γραμμές 381-389 προσθέτουμε νέα ερώτηση στον πίνακα Questions και ελέγχουμε αν η ενέργεια αυτή είναι επιτυχής.

Στις γραμμές 391-399 προσθέτουμε νέο μαθητή στον πίνακα Students και ελέγχουμε αν η ενέργεια αυτή είναι επιτυχής.

Στις γραμμές 401-405 ελέγχουμε στον πίνακα Students αν υπάρχει στοιχείο με κάποιο συγκεκριμένο username και password.

Στις γραμμές 407-425 ελέγχουμε αν υπάρχει στοιχείο στον πίνακα με βάση ένα συγκεκριμένο username και ελέγχουμε αν η ενέργεια αυτή είναι επιτυχής.

```
426
427     public boolean findUser (String username, String password) {
428         SQLiteDatabase db = this.getWritableDatabase();
429         String[] data = {username, password};
430         Cursor res;
431         try {
432             res = db.rawQuery("SELECT * FROM Students WHERE username = ? AND password = ?", data);
433             res.moveToFirst();
434             boolean b = res.getCount() == 1; // check if user exists
435
436             if (!res.isClosed()) {
437                 res.close();
438             }
439
440             return b;
441         } catch (Exception e) {
442             Log.e("Silly-Err: ", e.getMessage());
443             return false;
444         }
445     }
446
447     public boolean changePassword(String username, String newPassword) {
448         SQLiteDatabase db = this.getWritableDatabase();
449
450         // update password
451         String[] data = {newPassword, username};
452         Cursor res;
453
454         try {
455             res = db.rawQuery("UPDATE Students SET password = ? WHERE username = ? ", data);
456             res.moveToFirst();
457
458             boolean b = res.getPosition() != -1; // did something gone wrong
459
460             if (!res.isClosed()) {
461                 res.close();
462             }
463
464             return b;
465         } catch (Exception e) {
466             Log.e("Silly-Err: ", e.getMessage());
467             return false;
468         }
469     }
470
471     public String getMarks (String username) {
472         // get user's marks
473         SQLiteDatabase db = this.getWritableDatabase();
474         String[] data = {username};
475         Cursor res;
476
477         try {
478             res = db.rawQuery("SELECT * FROM Students WHERE username = ?", data);
479             res.moveToFirst();
```

```

480         // get user's marks
481         int marksIndex = res.getColumnIndexOrThrow("marks"); // -> -1 if column doesn't exist
482         String marks = res.getString(marksIndex); // ex: "0.55,0.59,0.63,0.89,0.0,0.0,0.0"
483
484         if (!res.isClosed()) {
485             res.close();
486         }
487     }
488
489     return marks;
490 } catch (Exception e) {
491     Log.e("Some-Stupid-Err: ", e.getMessage());
492     return "";
493 }
494 }
495
496 public boolean updateMark (String username, Double mark, int quiz) { // quiz: 1-7, mark: 0.0-1.0
497     // get user's marks
498     SQLiteDatabase db = this.getWritableDatabase();
499     String[] data = {username};
500     Cursor res;
501
502     try {
503         res = db.rawQuery("SELECT * FROM Students WHERE username = ?", data);
504         res.moveToFirst();
505
506         // get user's marks
507         int marksIndex = res.getColumnIndex("marks");
508         String[] marks = res.getString(marksIndex).split(",");
509
510         // update marks array
511         marks[quiz-1] = Double.toString(mark);
512         String finalMarks = "";
513         for (int i = 0; i < marks.length; i++) {
514             if (i < marks.length - 1) {
515                 finalMarks = finalMarks + marks[i] + ",";
516             } else {
517                 finalMarks = finalMarks + marks[i];
518             }
519         }
520
521         // update user's marks in database
522         String[] data2 = {finalMarks, username};
523
524         Cursor res2;
525
526         try {
527             res2 = db.rawQuery("UPDATE Students SET marks = ? WHERE username = ?", data2);
528             res2.moveToFirst();
529
530             boolean b2 = res2.getPosition() != -1; // did something gone wrong
531
532             if (!res.isClosed()) {
533                 res.close();
534             }

```

```
535             if (!res2.isClosed()) {
536                 res2.close();
537             }
538         }
539         return b2;
540     } catch (SQLiteException e) {
541         Log.e("Silly-Err: ", e.getMessage());
542         return false;
543     }
544     } catch (SQLiteException e) {
545         Log.e("Silly-Err: ", e.getMessage());
546         return false;
547     }
548 }
549 }
550
551 public String commentMark(int quizIndex, Double mark) { // quiz: 1-7, mark: 0.0-1.0
552     if (mark <= 0.63) {
553         return PyGuruHelper.bad[quizIndex];
554     }
555
556     if (mark > 0.63 && mark <= 0.75) {
557         return PyGuruHelper.good;
558     }
559
560     if (mark > 0.75 && mark < 0.9) {
561         return PyGuruHelper.veryGood;
562     }
563
564     return PyGuruHelper.excellent;
565 }
566
567 public ArrayList<HashMap<String, String>> getQuestions(int quiz) { // quiz: 1-7 and NOT 0-6!
568     SQLiteDatabase db = this.getWritableDatabase();
569     String[] data = {Integer.toString(quiz)};
570     ArrayList<HashMap<String, String>> questions = new ArrayList();
571
572     // find quiz's questions
573     Cursor res;
574
575     try {
576         res = db.rawQuery("SELECT * FROM Questions WHERE quizNo = ?", data);
577         res.moveToFirst();
578
579         while (res.isAfterLast() == false) {
580             HashMap<String, String> item = new HashMap();
581             item.put("id", res.getString(res.getColumnIndex("id")));
582             item.put("body", res.getString(res.getColumnIndex("body")));
583             item.put("quizNo", res.getString(res.getColumnIndex("quizNo")));
584             item.put("answers", res.getString(res.getColumnIndex("answers")));
585
586             questions.add(item);
587             res.moveToNext();
588         }
589     }
590 }
```

```

589             res.moveToNext();
590         }
591
592         if (!res.isClosed()) {
593             res.close();
594         }
595
596         // return seven random questions for simple quizzes(1-6)
597         // and 14 random questions for the final quiz
598         ArrayList<HashMap<String, String>> result = new ArrayList(); // final array list
599         Collections.shuffle(questions); // randomize questions
600
601         // check if we are about to view the final quiz or the simple ones
602         if (quiz < 7) {
603             for (int i = 0; i < 7; i++) {
604                 result.add(questions.get(i)); // take first 7 random questions
605             }
606         } else {
607             for (int i = 0; i < 14; i++) {
608                 result.add(questions.get(i)); // take first 14 random questions
609             }
610         }
611
612         return result;
613     } catch (SQLiteException e) {
614         Log.e("Silly-Err: ", e.getMessage());
615         return new ArrayList();
616     }
617 }
618
619     public ArrayList<HashMap<String, String>> getLessonsTitles() {
620         SQLiteDatabase db = this.getWritableDatabase();
621         ArrayList<HashMap<String, String>> lessons = new ArrayList();
622         Cursor res;
623
624         try {
625             res = db.rawQuery("SELECT * FROM Lessons", new String[]{});
626             res.moveToFirst();
627
628             while (res.isAfterLast() == false) {
629                 HashMap<String, String> item = new HashMap();
630                 item.put("title", res.getString(res.getColumnIndex("title")));
631                 item.put("lastVisit", res.getString(res.getColumnIndex("lastVisit")));
632
633                 lessons.add(item);
634                 res.moveToNext();
635             }
636
637             if (!res.isClosed()) {
638                 res.close();
639             }
640
641             return lessons;
642         } catch (Exception e) {
643             Log.e("Silly-Err: ", e.getMessage());

