RAJIV GANDHI INSTITUTE OF TECHNOLOGY

GOVERNMENT ENGINEERING COLLEGE

KOTTAYAM-686 501

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

CSD 334 MINI PROJECT REPORT

AlertMe

Submitted by

Abhishek Raymond (Reg. No: KTE20CS003)
Visakh Vijay O (Reg. No: KTE20CS059)
Abin Augustine (Reg. No: KTE20CS063)

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY
THIRUVANANTHAPURAM

JUNE 2023

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

GOVERNMENT ENGINEERING COLLEGE

KOTTAYAM-686 501

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

Vision of the Department

To be a centre of excellence for nurturing the young minds to become innovative computing professionals for the empowerment of society.

Mission of the Department

- To offer a solid foundation in computing and technology for crafting competent professionals.
- To promote innovative and entrepreneurial skills of students by exposing them to the forefront of developments in the field of computing.
- To inculcate strong ethical values in the young minds to work with commitment for the progress of the nation.

RAJIV GANDHI INSTITUTE OF TECHNOLOGY

GOVERNMENT ENGINEERING COLLEGE

KOTTAYAM-686 501

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that this report entitled ***AlertMe*** *is an authentic report of the project done by the team consisting of* ***ABHISHEK RAYMOND(Reg. No: KTE20CS003), VISAKH VIJAY O(Reg. No: KTE20CS059) and ABIN AUGUSTINE(Reg. No: KTE20CS063)*** *during the academic year* ***2022-23****, in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering of APJ Abdul Kalam Technological University, Thiruvananthapuram.*

GUIDE COORDINATOR

HEAD OF THE DEPARTMENT

Acknowledgement

Every successful project is the outcome of hard work and strong support and guidance given by a number of people. We sincerely appreciate the inspiration, support and guidance of all those people who have been instrumental in making this project a success.

We express our sincere gratitude to **Dr. Jalaja M. J., Former Principal** and **Dr. Prince A., Principal** for making the resources available at the right time without which this project would not have been a success.

We take this opportunity to express a deep sense of gratitude to **Prof. Kavitha N., Associate Professor & Head, Department of Computer Science and Engineering**.

We also take this opportunity to express our profound gratitude and deep regards to our guide **Prof. Vipin Vasu A V, Assistant Professor** for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him from time to time shall carry us a long way in the journey of life on which we are about to embark.

We also express our gratitude to Project Coordinators **Prof. Anil Kumar S** and **Prof. Nisha K.K** for the cordial support, valuable information and guidance, which helped us in completing this task through various stages.

Last, but not least, we thank **almighty, our parents** and **friends** for their constant encouragement without which this project would not have been possible.

Abhishek Raymond
Visakh Vijay O
Abin Augustine

Declaration

We, the undersigned hereby declare that the project report entitled Project **AlertMe**, submitted for partial fulfilment of the requirements for the award of the degree of Bachelor of Technology of the **APJ Abdul Kalam Technological University, Kerala** is a bonafide work done by us under the supervision of **Prof. Vipin Vasu A V**. This submission represents our idea in our own words where ideas or words of others have not been included; we have adequately and accurately cited and referenced the original sources. We have adhered to the ethics of academic honesty and integrity and have not misrepresented or fabricated any data, idea or on our submission. We understand that any violation of the above can result in disciplinary actions from the institute and/ or University and can evoke penal action from the sources which have not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed as the basis for awarding any degree, diploma or similar title of any other university.

Name of Students: Signatures:

ABHISHEK RAYMOND (Reg. No.: KTE20CS003)

VISAKH VIJAY O (Reg. No.: KTE20CS059)

ABIN AUGUSTINE (Reg. No.: KTE20CS063)

Batch: 2020 - 2024 November 18, 2023

Abstract

AlertMe is a user-friendly android application that enables swift emergency communication. With a simple tap on the panic button, the app sends real-time alerts to nearby users and designated emergency contacts during emergencies. The app will allow users to create an emergency profile, which will include their personal information, emergency contacts, and medical information. In the event of an emergency, the user will activate the alert system, which will send a notification to nearby users and emergency contacts.

The app's primary focus is on providing crucial location information and medical information to responders, allowing them to offer timely assistance. Leveraging geolocation services, **AlertMe** connects users with nearby potential helpers, enhancing the chances of quick response during critical situations.

In addition to notifying nearby users, **AlertMe** also ensures that registered emergency contacts receive prompt notifications. This feature helps users alert their close acquaintances, allowing for a wider support network during emergencies.

The potential impact of **AlertMe** is significant, as it has the potential to save lives in emergency situations. This project represents an important contribution to the field of emergency alert applications and can have a significant positive impact on society.

Contents

List of Figures

List of Tables

List of Algorithms

List of Symbols and Abbreviations

API	Application Programming Interface
FCM	Firebase Cloud Messaging
UI	User Interface
SDK	Software Development Kit
JWT	JSON Web Tokens
JSON	JavaScript Object Notation
OTP	One Time Password
SMS	Short Message Service
TLS	Transport Layer Securitys

Chapter 1

Introduction

AlertMe is an Android application designed to provide swift and effective emergency communication. During critical situations, the app sends real-time emergency alerts to nearby users and emergency contacts. With a simple tap on the panic button, users can trigger the system, which dispatches notifications containing location information and other essential details. This seamless process ensures timely assistance when it matters most. **AlertMe** aims to enhance user safety and facilitate quick responses during emergencies, making it a crucial tool in times of need.

1.1 Objectives

This project aims to achieve the following objectives.

1. Enhance user safety by providing a reliable emergency alert system for quick distress notifications to nearby app users and emergency contacts.
2. Empower users with a sense of security and peace of mind, knowing they have a powerful tool at their fingertips to seek help and assistance whenever and wherever they need it.

1.2 Motivation

The facts that motivated the choice of the project are the following.

1. Develop an emergency alert application that can help save lives.
2. Provide a fast and efficient way for individuals to alert nearby individuals and emergency contacts in case of an emergency.
3. Promote a helpful mindset to positively impact society. Encourage a culture of support and compassion in the time of an emergency.

1.3 Scope of the Project

- Need: Create a reliable platform for users to send alert notifications to nearby app users and emergency contacts during critical situations.
- Deliverables: A fully functional emergency alert app for Android, featuring panic button, user authentication, location integration, and notification system.
- Exclusions: Web-based version, physical hardware devices, real-time location, voice-based emergency call system, and integration with third-party emergency services are not included in the scope.
- Assumptions: Users have smartphones with Android, stable internet connection, and provide accurate information while using the app.

1.4 Prerequisites for the Reader

To understand the contents of this report, the reader is expected to meet the following prerequisites:

- Basic familiarity with app development using the Flutter framework and Dart programming language.
- Basic understanding of backend technologies like Node.js and the MongoDB database, which are utilized for data management.
- Basic familiarity with mobile applications and their functionalities.

1.5 Organization of the Report

Chapter 2 deals with the Feasibility Study (section 2.4.1), explores the Existing Systems (section 2.1), continues to the Gap Analysis (section 2.2) and then proceeds to the Proposed System (section 2.3), and the Requirements Engineering(section 2.4). Chapter 3 deals with the design of the project. The System Design (section 3.1) which provides a basic overview of different modules in the system in sections 3.1.1, 3.1.2 and 3.1.3. In section 3.2 of this chapter, we have the detailed design, where the user module design in section 3.2.1, algorithms related to the alert sending module in section 3.2.2 and algorithm of the alert receiving module (section 3.2.3) are given. Chapter 4 explores the Tools Used which are explained in section 4.1 and the Implementation of Modules that are shown in section 4.2 (module 1 in 4.2.1, module 2 in 4.2.2 and module 3 in 4.2.3). Chapter 5 describes the Testing Strategies Used in section 5.1 which includes Unit Testing in section 5.1.1 and Integration Testing in section 5.1.2 followed by the Testing Results in section 5.2 for each strategy given. Chapter 6 contains the Results (section 6.1) and Analysis (section 6.2) sections of the project. The report is summarised in Chapter 7.

Chapter 2

System Study and Requirement Engineering

At project commencement, a reality assessment is essential. This includes:

- Clarifying the platform's purpose and problem-solving role.
- Gauging the project's potential impact.
- Evaluating budget considerations.

Employing a System Study, involving Literature Survey and Requirement Engineering, helps answer these fundamental questions and establish a solid project foundation.

2.1 Existing Systems

2.1.1 Georeach Emergency App

- Georeach Emergency App [1] offers real-time location sharing with emergency responders and contacts.
- The SOS/panic button enables quick distress alerts with a tap, notifying nearby users and emergency contacts. A covert gesture option discreetly triggers the panic alert without opening the app.
- The app provides encrypted chat for secure communication during emergencies, safeguarding privacy.
- It sends SOS alerts via SMS and email, reaching contacts without the app for broader emergency alert coverage.

2.1.2 VithU

- VithU App [2] lets users select emergency contacts and send them SOS messages with real-time location for swift assistance.
- Users can easily set emergency contacts and activate SOS by double-pressing the power button. The app sends regular updates with location to contacts every 2 minutes during emergencies.

2.1.3 bSafe

- In BSafe App [3] Women can create a personal safety network, or *guardians*, comprising friends, family, colleagues, partners, and others, ensuring a reliable support system in times of need.

- Noteworthy features include voice alarm activation, live streaming, and automatic audio and video recording, enhancing the user's ability to alert guardians and record evidence during emergencies.
- The safety network's guardians can monitor the user's movements in real-time, allowing them to stay informed about her whereabouts for added security.
- The app enables users to activate the SOS button through a voice command, even if the phone is not in their hands, ensuring a quick and discreet way to call for help.
- Additionally, users can use the *Fake Call* feature, prompting their phone to ring and allowing them to discreetly exit unpleasant or unsafe situations. This feature adds an extra layer of protection and empowers users to manage uncomfortable encounters effectively.

2.1.4 Literature Review

A literature review involves a systematic assessment of published research, academic materials, and relevant sources pertaining to a specific topic. It aims to comprehensively analyze existing knowledge, highlight trends, gaps, and debates, and position the current research within the broader field. This process assists researchers in refining their research questions, identifying gaps in existing literature, and providing context for their own work, ultimately contributing to a deeper understanding of the subject and guiding the direction of their research.

2.1.4.1 Safety Souls mobile application for emergency response system

The report *Safety souls mobile application for emergency response system* [4] by Dina Hussein, Dina M. Ibrahim, et al., presents work on an emergency response system based on mobile techniques, by the development of a flexible and dynamic mobile application which offer tracking mechanism and information management. It focuses on the emergency management cycle, which defined the steps as mitigation, pre-preparedness, recovery and response in order to study the applicability of mobile technologies.

2.2 Gap Analysis

Some potential drawbacks were identified in each of the above-mentioned existing systems. They are discussed in the following.

Existing System	Analysis
Georeach [1] Emergency App	Georeach app heavily depends on stable mobile network coverage for optimal functioning, which could be an issue in areas with weak connectivity. The app's effectiveness is tied to user density in an area; fewer users may lead to delayed alerts and responses. This may lead to extended response times and reduced efficiency during emergencies.
VithU [2]	The app heavily relies on consistent internet connectivity, which may limit its functionality in areas with poor network coverage. Users can add only two emergency contacts, potentially inadequate in certain situations. False alarms and lack of added security features pose challenges, impacting user experience and reliability.
bSafe App [3]	The bSafe app's effectiveness relies on stable mobile network coverage, potentially limiting its functionality in weak signal areas. False alarms and subscription requirements pose challenges, causing stress and limiting accessibility. Technical issues, including GPS tracking, may impact its stability.
Safety Souls mobile application [4]	The Safety Souls mobile app is also reliant on consistent and reliable mobile network coverage for optimal operation, which might pose challenges in regions with inadequate network connectivity.

