Taskapolis – A Daily Task Management App

Submitted by

KARTHIK G **2116220701504**

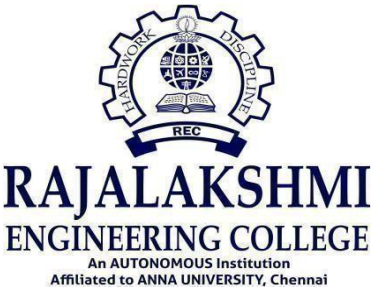
in partial fulfillment for the award of the degree

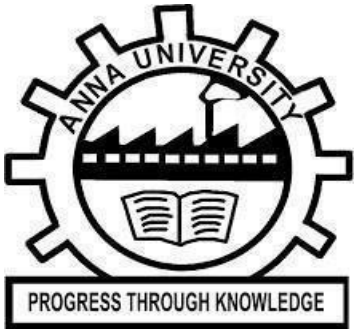
of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING





RAJALAKSHMI ENGINEERING COLLEGE

ANNA UNIVERSITY, CHENNAI

MAY 2025

BONAFIDE CERTIFICATE

Certified that this Project titled " **Taskapolis – A Daily Task Management App**" is the bonafide work of **"KARTHIK G (2116220701504) "** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. P. KUMAR, M.E., Ph.D.,

HEAD OF THE DEPARTMENT

Professor,

Department of Computer Science

And Engineering,

Rajalakshmi Engineering

College Thandalam,

Chennai – 602 105

SIGNATURE

Dr. N. Duraimurugan .,M.E Ph.D,

SUPERVISOR

Professor,

Department of Computer Science

And Engineering,

Rajalakshmi Engineering

College Thandalam,

Chennai – 602 105

Submitted to Project Viva-Voce Examination held on _____

Internal Examiner **External Examiner**

TABLE OF CONTENTS

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of ourlife and showering his blessings through the endeavor to put forth this report. Our sincere thanks to our Chairman **Mr. S. MEGANATHAN, B.E,F.I.E.**, our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.,** and our respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN**, **Ph.D.,** for providing us with the requisite infrastructure and sincere endeavoring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.,** our belovedPrincipal for his kind support and facilities provided to complete our workin time. We express our sincere thanks to **Dr.P.KUMAR, Ph.D.**, Professorand Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work. We convey oursincere and deepest gratitude to our internal guide, **Dr. M. Rakesh Kumar., M.E., Ph.D.,** Department of Computer Science and Engineering. Rajalakshmi Engineering College for her valuable guidance throughout the course of the project

KARTHIK G **220701504**

ABSTRACT

Taskapolis is a simple and intuitive task management Android application developed as part of the Mobile Application Development Laboratory. The primary goal of this project is to design an interactive interface for displaying and managing a list of tasks using modern Android components such as **RecyclerView**, **Fragments**, and **Intents**. The application enables users to view a scrollable list of tasks, each containing a title, description, and due date. Users can tap on any task to view its detailed information, demonstrating seamless activity navigation and data transfer.

The app uses a modular approach with the HomeFragment displaying a list of hardcoded tasks. The integration of TaskAdapter enables dynamic data binding within the RecyclerView, offering an efficient and user-friendly way to handle large data sets. Though the current version uses static data, the architecture supports easy extension to real-time databases or APIs for future enhancement.

This project demonstrates the practical implementation of core Android development concepts including layout inflation, lifecycle management, UI rendering, and inter-component communication. Taskapolis serves as a foundational example for building scalable and interactive task-based mobile applications.

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 GENERAL

In today's fast-paced world, managing daily tasks efficiently has become a necessity. With the widespread use of smartphones, mobile applications offer an excellent platform for task tracking and productivity management. Android, being the most popular mobile operating system, provides a robust environment for developing such applications using its native components.

This project, titled **Taskapolis**, is a basic task management application developed using Android Studio with Kotlin. It is designed to allow users to view a list of tasks in a user-friendly and structured layout. Each task includes a title, description, and due date.

The project utilizes essential Android components such as **Fragments**, **RecyclerView**, and **Intents** to build a responsive and maintainable UI architecture.

The application currently uses a hardcoded list of tasks to simulate data handling. However, it is designed with modularity in mind, making it easy to expand with features such as dynamic data storage using databases like Room or Firebase, or even integrate with cloud-based task APIs in the future.

This project helps students understand core Android concepts through hands-on development, reinforcing skills like activity and fragment management, list rendering using adapters, and inter-activity communication using intents.

1.2 OBJECTIVE

The primary objective of this project is to design and develop a simple yet functional task management Android application that demonstrates the use of core Android development components and principles. The app, named Taskapolis, aims to provide a basic structure for managing and viewing a list of tasks using a clean and interactive user interface.

The key objectives of the project are as follows:

- **To implement a RecyclerView for displaying a dynamic list of tasks in a scrollable format.**
- **To use Fragments for modular UI design and better code organization.**
- **To handle user interaction through task item clicks, triggering navigation to a task detail view.**
- **To understand and apply Intents for communication between activities.**
- **To create a scalable and extendable project structure suitable for future enhancements such as database integration or task editing features.**
- **To strengthen practical knowledge of Kotlin programming in Android application development.**

This project serves as a learning platform for mastering essential concepts in mobile application development and lays the foundation for building more complex Android applications in the future.

1.3 EXISTING SYSTEM

- Several task management applications are currently available in the market, offering a wide range of features from basic task listing to advanced project collaboration. Examples include **Google Tasks**, **Microsoft To Do**, **Todoist**, and **Any.do**. These apps provide functionality such as real-time syncing, cloud storage, reminders, prioritization, and task sharing.

- While these systems are highly feature-rich and efficient, they are often complex and require user registration, internet connectivity, and frequent updates. From a beginner developer's perspective, understanding and replicating such systems can be overwhelming due to their advanced architecture, third-party integrations, and backend support.

