RAJALAKSHMI ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

CS19611 - MOBILE APPLICATION DEVELOPMENT PROJECT REPORT

ALARMIFY

Submitted by

JOHN PRATHAP SINGH S 220701112

in partial fulfilment for the course for 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 titled **"ALARMIFY"** is the bonafide work of **JOHN PRATHAP SINGH S (220701112)**, 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 based on which a degree or award was conferred earlier.

SIGNATURE	**SIGNATURE**
DR. P. KUMAR	**Dr. V. KARTHICK**
Head of the Department	**Assistant Professor**
Computer Science and Engineering	Rajalakshmi Engineering College
Rajalakshmi Engineering College	Chennai - 602105
Chennai – 602105	

Submitted to the Project and Viva Voce Examination for the subject

CS19611 –Mobile Application Development Laboratory held on ________.

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

Initially, we thank the Almighty for being with us through every walk of our lives 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 endeavour 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,** Professor and Head of the Department of Computer Science and Engineering, for his guidance and encouragement throughout the project work. We convey our sincere thanks to our internal guide and Project Coordinator, **Dr. V. Karthick**, Rajalakshmi Engineering College for his valuable guidance throughout the course of the project.

JOHN PRATHAP SINGH S (220701112)

TABLE OF CONTENTS

CHAPTER 1

ABSTRACT

The Android Alarm Application, crafted with Kotlin, is a lightweight and user-friendly mobile solution designed to assist users in setting, managing, and customizing alarms effectively. Developed on the Android platform using Kotlin as the main programming language, the app employs contemporary Android architecture components like ViewModel, LiveData, and Room Database to enhance performance and ease of maintenance.

Key features of the app comprise the ability to set multiple alarms, options for snoozing and dismissing, repeat schedules, customizable alarm tones, and varying vibration patterns. Moreover, the app features a user interface built on Material Design principles, ensuring a seamless and intuitive experience for users.

To guarantee reliability, the application makes use of foreground services and AlarmManager to accurately trigger alarms even when the app is inactive or the device is in sleep mode. Users can easily edit or delete alarms as necessary, and notifications alert them when an alarm becomes active.

This project illustrates a practical application of Kotlin-based Android development, showcasing integration with system services, background execution, and modern UI practices, making it an ideal resource for students and developers looking to learn mobile application development with Kotlin.

CHAPTER 2

INTRODUCTION

2.1 GENERAL

ALARMIFY is a functional and user-friendly mobile application developed to help users set, manage, and customize alarms for their daily routines. Built using Android Studio and Kotlin, the app allows users to easily create multiple alarms with customizable times, labels, tones, and repeat options. It leverages Android components like `AlarmManager`, foreground services, and notification systems to ensure alarms are triggered reliably. The project follows the MVVM architecture for maintainable and scalable code and uses Room Database for storing alarm data locally. Featuring a clean and modern Material Design interface, the app aims to simplify the process of time management and punctuality in users' day-to-day life.

2.2 OBJECTIVE

- To develop a robust mobile application that enables users to efficiently create, manage, and customize alarms.
- To offer a simple, intuitive interface for setting alarms with options such as repeat schedules, tones, and labels.
- To implement local data storage using Room Database, ensuring alarms persist even without internet access.
- To help users manage their time better by providing reliable and timely alarm notifications.

2.3 EXISTING SYSTEM

Many existing alarm applications are either overloaded with unnecessary features or lack essential functionalities such as custom tones, repeat options, or offline support. Some apps rely heavily on internet access or third-party cloud storage, compromising user privacy and convenience. Moreover, many free alarm apps display intrusive ads or require account creation, making the experience less user-friendly. The proposed system addresses these limitations by offering a lightweight, offline-capable alarm app with a clean interface and essential time management features.