Table 2.1: Gap Analysis of the various existing systems.

2.3 Proposed System

The system proposed in this project is an emergency alert app designed to provide swift assistance and notifications during critical situations. The app aims to enhance user safety by offering various features such as a panic button, location sharing, and a user-friendly interface. Users can alert their designated emergency contacts and nearby app users with emergency alerts, along with their location.

The app includes a comprehensive set of functionalities, such as a login/sign-up page, an emergency contacts page for managing and notifying chosen contacts, a panic/alert button for immediate distress alerts, and a user profile creation page for adding personal details. Additionally, the system integrates map technology to enable location sharing during emergencies. The proposed system not only ensures the safety of users but also encourages community support by connecting nearby app users who can respond to distress alerts.

Advantages

- *Quick Emergency Response*: The app provides a swift and efficient way for users to send alert signals to their emergency contacts and nearby app users, ensuring prompt assistance during critical situations.
- *User-Friendly Interface*: The user-friendly interface makes it easy for users to navigate and access the app's features, even in high-stress situations, ensuring a seamless user experience.

- *Reliable Alerts*: In case of network unavailability, the app will wait till a stable connection is available.
- *Location Sharing*: The ability to share location details helps emergency responders and contacts quickly locate the user, facilitating a faster and more accurate response.
- *User Profile Details*: The inclusion of user profile information, such as medical conditions and blood group, aids emergency responders in providing appropriate assistance to the user's needs.
- *Community Support*: By connecting nearby app users who can respond to alerts, the app fosters a sense of community support, allowing users to receive help from those in close proximity.

Disadvantages

- *Network Dependence*: The app's effectiveness relies on mobile network coverage. In areas with poor or no network connectivity, users may face challenges in sending alert signals.
- *Device Compatibility*: App only supports android operating system.
- *Privacy Concerns*: Sharing personal information and location data could raise privacy concerns among users, particularly if not adequately secured.

2.3.1 Problem Statement

To develop an emergency alert application that uses technology to quickly notify nearby users and their emergency contacts in real-time during an emergency. Our system sends notifications with location information and other relevant details using a panic button, making it easier for people to get the help they need when they need it most.

2.3.2 System Model

The system model of the emergency alert application comprises three integral components: the client, representing the mobile application on users' devices and facilitating interactions; the server, serving as the central processing unit, managing authentication, profile storage, and alert distribution; and the database, storing crucial data like user profiles and alerts, ensuring data persistence and integrity for reliable system functionality.

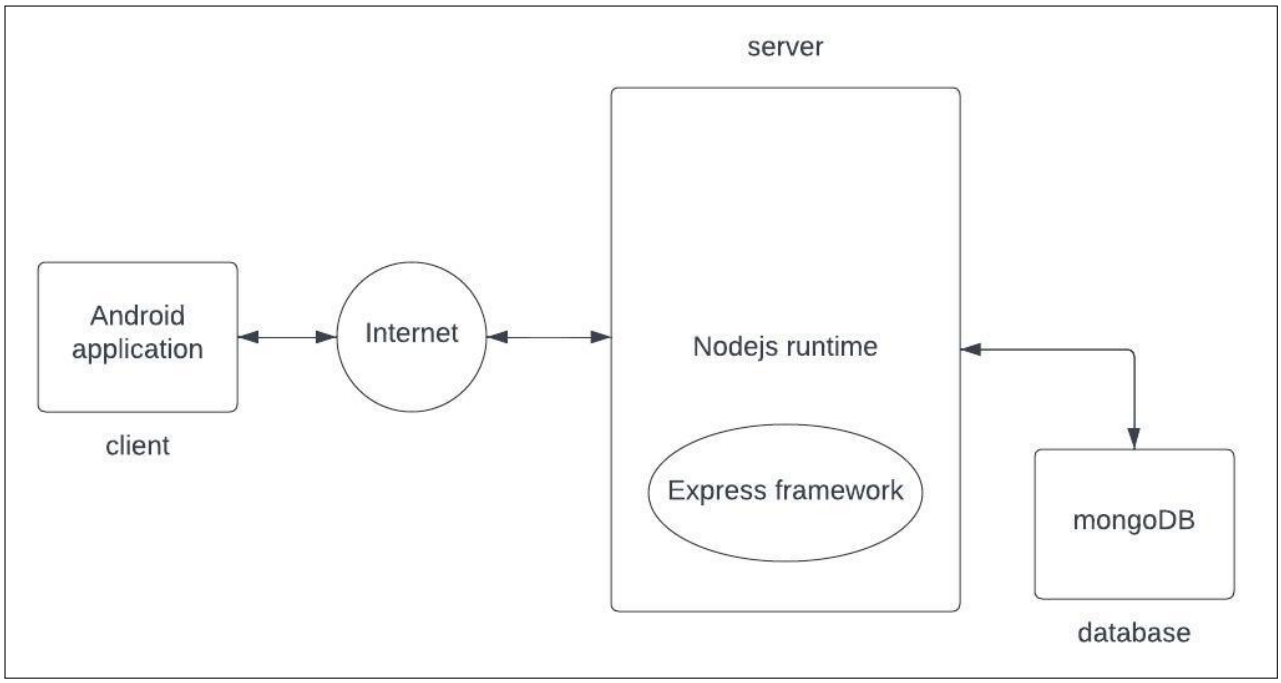


Figure 2.1: System Model

2.4 Requirements Engineering

The process of gathering software requirements from clients and then analysing and documenting them is known as requirements engineering. The goal of requirement engineering is to develop and maintain a sophisticated and descriptive 'System Requirements Specification' document. It essentially involves the following five activities.

1. Feasibility Study
2. Requirements Elicitation
3. Requirements Analysis
4. Requirements Specification
5. Requirements Validation

2.4.1 Feasibility Study

Feasibility check [5, 6] is a procedure that identifies, describes and evaluates the proposed systems and selects the best system for the task under consideration. An estimate is made of whether the identified user needs may be satisfied using current software and hardware technologies. The study will decide if the proposed system will be cost-effective from a business point of view and if it can develop given existing budgetary constraints. The key considerations involved in the feasibility analysis are

- Economic Feasibility

- Technical Feasibility
- Behavioural Feasibility

2.4.1.1 Economic Feasibility

Economic feasibility [5,6], refers to the practicality and viability of a project or initiative from a financial perspective. It involves assessing whether the benefits gained from undertaking the project outweigh the costs associated with its development, implementation, and ongoing operation. This evaluation includes considering factors like potential revenue generation, cost savings, return on investment, and the alignment of financial resources with the project's goals. Economic feasibility helps determine if the project is worth pursuing in terms of its ability to deliver value and contribute positively to the organization's financial health and objectives.

The system proposed demonstrates robust economic feasibility by capitalizing on available resources. Utilizing the open-source Flutter framework for development minimizes software costs. Moreover, its compatibility with widely owned smartphones eliminates the need for proprietary software and specialized hardware.

2.4.1.2 Technical Feasibility

Technical feasibility [5, 6] refers to the assessment of whether a proposed project or system can be successfully developed, implemented, and operated using the available technology and resources. It involves evaluating whether the required technology, infrastructure, skills, and tools are accessible and capable of supporting the project's objectives. This assessment examines factors such as compatibility with existing systems, the availability of skilled personnel, the feasibility of integrating various components, and the potential challenges that might arise during development and implementation. Technical feasibility helps ensure that the project can be executed without significant technological roadblocks and that the organization possesses the necessary resources to bring the project to fruition effectively.

The App enjoys solid technical feasibility as it operates seamlessly on users' existing devices, eliminating the need for extra hardware. The app's reliance on tried-and-tested tools and reliable assets further enhances its technical viability, ensuring a smooth and dependable user experience.

2.4.1.3 Behavioural Feasibility

Behavioural feasibility [5, 6] assesses whether a proposed project or system aligns with the intended users' behaviours, needs, and preferences. It involves understanding how potential users will interact with the system, how well they will adapt to it, and whether the system's functionalities are user-friendly and intuitive. Behavioural feasibility analysis often considers user feedback, expectations, and the overall user experience. This evaluation helps ensure that the project's outcomes will be readily accepted and embraced by the users, minimizing resistance to change and maximizing the system's adoption and effectiveness.

The app had a user-friendly design, utilizes recognizable elements, making it quickly graspable for users. Moreover, since the app operates on devices already owned by users, it enhances familiarity and user comfort, ensuring a seamless and user-oriented experience.

2.4.2 Requirements Elicitation

The central goal of the requirement elicitation process is to amass the project's requisites from stakeholders, who are individuals or groups closely associated with the software and serve as the wellspring of these requirements.

- *Examining current systems*: Existing systems were scrutinized during the literature review, where we assessed their advantages and drawbacks. These specifics are elaborated in section 2.1, supplemented by a gap analysis in section 2.2.

- *Ethnography and Observations*: Ethnography, an observational method, aids in comprehending operational processes and deriving software requirements to facilitate these processes. Analysts immerse themselves in the operational environment, uncovering latent system needs that mirror actual work practices, diverging from formal organizational processes.

  By immersing ourselves in user scenarios, we've recognized that certain complexities arise in emergency alert sending. Users often struggle with managing emergency contacts and comprehending alert functionalities. This first-hand experience guides our approach, ensuring that the app's design and features align with user expectations, ultimately enhancing its usability and effectiveness in critical situations.

2.4.3 Requirements Analysis

Requirement analysis for the Emergency Alert System App involves identifying and documenting the essential expectations from users and stakeholders. The process encompasses various tasks to ascertain the system's functional and non-functional needs. This analysis draws insights from stakeholder interviews, study of similar systems, and gathered data. The extracted information guides the app's development, culminating in the following requirements:

- *Emergency Alert Button*: The app's home screen prominently features an easily accessible emergency alert button that initiates a distress signal when pressed. A built-in delay of 5 seconds allows for cancellation if required, enhancing user control and reducing accidental alerts.

- *Location Sharing*: The system seamlessly integrates GPS and location technologies to provide accurate location information when an emergency alert is triggered, ensuring swift assistance.

- *Notification System*: The app's push notifications swiftly inform emergency contacts and nearby app users about the distress situation, accompanied by relevant user and medical information.

- *User Profile*: Users can create and manage their profiles, including personal details, emergency contacts, and relevant medical information. Profiles can be updated after registration to maintain accuracy.

2.4.4 Requirements Specification

When a requirements specification is written, it is important to remember that the main goal is to deliver the best product possible, and not to produce a perfect requirements specification. There are many good definitions that describe how a requirements specification should be written, but all have at least one part in common, which is the essence of requirements specification, namely the requirements. Requirements are divided into *functional requirements* and *non-functional requirements* [5, 6]. Functional requirements describe the functionality of the desired system that usually consists of some kind of calculation and results, given specific inputs. Non-functional requirements [5, 6] describe how quickly these calculations should be and how quickly the system will respond when its functionality is used. The SRS [5, 6] document is given in the Appendix.

2.4.4.1 Functional Requirements

Functional requirements encompass vital features and behaviours a system or software necessitates to effectively serve its intended purpose, steering its development and operational capabilities.

- The Alert Button should be prominently displayed on the home screen, easily accessible with a single tap, triggering the emergency alert after 5 seconds with an option to cancel within that time frame; additionally, it should appear as a persistent notification on the lock screen for convenient access.
- Push notifications must appear on the lock screen, accompanied by an alert sound, and be easily noticeable and clear to view.
- The user profile should include relevant medical information, and be editable after registration, with profile details transmitted during emergencies.
- The app should utilize GPS and other location technologies for precise user tracking, ensuring the highest level of accuracy possible.
- The application should offer a false alarm flagging option, marking flagged false alarms in the alerts list to prevent unnecessary panic and maintain system reliability.
- Notifications must promptly include user location, pertinent medical and user details, ensuring no delays in the delivery process.
- Users should be capable of creating emergency contacts that include phone numbers and relevant details to facilitate use during emergencies.

