Mobile Application Development
Project Report

CodeQuiz

Ayush Soni
Section A
140905076

Introduction

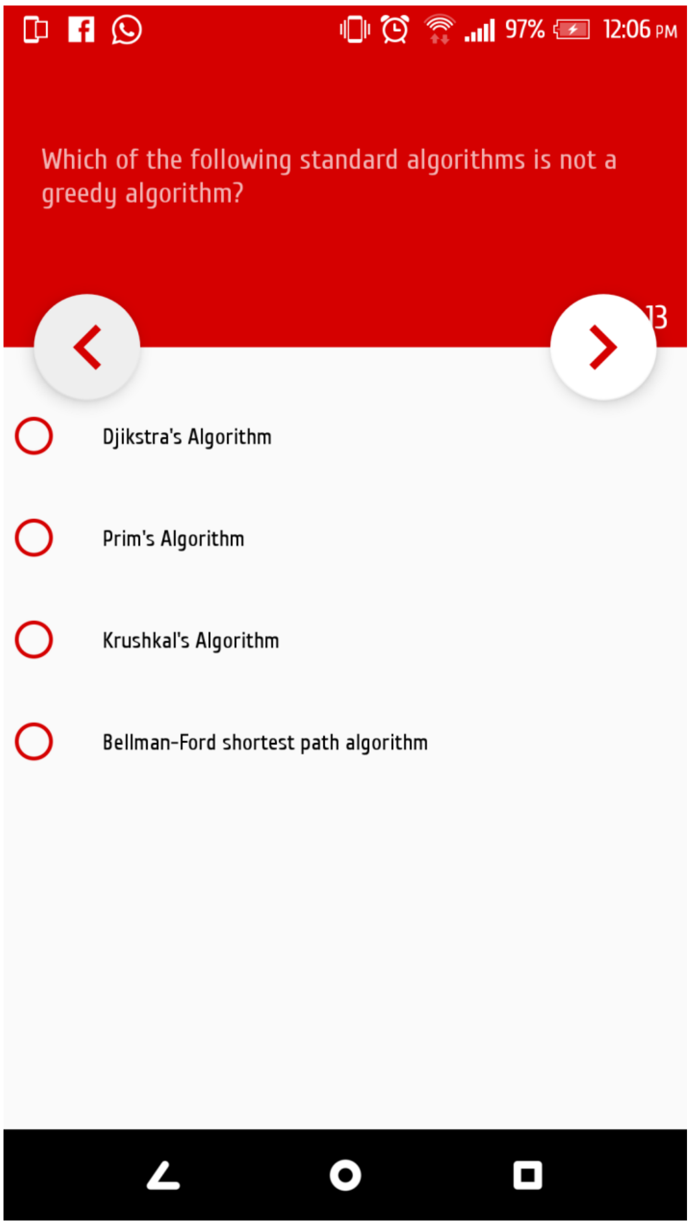
Code Quiz provides a convenient way for passionate coders to practice their coding skills, and gain knowledge on different aspects of coding. Users can take a quiz on any coding topic such as Algorithms, Data Structures, Input/Output, and assess themselves on the same. Comparing scores with friends will tell them which topics to focus on.

Code Quiz contains quizzes on different topics, each quiz containing more than one multiple choice question. The user will be displayed his score at the end of the test. The question base is extracted from sites like GeeksForGeeks, and other such computer science portals which are dedicated to coding enthusiasts. The need for such an application becomes apparent when preparing for placement interviews. The preliminary tests of core companies such as Microsoft IT and IDC, Amazon and Directi are mostly based on Input/Output, Time Complexity etc. Any software giant highly focuses on a candidate\'s coding skills, and tests them thoroughly, asking them about different data structures and algorithms, which is what Code Quiz focuses on.

The application has been shared on **GitHub** and hence, is open source. Git has been used throughout the development phase as a medium for version control.

https://github.com/Ayush-Soni/CodeQuiz.git

Design


Which of the following standard algorithms is not a greedy algorithm?

13

Djikstra's Algorithm

Prim's Algorithm

Krushkal's Algorithm

Bellman-Ford shortest path algorithm

- CodeQuiz uses Google's material design at its heart. A visual language that synthesizes the classic principles of good design with the innovation and possibility of technology and science.
- CodeQuiz uses animations at activity transitions to provide a visually stunning experience to the users.