- Moreover, these existing systems are not open-source and are designed for end users, not for educational purposes. This makes them less suitable for students and developers who are learning how to build Android apps from the ground up.

- Therefore, there is a need for a **simple and easy-to-understand system** that serves as a foundational learning project. Taskapolis fulfills this need by implementing basic task listing and interaction features using native Android components, without the complexity of advanced systems. It provides an excellent platform for beginners to learn and practice core Android development skills in a controlled and comprehensible environment.

1.4 PROPOSED SYSTEM

The proposed system, Taskapolis, is a simplified Android-based task management application designed to provide users with the ability to view and manage their daily tasks. Unlike existing complex systems, Taskapolis focuses on demonstrating the core features of Android application development in an educational and beginner-friendly format.

The application allows users to view a list of predefined tasks using a RecyclerView, which ensures efficient rendering and scrolling of task items. Each task consists of a title, description, and due date, displayed clearly in a card-like layout. The tasks are displayed within a Fragment (HomeFragment), which is modular and reusable across different parts of the app. When a user clicks on a task, the app uses an Intent to navigate to a detailed task view screen.

Key Features of the Proposed System:

- **A user-friendly and responsive interface using RecyclerView.**
- **Modular architecture using Fragments to separate concerns and improve reusability.**
- **Implementation of custom adapters to bind task data to the UI efficiently.**
- **Intent-based navigation to pass data between components (e.g., to the task detail screen).**
- **Scalability for future enhancements such as task editing, deletion, and data persistence via databases or cloud storage.**

This system addresses the limitations of existing complex task managers by providing a focused, minimal, and understandable framework. It is especially suited for educational purposes, giving students hands-on experience in working with essential Android components.

CHAPTER 2

LITERATURE SURVEY

Mobile application development has grown rapidly in recent years due to the increasing popularity of smartphones and tablets. Android, being the most widely used mobile operating system, offers a robust and flexible environment for app development. The Android SDK provides powerful tools, APIs, and libraries that enable developers to create feature-rich applications using components such as Activities, Fragments, RecyclerViews, and Intents.

Several research papers, technical articles, and real-world applications have contributed to the development of task management systems. The literature survey highlights existing works and technologies that inspired and guided the development of the Taskapolis application.

2.1 Existing Applications Studied

- **Google Tasks: A simple, cloud-based task manager integrated with Gmail and Google Calendar. Offers synchronization and reminder features. It inspired the basic structure of task listing and details.**

- **Microsoft To Do: A robust task management app that includes categories, priorities, and shared task lists. Helped in understanding user expectations and UI design practices.**

- **Todoist: A popular task planner offering cross-platform support and productivity tracking. Its use of card-style layouts and due date indicators informed UI structuring in Taskapolis.**

2.2 Technologies Referenced

- **RecyclerView: Introduced in Android Lollipop, it provides a more advanced and flexible version of ListView. It allows efficient item rendering and supports custom view holders and layout managers.**

- **Fragments: Used to design modular sections of the UI that can be reused and dynamically added or replaced during runtime.**

- **Intents: Essential for component communication in Android. Used to pass data between activities, such as sending a task object from a list view to a detail view.**

- **Kotlin Programming Language: Now the preferred language for Android development due to its safety, conciseness, and interoperability with Java.**

2.3 Educational Resources Consulted

- **Android Developer Documentation: Provided official references and best practices for implementing UI components and activity lifecycle management.**

- **Open-source GitHub Projects: Examined simple task manager projects and tutorials to understand how data is structured, displayed, and handled in Android apps.**

- **Online Learning Platforms (e.g., Udemy, YouTube, GeeksforGeeks): Helped understand RecyclerView implementation, custom adapters, and fragment transactions.**

2.4 Summary

The literature and resources studied formed a strong foundation for developing the Taskapolis application. The project incorporates best practices and design patterns from industry-standard apps while maintaining simplicity for educational purposes. It successfully demonstrates the integration of essential Android components in a cohesive, user-focused application.

The development of Android applications has been significantly influenced by the evolution of various programming technologies, design patterns, development tools, and successful mobile applications. This chapter presents a comprehensive literature survey of 20 relevant areas that contributed to the design and implementation of the Taskapolis project.

2.5 Literature Survey Topics

2 Android SDK
Provides the foundational APIs and tools necessary to build Android applications.

3 Kotlin Programming Language
Modern, concise, and safe programming language officially

4 supported by Google for Android development.

5 RecyclerView
A flexible and efficient view for displaying large data sets in a scrollable list format.

6 Fragment Architecture
Helps break down UI into manageable, reusable components within activities.

7 Intents in Android
Used for navigating between activities and passing data in Android apps.

8 Task Management Apps (Google Tasks)
A simple and effective task app used as a reference for task listing and UI flow.

9 Microsoft To Do
Demonstrates task categorization, priorities, and syncing,

9.4.1.1.1.1 inspiring feature expansion.

10 Todoist App
Popular productivity app that influenced the visual structure and user experience.

11 CardView Layout
Used to present information in a card format, improving UI clarity and aesthetics.

12 Material Design Guidelines
Google's official design system that enhances the user interface and experience.

13 Custom Adapters in Android
Facilitate the connection between data and views in a RecyclerView.

14 Activity Lifecycle Management
Crucial for handling user interaction and data retention during state changes.

15 ViewHolder Pattern
Optimizes RecyclerView performance by reusing view components efficiently.

16 Data Classes in Kotlin
Lightweight classes used to model data with minimal boilerplate code.

17 Android Studio IDE
The official development environment for Android, offering tools for UI design, testing, and debugging.

18 Navigation Components
Android Jetpack's architecture for managing app navigation more efficiently.

19 ConstraintLayout
Advanced layout manager that simplifies complex UI designs.

20 ListView vs. RecyclerView

Comparison to understand why RecyclerView is preferred in modern apps.