```

```
642     } catch (Exception e) {
643         Log.e("Silly-Err: ", e.getMessage());
644         return new ArrayList();
645     }
646 }
647
648 public String[] getLesson(int lesson){ // lesson: 1-6
649     SQLiteDatabase db = this.getWritableDatabase();
650     String[] data = {Integer.toString(lesson)};
651     Cursor res;
652     Cursor res2;
653
654     // store the date of last visit
655     Calendar c = Calendar.getInstance();
656     SimpleDateFormat dateFormat = new SimpleDateFormat("dd MMM yyyy - HH:mm:ss a");
657     String datetime = dateFormat.format(c.getTime());
658
659     try {
660         // find lesson
661         res = db.rawQuery("SELECT * FROM Lessons WHERE id = ?", data);
662         res.moveToFirst();
663
664         // get lesson's content
665         int bodyIndex = res.getColumnIndex("body");
666         int titleIndex = res.getColumnIndex("title");
667         int visitsIndex = res.getColumnIndex("visits");
668
669         String[] lessonEntry = {res.getString(titleIndex), res.getString(bodyIndex), Integer.toString(res.getInt(visitsIndex) + 1)};
670
671         // update lesson's visits whenever someone click it
672         String visits = Integer.toString(res.getInt(visitsIndex) + 1);
673         String[] data2 = {visits, datetime, Integer.toString(lesson)};
674
675         try {
676             res2 = db.rawQuery("UPDATE Lessons SET visits = ?, lastVisit = ? WHERE id = ? ", data2);
677             res2.moveToFirst();
678
679             if (!res.isClosed()) {
680                 res.close();
681             }
682
683             if (!res2.isClosed()) {
684                 res2.close();
685             }
686
687             return lessonEntry; // ex: title:"foo bar baz", body:"lorem ipsum...", visits:33 -> current user's visits!
688         } catch (Exception e) {
689             Log.e("Silly-Err: ", e.getMessage());
690             return new String[]{};
691         }
692         } catch (Exception e) {
693             Log.e("Silly-Err: ", e.getMessage());
694             return new String[]{};
695         }
696     }
```

Στις γραμμές 427-445 ελέγχουμε αν υπάρχει μαθητής στον πίνακα Students με συγκεκριμένο username και password και ελέγχουμε αν η ενέργεια αυτή είναι επιτυχής.

Στις γραμμές 447-469 αλλάζουμε το πεδίο password ενός στοιχείου του πίνακα Students με συγκεκριμένο username.

Στις γραμμές 471-494 βρίσκουμε έναν μαθητή στον πίνακα Students με βάση το όνομα του και επιστρέφουμε την τιμή του πεδίου marks το οποίο είναι όλοι οι βαθμοί ενός μαθητή. Έπειτα ελέγχουμε αν είναι επιτυχής η ενέργεια αυτή.

Στις γραμμές 496-550 βρίσκουμε ένα μαθητή στον πίνακα Students με βάση το όνομα του και αλλάζουμε στο πεδίο marks το βαθμό ενός συγκεκριμένου τεστ. Μετά ελέγχουμε αν η ενέργεια είναι επιτυχής.

Στις γραμμές 551-565 σχολιάζουμε το βαθμό ενός τεστ. Σε περίπτωση που ο βαθμός δεν είναι ικανοποιητικός επιστρέφουμε ένα string που περιέχει την αδυναμία που έχει ο μαθητής και τι πρέπει να κάνει για να εκληφθεί η αδυναμία του.

Στις γραμμές 567-615 βρίσκουμε στον πίνακα Questions τα στοιχεία που έχουν ένα συγκεκριμένο αριθμό quiz. Μετά ελέγχουμε αν είναι επιτυχής η ενέργεια αυτή.

Στις γραμμές 617-643 συλλέγουμε όλα τα δεδομένα από τον πίνακα Lessons και επιστρέφουμε μόνο τις τιμές των πεδίων title το οποίο παριστάνει τον τίτλο ενός μαθήματος.

Στις γραμμές 645-689 βρίσκουμε στον πίνακα Lessons το στοιχείο που έχει ένα συγκεκριμένο id, αυξάνουμε τον αριθμό επισκέψεων του μαθητή, αποθηκεύουμε την

τρέχουσα ημερομηνία και ώρα, ελέγχουμε αν έγιναν επιτυχώς αυτές οι ενέργειες και επιστρέφουμε όλα τα πεδιά του μαθήματος αυτού αν δεν υπάρχει κάποιο σφάλμα.

RegisterActivity.kt:

```
13  class RegisterActivity : AppCompatActivity() {
14
15      override fun onCreate(savedInstanceState: Bundle?) {
16          super.onCreate(savedInstanceState)
17          setContentView(R.layout.activity_register)
18
19          val dbHelper = PyGuruHelper(this)
20
21          val registerButton : Button = findViewById(R.id.registerButton)
22          registerButton.setOnClickListener {
23              // get user's input
24              val username : TextView = findViewById(R.id.newUsername)
25              val password : TextView = findViewById(R.id.newPassword)
26
27              if (!username.text.isEmpty() && !password.text.isEmpty()) {
28                  var res : Boolean = dbHelper.userExists()
29
30                  // student already registered
31                  if (res) {
32                      Toast.makeText(this, "You can have only 1 account per device.", Toast.LENGTH_LONG).show()
33                  } else {
34                      // new student
35                      res = dbHelper.insertUser(username.text.toString(), password.text.toString())
36                      if (res) {
37                          // go to login screen
38                          val intent = Intent(this, MainActivity::class.java)
39                          startActivity(intent) // start login activity
40                      } else {
41                          // something gone wrong try again
42                          Toast.makeText(this, "Something gone wrong, please try again.", Toast.LENGTH_LONG).show()
43                      }
44                  }
45              } else {
46                  Toast.makeText(this, "You must fill the form.", Toast.LENGTH_LONG).show()
47              }
48          }
49      }
50  }
```

ΣΤΙΣ γραμμές 21-49 ελέγχουμε αν ο χρήστης έχει συμπληρώσει τη φόρμα και μετρά αν έχει φτιάξει ήδη λογαριασμό. Αν συμβαίνει ένα από τα δυο εμφανίζουμε στην οθόνη του χρήστη το κατάλληλο μνήμα διαφορετικά κατευθύνουμε τον χρήστη στην σελίδα MainActivity για να συνδεθεί.

LessonsActivity.kt:

```
10  class LessonsActivity : AppCompatActivity() {
11
12     override fun onCreate(savedInstanceState: Bundle?) {
13         super.onCreate(savedInstanceState)
14         setContentView(R.layout.activity_lessons)
15
16         val dbHelper : PyGuruHelper = PyGuruHelper(this)
17
18         val lessonsViews : List<TextView> = listOf(
19             findViewById(R.id.lesson1),
20             findViewById(R.id.lesson2),
21             findViewById(R.id.lesson3),
22             findViewById(R.id.lesson4),
23             findViewById(R.id.lesson5),
24             findViewById(R.id.lesson6)
25         )
26
27         // setup click listeners which load the correct lesson
28         lessonsViews.forEachIndexed { index, lessonView ->
29             lessonView.setAllCaps(false)
30             lessonView.setOnClickListener {
31                 // store current lesson id
32                 val sharedPref: SharedPreferences = this.getSharedPreferences("PyGuruStudent", Context.MODE_PRIVATE)
33                 val editor : SharedPreferences.Editor = sharedPref.edit()
34                 editor.putInt("PyGuruLesson", index+1)
35                 editor.commit()
36
37                 val intent = Intent(this, LessonActivity::class.java)
38                 startActivity(intent) // start lesson activity
39             }
40         }
41
42         // get lesson titles from db
43         val lessonsTitles = dbHelper.getLessonsTitles()
44
45         if (lessonsTitles.isNotEmpty()) {
46             // update each textview's text
47             lessonsTitles.forEachIndexed { index, lessonsTitle ->
48                 val title = lessonsTitle.get("title")
49                 val lastVisit = lessonsTitle.get("lastVisit")
50                 val txt = lessonsViews[index].text.toString()
51                 lessonsViews[index].text = txt + " " + title + "\nlast visit: " + lastVisit
52             }
53         }
54     }
55 }
```

Στις γραμμές 27-53 προσθέτουμε event click listeners σε κάθε κουμπί που παριστάνει σύνδεσμο σε κάποιο μαθημα. Οι συναρτήσεις που διαχειρίζονται το click event αποθηκεύουν τον αριθμό του μαθήματος για μελλοντική χρήση τοπικά και μεταφέρνουν τον χρήστη στη σελίδα LessonActivity. Έπειτα βρίσκουμε όλους τους τίτλους των μαθητών και την τελευταία ημερομηνία επίσκεψης του κάθε μαθήματος και όλα αυτά τα χρησιμοποιούμε στο κείμενο των κουμπιών συνδέσμων.

