# Table of contents

# Abstract

**CrisisConnect** is an innovative web application designed to streamline the process of requesting and dispatching emergency services. Developed using the MERN (MongoDB, Express, React, Node.js) stack, CrisisConnect offers a reliable and user-friendly platform where individuals can quickly seek assistance in times of crisis. The application supports four types of emergency services: fire, police, hospital, and medical support.

Upon registration, users can log in to their personalized dashboard, where they can choose the type of emergency assistance they need. The system allows users to specify whether they require help at their current location or their registered home location. Once a request is submitted, it is relayed to the admin panel, where administrators can view and manage incoming requests. Administrators then coordinate with the nearest emergency response teams—such as police stations, fire brigades, hospitals, or medical shops—to dispatch the necessary help. Users receive confirmation emails once their requests are processed, ensuring they are kept informed throughout the process.

CrisisConnect also features a real-time live chat system, enabling users to communicate directly with administrators for immediate support and guidance. Additionally, the platform incorporates a donation system, allowing users to contribute to the maintenance and enhancement of CrisisConnect via QR code scanning.

With a focus on security, CrisisConnect employs robust authentication mechanisms and encrypted data transmission to protect user information. The application is designed to be scalable, secure, and user-centric, aiming to provide a vital service that can make a significant difference during emergencies. Future plans include expanding the platform to mobile devices and integrating advanced GPS tracking for more precise location services.

# Introduction

In today's fast-paced world, emergencies can arise at any moment, requiring swift and effective responses to ensure safety and well-being. However, coordinating emergency services such as fire, police, medical, and hospital assistance can often be cumbersome and time-consuming. To address these challenges, we present **CrisisConnect**, a comprehensive web application developed using the MERN (MongoDB, Express, React, Node.js) stack. CrisisConnect aims to provide a seamless and efficient platform for users to request and receive emergency assistance promptly and reliably.

CrisisConnect is designed with both users and administrators in mind. Users can easily register on the platform and gain access to a personalized dashboard where they can request various types of emergency support. The intuitive interface guides users through specifying their emergency needs and choosing whether assistance is required at their current or home location. Once a request is submitted, the system ensures that it is immediately visible to administrators who can then coordinate with the appropriate emergency services to dispatch help.

The administrative side of CrisisConnect features a powerful dashboard where administrators can manage and track user requests. This centralized system allows for quick decision-making and efficient coordination with local emergency response units. Additionally, the platform includes a live chat system, enabling real-time communication between users and administrators, further enhancing the responsiveness and effectiveness of the service.

Beyond emergency support, CrisisConnect also offers a donation feature, allowing users to contribute financially to the platform. Donations are facilitated through a simple and secure QR code scanning system, making it easy for users to support the ongoing development and maintenance of CrisisConnect.

CrisisConnect is built with robust security measures to protect user data and ensure the integrity of the system. With encrypted data transmission and secure authentication protocols, users can trust that their personal information is safeguarded.

In summary, CrisisConnect is a vital tool designed to bridge the gap between those in need of emergency assistance and the services capable of providing help. By leveraging modern web technologies and focusing on user-centric design, CrisisConnect aims to enhance the efficiency and effectiveness of emergency response, ultimately saving lives and improving community safety.

# Objective

The primary objective of **CrisisConnect** is to create a reliable, efficient, and user-friendly web application that facilitates the prompt coordination and delivery of emergency services. By leveraging the MERN (MongoDB, Express, React, Node.js) stack, CrisisConnect aims to achieve the following specific goals:

1. **Streamline Emergency Requests:**

- Provide an intuitive interface for users to quickly request assistance for fire, police, hospital, and medical emergencies.

- Enable users to specify their current or home location to ensure accurate dispatch of services.

2. **Enhance Administrative Efficiency:**

- Develop a comprehensive admin panel that allows administrators to view, manage, and respond to user requests in real-time.

- Facilitate coordination with local emergency response units, ensuring timely dispatch of assistance.

3. **Improve Communication:**

- Integrate a live chat system to enable real-time communication between users and administrators, providing immediate support and guidance during emergencies.

4. **Ensure User Safety and Data Security:**

- Implement robust security measures, including encrypted data transmission and secure authentication protocols, to protect user information and maintain system integrity.

5. **Enable Community Support:**

- Incorporate a donation feature, allowing users to financially support CrisisConnect through a secure QR code scanning system, ensuring the platform's sustainability and continuous improvement.

By achieving these objectives, CrisisConnect aims to significantly improve the efficiency and effectiveness of emergency response, providing a crucial service that enhances community safety and saves lives.

## Scope:

The scope of the **CrisisConnect** project encompasses the development of a comprehensive web application aimed at facilitating the seamless coordination and delivery of emergency services. The project will focus on the following key aspects:

1. **User Management:**

- Implement user registration and login functionalities, allowing individuals to create accounts and access the platform securely.

- Develop user profiles to store personal information and preferences for efficient emergency response.

2. **Dashboard and Request System:**

- Design an intuitive user dashboard where individuals can easily request emergency assistance for fire, police, hospital, and medical emergencies.

- Enable users to specify the type of assistance needed and their location (current or home) to ensure accurate dispatch of services.

3. **Admin Panel:**

- Create an administrative interface with features for viewing, managing, and responding to user requests in real-time.

- Implement functionalities for administrators to coordinate with local emergency response units and dispatch assistance promptly.

4. **Live Chat System:**

- Integrate a real-time chat system to facilitate communication between users and administrators, enabling immediate support and guidance during emergencies.

5. **Donation Feature:**

- Incorporate a donation mechanism to allow users to contribute financially to CrisisConnect's sustainability and continuous improvement.

- Develop a secure QR code scanning system for streamlined donation processing.

6. **Security and Data Protection:**

- Implement robust security measures, including encrypted data transmission and secure authentication protocols, to safeguard user information and ensure system integrity.

7. **Scalability and Future Enhancements:**

- Design the application architecture to be scalable, allowing for potential future enhancements such as mobile app integration and advanced GPS tracking for precise location services.

# Theoretical Background

The choice of the MERN stack (MongoDB, Express.js, React.js, Node.js) for **CrisisConnect** is grounded in its suitability for developing modern, dynamic web applications. Each component of the MERN stack offers unique advantages that contribute to the overall effectiveness and scalability of the project.

1. **MongoDB (M):** MongoDB is a NoSQL database that provides a flexible and scalable solution for storing and managing data. Its document-based model allows for easy handling of complex data structures, making it well-suited for applications with diverse data requirements. In CrisisConnect, MongoDB serves as the backend database, efficiently storing user details, emergency requests, and administrative data.

2. **Express.js (E):** Express.js is a lightweight and flexible web application framework for Node.js. It simplifies the process of building web applications by providing a robust set of features for routing, middleware, and HTTP request handling. Express.js facilitates the development of RESTful APIs, enabling seamless communication between the frontend and backend components of CrisisConnect.

3. **React.js (R):** React.js is a popular JavaScript library for building user interfaces. Its component-based architecture and virtual DOM (Document Object Model) enable developers to create dynamic and interactive user interfaces with ease. React.js is well-suited for single-page applications (SPAs) like CrisisConnect, offering enhanced performance and scalability compared to traditional server-rendered approaches.

4. **Node.js (N):** Node.js is a runtime environment that allows developers to run JavaScript code on the server-side. It provides a non-blocking, event-driven architecture that is highly efficient for handling asynchronous operations, such as I/O tasks and network requests. Node.js powers the backend of CrisisConnect, enabling real-time communication, data processing, and business logic implementation.

**Advantages of Using the MERN Stack for CrisisConnect:**

1. **Full Stack JavaScript Development:** With the MERN stack, developers can leverage their proficiency in JavaScript across the entire application stack, from frontend to backend. This streamlines development workflows and promotes code reusability, resulting in faster development cycles and easier maintenance.

2**. Scalability and Flexibility:** MongoDB's schema-less design and Node.js's non-blocking I/O model enable CrisisConnect to scale seamlessly to accommodate growing user bases and evolving data requirements. The flexibility of the MERN stack allows for easy adaptation to changing project needs and future enhancements.

3. **Performance Optimization:** React.js's virtual DOM and efficient rendering mechanisms contribute to a smoother user experience, minimizing page reloads and enhancing responsiveness. Combined with Node.js's lightweight architecture and Express.js's streamlined routing capabilities, CrisisConnect achieves optimal performance even under high traffic loads.

4. **Community Support and Ecosystem:** The MERN stack benefits from a vibrant and active developer community, with extensive documentation, tutorials, and third-party libraries available for building and extending web applications. This rich ecosystem accelerates development and provides access to a wide range of tools and resources for enhancing CrisisConnect's functionality.

# Problem Definition

**CrisisConnect** addresses several critical issues inherent in traditional emergency service response systems, aiming to overcome challenges such as:

1. **Lack of Efficient Communication Channels:** Traditional emergency response systems often rely on phone calls or physical visits to request assistance, leading to delays and inefficiencies in communication. CrisisConnect seeks to establish streamlined communication channels between users in distress and emergency service providers, enabling prompt and effective coordination during critical situations.

2. **Inaccurate Location Identification:** In emergency situations, accurately identifying the user's location is crucial for dispatching aid resources in a timely manner. Conventional methods for determining location, such as verbal descriptions or manual address entry, may be prone to errors or delays. CrisisConnect integrates geolocation services to accurately pinpoint the user's location, facilitating rapid deployment of assistance based on proximity to the emergency.

3. **Limited Visibility and Coordination for Administrators:** Emergency service administrators often face challenges in monitoring and coordinating response efforts across various incidents. Without a centralized platform for tracking incoming requests and managing resources, administrators may struggle to prioritize tasks and allocate resources efficiently. CrisisConnect provides administrators with a dedicated dashboard for monitoring incoming requests, coordinating response efforts, and communicating with users in real-time, enhancing visibility and coordination during emergency situations.

4. **Data Security and Privacy Concerns:** Traditional emergency response systems may lack robust mechanisms for protecting user data and ensuring privacy compliance. In an era of increasing cybersecurity threats and privacy regulations, safeguarding sensitive information is paramount. CrisisConnect implements robust security measures, including encryption protocols, access controls, and data validation mechanisms, to protect user information and ensure compliance with privacy regulations.

5. **Limited Accessibility and Reach:** Traditional emergency response systems may be inaccessible to certain populations or communities, particularly those in remote or underserved areas. Additionally, language barriers or technological limitations may further impede access to emergency services. CrisisConnect aims to bridge these accessibility gaps by providing a user-friendly web application accessible from any device with internet connectivity, ensuring that individuals in need can easily request assistance regardless of their location or circumstances.