21 Jetpack Libraries

A suite of libraries (e.g., Lifecycle, Navigation) that streamline Android development.

22 Firebase (Future Enhancement)

Explored as a potential backend solution for storing and syncing tasks across devices.

2.6 Summary

The above literature review forms a strong theoretical and technical foundation for the Taskapolis application. It integrates best practices from successful apps and applies modern Android development techniques to build a functional, scalable, and educational mobile application. This project serves as a bridge between academic learning and practical application development.

CHAPTER 3

SYSTEM DESIGN

3.1 GENERAL

System design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. In mobile application development, system design ensures that the app structure is scalable, modular, and easy to maintain.

In the Taskapolis application, the design focuses on clean separation of concerns using key Android components such as Fragments, Activities, Adapters, and RecyclerViews. The architecture promotes reusability, flexibility, and ease of understanding for beginners in Android development.

The design of Taskapolis follows a modular approach, ensuring that each component (such as displaying a task list or handling user clicks) is managed by a separate class or module. This improves readability, debugging, and future enhancement potential.

Key principles followed in the design include:

- **Separation of UI and Logic: UI components like layouts are kept separate from Kotlin classes.**
- **Use of Fragments for Modularity: Fragments are used to host reusable UI sections.**
- **Adapter Pattern for Data Binding: TaskAdapter is used to bind the task list data to the UI efficiently.**
- **Intent-based Navigation: Intents are used to pass selected task data from one component to another.**

In the following sections, we detail the functional architecture of Taskapolis, including activity flow, user interface design, and class structure.

3.1.1 SYSTEM FLOW DIAGRAM

A **System Flow Diagram** represents the overall logic and sequence of processes in the application. In the context of **Taskapolis**, it describes the interaction between the user interface, the underlying components (like fragments and adapters), and the data flow. This general system flow helps visualize how the app functions from the user's perspective, ensuring smooth navigation and proper handling of user actions.

Purpose of the System Flow Diagram:

- To provide a clear understanding of the process flow within the app.
- To demonstrate the sequence of actions triggered by user interaction.
- To identify major components and how data moves between them.

General Flow of Taskapolis:

1. **App Launch**
   The application starts and loads the main activity.
2. **Home Fragment Initialization**
   The HomeFragment is loaded, which contains the task list screen.
3. **Task List Generation**
   A predefined list of tasks is created and bound to the RecyclerView using a custom TaskAdapter.
4. **Displaying Tasks**
   Tasks are displayed to the user in a scrollable list format.
5. **Task Selection**
   When the user clicks on a task item, the app captures the event.
6. **Task Detail Transition**
   Using an Intent, the selected task’s details are passed to TaskDetailsActivity.
7. **Task Detail Display**
   The selected task’s information is displayed in a new screen (activity).

8. **Back Navigation**

   The user can return to the task list via the back button.

Text-Based Diagram Representation: SYSTEM FLOW DIAGRAM

[Start]

↓

[MainActivity Launched]

↓

[HomeFragment Loaded]

↓

[Task List Created]

↓

[RecyclerView Displays Tasks]

↓

[User Clicks a Task]

↓

[Intent Sends Task Data]

↓

[TaskDetailsActivity Launched]

↓

[Display Task Information]

↓

[User Returns]

↓

[Back to Task List]