QuizzesActivity.kt:

```
10  class QuizzesActivity : AppCompatActivity() {
11
12     override fun onCreate(savedInstanceState: Bundle?) {
13         super.onCreate(savedInstanceState)
14         setContentView(R.layout.activity_quizzes)
15
16         // find all quizzes
17         val quizzes : List<TextView> = listOf(
18             findViewById(R.id.quiz1),
19             findViewById(R.id.quiz2),
20             findViewById(R.id.quiz3),
21             findViewById(R.id.quiz4),
22             findViewById(R.id.quiz5),
23             findViewById(R.id.quiz6)
24         )
25
26         val finalQuiz : TextView = findViewById(R.id.quiz7)
27         finalQuiz.setAllCaps(false)
28
29         // setup listeners
30         quizzes.forEachIndexed { index, quiz ->
31             quiz.setAllCaps(false)
32             quiz.setOnClickListener{
33                 // store current quiz id
34                 val sharedPref: SharedPreferences = this.getSharedPreferences("PyGuruStudent", Context.MODE_PRIVATE)
35                 val editor : SharedPreferences.Editor = sharedPref.edit()
36                 editor.putInt("PyGuruQuiz", index+1)
37                 editor.commit()
38
39                 // go to that quiz
40                 val intent = Intent(this, QuizActivity::class.java)
41                 startActivity(intent) // start quiz activity
42             }
43         }
44
45         finalQuiz.setOnClickListener {
46             // store current quiz id
47             val sharedPref: SharedPreferences = this.getSharedPreferences("PyGuruStudent", Context.MODE_PRIVATE)
48             val editor : SharedPreferences.Editor = sharedPref.edit()
49             editor.putInt("PyGuruQuiz", 7)
50             editor.commit()
51
52             // go to that quiz
53             val intent = Intent(this, FinalQuizActivity::class.java)
54             startActivity(intent) // start final quiz activity
55         }
56     }
57 }
```

Σε αυτό το αρχείο τοποθετούμε σε όλα τα κουμπιά που αναπαριστούν συνδέσμους στα καταλληλά quiz click event listeners. Ο κάθε listener όταν καλείται αποθηκεύει τον αριθμό του quiz τοπικά μελλοντική χρήση και μεταφέρει τον χρήστη στη σελίδα QuizActivity αν είναι σύνδεσμος για ένα απλό quiz και στο FinalQuizActivity αν είναι το τελικό quiz.

ForgotPwdActivity.kt:

```
10  class ForgotPwdActivity : AppCompatActivity() {
11
12      override fun onCreate(savedInstanceState: Bundle?) {
13          super.onCreate(savedInstanceState)
14          setContentView(R.layout.activity_forgot_pwd)
15
16          val dbHelper = PyGuruHelper(this)
17
18          val fpwdBtn : Button = findViewById(R.id.fpwd_btn)
19          fpwdBtn.setOnClickListener {
20              val uname : String = findViewById<TextView>(R.id.fpwd_uname).text.toString()
21              val pwd : String = findViewById<TextView>(R.id.fpwd_password).text.toString()
22
23              if (uname.isNotEmpty() && pwd.isNotEmpty()) {
24                  val res = dbHelper.findUsername(uname)
25                  if (res) {
26                      // update password
27                      val res = dbHelper.changePassword(uname, pwd)
28
29                      // show failure message
30                      if (!res) {
31                          Toast.makeText(this, "Something gone wrong try again.", Toast.LENGTH_LONG).show()
32                      }
33
34                      // show success message
35                      Toast.makeText(this, "Your password was successfully changed.", Toast.LENGTH_LONG).show()
36
37                      // go to login activity
38                      val intent = Intent(this, MainActivity::class.java)
39                      startActivity(intent) // start login activity
40                  } else {
41                      Toast.makeText(this, "Invalid username.", Toast.LENGTH_LONG).show()
42                  }
43              } else {
44                  // show error
45                  Toast.makeText(this, "You must fill the form.", Toast.LENGTH_LONG).show()
46              }
47          }
48      }
49  }
```

Στις γραμμές 18-48 προσθέτουμε έναν click listener στο κουμπί της φόρμας που είναι υπεύθυνη για την αλλαγή κωδικού. Ο click listener ελέγχει αν έχουν δοθεί όλα τα απαραίτητα στοιχεία από τον χρήστη και αλλάζει τον κωδικό πρόσβασης του. Εάν λείπει κάποιο από τα στοιχεία της φόρμας η προκύψει κάποιο σφάλμα κατά την αλλαγή του κωδικού του χρήστη τότε εμφανίζουμε το κατάλληλο μήνυμα στην οθόνη.

ProgressActivity.kt:

```
9  class ProgressActivity : AppCompatActivity() {
10
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         setContentView(R.layout.activity_progress)
14
15         val dbHelper : PyGuruHelper = PyGuruHelper(this)
16         val marksViews : List<TextView> = listOf(
17             findViewById(R.id.test1),
18             findViewById(R.id.test2),
19             findViewById(R.id.test3),
20             findViewById(R.id.test4),
21             findViewById(R.id.test5),
22             findViewById(R.id.test6),
23             findViewById(R.id.test7) // final test!
24         )
25         val commentsViews : List<TextView> = listOf(
26             findViewById(R.id.comment1),
27             findViewById(R.id.comment2),
28             findViewById(R.id.comment3),
29             findViewById(R.id.comment4),
30             findViewById(R.id.comment5),
31             findViewById(R.id.comment6),
32             findViewById(R.id.comment7) // final test!
33     )
34
35     // get user's name
36     val sharedPref: SharedPreferences = this.getSharedPreferences("PyGuruStudent", Context.MODE_PRIVATE)
37     val username : String = sharedPref.getString("PyGuruUser", "")
38
39     val marksColView : TextView = findViewById(R.id.marksColumn)
40     val txt = marksColView.text.toString()
41     marksColView.text = username + "'s " + txt
42
43     // find user's marks
44     val marksStr : String = dbHelper.getMarks(username)
45
46     if (!marksStr.isEmpty()) {
47         val marksStrs = marksStr.split(",")
48         val realMarks = marksStrs.map { markStr -> markStr.toDouble() }
49
50         // display marks and comments
51         marksViews.forEachIndexed { index, markView ->
52             // check if user passed the current quiz
53             val mark = marksStrs[index].toDouble()
54
55             var testRes : String = "FAIL"
56             if (mark >= 0.63) {
57                 testRes = "PASS"
58             }
59
60             // show mark
61             markView.text = markView.text.toString() + " " + marksStrs[index] + " " + testRes
62             commentsViews[index].text = dbHelper.commentMark(index, realMarks[index]) // show comment
63         }
64     }
65 }
```

Στις γραμμές 15-23 δημιουργούμε references για τα TextViews που περιέχουν τους βαθμούς του χρήστη.

Στις γραμμές 25-33 δημιουργούμε references για τα TextViews που περιέχουν σχόλια για τους βαθμούς του χρήστη.

Στις γραμμές 43-63 κάνουμε τα εξής:

Βρίσκουμε τους βαθμούς του χρήστη μέσω του ονόματος του που έχει αποθηκευτεί στη συσκευή, σχολιάζουμε τους

βαθμούς του και δείχνουμε τα σχόλια και τους βαθμούς του. Για την παρουσίαση των βαθμών και των σχολίων του αλλάζουμε το κείμενο των TextView references που έχουμε δημιουργήσει.

