**To-Do List mobile application**

# A MINI PROJECT REPORT

***Submitted by***

**LOGESH D (220701144)**

***in partial fulfillment for the course***

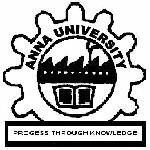
# CS19611 – MOBILE APPLICATION DEVELOPMENT LABORATORY

***of the degree of***

**BACHELOR OF ENGINEERING**

**in**

# COMPUTER SCIENCE AND ENGINEERING



**RAJALAKSHMI ENGINEERING COLLEGE**

**RAJALAKSHMI NAGAR**

**THANDALAM**

**CHENNAI – 602 105**

**MAY 2025**

**RAJALAKSHMI ENGINEERING COLLEGE**

**CHENNAI - 602105**

**BONAFIDE CERTIFICATE**

Certified that this project report **“To-Do List mobile application”** is the bonafide work of **“2116220701519 PONNISH RAJ SP”** who carried out the project work (CS19611-Mobile Application Development Laboratory) under my supervision.

SIGNATURE

Dr. P.Kumar

# HEAD OF THE DEPARTMENT

Professor and Head

Department of

Computer Science and Engineering

Rajalakshmi Engineering College

Rajalakshmi Nagar

Thandalam

Chennai - 602105

SIGNATURE

Dr. V. KARTHICK

# 

AssociateProfessor

Department of

Computer Science and Engineering

Rajalakshmi Engineering College

Rajalakshmi Nagar

Thandalam

Chennai - 602105

# ACKNOWLEDGEMENT

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

Our sincere thanks to **Dr. S.N.Murugesan, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. P.Kumar, M.E., Ph.D.,** Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work. We convey our sincere and deepest gratitude to our internal guide, **Dr. V. KARTHICK.,** Associate Professor, Department of Computer Science and Engineering, Rajalakshmi Engineering College for their valuable guidance throughout the course of the project. We are very glad to thank our Project Coordinator, **Dr. V. KARTHICK.,** Associate Professor , Department of Computer Science and Engineering for his useful tips during our review to build our project.

**LOGESH D (220701144)**

# ABSTRACT

In today’s fast-paced world, managing daily tasks efficiently is crucial for maintaining productivity. The **Compose To-Do** app is a modern, user-friendly task management application built using **Jetpack Compose**, Google’s modern UI toolkit for native Android development. This project aims to provide users with a clean, responsive, and minimalistic interface to organize their personal and professional tasks effortlessly.

The application follows a **modular architecture** with separation of concerns between UI, domain logic, and data layers. It uses **Room Database** for offline data persistence, ensuring users can manage their tasks without needing continuous internet access. Key functionalities include task creation, real-time updates, local notifications, and persistent storage.

Unlike traditional XML-based Android apps, Compose To-Do leverages **Kotlin + Jetpack Compose**, which simplifies UI development by using declarative components. This approach not only enhances performance and flexibility but also reduces boilerplate code, making the app more maintainable and scalable.

The app supports **Google services integration**, allowing secure access and potential for cloud sync features in future updates. With its intuitive interface and lightweight structure, Compose To-Do serves as a perfect example of how modern Android technologies can improve user experience while maintaining technical robustness.

This project demonstrates a practical implementation of modern Android development techniques, combining simplicity, functionality, and clean architecture to address the everyday need for efficient task management.

# TABLE OF CONTENTS

**CHAPTER NO. TITLE PAGE NO.**

**ABSTRACT 4**

**LIST OF FIGURES 6**

1. **INTRODUCTION 7** 
   1. GENERAL 7
   2. OBJECTIVE 8
   3. EXISTING SYSTEM 9
   4. PROPOSED SYSTEM 10
2. **LITERATURE REVIEW 11** 
   1. GENERAL 11
3. **SYSTEM DESIGN 13** 
   1. GENERAL 13
      1. SYSTEM FLOW DIAGRAM 13
      2. ARCHITECTURE DIAGRAM 14
      3. USE CASE DIAGRAM 15
4. **PROJECT DESCRIPTION 16** 
   1. METHODOLOGIE 16
      1. MODULES  **18**
5. **CONCLUSIONS 24** 5.1 GENERAL

**APPENDICES 25**

**REFERENCES 31**

**LIST OF FIGURES**

**FIGURE NO. FIGURE NAME PAGE NO.**

**3.1 SYSTEM FLOW DIAGRAM 13 3.2 ARCHITECTURE DIAGRAM 14 3.3 USE CASE DIAGRAM 15 3.4 TASK DASHBOARD 25 3.5 EDIT TASK 25 3.6 ADD TASK 25 3.7 ADD REMINDER 25 CHAPTER 1**

**INTRODUCTION**

# 1.1 GENERAL

In today’s fast-moving world, people often find it difficult to manage their daily tasks and responsibilities effectively. With so many things to handle—whether it's work, studies, or personal life—individuals need a simple and reliable way to stay organized. This demand has led to the development of various task management tools, among which to-do list applications are the most popular. To address this growing need with modern solutions, the **Compose To-Do** app has been developed using the latest Android technologies.