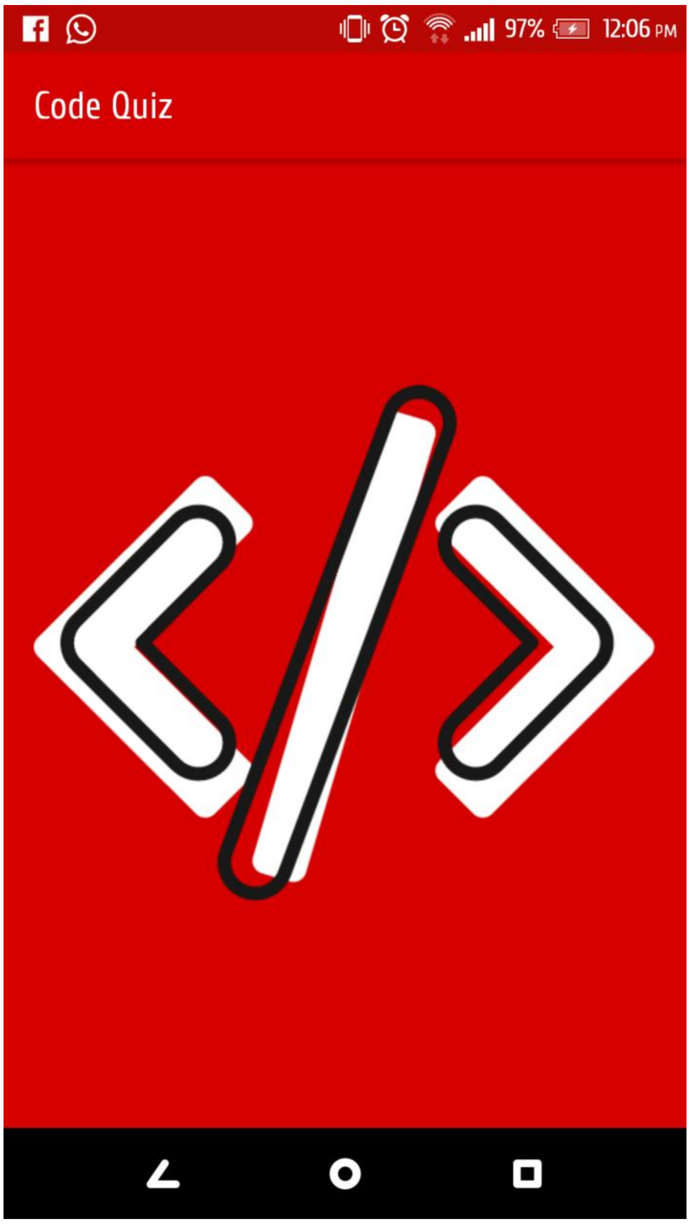
Activities

- Splash Screen
- App Description
- Login
- Quiz List
- Quiz (*Fragment*)
- Display Result

Important java classes

- Database Helper
- User Details
- Quiz Details
- Quiz Question
- Answer Details
- Question Answer

Splash Screen Activity



The splash screen activity welcomes the users to the CodeQuiz by displaying the logo. Splash screens typically serve to enhance the look and feel of any application or web site, hence they are often visually appealing. On the backend, the activity runs a thread using Handler, which makes an intent to start the next activity after a specified time.

Important code

SplashScreenActivity.java

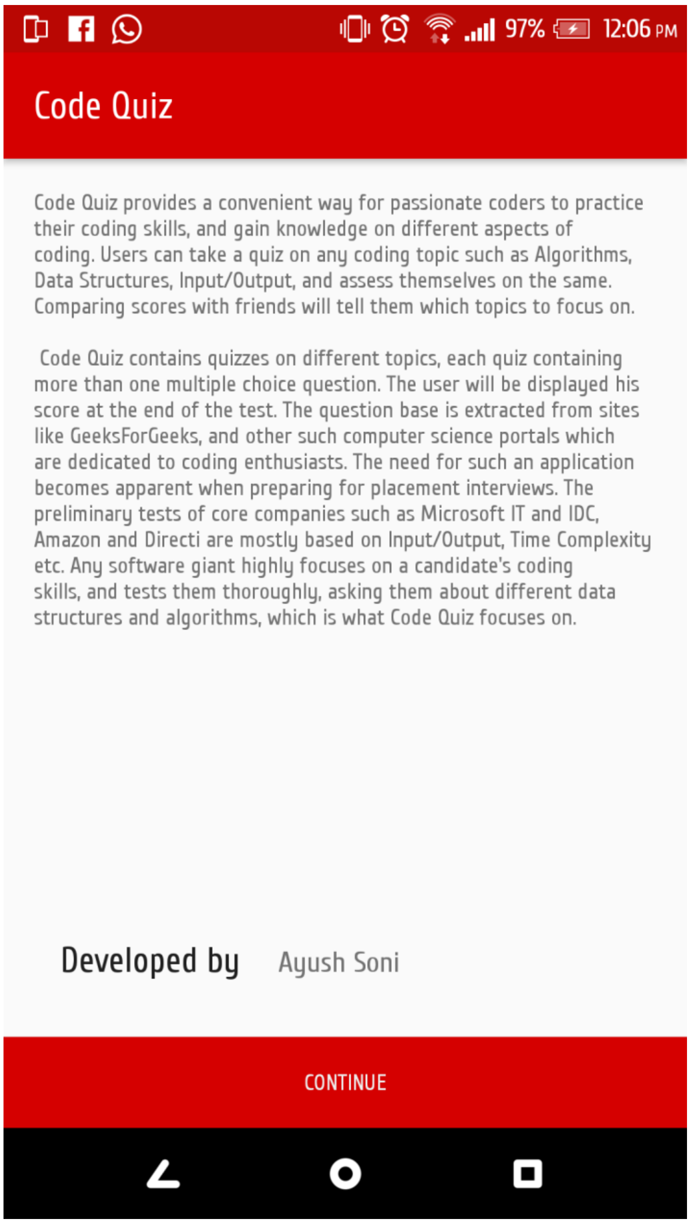
```
new Handler().postDelayed(new Runnable(){
            @Override
            public void run() {
               Intent mainIntent = new Intent(SplashScreenActivity.this,
ApplicationDescriptionActivity.class);
               SplashScreenActivity.this.startActivity(mainIntent);
               overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
               SplashScreenActivity.this.finish();
            }
         }, SPLASH_DISPLAY_LENGTH);
```

