# Lab Assignment #2 – Designing, Developing, and Architecting Interactive Android Applications with Material Design 3 and Advanced Features

Due Date: Mid-night (11.59 pm) **13th Oct 2024** **Marks/Weightage: 30/8%**

End Date: Mid-night (11.59 pm) 1***6th Oct. 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:

- Apply Material Design 3 color schemes to create visually appealing and consistent app themes.
- Design responsive UIs that adapt to large screens and foldable devices using WindowSizeClass.
- Implement accessibility features to ensure apps are usable by all users, including those with disabilities.
- Implement the Model-View-ViewModel (MVVM) architecture to enhance code separation and maintainability.
- Create composable functions using LazyColumn to efficiently display lists.
- Utilize Jetpack libraries to streamline and enhance app development.
- Apply dependency injection to improve code modularity and testability.

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 individually by all the students.
- 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_Lab02. 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_Lab02.zip (if your section is 003)**

## Exercise 1

You will develop a new app to help users manage their tasks efficiently with a visually appealing interface and advanced features. The app should include the following functionalities:

- A home screen displaying a list of tasks.
- A screen to create a new task.
- A screen to view and edit an existing task.
- Responsive layouts and accessibility features.
- Implementation of MVVM architecture.
- Use of Jetpack libraries and dependency injection.

### Features and Implementations

1. **Material Design 3**:
   - Apply Material Design 3 principles and color schemes to the app.
   - Use Material Design components (e.g., AppBar, Buttons, Cards, TextFields) to enhance the UI.
2. **Responsive UI Design**:
   - Implement responsive layouts using WindowSizeClass to adapt the app for large screens and foldable devices.
3. **Accessibility**:
   - Add accessibility features such as content descriptions, larger touch targets, and proper navigation for screen readers.
4. **App Architecture (MVVM)**:
   - Implement the MVVM architecture.
   - Create ViewModel classes for managing UI-related data.
   - Use StateFlow to observe data changes in the UI.
5. **Jetpack Compose and LazyColumn**:
   - Use LazyColumn to display lists efficiently.
   - Create composable functions for the list items and other UI components.
6. **Jetpack Libraries and Dependency Injection**:
   - Utilize Jetpack libraries.
   - Implement dependency injection.

### Activities and Navigation

1. **Home Activity**:
   - Displays a list of tasks using a LazyColumn.
   - Each task item displays a title and a status indicator.
   - Includes a Floating Action Button (FAB) to navigate to the Create Task Activity.
2. **Create Task Activity**:
   - Contains input fields for the task title, description, and due date.

- Includes a save button to save the task and return to the Home Activity.

3. **View/Edit Task Activity**:

- Displays the selected task's details.
- Allows the user to edit the task.
- Includes a save button to save the changes and return to the Home Activity.

**(10 marks)**

**Evaluation table:**

| Item | Percentage of Total Mark | Details |
|---|---|---|
| **Functionality:** | **80%** | |
| **Correct implementation of MVVM architecture:** | | |
| ViewModel classes and StateFlow | 30% | Ensure ViewModel classes manage UI-related data and StateFlow are used for observing data changes. |
| **Correct implementation of Material Design 3:** | | |
| Use of Material Design components and color schemes | 20% | Apply Material Design components (AppBar, Buttons, Cards, TextFields) and color schemes. |
| **Implementation of responsive UI and accessibility features:** | | |
| Responsive layouts using WindowSizeClass | 10% | Implement responsive UI layouts that adapt to different screen sizes, including foldable devices. |
| Accessibility features for usability | 10% | Add accessibility features such as content descriptions, larger touch targets, and proper navigation for screen readers. |
| **Correct implementation of Jetpack Compose and LazyColumn:** | | |
| Use of LazyColumn for efficient list display | 10% | Implement LazyColumn for displaying lists efficiently. |
| **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%** | |