**Compose To-Do** is a lightweight and efficient task management app built using **Jetpack Compose**, a modern UI toolkit from Google. Unlike traditional Android apps that use XML layouts, this app uses a declarative approach to create responsive and dynamic user interfaces. Users can easily create, update, and delete their tasks through a smooth and intuitive interface. The application uses a clean and modular architecture, separating the UI, domain logic, and data handling, which makes the code easier to maintain and extend. It also uses **Room Database** to store tasks locally, allowing users to access their task list even when offline.

Overall, this project serves as a great example of how modern Android development practices can improve user experience and application performance. The **Compose To-Do** app is not just a productivity tool—it is a digital assistant that helps users gain control over their daily routines. By combining simplicity, functionality, and clean design, the app aims to make task management easy and stress-free for all types of users.

# 1.2 OBJECTIVE

Thanks for the reference format! I understand you want your **Proposed System** section to follow the **same point-by-point structure** as the Pet Care Reminder App example—but tailored to your **Compose To-Do app** project

# 1.2 PROPOSED SYSTEM

The primary objective of the **Compose To-Do** application is to help users manage their daily tasks and schedules efficiently through a modern Android mobile app. The key features of the proposed system are as follows:

1. To develop a **user-friendly mobile application** that allows users to create, view, edit, and delete tasks with ease, using an intuitive and clean interface.
2. To implement a **task organization system** where users can track tasks based on priority, deadlines, or status (e.g., pending, completed).
3. To ensure **offline accessibility** using **Room Database**, so users can manage their to-do list even without an internet connection.
4. To apply a **clean architecture structure** by separating the app into distinct layers: presentation (UI), domain (logic), and data (storage), ensuring better maintainability and scalability.
5. To utilize **Jetpack Compose** for building a responsive and modern UI with less boilerplate code compared to traditional XML-based Android apps.
6. To demonstrate the use of modern Android technologies in solving real-world productivity problems, and to offer users a practical digital solution for managing time effectively.

# 1.3 EXISTING SYSTEM

In the current scenario, most users manage their daily tasks using manual methods or general-purpose applications. These include setting alarms, writing in notebooks, or using built-in calendar apps to keep track of what needs to be done. While these approaches offer basic support, they are not specifically designed for structured task management, which often leads to clutter, confusion, or missed tasks.

Some existing to-do list applications are available in the market, but many of them are either too minimal with limited functionality or too complex with features such as cloud sync, team collaboration, AI integrations, and more. These unnecessary features can overwhelm users who are simply looking for a clean and efficient way to manage personal tasks.

Additionally, many of the existing solutions rely on traditional XML-based UI development, which often results in a rigid, outdated user interface. They may also lack offline support, making them unreliable in situations where internet connectivity is limited. As a result, users may struggle to stay productive and organized using such tools.

Therefore, there is a clear need for a lightweight, modern, and offline-capable task manager built with current technologies. The **Compose To-Do** app aims to fill this gap by offering a simple, responsive, and user-friendly solution using Jetpack Compose and Room Database, tailored specifically for individuals who want to manage their tasks efficiently without any unnecessary complexity.

# 1.4 PROPOSED SYSTEM

The **Compose To-Do** application is a dedicated mobile productivity tool designed to simplify daily task management using modern Android development tools like Jetpack Compose and Room Database. Unlike generic calendar apps or overly complex task managers, this system focuses on delivering a clean, responsive, and user-friendly experience that supports efficient task tracking and organization.

The proposed system allows users to:

1. **Create and Manage Tasks Easily:** Users can add new tasks by entering basic information like task name and optional description. Tasks can be marked as complete or edited at any time, providing flexibility for daily planning.
2. **Organize Tasks by Status:** Users can view their tasks in a clean layout and identify which tasks are pending or completed. This helps users stay focused and prioritize their work based on progress.
3. **Access Tasks Offline:** With **Room Database** integration, users can manage their to-do list even without an internet connection, ensuring continuous access to their tasks in any environment.
4. **Modern UI with Jetpack Compose:** The app features a minimalistic, responsive, and visually appealing interface built using Jetpack Compose, offering a smooth user experience with fewer lines of code and better performance than traditional XML-based designs.

**CHAPTER 2**

# 2.1 LITERATURE SURVEY

Task management applications have become integral to modern productivity, helping users organize, track, and complete daily activities efficiently. With the advancement of mobile technologies, especially in Android development, there has been a significant increase in the development of to-do list and reminder applications. These apps aim to provide users with intuitive and reliable tools to manage personal and professional tasks.

1. Conventional To-Do List Applications: Popular apps like Microsoft To Do, Todoist, and Google Tasks offer comprehensive features such as task creation, categorization, reminders, collaboration, and cloud synchronization. However, many of these apps require account sign-in, are resource-heavy, or include premium features that limit accessibility for basic users.
2. Minimalist Task Apps: Lightweight applications such as Tasks.org and Simple To-Do cater to users seeking a clutter-free interface. These apps focus on simplicity but may lack modern design practices and fluid user experiences provided by newer frameworks like Jetpack Compose.
3. Jetpack Compose in Task Management: Jetpack Compose, Android's modern toolkit for building native UI, allows developers to create dynamic and responsive user interfaces with less boilerplate. Recent task management apps built with Compose offer better performance, customization, and maintainability. The use of Compose is still evolving, and few open-source apps leverage its full capabilities in task tracking.
4. Academic Insights: Research in the field of productivity tools emphasizes the importance of usability, accessibility, and user engagement. Studies suggest that applications with clean UI, offline capability, and minimal distractions lead to higher task completion rates. Yet, many academic prototypes do not focus on modern development frameworks or actual implementation on mobile platforms.