XML

activity_splash_screen.xml

```
<RelativeLayout ....>
 <ImageView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      app:srcCompat="@drawable/logomkra"
      android:layout_centerVertical="true"
      android:layout_centerHorizontal="true"
      android:id="@+id/imageView2" />
</RelativeLayout>
```

App Description Activity


12:06 PM

Code Quiz

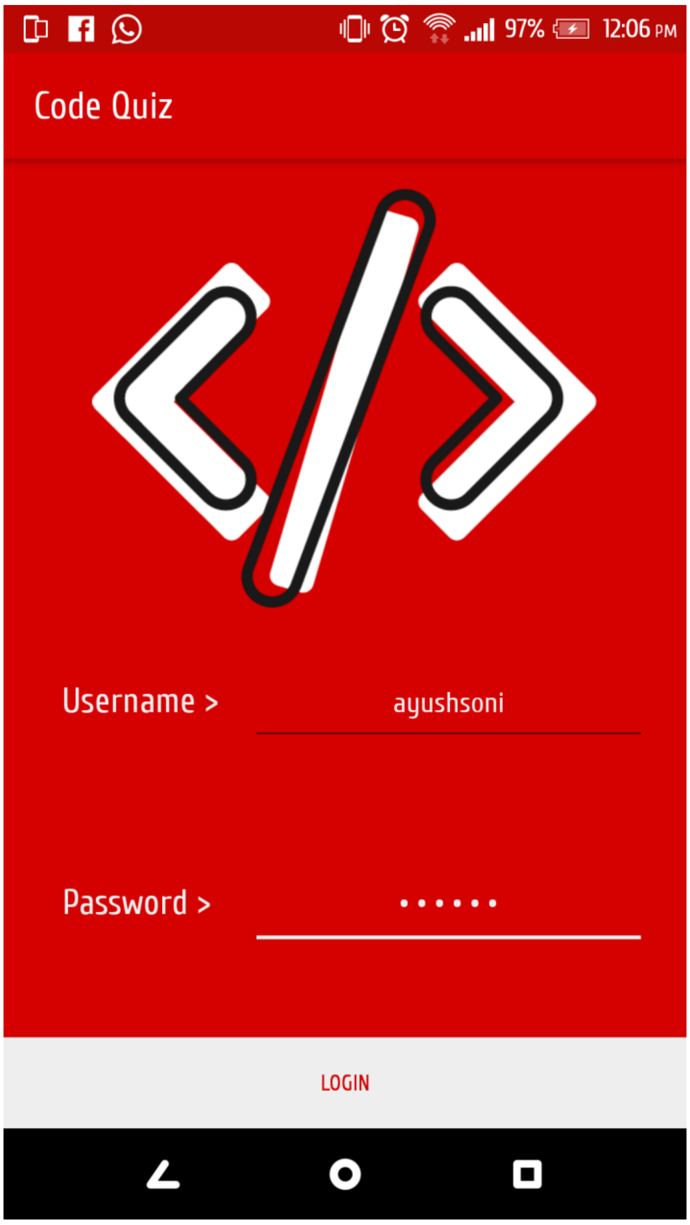
Code Quiz provides a convenient way for passionate coders to practice their coding skills, and gain knowledge on different aspects of coding. Users can take a quiz on any coding topic such as Algorithms, Data Structures, Input/Output, and assess themselves on the same. Comparing scores with friends will tell them which topics to focus on.

Code Quiz contains quizzes on different topics, each quiz containing more than one multiple choice question. The user will be displayed his score at the end of the test. The question base is extracted from sites like GeeksForGeeks, and other such computer science portals which are dedicated to coding enthusiasts. The need for such an application becomes apparent when preparing for placement interviews. The preliminary tests of core companies such as Microsoft IT and IDC, Amazon and Directi are mostly based on Input/Output, Time Complexity etc. Any software giant highly focuses on a candidate's coding skills, and tests them thoroughly, asking them about different data structures and algorithms, which is what Code Quiz focuses on.

Developed by Ayush Soni

The activity tells the users about the purpose of the application, and what the application aims to deliver to the users. It can be considered as a summary of functionalities that CodeQuiz provides to the users. Along with the description, this activity includes the developer's name, and an option to continue to the login screen of the application.

