

TASK REMINDER APP

CS19611 – Mobile Application Development Laboratory

Submitted by

SHUBALEKHA G

(2116220701275)

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

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

BONAFIDE CERTIFICATE

Certified that this Project titled “**Task Reminder App**” is the bonafide work of “**SHUBALEKHA G (2116220701275)**” 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. Duraimurugan N., M.Tech., Ph.D.,

SUPERVISOR

Professor

Department of Computer Science
and Engineering,

Rajalakshmi Engineering

College, Chennai-602 105.

Submitted to Project Viva-Voce Examination held on _____

Internal Examiner

External Examiner

ABSTRACT

In today's fast-paced world, managing daily responsibilities and meeting deadlines has become increasingly challenging for individuals across all age groups. Forgetting tasks can lead to decreased productivity, missed opportunities, and elevated stress levels. To address this modern problem, our project titled "**Task Reminder App**" offers a mobile-based solution that helps users stay organized and on track with their commitments.

Task Reminder App is designed to allow users to effortlessly create, manage, and complete tasks by setting timely reminders. The app provides a centralized platform where users can add tasks with titles, descriptions, due dates, and priorities. With intelligent reminder notifications, users receive timely alerts for upcoming tasks, ensuring nothing slips through the cracks. A clean and intuitive interface presents tasks in a categorized list—such as Today's Tasks, Upcoming Tasks, and Completed Tasks—for better clarity and planning.

Built using Android Studio with Kotlin and utilizing Room Database for offline task storage and AlarmManager for scheduling notifications, Task Reminder App is optimized for smooth performance on Android devices. The application promotes efficient time management and boosts productivity by turning mobile phones into personal reminder assistants.

This project not only highlights our technical proficiency in mobile app development but also addresses a practical, real-world need for effective task management in the digital age.

ACKNOWLEDGMENT

Initially we thank the Almighty for being with us through every walk of our life 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 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 Project Coordinator, **Mr. N. DURAI MURUGAN, M.Tech., Ph.D.**, Professor Department of Computer Science and Engineering for his useful tips during our review to build our project and for his valuable guidance throughout the course of the project.

SHUBALEKHA G 2116220701275

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iii
	ACKNOWLEDGMENT	iv
	LIST OF TABLES	viii
	LIST OF FIGURES	ix
	LIST OF ABBREVIATIONS	x
1.	INTRODUCTION	10
	1.1 GENERAL	10
	1.2 OBJECTIVES	10
	1.3 EXISTING SYSTEM	11
2.	LITERATURE SURVEY	11
3.	PROPOSED SYSTEM	13
	3.1 GENERAL	13
	3.2 SYSTEM ARCHITECTURE DIAGRAM	13
	3.3 KEY MODULES	14
	3.4 DEVELOPMENT ENVIRONMENT	15
	3.4.1 HARDWARE REQUIREMENTS	15
	3.4.2 SOFTWARE REQUIREMENTS	16
	3.5 DESIGN THE ENTIRE SYSTEM	17
	3.5.1 ACTIVITY DIAGRAM	17

	3.4.2 DATA FLOW DIAGRAM	18
4.	MODULE DESCRIPTION	19
	4.1 MAIN MODULES	20
	4.1.1 MAIN ACTIVITY	20
	4.1.2 ADD TASK MODULE	20
	4.1.3 REMINDER NOTIFICATION MODULE	21
	4.1.4 TASK LIST VIEW MODULE	21
	4.1.5 COMPLETE/DELETE TASK MODEL	21
	4.2 MODULE INTERACTION	22
5.	IMPLEMENTATIONS AND RESULTS	23
	5.1 IMPLEMENTATION	24
	5.2 DEVELOPMENT ENVIRONMENT	24
	5.3 FUNCTIONALITY IMPLEMENTED	24
	5.4 OUTPUT SCREENSHOTS	25
6.	CONCLUSION AND FUTURE ENHANCEMENT	28
	6.1 CONCLUSION	28
	6.2 FUTURE ENHANCEMENT	28

LIST OF TABLES

TABLE NO	TITLE	PAGE NO
3.1	HARDWARE REQUIREMENTS	15
3.2	SOFTWARE REQUIREMENTS	16

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
3.1	SYSTEM ARCHITECTURE	13
3.2	ACTIVITY DIAGRAM	17
3.3	DFD DIAGRAM	18
5.1	LOGIN SCREEN	25
5.2	HOME SCREEN DISPLAYING TASKS	25
5.3	REMINDER NOTIFICATION POP-UP	26
5.4	TASK COMPLETION CONFIRMATION DIALOG	26
5.5	EDIT/DELETE TASK OPTION AND MESSAGE	27