From this review, it is evident that while many solutions exist, there is an opportunity to build a lightweight, Compose-based To-Do application that balances simplicity with modern UI/UX. Such an app can serve users who need an elegant and efficient way to manage tasks without unnecessary complexity.

This project, Compose-ToDo, aims to address this gap by creating an Android application with Jetpack Compose, focused on clean design, essential functionality, and an improved user experience for managing daily tasks.

CHAPTER 3

SYSTEM DESIGN

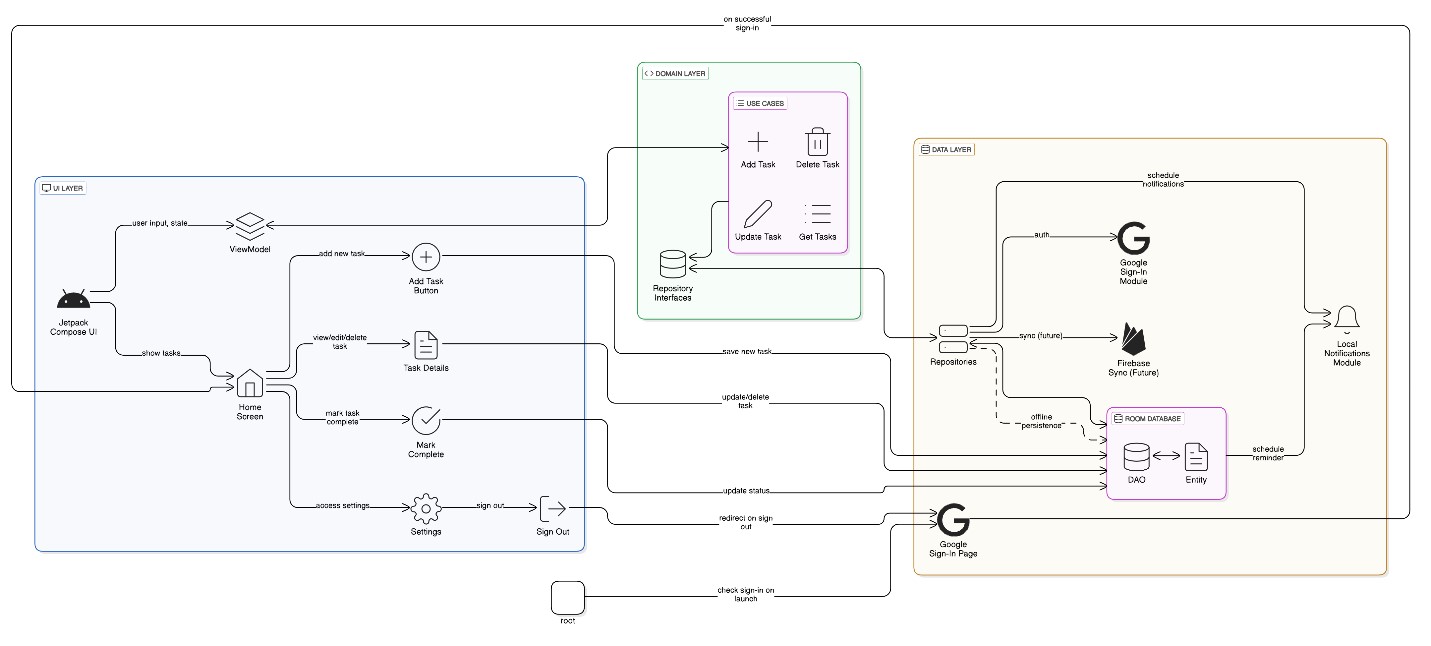
3.1 GENERAL

The system design of the Compose To-Do App focuses on creating a clean, modern, and efficient mobile application that helps users manage their daily tasks with ease. The design approach emphasizes simplicity and smooth navigation, utilizing Jetpack Compose to build a responsive and dynamic user interface. The application consists of essential components such as task creation, task listing, editing, and deletion functionalities. Each component is designed to provide a minimal learning curve while ensuring reliable performance. Data is stored locally on the device using Room Database, allowing users to access and manage tasks offline with high speed and consistency. The system also uses coroutine-based operations to handle background tasks efficiently, maintaining responsiveness without draining device resources.