Login Activity



Username > ayushsoni

Password > ••••••

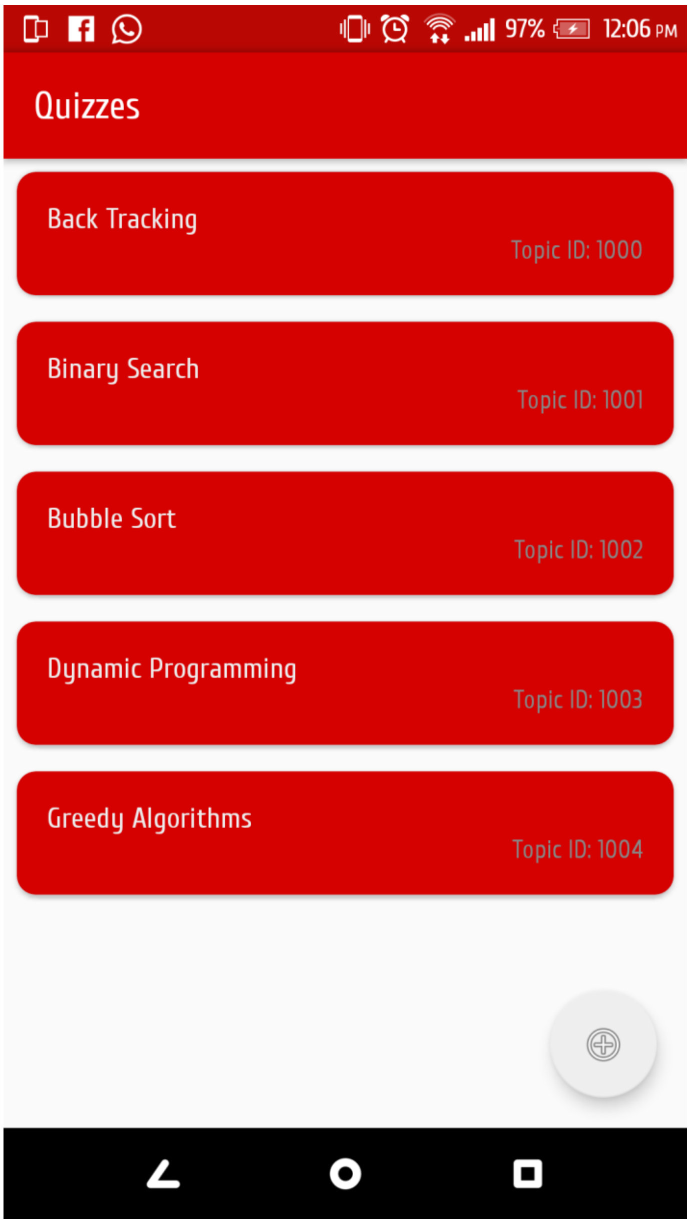
The login activity allows an authentication mechanism for users. The activity takes user details, and authenticates using SQL Database stored in the application. If the user exists, he is taken to the next activity, otherwise, the user is shown a toast informing him that the entered details are invalid. The functions used in this activity for SQL operations are defined in the DatabaseHelper class.

Important code

LoginActivity.java

```
loginButton.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {
             EditText editTextUsername = (EditText) findViewById(R.id.editTextUsername);
             EditText editTextPassword = (EditText) findViewById(R.id.editTextPassword);
             String specifiedUsername = editTextUsername.getText().toString();
             if((editTextPassword.getText().toString()).equals((new
DatabaseHelper(getApplicationContext())).getUserDetails(specifiedUsername).getPassword
())) {
                Intent i = new Intent(LoginActivity.this, QuizListActivity.class);
                startActivity(i);
                overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
                finish();
             }
             else {
                Toast.makeText(getApplicationContext(), "Invalid credentials. Try Again.",
Toast.LENGTH_SHORT).show();
             }
          }
      });
```

Quiz List Activity


12:06 PM

97%

Quizzes

Back Tracking

Topic ID: 1000

Binary Search

Topic ID: 1001

Bubble Sort

Topic ID: 1002

Dynamic Programming

Topic ID: 1003

Greedy Algorithms

Topic ID: 1004

The quiz list activity displays the various quizzes available to the users of the application. Each item, is a card view. The card view is enclosed in a scroll view. This activity uses the DatabaseHelper class, to obtain the list of all the quiz topics from the table storing information about the quizzes. The data is obtained in an ArrayList<QuizDetails> where each QuizDetail object has all the information about the quiz, such as name and ID. Recycler views have been used for each item. The code shows important parts of this activity, and the class extending fragments.

