Mobile App Development
Midterm Make-Up Exam

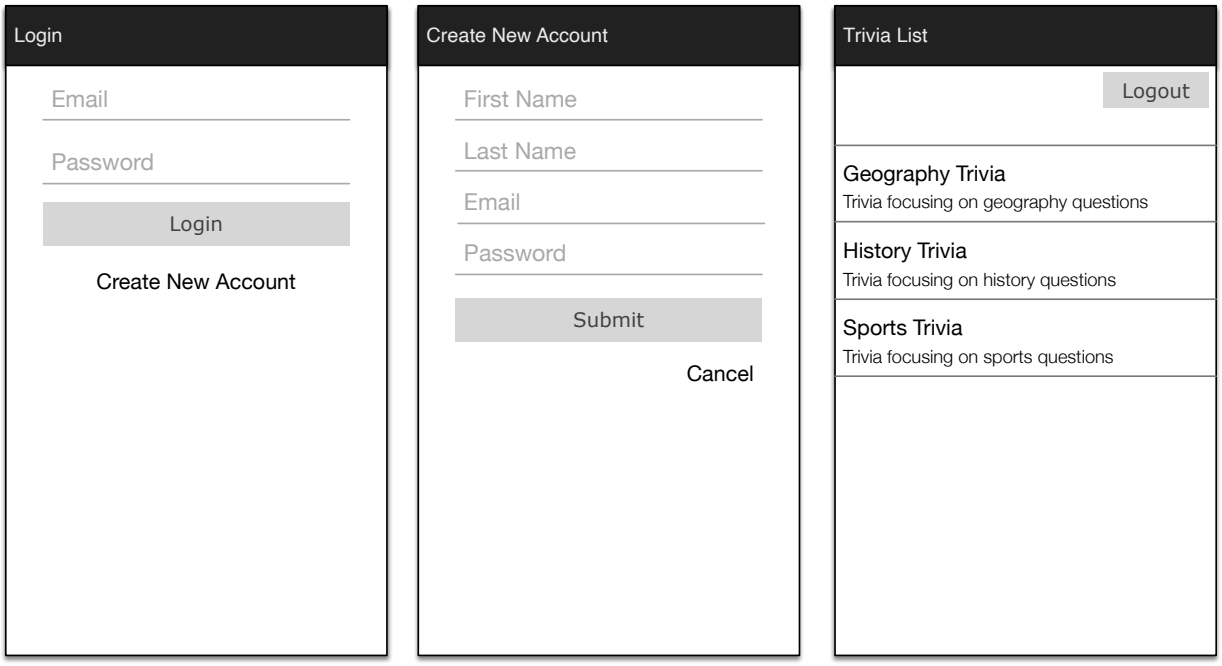
Basic Instructions:

1. This is the Midterm Make-Up Exam, which will count for 20% of the total course grade. The maximum you can score on this assignment is 80 out of 100.
2. This Midterm is an individual effort. Each student is responsible for her/his own Midterm and its submission.
3. Once you have picked up the exam, you may not discuss it in any way with anyone until the exam period is over.
4. During the exam, you are allowed to use the course videos, slides, and your code from previous home works and in class assignments. You can use the internet to search for answers. You are NOT allowed to use code provided by other students or solicit help from other online persons.
5. Answer all the exam parts, all the parts are required.
6. Please download the support files provided with the Midterm and use them when implementing your project.
7. Your assignment will be graded for functional requirements and efficiency of your submitted solution. You will loose points if your code is not efficient, does unnecessary processing or blocks the UI thread.
8. Create a zip file which includes all the project folder, any required libraries, and your presentation material. Submit the exported file using the provided canvas submission link.
9. **Do not try to use any Social Messenger apps, Emails, Or Cloud File Storage services in this exam.**
10. **Failure to follow the above instructions will result in point deductions.**
11. **Any violation of the rules regarding consultation with others will not be tolerated and will result disciplinary action and failing the course.**

Midterm Make-Up Exam (100 Points)

In this assignment you will develop a Trivia application. You are provided with a postman file that contains all the required APIs for this application.