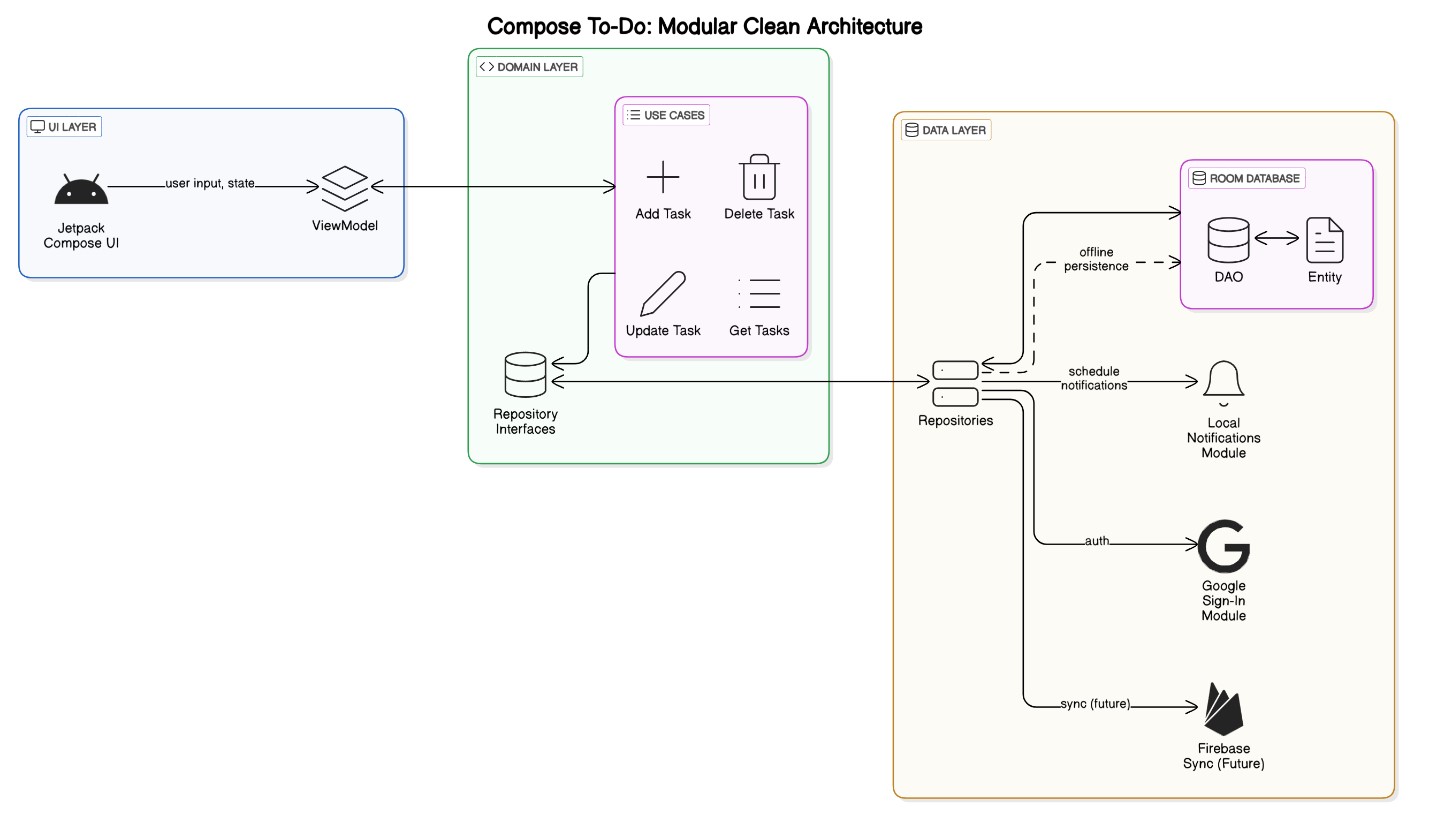
In addition, the system is built with a modular architecture to ensure maintainability and scalability. Each screen or feature is implemented as a composable function, which encourages reusability and code clarity. State management is handled using ViewModels and LiveData/StateFlow to maintain consistency across UI updates. Error handling is incorporated to manage invalid inputs and maintain data integrity.

The app’s lightweight structure ensures that it runs efficiently on a wide range of Android devices. The design also prepares the system for future enhancements such as task reminders, notifications, and cloud sync support. Ultimately, the system design delivers a user-centric experience, making everyday task tracking simple, fast, and reliable.