# System Analysis and Design

To ensure the effective development of **CrisisConnect**, a comprehensive system analysis and design process is conducted based on user requirements.

The following outlines the key steps and components of this process:

1. **Requirement Elicitation:**

- Conduct interviews, surveys, and workshops with stakeholders, including potential users, administrators, and emergency service providers, to gather requirements.

- Identify the primary functionalities and features desired by users, such as registration, emergency request submission, real-time communication, and administrator dashboard.

2. **Requirement Analysis:**

- Analyze the gathered requirements to identify user needs, system constraints, and business objectives.

- Prioritize requirements based on their importance and feasibility, considering factors such as user impact, technical complexity, and project timeline.

- Document functional requirements, including use cases, user stories, and system specifications, to guide the design and development process.

3. **System Design:**

- Design the architecture and components of CrisisConnect based on the analyzed requirements.

- Define the system's overall structure, including frontend (user interface), backend (server-side logic), and database components.

- Select appropriate technologies and frameworks, such as the MERN stack (MongoDB, Express.js, React.js, Node.js), for implementing different system modules.

- Create wireframes, mockups, and prototypes to visualize the user interface and user interactions, ensuring alignment with user expectations and usability principles.

4. **Database Design:**

- Design the database schema to efficiently store and manage user data, emergency requests, administrative actions, and other relevant information.

- Define relationships between different data entities and establish data integrity constraints to maintain consistency and accuracy.

- Select appropriate database technologies, such as MongoDB, and optimize data storage and retrieval operations for performance and scalability.

5. **User Interface Design:**

- Design the user interface (UI) of CrisisConnect to be intuitive, user-friendly, and accessible across different devices and screen sizes.

- Incorporate usability principles and best practices to enhance user experience and facilitate task completion.

- Create UI mockups and prototypes to validate design concepts and gather feedback from stakeholders before implementation.

6. **System Prototyping:**

- Develop prototypes or minimum viable products (MVPs) to validate system functionalities and gather user feedback iteratively.

- Implement core features and workflows to demonstrate the system's capabilities and solicit feedback from users and stakeholders.

- Iterate on the prototypes based on user feedback and refine the design and implementation as needed to address identified issues and improve user satisfaction.

7. **System Testing:**

- Conduct thorough testing of the system to validate its functionality, reliability, and performance.

- Perform unit testing, integration testing, and system testing to identify and address any defects or inconsistencies in the system.

- Implement automated testing frameworks, such as Jest and React Testing Library, to streamline the testing process and ensure comprehensive test coverage.

8. **Deployment and Maintenance:**

- Deploy the finalized version of CrisisConnect on a production environment, such as a web server or cloud platform, to make it accessible to users and administrators.

- Monitor system performance, usage metrics, and user feedback to identify areas for improvement and address any issues that arise post-deployment.

- Implement a maintenance plan to provide ongoing support, updates, and enhancements to the system, ensuring its continued effectiveness and relevance over time.

## Feasibility Study

A feasibility study is conducted to assess the viability and practicality of implementing the CrisisConnect project. This study evaluates various aspects, including technical, economic, operational, and schedule feasibility, to determine whether the project is feasible and worth pursuing.

Certainly! Here's a feasibility study for the CrisisConnect project:

1. **Technical Feasibility:**

- Platform Compatibility: The MERN stack (MongoDB, Express.js, React.js, Node.js) is widely used and offers robust support for building web applications. All the technologies required for the project are well-documented and have active developer communities, making it feasible from a technical standpoint.

- Scalability: The architecture of the MERN stack allows for scalability, making it suitable for handling potentially large numbers of users and emergency requests. MongoDB's flexible schema and horizontal scaling capabilities can accommodate growth in data and traffic.

- Integration with APIs: Integrating with APIs to obtain location-based services and communication with emergency services (police stations, fire brigades, hospitals) is technically feasible. Many APIs are available for these purposes, and they can be integrated into the application's backend.

2. **Operational Feasibility:**

- User Adoption: The concept of CrisisConnect addresses a critical need for emergency assistance services. Given the importance of such services, there is likely to be a high level of user adoption, especially in areas where emergency response infrastructure may be lacking or inefficient.

- Ease of Use: The user interface of CrisisConnect will be designed to be intuitive and user-friendly, catering to individuals in distressful situations. Proper training and onboarding materials can further enhance the ease of use for both users and administrators.

- Administrative Effort: While managing emergency requests and coordinating with emergency services will require administrative effort, the streamlined workflow and automation features of CrisisConnect can minimize the burden on administrators. Efficient processes and clear communication channels will facilitate smooth operations.

3. **Economic Feasibility:**

- Development Costs: The development costs for CrisisConnect will primarily involve the allocation of resources for software development, including personnel, infrastructure, and software licenses. However, the use of open-source technologies within the MERN stack can mitigate some of these costs.

- Operational Costs: Operational costs will include hosting expenses, maintenance, and support. These costs can vary depending on factors such as the scale of the application and the chosen hosting provider. However, they are generally manageable and can be offset by the benefits provided by the application.

- Revenue Generation: While CrisisConnect primarily focuses on providing a public service, there may be opportunities for revenue generation through partnerships, sponsorships, or premium features. Monetization strategies can be explored in the future to ensure the sustainability of the platform.

## System Planning

**Project Life Cycle Used: Iterative Waterfall Model**

1. **Project Initiation:**

- Define the project objectives and scope, focusing on the development of CrisisConnect, an emergency service web application.

- Identify key stakeholders, including users, administrators, emergency service providers, and regulatory authorities.

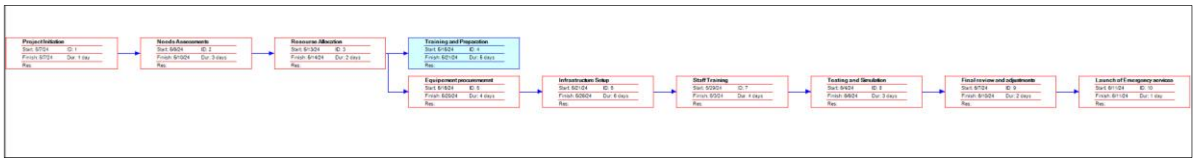
- Conduct initial feasibility analysis to assess the technical, operational, economic, and legal aspects of the project.

- Establish project governance structure, roles, and responsibilities.

2. **System Planning:**

- Develop a comprehensive project plan outlining the activities, milestones, and deliverables for each phase of the iterative waterfall model.

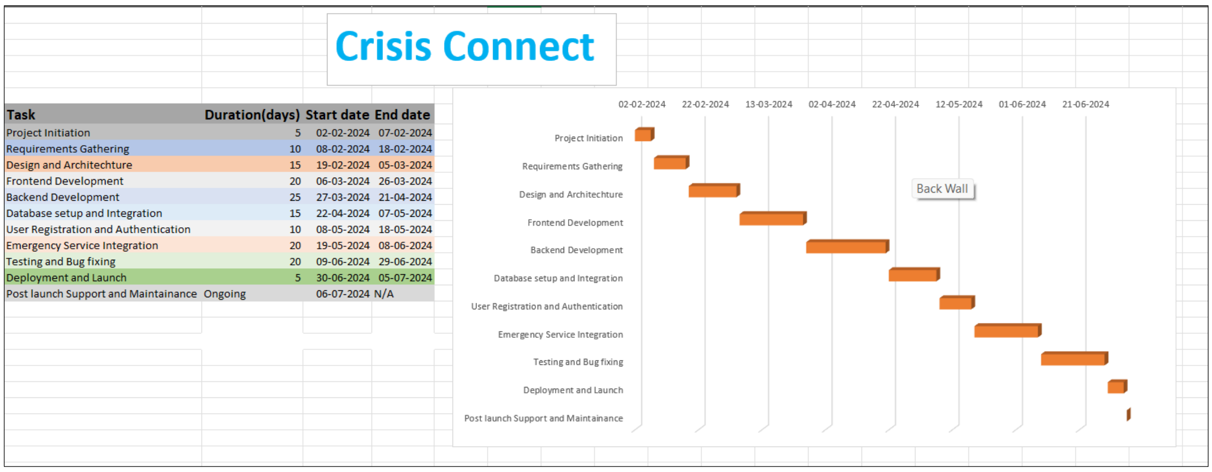
- Create a PERT (Program Evaluation and Review Technique) chart to visualize the sequence of tasks, dependencies, and critical paths.



PERT Chart

3. **Gantt Chart:**

- Translate the PERT chart into a Gantt chart, depicting the timeline for project activities, including analysis, design, development, testing, deployment, and maintenance phases.


Crisis Connect

| Task | Duration(days) | Start date | End date |
|---|---|---|---|
| Project Initiation | 5 | 02-02-2024 | 07-02-2024 |
| Requirements Gathering | 10 | 08-02-2024 | 18-02-2024 |
| Design and Architechture | 15 | 19-02-2024 | 05-03-2024 |
| Frontend Development | 20 | 06-03-2024 | 26-03-2024 |
| Backend Development | 25 | 27-03-2024 | 21-04-2024 |
| Database setup and Integration | 15 | 22-04-2024 | 07-05-2024 |
| User Registration and Authentication | 10 | 08-05-2024 | 18-05-2024 |
| Emergency Service Integration | 20 | 19-05-2024 | 08-06-2024 |
| Testing and Bug fixing | 20 | 09-06-2024 | 29-06-2024 |
| Deployment and Launch | 5 | 30-06-2024 | 05-07-2024 |
| Post launch Support and Maintainance | Ongoing | 06-07-2024 | N/A |

Gnatt Chart

4. **Requirements Analysis:**

- Gather and analyze requirements from stakeholders, focusing on user needs, emergency scenarios, system functionalities, and administrative capabilities.

- Document functional and non-functional requirements, prioritizing them based on importance and feasibility.

- Create use cases, user stories, and system requirements specifications to capture the desired behavior and features of CrisisConnect.

5. **System Design:**

- Design the system architecture, including database schema, backend APIs, frontend components, and user interfaces.

- Develop wireframes and mockups to visualize the layout and flow of the application.

- Define data models, API endpoints, authentication mechanisms, and communication protocols.

6. **Implementation:**

- Develop the CrisisConnect application iteratively, following the design specifications and coding standards.

- Implement frontend components using React.js for the user interface and user interactions.

- Build backend APIs and business logic using Node.js and Express.js, integrating with MongoDB for data storage and retrieval.