ProfileActivity.kt:

```
10  class ProfileActivity : AppCompatActivity() {
11
12     override fun onCreate(savedInstanceState: Bundle?) {
13         super.onCreate(savedInstanceState)
14         setContentView(R.layout.activity_profile)
15
16         val dbHelper = PyGuruHelper(this)
17         // get user's name
18         val sharedPref: SharedPreferences = this.getSharedPreferences("PyGuruStudent", Context.MODE_PRIVATE)
19         val username = sharedPref.getString("PyGuruUser", "")
20
21
22         val newPassword : TextView = findViewById(R.id.newPasswordField)
23         val repeatPassword : TextView = findViewById(R.id.repeatPasswordField)
24         val changePassword : Button = findViewById(R.id.changePassword)
25
26         changePassword.setOnClickListener {
27             val password1 = newPassword.text.toString()
28             val password2 = repeatPassword.text.toString()
29
30             if (!password1.isEmpty() && !password2.isEmpty()) {
31                 if (!password1.equals(password2, false)) {
32                     Toast.makeText(this, "Passwords are not equal.", Toast.LENGTH_LONG).show()
33                 } else {
34                     val res = dbHelper.changePassword(username, password1)
35                     if (!res) {
36                         Toast.makeText(this, "Something gone wrong try again.", Toast.LENGTH_LONG).show()
37                     }
38                     Toast.makeText(this, "Your password was successfully changed.", Toast.LENGTH_LONG).show()
39                 }
40             } else {
41                 Toast.makeText(this, "You must fill the form.", Toast.LENGTH_LONG).show()
42             }
43         }
44     }
45 }
46 }
```

Στις γραμμές 17-45 βρίσκουμε τα πεδία της φόρμας και τα αποθηκεύουμε, διαβάζουμε το όνομα του χρήστη που είναι αποθηκευμένο στη συσκευή και προσθέτουμε έναν click listener στο κουμπί της φόρμας. Ο click listener ελέγχει αν έχουμε συμπληρωθεί όλα τα πεδία της φόρμας, αν ο χρήστης έχει γράψει σωστά τον νέο κωδικό πρόσβασης, αλλάζει το κωδικό του χρήστη και ελέγχει αν η αλλαγή του κωδικού ήταν επιτυχής. Αν προέκυψε κάποιο σφάλμα κατά την αλλαγή του κωδικού του χρήστη ή αν κάποιο στοιχείο

της φόρμας λείπει ή δεν ταυτίζονται οι κωδικοί που έδωσε ο χρήστης τότε δείχνουμε στην οθόνη το κατάλληλο μήνυμα.

LessonActivity.kt:

```
9  class LessonActivity : AppCompatActivity() {
10
11    override fun onCreate(savedInstanceState: Bundle?) {
12        super.onCreate(savedInstanceState)
13        setContentView(R.layout.activity_lesson)
14
15        val dbHelper = PyGuruHelper(this)
16
17        // get lesson's id
18        val sharedPref: SharedPreferences = this.getSharedPreferences("PyGuruStudent", Context.MODE_PRIVATE)
19        val lessonId = sharedPref.getInt("PyGuruLesson", -1)
20
21        val lessonView: TextView = findViewById(R.id.currentLesson)
22        val lessonVisits: TextView = findViewById(R.id.visitsText)
23        val lessonBody: TextView = findViewById(R.id.lessonBody)
24
25        if (lessonId > 0) {
26            // get lesson's data -> array of 4 elements
27            val lessonData: List<String> = dbHelper.getLesson(lessonId).asList()
28
29            if (!lessonData.isEmpty()) {
30                // show lesson's data if data exist
31                lessonView.text = lessonData[0] // title
32                lessonBody.text = lessonData[1] // content
33                lessonVisits.text = "visits: " + lessonData[2] // visits + 1
34            } else {
35                // no data for that lesson id
36                lessonView.text = "<NONE>"
37                lessonBody.text = "<NONE>"
38                lessonVisits.text = "<NONE>"
39            }
40        } else {
41            // invalid lesson id
42            lessonView.text = "<NONE>"
43            lessonBody.text = "<NONE>"
44            lessonVisits.text = "<NONE>"
45        }
46    }
47 }
```

Στις γραμμές 15-46 διαβάζουμε το id του μαθήματος που έχει αποθηκευτεί στη συσκευή, βρίσκουμε τα TextViews υπεύθυνα για την παρουσίαση του περιεχομένου του μαθήματος που θέλει να διαβάσει ο χρήστης, αναζητούμε στον πίνακα Lessons το μάθημα που έχει ίδιο id με αυτό που έχουμε αποθηκεύσει στη συσκευή και αλλάζουμε το κείμενο των TextView references.

TeacherLoginActivity.kt:

```
12 class TeacherLoginActivity : AppCompatActivity() {
13
14     override fun onCreate(savedInstanceState: Bundle?) {
15         super.onCreate(savedInstanceState)
16         setContentView(R.layout.activity_teacher_login)
17
18         val dbHelper = PyGuruHelper(this)
19
20         val registerTxt : TextView = findViewById(R.id.teacherRegister)
21         registerTxt.setOnClickListener {
22             // go to teacher's register activity
23             val intent = Intent(this, NewTeacherActivity::class.java)
24             startActivity(intent) // start new teacher activity
25         }
26
27         val loginBtn : Button = findViewById(R.id.teacherLoginBtn)
28         loginBtn.setOnClickListener {
29             val username : TextView = findViewById(R.id.teacherUname)
30             val password : TextView = findViewById(R.id.teacherPwd)
31
32             if (username.text.isNotEmpty() && password.text.isNotEmpty()) {
33                 val uname = username.text.toString()
34                 val pwd = password.text.toString()
35
36                 val res = dbHelper.findTeacher(uname, pwd)
37                 // teacher exists
38                 if (res) {
39                     // save teacher's username
40                     val sharedPref: SharedPreferences = this.getSharedPreferences("PyGuruStudent", Context.MODE_PRIVATE)
41                     val editor: SharedPreferences.Editor = sharedPref.edit()
42                     editor.putString("PyGuruTeacher", uname)
43                     editor.commit()
44
45                     // go to teacher's main menu
46                     val intent = Intent(this, TeacherMenuActivity::class.java)
47                     startActivity(intent) // start teacher's main menu activity
48                 } else {
49                     // teacher doesn't exist
50                     Toast.makeText(this, "Username/Password is wrong.", Toast.LENGTH_LONG).show()
51                 }
52             } else {
53                 Toast.makeText(this, "You must fill the form.", Toast.LENGTH_LONG).show()
54             }
55         }
56     }
}
```

Στις γραμμές 20-25 προσθέτουμε έναν click listener στο Text View που προτρέπει τον εξεταστή να κάνει εγγραφή αν δεν έχει φτιάξει ακόμη λογαριασμό. Αν ο click listener κληθεί μεταφέρει τον χρήστη στη σελίδα NewTeacherActivity για να κάνει την εγγραφή του.

Στις γραμμές 27-54 προσθέτουμε έναν click listener στο κουμπί της φόρμας που είναι υπεύθυνη για την σύνδεση του εξεταστή. Αν ο click listener κληθεί τότε ελέγχει αν ο χρήστης έχει δώσει όλα τα απαραίτητα στοιχεία και αν υπάρχει στον πίνακα Teachers εξεταστής με το δοσμένο username και password. Αν κάποιο σφάλμα προκύψει κατά

την εύρεση εξεταστή ή λείπει κάποιο πεδίο από τη φόρμα τότε δείχνουμε στην οθόνη το κατάλληλο μήνυμα. Αν υπάρχει εξεταστής στον πίνακα Teachers και δεν λείπουν στοιχεία από τη φόρμα τότε αποθηκεύεται το όνομα του εξεταστή και κατευθύνεται στη σελίδα TeacherMenuActivity.