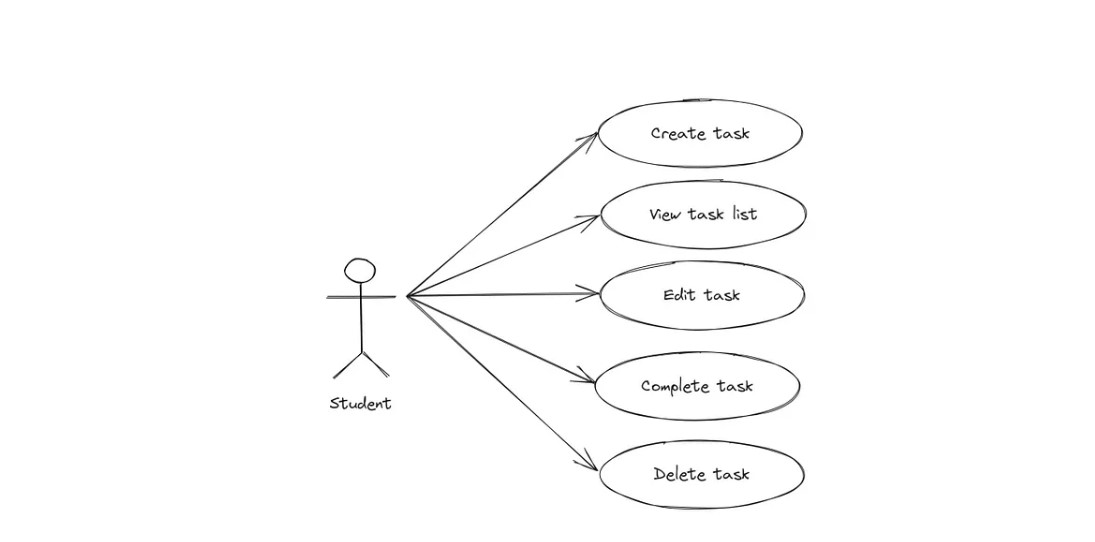
## 3.1.1 SYSTEM FLOW DIAGRAM



## 3.1.2 ARCHITECTURE DIAGRAM



## 3.1.3 USE CASE DIAGRAM



**CHAPTER 4 PROJECT DESCRIPTION**

# 4.1 METHODOLOGY

The development of the **Compose To-Do** app followed a structured Agile Software Development Lifecycle (SDLC) to deliver a clean, efficient, and user-centric task management experience. Agile allowed iterative improvements, continuous testing, and frequent user feedback integration.

## 1. Requirement Gathering and Analysis

Requirements were identified by analyzing common problems faced by users when managing daily tasks. The core needs included:

* Easy-to-use task creation and management
* Offline access to tasks
* Clean and modern UI
* Smooth and responsive performance
* Separation of logic for better scalability

These needs were analyzed and prioritized to define the essential features for the MVP (Minimum Viable Product).

## 2. Design

* **Use Case and UI Design:** Use case diagrams were sketched to define user interaction, and Compose-based UI wireframes were created for a minimal, intuitive design.
* **System Architecture:** The app uses a Clean Architecture model with wellseparated layers: UI (presentation), Domain (logic), and Data (persistence). ● **Technology Stack:**

○ **Frontend:** Jetpack Compose (for reactive and modern UI)

○ **Database:** Room (for local, offline data persistence)

○ **Architecture:** MVVM pattern with dependency injection

## 3. Development

Development was divided into sprints for focused feature implementation:

* **Sprint 1:** Task creation, update, delete, and completion status
* **Sprint 2:** Room database integration and offline task storage
* **Sprint 3:** UI refinement using Compose, task filters, and bug fixes ● **Sprint 4:** Data layer optimization, code cleanup, and testing

Each sprint involved feature reviews, code refactoring, and iterative updates based on feedback.

## 4. Testing

● **Unit Testing:** Core logic such as task CRUD operations and filters were tested individually. ● **Integration Testing:** Verified proper data flow between UI, ViewModel, and Repository. ● **User Testing:** Testers provided feedback on task creation flow, UI intuitiveness, and speed.

## 5. Deployment

The app is deployed as an Android APK and can be uploaded to the Google Play Store. The Room database ensures offline access without a need for server-side deployment.

## 6. Maintenance and Future Enhancements

The app is continuously monitored for performance and user feedback. Planned future features include:

* Cloud backup and Google account sync
* Deadline-based notifications and reminders
* Category-based task grouping
* Dark mode and UI personalization options

## 4.1.1 MODULES

The Compose To-Do application is modular, with each component responsible for a specific core function. This improves maintainability, scalability, and debugging.

Below are the main modules:

### 1. User Interface (UI) Module

* **Purpose:** Presents a clean and intuitive Compose-based interface.
* **Features:**

○ Task list screen

○ Add/Edit task screen

○ Task completion toggle and deletion

● **Tools Used:** Jetpack Compose, Material Design Components

### 2. Task Management Module

* **Purpose:** Manages creation, updating, and deletion of tasks.
* **Features:**

○ Add new task with title and description

○ Edit existing tasks

○ Mark tasks as completed or pending

○ Delete tasks permanently

### 3. Local Data Storage Module

* **Purpose:** Handles persistent storage of tasks offline.
* **Features:**

○ Save tasks locally using Room Database

○ Real-time task list updates

○ Efficient querying for completed/pending task filters ● **Tools Used:** Room, SQLite

### 4. ViewModel and Repository Module

* **Purpose:** Acts as a mediator between UI and data layers.
* **Features:**

○ Stores UI state

○ Provides observable LiveData/StateFlow

○ Handles data from Room through Repository pattern

### 5. Architecture & Dependency Module

* **Purpose:** Ensures clean code separation and scalability.
* **Features:**

○ MVVM architecture

○ Repository abstraction layer

○ Dependency Injection (DI) using Hilt or manual DI (if applicable) **CHAPTER 6**

# CONCLUSION 6.1 GENERAL

The **Compose To-Do** app serves as a practical and efficient solution for individuals seeking to manage their daily tasks in a clean and organized manner. By offering a modern user interface built with Jetpack Compose, along with a robust Room database for offline storage, the application ensures that users can create, track, and manage their to-do items with ease and reliability.

The app was designed with a strong focus on simplicity, performance, and user experience. From adding tasks to marking them as completed, every feature is optimized to support productivity without overwhelming the user. The use of clean architecture and MVVM pattern helps maintain clear separation of concerns, which improves maintainability and scalability for future development.

Through iterative sprints, rigorous testing, and a user-centered approach, the Compose To-Do app successfully meets its core objectives of being fast, lightweight, and user-friendly. It simplifies task organization and encourages better time management among users.