Important code

QuizListActivity.java

```
public void onListFragmentInteraction(QuizDetails item) {
        Log.d("Quiz list","Selected item: " + item);
        Intent intent = new Intent(QuizListActivity.this, Quiz.class);
        intent.putExtra("quizID",item.getQuizId());
        startActivity(intent);
        overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
    }
```

QuizListActivityFragment.java

```
public class QuizListActivityFragment extends Fragment {
    public QuizListActivityFragment() {
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_quiz_list, container, false);
    }
}
```

XML

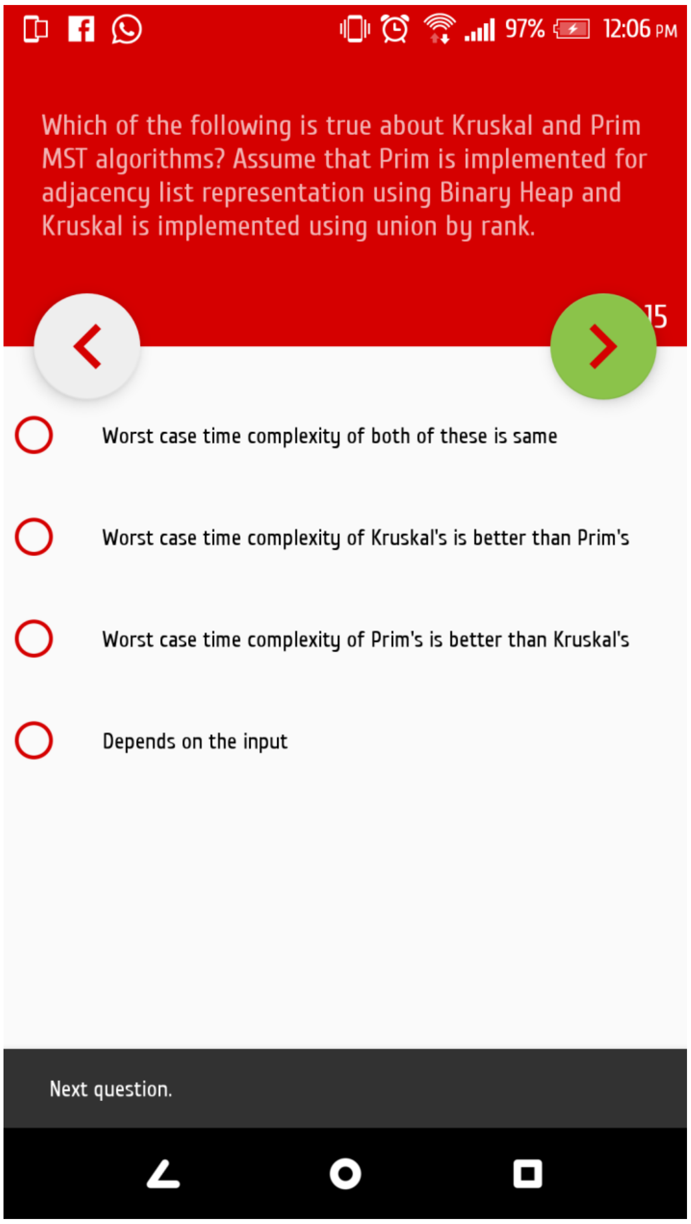
fragment_quiz_list.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/content_quiz_list"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="@dimen/activity_vertical_margin"
   android:paddingLeft="@dimen/activity_horizontal_margin"
   android:paddingRight="@dimen/activity_horizontal_margin"
   android:paddingTop="@dimen/activity_vertical_margin"
   android:textAlignment="center"
   tools:context="com.area51.ayush.codequiz.QuizListActivityFragment"
   tools:showIn="@layout/activity_quiz_list">


</RelativeLayout>
```

fragment_item_list.xml

```
<android.support.v7.widget.RecyclerView
xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/list"
   android:name="com.area51.ayush.codequiz.ItemFragment"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:layout_marginLeft="16dp"
   android:layout_marginRight="16dp"
   app:layoutManager="LinearLayoutManager"
   tools:context="com.area51.ayush.codequiz.ItemFragment"
   tools:listitem="@layout/fragment_item" />
```

Quiz Activity


12:06 PM

Which of the following is true about Kruskal and Prim MST algorithms? Assume that Prim is implemented for adjacency list representation using Binary Heap and Kruskal is implemented using union by rank.

15

- Worst case time complexity of both of these is same
- Worst case time complexity of Kruskal's is better than Prim's
- Worst case time complexity of Prim's is better than Kruskal's
- Depends on the input

Next question.

The quiz activity is the activity which shows the users, the different questions in the quiz. The floating action buttons can be used to navigate through different questions. SnackBars are used to tell user that he is looking at the next question depending upon when he chooses to press the floating action button for next question. On the last question, the button turns green, indicating that pressing it would evaluate him for the currently chosen answers, and give a notification with this result. Notifications are used so as to not disturb the user while he is taking the quiz, so if he presses the green button by mistake, he is not taken to the display result activity. The activity heavily relies on DatabaseHelper class to provide various functions, from getting questions for quizzes to getting answers for questions, and evaluating the user.

Important code

Quiz.java

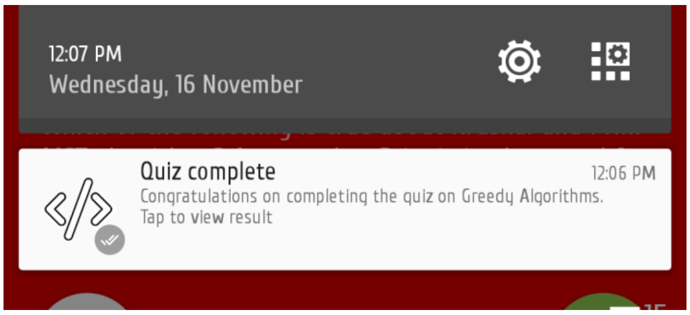
```
public class Quiz extends AppCompatActivity {
    RadioGroup radioGroup;
    int currentQuestionID;
    int currentQuestionNo = 0;
    TextView textView;
    int quizId;
    Intent intent;
    DatabaseHelper db;
    String quizName;
    CollapsingToolbarLayout collapsingToolbarLayout;
    Toolbar toolbar;
    ArrayList<QuizQuestion> allQuestionsForQuiz;
    FloatingActionButton fab;
    FloatingActionButton fab1;
    int[] answerSelected;
    boolean[] score;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_quiz);
        toolbar = (Toolbar) findViewById(R.id.toolbar);
    }

