Mobile App Development
Midterm Exam

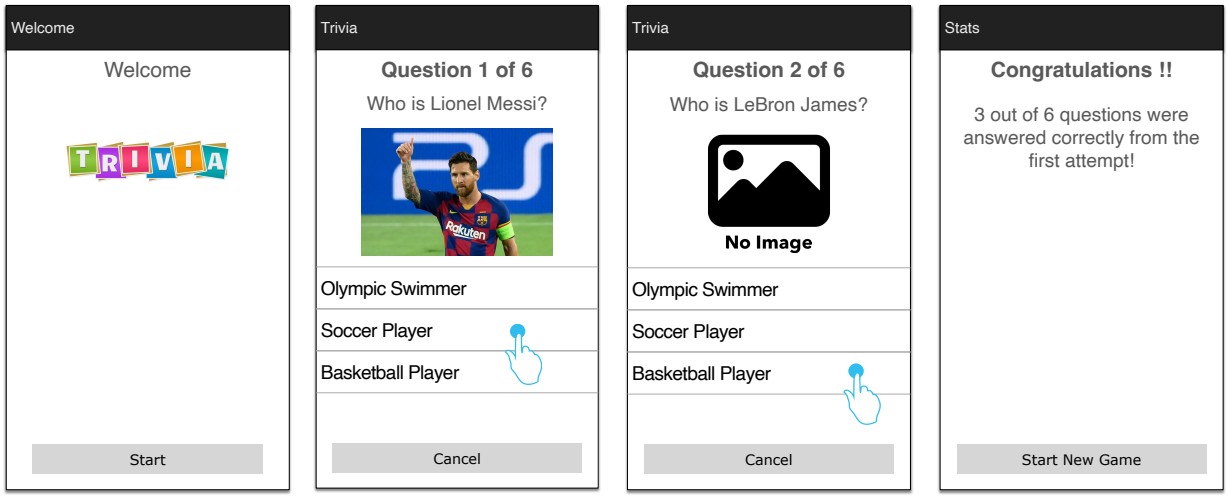
Basic Instructions:

1. This is the Midterm Exam, which will count for 15% of the total course grade.
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. During the midterm the teaching assistants and Instructors will pass by each student and ask them to demonstrate their application. Your interaction with the teaching assistants and instructors will be taken into consideration when grading your exam submission.
7. Please download the support files provided with the Midterm and use them when implementing your project.
8. 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.
9. 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.
10. **Do not try to use any Social Messenger apps, Emails, Or Cloud File Storage services in this exam.**
11. **Failure to follow the above instructions will result in point deductions.**
12. **Any violation of the rules regarding consultation with others will not be tolerated and will result disciplinary action and failing the course.**

Midterm Exam (100 Points)

In this assignment you will implement a trivia application.

1. You are provided with a PostMan File that includes the description of all the APIs required for this application. There are only two APIs to use in this project.
2. Use the OkHttp library in this app in order to make all the http connections and API calls. All the data returned by the APIs is in JSON format.
3. All the network calls should be done in a background thread.
4. All UI changes, updates and edits should be performed on the main thread.
5. This app will have one Activity and 3 Fragments, **all communication between fragments should be managed by the activity.**
6. You are required to use the provided skeleton application.



(a) Welcome Fragment (b) Trivia Fragment (c) Trivia Fragment (d) Stats Fragment

Figure 1, App Wireframe

Part 1 : Welcome Fragment (30 Points)

This screen allows the user to start a new trivia game, as shown in Figure 1(a). The requirements are as follows:

1. When the user clicks the “Start” button:
   a. Retrieve the trivia questions using the **/api/trivia** API. You should use OkHttp to get the trivia questions.
   b. Use the provided Question class to parse each question. Use the Answer class to parse each answer for a given question. After parsing the trivia questions, you should have an array list of questions, where each question has a list of answers.
   c. Upon completing the parsing of the trivia questions, the list of questions should be sent to the Main Activity, which should:
      i. Replace the current fragment with the Trivia fragment.
      ii. Send the trivia questions list to the Trivia fragment
   d. In case of any errors display a Toast Message indicating the error.

Part 2 : Trivia Fragment (60 Points)

This fragment allows the user to answer each of the trivia questions as shown in Fig1(b) The requirements are as follows:

1. At the top show the current question number and indicate the total number of questions, for example “Question 1 of 6”.
2. Display the question details:
    a. Question text: which is the question.
    b. Question image: some questions will have a not null ‘question_url’ value which is a url of an image describing the question. Some questions will have a null ‘question_url’ and should show no image as shown in Fig 1(c). Use the Picasso library to display the provided image url.
    c. Question Answers: each question will be accompanied with a list of possible answers provided by the api. Display the list of possible answers for the displayed question using a ListView as shown in Figure 1(b).
3. Clicking on an answer row item from the ListView of the possible answers:
    a. Communicate with the **/api/trivia/checkAnswer** API to check the provided answer. This API should be provided with the ‘question_id’ and the selected ‘answer_id’. The API JSON response will indicate if the the selected answer is correct or not.
    b. If the answer is correct, then refresh the displayed question to display the next question, question image url, and the answers for that question.
    c. If the answer is correct, and the this is the last question in the trivia, then communicate with the Main Activity, which should:
        i. Replace the current fragment with the Stats fragment.
        ii. Send the Stats fragment the required statistics.
    d. If the answer is incorrect, then display a Toast message indicating that the selected answer is incorrect.
4. Clicking the “Cancel” button should communicate with the Main Activity, which should:
    a. Replace the current fragment with the Welcome fragment.

Part 3: Stats Fragment (10 Points)

This fragment shows the statistics summary indicating the number of questions that were answered correctly from the first attempt as shown in Fig 1(d). The requirements are as follows:

1. These statistics should be accumulated in the Trivia fragment, and should be sent to this fragment to be displayed.
2. Clicking the “Start New game” button should communicate with the Main Activity, which should:
    a. Replace the current fragment with the Welcome fragment.

Section:	
Student Name:	
Student ID:	

Part #	Features	Total	Grade	Comments
P1	Welcome Fragment: Connect to the api, parse the trivia questions and answers, and send the list of questions to the Trivia Fragment	30		
P2	Trivia Fragment: Display question and answers correctly.	20		
P2	Trivia Fragment: Display question image url correctly.	10		
P2	Trivia Fragment: Display the answers for the question in a ListView	10		
P2	Trivia Fragment: Connect to the check answer api, parse the returned JSON, and able to display the next question if the answer is correct, or display a toast if the answer is in-correct.	20		
P3	Stats Fragment: collecting the statistics correctly in the Trivia fragment and displaying them correctly in the Stats fragment.	10		
	Total	**100**		

Table 1: Grading Key