LIST OF ABBREVIATIONS

Abbreviation	Full Form
UI	User Interface
UX	User Experience
API	Application Programming Interface
IDE	Integrated Development Environment
DB	Database
CRUD	Create, Read, Update, Delete
OTP	One-Time Password
FCM	Firebase Cloud Messaging
DFD	Data Flow Diagram
ERD	Entity Relationship Diagram
XML	Extensible Markup Language
JVM	Java Virtual Machine
SDK	Software Development Kit
MVVM	Model-View-ViewModel
UI/UX	User Interface / User Experience
PDF	Portable Document Format

CHAPTER 1

INTRODUCTION

1.1 GENERAL

In the modern digital era, individuals often juggle multiple responsibilities in both personal and professional spheres. With increasing workloads and distractions, it is common for people to forget important tasks and deadlines, resulting in stress and reduced productivity. A reliable system to plan, manage, and be reminded of daily tasks has become a necessity.

The Task Reminder App is developed as a simple, intuitive mobile application that addresses this problem. It enables users to create, view, and manage daily tasks efficiently while ensuring they receive timely reminders. By providing a user-friendly interface with core features like task categorization, due dates, notifications, and task completion tracking, the app helps users stay organized and focused.

This mobile application aims to promote better time management, increase productivity, and reduce task-related anxiety by serving as a digital assistant that reminds users of their responsibilities proactively. Task Reminder App is designed to be lightweight, responsive, and accessible to all age groups.

1.2 OBJECTIVE

The primary objective of this project is to design and implement a mobile-based reminder application that:

Provides a centralized space to manage and organize daily tasks.

Allows users to add tasks with descriptions, due dates, and priorities.

Sends timely notifications to remind users of upcoming tasks.

Enables editing, deleting, and marking tasks as completed.

Offers a clean and intuitive UI for better user experience and usability.

1.3 EXISTING SYSTEM

Currently, many individuals rely on manual methods such as writing in notebooks, setting alarms, or using basic phone calendar apps to manage tasks. These methods are not always efficient and often lack flexibility, customizability, and reminder accuracy. Some existing task management applications do exist, but they are often cluttered with excessive features, difficult to navigate, or locked behind premium subscriptions. Moreover, most of them do not cater to the needs of users looking for a minimalistic, focused, and easy-to-use reminder tool. This highlights the need for a dedicated Task Reminder App that is simple, accessible, and built specifically for managing and reminding users of their day-to-day responsibilities.

CHAPTER 2 LITERATURE SURVEY

A literature survey is essential to understand the existing approaches and tools used for task management and reminder-based applications. The research highlights that while several productivity tools are available, many of them are either overly complex, lack customization, or do not prioritize timely and accurate reminders. This creates a gap for a lightweight, user-friendly mobile solution tailored for personal task management and daily planning.

2.1 Existing Applications

- **GoogleKeep/MicrosoftToD/AppleReminders**

These are popular task management applications that offer checklist and reminder functionalities. However, they are often tied to specific ecosystems (Android, Microsoft, iOS), and users switching platforms may lose data or encounter compatibility issues. Additionally, they may require cloud sign-ins and are not

always optimized for offline use.

- **Trello/Notion**

These tools are powerful but primarily geared toward project management and collaboration. They are too feature-heavy for users seeking simple task reminders. For personal daily use, they can feel overwhelming and complicated.

- **AlarmApps/CalendarApps**

Many users rely on alarm clocks or calendar apps to remember tasks. These tools are not built specifically for task organization and do not support tracking, categorization, or completion status effectively.

2.2 Limitations of Existing Systems

- Most apps lack a minimal and focused user interface tailored for daily personal task tracking.
- Reminder notifications are either too generic or do not offer flexibility like snoozing, recurring reminders, etc.
- There is often no offline support, with many requiring continuous internet connectivity or cloud sync.
- Many tools don't provide a centralized view of pending, upcoming, and completed tasks.
- Users frequently experience feature overload in apps not designed with simplicity and productivity in mind.

2.3 Need for the Proposed System

The proposed mobile application Task Reminder App addresses these limitations by offering:

A clean and intuitive interface for effortless task creation and organization.