7. **Testing:**

- Conduct unit testing, integration testing, and system testing at each iteration to ensure the reliability, functionality, and performance of the application.

- Perform user acceptance testing (UAT) with stakeholders to validate that the system meets their requirements and expectations.

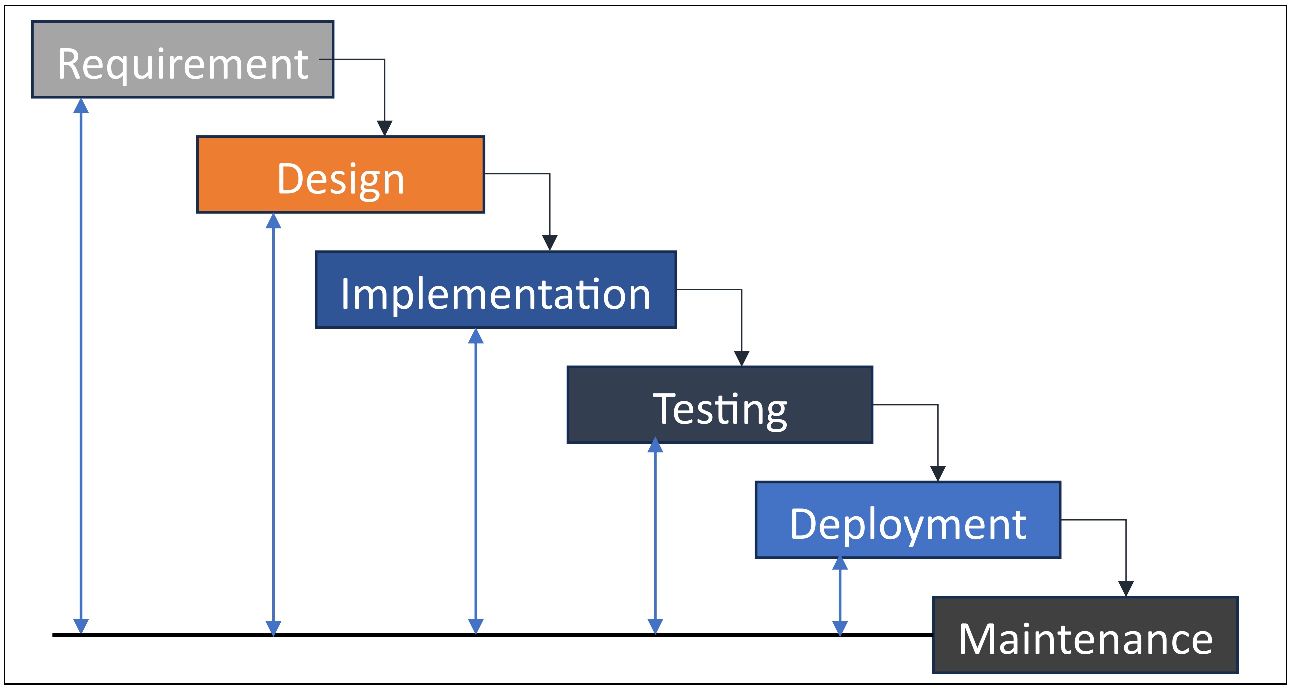
- Address any defects or issues identified during testing and refine the application accordingly.

8. **Deployment:**

- Deploy CrisisConnect to a production environment, configuring servers, databases, and networking infrastructure as needed.

- Conduct deployment testing to verify that the application functions correctly in the production environment.

- Monitor system performance, scalability, and security post-deployment, implementing any necessary optimizations or updates.



Iterative Waterfall Model

# Software Requirement Specification

## Functional Requirements :

1. **User Registration and Authentication:**

- Users should be able to register for an account by providing necessary personal details such as name, email, and password.

- The system should authenticate users securely during the login process, verifying credentials against the stored user database.

- Password reset functionality should be provided to allow users to recover their accounts in case of forgotten passwords.

2. **Dashboard for Users:**

- Upon successful login, users should be directed to a dashboard where they can access emergency support services.

- The dashboard should prominently display options for requesting assistance in various emergency scenarios, such as fire, police, hospital, and medical help.

3. **Emergency Service Requests:**

- Users should be able to initiate emergency service requests by selecting the type of assistance needed (fire, police, hospital, medical).

- For each type of emergency, users should have the option to choose whether they need assistance at their current location or home location.

- Upon selecting the location, users should be prompted to confirm the request, triggering the dispatch of emergency services to the specified location.

4. **Admin Panel:**

- Administrators should have access to a dedicated admin panel where they can view and manage incoming emergency service requests.

- The admin panel should provide functionalities for filtering, sorting, and searching requests based on various criteria such as type, location, and status.

- Administrators should be able to review details of each request, including user information, request timestamp, and location coordinates.

5. **Emergency Service Coordination:**

- Upon receiving an emergency service request, administrators should be able to coordinate with relevant emergency service providers (police stations, fire brigades, hospitals, medical shops) in the user's location.

- The system should facilitate communication between administrators and emergency service providers, enabling them to dispatch assistance promptly and efficiently.

6. **Location-based Services:**

- The system should utilize location-based services (e.g., GPS or geocoding APIs) to determine the user's current location accurately.

- Users should have the option to manually enter their home location or allow the system to detect it automatically based on their device's GPS coordinates.

8. **Security and Data Privacy:**

- The system should implement robust security measures to protect user data and prevent unauthorized access.

- User authentication and data transmission should be encrypted using industry-standard protocols (e.g., HTTPS).

- Personal and sensitive information collected from users should be handled in compliance with relevant data protection regulations.

9. **Real-time Chat Service :**

- User can communicate directly with the admin via real-time chat support.

## Non-Functional Requirements :

1. **Performance:**

- The system should be responsive and capable of handling simultaneous requests from multiple users without significant delays.

- Response times for critical functions such as user authentication, emergency service requests, and notifications should be minimized.

- The application should be optimized for efficient use of system resources to ensure scalability and reliability under heavy loads.

2. **Reliability:**

- The system should be highly reliable, with minimal downtime and disruptions to service availability.

- Data integrity and consistency should be maintained at all times, with mechanisms in place to prevent data loss or corruption.

- Backup and recovery procedures should be implemented to safeguard against potential failures or disasters.

3. **Security:**

- The system should adhere to industry best practices for security, protecting against common threats such as unauthorized access, data breaches, and cyberattacks.

- User authentication and authorization mechanisms should be robust, with support for secure password hashing, session management, and role-based access control.

- Personal and sensitive user data should be encrypted both in transit and at rest to prevent unauthorized disclosure.

4. **Scalability:**

- The architecture of the system should be designed to scale horizontally to accommodate increasing numbers of users and emergency service requests.

- Load balancing mechanisms should be implemented to distribute traffic evenly across multiple servers or instances to prevent bottlenecks.

- Database scalability should be ensured through techniques such as sharding or replication to handle growing volumes of data.

5. **Usability:**

- The user interface should be intuitive and user-friendly, with clear navigation paths and informative feedback messages.

- Accessibility features should be implemented to ensure that the application is usable by individuals with disabilities, adhering to relevant accessibility standards (e.g., WCAG).

- The application should be compatible with a wide range of devices and screen sizes, providing a consistent user experience across different platforms.

6. **Maintainability:**

- The codebase should be well-organized, modular, and thoroughly documented to facilitate ease of maintenance and future enhancements.

- Version control systems such as Git should be used to track changes to the codebase and facilitate collaboration among developers.

- Automated testing suites should be implemented to ensure code quality and detect regressions during development and maintenance cycles.

## Development Platform:

The development platform for CrisisConnect involves selecting the tools, technologies, and frameworks necessary for building and deploying the web application.

Given the requirements and objectives of CrisisConnect, here are the components of the development platform:

1. **Frontend Development:**

- React.js: React.js will serve as the primary frontend library for building the user interface of CrisisConnect. Its component-based architecture and virtual DOM make it well-suited for creating dynamic and interactive UI elements.

- HTML/CSS/JavaScript: Alongside React.js, standard web technologies such as HTML, CSS, and JavaScript will be used for frontend development to style and enhance the user interface.

2. **Backend Development:**

- Node.js: Node.js will power the backend of CrisisConnect, providing a runtime environment for executing server-side JavaScript code. Its non-blocking, event-driven architecture makes it ideal for handling asynchronous operations, such as I/O tasks and network requests.

- Express.js: Express.js, a lightweight and flexible web application framework for Node.js, will be used to build the backend API layer of CrisisConnect. It simplifies routing, middleware handling, and HTTP request processing, facilitating the development of RESTful APIs.

3. **Database:**

- MongoDB: MongoDB, a NoSQL database, will be used as the backend database for CrisisConnect. Its document-based model and scalability make it well-suited for storing user data, emergency requests, and administrative actions.

4. **Development Tools :**

- Visual Studio Code: Visual Studio Code, a lightweight and versatile code editor, may be used for writing, debugging, and testing code during development.

- Git: Git, a distributed version control system, will be used for managing source code, versioning, and collaboration among development team members.

By leveraging these tools, technologies, and frameworks, CrisisConnect can be developed efficiently, ensuring scalability, reliability, and performance in providing emergency service provision.

## Roles of tools in development :

In the development of **CrisisConnect**, various tools play specific roles in facilitating different aspects of the development process.

Here are the roles of the key tools used in the project:

1. **Visual Studio Code (VS Code):**

- Role: VS Code serves as the primary integrated development environment (IDE) for writing, editing, and debugging code across frontend and backend components of CrisisConnect.

- Features: It provides syntax highlighting, code completion, and debugging tools for JavaScript, HTML, CSS, and other programming languages used in the project.

- Role in Development: VS Code streamlines the coding process, improves productivity, and ensures consistency in code formatting and style among development team members.

2. **Git and GitHub:**

- Role: Git is a distributed version control system used for managing source code, tracking changes, and enabling collaboration among developers. GitHub is a web-based platform that hosts Git repositories and provides additional collaboration features.

- Features: Git allows developers to create branches, commit changes, merge code, and resolve conflicts, while GitHub facilitates code review, issue tracking, and project management.

- Role in Development: Git and GitHub enable version control, code sharing, and teamwork, ensuring that developers can work on different features concurrently, track changes, and coordinate their efforts effectively.

3. **React.js:**

- Role: React.js is a JavaScript library for building user interfaces, providing a component-based architecture and a virtual DOM for efficient UI rendering.

- Features: React.js allows developers to create reusable UI components, manage component state, and handle user interactions using JSX syntax.