2.4.4.2 Non Functional Requirements

Non-functional requirements define the qualities, constraints, and characteristics that a system or software must possess to meet user expectations and industry standards, encompassing aspects like performance, security, usability, and scalability.

Performance Requirements

- *Scalability*: The emergency response system should be able to handle a large number of emergency alerts and users simultaneously without significant performance degradation.

- *Reliability*: The emergency response system should be able to handle different types of network conditions, such as low bandwidth or intermittent connectivity. in case of network unavailability, the system should be able to send notification whenever the device reconnects to the network.
- *Response Time*: The emergency response system should be able to send emergency alerts within a specified time-frame to ensure timely response to emergencies.
- *Concurrency*: System should be able to manage multiple alerts concurrently.
- *Real-time updates*: Location information should be updated in real-time.

Safety Requirements

- *False Alerts*: False alerts should be flagged to avoid unnecessary panic.
- *Data Privacy*: The app must abide by all applicable data privacy laws and rules, including the California Consumer Privacy Act (CCPA) and the General Data Protection Regulation (GDPR). User information must be safely kept in the database and transmitted using encryption.

Security Requirements

- The emergency response system should use encryption to protect sensitive data, such as medical information and emergency contact information.
- The emergency response system should be able to authenticate users and ensure that only authorized users can send emergency alerts.
- The emergency response system should use secure communication protocols, such as TLS, to protect user data and ensure privacy.
- Rate limiting mechanisms to restrict the number of requests that can be made from a single IP address in a given period of time, to prevent Denial of Service attacks.

Software Quality Attributes

- *Scalability*: The system's capacity to handle more users, alerts, and components while maintaining performance. This involves load balancing, scaling strategies, and fault tolerance.
- *Portability*: The system's adaptability to various hardware devices with minimal adjustments.
- *Reliability*: Ensuring consistent, accurate performance without errors, encompassing fault tolerance, error recovery, and redundancy.
- *Security*: Safeguarding user data, ensuring confidentiality, integrity, and availability through encryption, secure protocols, access control, and auditing.
- *Testability*: Facilitating effective testing for correctness, reliability, and performance. This involves automated testing, component testability, and simplified test environment setup.

2.4.5 Requirements Validation

Requirement validation is an essential phase in ensuring the robustness and alignment of our emergency alert application's specifications with user expectations. This process evaluates the completeness, consistency, and quality of the outlined requirements. Through systematic analysis, we have confirmed that the application effectively presents critical features such as the Alert Button's accessibility, user profile accuracy, real-time tracking capabilities, and seamless interaction with emergency contacts and authorities. Additionally, the validation process encompassed non-functional aspects, including interface usability, software reliability, and security measures, all of which were meticulously verified to meet user needs. This requirement validation guarantees the application's reliability, user-centric design.

2.5 Summary

This chapter covers the existing emergency alert applications, pointing out their limitations. The discussion moves into requirements engineering, including feasibility study, requirement collection, analysis, specification, and validation phases. We look at various ways to gather requirements and the standard documentation for them. Finally, the chapter concludes by listing the functional and non-functional needs of the project and how we ensure they are correct.

Chapter 3

Design

The design phase [5,6] is pivotal for creating a structured solution to the defined requirements. It bridges the gap between the problem and its resolution, transitioning from conceptual needs to actionable plans. This phase, encompassing system and detailed design, plays a pivotal role in determining software quality and subsequent stages like testing and maintenance [5,6]. The outcome is a comprehensive solution document, guiding implementation, testing, and ongoing maintenance. Further sections will elaborate on both system design and detailed design aspects.

3.1 System Design

System Design [5, 6] aims to identify the modules that should be present in the system, the specification of these modules and how they interact with each other to produce the desired results. At the end of the system design, all the major data structures, file formats, output formats and major modules in the system and their specifications are decided.

We partition the system design into distinct modules, each addressing a distinct facet of the application. They are:

- Module 1: User Module
- Module 2: Alert Sending Module
- Module 3: Alert Receiving Module

3.1.1 Module 1: User Module

The User Module forms the core of user engagement and personalization within the app. It encompasses essential functionalities such as OTP-based authentication for secure registration and login. Users can create comprehensive profiles, including personal information and medical information like blood group. Additionally, users have the ability to set up their emergency contacts to send alert via SMS during emergencies.

3.1.2 Module 2: Alert Sending Module

The Alert Sending Module plays a pivotal role in initiating and broadcasting emergency alerts. When a user activates the alert, the module ensures that notifications are sent promptly to fellow app users and designated emergency contacts. This module leverages multiple communication channels, such as push notifications and SMS, to ensure a widespread and timely alert dissemination network.

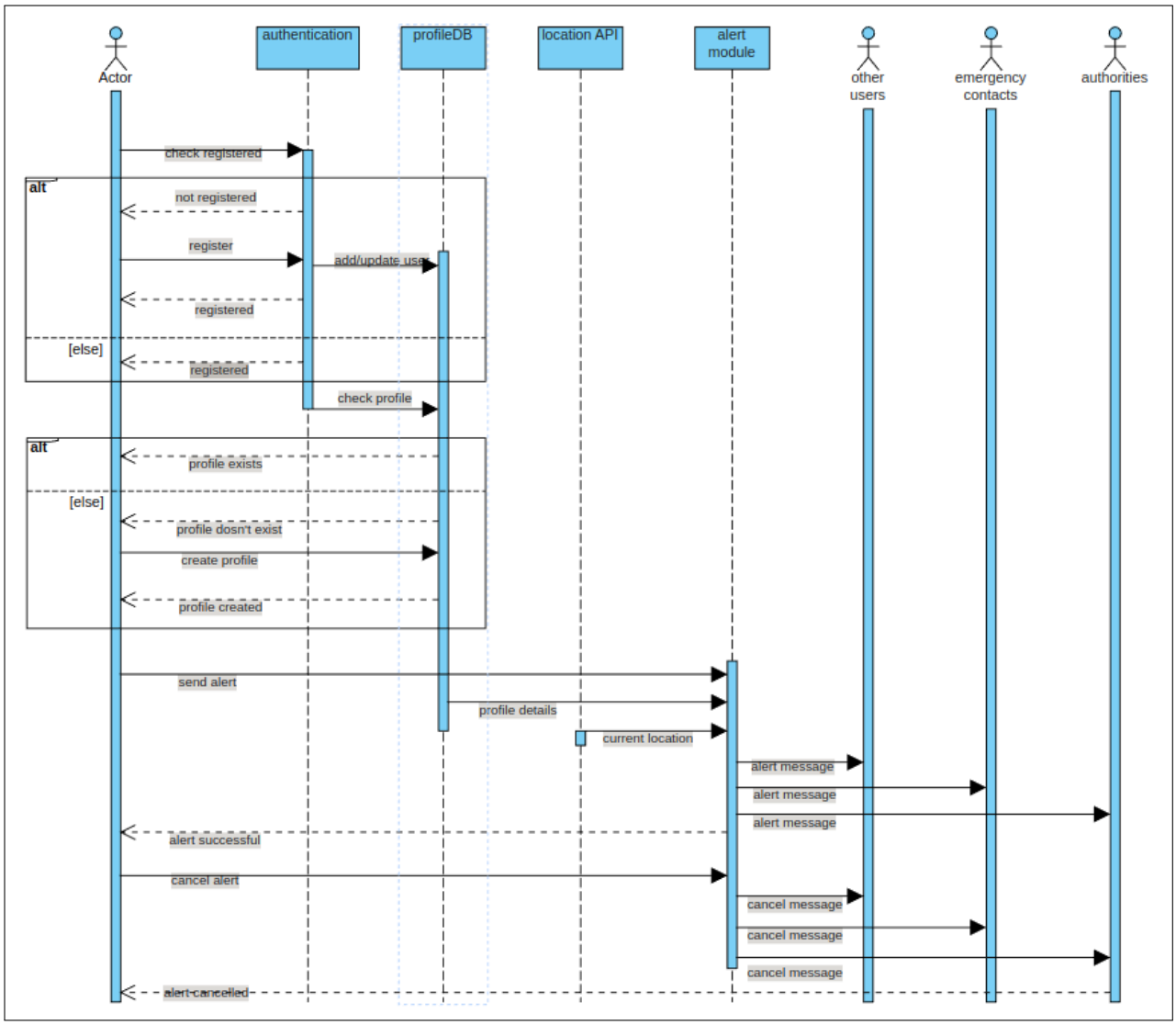


Figure 3.1: Overall Sequence Diagram

3.1.3 Module 3: Alert Receiving Module

The Alert Receiving Module focuses on facilitating user awareness and effective responses to incoming alerts. It manages the reception of alerts, presenting them as real-time notifications within the app. Users can delve deeper into alert details, contact the distressed user, visualize locations on a map, and address false alerts if necessary.

These modules collectively shape a comprehensive and intuitive emergency alert system app, ensuring rapid alert initiation, broad distribution, and informed response capabilities.

3.2 Detailed Design

3.2.1 Module 1 : User Module

The User Module is a cohesive amalgamation of several distinct functionalities, each serving a specific purpose within the app's framework. These functionalities encompass diverse aspects such as User Registration, profile creation, settings configuration, and emergency contact establishment. By seamlessly integrating these components, the User Module ensures a comprehensive and streamlined user experience, contributing to the app's overall efficiency and user-centric design.