Real-time, customizable reminders using Android's AlarmManager or WorkManager.

Offline-first functionality with local Room database storage.

A focused feature set: add/edit/delete/complete tasks with minimal taps.

Optional enhancements such as priority tagging, task filtering, and light/dark modes.

Task Reminder App is built to serve users who want a simple yet powerful solution to manage their daily activities without distractions, making it a reliable digital assistant for personal productivity.

CHAPTER 3

PROPOSED SYSTEM

3.1 GENERAL

The Task Reminder App is a mobile application designed to help users efficiently manage their daily tasks and receive timely reminders. The proposed system offers a minimal yet functional interface where users can add, view, edit, and delete tasks with ease. By addressing the usability gaps in existing task management solutions, this app provides a smooth user experience that enhances productivity and reduces the mental load of remembering important activities.

Ninaivu focuses on simplicity and performance, making it suitable for users of all age groups. It ensures offline access with local data storage and utilizes system notifications to alert users of upcoming tasks.

3.2 SYSTEM ARCHITECTURE DIAGRAM

The application follows a clean, modular structure built using Kotlin in Android Studio, following MVVM (optional) for future scalability:

SplashActivity: Initial loading screen with app branding.

MainActivity: Displays the list of tasks and includes options to add, edit, or delete tasks.

AddTaskActivity: A form-based interface where users input the task title, description, due date, and time.

ReminderService: A background service or AlarmManager/WorkManager component

that triggers notifications at scheduled times.

TaskRepository: Manages all task-related data using Room Database for local storage. This architecture ensures maintainability, separation of concerns, and smooth performance.

3.3 KEY MODULES

- **Task Management Module**

Users can create new tasks with relevant details like name, due date, and notes. Each task can be edited, deleted, or marked as completed.

- **Reminder Notification Module**

The app schedules reminders using Android's native alarm services. Notifications are triggered based on task due time, keeping the user alerted in real-time.

- **UI and Styling Module**

The interface includes card-style task views, modern material design components, intuitive navigation, color themes, and toast messages for interactivity and feedback.

- **Persistent Local Storage**

Tasks are saved using **Room Database**, allowing offline access and ensuring data is retained between sessions.

- **Future Enhancement Ready**

The system is modular and can easily support advanced features like recurring tasks, categorization (Work/Personal), or cloud sync in future versions.

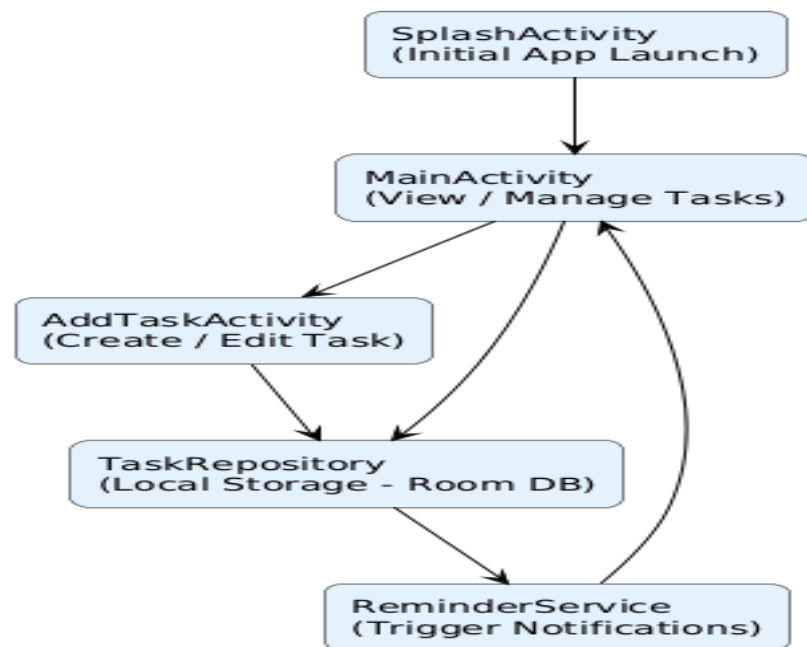


Fig 3.1: System Architecture

3.4 DEVELOPMENTAL ENVIRONMENT

3.4.1 HARDWARE REQUIREMENTS

The successful development and testing of the Task Reminder App depend on compatible hardware that supports Android development environments. Ensuring proper hardware helps maintain optimal performance when compiling, deploying, and debugging the app across emulators and physical devices.