CHAPTER 3

Several mobile applications currently exist that provide alarm and time management functionalities, such as **Google Clock**, **Alarmy**, and **Timely Alarm Clock**. These apps help users set multiple alarms, customize tones, and manage their daily schedules. However, many of them exhibit the following limitations:

- Overcomplicated interfaces and settings that overwhelm basic users.
- Frequent in-app advertisements and hidden premium features.
- Dependency on continuous internet connectivity for some advanced functionalities.
- Unreliable alarm triggers in low-power or sleep modes due to background execution limitations.

Research in mobile productivity app development emphasizes the importance of **simplicity**, **reliability**, **offline accessibility**, and **efficient alarm management** to enhance user experience and adoption. Many existing alarm apps either focus heavily on feature-rich customization—making them confusing for casual users—or provide very basic functionality without flexibility or consistency in alarm behavior.

Studies indicate that users prefer alarm applications that offer:

- **Quick alarm setup.**
- **Intuitive scheduling.**
- **Reliable notifications.**
- **Minimal distractions from ads or unnecessary features**.

Additionally, **local data storage** and **privacy** are becoming increasingly important for users who want full control over their settings without the need to sync with cloud-based services.

CHAPTER 4

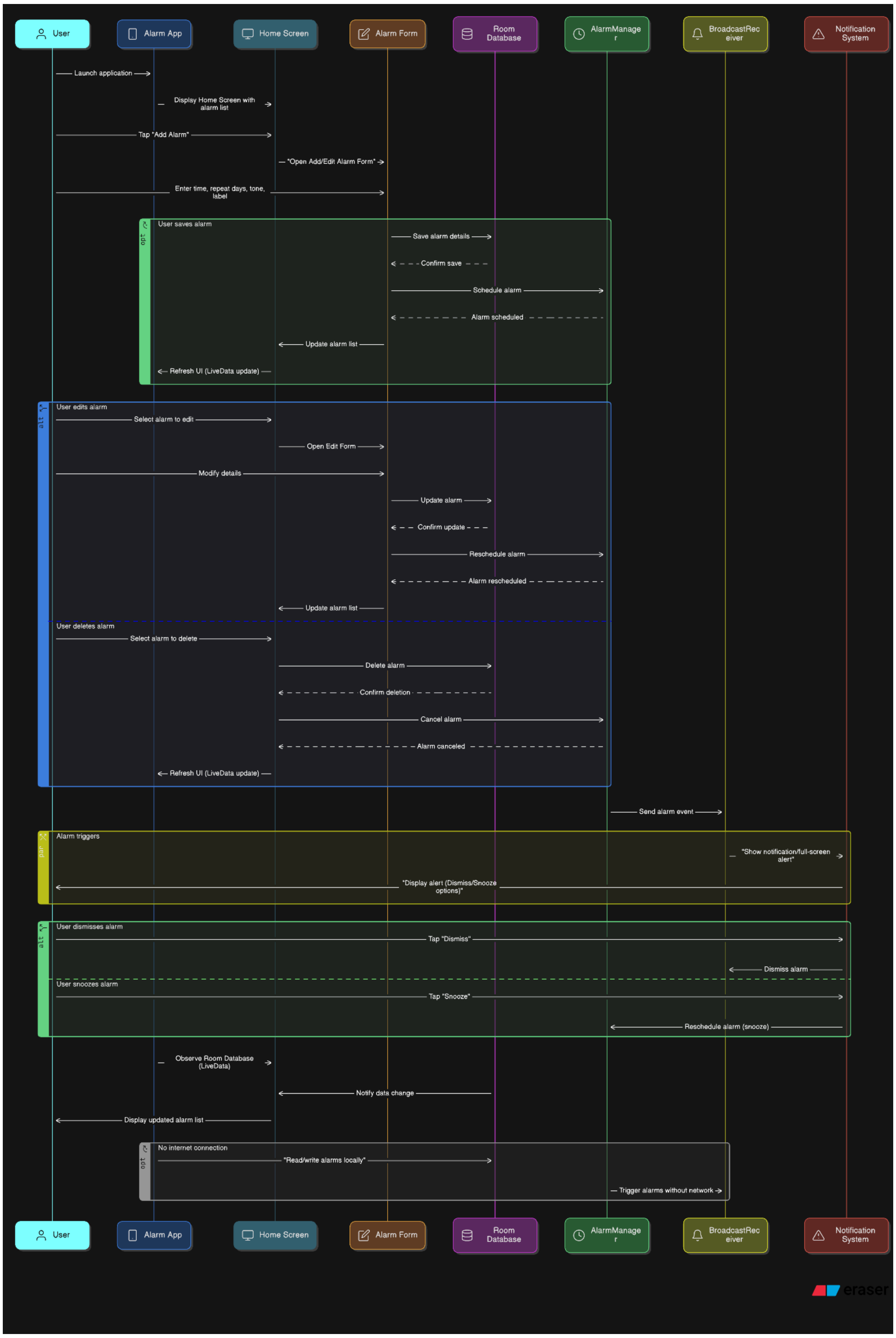
PROPOSED SYSTEM

4.1 SYSTEM OVERVIEW

The **Alarm Application** simplifies time management by providing users with an easy-to-use platform for creating, customizing, and managing alarms. Designed for reliability and user-friendliness, the app features quick alarm setup, repeat scheduling, customizable tones, and offline functionality. Built with Kotlin and Android Studio, the application ensures alarms are triggered accurately using system services like `AlarmManager`, even when the device is idle or the app is closed. With a clean Material Design interface and local storage via Room Database, the app enables users to stay punctual and organized without the need for constant internet access.