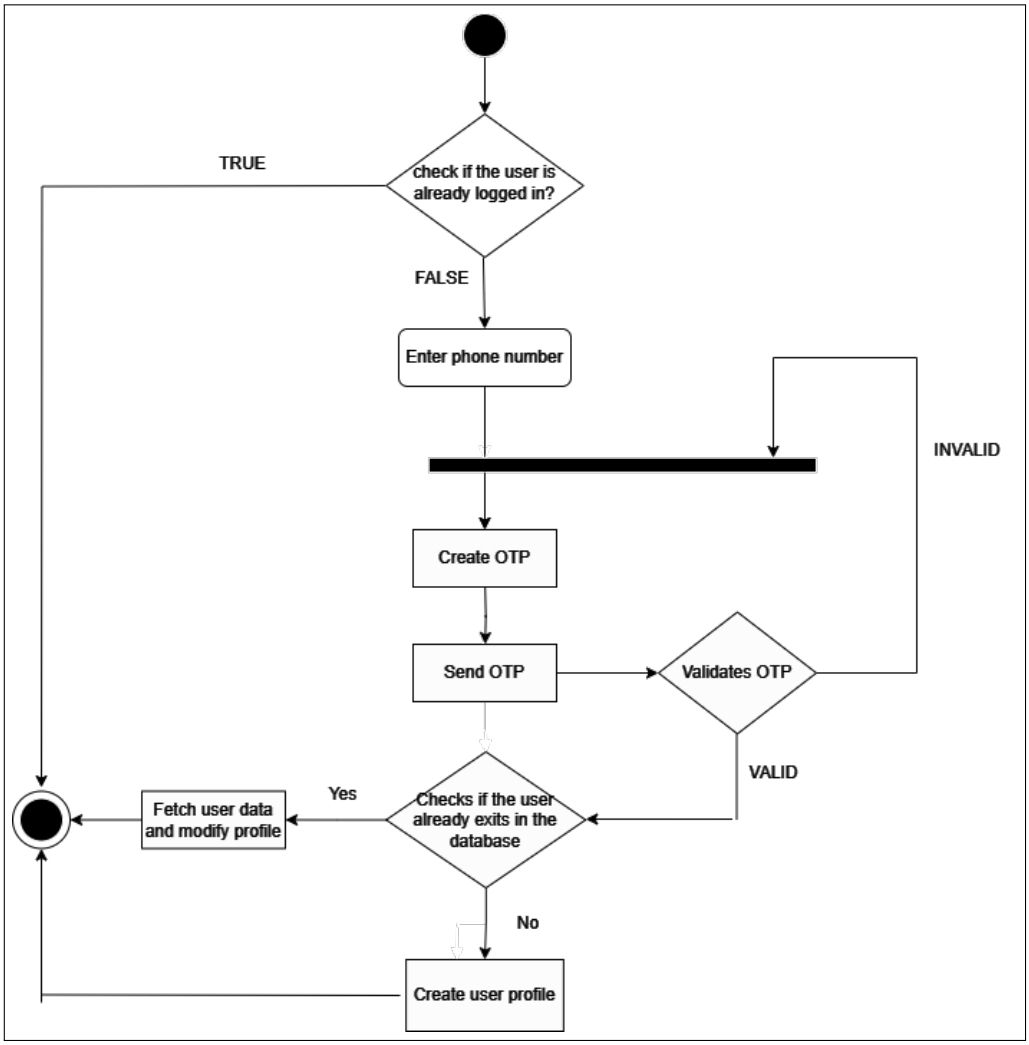


Figure 3.2: User Registration and Profile Creation

Algorithm 1: Algorithm for User Registration and Profile Creation

```
Data: User Details
Result: User Registration and Profile Creation
while userNotLoggedIn() do
    phone_number = getUserInput();
    sendOTP(phone_number);
    received_otp = getUserInput();
    if verifyOTP(received_otp) then
        sendPhoneNumberToServer(phone_number);
        if phoneNumberExistsInDatabase(phone_number) then
            token = getTokenFromServer();
        else
            token = getNewTokenFromServer();
        end
        profile_data = getUserProfileData(token);
        if profile_data exists then
            loadProfile(profile_data);
        end
        name = getUserInput();
        date_of_birth = getUserInput();
        blood_group = getUserInput();
        medical_conditions = getUserInput();
        saveUserProfileToServer(token, name, date_of_birth, blood_group,
         medical_conditions);
    end
end
```

3.2.2 Module 2 : Alert Sending Module

The Alert Sending Module in our project initiates emergency alerts. Upon pressing the alert button, the module checks for cancellation within 5 seconds. If not cancelled, it verifies network availability. If the network is stable, it retrieves the user's location and proceeds. The server sends alerts to app users. Simultaneously, the module sends the alerts to emergency contacts via SMS. This process ensures efficient and effective alert dissemination to both app users and designated contacts.

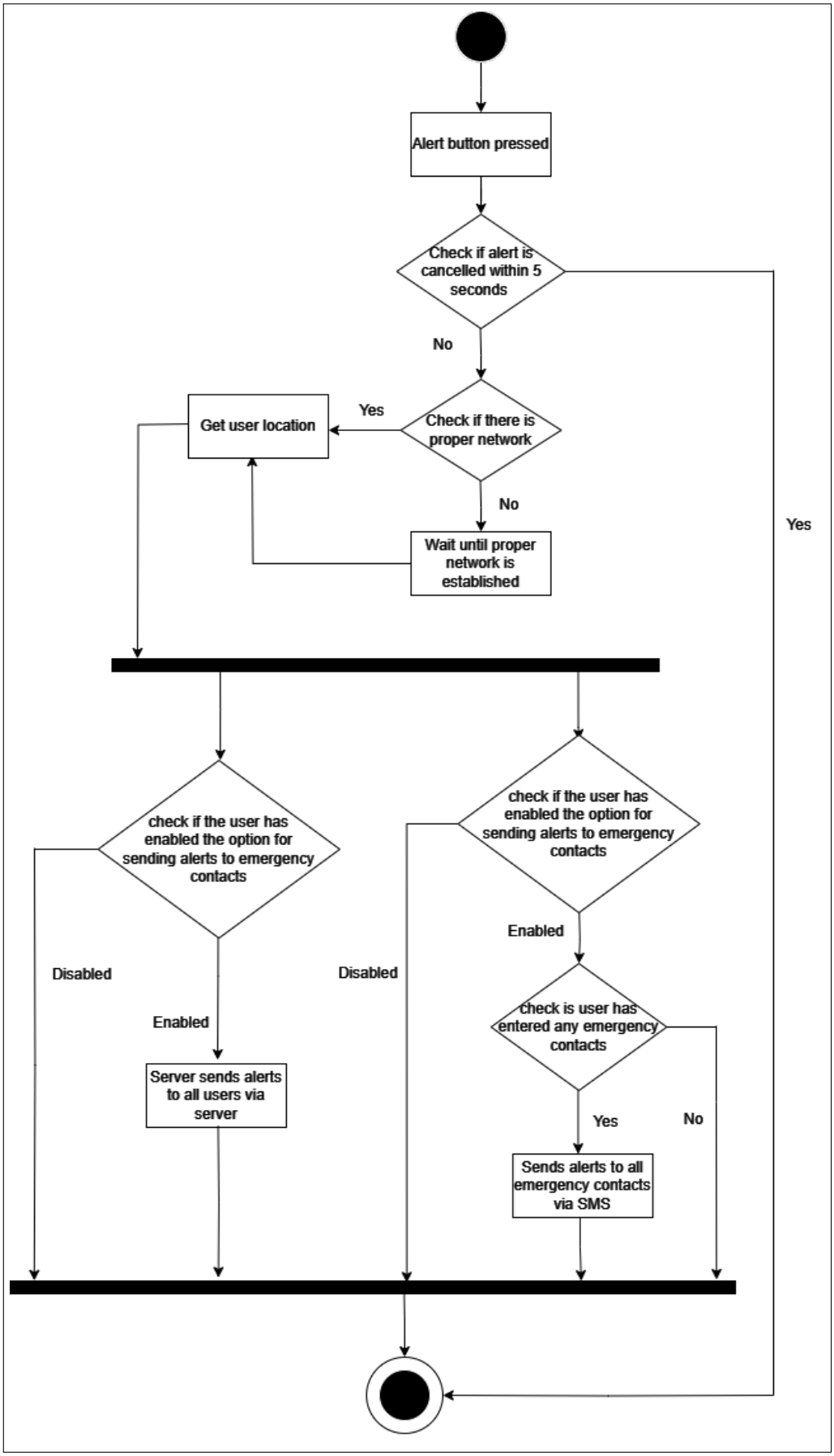


Figure 3.3: Alert Sending

Algorithm 2: Algorithm for sending alerts

```
Data: User Data, Location Data
Result: Sending Alert
if Emergency Alert Button Pressed then
    Start Countdown (5 seconds);
end
while Countdown not Expired do
    if Cancel Button Pressed then
        exit;
    end
end
if Proper Network Connection Established then
    Obtain User's Current Location;
    Fetch User Data;
    if Emergency Contacts Alert Enabled then
        Retrieve Emergency Contacts;
        Send SMS Alerts to Emergency Contacts;
    end
    if App Users Alert Enabled then
        Send Alert Data to Server;
    end
else
    Wait for Stable Network Connection;
end
```

3.2.3 Module 3 : Alert Receiving Module

The Alert Receiving Module of our project effectively manages incoming emergency alerts, delivering timely notifications with essential details. Users can access nearby alerts, view comprehensive information, communicate directly, and locate incidents on a map. The *flag as false* option enables differentiation between true emergencies and alerts. This module enhances community awareness and responsiveness to local emergencies.

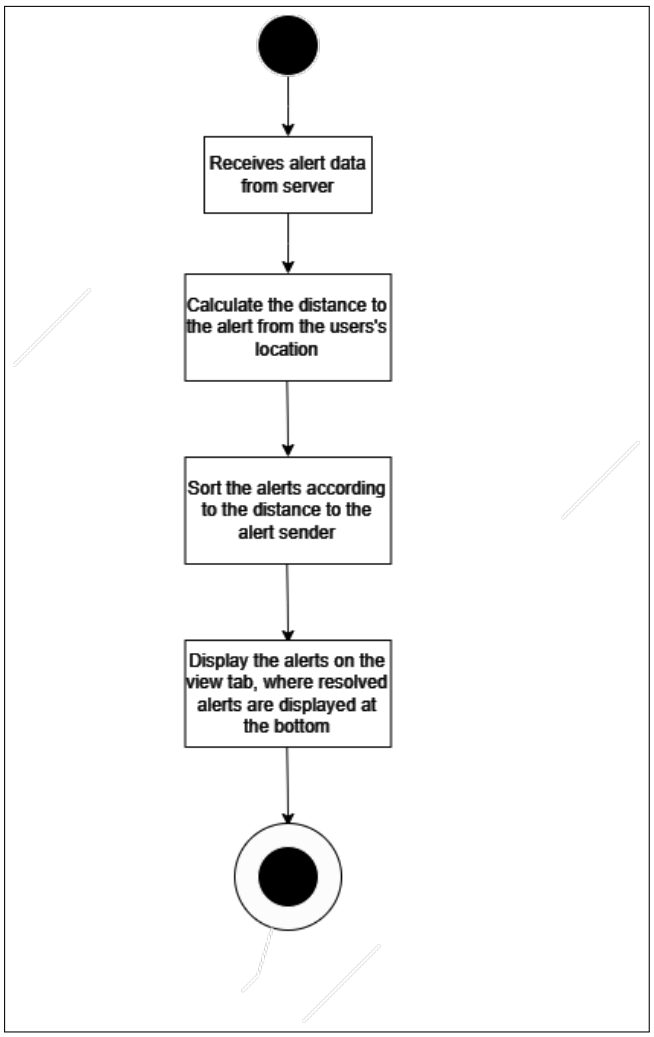


Figure 3.4: Alert Receiving

Algorithm 3: Alert Notification Algorithm

Data: Alert Data
Result: Receiving Alert Notification
if *Server Receives an Alert* **then**
  Dispatch Alert to All Users through FCM Tokens;
end
Mobile Devices Generate Push Notifications Upon Receiving Alerts;

Algorithm 4: Algorithm For Displaying All Alerts

Data: Alert Datas
Result: View All Alerts
alertDatas = RetrieveAllAlertDataFromServer();
currentLocation = getUserLocation();
calculate distances of alerts with currentLocation;
Sort alertDatas by Distance and Status;
Display Sorted Alerts;

Algorithm 5: Algorithm For Displaying Alerts Details

Data: AlertId ,Alert Data
Result: View Alert Details
1 alertData = RetrieveAlertDetails(AlertId);
2 Display AlertDetails;

3.3 Database design

The database design includes three main tables: userDB containing fields for user identification, phone number, authentication token, and a reference to the user's profile; profileDB with fields for user profile information such as name, blood group, medical details, and date of birth; and alertDB comprising fields for alert details such as time, location, view count, flag count, and status.