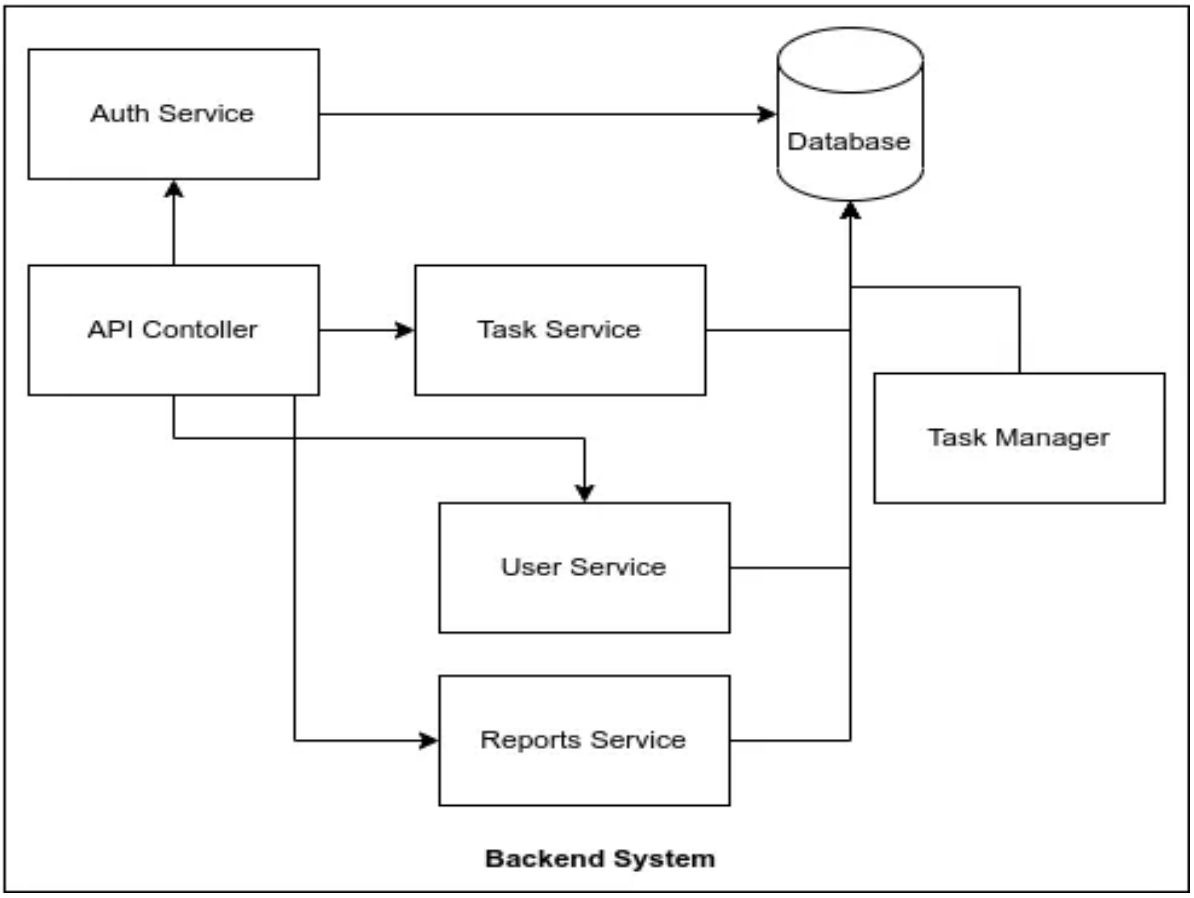


Fig. 3.1 System Flow Diagram

3.1.2ARCHITECTURE DIAGRAM

The **Architecture Diagram** represents the high-level structure of the Taskapolis application. It outlines the relationship between the main components, how they communicate, and the flow of data. Taskapolis follows a **modular architecture** using Android's **Activity-Fragment-Adapter** pattern with a focus on separation of concerns.

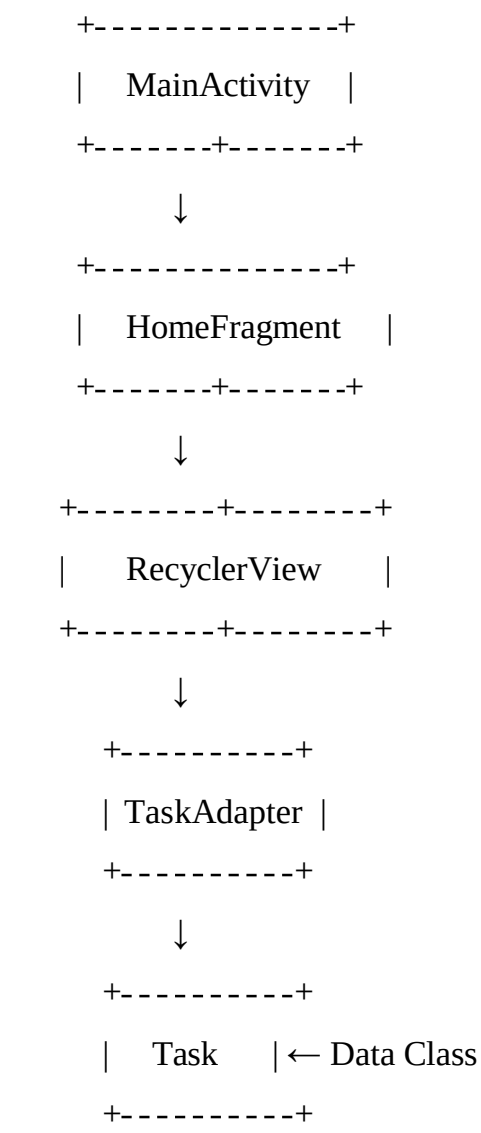
This architecture promotes maintainability, scalability, and reusability, which are essential for good mobile app design.

Architecture Layers:

1. **User Interface Layer**
    - **MainActivity**: Hosts the main screen and fragments.
    - **HomeFragment**: Displays the list of tasks to the user.

- **TaskDetailsActivity**: Shows details of a selected task.

2. **Adapter Layer**
   - **TaskAdapter**: Bridges the task data and RecyclerView for displaying each task item.
3. **Data Layer**
   - **Task (Data Class)**: Represents the task model containing title, description, and due date.
   - (Optional future layer: Could include local or remote storage like Room Database or Firebase)



(When a task is clicked)