- Role in Development: React.js powers the frontend of CrisisConnect, enabling the creation of dynamic, interactive, and responsive user interfaces that enhance user experience and engagement.

4. **Node.js and Express.js:**

- Role: Node.js is a runtime environment for executing JavaScript code on the server-side, while Express.js is a web application framework for Node.js, providing features for routing, middleware, and HTTP request handling.

- Features: Node.js enables non-blocking, event-driven I/O operations, making it well-suited for building scalable and performant server applications. Express.js simplifies the development of backend APIs, facilitating the creation of RESTful endpoints.

- Role in Development: Node.js and Express.js power the backend of CrisisConnect, handling user authentication, request processing, database interactions, and other server-side logic.

5. **MongoDB:**

- Role: MongoDB is a NoSQL database used for storing and managing data in CrisisConnect, including user details, emergency requests, and administrative actions.

- Features: MongoDB's document-based model and scalability make it suitable for handling diverse data types and accommodating the dynamic nature of emergency service data.

- Role in Development: MongoDB serves as the backend database for CrisisConnect, providing efficient data storage, retrieval, and querying capabilities for managing user data and system state.

These tools collectively play essential roles in the development lifecycle of CrisisConnect, enabling collaboration, code development, version control, deployment, and maintenance of the web application.

# Cost and benefit analysis

## Cost Analysis:

1. **Development Costs:**

- Infrastructure: Procuring hardware and software required for development, including development servers, licenses, and development tools.

- Training: Providing training sessions for team members to familiarize them with the technologies and methodologies used in the project.

2. **Operational Costs:**

- Hosting: Monthly or annual expenses for hosting the application on cloud servers or dedicated hosting services.

- Support: Budget for providing customer support, handling user inquiries, and addressing technical issues.

## Benefit Analysis:

1. **Social Impact:**

- Lives Saved: CrisisConnect has the potential to save lives by connecting individuals in emergencies with timely assistance from emergency service providers.

- Community Support: Providing a valuable public service that enhances community safety and well-being, fostering trust and cooperation among residents and authorities.

2. **Efficiency and Effectiveness:**

- Faster Response Times: By streamlining the process of requesting emergency assistance and coordinating with service providers, CrisisConnect can reduce response times and improve outcomes for individuals in distress.

- Resource Optimization: Efficient allocation of emergency resources based on real-time demand and location data, maximizing the effectiveness of emergency response efforts.

3. **Cost Savings:**

- Reduced Emergency Response Costs: By optimizing resource allocation and minimizing delays, CrisisConnect can potentially reduce the overall costs associated with emergency response operations.

- Preventative Measures: Early intervention facilitated by CrisisConnect can help prevent emergencies from escalating, reducing the need for costly emergency interventions.

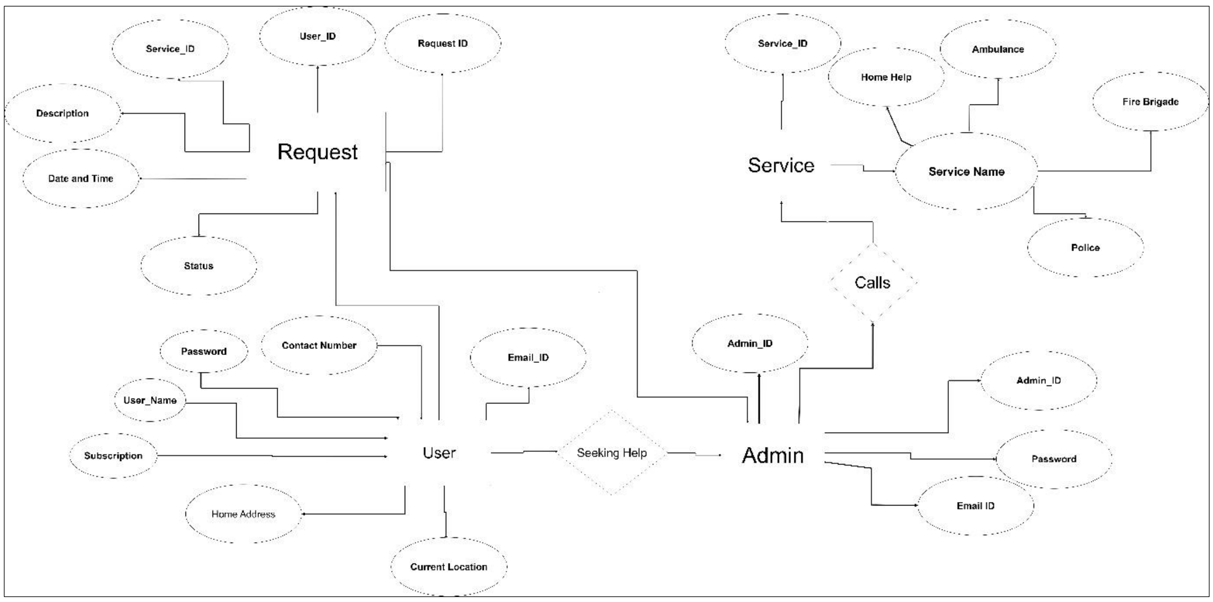
4. **User Satisfaction and Engagement:**

- Positive User Experience: Providing users with a reliable and easy-to-use platform for accessing emergency services can lead to high levels of satisfaction and engagement.

- Repeat Usage: Satisfied users are likely to return to CrisisConnect in future emergencies and may recommend the platform to others, contributing to its growth and sustainability.