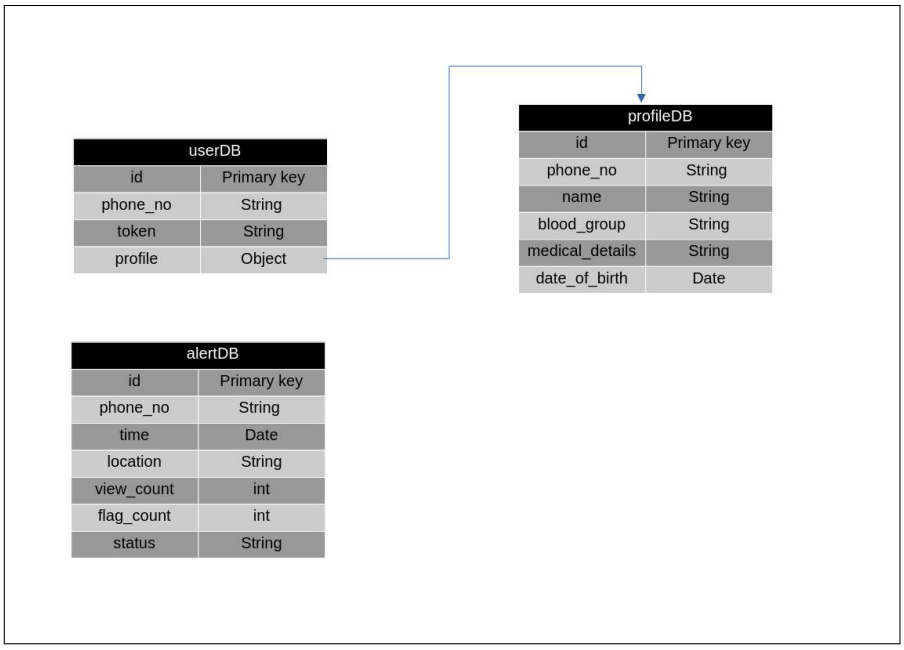


Figure 3.5: Database Schema

3.4 Summary

The design phase of our emergency alert application encompasses a deliberate and holistic strategy for creating an efficient and user-centric system. This process involves the careful delineation of key modules: the User Module, responsible for user interactions; the Alert Sending Module, facilitating prompt alerts dissemination; and the Alert Receiving Module, ensuring timely receipt of critical notifications. These modules were meticulously designed to align with the overall goal of enhancing emergency response capabilities. Through detailed design, the interfaces, interactions, and data flow within each module were refined to ensure optimal usability and seamless communication. This design approach underscores the significance of user experience, technical reliability, and the seamless integration of functionalities.

Chapter 4

Implementation

The development or implementation stage [5, 6] is the stage where the development of the code and modules are performed. The development and integration of the units and creation of the project build are done according to the design documents' specifications. The goal of this phase is to translate the design of the proposed system into a working model. Since this phase affects the testing phase, the coding for the project is performed in such a way as to maintain simplicity and clarity.

4.1 Tools Used

since our application is time-critical and requires a faster response, the following tools and technologies were used:

4.1.1 Flutter

Flutter [7] is an open-source UI software development kit (SDK) developed by Google. It allows for cross-platform app development, enabling the creation of Android and iOS applications from a single codebase. Flutter provides a fast and efficient development experience with hot reload, facilitating quick iterations and bug fixing. Its rich set of pre-built widgets simplifies UI development, and it offers excellent performance due to its native compilation approach. Flutter was chosen for its ability to develop a highly responsive and user-friendly interface, which is perfect for our use case

4.1.2 Node.js

Node.js [8] is an open-source, cross-platform runtime environment that executes JavaScript code outside the browser. It empowers high-performance, scalable server-side applications through an event-driven, non-blocking I/O model for efficient handling of multiple connections. Node.js is well-suited for an emergency response system due to the following reasons:

- Real-time Communication: Crucial for alerts and notifications, Node.js efficiently handles data streams, swiftly delivering emergency information to users and responders.
- Real-time Database Sync: Node.js seamlessly integrates with databases like Firebase, ensuring instant, updated info for users and responders in emergencies.
- Scalability: Node.js manages high user activity during emergencies using non-blocking I/O, preventing bottlenecks.

4.1.3 MongoDB

MongoDB [9], a popular NoSQL database, is designed for efficient storage and retrieval of large volumes of unstructured or semi-structured data in JSON-like documents. It offers flexibility, scalability, and a hosted version called MongoDB Atlas. With fast read and write operations, MongoDB suits real-time data needs, crucial in time-sensitive applications like emergency alerts. Its geospatial indexing enables efficient location-based queries. The JSON-like format harmonizes with Node.js, ensuring seamless integration and efficient development.

4.1.4 Firebase

Firebase [10] is a comprehensive mobile and web application development platform offered by Google. It provides a suite of cloud-based services and tools that facilitate various aspects of app development, including real-time database, authentication, cloud storage, and cloud messaging.
In this project, firebase was used for authentication and real-time notifications.

4.1.4.1 Firebase OTP-Based Authentication

- Firebase OTP authentication offers secure user verification via One-Time Passwords (OTPs) sent to mobile devices for registration or login.

4.1.4.2 Firebase Cloud Messaging (FCM)

- Firebase Cloud Messaging (FCM) [10] is a key Firebase feature for sending push notifications and messages to multiple platforms, ensuring reliable real-time delivery to devices.

4.2 Module Implementation

In this section, implementation of each module is explained in detail.

4.2.1 Module 1: User Module

This module manages user registration, authentication, and profile management. It ensures secure access to the application and enables users to create, update, and maintain their profiles, including medical information and emergency contacts.

4.2.1.1 User registration

User registration involves two key steps: OTP-based authentication and user account creation. The Firebase authentication module handles OTP authentication. It sends an OTP to the user's provided phone number, which they verify by entering it. Successful OTP validation generates a *credential* key.
This *credential* key, coupled with the user's phone number, is then transmitted to the server using the *registerUser()* function, via POST request.

The backend server verifies this credential using Firebase Admin SDK and generates a JWT token and sends it back to the client as a response to the POST request. The client stores the phone number and JWT token in flutter-secure-storage. The backend server stores the phone number credential and JWT token into the database.
After registration, emergency profile creation commences.

4.2.1.2 User Login

User login follows the same steps as registration. After entering and validating OTP, *registerUser()* function is called, which sends the data to the backend server. If the user already exists in the backend database, JWT token of the user is sent back to the client. Client stores phone number and token using flutter-secure-storage. Then the existing profile of the user is returned to the client.

4.2.1.3 Profile Creation

Emergency profile is created at the client side and sent to the backend server using a POST request. The backend server stores this emergency profile in database *profileDB*. This profile will be accessed at the time of sending alerts.

4.2.1.4 Emergency Contacts

Users can create their emergency contacts, which can be edited anytime through the settings menu. These contact details are saved in the Android app using shared preferences.

In shared preferences, we organize the emergency contacts as list of contacts. Each contact contains a name and phone number. To distinguish between different contacts, we assigned a unique *contactKey* to each contact.

4.2.2 Module 2: Alert Sending Module

Module 2, recognized as the Alert Sending Module. Activation of this module is initiated by pressing the alert sending button. Alert sending button is situated at both home screen and as a permanent notification. Pressing this button calls *startCountDown()* function, which starts a 5-second count-down for cancelling the alert in case of accidentally pressing the button. After the countdown expires, *sendAllAlerts()* function is called. *sendAllAlerts()* function checks the settings and activates functions to send alerts to nearby users and/or emergency contacts, according to the preferences.

4.2.2.1 Alert Nearby Users

If enabled, the alert is transferred to nearby users of the application. When an alert is initiated, it is seamlessly relayed to the backend Node.js server, accompanied by essential details such as the user's location, timestamp, and user ID. The backend server integrates the user profile from *profileDB* into the alert and converts it into a structured JSON format. This refined alert is then stored in the alert database, *alertDB*. Upon receiving such an alert, the backend server immediately sends the alert to other users, with firebase cloud messaging services.

4.2.2.2 Alert Emergency Contacts

In this section, the alert is sent to the emergency contacts that the user set up earlier. At the client side of the application, the location and time of the alert is attached to a message. The message is then sent to the emergency contacts via SMS. Each contact stored in shared preferences is accessed with *contactKey*. SMS is sent to each of the contacts with help of telephony package provided by Flutter.

4.2.3 Module 3: Alert Receiving Module

4.2.3.1 Alert Notification

Alert notification is enabled by Firebase Cloud Messaging service(FCM). Upon receiving an alert, backend server send the alert details to other client devices using Firebase Admin SDK, with FCM service. This is done by using *fcmtoken* of each device stored in the *tokens* array.

4.2.3.2 Nearby Alerts

The Nearby Alert module displays alerts close to the user's location. Upon accessing this section, a GET request is sent to the backend Node.js server, which retrieves nearby alerts in JSON format. The client app assesses alert locations relative to the user's coordinates, sorts them by proximity, and presents them as tiles. Each tile shows the alert initiator's name, distance from the user, and time elapsed since dispatch. Resolved or irrelevant alerts are marked as Aborted.

4.2.3.3 Alert Details

When selecting a particular alert from the nearby alert section, a GET request is sent to the Node.js server requesting details regarding that alert. The backend server returns the alert details in JSON format. The data received from the server is then displayed in the alert details page. Alert details contain the information from the user's emergency profile, location of the user, and the time the alert is dispatched. Location of the user can be viewed in google map using *google map API*. Contact button in the alert details page works by accessing the contact number in the default phone application of the device.
First time a user visits a particular alert, view count of the alert is updated with *updateView()* function.
Flag alert process works similarly by updating the *falseCount* related with the alert, using *updateCount()* function.

4.3 Summary

The Implementation section of the project report encompasses three key subsections: User Module, Alert Sending Module, and Alert Receiving Module. The User Module handles user authentication, profile creation, and emergency contact setup. The Alert Sending Module triggers alerts to nearby users and emergency contacts via push notifications and SMS. The Alert Receiving Module displays alert notifications and details using Firebase Cloud Messaging and Node.js API. This section highlights the integration of advanced technologies and user-centered design for an efficient emergency alert application.

Chapter 5

Testing

Software Testing [5, 6] is a method to check whether the actual software product matches expected requirements and to ensure that the software product is Defect free. It involves the execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements [5, 6].

5.1 Testing Strategies Used

In this section, the strategies that were employed to test our project will be discussed. The following list outlines the testing strategies that were utilized in our project to ensure its quality and effectiveness.

5.1.1 Unit Testing

Unit testing verifies individual components in isolation, ensuring proper functionality and early defect detection. It simplifies testing, aids documentation, and supports continuous integration. This practice enhances software reliability, promotes sound coding practices, and facilitates seamless deployment.

5.1.1.1 Black Box Testing

Black-box testing is a software testing technique where the tester evaluates the functionality of an application without having access to its internal code or implementation details. The tester treats the application as a "black box," focusing solely on its inputs, outputs, and expected behavior, without any knowledge of how the application achieves its results.

In black-box testing, the tester's goal is to verify whether the application meets its specified requirements and functions correctly from the user's perspective.

5.1.1.2 White Box Testing

White-box testing is a software testing technique that involves examining the internal structure and logic of an application's code. Testers have access to the source code and use this knowledge to design test cases that explore the application's internal paths, branches, and data flows. While white-box testing provides deep insights into the application's internal workings, it does not guarantee complete verification of the system's external behaviour.

5.1.2 Integration Testing

Integration testing is a software testing technique that focuses on evaluating the interactions and integration points between different modules or components of a software system. The primary goal of integration testing is to identify defects that may arise when multiple units

are combined and to ensure that the integrated components work together as intended. Integration testing ensures that the integrated software system behaves as a cohesive unit, verifying that all components function together seamlessly and that the system satisfies the specified requirements.

5.2 Testing Results

Tests were conducted using the strategies mentioned earlier. The results obtained from the tests are as follows:

5.2.1 Results of Black Box Testing

The emergency alert button successfully activates the emergency alert system and sends alerts to nearby users and emergency contacts. The user can create, update, and delete an emergency profile along with medical and other relevant information and the same is transmitted along with the emergency alerts. The location transmitted with the emergency alerts is accurate and precise, featuring an error margin of approximately ± 30 meters. Nearby users receive timely notifications when an emergency alert is triggered.

5.2.2 Results of White Box Testing

Rigorous white box testing of the emergency alert application revealed good code functionality with successful outcomes for the majority of functions. The testing process included diverse scenarios, edge cases, and boundary conditions, ensuring code correctness. Minor issues were promptly addressed, further reinforcing the application's reliability. This thorough testing has enhanced the application's overall quality, performance, and user experience, validating its optimal functionality across various scenarios.

