Lab 1: Mobile Programming

Total possible points: 830 Points
Bonus points: 360 points
Minimum points required: 200 points

This project is designed to help you gain some more hands-on experience with mobile programming. This project consists of two parts. The first part involves adding two new features to the GeoQuiz app we covered in class. The second part is open-ended, in which we ask you to pick and study one open-source mobile-related project.

Part 1: The GeoQuiz Challenges (170 Points)

In this part, you will implement two new features for the GeoQuiz app.

Set up the starter code in Android Studio and run it on an Android device. Make sure you understand the behaviors of this version of GeoQuiz. Specifically, you will observe two things. First, users can rotate the screen to hide their cheating acts after they cheat. Instead of seeing the toast that says, "Cheating is wrong," they will see the toast that says "Correct answer" after navigating to the MainActivity and answering the question. Second, you will see that once the user cheats, GeoQuiz will present them with the same toast of "Cheating is wrong."

The two new features will address the issues described above. The first feature you need to implement is to persist CheatActivity's UI state across rotation and process death using the techniques we discussed in class. The second feature is to update GeoQuiz to track whether the user cheated on a question-by-question basis.

Deliverables

For this part, you should include the following deliverables in the Part 1 folder.

- **The debug apk with the name NewGeoQuiz.apk**. You can do so in Android Studio by following the screenshot below. Afterwards, the debug apk can be located in the app/build/outputs/apk/debug directory.
- **The source code of your app**. Please do not include build-related files in the submission, which can significantly increase the submission size. Similarly, you can do so in Android Studio by following the screenshot below with the *Clean Project* option.
- **A screen recording of the implemented features.** This recording should include your voice-over briefly explaining to the teaching staff how the implementation satisfies the feature requirements. Both Android Studio and Android phones have built-in screen recording functionality. Please keep the recording as concise as possible.

- **A README file.** You need to include your team information (name and email address) and brief explanations of how you implement the required app features. Only plaintext writeups are accepted, and markdown files are also okay.