NewTeacherActivity.kt:

```
10  class NewTeacherActivity : AppCompatActivity() {
11
12      override fun onCreate(savedInstanceState: Bundle?) {
13          super.onCreate(savedInstanceState)
14          setContentView(R.layout.activity_new_teacher)
15
16          val dbHelper = PyGuruHelper(this)
17          val registerBtn : Button = findViewById(R.id.teacher_register)
18          registerBtn.setOnClickListener {
19              val uname : TextView = findViewById(R.id.new_teacher_name)
20              val unameTxt = uname.text.toString()
21
22              val pwd : TextView = findViewById(R.id.new_teacher_pwd)
23              val pwdTxt = pwd.text.toString()
24
25              if (unameTxt.isNotEmpty() && pwdTxt.isNotEmpty()) {
26                  // check if teacher exists
27                  var res = dbHelper.findTeacher(unameTxt, pwdTxt)
28                  if (!res) {
29                      // new teacher
30                      res = dbHelper.insertTeacher(unameTxt, pwdTxt)
31                      if (res) {
32                          // go to login screen
33                          val intent = Intent(this, TeacherLoginActivity::class.java)
34                          startActivity(intent) // start login activity
35                      } else {
36                          // something gone wrong try again
37                          Toast.makeText(this, "Something gone wrong, please try again.", Toast.LENGTH_LONG).show()
38                      }
39                  } else {
40                      Toast.makeText(this, "Teacher exists.", Toast.LENGTH_LONG).show()
41                  }
42              } else {
43                  Toast.makeText(this, "You must fill the form.", Toast.LENGTH_LONG).show()
44              }
45          }
46      }
47 }
```

Στις γραμμές 16-45 προσθέτουμε click listener στο. Κουμπί της φόρμας που είναι υπεύθυνη για την εγγραφή ενός εξεταστή. Στον click listener ελέγχουμε αν ο χρήστης έχει συμπληρώσει τη φόρμα εγγραφής και αν υπάρχει εξεταστής με αυτά τα στοιχεία. Αν ισχύει ένα από τα δυο ενδεχόμενα εμφανίζουμε το κατάλληλο μήνυμα. Αν δεν ισχύει κάποιο από τα δυο ενδεχόμενα προσθέτουμε το νέο στοιχείο στον πίνακα Teachers και ελέγχουμε αν η ενέργεια αυτή ήταν

επιτυχής. Αν δεν ήταν τότε εμφανίζουμε το μήνυμα της αποτυχίας.

ProfileActivity.kt:

```
11  class ProfileActivity : AppCompatActivity() {
12
13      override fun onCreate(savedInstanceState: Bundle?) {
14          super.onCreate(savedInstanceState)
15          setContentView(R.layout.activity_profile)
16
17          val dbHelper = PyGuruHelper(this)
18          // get user's name
19          val sharedPref: SharedPreferences = this.getSharedPreferences("PyGuruStudent", Context.MODE_PRIVATE)
20          val username = sharedPref.getString("PyGuruUser", "")
21
22          val newPassword : TextView = findViewById(R.id.newPasswordField)
23          val repeatPassword : TextView = findViewById(R.id.repeatPasswordField)
24          val changePassword : Button = findViewById(R.id.changePassword)
25
26          changePassword.setOnClickListener {
27              val password1 = newPassword.text.toString()
28              val password2 = repeatPassword.text.toString()
29
30              if (!password1.isEmpty() && !password2.isEmpty()) {
31                  if (!password1.equals(password2, false)) {
32                      Toast.makeText(this, "Passwords are not equal.", Toast.LENGTH_LONG).show()
33                  } else {
34                      val res = dbHelper.changePassword(username, password1)
35                      if (!res) {
36                          Toast.makeText(this, "Something gone wrong try again.", Toast.LENGTH_LONG).show()
37                      }
38                      Toast.makeText(this, "Your password was successfully changed.", Toast.LENGTH_LONG).show()
39                  }
40              } else {
41                  Toast.makeText(this, "You must fill the form.", Toast.LENGTH_LONG).show()
42              }
43          }
44      }
45  }
```

Σε αυτό το αρχείο διαβάζουμε το όνομα του μαθητή που έχει αποθηκευτεί στη συσκευή κατά την σύνδεση του ενός μαθητή και προσθέτουμε έναν click listener στο κουμπί της φόρμας που είναι υπεύθυνο για την αλλαγή κωδικού ενός συνδεδεμένου μαθητή. Στον click listener ελέγχουμε αν ο μαθητής έχει συμπληρώσει όλα τα πεδία της φόρμας και αν τα πεδία είναι ίσα. Αν δεν είναι ιδιά ή λείπει κάποιο από αυτά τότε εμφανίζεται στην οθόνη του μαθητή το κατάλληλο μήνυμα. Αν είναι συμπληρωμένα τα πεδία και ιδιά τότε αλλάζουμε τον κωδικό του χρήστη στη βάση δεδομένων μας και ελέγχουμε αν έγινε με επιτυχία η αλλαγή αυτή. Αν

προέκυψε κάποιο σφάλμα ενημερώνουμε τον χρήστη με το κατάλληλο μήνυμα.

ResetTeacherPwdActivity.kt:

```
11  class ResetTeacherPwdActivity : AppCompatActivity() {
12
13     override fun onCreate(savedInstanceState: Bundle?) {
14         super.onCreate(savedInstanceState)
15         setContentView(R.layout.activity_reset_teacher_pwd)
16
17         val dbHelper = PyGuruHelper(this)
18         // get user's name
19         val sharedPref: SharedPreferences = this.getSharedPreferences("PyGuruStudent", Context.MODE_PRIVATE)
20         val teacher = sharedPref.getString("PyGuruTeacher", "")
21
22
23         val resetPwdBtn : Button = findViewById(R.id.reset_teachers_pwd)
24         resetPwdBtn.setOnClickListener {
25             val password1 = findViewById<TextView>(R.id.new_teachers_pwd).text.toString()
26             val password2 = findViewById<TextView>(R.id.repeat_teachers_pwd).text.toString()
27
28             if (!password1.isEmpty() && !password2.isEmpty()) {
29                 if (!password1.equals(password2, false)) {
30                     Toast.makeText(this, "Passwords are not equal.", Toast.LENGTH_LONG).show()
31                 } else {
32                     val res = dbHelper.changeTeachersPassword(teacher, password1)
33                     if (!res) {
34                         Toast.makeText(this, "Something gone wrong try again.", Toast.LENGTH_LONG).show()
35                     }
36                     Toast.makeText(this, "Your password was successfully changed.", Toast.LENGTH_LONG).show()
37                 }
38             } else {
39                 Toast.makeText(this, "You must fill the form.", Toast.LENGTH_LONG).show()
40             }
41         }
42     }
43 }
```

Στο αρχείο αυτό διαβάζουμε το όνομα του εξεταστή το οποίο έχει αποθηκευτεί στη συσκευή κατά τη σύνδεση του και προσθέτουμε έναν click listener στο κουμπί της φόρμας που είναι υπεύθυνο για την αλλαγή κωδικού ενός συνδεδεμένου εξεταστή. Στον click listener ελέγχουμε αν ο εξεταστής έχει συμπληρώσει όλα τα πεδία της φόρμας και αν τα πεδία είναι ίσα. Αν δεν είναι ιδιά ή λείπει κάποιο από αυτά τότε εμφανίζεται στην οθόνη του εξεταστή το κατάλληλο μήνυμα. Αν είναι συμπληρωμένα τα πεδία και ιδιά τότε αλλάζουμε τον κωδικό του εξεταστή στη βάση δεδομένων μας και ελέγχουμε αν έγινε με επιτυχία η αλλαγή αυτή. Αν προέκυψε κάποιο σφάλμα ενημερώνουμε τον χρήστη με το κατάλληλο μήνυμα.

