Lab Assignment #3 – Developing Robust Android Applications with Networking, Navigation, and Data Persistence

Due Date: Mid-night (11.59 pm) **12th Nov. 2024** **Marks/Weightage: 30/15%**
[Sec001- 14th Nov, Sec002-12th Nov, Sec003-13th Nov]

End Date: Mid-night (11.59 pm) 1***5th Nov. with 20% late penalty. No Exceptions***

Note: You are required to demonstrate the assignment as per scheduled lab session as announced by your teacher. 25% penalty for not demonstrating the assignment

IDE: Android Studio – Koala Version and Kotlin Jetpack Compose

Purpose: The purpose of this lab assignment is to:

- Set up Retrofit for making network calls in Android applications.
- Implement network calls using Kotlin coroutines and Retrofit.
- Handle responses and errors effectively in network operations.
- Implement navigation between different Compose destinations.
- Pass arguments between destinations within an Android app.
- Design responsive navigation for foldables and large screens.
- Utilize the resizable emulator to test navigation on various screen sizes.
- Set up and configure the Room library in an Android project.
- Create entities, DAOs, and the database for storing data locally.
- Implement methods for saving, reading, updating, and deleting data using Room.
- Implement the MVVM architectural pattern for better code organization and maintenance.
- Utilize the repository pattern to abstract data access and management.

References: Textbook, ppt slides, class examples, and Android tutorials (**https://developer.android.com/develop/ui/views/theming/look-and-feel**). This material provides the necessary information that you need to complete the exercises.

Be sure to read the following general instructions carefully:
- This assignment must be completed in pairs by all the students. This is based on pair programming.
- You will have to **demonstrate your solution in a scheduled lab session** and upload the solution on eCentennial through the **assignment link under Assessments**.

Android Project Naming Rules:

Step 01: You must name your Android Studio **project** according to the following rule:
yourfullname_COMP304*SectionNumber*_Labnumber
For Example: **johnsmith_COMP304Sec003_Lab03. Save location drive name can be C:\COMP304\Assignments or D:\COMP304\Assignments etc.**
If you have more than one exercise in the assignment, then you need to create separate project for each exercise.

Step 02: Submission rules

Once you complete, run and test projects for all the exercises, then submit your projects as one **zip file and it should be** named according to the following rule:
yourfullname_COMP304SectionNumber_Labnumber.zip.
Example: **johnsmith_COMP304Sec003_Lab03.zip (if your section is 003)**

Exercise 1:

You will develop a new app to help users track weather conditions for different locations. The app should include the following functionalities:

- Network calls to fetch weather data from an API.
- Navigation between different screens to display weather details.
- Local data storage using the Room library.
- Implementation of MVVM architecture and repository pattern for better code organization and data management.

Features and Implementations:

1. **Networking with Retrofit and Kotlin Coroutines**:
   - Set up Retrofit for making network calls to a weather API.
   - Implement network calls using Kotlin coroutines to handle asynchronous tasks.
   - Handle responses and errors effectively in network operations.
2. **Jetpack Navigation**:
   - Implement navigation between different Compose destinations.
   - Pass arguments between destinations within the app (e.g., passing location details).
   - Design responsive navigation for foldables and large screens.
   - Utilize the resizable emulator to test navigation on various screen sizes.
3. **Local Data Storage with Room**:
   - Set up and configure the Room library in the project.
   - Create entities, DAOs, and the database for storing favorite locations and weather data locally.
   - Implement methods for saving, reading, updating, and deleting data using Room.
4. **App Architecture (MVVM and Repository Pattern)**:
   - Implement the MVVM architectural pattern.
   - Create ViewModel classes for managing UI-related data and business logic.
   - Utilize the repository pattern to abstract data access and management.

Evaluation table:

Item	Percentage of Total Mark	Details
Functionality:	**80%**	
Correct implementation of Retrofit and Kotlin coroutines:		
Network setup and asynchronous handling	20%	Ensure Retrofit is set up correctly and network calls are handled using Kotlin coroutines.
Correct implementation of Jetpack Navigation:		

Item	Percentage of Total Mark	Details
Navigation between Compose destinations	10%	Implement navigation and argument passing between different screens.
Implementation of responsive UI and Room library:		
Responsive navigation and local data storage	30%	Design responsive navigation layouts and configure Room for local data storage.
Correct implementation of MVVM and Repository Pattern:		
ViewModel and repository setup	20%	Ensure ViewModel classes manage UI data and repositories abstract data access.
Friendliness:	**15%**	
Alignments of UI controls	10%	UI controls should be properly aligned and organized, providing a visually appealing layout.
Friendly I/O	5%	The app should provide a user-friendly interface with intuitive input/output operations.
Comments, Correct Naming of Variables, Methods, Classes, etc.	**5%**	Code should be well-documented with appropriate comments. Variables, methods, and classes should follow proper naming conventions.
Total	**100%**	