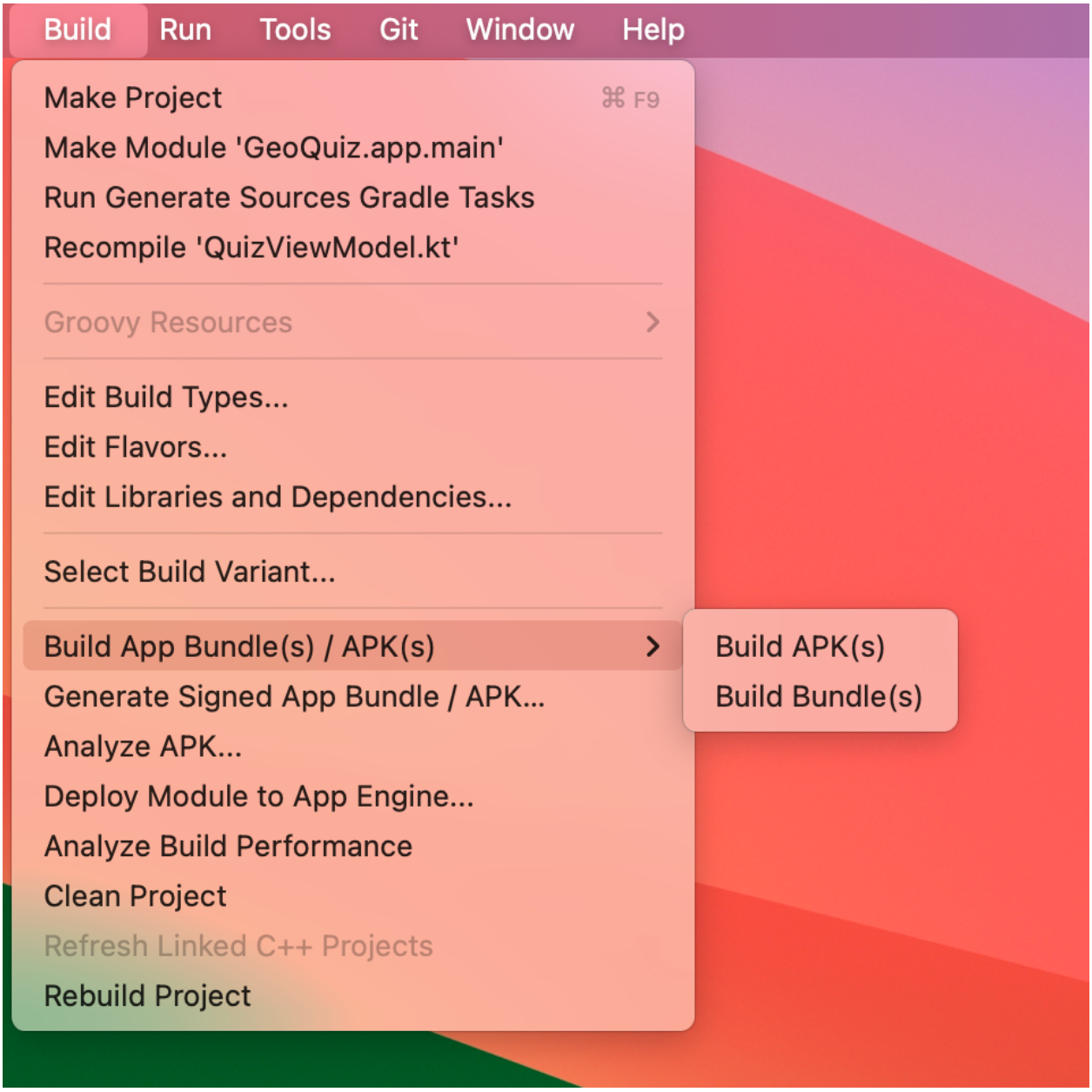

Figure 1: how to remove build-related files and generate a debug apk.

Gradings

Your submission will be graded by running it with Android Studio and the built-in emulator (Medium Phone API 35) and evaluating its functionality. In general, your submission will receive points for every feature requirement. Your app will not receive partial credit for incomplete feature implementation. Completion will be assessed based on all the deliverables described above. Professionalism includes good project organization, clean code (e.g., not a lot of unused code), necessary comments, and clear writing.

Requirement	**Brief Description**	**Points total**
apk	Included?	10
source code	Professionalism?	10
Screen recording	Clear?	20
Voice over	Clear?	10
Feature 1	Complete?	50
Feature 2	Complete?	50
README	Clear?	20

The check of the requirements should be self-explanatory based on the brief description. Below, we provide a more detailed description of how we check for each requirement. We will use the same criteria to grade requirements for all labs in this course and omit this detailed description for future labs. You are always welcome to check with the teaching staff if you have any questions.

What do we check?

- For each feature, we check if they are correctly implemented by looking at the source code, your README, screen recording, and the apk.
- For source code, we check to see if it consists of reasonable implementations and comments.
- For screen recording, we check if the recording includes a clear overview of the app's features.
- For voice-over, we check to make sure the descriptions match the app's features.
- For README, we check to see if it clearly includes all the information.

Part 2: Open-Source Mobile App Investigation (300 Points)

In this part, you will learn more about mobile programming by studying an open-source mobile project. Each project you study, based on the description below, will earn you some points. You will get the most points for the first project, and fewer points for the subsequent project. In other words, we use a decaying point system designed to reward your learning but also to reflect the diminishing return and help you allocate time to other course assignments.

For this part, you have to pick one of the mobile apps from the following list to study.

- The CriminalIntent app from the *Big Nerd Ranch Guide* book.
- The Todo app from the Google architecture sample: https://github.com/android/architecture-samples (There are many branches, but you only need to investigate the main branch).

What do you need to do?

- **Set up, compile, and run the app.** Interact with the app to understand its features.
- **Understand the app's workflow.** A reasonable way is to use `logcat` to add debug information throughout the app. You can also look into using the IDE's debugger. Add comments to different parts of the project source code to document your understanding. If there are online resources, e.g., codelabs, blogs, and tutorials, on the project you selected, please feel free to use them as references. However, you should complete the required deliverables by writing in your own words.
- **Deep dive into how two key features are implemented**. You should understand the core concepts behind each key feature and how the key APIs work. If you choose any apps from the Big Nerd Ranch Guide book, reading through the relevant chapters should give you a good starting point, and you can fill the knowledge gap by looking up the Android official documentation. How much detail should you go into? Concrete enough to provide the logical steps for replicating the feature. Here is a quick example. Assume one of the key features is to allow the user to take a photo with the camera. You should talk about the permission setup in `AndroidManifest`, how to use Intent to start the camera app, how to use `FileProvider` to store the photo, and how to view the photo as the `bitmap`.

Deliverables

For this part, you should include the following deliverables in the Part 2 folder.

- **The debug apk**. You can do so in Android Studio by following the screenshot as described in Part 1. Afterwards, the debug apk can be located in the app/build/outputs/apk/debug directory.

- **The source code of your app**. Please do not include build-related files in the submission, which can significantly increase the submission size. Similarly, you can do so in Android Studio by following the screenshot below with the *Clean Project* option.