TeacherMenuActivity.kt:

```
11  class TeacherMenuActivity : AppCompatActivity() {
12
13      override fun onCreate(savedInstanceState: Bundle?) {
14          super.onCreate(savedInstanceState)
15          setContentView(R.layout.activity_teacher_menu)
16
17          val welcomeTxt : TextView = findViewById<TextView>(R.id.teacher_welcome)
18          var txt : String = welcomeTxt.text.toString()
19
20          // get teacher's name
21          val sharedPref: SharedPreferences = this.getSharedPreferences("PyGuruStudent", Context.MODE_PRIVATE)
22          txt = txt + " " + sharedPref.getString("PyGuruTeacher", "") + " !"
23
24          // update text
25          welcomeTxt.text = txt
26
27          // setup listeners
28          val logoutBtn : Button = findViewById(R.id.teacher_logout)
29          logoutBtn.setOnClickListener {
30              // reset teacher's name
31              val sharedPref: SharedPreferences = this.getSharedPreferences("PyGuruStudent", Context.MODE_PRIVATE)
32              val editor : SharedPreferences.Editor = sharedPref.edit()
33              editor.putString("PyGuruTeacher", "")
34              editor.commit()
35
36              // go to login screen
37              val intent = Intent(this, TeacherLoginActivity::class.java)
38              startActivity(intent) // start login activity
39          }
40
41          val resetPwdBtn : Button = findViewById(R.id.reset_teacher_pwd)
42          resetPwdBtn.setOnClickListener {
43              // start reset password activity
44              val intent = Intent(this, ResetTeacherPwdActivity::class.java)
45              startActivity(intent) // start profile activity
46          }
47
48          val addQuestionBtn : Button = findViewById(R.id.add_question_btn)
49          addQuestionBtn.setOnClickListener {
50              // go to new question activity
51              val intent = Intent(this, NewQuestionActivity::class.java)
52              startActivity(intent) // start new question activity
53          }
54      }
55  }
```

Στις γραμμές 17-25 δημιουργούμε το κείμενο “Welcome <username>!”, όπου το username έχει αποθηκευτεί στη συσκευή από τη στιγμή που έχει κάνει σύνδεση ο χρήστης ως εξεταστής.

Στις γραμμές 27-53 προσθέτουμε click listeners στα κουμπιά για αποσύνδεση, αλλαγή κωδικού εξεταστή και πρόσθεση νέας ερώτησης. Οι click listeners αυτοί κατευθύνουν τον συνδεδεμένο εξεταστή στις σελίδες TeacherLoginActivity, ResetTeacherPwdActivity και NewQuestionActivity αντίστοιχα.

QuizActivity.kt:

```
13  class QuizActivity : AppCompatActivity() {
14
15      override fun onCreate(savedInstanceState: Bundle?) {
16          super.onCreate(savedInstanceState)
17          setContentView(R.layout.activity_quiz)
18
19          // get quiz's id
20          val sharedPref: SharedPreferences = this.getSharedPreferences("PyGuruStudent", Context.MODE_PRIVATE)
21          val quiz = sharedPref.getInt("PyGuruQuiz", -1)
22
23          // set title
24          val quizView : TextView = findViewById(R.id.quizView)
25          quizView.text = "Quiz " + quiz
26
27          // get quiz's questions
28          val dbHelper = PyGuruHelper(this)
29          val questions = dbHelper.getQuestions(quiz) // 7 questions!
30
31          // display questions
32          val questionsBtns : List<Button> = listOf(
33              findViewById(R.id.q_1), // q_x -> question x
34              findViewById(R.id.q_2),
35              findViewById(R.id.q_3),
36              findViewById(R.id.q_4),
37              findViewById(R.id.q_5),
38              findViewById(R.id.q_6),
39              findViewById(R.id.q_7)
40          )
41
42          questions.forEachIndexed { index, question ->
43              questionsBtns[index].text = question.get("body") // show question
44
45              // find answers -> ["answer-1,y", "answer-2,n", "answer-3,n"]
46              val answers = question.get("answers")!!.split(";")
47
48              // setup listeners that show possible answers
49              questionsBtns[index].setAllCaps(false)
50              questionsBtns[index].setOnClickListener {
51                  val quiz = sharedPref.getInt("PyGuruQuiz", -1)
52
53                  if (quiz > 0) {
54                      // save all choices(available answers)
55                      val editor: SharedPreferences.Editor = sharedPref.edit()
56                      editor.putString("PyGuruChoice1", answers[0])
57                      editor.commit()
58                      editor.putString("PyGuruChoice2", answers[1])
59                      editor.commit()
60                      editor.putString("PyGuruChoice3", answers[2])
61                      editor.commit()
62
63                      // save question's index
64                      editor.putString("PyGuruQuestion", (index + 1).toString())
65                      editor.commit()
66                  }
67              }
68          }
69      }
70  }
```

Στις γραμμές 20-21 διαβάζουμε το id του τεστ που θέλει να απαντήσει ο μαθητής. Το id του τεστ έχει αποθηκευτεί στη συσκευή από τη στιγμή που επέλεξε ο μαθητής το τεστ από το QuizzesActivity.

Στις γραμμές 23-25 θέτουμε το περιεχόμενο του TextView που δείχνει το τεστ που επέλεξε ο μαθητής με “Quiz <id>”.

```

67         // go pick an answer
68         val intent = Intent(this, AnswerActivity::class.java)
69         startActivity(intent) // start answer activity
70     } else {
71         Toast.makeText(this, "You have already completed the test.", Toast.LENGTH_LONG).show()
72     }
73 }
74
75 }
76
77 val doneBtn : Button = findViewById(R.id.finishedQuiz)
78 doneBtn.setOnClickListener {
79     val ans = listOf(
80         sharedPreferences.getString("PyGuruAnswer1", ""),
81         sharedPreferences.getString("PyGuruAnswer2", ""),
82         sharedPreferences.getString("PyGuruAnswer3", ""),
83         sharedPreferences.getString("PyGuruAnswer4", ""),
84         sharedPreferences.getString("PyGuruAnswer5", ""),
85         sharedPreferences.getString("PyGuruAnswer6", ""),
86         sharedPreferences.getString("PyGuruAnswer7", ""))
87 }
88
89 // check if every question is answered
90 if (ans.all { el -> !el.equals("") } && quiz > 0) {
91
92     // find correct answers
93     val correctAns = ans.filter { el ->
94         val content = el.split(",")
95         content[1].equals("y")
96     }
97
98     // calculate mark and update the old one
99     val mark = correctAns.size / 7.toDouble()
100    val username = sharedPreferences.getString("PyGuruUser", "")
101    val res = dbHelper.updateMark(username, mark, quiz)
102    if (!res) {
103        Toast.makeText(this, "Something went wrong! Try again.", Toast.LENGTH_LONG).show()
104    }
105
106    // make test invalid by removing the quiz's index
107    val editor : SharedPreferences.Editor = sharedPreferences.edit()
108    editor.putInt("PyGuruQuiz", -1)
109    editor.commit()
110
111    // remove stored answers
112    for (i in 1..7) {
113        editor.putString("PyGuruAnswer$i", "")
114        editor.commit()
115    }
116
117    // find which answers are right and which are wrong
118    var ansMsgs = ans.mapIndexed { i, a ->
119        val content = a.split(",")
120        var res = "Answer-" + (i+1) +": "+ content[0] + " -> "
121        if (content[1].equals("y")) {