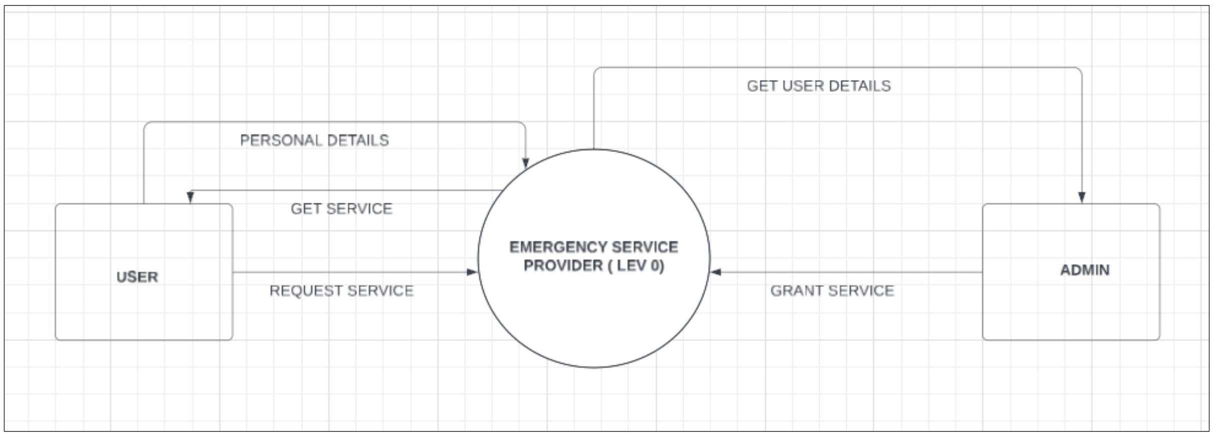
# Software Design Specification

## ERD :

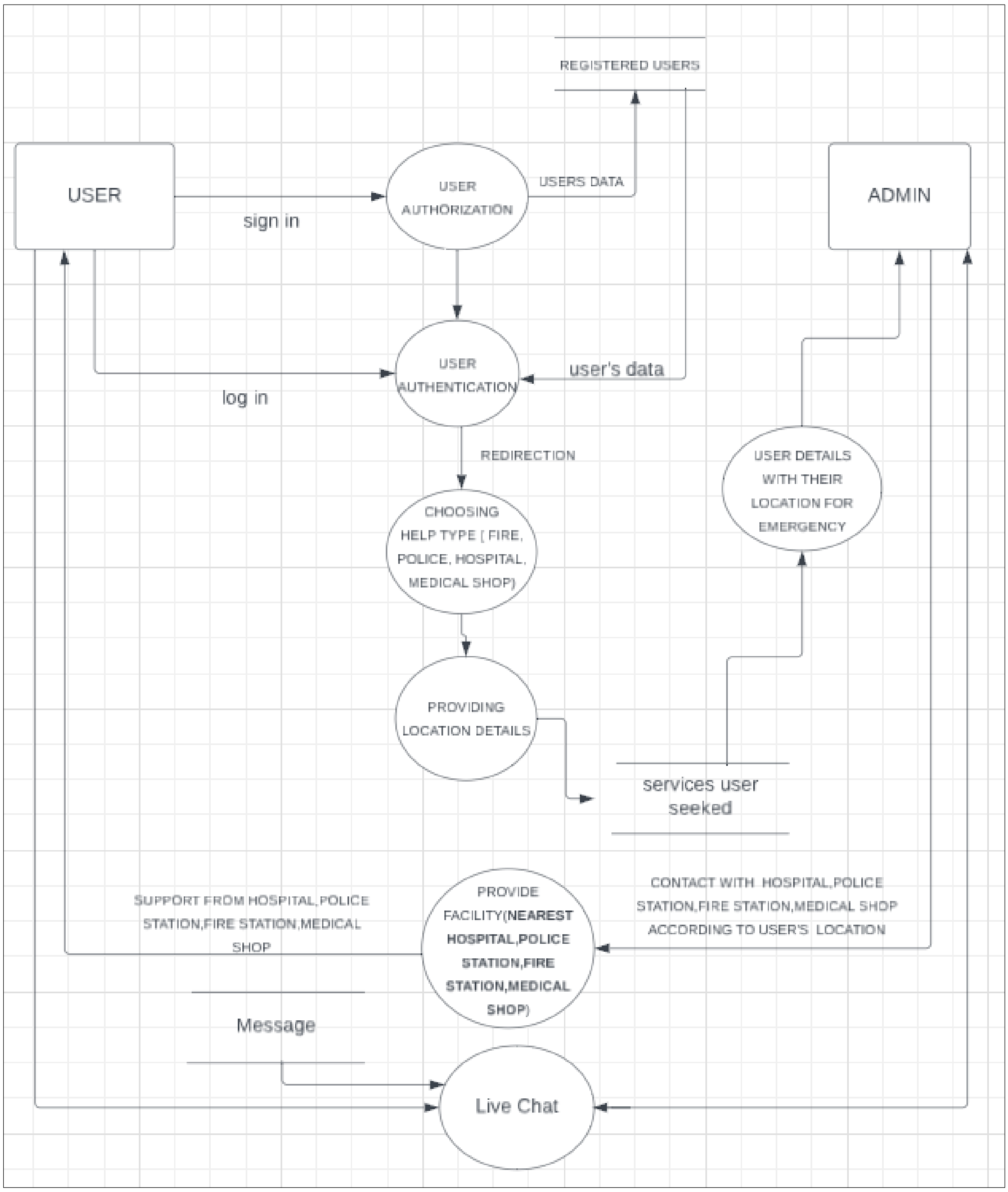


Entity-Relationship Diagram

## DFD :

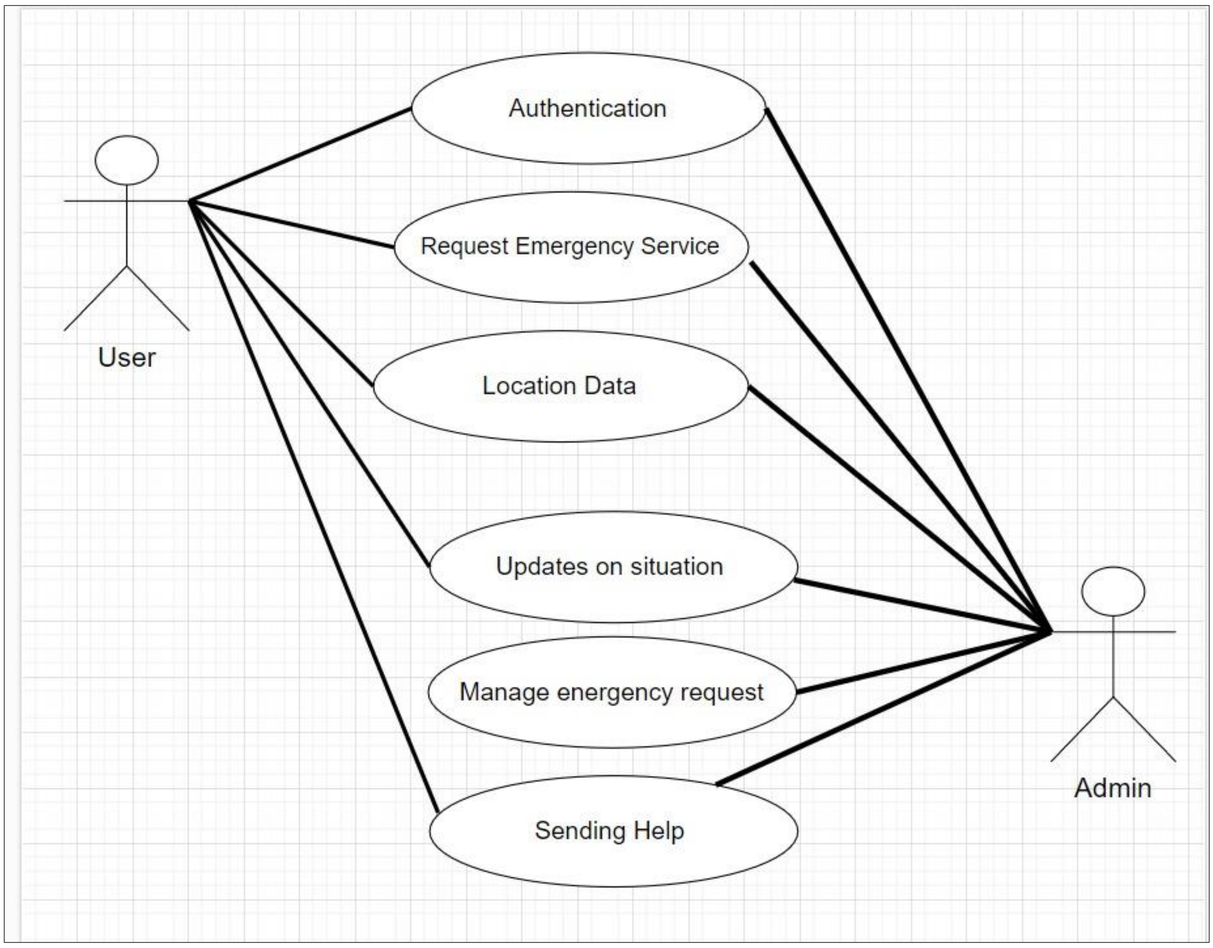


Level-0 DFD


REGISTERED USERS
USER
sign in
USER AUTHORIZATION
USERS DATA
ADMIN
USER AUTHENTICATION
user's data
log in
REDIRECTION
USER DETAILS WITH THEIR LOCATION FOR EMERGENCY
CHOOSING HELP TYPE ( FIRE, POLICE, HOSPITAL, MEDICAL SHOP)
PROVIDING LOCATION DETAILS
services user seeked
CONTACT WITH HOSPITAL,POLICE STATION,FIRE STATION,MEDICAL SHOP ACCORDING TO USER'S LOCATION
SUPPORT FROM HOSPITAL,POLICE STATION,FIRE STATION,MEDICAL SHOP
PROVIDE FACILITY(NEAREST HOSPITAL,POLICE STATION,FIRE STATION,MEDICAL SHOP)
Message
Live Chat

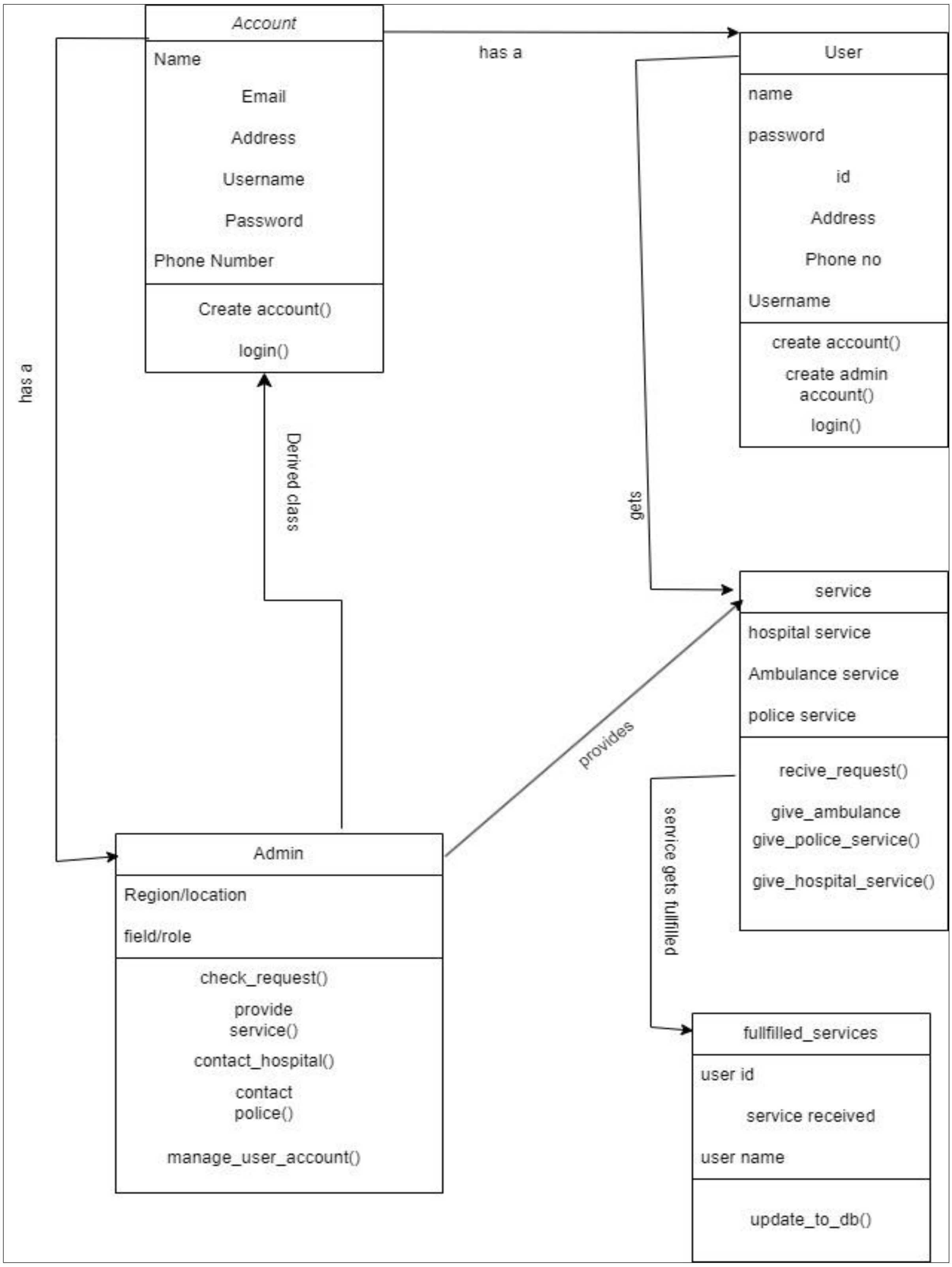
Level-1 DFD

## USE CASE DIAGRAM :



Use case diagram

## CLASS DIAGRAM :



Class Diagram

# Testing

## Testing Strategies:

1. **Unit Testing:**

- Focus on testing individual components or functions of the application to ensure they work as expected.

- Use automated testing frameworks such as Jest and Mocha for JavaScript to create and run unit tests.

- Test cases should cover a wide range of scenarios, including edge cases and error conditions.

2. **Integration Testing:**

- Test the interactions between different components and modules of the application.

- Ensure that integrated components work together as intended and data flows correctly across the system.

3. **System Testing:**

- Conduct end-to-end testing of the entire application to ensure it meets the specified requirements.

- Simulate real-world scenarios to validate the overall functionality, performance, and reliability of the system.

- Perform both manual and automated tests to cover different aspects of the system.

4. **Security Testing:**

- Perform security assessments to identify and mitigate potential vulnerabilities in the application.

- Ensure that the application adheres to security best practices and protects sensitive user data.

## Test Plan:

1. **Introduction:**

- Purpose: Outline the testing strategies and plan for ensuring the quality and reliability of CrisisConnect.

- Scope: Define the scope of testing, including components and features to be tested.

- Objectives: Ensure that the application meets functional and non-functional requirements, is free of defects, and provides a positive user experience.

2. **Test Items:**

- User Registration and Authentication
- User Dashboard
- Emergency Service Requests (Fire, Police, Hospital, Medical)
- Admin Panel
- Location-based Services

3. **Testing Approach:**

- Unit Testing: Develop and execute unit tests for individual components using Jest and Mocha.
- Integration Testing: Test interactions between modules using Postman and Selenium.
- System Testing: Perform end-to-end testing of the entire application.
- Security Testing: Perform security assessments.