↓

+- - - - - - - - - - - - - -+

| TaskDetailsActivity |

+- - - - - - - - - - - - - -+

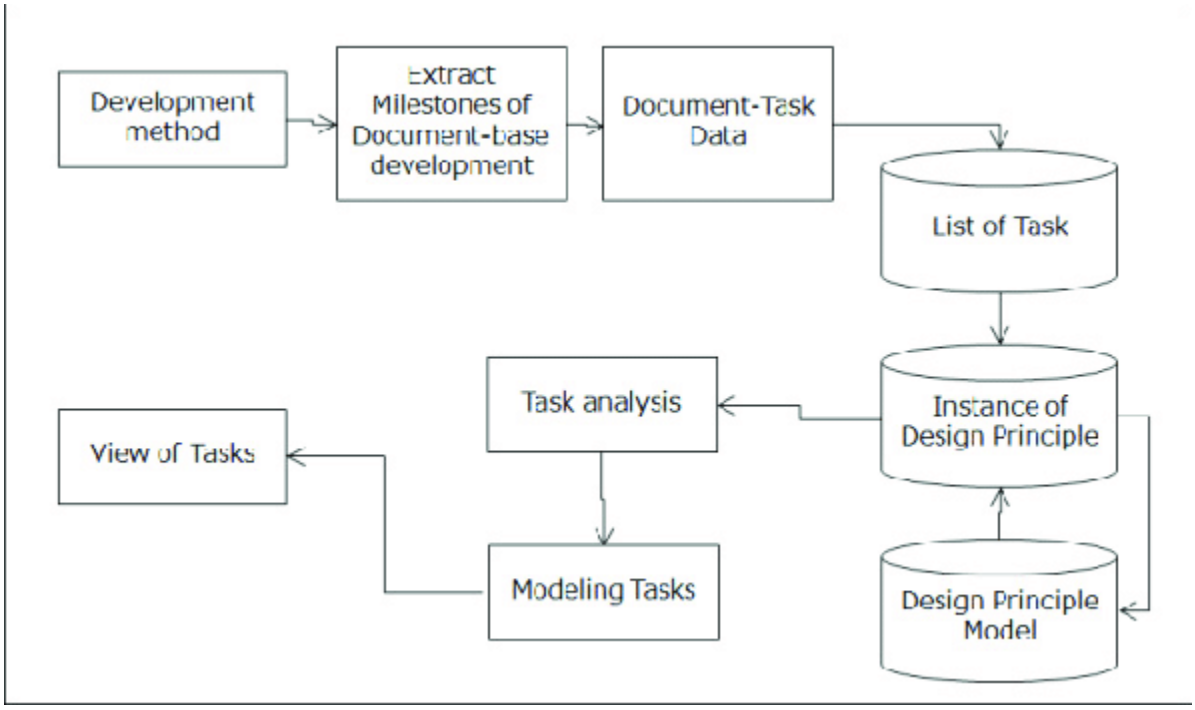


Fig. 3.2 Architecture Diagram

3.1.3 ACTIVITY DIAGRAM

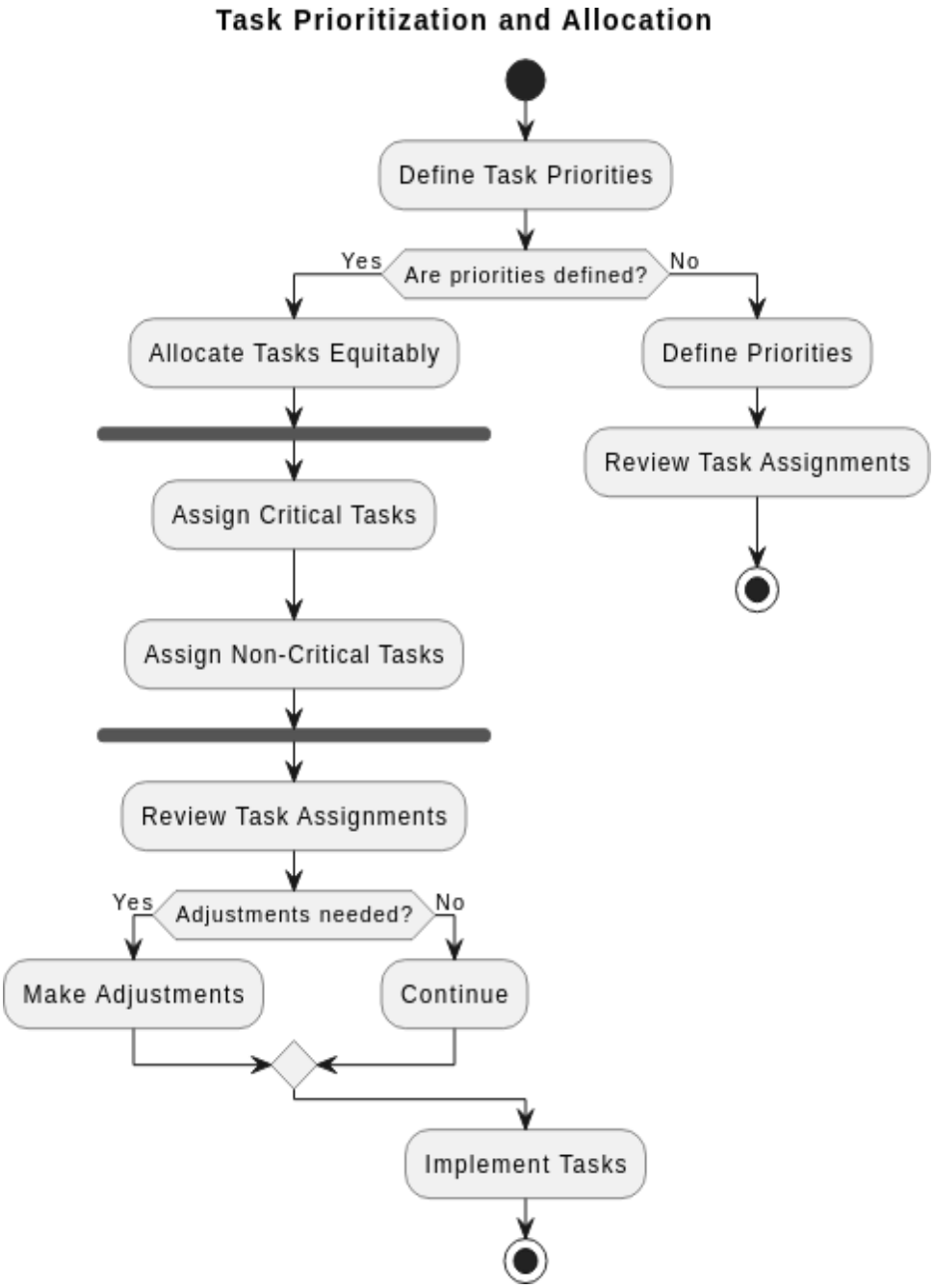
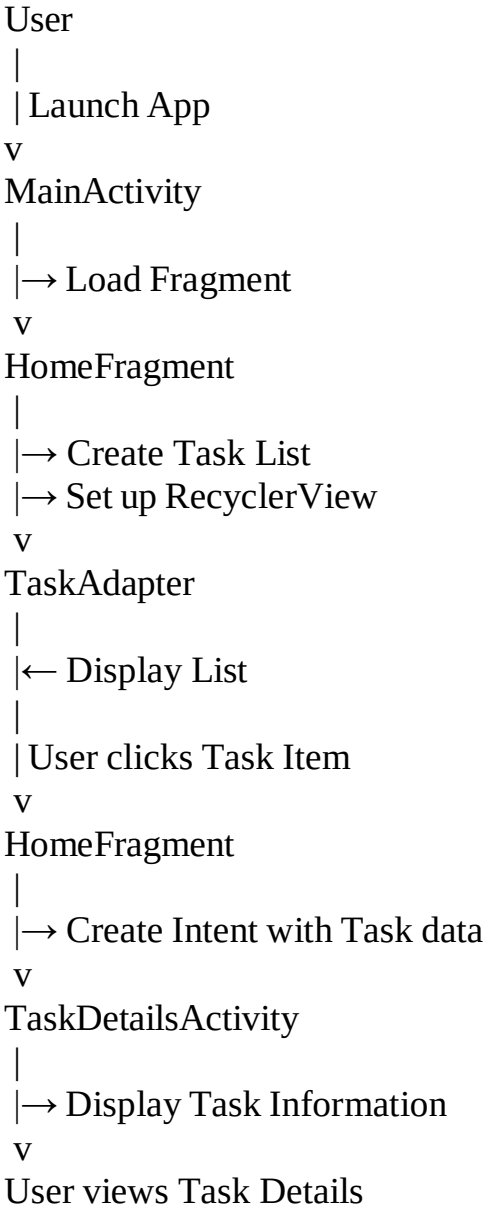