Table 3.1 Hardware Requirements

COMPONENTS	SPECIFICATION
PROCESSOR	Intel i5 or above (recommended)
RAM	8 GB RAM OR Higher
HARD DISK	Minimum 4 GB free space
DISPLAY	1280 x 720 resolution or higher
SMARTPHONE	Android 7.0 (API 24) and above

3.4.2 SOFTWARE REQUIREMENTS

The software environment plays a vital role in the development lifecycle of the mobile application. The Task Reminder app is developed using Android Studio and Kotlin, with room for integration of advanced libraries and features in future iterations. The following tools are used during development and testing:

Table 3.2 Software Requirements

SOFTWARE COMPONENTS	DESCRIPTION
Operating System	Windows 10 / macOS / Linux
IDE	Android Studio (Electric Eel / later)
Programming Language	Kotlin (with Jetpack Compose)
Design Tool	XML for UI (Jetpack Compose UI Toolkit)
Emulator / Device	Android Emulator or physical Android phone
Database (Optional)	Firebase (can be added in future)

3.5 DESIGN OF THE ENTIRE SYSTEM

3.5.1 ACTIVITY DIAGRAM

The Activity Diagram represents the logical flow of the Task Reminder App, from launching the application to creating, viewing, and managing tasks. It illustrates how users interact with the system through different screens and functionalities, giving a high-level overview of user actions and system responses.

Key Activities:

- Launch the application.
- View the splash screen.
- Navigate to MainActivity.

- Perform one of the following actions:
- Add Task
- Enter task details (title, description, date, time).
- Save task.
- View Tasks
- Scroll through the list of pending tasks.
- Delete Task
- Select and confirm task deletion.
- Edit Task
- Modify task details and update.

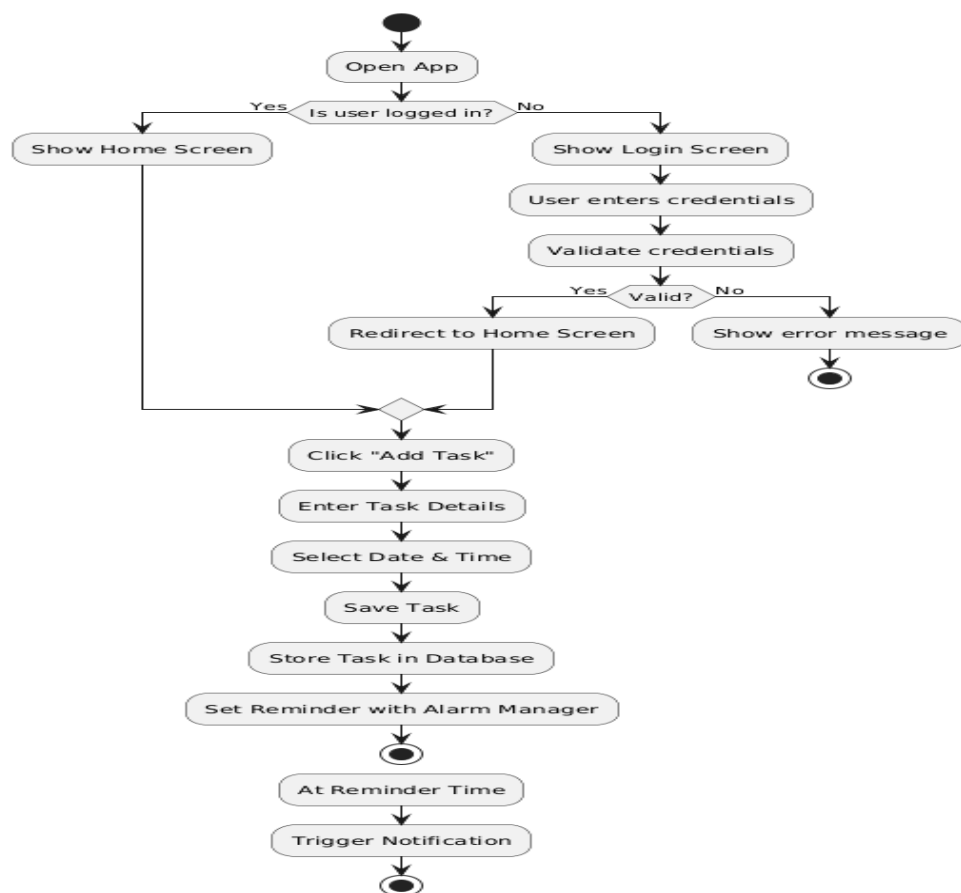


Fig 3.2: Activity Diagram

3.4.2 DATA FLOW DIAGRAM

The Data Flow Diagram (DFD) provides a high-level visualization of how data flows through the Task Reminder App. It identifies the main entities, processes, and data stores involved in task management and reminder notification.