## Test Results:

### Module 1 : Registration ( Sign Up )

| Test Case ID | Description | Expected Result | Actual Result | Status |
|---|---|---|---|---|
| **TC-SU-01** | Input Invalid Height ( Ex : 1000 Cm ) | Wrong Validation | Submission Failure | Passed |
| **TC-SU-02** | Input Invalid Weight ( Ex : 1000 Kg ) | Wrong Validation | Submission Failure | Passed |
| **TC-SU-03** | Input Invalid Phone Number ( Less Than 10 Digit) | Wrong Validation | Submission Failure | Passed |
| **TC-SU-04** | Input Invalid Phone Number ( More Than 10 Digit) | Wrong Validation | Submission Failure | Passed |

| | | | | |
|---|---|---|---|---|
| **TC-SU-05** | Input Wrong Email Id Format | Wrong Validation | Submission Failure | Passed |
| **TC-SU-06** | Input Password ( Less Than 8 Characters ) | Wrong Validation | Submission Failure | Passed |
| **TC-SU-07** | Trying To Create A New Account With Existing Email Id | Registration Blocked | Submission Failure | Passed |
| **TC-SU-08** | Fill With Invalid Adhar Card Number ( Less Than 12 Digits ) | Wrong Validation | Submission Failure | Passed |
| **TC-SU-09** | Fill With Invalid Adhar Card Number ( More Than 12 Digits ) | Wrong Validation | Submission Failure | Passed |
| **TC-SU-10** | Filled With Password And Confirm Password Are Different | Wrong Validation | Submission Failure | Passed |
| **TC-SU-11** | Fill Without Entering Name | Wrong Validation | Submission Failure | Passed |
| **TC-SU-12** | Fill Without Selecting Sex | Wrong Validation | Submission Failure | Passed |
| **TC-SU-13** | Fill Without Entering Height | Wrong Validation | Submission Failure | Passed |
| **TC-SU-14** | Fill Without Entering Weight | Wrong Validation | Submission Failure | Passed |
| **TC-SU-15** | Fill Without Selecting Blood Group | Wrong Validation | Submission Failure | Passed |
| **TC-SU-16** | Fill Without Entering Date Of Birth | Wrong Validation | Submission Failure | Passed |
| **TC-SU-17** | Fill Without Entering Email Id | Wrong Validation | Submission Failure | Passed |
| **TC-SU-18** | Fill Without Entering Phone Number | Wrong Validation | Submission Failure | Passed |
| **TC-SU-19** | Fill Without Entering Address | Wrong Validation | Submission Failure | Passed |
| **TC-SU-20** | Fill Without Entering Password | Wrong Validation | Submission Failure | Passed |
| **TC-SU-21** | Fill Without Entering Confirm Password | Wrong Validation | Submission Failure | Passed |

| | | | | |
|---|---|---|---|---|
| **TC-SU-22** | Fill With Correct Information | Success | Registration Successful | Passed |

**Module 2 : Log In**

| Test Case ID | Description | Expected Result | Actual Result | Status |
|---|---|---|---|---|
| **TC-LI-01** | Enter Unregisterd Email ID | Wrong Validation | Login Failure | Passed |
| **TC-LI-02** | Enter Registered Email ID With Wrong Password | Wrong Validation | Login Failure | Passed |
| **TC-LI-03** | Clicking Submit Without Entering Email ID | Wrong Validation | Login Failure | Passed |
| **TC-LI-04** | Clicking Submit Without Entering Password | Wrong Validation | Login Failure | Passed |
| **TC-LI-05** | Input Registered Email ID And Correct Password | Success | Login Successful | Passed |

**Module 3 : Password Reset**

| Test Case ID | Description | Expected Result | Actual Result | Status |
|---|---|---|---|---|
| **TC-PR-01** | Enter Unregistered Email ID To Verify | Wrong Validation | Verification Failure | Passed |
| **TC-PR-02** | Enter Registered Email ID To Verify | Success | Otp Sended Successfully | Passed |
| **TC-PR-02** | Enter Wrong OTP To Validate | Wrong Validation | Verification Failure | Passed |
| **PC-PR-03** | Enter Valid OTP To Validate | Success | Verification Successfully | Passed |
| **TC-PR-04** | Setting New Password | Success | Password Set Successfully | Passed |

| | | | | |
|---|---|---|---|---|
| **TC-PR-05** | Trying To Login With Old Password | Wrong Validation | Login Failure | Passed |
| **TC-PR-06** | Login With New Password | Success | Login Successful | Passed |

**Module 4 : Requesting Emergency Services**

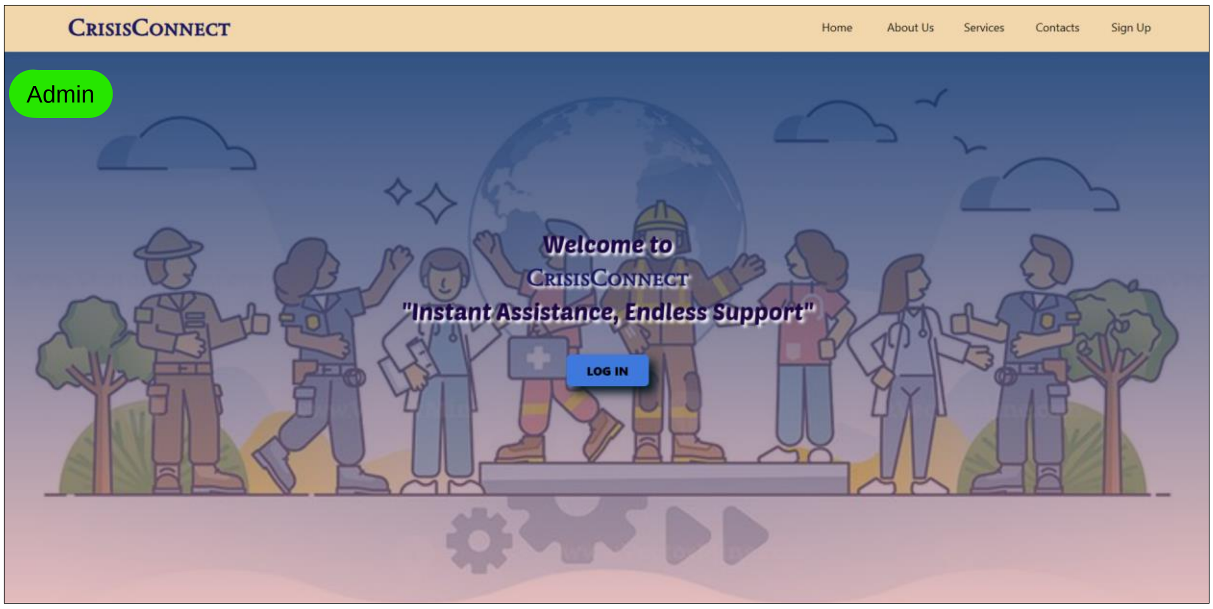
| Test Case ID | Description | Expected Result | Actual Result | Status |
|---|---|---|---|---|
| **TC-RS-01** | Requesting Fire Help For Current Location | Successful Request | Request Sent Successfully | Passed |
| **TC-RS-02** | Requesting Fire Help For Home Location | Successful Request | Request Sent Successfully | Passed |
| **TC-RS-03** | Requesting Police Help For Current Location | Successful Request | Request Sent Successfully | Passed |
| **TC-RS-04** | Requesting Police Help For Home Location | Successful Request | Request Sent Successfully | Passed |
| **TC-RS-05** | Requesting Hospital Help For Current Location | Successful Request | Request Sent Successfully | Passed |
| **TC-RS-06** | Requesting Hospital Help For Home Location | Successful Request | Request Sent Successfully | Passed |
| **TC-RS-07** | Requesting Medical Help For Current Location | Successful Request | Request Sent Successfully | Passed |
| **TC-RS-08** | Requesting Medical Help For Home Location | Successful Request | Request Sent Successfully | Passed |

**Security Testing :**

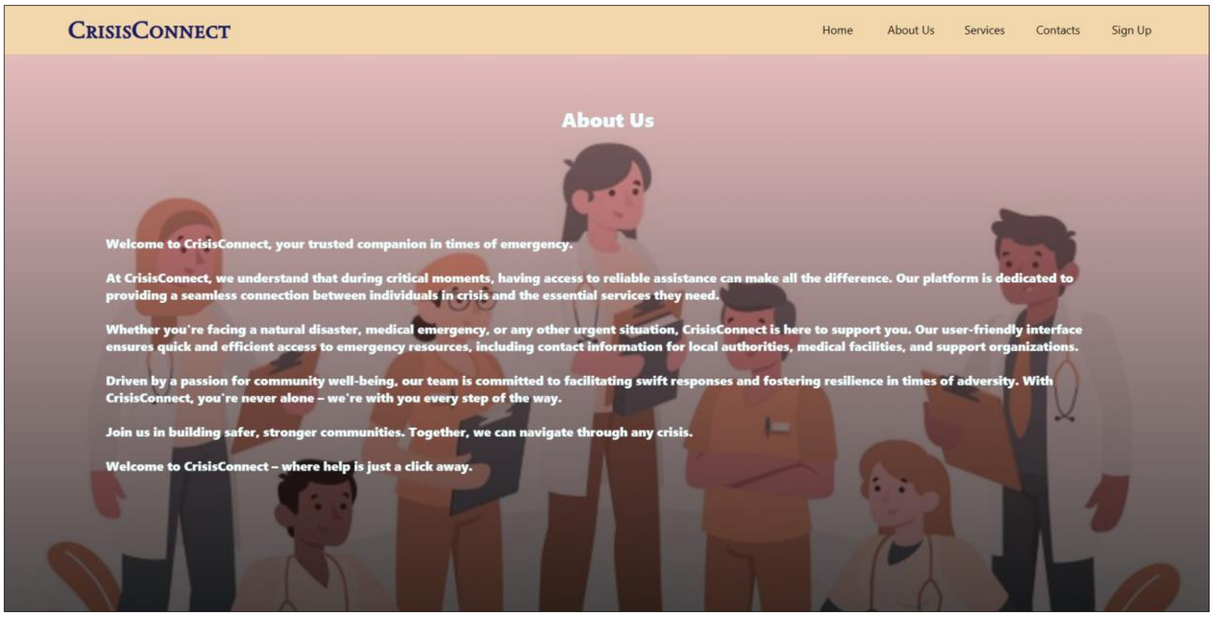
| Test Case ID | Description | Expected Result | Actual Result | Status |
|---|---|---|---|---|
| **TC-ST-01** | Trying To Access User Dashboard Without Login By Changing URL | Access Blocked | Not Accessible | Passed |

|  |  |  |  |  |
|---|---|---|---|---|
| **TC-ST-02** | Trying To Access User Sub-Dashboard Without Login By Changing URL | Access Blocked | Not Accessible | Passed |
| **TC-ST-03** | Trying To Access Admin Dashboard Without Login By Changing URL | Access Blocked | Not Accessible | Passed |
| **TC-ST-04** | Trying To Access Sub-Dashboard Without Login By Changing URL | Access Blocked | Not Accessible | Passed |

# User Manual of CrisisConnect



The *Landing Page* of **CrisisConnect** , which includes *About Us* , *Our Services* and our *Contact* details. New users can register themselves to **CrisisConnect** by clicking *Signup* on the top right corner. And existing **CrisisConnect** users should login themselves ( to get emergency services ) by clicking *Login* option.