- **A screen recording of the app's features with your voice-over**. This recording should briefly explain to the teaching staff the key features and how this app satisfies the selection requirements. Both Android Studio and Android phones have built-in screen recording functionality. Please keep the recording as concise as possible.

- **A technical write-up about the ins and outs of the project**. This write-up should contain enough details so that readers can gain a high-level understanding of the project and the key steps to implement the project. Things you should consider including are steps to set up the project (especially if there are dependency nightmares), an overview of the app, key features of the app, and how each feature is implemented.

- **A learning note** that summarizes what you have learned.

- **A README file.** You need to include your team information (name and email address) and provide additional information you want to share with the teaching staff. Only plaintext writeups are accepted, and markdown files are also okay.

Grading

The completion of each requirement for the first mobile project you investigate will give you the full points. We will skim through your submission to check if they are reasonable.

Requirement	**Brief Description**	**Points total**
apk	Included?	10
source code	Reasonably commented?	10
Screen recording	Clear?	20
Voice over	Clear?	10
Technical writeup - project setup	Clear?	20
Technical writeup – app overview	Clear?	50
Technical writeup - key feature 1	Clear?	50
Technical writeup - key feature 2	Clear?	50
Learning notes	Included?	70
README	Clear?	10

Part Bonus 1: Refactoring The GeoQuiz App (160 Points)

In Part 2, you have learnt the canonical design of the list-detail layout. It is time to apply that to give the GeoQuiz a brand-new look. Refactor the GeoQuiz app from Part 1 so that the user is presented a list of questions once they launch the app, and clicking each item on the list will allow them to view the details of the question.

Deliverables

For this part, you should include the following deliverables in the Part Bonus 1 folder.

- **The debug apk**. You can do so in Android Studio by following the screenshot as described in Part 1. Afterwards, the debug apk can be located in the app/build/outputs/apk/debug directory.
- **The source code of your app**. Please do not include build-related files in the submission, which can significantly increase the submission size. Similarly, you can do so in Android Studio by following the screenshot below with the *Clean Project* option.
- **A screen recording of the app's features with your voice-over**. This recording should briefly explain to the teaching staff how you design the list-detail GeoQuiz app. Both Android Studio and Android phones have built-in screen recording functionality. Please keep the recording as concise as possible.
- **A README file.** You need to include your team information (name and email address) and brief explanations of how you implement the required app features. Only plaintext writeups are accepted, and markdown files are also okay.

Gradings

Your submission will be graded by running it with Android Studio and the built-in emulator (Medium Phone API 35) and evaluating its functionality. In general, your submission will receive points for every feature requirement. Your app will not receive partial credit for incomplete feature implementation. Completion will be assessed based on all the deliverables described above. Professionalism includes good project organization, clean code (e.g., not a lot of unused code), and clear writing.

Requirement	Brief Description	Points total
apk	Included?	10
source code	Professionalism?	10
Screen recording	Clear?	20

Voice over	Clear?	10
List-detail view	Complete?	100
README	Clear?	10

Bonus Part 2: More Open-Source Mobile App Investigation (200 Points)

If you enjoy reading and understanding the source code for mobile projects, you can earn up to 200 bonus points. Earning bonus points earlier in the term not only helps you get more comfortable for mobile programming but also gives you more flexibility in scheduling subsequent course work to achieve your learning objectives. Therefore, we recommend you attempt this bonus part if you have time.

We will use the same grading rubric as in Part 2 to assign points, with the key difference that each invested project will earn you 100 points. That is, points will be proportionally scaled for each requirement.

Here are some tips and guidelines for picking additional projects.

- You should pick a project that is of reasonable complexity. For example, other apps from the *Big Nerd Ranch Guide* book are reasonable options, and so are the sample apps from Google.
- The app should at least touch upon two of the four major themes in mobile programming: user interface, sensor data, persistence, and network.
- The app should have a focus on one of the four themes above.
- We strongly encourage you to pick a project that can provide useful information and practices for your final project.
- This open-source project can be written in any languages, as long as it can be compiled to run on Android devices.

Checkpoint Contributions

Students must submit work that demonstrates substantial progress towards completing the project on the checkpoint date. Substantial progress is judged at the discretion of the grader to allow students flexibility in prioritizing their efforts. For this assignment, completion of the first part will be considered as making substantial progress. Projects that successfully submit a checkpoint demonstrating significant progress will receive 10 points.

Errata

Report errors by emailing tian@wpi.edu with the subject line *[CS4518] Lab [Number]*. Thank you for taking the time to help us improve our courses.