    @Override
    protected void onResume() {
        super.onResume();
        radioGroup = (RadioGroup) findViewById(R.id.optionsRadioGroup);
        collapsingToolbarLayout = (CollapsingToolbarLayout)
findViewById((R.id.toolbar_layout));
        setSupportActionBar(toolbar);
        db = new DatabaseHelper(getApplicationContext());
        intent = getIntent();
        textView = (TextView) findViewById(R.id.question_text_view);
        quizId = intent.getIntExtra("quizID",0);
        quizName = db.getQuizName(quizId);
        allQuestionsForQuiz = db.getAllQuestionsOfQuiz(quizId);
        answerSelected = new int[allQuestionsForQuiz.size()];
        score = new boolean[allQuestionsForQuiz.size()];
        for(int i=0; i<allQuestionsForQuiz.size(); i++) {
            answerSelected[i] = -1;
            score[i] = false;
        }

        Log.i("","***************************Question:
"+allQuestionsForQuiz.get(currentQuestionNo).getQuestion());
        Log.i("","***************************Current question id: "+ currentQuestionID);

        fab = (FloatingActionButton) findViewById(R.id.fabEnd);
        fab1 = (FloatingActionButton) findViewById(R.id.fabStart);
        if(allQuestionsForQuiz.isEmpty()) {
            Toast.makeText(getApplicationContext(), "No questions for quizId: "+ quizId,
Toast.LENGTH_SHORT).show();
        }
        else {
            if (currentQuestionNo < allQuestionsForQuiz.size()) {
                updateQuestion(currentQuestionNo);
            }
        }

        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                int checkedRadioButtonId = radioGroup.getCheckedRadioButtonId();
                if (checkedRadioButtonId == -1) {
                    //Do nothing
                }
                else{
                    answerSelected[currentQuestionNo]=checkedRadioButtonId;
                    RadioButton radioButton = (RadioButton) findViewById(checkedRadioButtonId);
                    String answerString = (String)radioButton.getText();
                    if(answerString.equals(db.getAnswerForQuestion(currentQuestionID))){
                        score[currentQuestionNo] = true;
                    }
                    else {
                        score[currentQuestionNo] = false;
                    }
                }
                currentQuestionNo++;
                if(currentQuestionNo<allQuestionsForQuiz.size()) {
                    Snackbar.make(view, "Next question.", Snackbar.LENGTH_LONG)
                            .setAction("Action", null).show();
                    updateQuestion(currentQuestionNo);
                }
                else {
                    int totalScore = 0;
                    for(int i=0; i<score.length;i++) {
                        if(score[i])
                            totalScore++;
                    }
                    //Snackbar.make(view, "Total: "+totalScore+"/"+allQuestionsForQuiz.size(),
Snackbar.LENGTH_LONG)
                    //        .setAction("Action", null).show();
                    issueNotification(totalScore, allQuestionsForQuiz.size());
                    currentQuestionNo--;
                }
            }
        });

        fab1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                currentQuestionNo--;
                if(currentQuestionNo>=0) {
                    Snackbar.make(view, "Previous question.", Snackbar.LENGTH_LONG)
                            .setAction("Action", null).show();
                    updateQuestion(currentQuestionNo);
                }
                else {
                    Snackbar.make(view, "This is the first question.", Snackbar.LENGTH_LONG)
                            .setAction("Action", null).show();
                    currentQuestionNo++;
                }
            }
        });
    }

    void updateQuestion(int currentQuestionNo) {
        currentQuestionID = allQuestionsForQuiz.get(currentQuestionNo).getQuestionId();
        ArrayList<AnswerDetails> allAnswersForQuestion =
db.getAnswersForQuestion(currentQuestionID);
        collapsingToolbarLayout.setTitle(Integer.toString(currentQuestionID));
        textView.setText(allQuestionsForQuiz.get(currentQuestionNo).getQuestion());
        Log.i("", "***************************UPDATE::allAnswersForQuestion.size(): " +
allAnswersForQuestion.size());
        Log.i("", "***************************UPDATE::Question: " +
allQuestionsForQuiz.get(currentQuestionNo).getQuestion());
        Log.i("", "***************************UPDATE::Current question id: " +
currentQuestionID);
        if (allAnswersForQuestion.size()==0) {
            for (int i = 0; i < radioGroup.getChildCount(); i++) {
                ((RadioButton) radioGroup.getChildAt(i)).setText(new String("No options"));
            }
        } else {
            for (int i = 0; i < radioGroup.getChildCount(); i++) {
                ((RadioButton)
radioGroup.getChildAt(i)).setText(allAnswersForQuestion.get(i).getAnswer());
            }
        }