Welcome to CrisisConnect, your trusted companion in times of emergency.

At CrisisConnect, we understand that during critical moments, having access to reliable assistance can make all the difference. Our platform is dedicated to providing a seamless connection between individuals in crisis and the essential services they need.

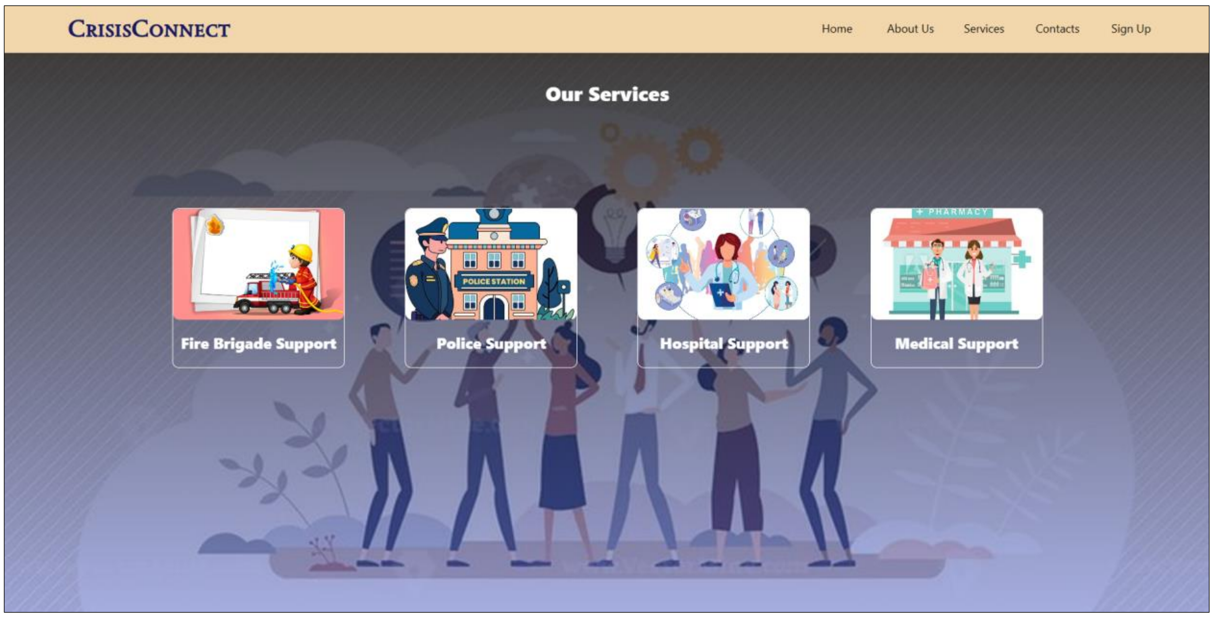
Whether you're facing a natural disaster, medical emergency, or any other urgent situation, CrisisConnect is here to support you. Our user-friendly interface ensures quick and efficient access to emergency resources, including contact information for local authorities, medical facilities, and support organizations.

Driven by a passion for community well-being, our team is committed to facilitating swift responses and fostering resilience in times of adversity. With CrisisConnect, you're never alone – we're with you every step of the way.

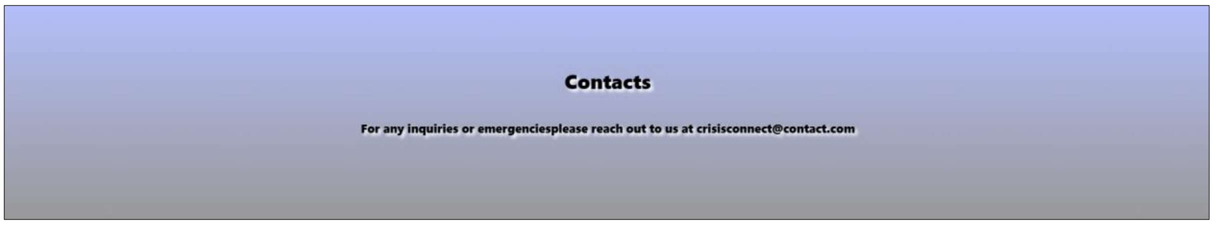
Join us in building safer, stronger communities. Together, we can navigate through any crisis.

Welcome to CrisisConnect – where help is just a click away.

About **CrisisConnect**



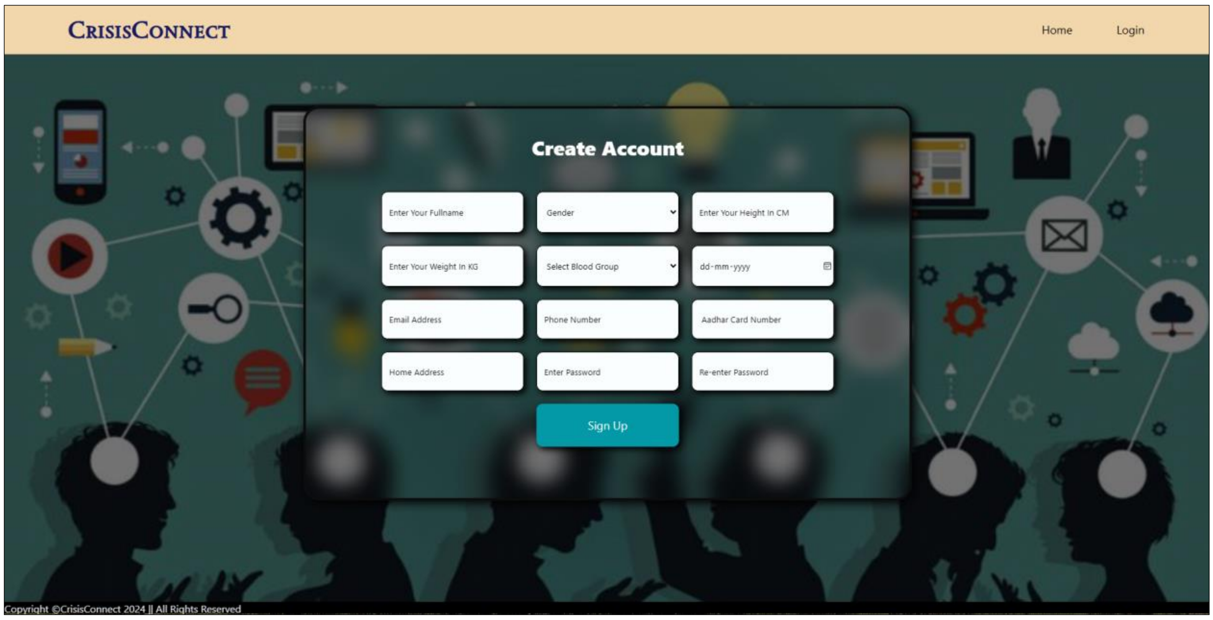
Services provided by **CrisisConnect**



**Contacts**

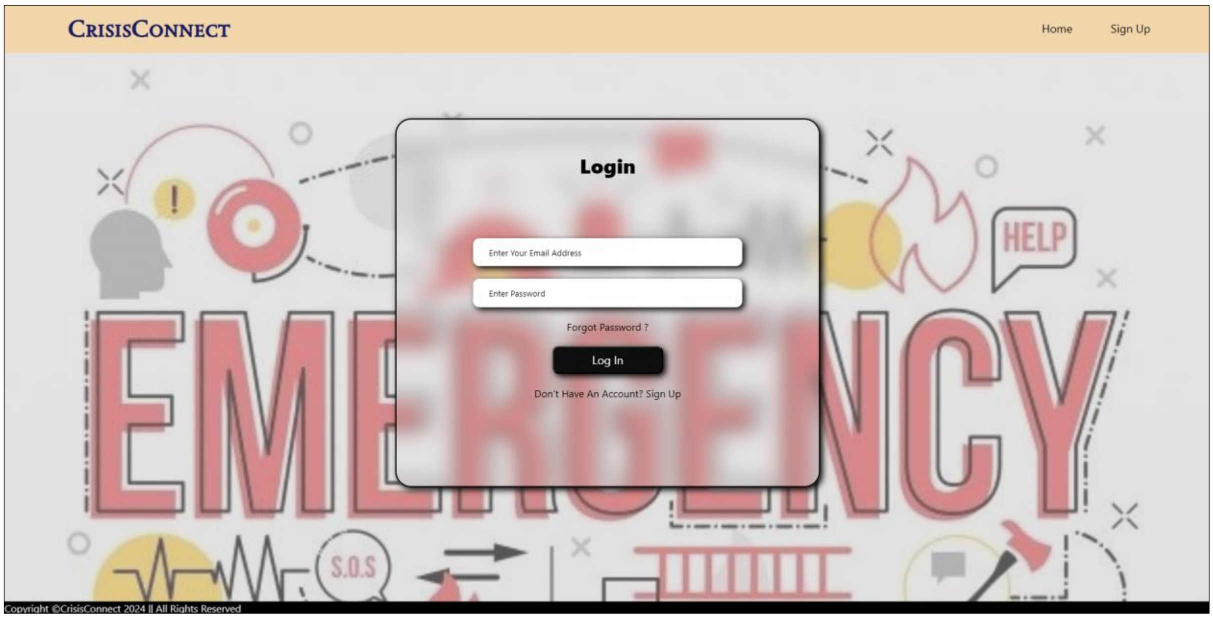
For any inquiries or emergenciesplease reach out to us at crisisconnect@contact.com