Components:

- **User Input:**
 - The user provides task details including title, description, date, and time.
- **Task Manager Process:**
 - Accepts and validates the task input.
 - Stores tasks locally (e.g. in a local database or in-memory list).
 - Interfaces with the Reminder Scheduler.
- **Reminder Scheduler:**
 - Reads scheduled task times.
 - Sets alarms or local notifications for reminders.
 - Triggers a notification when the reminder time is reached.
- **Local Task Store:**
 - A local storage (e.g. SQLite or in-memory) holding all task data.
 - Allows read/write access for task creation, editing, and deletion.
- **Notification Service:**
 - Listens for scheduled reminders.
 - Displays notifications to the user when due.

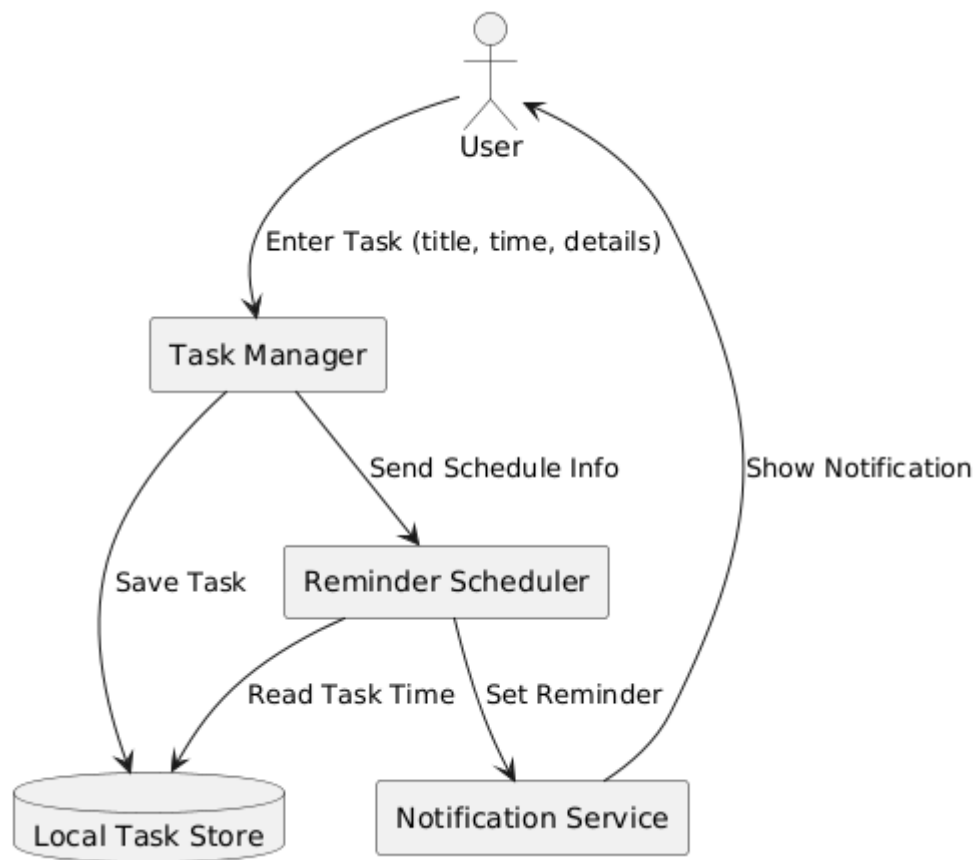


Fig 3.3:Data Flow Diagram

CHAPTER 4

MODULE DESCRIPTION

The Task Reminder App is divided into clearly defined modules that handle distinct responsibilities such as user task input, task management, reminder scheduling, and notification handling. This modular approach ensures maintainability, clarity in functionality, and a responsive user experience.

4.1 MAIN MODULES

4.1.1 MainActivity (Task Dashboard & Entry Point)

This is the main screen that serves as the task dashboard and the entry point of the app.

Features:

Displays a list of existing tasks with time and description.

Offers an "Add Task" button to navigate to the task creation screen.

