FACULTY OF SCIENCES, TECHNOLOGY INNOVATION

DIPLOMA IN INFORMATION AND COMMUNICATION TECHLOGY

GROUP ASSIGNMENT

TO: MR. VISION THONDOYA

FROM: GROUP B

TITLE: NOTEPAD APPLICATION ASSIGNMENT

DATE OF SUBMISSION: 12TH SEPTEMBER 2025

MEMBERS:

EMMANEL KHAUKE DICT0923

VITUMBIKO KAUTEMBA DICT0823

LEORNARD MWATAMADZI DICT1923

Notepad App Development Report

Introduction

The process of developing a Notepad app for Android was quite exciting and also some how challenging. The main goal of the app is to give users a simple way to create, read, and delete notes whenever they need to. We wanted it to be straightforward and easy to use, so that anyone looking for a quick note-taking tool could rely on it. we talk about the main features of the app, how we applied programming principles like Separation of Concerns, the Single Responsibility Principle, and User-Defined Functions, and also the challenges we came across during development. We also explain how we managed to solve those challenges along the way.

The Notepad app includes the following Main features:

• User Authentication: Users can sign up and log in to access their personal notes. The login system ensures notes are tied to a specific user, maintaining privacy and data separation.

• Note Creation: Users can create notes with a title and content in a single activity. After saving, the note appears immediately in a list of saved notes, for quick note-taking.

• Note Viewing: All notes for the logged-in user are displayed in a Recycler View on the same page, showing the title and content for easy access.

• Note Deletion: Users can delete notes by taping a delete icon next to each note, with a confirmation dialog to prevent accidental deletions.

Programming Principles Applied

Separation of Concerns (SOC)

The app is structured to separate different functionalities. The UI logic is handled in NotesActivity .kt, which manages user input and displays notes in a RecyclerView. Data access is handled by NoteRepository .kt, which interacts with the Room database via NoteDao.kt. The NotesViewModel.kt acts as a bridge between the UL and data layers, handling business logic and ensuring data is fetched or saved asynchronously. This separation makes the code easier to debug and modify. this helps us to update the UI without touching the database logic.

Single Responsibility Principle (SRP)

Each class has a single, well-defined purpose. The NoteDao.kt only handles database queries (insert, delete, select). The NoteRepository .kt encapsulates data operations, ensuring the ViewModel doesn't directly interact with the database. The NoteAdapter.kt is responsible for displaying notes in the RecyclerView. This approach reduces complexity and makes each component easier to test. With this, we can test database inserts without involving the UI.

User-Defined functions (UDF s)

we created custom functions to handle repetitive tasks. In NotesView-Model.kt, the insertNote and deleteNote functions encapsulate the logic for saving and deleting notes, using coroutines for asynchronous execution. In NotesActivity .kt, a helper function in the NoteAdapter.kt manages binding note data to the RecyclerView. These functions improve code reusability and readability, reducing duplication. The insertNote function ensures every save operation refreshes the note list automatically .

Challenges Faced and Solutions

Developing the Notepad app presented several challenges, particularly with saving notes. Below, we outline the key issues and how we addressed them:

Challenge 1: Notes Not Saving to Database, notes were not being saved to the database, and no errors were displayed, making debugging dificult. The issue stemmed from a misconfigured App-Database.kt, where the Note entity were not included in the Database annotation., the insert operation in NoteRepository .kt were not properly handling coroutine scopes, causing silent failures.

Our Solution: we updated AppDatabase.kt to include Note:class in the entities list and added a noteDao() method. we also switched to using viewModelScope in NotesViewModel.kt for database operations, ensuring inserts were performed on a background thread and the UI were updated on the main thread. we added detailed logging.

Challenge 2: Notes Not Displaying in RecyclerView Even after saving, notes didn't appear in the RecyclerView immediately. The problem was that the LiveData in NotesViewModel.kt wasn't refresh after inserts.

Solution: we modified insertNote in NotesViewModel.kt to fetch the updated note list after each insert and post it to the Live Data. This ensured the RecyclerView reflected new notes. we also verified that NoteAdapter.kt used ListAdapter with DiffiUtil for efficient updates.

Challenge 3: User ID Not Passed Correctly The app sometimes failed to associate notes with the correct user because the user id wasn't passed reliably from Login Activity .kt toNotesActivity .kt .

Solution: we ensured LoginActivity .kt passed the user id via an Intentextraandvalidateditin NotesActivity .kt , displaying a toast and closing the activity if invalid. This made the app more robust.

These challenges taught us the importance of thorough testing and logging. While the app now works, we plan to add features like note editing and image support in the future, which will require careful handling of permissions and storage.

Conclusion

The Notepad app, functional note-taking experience with user authentication, note creation, viewing, and deletion. By applying SOC, SRP, and UDF s, we created a maintainable codebase

that separates UI, data, and business logic. Despite challenges with database saves, RecyclerView updates, and user ID handling, we resolved these through proper database configuration, coroutine usage, and robust error handling. The app meets its core requirements, but future improvements could include better error messages and additional features like search functionality .