4.2 SYSTEM ARCHITECTURE

- The user launches the application and is directed to the **Home Screen**, which displays a list of all scheduled alarms.
- The user can **add, edit, or delete alarms** using interactive forms, specifying details like time, repeat days, tone, and labels.
- Alarm details are stored in the **Room Database** for persistent offline access.
- The system uses **AlarmManager** and **BroadcastReceiver** to schedule alarms and trigger notifications at the specified time, even when the app is not running.
- When an alarm goes off, a **notification or full-screen alert** is displayed with options to **dismiss** or **snooze**.
- Any changes to the alarm list are reflected immediately in the UI, ensuring real-time updates using **LiveData** and the **MVVM architecture**.



(Fig 3.1 System Architecture)

CHAPTER 5

MODULE DESCRIPTION

5.1 MODULES

- **Alarm Management Module**
  Enables users to **add, edit, or delete alarms**, specifying time, repeat days, labels, and tones. Ensures each alarm is uniquely identified and stored locally using Room Database for persistent and reliable access.

- **Alarm Scheduling Module**
  Handles the **accurate scheduling and triggering** of alarms using Android's `AlarmManager` and `PendingIntent`. Ensures alarms function even when the app is closed or the device is idle.

- **Alarm Trigger & Notification Module**
  Uses a `BroadcastReceiver` to **listen for alarm events** and **trigger notifications or full-screen alerts** at the scheduled time, with options like dismiss or snooze.

- **Data Persistence Module**
  Implements **Room Database** to store alarm configurations and user preferences locally, ensuring **offline functionality** and data reliability across sessions.

- **UI/UX Module**
  Provides a **responsive, minimalistic interface** following Material Design principles. Ensures **smooth navigation**, quick alarm setup, and a visually clean experience for users.