1. You are required to use OkHttp library in this application in order to make all the http connections and API calls. All the data returned by the APIs is in JSON format.
2. All the network calls and parsing should be done in a background thread.
3. All UI changes, updates and edits should be performed on the main thread only.
4. In this app you will have only one Activity and 4 fragments, all communication between fragments should be managed by the activity.



(a) Login Fragment (b) Register Fragment (c) Trivia List Fragment

Figure 1, Application Wireframe

Part 1: Login Fragment (10 Points)

The interface should be created to match the UI presented in Figure 1(a). The requirements are as follows:

1. Upon entering the email and password:
   a. Clicking “Login” button, if all the inputs are not empty, you should attempt to login the user by using the **/api/login** API.
   b. If the login is successful, then communicate the returned and parsed authentication token (See Figure 2) to the activity and replace the current fragment with the Trivia List Fragment.
   c. If login is not successful, show an alert dialog showing the error message returned by the api.
   d. If there is missing input, show an alert dialog indicating missing input.
2. Clicking the “Create New Account” should replace this fragment with the Create New Account Fragment.

```
{
    "status": "ok",
    "token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE2MTc0MjgzMTUsImV4cCI6MTY0ODk
2NDMxNSwianRpIjoibmNPMzlNVWJYV2ZraVdsMWdtcEUxIiwidXNlciI6Mn0.43KgPvMaxOnkNC51
ClZ9nqW2oit5Rm_2jDOo1u6SzmA",
    "user_id": 2,
    "user_email": "a@a.com",
    "user_fname": "Alice",
    "user_lname": "Smith",
    "user_role": "USER"
}
```


Figure 2, Highlighting the Auth Token returned by login and signup

Part 2: Create New Account Fragment (10 Points)

The interface should be created to match the UI presented in Figure 1(b). The requirements are as follows:

1. This fragment should allow a user to create a new account. Upon entering the first name, last name, email and password, clicking the Submit button should:
   a. If all the inputs are not empty, you should attempt to signup the user by using the **/api/signup** API.
   b. If the registration is successful, then parse the returned response, and send the authentication token to the activity and replace the current fragment with the Trivia List Fragment.
   c. If the registration is not successful, show an alert dialog showing the error message returned by the api.
   d. If there is missing input, show an alert dialog indicating missing input.
2. Clicking “Cancel” should replace this fragment with the Login Fragment.

```
//token received /api/login or api/signup
String token = "eyJ0eXAiOiJKV1QiLCJhb......";
Request request = new Request.Builder()
        .url("https://www.theappsdr.com/api/trivias")
        .addHeader("Authorization", "BEARER "  + token)
        .build();
```

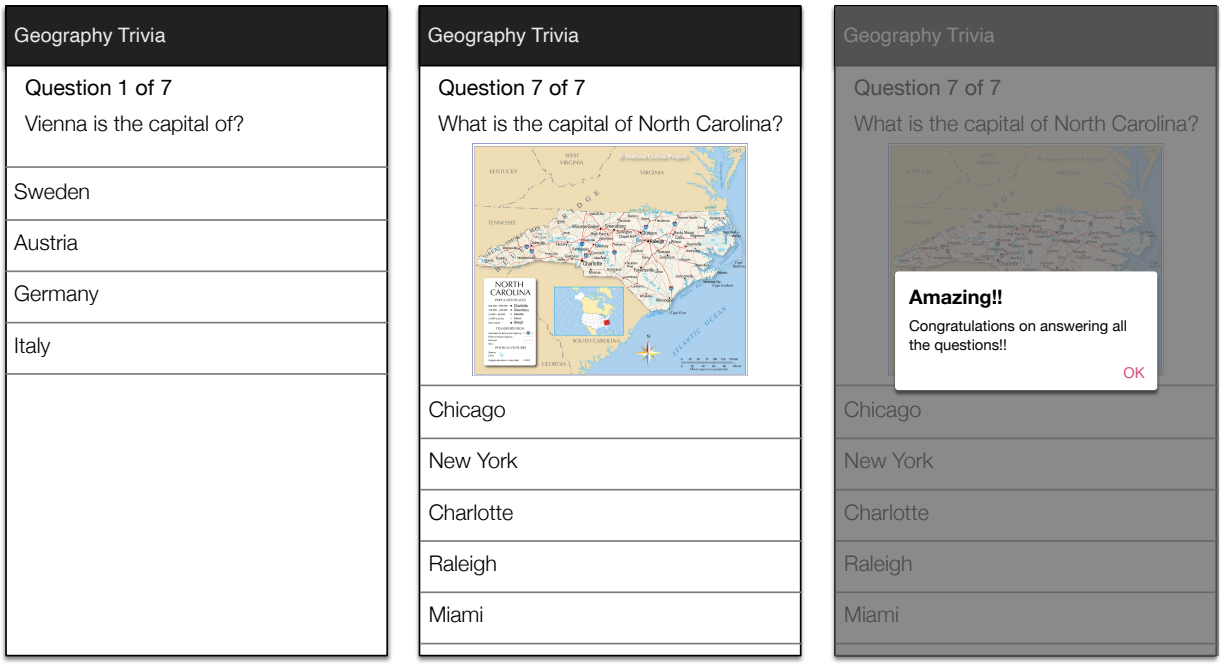
Figure 3, Code showing how to add Authorization Header to Request

