
**FACULTY OF SCIENCES, TECHNOLOGY INNOVATION**

**INFORMATION AND COMMUNICATION TECHLOGY**

**FROM** :GROUP 4 WEEKEND STUDENTS

**TO** :MR. V. THONDOYA

**GROUP MEMBERS:**

| **Name** | **Reg No.** |
|---|---|
| RONEX KANDUNA | DICT0922 |
| GOODLUCKY HARA | BICTU1024 |
| DUMISANI NKHATA | DICT1823 |
| TAONGA MUNTHALI | BICTU2724 |

**PROGRAMME OF STUDY** :DIPLOMA IN INFORMATION AND COMMUNICATION TECHNOLOGY

**LEVEL** : 2

**SEMESTER** : 4

**COURSE NAME** : MOBILE APP DEVELOPMENT

**COURSE CODE** : BICT 2406

**ASSIGNMENT TITLE** : DEVELOP A NOTEPAD APP

**DUE DATE** :12TH SEPTEMBER 2025

## INTRODUCTION

We developed this app as a startup to better understanding mobile app development. We have learnt using android studio, jetpack compose and room database. It is a simple note taking application that supports user login, allowing users to create, edit, view and remove their notes.

With this project we have learnt good programming practices which include the separation of concern (SOC), single responsibility principle (SRP) and the use of defined functions (UDFs).

## 1.0 MAIN FEATURES

**Login:** A user enters a name and then taps a login button. Each signed user is presented with a sign-out option.

**Creating notes:** Users are able to create notes. The interface for creating a new note has a title, body text and an optional attachments such as images or other files.

**Viewing notes:**

They can also view a list of notes that can be scrolled. For a new screen without notes the app displays a friendly message asking a user to tap a + icon so they can add a note.

**Updating notes:** The existing notes can be updated by taping on the edit option.

**Removing notes:** The existing notes can be deleted permanently by taping a delete option.

**Data persistence:**

There is no special database for storing notes, instead all user data is stored locally using jetpack room. This makes the data to be restored even if the app crashes or restarts.

**User interface:**

Using jetpack compose helped me to come up with an intuitive user interface. Components used include Material design tools such as Scaffold, TopAppBar, FloatingActionButton and elevetedCard. These components were used for a better user experience. We also updated the application's icon rather than android's defaults.

## 2.0 GOOD PROGRAMMING PRINCIPLES

### (a) separation of concern

- **UI Layer:** This consist of composables including NotesScreen, SignInScreen, EditNoteScreen) to present a user with interactive screens

- **View Models:** These are used to manage the state of the application and interact with the data repository.
- **Repository layer:** This consists of logic for access and storage of notes in Room
- **Data Layer:** It has NoteDao, NoteEntity and NoteDatabase that handle persistence

The SOC easures that a layer is confined within is specific task, making the application easier to maintain.

**(b) Single responsibility principle (SRP)**

Classes and functions have independent specific purposes.

- AuthViewModel handles login state
- NoteViewModel handles note crud (Create, Read, Update and Delete) operations.
- NotesDao consists od methods to access data
- IS screens presents content and invokes actions.

**(c) Use of defined functions**

To prevent duplicates and improve usability we defined functions for each specific action.

- saveNote manages crating new note and updating the existing notes. Is contain user input fields such as title, body and image.
- deleteNote () removes existing note from repository.
- onAdd(), onEdit(), onSignout() functions are paseed to UI tools for navigation.

## 3.0 CHALLENGES ENCOUNTERED AND SOLUTIONS

1. **Dependency errors**

We encountered unresolved references for KSP, Coil and Navigation despite successfully adding their dependences in build.gradle.kts. To resolve we updated build.gradle.kts to use ksp version that is compatible with the latest Kotlin eg., 2.0.21 – 1.0.27 and added the missing coil dependences.

2. **stateFlow and state mismatch.**

The NoteViewModel exposed StateFlow<List<Note>> yet the UI required State<List<Note>> the solution was to use collectAsState(initial = empty()) function inside the composable to convert StateFlow into a compose friendly state.

3. **Experimental API**

We encountered an error with using Material 3 API called TopAppBar as it is experimental in android. To resolve I annotated with `@OptIn(ExperimentalMaterial3Api::class)` feature above this API

4. **Icon of the application**

Initially the app used default icon for Android launcher. We created an icon image using resources in Android Studio's Image Asset tool that replaced the default launcher

5. **Other errors**

Some syntax related errors were encountered. To resolve we had to search for relevant topic in forums on google. References are included at the end of this report.

## 4.0 CONCLUSION

We have advanced my knowledge and programming skills with this project. We understand practically that SOC, SRP and UDF makes a project organized, modular and easier to maintain.

This app can be upgraded by adding features such as search, and cloud synchronization

**REFERENCES**


1. Ambiguous graddle errors https://stackoverflow.com/questions/53826669/conflicting-import-imported-name-toast-is-ambiguous
2. Coil errors solution https://stackoverflow.com/questions/74051761/unresolved-reference-compose-in-coil-compose-asyncimage
3. Datastore errors https://github.com/android/codelab-android-datastore/issues/75
4. Export project to github repository https://www.youtube.com/watch?v=GhfJTOu3_SE
5. https://www.google.com/search?q=notepad+icon&oq=notepad+icon&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTIHCAEQIRiPAtIBCDMzNzJqMGo3qAIAsAIA&sourceid=chrome&ie=UTF-8
6. Ksp dependency errors https://kotlinlang.org/docs/compose-compiler-migration-guide.html
7. Learning android fundamentals. https://developer.android.com/guide/components/fundamentals?hl=en
8. Room errors solution https://stackoverflow.com/questions/78008527/unresolved-reference-kapt-and-ksp-in-android-studio-when-trying-to-do-the-setup response 12
9. State type mismatch errors https://stackoverflow.com/questions/78617852/android-viewmodel-mutablestateflow-type-mismatch