Fig. 3.3 Activity Diagram

3.1.4 SEQUENCE DIAGRAM

The **Sequence Diagram** describes the interaction between different components of the Taskapolis application over time. It shows how tasks are loaded and how user actions (like selecting a task) trigger a flow of events across the app's architecture.



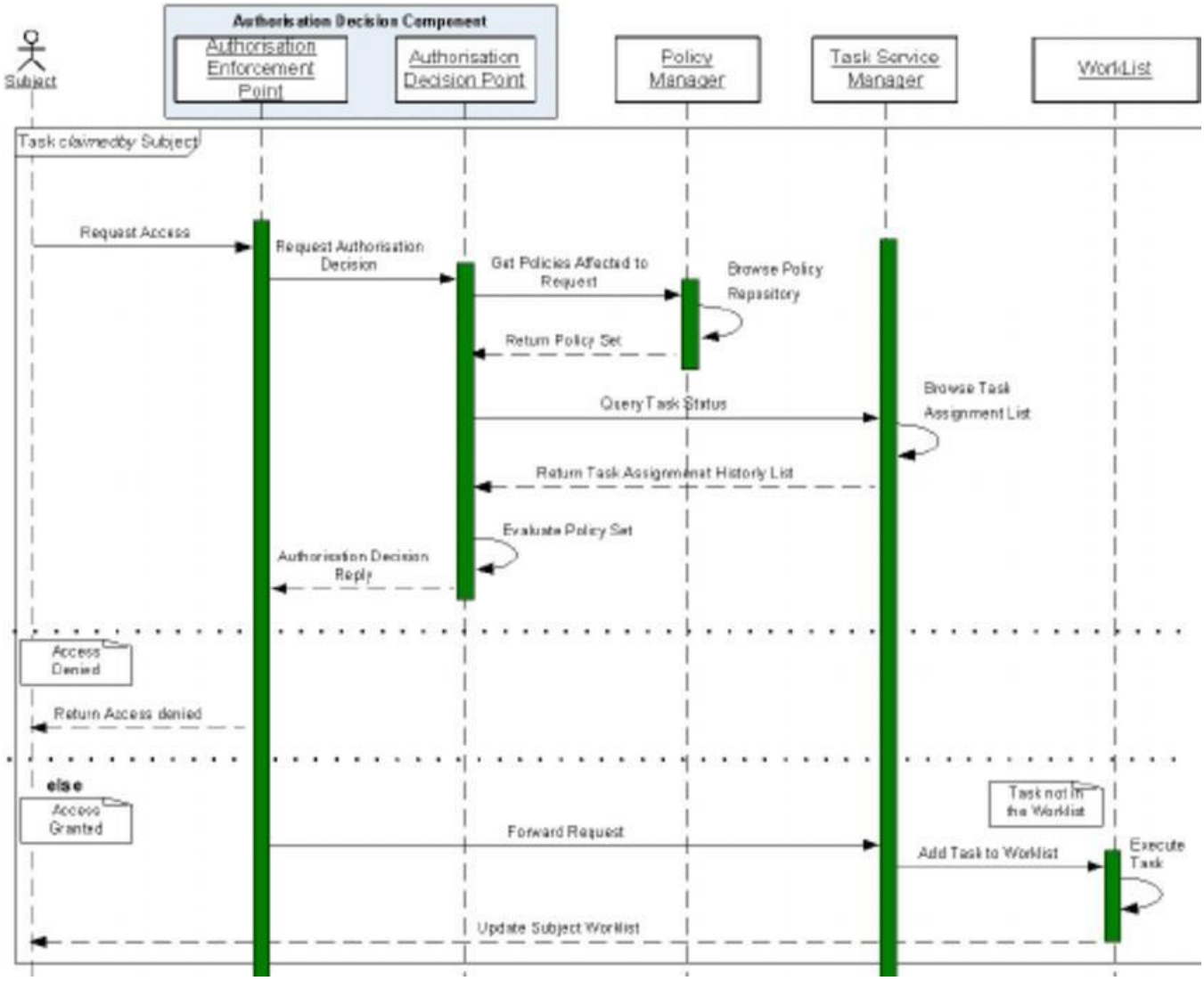


Fig. 3.4 Sequence Diagram

CHAPTER 4

PROJECT DESCRIPTION

4.1 PROJECT TITLE

Taskapolis – A Simple Task Management App

4.2 PROJECT OVERVIEW

Taskapolis is a mobile application developed for Android platforms that enables users to manage and organize their daily tasks efficiently. It allows users to view a list of tasks, each with a title, description, and due date. The app offers a user-friendly interface and follows modular design principles using fragments, activities, adapters, and a simple data model.