5.2.3 Results of Integration Testing

Integration testing affirmed seamless interaction between the User Module, Alert Sending Module, and Alert Receiving Module. The modules effectively communicated and executed functionalities, ensuring data integrity and system stability. Minor glitches were swiftly resolved, reinforcing the application's overall cohesion. These results validate the application's reliability in delivering accurate alerts while maintaining user data security.

5.3 Summary

Our comprehensive testing methodology encompassed Black Box Testing, White Box Testing, and Integration Testing. Through Black Box Testing, we verified accurate alerts and seamless user experiences; White Box Testing validated code robustness, while Integration Testing ensured smooth module collaboration and data exchange. These combined approaches fortified the reliability, user-centric functionality, and emergency response capabilities of the AlertMe application, ensuring a robust and efficient system.

Chapter 6

Results

6.1 Results

The following objectives were achieved through the course of developing **AlertMe**:

1. *Successful Development of a Real-Time Emergency Alert System*: **AlertMe** was effectively designed and implemented as an android application to serve as a real-time emergency alert system. The app enables users to send emergency alerts to nearby users and emergency contacts during critical situations.

2. *User-Friendly Platform*: **AlertMe** offers an intuitive and straightforward interface, allowing users to quickly and easily trigger emergency alerts when needed most.

3. *Seamless Communication of Critical Details*: **AlertMe** efficiently communicates crucial information alongside emergency alerts, including location details and relevant information to facilitate rapid response and assistance.

4. *Efficient Emergency Alert Generation*: The development process ensured that **AlertMe** provides prompt alert generation, enabling users to seek immediate help and support during emergencies.

5. *Incorporation of Contact Management*: The app successfully integrates a Contact Management Module, allowing users to designate emergency contacts who will receive alerts in case of emergencies.

The objectives of the app were accomplished by breaking down all the necessary functionalities into distinct modules and then integrating them cohesively within the app. The modules include User Module, Alert Sending Module, and Alert Receiving Module, which work seamlessly together to provide a comprehensive and user-friendly emergency alert application.

Upon opening the app, the user is prompted to enter the phone number for OTP verification. After verification is done, the user will be redirected to emergency profile page. If a registered account associated with the provided phone number is found, the app fetches the corresponding profile details from the database and presents them to the user. Users can either keep the existing details unchanged or make necessary modifications. However, if no account is found for the given phone number, users will be prompted to enter their details for creating a new profile.

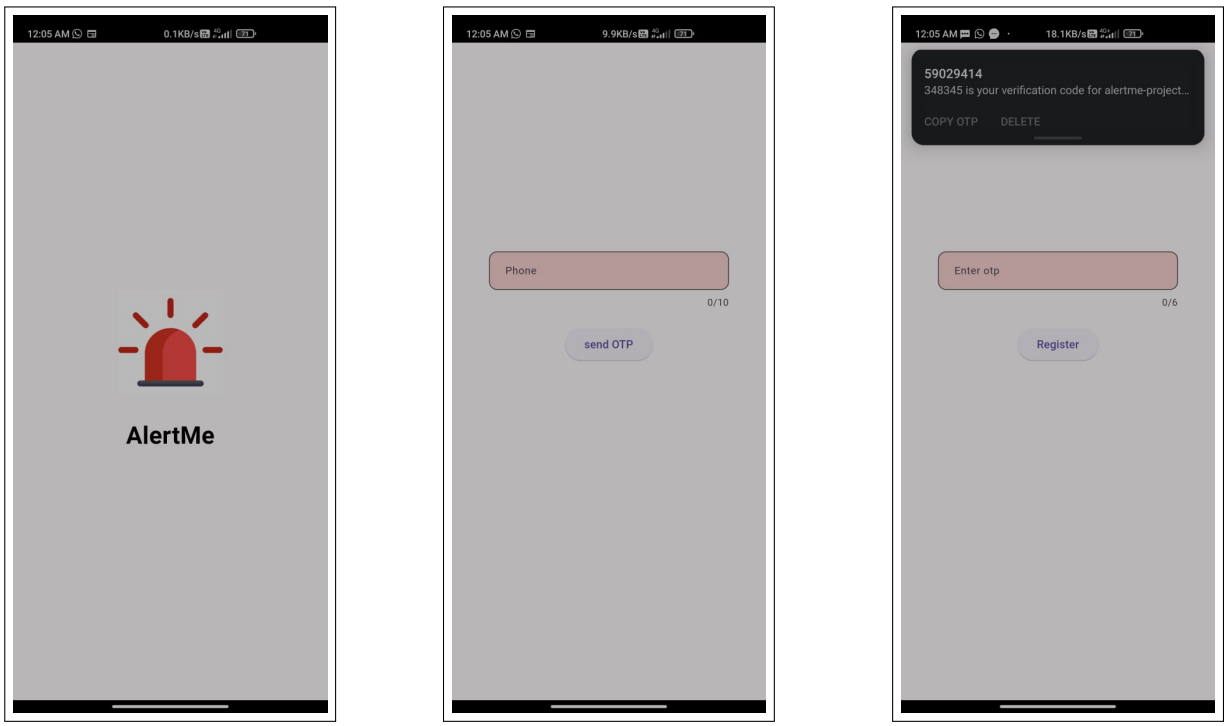
After the user creates their account, they are directed to the settings page to configure their preferences. Here, the user can select the desired alert recipients and enable options like the permanent notification for quick access to the panic button. Additionally, users have the flexibility to log out or delete their account from this settings page.

Upon proceeding to the home page, the user can press the panic button, initiating a five-second countdown, the user can cancel the alert during this countdown. Following the countdown, emergency alerts are dispatched to the designated recipients. Additionally, users have the option to select the type of alert after the alert is already sent. In the event of

the emergency being resolved, the alert abort button serves to cancel the alert, so further responses are not initiated.

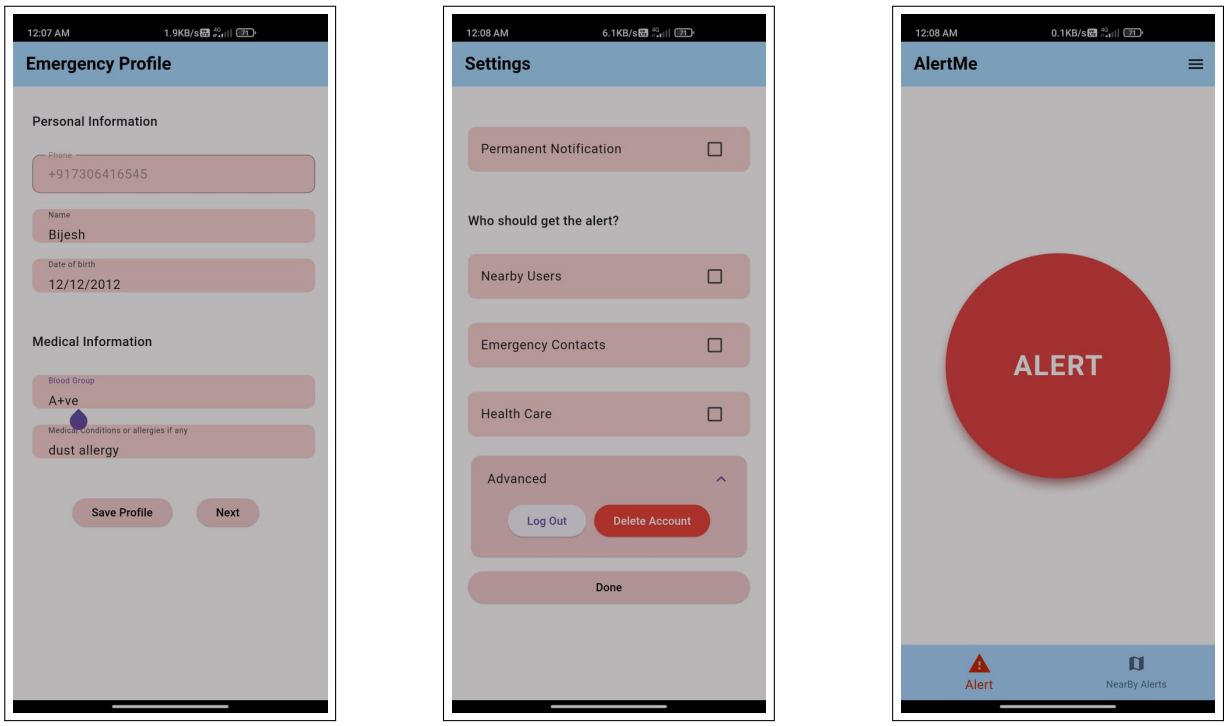
When an alert is received, a notification is pushed to client devices. In the nearby alerts page, all the alerts are displayed in non-decreasing order of distance. Individual alert details can be viewed by selecting an alert.

The drop-down menu offers convenient access to various essential pages. Users can redirect to the settings page to adjust their preferences, the emergency profile page to edit their profile information, or the emergency contacts page to set up their emergency contact list. Additionally, the FAQ page provides more information about the app's functionalities, while the About page offers comprehensive details about the app itself. Moreover, users have the option to quit the app when desired.



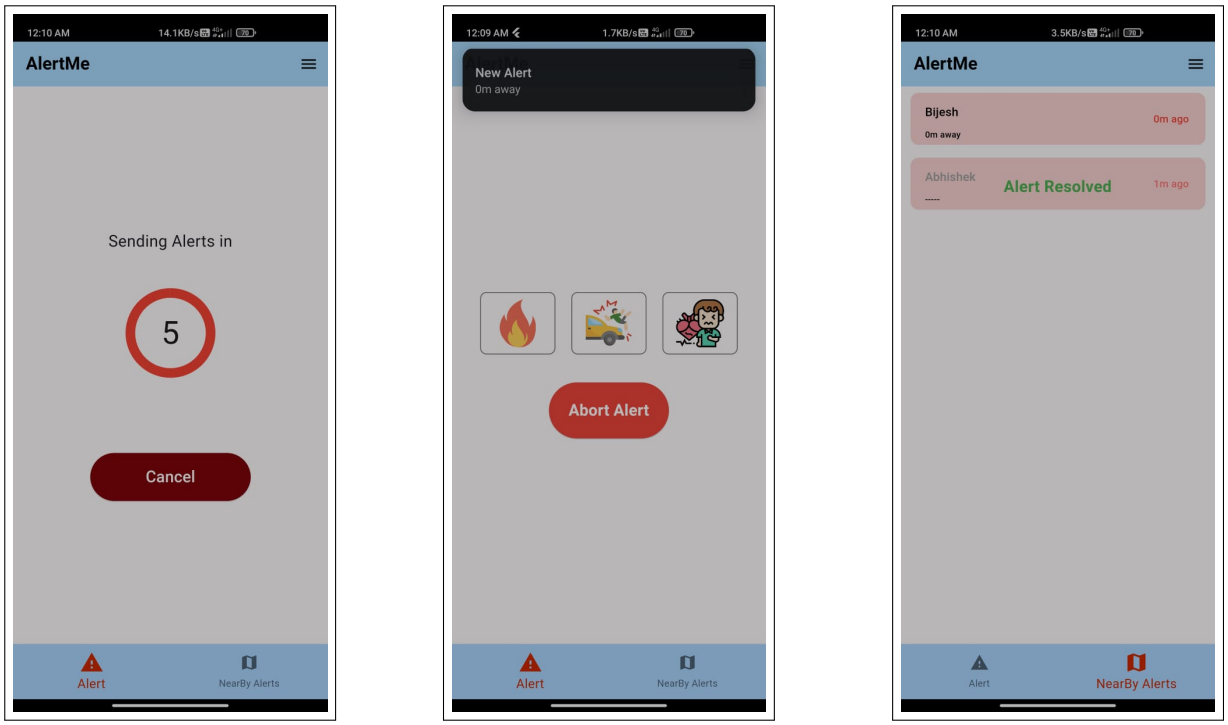
(a) loading page (b) Registration page (c) OTP verification page