Part 3 : Trivia List Fragment (30 Points)

This screen enables the user to view the trivia list. As shown in Figure 1(c), The requirements are as follows:

1. Clicking the “Logout” menu button should delete the auth token, and replace this fragment with the Login Fragment.
2. The list of available trivias should be retrieved be calling the **/api/trivias** API. **Note that this API requires the Authorization header to include the token, please create the OkHttp request and include the header as shown in Figure 3.**
   a. Create a Trivia class, and parse the returned list of trivias into an ArrayList containing the parsed Trivia objects. Use the parsed list of Trivia objects to display the trivia list in a ListView or RecyclerView.

3. Each row item should should display the trivia title and trivia description as shown in Figure 1(c).
4. Clicking on a row item should replace the current fragment with the Trivia Fragment, pass it the required data and Push the current fragment on the back stack.



(a) Trivia Fragment, showing the first question with no image

(b) Trivia Fragment, showing a question with an image

(c) Trivia Fragment, showing alert dialog with the last correctly answered question.

Figure 4, Application Wireframe

Part 4 : Trivia Fragment (50 Points)

The interface should be created to match Figure 4(a). The requirements are as follows:

1. The Fragment title should be set as the title of the trivia selected.
2. The /api/trivia/{trivia_id} should be called to retrieve the trivia details, which will include the list of trivia questions and the possible answer choices for each question. **Note that this API requires the Authorization header to include the token, please create the OkHttp request and include the header as shown in Figure 3.**
   a. Create a Question and Answer classes that should enable you to parse and store the trivia questions and answers.
   b. Upon receiving the trivia details from the **/api/trivia/{trivia_id}**, the response should be parsed, and questions should be stored in an ArrayList.
   c. Display the first question in the list as shown in Figure 4(a).
      i. Display Question # of total, for example “Question 1 of 7” indicating that the displayed question is the first out of 7 questions.
      ii. Display the question text.
      iii. If the question has a null “question_url” do not show the question image as

shown in Figure 4(a). If the question has a non-null “question_url” then display the question image using the Picasso library as shown in Figure 4(b).

iv. Display the list of the question’s possible answers in a ListView or RecyclerView as shown in Figures 4(a-b).

3. Clicking on an answer from the ListView or RecyclerView:
   a. You should check if the selected answer is correct or incorrect by calling the **/api/trivia/check-answer** api, which requires the question_id and answer_id and returns a boolean isCorrectAnswer attribute to indicate if the provided answer_id is is correct or incorrect for the provided question_id. **Note that this API requires the Authorization header to include the token, please create the OkHttp request and include the header as shown in Figure 3.**
   b. If the selected answer is incorrect, then display an alert dialog indicating that the answer is incorrect.
   c. If the select answer is correct, then display the next question of the trivia
      i. Update Question # of total to show the new question number.
      ii. Update the display to show the new question text and image.
      iii. Refresh the ListView/RecyclerView to show the list of answer for the new question.
   d. If the currently correctly answered question is the last question in the list of questions, display an alert dialog indicating that the user has completed this trivia as shown in Figure 4(c). Upon clicking on the alert’s dialog “OK” button the app should pop the back stack which should return back to the Trivia List Fragment.