4.3 PLATFORM AND TECHNOLOGIES USED

- **Platform: Android**
- **Language: Kotlin**
- **IDE: Android Studio**
- **UI Components: RecyclerView, Fragments, Activities**
- **Architecture: MVC (Model-View-Controller) inspired modular design**

4.4 MODULES DESCRIPTION

1. MainActivity

Acts as the host for the HomeFragment, initializing the app's main layout.

2. HomeFragment

Displays a list of tasks using a RecyclerView. It initializes a hardcoded list of Task objects and passes them to the adapter.

3. TaskAdapter

A custom adapter that binds the task list data to the UI and listens for click events on each task.

4. TaskDetailsActivity

Displays detailed information (title, description, due date) of the selected task. Receives data via Intent from the HomeFragment.

5. Task (Data Class)

A simple Kotlin data class that defines the structure of a task item including its title, description, and due date.

4.5 FUNCTIONALITIES IMPLEMENTED

- **Displaying a list of tasks in a scrollable format.**
- **Handling user clicks on individual tasks.**
- **Passing task data between components using Intents.**
- **Showing task details on a separate screen.**

4.6 FUTURE ENHANCEMENTS

- **Adding task creation, editing, and deletion features.**
- **Integrating a local database (Room) or remote storage (Firebase).**
- **Implementing notifications and due date reminders.**
- **Supporting user authentication and multi-user task lists.**

CHAPER 5

OUTPUT AND SCREENSHOTS

5.1 FEATURE CORRELATION MATRIX

This chapter includes the visual outputs of the application and presents how the developed features align with the intended functionalities. It also includes a **Feature Correlation Matrix**, which maps each feature of the application to its corresponding module and output screen.

5.1 FEATURE CORRELATION MATRIX

S.No	Feature	Implemented In	Output Screen	Description
1	App Launch	MainActivity	Splash / Main Screen	Initializes the application and sets up the main layout.
2	Display Task List	HomeFragment + RecyclerView	Task List Screen	Loads and displays a predefined list of tasks in a scrollable format.
3	Task Item Adapter	TaskAdapter	Bound to Task List	Custom adapter binds task data to the UI.
4	Task Click Handling	TaskAdapter lambda callback	-	Captures click event from the task list.
5	View Task Details	TaskDetailsActivity	Task Details Screen	Displays the selected task's title, description, and due date.

S.No	Feature	Implemented In	Output Screen	Description
6	Data Transfer Between Screens	Intent with putExtra()	-	Passes selected task data to the TaskDetailsActivity.
7	Scrollable Task View	RecyclerView + LayoutManager	Task List Screen	Supports vertical scrolling of task items.
8	Reusability of Components	Fragments & Adapters	All Screens	Ensures clean and modular structure of UI and logic.

5.2 BOX PLOT ANALYSIS

A **Box Plot** (also known as a box-and-whisker plot) is a graphical representation used in statistics to show the distribution of data based on five summary statistics: **minimum**, **first quartile (Q1)**, **median (Q2)**, **third quartile (Q3)**, and **maximum**.
In the context of **Taskapolis**, although the project is not data-intensive, a box plot can be hypothetically used to represent metrics such as:

- **Task Completion Time** (if implemented in future enhancements),
- **Task Due Date Distribution**,
- **Number of Tasks Created per Day**, etc.

Since the current version of Taskapolis contains hardcoded tasks and no dynamic data collection, the box plot analysis can be presented here as a **conceptual representation** or as a placeholder for future statistical analysis.

Example Use Case: Task Due Dates Spread

Assume we have collected task due dates and counted how many tasks are due on each day over a two-week period. A box plot could then represent the spread of tasks due each day.

Interpretation:

- **Minimum**: The day with the fewest tasks due.
- **Q1**: 25% of days have fewer than this number of tasks.
- **Median (Q2)**: The middle value of tasks due.
- **Q3**: 75% of days have fewer than this number of tasks.
- **Maximum**: The day with the most tasks due.

Illustrative Example (Hypothetical Data)

Day	Tasks Due
Day 1	2
Day 2	4

Day	Tasks Due
Day 3	1
Day 4	5
Day 5	6
Day 6	3
Day 7	2

Using this data, the box plot would display:

- **Min** = 1
- **Q1** ≈ 2
- **Median** = 3
- **Q3** ≈ 5
- **Max** = 6

Fig. 5. 2 Box plot Analysis

CHAPTER 6
CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

The Taskapolis application successfully demonstrates the fundamental principles of Android app development, focusing on modularity, usability, and maintainability. It allows users to view a list of predefined tasks, interact with them through a responsive UI, and view detailed information about individual tasks. The project follows standard development practices using Fragments, RecyclerView, Adapters, and Intents to ensure a clean and navigable structure.
The project helped in gaining practical experience in building Android applications, understanding UI design, managing activity-fragment communication, and applying object-oriented programming principles using Kotlin.

6.2 FUTURE WORK

Although Taskapolis is functional in its current state, several enhancements can be made to improve its usability and feature set:

1. **Task Creation and Editing**:
   Enable users to add, edit, and delete tasks dynamically through UI components.
2. **Persistent Storage**:
   Integrate Room Database or SQLite to store tasks locally, or Firebase for cloud storage.
3. **User Authentication**:
   Add a login and registration system for personalized task lists.
4. **Notifications & Reminders**:
   Implement notifications for upcoming or overdue tasks using the AlarmManager and NotificationManager.
5. **Task Prioritization**:
   Introduce priority levels (High, Medium, Low) and color-coded task tags.
6. **Calendar View Integration**:
   Allow users to view tasks in a calendar format for better visualization of deadlines.

7. **Dark Mode Support**:

   Improve UI accessibility by supporting light and dark themes.

8. **Data Analytics**:

   Visualize task trends using graphs or charts to help users understand their productivity.