In conclusion, the Compose To-Do app not only achieves its functional goals but also lays the groundwork for future upgrades, such as cloud syncing, notifications, categorization, and enhanced personalization. It stands as a strong example of how modern Android development practices can be used to build meaningful, everyday utility applications.

**APPENDICES**

# APPENDIX A: SOURCE CODE SNIPPETS

## A.1 Add To-Do Task

val task = hashMapOf(

"title" to taskTitle,

"description" to taskDescription,

"timestamp" to System.currentTimeMillis(),

"status" to "pending"

)

FirebaseFirestore.getInstance().collection("users")

.document(FirebaseAuth.getInstance().uid!!)

.collection("tasks")

.add(task)

.addOnSuccessListener {

Toast.makeText(this, "Task added successfully",

Toast.LENGTH\_SHORT).show()

}

.addOnFailureListener {

Toast.makeText(this, "Failed to add task", Toast.LENGTH\_SHORT).show()

}

}

## A.2 Set Task Reminder

val alarmManager = getSystemService(Context.ALARM\_SERVICE) as AlarmManager

val intent = Intent(this, ReminderReceiver::class.java) intent.putExtra("taskTitle", taskTitle)

val pendingIntent = PendingIntent.getBroadcast(this, 0, intent,

PendingIntent.FLAG\_UPDATE\_CURRENT)

val calendar = Calendar.getInstance().apply { set(Calendar.HOUR\_OF\_DAY, hour) set(Calendar.MINUTE, minute)

}

alarmManager.setExact(AlarmManager.RTC\_WAKEUP, calendar.timeInMillis, pendingIntent)

## A.3 BroadcastReceiver for Task Notifications

class ReminderReceiver : BroadcastReceiver() { override fun onReceive(context: Context, intent: Intent) { val taskName = intent.getStringExtra("taskName")

val notificationManager =

context.getSystemService(Context.NOTIFICATION\_SERVICE) as

NotificationManager

val notification = NotificationCompat.Builder(context, "reminder\_channel")

.setContentTitle("Pet Care Reminder")

.setContentText("Time for: $taskName")

.setSmallIcon(R.drawable.ic\_notification)

.setPriority(NotificationCompat.PRIORITY\_HIGH)

.build()

notificationManager.notify(1001, notification)

} }

**A.4 Creating Notification Channel**

**if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.O) { val channel = NotificationChannel( "task\_channel", "Task Notifications", NotificationManager.IMPORTANCE\_HIGH ).apply { description = "Channel for task reminders"**

**}**

**val notificationManager = getSystemService(NotificationManager::class.java) notificationManager.createNotificationChannel(channel)**

**}**

## A.5 Mark Task as Done

val taskRef = FirebaseFirestore.getInstance()

.collection("users")

.document(FirebaseAuth.getInstance().uid!!)

.collection("tasks")

.document(taskId)

taskRef.update("status", "done")