Allows users to delete or edit existing tasks.

UI Components Used:

RecyclerView / LazyColumn (for task listing)

FloatingActionButton (to add task)

Toasts (for user feedback)

Material Design components (cards, buttons)

4.1.2 AddTaskActivity (Task Creation Screen)

This module is responsible for collecting task details from the user.

Features:

Users can input task title, description, date, and time.

Validates that required fields are not left empty.

Stores task in local database or memory.

Triggers the reminder scheduler for the task.

UI Components Used:

TextInputEditText

DatePickerDialog & TimePickerDialog

Buttons (Save, Cancel)

Toast messages

4.1.3 ReminderScheduler

Handles the scheduling of reminders based on the task's date and time.

Features:

Receives task time from AddTaskActivity.

Sets local alarm/notification using AlarmManager or WorkManager.

Cancels or reschedules reminders when tasks are updated or deleted.

Components Used:

AlarmManager or WorkManager (depending on implementation)

PendingIntent

BroadcastReceiver for trigger events

4.1.4 NotificationReceiver

This module listens for alarm triggers and shows notifications.

Features:

Triggers a local notification when task time is reached.

Provides a concise message with task title and description.

Taps on notification can redirect user to MainActivity.

Components Used:

BroadcastReceiver

NotificationManager

NotificationCompat.Builder

4.1.5 Task Data Model

A Kotlin data class representing each task.

kotlin

CopyEdit

```
data class Task(  
    val id: Int,  
    val title: String,  
    val description: String,  
    val dateTime: Long  
)
```

Purpose:

Maintains consistent structure of task data.

Supports serialization and easy storage.

4.1.6 TaskRepository (Local Storage Handler)

Handles storing, retrieving, updating, and deleting tasks locally.

Purpose:

Acts as a local database interface (can use Room or in-memory list).

Ensures task data is persistent and accessible throughout the app.

Data Structure:

kotlin

CopyEdit

```
object TaskRepository {  
    val taskList = mutableListOf<Task>()  
}
```

4.2 MODULE INTERACTION

- MainActivity displays tasks and links to AddTaskActivity.
- AddTaskActivity captures task data and sends it to TaskRepository and ReminderScheduler.
- ReminderScheduler sets up alarms using Android's system services.
- NotificationReceiver listens for these alarms and shows notifications to the user.
- All modules interact with a shared TaskRepository to maintain consistent task data.

CHAPTER 5

IMPLEMENTATION AND RESULTS

5.1 IMPLEMENTATION

This chapter outlines the implementation process of the Task Reminder App using Android Studio and Kotlin. The app was built with Jetpack Compose for a modern UI experience, enabling fast prototyping, real-time previews, and a clean separation of logic and interface. The goal was to provide users with a simple, efficient way to manage personal tasks and receive timely reminders through notifications.

5.2 DEVELOPMENT ENVIRONMENT

Component	Description
IDE	Android Studio (Hedgehog / Electric Eel)
Language	Kotlin
UI Framework	Jetpack Compose
OS for Development	Windows 10 / macOS
Emulator	Android Emulator (API 30+) or real device

5.2 FUNCTIONALITY IMPLEMENTED:

Task Creation and Validation

Users can input task title, description, date, and time.

Basic input validation ensures no required field is left empty.

Date and time pickers simplify user interaction for scheduling.

Task Management

Main screen displays a scrollable list of all saved tasks.

Each task includes options to edit or delete.

Changes are immediately reflected in the task list.

Tasks are stored in a local repository (memory-based or Room DB optional).

Reminder Scheduling

Upon saving a task, the app schedules a local reminder using `AlarmManager`.

Each task's scheduled time is monitored to trigger a notification.

Alarms are canceled or rescheduled if the task is deleted or modified.

Notifications

At the specified time, the app displays a local notification.

Notifications include task title and description.

Clicking the notification redirects the user back to the app.

User Interface (UI)

Built entirely with Jetpack Compose for reactivity and clean design.

Uses `LazyColumn`, `OutlinedTextField`, and `MaterialButton` for a consistent look.

Floating Action Button (FAB) to add tasks.

`AlertDialogs` for confirmations and user prompts.

UI adapts responsively across different device sizes.

Data Storage

Implemented a shared `TaskRepository` singleton for managing task data across screens.

Tasks are stored in a `mutableListOf<Task>()` structure.