```

Στις γραμμές 31-75 επιλέγουμε από τη βάση δεδομένων μας επτά τυχαίες ερωτήσεις για το επιλεγμένο τεστ και προσθέτουμε click listeners στα TextViews που περιέχουν της ερωτήσεις του επιλεγμένου τεστ. Ο κάθε click listener αφού ελέγχει την τιμή του id του τεστ αποθηκεύει και το id της ερώτησης και όλες τις πιθανές απαντήσεις στη συσκευή για μελλοντική χρήση και μεταφέρουμε τον μαθητή στη σελίδα AnswersActivity. Αν δεν είναι σωστό το id του τεστ δείχνουμε στην οθόνη του χρήστη το κατάλληλο μήνυμα.

```

121         if (content[1].equals("y")) {
122             res = res + "Correct"
123         } else {
124             res = res + "Wrong"
125         }
126     }
127 }
128
129 ansMsgs = listOf("Your answers:") + ansMsgs
130
131 // show user's mark and items
132 // create the builder
133 val dialogBuilder = AlertDialog.Builder(this)
134
135 // prepare items
136 val msgs = ansMsgs.plus("Your mark: ${mark.toString()}(${correctAns.size}/7)")
137 val items = msgs.toTypedArray() // convert string list to string array
138
139 // show the items
140 dialogBuilder.setItems(items, null)
141 dialogBuilder.setPositiveButton("OK", null)
142
143 // create alert box
144 val alertDialog = dialogBuilder.create()
145 // set title
146 alertDialog.setTitle("Your answers and mark")
147 // show alert box
148 alertDialog.show()
149 } else {
150     Toast.makeText(this, "Either there are some missing questions or you have already completed the test.", Toast.LENGTH_LONG).show()
151 }
152 }
153 }
154 }
```

Στις γραμμές 77-153 προσθέτουμε έναν click listener στο κουμπί που πατάει ο μαθητής όταν τελειώσει το τεστ. Σε αυτόν τον click listener ελέγχουμε αν έχει απαντήσει σε όλες τις ερωτήσεις και αν είναι έγκυρη η τιμή του id του quiz. Αν και οι δυο συνθήκες ισχύουν τότε φιλτράρουμε της σωστές απαντήσεις (οι οποίες έχουν αποθηκευτεί στη συσκευή από το AnswerActivity) για τον υπολογισμό του βαθμού του μαθητή σε αυτό το τεστ, διαγράφουμε από την συσκευή τον αριθμό του quiz και τις αποθηκευμένες απαντήσεις, αλλάζουμε τον βαθμό του χρήστη στη βάση δεδομένων μας και δείχνουμε στην οθόνη του χρήστη ένα παράθυρο που δείχνει σε ποιες ερωτήσεις είναι σωστές, ποιες είναι λάθος και ποιος είναι ο βαθμός του. Αν προκύψει κάποιο λάθος κατά την αλλαγή του βαθμού του χρήστη στο τεστ που επέλεξε ή αν δεν έχει απαντήσει όλες τις ερωτήσεις ή αν δεν είναι έγκυρο το id του τεστ τότε δείχνουμε το κατάλληλο μήνυμα στην οθόνη του χρήστη.

AnswerActivity.kt:

```
9  class AnswerActivity : AppCompatActivity() {
10
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         setContentView(R.layout.activity_answer)
14
15         val sharedPref: SharedPreferences = this.getSharedPreferences("PyGuruStudent", Context.MODE_PRIVATE)
16
17         val choices = listOf<String>(
18             sharedPref.getString("PyGuruChoice1", ""),
19             sharedPref.getString("PyGuruChoice2", ""),
20             sharedPref.getString("PyGuruChoice3", ""))
21     )
22
23         val question = sharedPref.getString("PyGuruQuestion", "")
24
25         val choicesBtns : List<RadioButton> = listOf(
26             findViewById(R.id.choice1),
27             findViewById(R.id.choice2),
28             findViewById(R.id.choice3)
29         )
30
31         choicesBtns.forEachIndexed { index , btn ->
32             // show answer
33             var choice = choices[index].split(",")
34             btn.text = choice[0] // choice[0] -> answer's text
35
36             // store selected answer
37             btn.setOnClickListener {
38                 val editor: SharedPreferences.Editor = sharedPref.edit()
39                 editor.putString("PyGuruAnswer" + question, choices[index])
40                 editor.commit()
41             }
42         }
43     }
44 }
```

Σε αυτό το αρχείο διαβάζουμε τις απαντήσεις της ερώτησης που επέλεξε ο χρήστης. Αυτές οι απαντήσεις είναι αποθηκευμένες στη συσκευή από τη στιγμή που ο χρήστης επιλέξει μια ερώτηση από ένα τεστ. Έπειτα προσθέτουμε click listeners στα κουμπιά που περιέχουν τις απαντήσεις σε μια ερώτηση και θέτουμε το κείμενο των κουμπιών αυτών με το κείμενο των απαντήσεων. Ο κάθε click listener αποθηκεύει τη απάντηση του για την ερώτηση που επέλεξε να απαντήσει.

FinalQuizActivity.kt:

Ο παρακάτω κώδικας έχει τις ιδίες ακριβώς λειτουργίες με τον κώδικα στο αρχείο QuizActivity.kt.

```
12  class FinalQuizActivity : AppCompatActivity() {
13
14      override fun onCreate(savedInstanceState: Bundle?) {
15          super.onCreate(savedInstanceState)
16          setContentView(R.layout.activity_final_quiz)
17
18          // get quiz's id
19          val sharedPref: SharedPreferences = this.getSharedPreferences("PyGuruStudent", Context.MODE_PRIVATE)
20          val quiz = sharedPref.getInt("PyGuruQuiz", -1)
21
22          // get quiz's questions
23          val dbHelper = PyGuruHelper(this)
24          val questions = dbHelper.getQuestions(quiz) // 14 questions!
25
26          // display questions
27          val questionsBtns : List<Button> = listOf(
28              findViewById(R.id.fq1), // fqX -> final quiz's question X
29              findViewById(R.id.fq2),
30              findViewById(R.id.fq3),
31              findViewById(R.id.fq4),
32              findViewById(R.id.fq5),
33              findViewById(R.id.fq6),
34              findViewById(R.id.fq7),
35              findViewById(R.id.fq8),
36              findViewById(R.id.fq9),
37              findViewById(R.id.fq10),
38              findViewById(R.id.fq11),
39              findViewById(R.id.fq12),
40              findViewById(R.id.fq13),
41              findViewById(R.id.fq14)
42          )
43
44          questions.forEachIndexed { index, question ->
45              questionsBtns[index].text = question.get("body") // show question
46
47              // find answers -> ["answer-1,y", "answer-2,n", "answer-3,n"]
48              val answers = question.get("answers")!!.split(";")
49
50              // setup listeners that show possible answers
51              questionsBtns[index].setAllCaps(false)
52              questionsBtns[index].setOnClickListener {
53                  val quiz = sharedPref.getInt("PyGuruQuiz", -1)
54
55                  if (quiz > 0) {
```

```

101         // find correct answers
102         val correctAns = ans.filter { el ->
103             val content = el.split(",")
104             content[1].equals("y")
105         }
106
107         // calculate mark and update the old one
108         val mark = correctAns.size / 14.toDouble()
109         val username = sharedPref.getString("PyGuruUser", "")
110         val res = dbHelper.updateMark(username, mark, quiz)
111         if (!res) {
112             Toast.makeText(this, "Something went wrong! Try again.", Toast.LENGTH_LONG).show()
113         }
114
115         // make test invalid by removing the quiz's index
116         val editor : SharedPreferences.Editor = sharedPref.edit()
117         editor.putInt("PyGuruQuiz", -1)
118         editor.commit()
119
120         // remove stored answers
121         for (i in 1..14) {
122             editor.putString("PyGuruAnswer$i", "")
123             editor.commit()
124         }
125
126         // find which answers are wright and which are wrong
127         var ansMsgs = ans.mapIndexed { i, a ->
128             val content = a.split(",")
129             var res = "Answer-" + (i+1) +": "+ content[0] + " -> "
130             if (content[1].equals("y")) {
131                 res = res + "Correct"
132             } else {
133                 res = res + "Wrong"
134             }
135             res
136         }
137
138         ansMsgs = listOf("Your answers are:") + ansMsgs
139
140         // show user's mark and answers
141         // create the builder
142         val dialogBuilder = AlertDialog.Builder(this)
143
144         // prepare the items
145         val msgs = ansMsgs.plus("Your mark is: ${mark.toString().substring(0,4)}(${correctAns.size}/14)")