        if(currentQuestionNo==allQuestionsForQuiz.size()-1) {
        fab.setBackgroundTintList(ColorStateList.valueOf(Color.parseColor("#8BC34A")));
        }
        if(currentQuestionNo<allQuestionsForQuiz.size()-1) {
            fab.setBackgroundTintList(ColorStateList.valueOf(Color.parseColor("#FFFFFF")));
        }
        if(answerSelected[currentQuestionNo]!=-1) {
            RadioButton rbutton = (RadioButton)
findViewById(answerSelected[currentQuestionNo]);
            rbutton.setChecked(true);
        }
    }

    public void issueNotification(int totalScore, int totalQuestions) {
        NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this)
                .setSmallIcon(R.drawable.success)
                .setLargeIcon(BitmapFactory.decodeResource(getResources(),
R.drawable.logomkra))
                .setContentTitle("Quiz complete")
                .setContentText("Congratulations on completing the quiz on " + quizName +".")
                .setSubText("Tap to view result");

        Intent intent = new Intent(getApplicationContext(), DisplayResultActivity.class);
        intent.putExtra("quizname", quizName);
        intent.putExtra("totalmarks", totalScore);
        intent.putExtra("totalquestions", totalQuestions);
        intent.putExtra("notificationid", 1);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent,
FLAG_UPDATE_CURRENT);
        mBuilder.setContentIntent(pendingIntent);
        NotificationManager notificationManager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
        notificationManager.notify(1, mBuilder.build());
    }
}
```

XML

content_quiz.xml

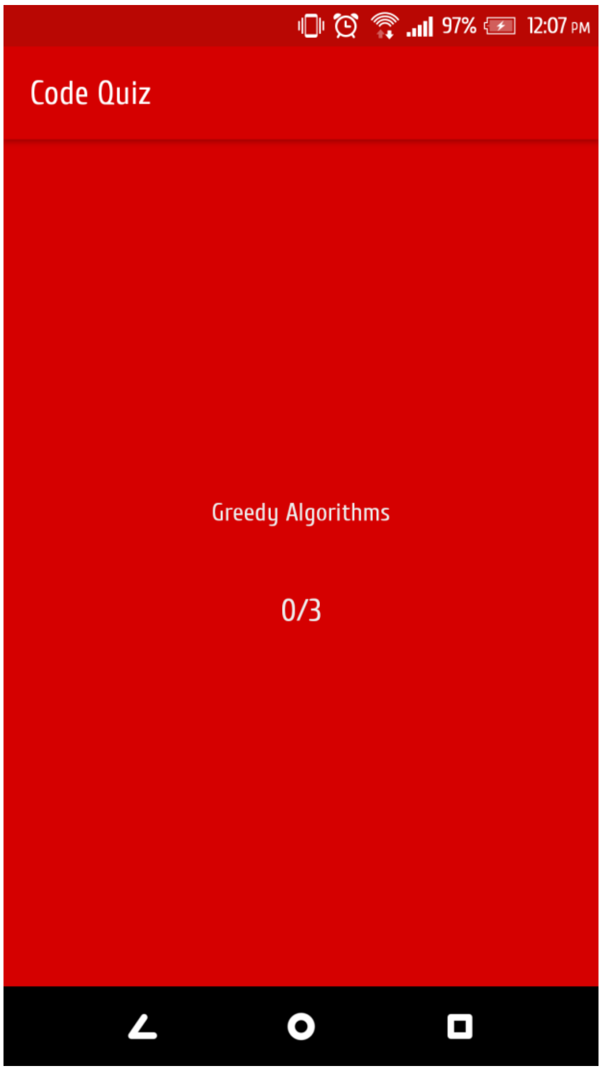
```
<android.support.v4.widget.NestedScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginTop="20dp"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="com.area51.ayush.codequiz.Quiz"
    tools:showIn="@layout/activity_quiz">
    <RadioGroup
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/optionsRadioGroup"
        android:orientation="vertical">
        <RadioButton
            android:layout_width="match_parent"
            android:buttonTint="@color/colorPrimary"
            android:padding="20dp"
            android:text="@string/coming_soon"
            android:layout_height="wrap_content" />
        <RadioButton
            android:layout_width="match_parent"
            android:text="@string/coming_soon"
            android:padding="20dp"
            android:buttonTint="@color/colorPrimary"
            android:layout_height="wrap_content" />
        <RadioButton
            android:layout_width="match_parent"
            android:text="@string/coming_soon"
            android:padding="20dp"
            android:buttonTint="@color/colorPrimary"
            android:layout_height="wrap_content" />
        <RadioButton
            android:layout_width="match_parent"
            android:text="@string/coming_soon"
            android:padding="20dp"
            android:buttonTint="@color/colorPrimary"
            android:layout_height="wrap_content" />
    </RadioGroup>
</android.support.v4.widget.NestedScrollView>
```

Notification and Display Result Activity



12:07 PM
Wednesday, 16 November

Quiz complete 12:06 PM
Congratulations on completing the quiz on Greedy Algorithms.
Tap to view result



Code Quiz

Greedy Algorithms

0/3

Once the user presses the floating action button on the last question, a notification pops up, as shown above. The notification, when tapped, launches the DisplayResultActivity. This activity shows user the name of the quiz (to make sure it's displaying the result of the same quiz he attempted), his score, and the total number of questions in the quiz.

Important code

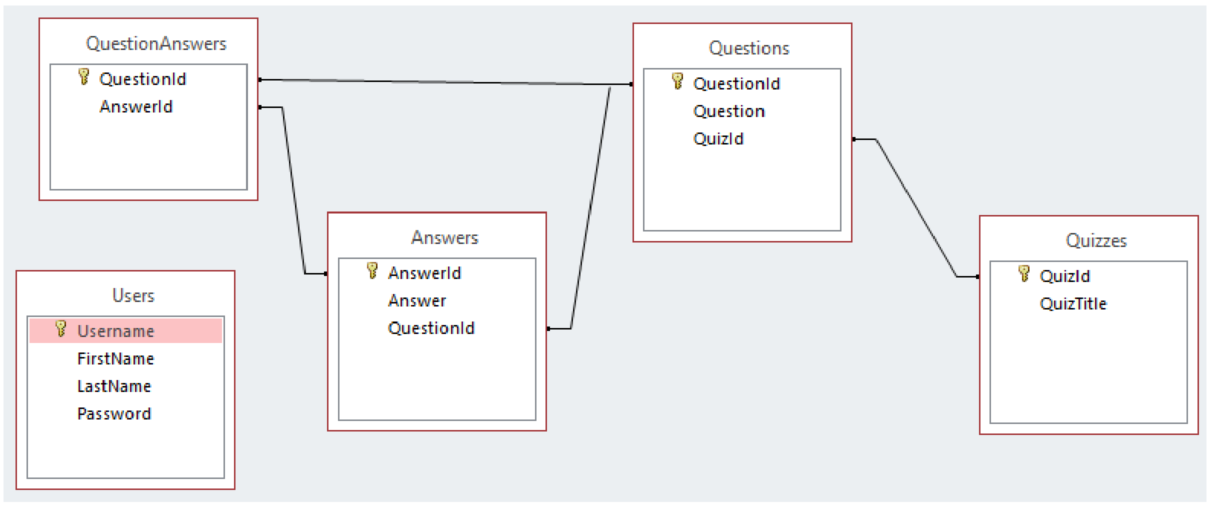
DisplayResultActivity.java