Figure 6.1: Loading page, Registration Page and OTP verification Page



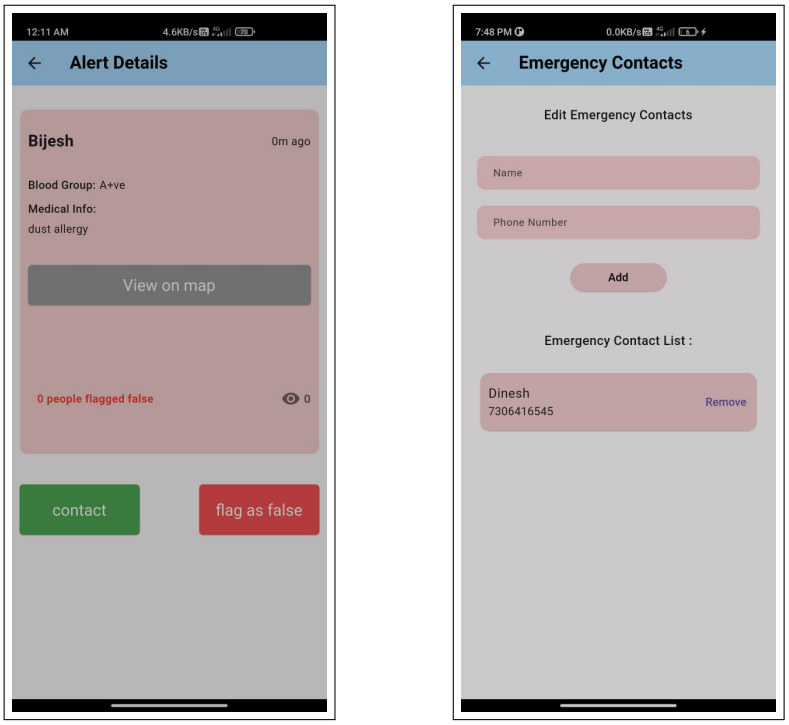
(a) Profile page (b) Settings page (c) Home page

Figure 6.2: Profile page, Settings Page and Home Page



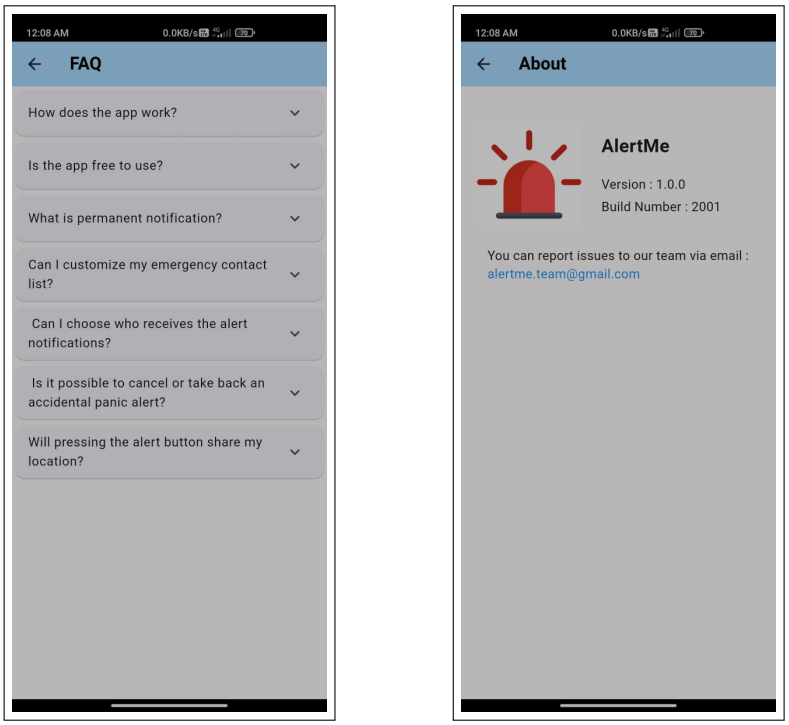
(a) Countdown (b) Alert Sent & notification (c) Alert Page

Figure 6.3: Countdown ,Alert Sent and notification , Alert Page



(a) Alert Details page (b) Emergency contacts page

Figure 6.4: Alert Details page and Emergency contacts page.



(a) FAQ page (b) About page

Figure 6.5: FAQ page and About page.

6.2 Analysis

The result analysis section of a project report provides an in-depth examination and interpretation of the findings and outcomes obtained during the project. It aims to analyze the data collected, draw meaningful conclusions, and evaluate the extent to which the project objectives have been achieved. Through a systematic and comprehensive analysis, this section offers valuable insights into the significance and implications of the results, enabling readers to understand the project's impact and effectiveness.

1. All non-functional requirements were met, including:
   (a) **Interface Requirements:** The app's user interface is designed with Android Material Design, providing a user-friendly and visually appealing experience.
   (b) **Software Quality Requirements:** The app demonstrates smooth performance and minimal lag during usage. All identified bugs and glitches were promptly addressed during development.
   (c) **Hardware Requirements:** The app runs smoothly on Android devices with varying hardware specifications.
2. Through Observations of the users, they find it seamless to create their emergency profile, inputting necessary details such as emergency contacts and medical information. Upon pressing the Alert Button, user observations reveal that emergency alerts are promptly sent, providing real-time location data and relevant information to nearby users and contacts. Hence, the app correctly performs the sending alert function.
3. The app efficiently delivers push notifications to users during emergency situations, as evidenced by user feedback. When receiving alerts, users can readily view important details like real-time location and medical information, facilitating swift response and informed decision-making during critical moments. The observations affirm the app's effectiveness in alert reception and information dissemination.

6.3 Summary

In this chapter, the results of the work are explained in detail. It begins with the objectives that were achieved through the project and also how these objectives were achieved as well through the use of the outputs obtained and responses from the user.

Chapter 7

Conclusion

7.1 Summary

This project report details the development process of **AlertMe**, an android application designed to be a real-time emergency alert application.

Chapter 1 introduces the work, outlining objectives, motivations, scope, and prerequisites for the app. The primary goal is to create a user-friendly platform that sends emergency alerts to nearby users and emergency contacts during critical situations.

Chapter 2 explores existing emergency alert systems and identifies the need for an interactive and efficient solution like **AlertMe**. The project is compared with similar applications to understand its unique features and potential improvements.

Chapter 3 focuses on the system design, dividing **AlertMe** into modules such as the User Module, Alert Sending Module, and Alert Receiving Module. Detailed design specifications and algorithms are developed for each module.

Chapter 4 transitioned us to implementation, where theory turned tangible. We created a user-centric UI using Flutter/Dart. A robust Node.js server orchestrated seamless API calls, working alongside the MongoDB database to store crucial data.

Chapter 5 presents testing of the developed software. Rigorous testing methods, including black box, white box and integration testing, is conducted to address any issues. **AlertMe** effectively meets its primary objectives, offering a straightforward emergency alert system with clear communication of critical details.

In Chapter 6, we delve into the project's outcomes, drawn from attentive user observations. These results affirm the application's adherence to requirements, highlighting its user-centered design. The chapter underscores the successful integration of non-functional necessities, safety protocols, security measures, and software quality attributes.

In conclusion, **AlertMe** stands as a reliable and user-friendly emergency alert application, providing timely assistance during emergencies and promoting safety and security in critical situations.

7.2 Recommendations for Future Works

For performing future research on this work, we recommend the following.

1. *Message Sending Functionality*: Consider adding the functionality to send voice/text messages to users or emergency contacts after sending the alerts. This feature would enable users to provide additional details or updates during emergencies, enhancing communication and coordination.

2. *Alerts to Individuals within a Locality*: Implement the capability to send alerts to specific individuals within a fixed locality. This feature would allow users to target nearby individuals who may offer immediate assistance in emergency situations, fostering a stronger sense of community support.

3. *Real-Time Location Transmission*: Integrate real-time location transmission alongside alerts. Providing location information to responders and authorities can expedite response times and ensure accurate assistance during critical situations.

4. *Alerting Health Care Services*: Incorporate the ability to send alerts directly to health care services. This feature could be crucial in medical emergencies, enabling rapid medical response and potentially saving lives.

5. *Group Alerting*: Implement the option to send alerts to pre-defined groups of contacts, such as family or colleagues. Group alerting would streamline communication and ensure that relevant individuals are promptly informed during emergencies.

6. *Integration with Wearable Devices*: Explore the possibility of integrating **AlertMe** with wearable devices, such as smartwatches or fitness trackers. This integration could offer an additional layer of convenience and accessibility during emergency situations.

7. *Multi-Language Support*: Provide support for multiple languages to cater to a broader user base. Offering **AlertMe** in different languages would enhance accessibility and user engagement.

Appendix A

Software Requirement Specification

A.1 Introduction

A.1.1 Purpose

This document outlines the functional and non-functional requirements of the project. It provides a comprehensive overview of the features and functionalities of the app, along with the intended audience, document conventions, project scope, and references.

A.1.2 Document Conventions

- Fonts: Font used in the document is Computer Modern, entire document follows this font. Font size used is 12pt.
- Section Headings: Section Heading are in larger font size of 17pt, bold, left-aligned and numbered.
- Sub Section Headings: Sub Section headings are slightly smaller at the size of 14pt, also bold, left-aligned and numbered.
- Sub sub Section Headings: Sub Sub Section headings are smaller than subsection headings at the size of 12pt, also bold, left-aligned and numbered.
- Highlighting: Section headings and Sub Section Headings are highlighted with bold text.
- Numbering: Section are numbered using decimal numbers. Sub Section Headings are numbered using decimal points after section heading numbers.
- Lists: For ordered lists, decimal numbers are used and font is intended.
  For unordered lists, bullet points are used and font is intended.

A.1.3 Intended Audience and Reading Suggestions

The Software Requirements Specification (SRS) document is meant for the project evaluation team, project supervisor, and project guide who are responsible for evaluating the team's work.

To ensure that all project requirements have been addressed, it is recommended that the evaluation team carefully review this document. The SRS document contains a comprehensive overview of the project's scope, including functional and non-functional requirements, design and implementation constraints.
Technical terms and acronyms used throughout the document are defined in the Glossary (Appendix A). Additionally, the Issues List (Appendix B) provides a dynamic list of unresolved requirements issues, pending decisions, and conflicts that require resolution.

A.1.4 Project Scope

The specified software is an emergency alert app, notifying users and emergency contacts. It offers a panic button, real-time location sharing, and aims for swift emergency response. Objectives include user-friendly communication and minimal response time. The scope encompasses an Android app with panic button, location sharing, contact and profile management, and Google Maps integration. The project excludes hardware or software for non-Android devices.

A.1.5 References

1. **GeoReach Article:**
   `https://hypersense-software.com/projects/georeach-emergency-app`
2. **VithU Article:**
   https://www.novelwebcreation.com/blog/for-women-safety-v-channel-launched-vithu-mobile-app
3. **BSafe Documentation:**
   `http://ijcem.in/wp-content/uploads/2014/10/BSafe-BSecure.pdf`

A.2 Overall Description

A.2.1 Product Perspective

The emergency alert application is a new, self-contained product that is designed to replace existing emergency alert systems. The application uses technology to quickly notify nearby users, emergency contacts, and emergency services in real-time during an emergency. It sends notifications with location information and other relevant details using a panic button, making it easier for people to request for help they need when they need it most.