```

```

144         // prepare the items
145         val msgs = ansMsgs.plus("Your mark is: ${mark.toString().substring(0,4)}(${correctAns.size}/14)")
146         val items = msgs.toTypedArray() // convert string list to string array
147
148         // show the items
149         dialogBuilder.setItems(items, null)
150         dialogBuilder.setPositiveButton("OK", null)
151         dialogBuilder.show()
152
153         // create alert box
154         val alertDialog = dialogBuilder.create()
155         // set title
156         alertDialog.setTitle("Your answers and mark")
157         // show alert box
158         alertDialog.show()
159     } else {
160         Toast.makeText(this, "Either there are some missing questions or you have already completed the test.", Toast.LENGTH_SHORT).show()
161     }
162 }
163 }
164 }
```

NewQuestionActivity.kt:

```
11  class NewQuestionActivity : AppCompatActivity() {
12
13      override fun onCreate(savedInstanceState: Bundle?) {
14          super.onCreate(savedInstanceState)
15          setContentView(R.layout.activity_new_question)
16
17          val dbHelper = PyGuruHelper(this)
18          val sharedPref: SharedPreferences = this.getSharedPreferences("PyGuruStudent", Context.MODE_PRIVATE)
19          val editor : SharedPreferences.Editor = sharedPref.edit()
20
21          // setup right and wrong answers
22          for (i in 1..3) {
23              editor.putString("PyGuruRightAnswer$i", "n") // every answer is wrong
24              editor.commit()
25          }
26
27          val quizBtns : List<Button> = listOf(
28              findViewById(R.id.quiz_1_btn),
29              findViewById(R.id.quiz_2_btn),
30              findViewById(R.id.quiz_3_btn),
31              findViewById(R.id.quiz_4_btn),
32              findViewById(R.id.quiz_5_btn),
33              findViewById(R.id.quiz_6_btn),
34              findViewById(R.id.quiz_7_btn)
35          )
36
37          val ansBtns : List<Button> = listOf(
38              findViewById(R.id.ans_1_btn),
39              findViewById(R.id.ans_2_btn),
40              findViewById(R.id.ans_3_btn)
41          )
42
43          // setup listeners
44          quizBtns.forEachIndexed { i, btn ->
45              btn.setOnClickListener {
46                  editor.putInt("PyGuruAtQuiz", i+1)
47                  editor.commit()
48              }
49          }
50
51          ansBtns.forEachIndexed { i, btn ->
52              btn.setOnClickListener {
53                  editor.putString("PyGuruRightAnswer${i + 1}", "y")
54                  editor.commit()
55              }
56          }
57      }
58  }
```

Στις γραμμές 17-41 βρίσκουμε τα κουμπιά με τα οποία ο εξεταστής επιλεγεί σε ποιο τεστ θα προσθέσει μια νέα ερώτηση, τα κουμπιά με τα οποία ο εξεταστής επιλεγεί ποια απάντηση είναι σωστή και αποθηκεύουμε την εγκυρότητα των απαντήσεων.

Στις γραμμές 43-56 προσθέτουμε στα κουμπιά με τα οποία ο εξεταστής επιλεγεί ποια απάντηση είναι σωστή και στα κουμπιά με τα οποία ο εξεταστής επιλεγεί σε ποιο τεστ θα προσθεσουμε μια νέα ερώτηση click listeners. Οι click

listeners στα κουμπιά του quizBtns array αποθηκεύουν το id του τεστ στο οποίο θέλει ο εξεταστής θα προσθέσει μια νέα ερώτηση ενώ οι click listeners στα κουμπιά του ansBtns array αποθηκεύουν ποια απάντηση είναι η σωστή.

```
59         val addBtn : Button = findViewById(R.id.add_question_btn)
60         addBtn.setOnClickListener {
61             val questionBody = findViewById<TextView>(R.id.question_body).text.toString()
62             val ansTxts : List<String> = listOf(
63                 findViewById<TextView>(R.id.ans1_txt).text.toString(),
64                 findViewById<TextView>(R.id.ans2_txt).text.toString(),
65                 findViewById<TextView>(R.id.ans3_txt).text.toString()
66             )
67
68             val quizNo = sharedPref.getInt("PyGuruAtQuiz", -1)
69
70             // get right and wrong answers
71             val rightAnswers : List<String> = listOf(
72                 sharedPref.getString("PyGuruRightAnswer1", ""),
73                 sharedPref.getString("PyGuruRightAnswer2", ""),
74                 sharedPref.getString("PyGuruRightAnswer3", "")
75             )
76
77             // are all answer txt filled?
78             val b1 = ansTxts.all { txt -> txt.isNotEmpty() }
79
80             // is there any right answer?
81             val b2 = rightAnswers.any { a -> a.equals("y") }
82
83             // check if everything is filled
84             if (questionBody.isNotEmpty() && b1 && quizNo != -1 && b2) {
85                 // get answers
86                 val realAns : List<String> = ansTxts.mapIndexed { i, ans ->
87                     ans + "," + sharedPref.getString("PyGuruRightAnswer${i + 1}", "")
88                 }
89
90                 // create answers text
91                 var realAnsTxt = ""
92                 realAns.forEachIndexed { i, a ->
93                     if (i < 2) {
94                         realAnsTxt = realAnsTxt + a + ";"
95                     } else {
96                         realAnsTxt = realAnsTxt + a
97                     }
98                 }
99                 Toast.makeText(this, "$realAnsTxt", Toast.LENGTH_LONG).show()
100
101             // add question
102             val res = dbHelper.insertQuestion(questionBody, quizNo, realAnsTxt)
103
104             // check insertion's result
105             if (res) {
106                 Toast.makeText(this, "Your question has been added.", Toast.LENGTH_LONG).show()
107             } else {
108                 Toast.makeText(this, "Something gone wrong, please try again.", Toast.LENGTH_LONG).show()
109             }
110         } else {
111             Toast.makeText(this, "You must fill the form.", Toast.LENGTH_LONG).show()
112         }
113     }
114 }
115 }
```

Στις γραμμές 59-113 που προσθέτουμε click listener στο κουμπί που προσθέτει στη βάση δεδομένων την νέα ερώτηση του εξεταστή. Ο click listener ελέγχει αν όλα τα στοιχεία της φόρμας είναι συμπληρωμένα. Αυτό επιτυγχάνεται όταν ελέγχει αν έχουν αποθηκευτεί στη συσκευή απαντήσεις σωστές και λάθος, αν έχουν αποθηκευτεί απαντήσεις και αν έχει αποθηκευτεί το id ενός τεστ. Αν υπάρχουν απαντήσεις, id τεστ και σωστές και λάθος απαντήσεις τότε προσθέτουμε στη βάση δεδομένων τη νέα ερώτηση. Σε περίπτωση σφάλματος κατά τη πρόσθεση της ερώτησης δείχνουμε το κατάλληλο μήνυμα στην οθόνη του χρήστη. Αν κάποιο από τα πεδία της φόρμας δεν έχει συμπληρωθεί ή αν δεν έχουν αποθηκευτεί τα απαραίτητα στοιχεία για την πρόσθεση της ερώτησης ενημερώνουμε τον χρήστη.

HelperActivity.kt:

```
1 package gr.example.zografos.vasileios.pythonguru
2
3 import android.support.v7.app.AppCompatActivity
4 import android.os.Bundle
5
6 class HelpActivity : AppCompatActivity() {
7
8     override fun onCreate(savedInstanceState: Bundle?) {
9         super.onCreate(savedInstanceState)
10        setContentView(R.layout.activity_help)
11    }
12 }
13
```

Σε αυτό το αρχείο φορτώνουμε τη σελίδα που περιέχει συχνές ερωτήσεις σχετικές με το περιεχόμενο της εφαρμογής.