Lab Assignment #2 – Developing a Basic Android Application using Kotlin and Jetpack Compose with Multiple Activities, Nav Controller

Due Date: Mid-night (11.59 pm) **11th Oct. 2025** **Marks/Weightage: 30/12%**

End Date: Mid-night (11.59 pm) ***14th Oct. 2025 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 – Meerkat Feature Drop Version (2024.3.2) and Kotlin Jetpack Compose

Purpose: The purpose of this lab assignment is to:

- Apply the advance level of the Kotlin programming language.
- Develop and implement UI components using composable functions, previews, and modifiers.
- Design and construct complex UI layouts using Column, Row, Box, and Lists in Jetpack Compose.
- Navigation using Nav Controller and using of Data Classes and Card
- Apply Jetpack Compose best practices to create responsive and maintainable user interfaces.

References: Textbook, ppt slides, class examples, and Android tutorials (**http://developer.android.com/training/basics/firstapp/creating-project.html**). 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_COMP304Sec001_Lab02. Save location drive name should be C:\COMP304-001-S25\Assignments or D:\COMP304-001-F25\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 (ONLY .zip file format is allowed. No .rar or .7z etc. file formats allowed) and it should be** named according to the following rule:
yourfullname_COMP304SectionNumber_Labnumber.zip.

Example: **johnsmith_COMP304Sec401_Lab02.zip (if your section is 001)**

Exercise 1: *(Refer **NoteApp** covered during the class- under Content → WEEK05. You are not allowed to modify it in-place. For assignments, you are required to create a new app as per instructions mentioned in this assignment. Yes you can copy the code from class examp* **[20 marks]**

In this assignment, you will develop a simple Android application using Kotlin and Jetpack Compose. The purpose is to get you familiar with the Android development environment, understand the basics of the Kotlin programming language, and learn the fundamental principles of Jetpack Compose for building UI components. You will develop a basic app with multiple activities to demonstrate your understanding of these concepts.

Assignment Requirements:

App Overview

You will build a basic note-taking app named "**WritingPadApp**" The app will have the following features:

- A home screen/activity displaying a list of notes.
- A screen/activity to create a new note.
- A screen to view and edit an existing note.

Activities and Navigation:

1. Home Activity:
   - Displays a list of notes using a LazyColumn.
   - Each note item displays a title, truncated description review, e-mail id, and date
   - Includes a Floating Action Button (FAB) to navigate to the Create WritingPad Activity.

2. Create WritingPad Activity/Screen (for creating notes):
   - Contains input fields for the note title, note description, e-mail, priority level (Low, Medium, High using Drop Down) and due date (5 items of information)
   - Includes a **Save** button to save the note and return to the Home Activity/Screen.

3. View/Edit Note Activity/Screen
   - Displays the selected note's title and description, e-mail, priority level and due date
   - Allows the user to edit the note.
   - Includes a **Save/Update** button to save the changes and return to the Home Activity.

Jetpack Compose UI Elements:

1. Home Activity:
   - Use LazyColumn for the list of notes.
   - Use Card for individual note items, i.e. each card should contain title, description, priority level and date

- Use FloatingActionButton for the adding/creating notes.

2. Create WritingPad Activity/Screen:
   - Use Column to arrange input fields vertically.
   - Use TextField for the note title, description, priority and due date
   - Use Button for the Save action

3. View/Edit Note Activity/Screen
   - Similar to Create Note Activity but pre-filled with existing note data.

Additional Requirements:

1. App Architecture:
   - For simplicity, use a basic architecture without MVVM for this assignment.
   - Create a Note data class.
   - Use basic Kotlin collections to manage the list of notes.

2. Kotlin Basics:
   - Implement basic Kotlin programming concepts such as variables, control structures, functions, and classes.
   - Use Kotlin collections to manage the list of notes.

3. **Setup and IDE**:
   - Ensure Android Studio is set up to use Kotlin (Meerkat Feature Drop version).
   - Target Android 14 (API Level 34) or higher for your application.

Evaluation table/Rubric:

Item	**Percentage of Total Mark**	**Details**
Functionality:	**80%**	
Correct implementation of activities:		
Class code for main activity and two other activities	20%	Ensure the main activity (Home Activity), Create Note Activity, and View/Edit Note Activity are correctly implemented.
UI in Jetpack Compose	20%	The UI components should be implemented using Jetpack Compose, including LazyColumn, Card, TextField, Button, and FloatingActionButton.
Correct implementation of Event Handlers and life cycle methods:		
Event handlers for UI interactions	20%	Proper handling of user interactions, such as adding a note, editing a note, and saving changes.

Item	Percentage of Total Mark	Details
Life cycle methods for activities	20%	Correct implementation of life cycle methods (e.g., onCreate, onStart, onResume) to manage activity state transitions.
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%**	