```
public class DisplayResultActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_display_result);

        Intent intent = getIntent();
        String quizName = intent.getStringExtra("quizname");
        int totalMarks = intent.getIntExtra("totalmarks", 0);
        int totalQuestions = intent.getIntExtra("totalquestions", 0);
        int notificationId = intent.getIntExtra("notificationid", 0);
        TextView textView = (TextView) findViewById(R.id.result_quiz_textview);
        TextView textView1 = (TextView) findViewById(R.id.result_display_textview);
        textView.setText(quizName);
        String marks = totalMarks + "/" +  totalQuestions;
        textView1.setText(marks);
        NotificationManager notificationManager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
        notificationManager.cancel(notificationId);
    }

    @Override
    protected void onResume() {
        super.onResume();
    }
}
```

Important java classes

Database Helper

The database helper class, as the name suggests, provides the application with various functions that are needed for communication with the database. The database schema can be summarized by the following diagram.



The class provides enough functions to the application so that the application can smoothly store and retrieve data from various tables shown above. The main functions of this class are:

- Creation of database
- Storing data to the database
- Fetching data from the database
- Updating data in the database

User Details

The class represents a row in the 'Users' table. The main features of this class are:

- Store user details
- Provide getter and setter methods for the user details

Quiz Details

The class represents a row in the ‘Quizzes’ table. The main features of this class are:

- Store quiz details
- Provide getter and setter methods for the quiz details

Quiz Question

The class represents a row in the ‘Questions’ table. The main features of this class are:

- Store question details
- Provide getter and setter methods for the question details

Answer Details

The class represents a row in the ‘Answer’ table. The main features of this class are:

- Store answer details
- Provide getter and setter methods for the answer details

Question Answer

The class represents a row in the ‘Question Answer’ table. It maps the answers to the correct question ID. The main features of this class are:

- Store questionID and AnswerID
- Provide getter and setter methods for the questionID and answerID

Future

- Include ability to create user profiles
- Save the user’s recently taken quizzes and results
- Include hints for questions
- Include timed quizzes
- Include ability to see which questions were answered incorrectly
- Include ability to take competitive quizzes (user vs user)

References

stackoverflow.com

developer.android.com

udacity.com