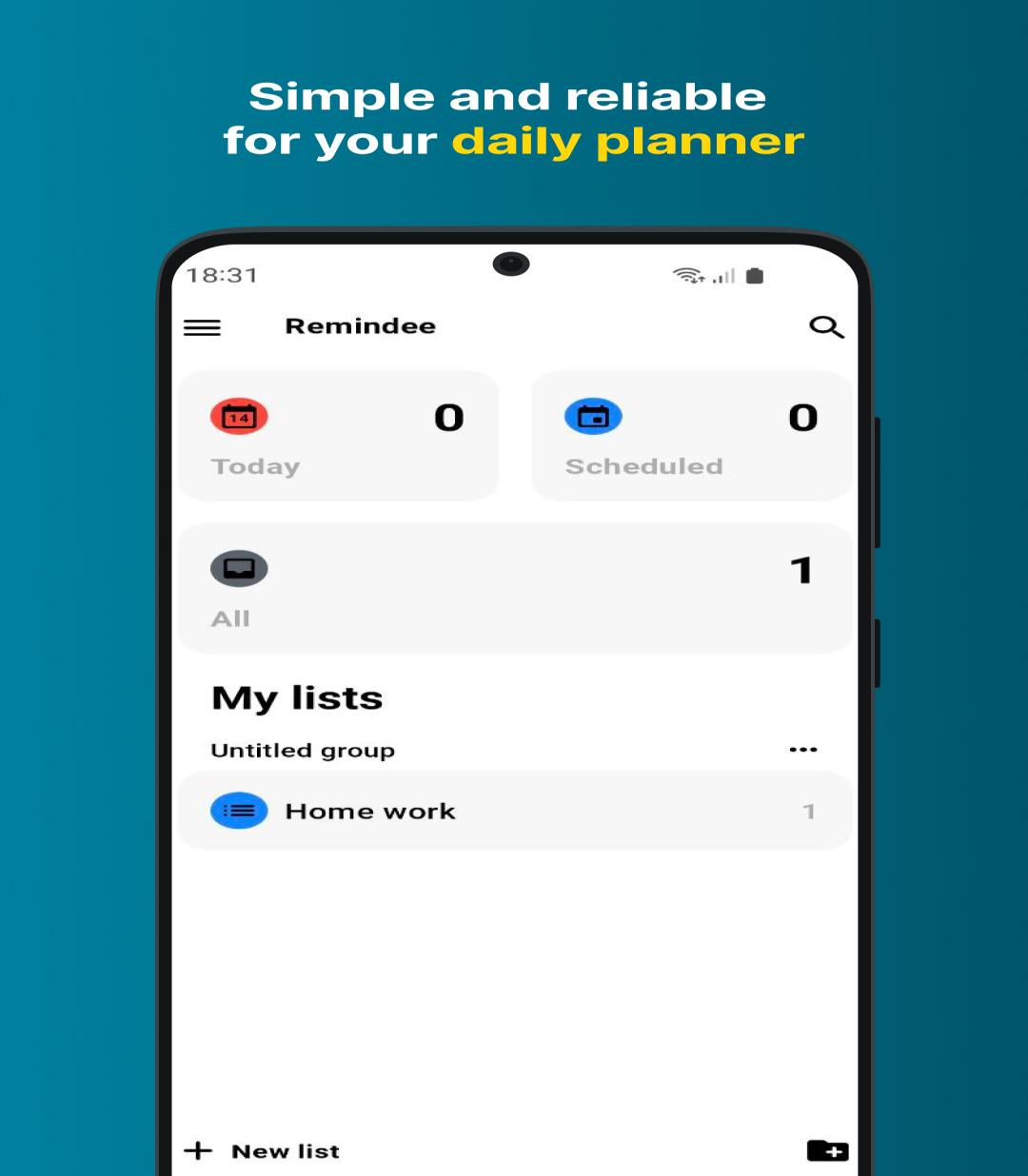
.addOnSuccessListener {

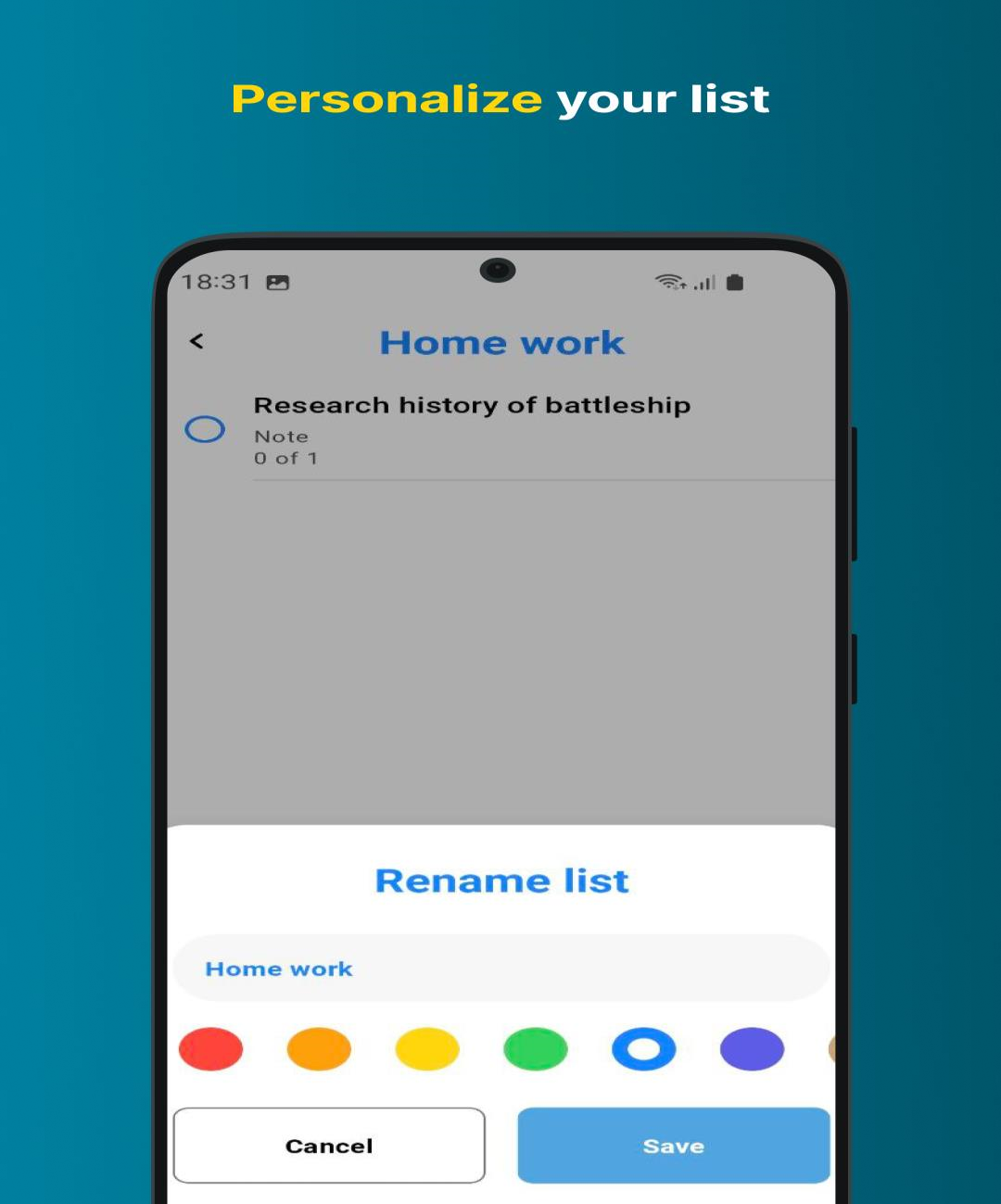
Toast.makeText(this, "Task marked as done",

