Ahnaf Tahmid –IT23SP

Course: Product Design and Development

Project: StudyBuddy Planner

Version: v1.0

StudyBuddy Planner – Project Report

1. Overview

StudyBuddy Planner is a simple study task planner for students. The goal is to help users keep track of their daily study tasks and see a quick overview of how much work is completed or still pending. The app focuses on being small, clear and fully functional, instead of trying to do too many things.

2. Architecture & Tech Stack

- Platform: Android, Kotlin
- UI: Jetpack Compose + Material 3
- Architecture: MVVM (Model–View–ViewModel)
- State Management: ViewModel + StateFlow
- Navigation: Jetpack Compose Navigation (NavHost, NavController)
- Data Storage: DataStore (local persistence) via a TaskRepository
- Testing: JUnit unit test for task statistics

The app is structured into three main layers:

- UI layer: Composables such as HomeScreen, AddEditTaskScreen and StatsScreen.
- ViewModel layer: TaskViewModel exposes the current list of tasks as StateFlow<List<Task>> and provides functions like addTask, toggleDone, clearAll, and stats.
- Data layer: TaskRepository reads and writes the list of Task objects using DataStore, so tasks are stored on the device and survive app restarts.

3. Main Features Implemented

- ○ Home screen showing a list of tasks with title, subject and due date
- ○ Add/Edit screen to create new tasks with input validation
- ○ Stats screen showing total, completed, and pending task counts
- ○ Checkboxes on the list to mark tasks as done / not done
- ○ “Clear all tasks” button on the stats screen
- ○ Light/Dark theme support through Material 3 theme

4. State & Data Persistence

The TaskViewModel holds all tasks in memory using MutableStateFlow. On startup, the ViewModel collects data from TaskRepository.tasksFlow. If there is no saved data yet, it inserts two sample tasks and persists them. After every change (add, toggle, clear), the ViewModel updates the in-memory list and calls repository.saveTasks() so DataStore always stays in sync.

5. Testing & GitHub Workflow

A JUnit unit test (TaskStatsTest) verifies the pure function calculateTaskStats(), which returns a Triple(total, done, pending). The project is in a public GitHub repository with multiple commits and a README.md including screenshots and instructions. A release v1.0 has been created with an APK attached so the app can be installed on a real device.

6. Challenges & Lessons Learned

- Understanding how to connect Jetpack Compose, Navigation, and ViewModel together took some time, but it became clearer after building the navigation graph and state flow step by step.
- Saving objects with DataStore required designing a simple data model and thinking about persistence beyond just in-memory lists.
- Creating a clean UI in Compose showed how powerful composables are, but also how important it is to structure them into small reusable parts.
- Using GitHub releases and tagging v1.0 helped to understand a more professional workflow for versioning and sharing APKs.

Overall, this project helped me practice a complete Android development flow: from idea and UI design to state management, persistence, testing, GitHub and release.