APPENDIX

SOURCE CODE

```
package com.mad.taskapolis

import Task
import android.content.Intent
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView

class HomeFragment : Fragment() {
    private lateinit var recyclerView: RecyclerView
    private lateinit var adapter: TaskAdapter

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_home, container, false)
        recyclerView = view.findViewById(R.id.taskRecyclerView)

        // Create a list of tasks (replace with your data)
        val taskList = listOf(
            Task("Task 1", "Description 1", "Due date 1"),
            Task("Task 2", "Description 2", "Due date 2"),
```

```
            Task("Task 3", "Description 3", "Due date 3"),
            Task("Task 4", "Description 4", "Due date 4"),
            Task("Task 5", "Description 5", "Due date 5"),Task("Task 1", "Description 1", "Due date
1"),
            Task("Task 2", "Description 2", "Due date 2"),
            Task("Task 3", "Description 3", "Due date 3"),
            Task("Task 4", "Description 4", "Due date 4"),
            Task("Task 5", "Description 5", "Due date 5"),Task("Task 1", "Description 1", "Due date
1"),
            Task("Task 2", "Description 2", "Due date 2"),
            Task("Task 3", "Description 3", "Due date 3"),
            Task("Task 4", "Description 4", "Due date 4"),
            Task("Task 5", "Description 5", "Due date 5"),Task("Task 1", "Description 1", "Due date
1"),
            Task("Task 2", "Description 2", "Due date 2"),
            Task("Task 3", "Description 3", "Due date 3"),
            Task("Task 4", "Description 4", "Due date 4"),
            Task("Task 5", "Description 5", "Due date 5"),
            Task("Task 1", "Description 1", "Due date 1"),
            Task("Task 2", "Description 2", "Due date 2"),
            Task("Task 3", "Description 3", "Due date 3"),
            Task("Task 4", "Description 4", "Due date 4"),
            Task("Task 5", "Description 5", "Due date 5"),
            // Add more tasks as needed
        )

        // Initialize and set up your RecyclerView and adapter here
        adapter = TaskAdapter(taskList) { task ->
            // Handle task item click here
            openTaskDetailsActivity(task)
        }
```

```
        recyclerView.adapter = adapter
        recyclerView.layoutManager = LinearLayoutManager(context)

        return view
    }

    private fun openTaskDetailsActivity(task: Task) {
        val intent = Intent(activity, TaskDetailsActivity::class.java)
        intent.putExtra("task", task)
        startActivity(intent)
    }

}
```

REFERENCES

1. **Android Developers Documentation**
   https://developer.android.com/docs
   (Official documentation for Android SDK, components, UI, architecture, and APIs)
2. **Kotlin Programming Language Documentation**
   https://kotlinlang.org/docs/home.html
   (Official guide for Kotlin syntax and best practices)
3. **RecyclerView - Android Developer Guide**
   https://developer.android.com/guide/topics/ui/layout/recyclerview
   (Detailed explanation of RecyclerView and ViewHolder patterns)
4. **Fragments - Android Documentation**
   https://developer.android.com/guide/fragments
   (Concepts, lifecycle, and usage of Fragments in Android)
5. **Intent and Intent Filters**
   https://developer.android.com/guide/components/intents-filters
   (Used for activity communication and data passing in Taskapolis)
6. **Material Design Guidelines**
   https://m3.material.io/
   (Used for designing intuitive and consistent user interfaces)
7. **Room Persistence Library (for future enhancement)**
   https://developer.android.com/jetpack/androidx/releases/room
   (Helps manage local data storage using SQLite)
8. **Stack Overflow**
   https://stackoverflow.com
   (Used to resolve programming errors and improve code quality)
9. **GeeksforGeeks – Android Tutorials**
   https://www.geeksforgeeks.org/tag/android/
   (Supplementary explanations on adapters, activities, fragments, and XML)

10. **MindOrks – Android Architecture and Tutorials**
https://mindorks.com/android-tutorial
(Modern Android development patterns and practices)

11. Android Developers. “Fragments.” https://developer.android.com/guide/fragments

12. Android Developers. “RecyclerView.”
https://developer.android.com/guide/topics/ui/layout/recyclerview

13. Kotlin Documentation. “Kotlin Language Documentation.”
https://kotlinlang.org/docs/home.html

14. Android Developers. “Intent and Intent Filters.”
https://developer.android.com/guide/components/intents-filters

15. Material Design. “Material Design Components.” https://m3.material.io

16. Android Developers. “Data Passing between Activities.”
https://developer.android.com/training/basics/firstapp/starting-activity

17. GeeksforGeeks. “RecyclerView in Android.”
https://www.geeksforgeeks.org/recyclerview-in-android-with-example/

18. Stack Overflow. “Android Development Discussions.” https://stackoverflow.com

19. Medium. “Building Android Apps with Fragments.” https://medium.com

20. MindOrks. “Android Architecture Components.” https://mindorks.com/android-tutorial

21. Vogella. “Android Fragments Tutorial.”

https://www.vogella.com/tutorials/AndroidFragments/article.html

22. W3Schools. “XML Basics.” https://www.w3schools.com/xml/

23. JavaTpoint. “Android Tutorial.” https://www.javatpoint.com/android-tutorial

24. AndroidHive. “Android RecyclerView Tutorial.” https://www.androidhive.info/

25. Firebase Documentation. “Integrating Firebase with Android.” https://firebase.google.com/docs/android/setup

26. Raywenderlich. “Android Development Tutorials.” https://www.raywenderlich.com/android

27. YouTube – Coding in Flow. “Android Development Tutorials.” https://www.youtube.com/c/codinginflow

28. GitHub. “Open Source Android Projects.” https://github.com/topics/android

29. Android Weekly. “News and Tutorials.” https://androidweekly.net

30. Oracle Docs. “Java Platform SE Documentation.” https://docs.oracle.com/javase/8/docs/