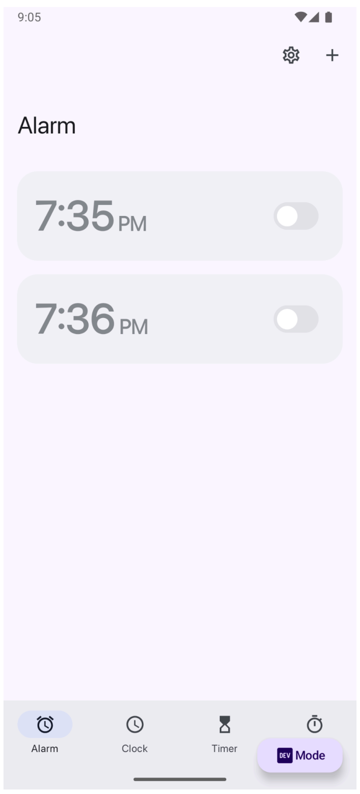
CHAPTER 6

IMPLEMENTATION AND RESULTS

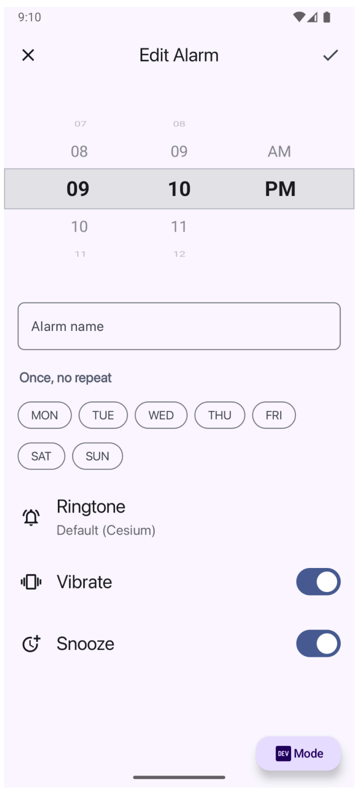
6.1 TOOLS USED

- Android Studio: For building the app.
- Kotlin: For programming the game logic.
- XML: For designing the user interface.
- Room Database: Provides local data storage to save alarm configurations persistently.
- AlarmManager: Android component for scheduling alarms to go off at a specified time.
- SQLite: For storing custom Truths/Dares.

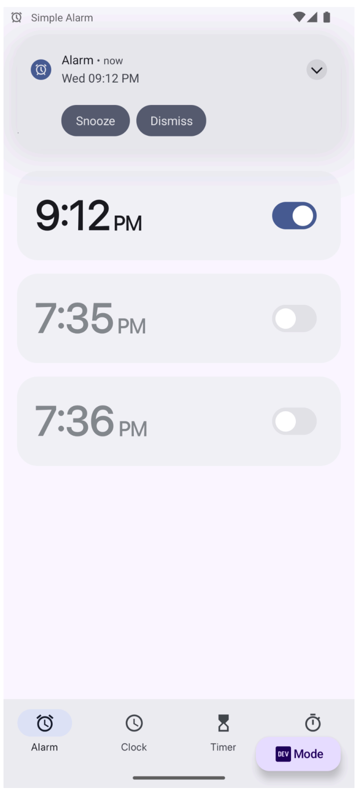
6.2 OUTPUT SCREENSHOTS



(Fig 6.1 App Home Page)



(Fig 6.2 Adding new alarm)



(Fig 6.3 Alarm time is up)

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

The **Alarm Application** offers a simple, efficient, and reliable solution for time management. With features like customizable alarm settings, repeat scheduling, and offline functionality, it ensures users can stay organized and punctual. The clean and intuitive user interface built with Material Design enhances user experience, while robust backend components like `AlarmManager`, `BroadcastReceiver`, and `Room Database` ensure consistent performance. Whether used for daily routines, reminders, or important tasks, this application serves as a dependable tool for users seeking timely notifications and stress-free scheduling.

6.2 FUTURE ENHANCEMENT

•**Recurring Alarms with Smart Labels**: Allow users to set dynamic labels (e.g., “Meeting every Monday”) and recurring rules with more granularity.

Vibration and Snooze Customization: Let users configure vibration patterns and snooze durations.

Voice-Control Integration: Enable alarm creation and management through voice commands (e.g., Google Assistant).

Dark Mode Support: Improve visual comfort by offering a night-friendly UI theme.

Cloud Backup & Sync: Allow users to back up and sync alarms across multiple devices using cloud services.

REFERENCES

1) Android Developer Documentation
2) Material Design Guidelines
3) Room Database – Android Jetpack