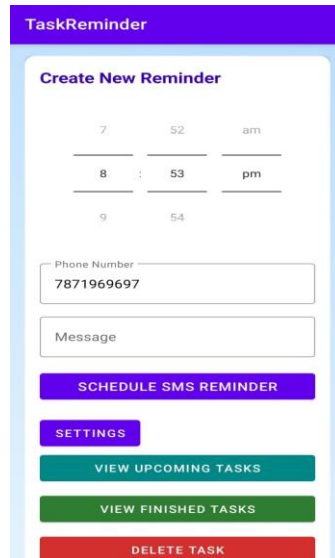
Easily extendable to Room or `SharedPreferences` if persistence is needed.

User Feedback

Toast messages notify users of successful actions like task creation, updates, and deletions.

Errors are displayed with context-sensitive messages (e.g., missing input fields).

5.4 OUTPUT SCREENSHOTS



The screenshot shows the 'TaskReminder' app's login screen. At the top is a purple header with the text 'TaskReminder'. Below it is a white box titled 'Create New Reminder'. Inside this box, there is a time picker with three rows: the first row shows '7', '52', and 'am'; the second row shows '8', '53', and 'pm'; the third row shows '9' and '54'. Below the time picker are two input fields: 'Phone Number' with the value '7871969697' and 'Message'. Below these fields are five buttons: a purple 'SCHEDULE SMS REMINDER' button, a purple 'SETTINGS' button, a teal 'VIEW UPCOMING TASKS' button, a green 'VIEW FINISHED TASKS' button, and a red 'DELETE TASK' button.

Fig 5.1 Login Screen

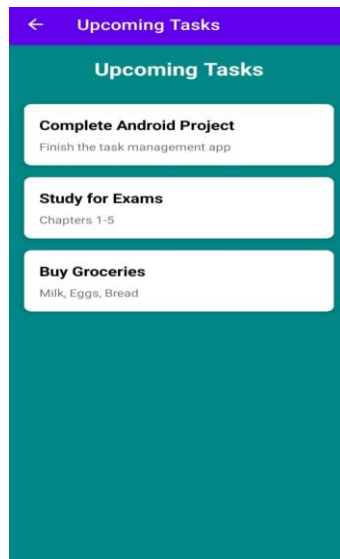
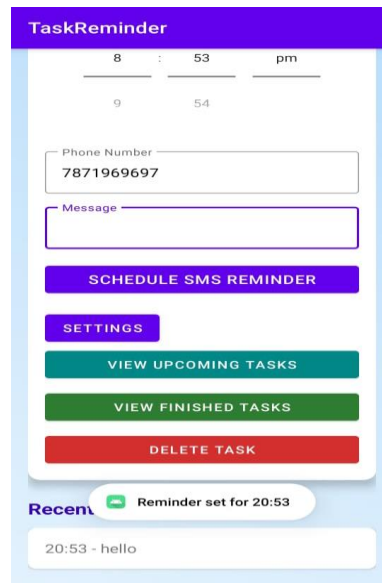
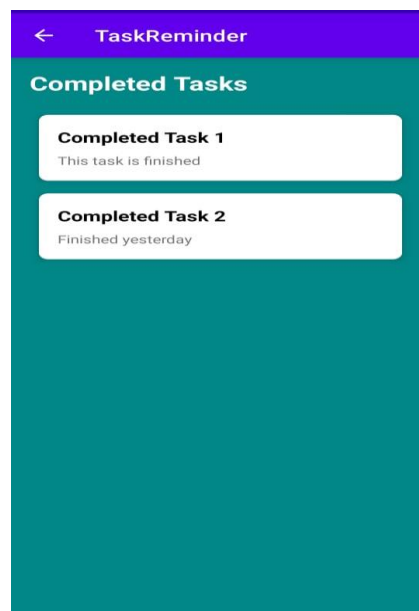


Fig 5.2 Home screen displaying task



The image shows a mobile app interface titled "TaskReminder". At the top, there is a time picker set to 8:53 pm. Below it, a phone number field contains "7871969697" and a message field is empty. There are four buttons: "SCHEDULE SMS REMINDER" (purple), "SETTINGS" (purple), "VIEW UPCOMING TASKS" (teal), and "VIEW FINISHED TASKS" (green). A red button labeled "DELETE TASK" is at the bottom. A notification pop-up at the bottom shows a green checkmark icon and the text "Reminder set for 20:53". Below the notification, a list item shows "20:53 - hello".