A.2.2 Product Features

The major features of the emergency alert application include:

- Personal profile creation with emergency information like contacts, blood type, and other pertinent medical details.
- Easily accessible and reliable panic button.
- Real-time location sharing to alert nearby users, emergency contacts, and emergency services.
- Google Maps integration to view the current location of the person in emergency.
- Facilities to report false alarms and prevent the sending of notifications that may cause panic and unnecessary emergency responses.
- Security features to protect user's personal data and prevent unauthorized entry.
- Intuitive and user-friendly interface.
- Strong user authentication mechanism to ensure that only authorized users can access the system and send emergency notifications.
- Rate limiting mechanisms to restrict the number of requests that can be made from a single IP address in a given period of time.

A.2.3 User Classes and Characteristics

The various user classes that we anticipate using the emergency alert application include:

- General public: Anyone in the area who needs emergency assistance can use the system to send an alert and request help.
- Nearby Users: Individuals who are nearby the location of a personal emergency will receive an emergency notification.
- Family and friends of the person in distress: Emergency contacts can receive alerts from the system to notify them of an emergency
- Emergency responders: Police, fire, and medical personnel can use the system to receive alerts and coordinate their response to emergencies.
- Healthcare providers: Medical professionals, including hospitals and clinics, can use the system to receive medical information about patients in emergency situations.

A.2.4 Operating Environment

The emergency alert application operates on Android mobile devices with version 10.0 and above, utilizing an internet connection and GPS location services to function optimally. The application is designed to provide a reliable user experience, taking advantage of the latest Android OS features while ensuring the protection of user data with security measures.

A.2.5 Design and Implementation Constraints

1. **Hardware Constraints:**
    - Software should be designed for smartphones with android 10.0 and above Operating System
2. **Security Considerations:**
    - Data Privacy Regulations Compliance: The emergency alert application must comply with all relevant data privacy regulations to ensure the protection of user data and maintain the confidentiality of sensitive information.
    - Reliability in Emergency Situations: The emergency alert application must be reliable and able to function in emergency situations, providing accurate and timely information to nearby users, emergency contacts and emergency services in times of crisis.
3. **Technical Constraints:**

- Front end of the application should be made for android application with flutter framework.
- Back end of the application should be made with Node.js with express.js framework.
- MongoDB will be used as database.
- The system should use RESTful API for client server communication.

4. **Design constraints:**

   - Intuitive and User-Friendly Interface: The emergency alert application must provide an intuitive and user-friendly interface that is easy to navigate and understand, even for users with limited technical knowledge or experience.
   - The software should follow android material UI.

A.2.6 User Documentation

User documentation components that will be delivered with the emergency alert application include:

- Application Demonstration: A detailed demonstration of the application's functionality will be provided to users. This demonstration will encompass all the essential features and functionalities of the application, including how to navigate through the application and how to configure and set up the emergency alert system.
- Frequently Asked Questions (FAQs): A section dedicated to addressing common questions and concerns that users may have about the application will be included. This section will cover a range of topics, including troubleshooting, system requirements, and how to use the application effectively.
- User Manual: A comprehensive user manual that will provide detailed information on the application's features and functionalities, as well as best practices for optimizing its performance.

A.2.7 Assumptions and Dependencies

Assumptions and dependencies for the emergency alert application project include:

1. **Assumptions:**

- Reliable internet and GPS services are essential for the app to send and receive notifications accurately. Users are assumed to have access to these services where the app is used.
- Mobile devices with compatible Android operating systems are assumed for using the emergency alert app.
- Assumed is the cooperation of emergency responders and healthcare professionals, who will use the app to access user information for timely assistance in emergencies.
- Basic tech understanding is assumed for users to navigate emergency alert app.
- Accurate user info provision is assumed for medical condition, location, and contacts in the emergency alert app.
- The app's effectiveness relies on emergency service availability in the user's location, assuming responders are accessible.
- Assumed is integration with third-party apps like Google Maps; development will address compatibility.

2. **Dependencies:**

- Google Maps API: The emergency response system is dependent on the Google Maps API to provide accurate location information for users and emergency responders.
- SMS Services: The system requires integration with a third-party SMS service provider to ensure reliable delivery of emergency alerts via SMS.
- Android API: The system requires integration with Android APIs to enable real-time location tracking and to receive push notifications.

A.3 System Features

A.3.1 Alert Button

A.3.1.1 Description and Priority

A big, easy-to-access button on the app's home screen that sends an emergency alert to nearby users, authorities, and emergency contacts when pressed.

- High Priority

A.3.1.2 Stimulus/Response Sequences

User pressing the alert button from notification bar or from inside the application triggers the emergency alert

A.3.1.3 Functional Requirements

- The Alert Button should be prominently displayed on the home screen of the app.
- The Alert Button should be easily accessible with a single tap.
- The Alert Button should trigger the emergency alert 5 seconds after being pressed.
- 5 second time is given to cancel the emergency alert
- alert button should also be shown as a permanent notification on lock screen for easy access.

A.3.2 Push notifications

A.3.2.1 Description and Priority

The app should use push notifications to notify users of emergency situations even when the app is not open.

- High Priority

A.3.2.2 Stimulus/Response Sequences

User pressing the alert button, application triggers push notification in other users

A.3.2.3 Functional Requirements

- Push notifications should be shown in all situations, including in lock screen.
- Push notification should have an alert sound.
- Notification should be easily noticeable and clear to view.

A.3.3 User Profile

A.3.3.1 Description and Priority

user profile includes emergency contact information, medical information, location

- High priority

A.3.3.2 Stimulus/Response Sequences

User will be creating an emergency profile upon registration

A.3.3.3 Functional Requirements

- User profile should contain emergency contact, emergency contacts will be notified when the user is pressing alert button.
- User profile should contain relevant medical information which may be useful in case of an emergency.
- User profile should be created at registration time and should be able to update/edit details after registration.
- Details in user profile should be sent upon pressing the emergency button.

A.3.4 Map Integration

A.3.4.1 Description and Priority

Google Map integration to view current location of the person in emergency

- Moderate priority

A.3.4.2 Stimulus/Response Sequences

Real time location will be sent along with emergency alerts.

A.3.4.3 Functional Requirements

- The app should use GPS and other location technologies to track the user's location.
- Location should be accurate as possible.
- Location should be real-time and should be updated in specified interval of time.

A.3.5 False alarm detection

A.3.5.1 Description and Priority

Report false alarms and prevent the sending of notifications that may cause panic and unnecessary emergency responses.

- Moderate priority

A.3.5.2 Stimulus/Response Sequences

False alarms can be flagged inside the application

A.3.5.3 Functional Requirements

- There should be an option to flag false alarms inside the application
- False alarms should be removed from alerts list
- False should not be allowed to cause panic

A.3.6 Location-Based Notifications

A.3.6.1 Description and Priority

The app should use the user's current location to send notifications to nearby users in the event of an emergency

- Moderate priority

A.3.6.2 Stimulus/Response Sequences

Notification is to be sent when the alert button is pressed

A.3.6.3 Functional Requirements

- Notification should contain real-time location of the user
- Relevant medical and user information should be in the notification
- Notification should not be delayed

A.3.7 Emergency contacts

A.3.7.1 Description and Priority

Integrate emergency contact information, such as phone number, to receive notifications in the event of an emergency.

- Moderate priority

A.3.7.2 Stimulus/Response Sequences

Notification is sent to emergency contacts when the alert button is pressed

A.3.7.3 Functional Requirements

- User should be able to create emergency contacts
- Emergency contacts should have phone number and other details to help user

A.4 External Interface Requirements

A.4.1 User Interfaces

- Interface is in android material Design
- User Interface include:
  - Registration page: first time users will be able to register
  - Login page: registered users can log in from login page
  - User profile page: after registration user will be able to create user profile will emergency contacts and relevant medical information
  - Alert Button: Alert button inside the app will be used to send emergency alerts
  - Notification Button: Emergency alert button in lock screen notification bar for easy access

A.4.2 Hardware Interfaces

- Supported Hardware: Android 10+
- Hardware with GPS and Wi-Fi support
- Wi-Fi and cellular information for location services
- Access the device's GPS for location services
- Cellular data access for SMS services and network connection

A.4.3 Software Interfaces

- Node.js 18.15.0 : As Backend
- NPM 9.5.0
- MongoDB 6.0 : As Database
- Flutter 3.7.11
- RESTful API: For data transfer between client and server

A.4.4 Communications Interfaces

- Internet connection is required for client to server communication
- Data transfer between client and server will through HTTPS
- Use secure communication protocols, such as SSL or TLS, to protect user data and ensure privacy.
- SMS will be used to notify nearby users, authorities, and emergency contacts.
- The app will use the HTTP protocol for communication between the front end and back end components.
- RESTful API will be used for communication between the app and external APIs.

A.5 Other Nonfunctional Requirements

A.5.1 Performance Requirements

1. Scalability: System must handle numerous alerts and users simultaneously with minimal performance impact.
2. Reliability: System should manage diverse network conditions, including low bandwidth and intermittent connectivity. It must send notifications upon device network reconnection during unavailability.
3. Response Time: The emergency response system should be able to send emergency alerts within a specified time-frame to ensure timely response to emergencies.
4. Concurrency: System should be able to manage multiple alerts concurrently
5. Real-time updates: Location information should be updated in real-time

A.5.2 Safety Requirements

1. False Alerts: False alerts should be flagged to avoid unnecessary panic
2. Data Privacy: The app must abide by all applicable data privacy laws and rules, including the California Consumer Privacy Act (CCPA) and the General Data Protection Regulation (GDPR). User information must be safely kept in the database and transmitted using encryption.

A.5.3 Security Requirements

1. The emergency response system should use encryption to protect sensitive data, such as medical information and emergency contact information.
2. The emergency response system should be able to authenticate users and ensure that only authorized users can send emergency alerts.
3. The emergency response system should use secure communication protocols, such as SSL or TLS, to protect user data and ensure privacy.
4. Rate limiting mechanisms to restrict the number of requests that can be made from a single IP address in a given period of time, to prevent Denial of Service attacks

A.5.4 Software Quality Attributes

1. Scalability: The emergency response system should efficiently scale to accommodate more users, alerts, and components while maintaining performance. This involves load balancing, scaling, and fault tolerance.
2. Portability: The emergency response system should run on various hardware devices with minimal adjustments.
3. Reliability: Should perform accurately, consistently, and without errors, including fault tolerance and redundancy.
4. Security: Secure user data with encryption, access control, and auditing for confidentiality, integrity, and availability.
5. Testability: Enable comprehensive testing for correctness, reliability, and performance through test automation, component testability, and simplified test environment setup.

A.6 Issues List

- Function to send emergency alert to authorities is not implementing.
- Using android gestures to trigger the emergency alert is not implementing.

References

[1] Georeach. Emergency app. https://hypersense-software.com/projects/georeach-emergency-app.

[2] VithU. Emergency app. https://www.novelwebcreation.com/blog/for-women-safety-v-channel-launched-vithu-mobile-app.

[3] Bsafe. Emergency app. http://ijcem.in/wp-content/uploads/2014/10/BSafe-BSecure.pdf.

[4] Dina M. Ibrahim et al. Safety souls mobile application for emergency response system. https://www.researchgate.net/publication/343480471_Safety_Souls_Mobile_Application_for_Emergency_Response_System, 2020.

[5] Ian Sommerville. *Software Engineering (tenth edn. global edition)*. Boston, MA: Pearson Education, Inc, 2016.

[6] Roger S Pressman. *Software engineering: a practitioner's approach*. Palgrave macmillan, 2005.

[7] Flutter documentation. https://docs.flutter.dev.

[8] Node documentation. https://nodejs.org/en/docs.

[9] Mongodb documentation. https://www.mongodb.com/docs.

[10] Firebase documentation. https://firebase.google.com/docs/guides.

Index