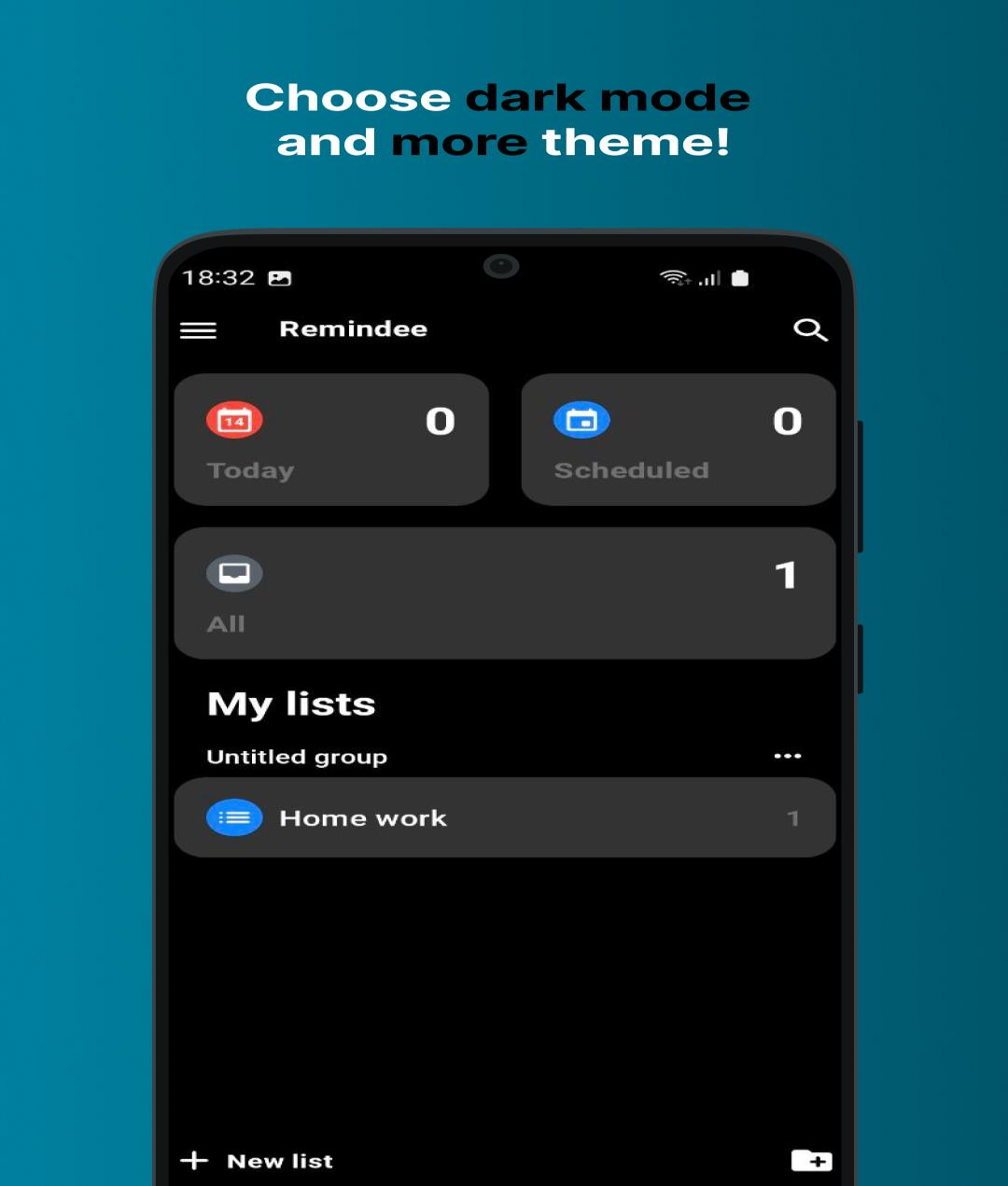
Toast.LENGTH\_SHORT).show()

}

# APPENDIX B: APPLICATION SCREENSHOTS







# REFERENCES

1. Android Developers. (2023). *AlarmManager Class*. Retrieved from<https://developer.android.com/reference/android/app/AlarmManager>

1. Firebase. (2023). *Firebase Authentication Documentation*. Retrieved from https://firebase.google.com/docs/auth

1. Firebase. (2023). *Cloud Firestore Documentation*. Retrieved from https://firebase.google.com/docs/firestore

1. Google Developers. (2023). *Create and Manage Notification Channels*.

Retrieved from<https://developer.android.com/training/notify-user/channels>

1. JetBrains. (2023). *Kotlin Language Documentation*. Retrieved from https://kotlinlang.org/docs/home.html

1. Mayo Clinic Staff. (2022). *Pet Care: Tips for a Healthier Pet*. Retrieved from<https://www.mayoclinic.org/healthy-lifestyle/pet-care>

1. Nielsen, M. (2020). *Designing User Interfaces for Mobile Apps*. UX Collective. Retrieved from [https://uxdesign.cc](https://uxdesign.cc/)