Fig 5.3 Reminder notification pop-up



The image shows a mobile app interface titled "TaskReminder" with a back arrow. The screen displays a section titled "Completed Tasks" with a teal background. There are two task entries, each in a white box with a teal border. The first entry is "Completed Task 1" with the subtext "This task is finished". The second entry is "Completed Task 2" with the subtext "Finished yesterday".

Fig 5.4 Task completion confirmation dialog

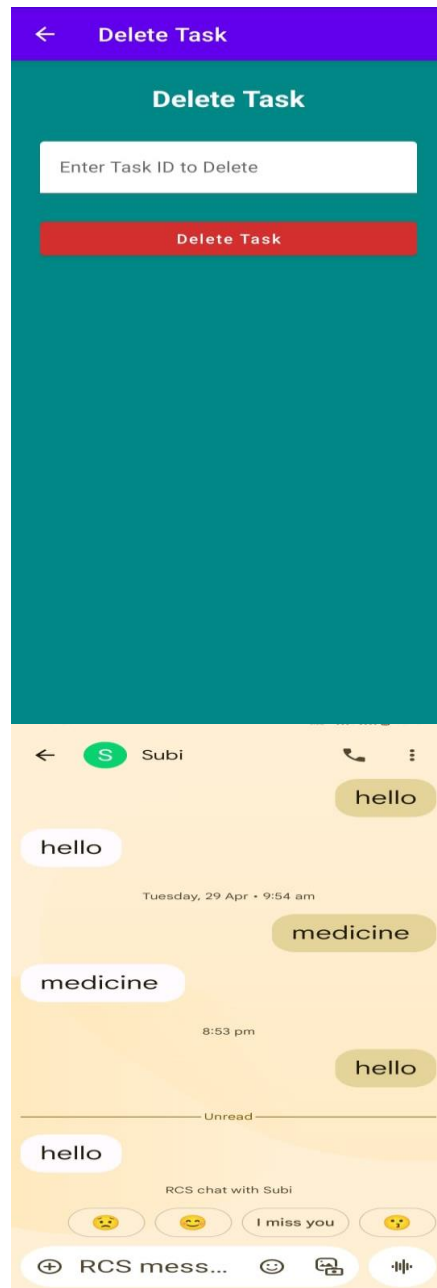


Fig 5.5 Edit/Delete task option and message

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

The Task Reminder App was developed with the goal of helping users manage their daily tasks more efficiently by providing an intuitive interface for creating and scheduling task-based reminders. The project successfully delivers on its core promise by enabling users to:

Create tasks with specific titles, descriptions, dates, and times.

Store and view tasks through a clean, scrollable task dashboard.

Receive timely reminders via notifications without requiring an internet connection.

Benefit from a modern, user-friendly interface built using Jetpack Compose.

Experience responsive task handling with local data storage and reactive UI components.

The app architecture emphasizes modularity and simplicity, making it easy to maintain and expand in the future. Kotlin and Jetpack Compose have been instrumental in delivering a concise and efficient development workflow, with reusable components and state management using `remember` and `mutableStateOf`.

This project served as a practical implementation of mobile development principles and improved the understanding of Android's system services like `AlarmManager` and `NotificationManager`. It reinforces how thoughtful app design can improve user productivity in day-to-day life.

6.2 FUTURE ENHANCEMENT

While the current version of the Task Reminder App meets the basic needs of individual task management, several future features are proposed to expand its functionality, scalability, and user reach:

- Persistent Storage with Room DB:

Implement Room database to save tasks permanently, even after the app is

closed or the device is restarted.

- **Recurring Reminders:**
Allow users to set tasks to repeat daily, weekly, or monthly.
- **Priority & Categorization:**
Enable marking tasks as High, Medium, or Low priority and sorting by categories like Personal, Work, Health, etc.
- **Cloud Backup & Sync:**
Use Firebase or Google Drive to back up tasks and sync them across multiple devices.
- **User Authentication:**
Introduce a simple login system to identify users and protect their personal task data.
- **Dark Mode Support:**
Provide theme customization to enhance visual comfort and accessibility.
- **Voice Input for Task Creation:**
Allow users to quickly add tasks using voice commands or speech-to-text.
- **Smart Suggestions:**
Analyze user behavior over time and suggest task times or categories automatically.
- **Widget Integration:**
Add a home screen widget to quickly view and mark tasks as done without opening the app.
- **Analytics Dashboard:**
Show weekly or monthly task completion stats to help users track their productivity.