Contact details of **CrisisConnect**

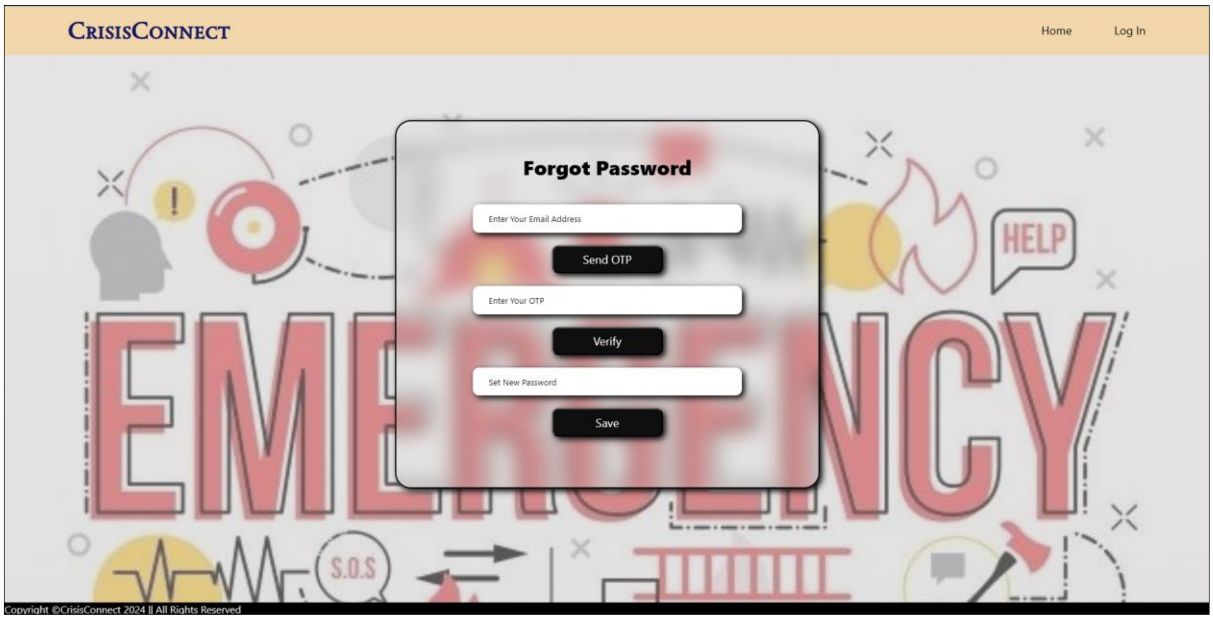


Copyright ©CrisisConnect 2024 || All Rights Reserved

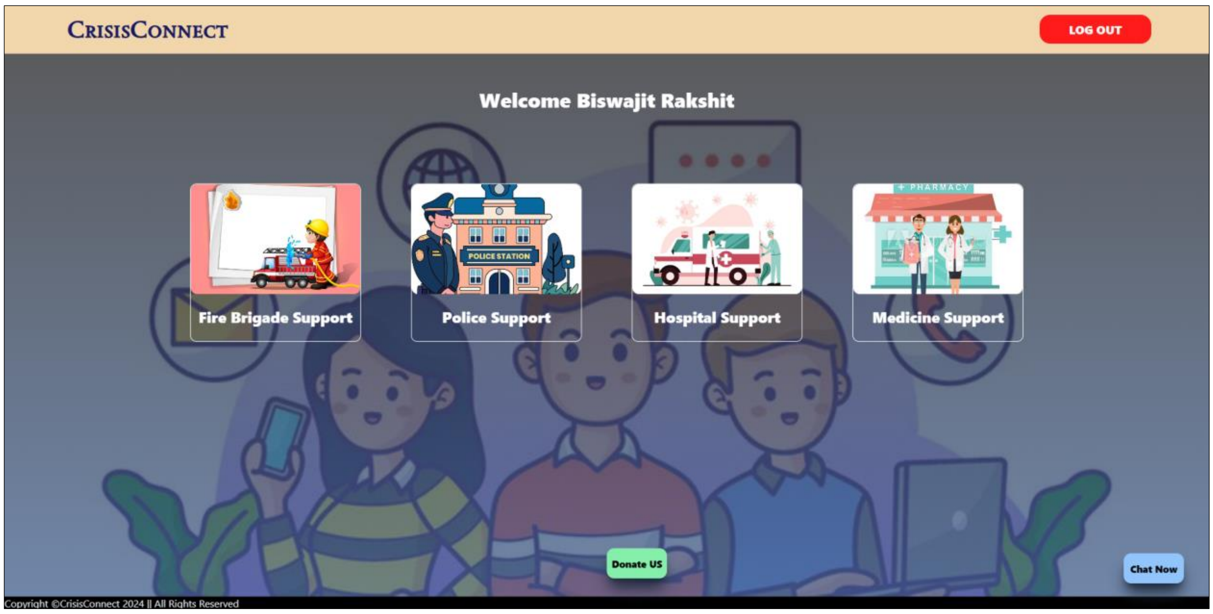
*Signup Page* Of **CrisisConnect ,** new users need to register themselves with valid information to get emergency support from us.



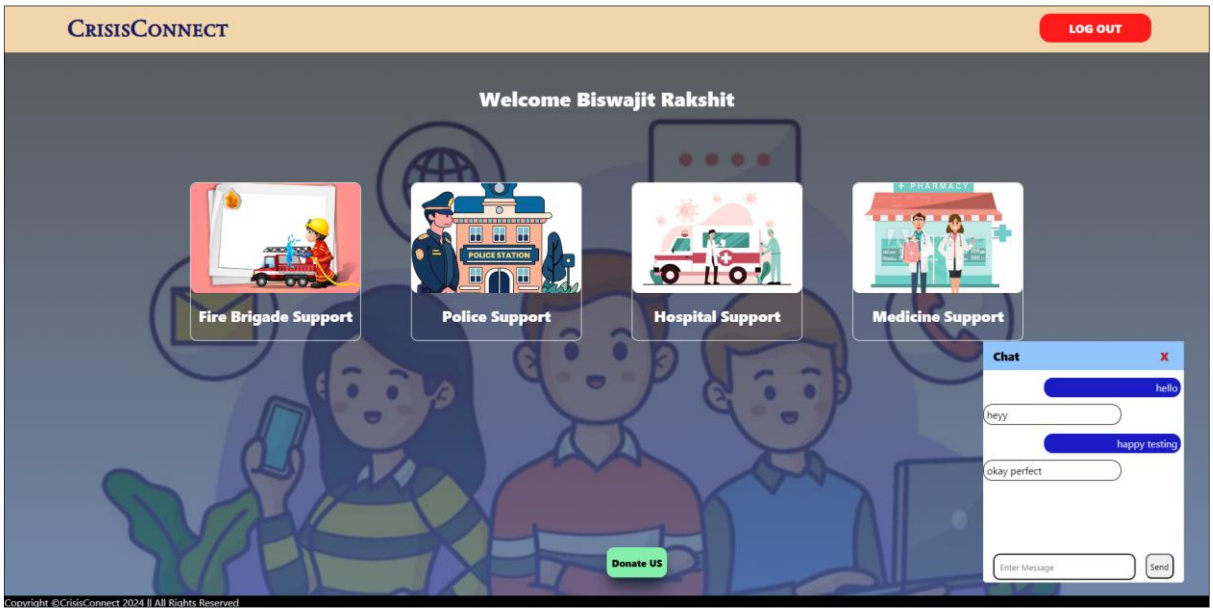
*Login Page* of **CrisisConnect ,** existing users need to login themselves with their registered email id and password to get emergency support from us. If any user forgot their registered password they can set new password by simply clicking on *Forgot Password* .



*Forgot Password* page of **CrisisConnect** , if a user need to set new password for their profile they can simply do it from this page. Just need to enter the email id and validate the OTP that send to their email. After validating the OTP they can set a new password for their profile.

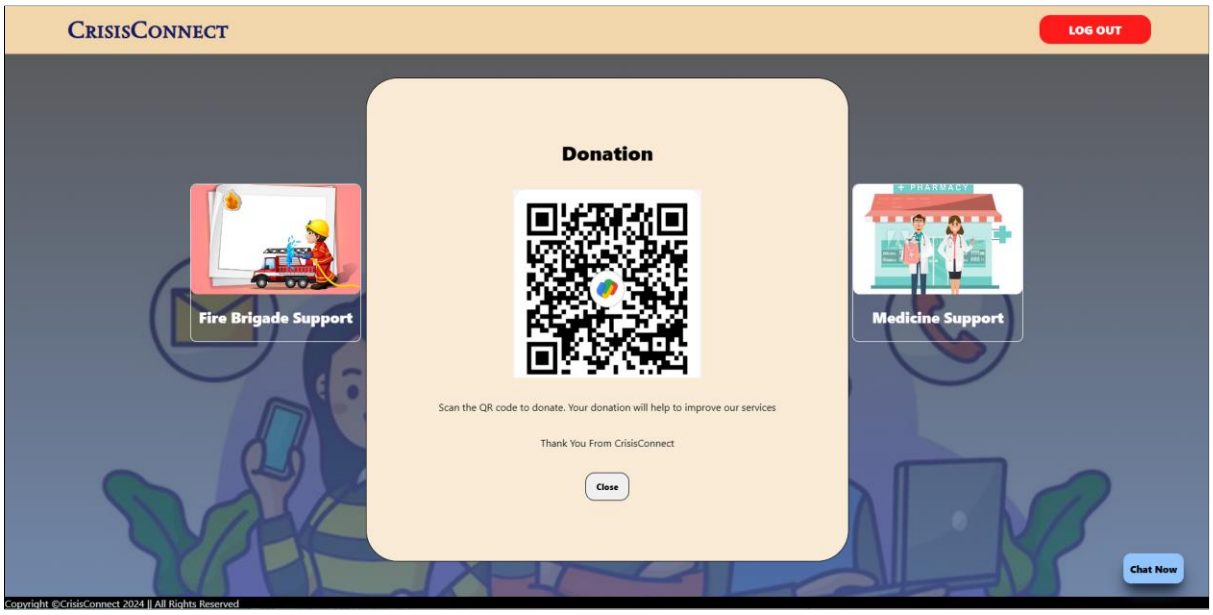


After successful login user will navigate to the *User Dashboard* of **CrisisConnect**. Here user can choose what kind of help/support they need among those services. Also they can donate to the CrisisConnect fund by clicking on *Donate Us* button. Also they can directly contact with the admins via live-chat support by clicking on the *Chat Now* option.

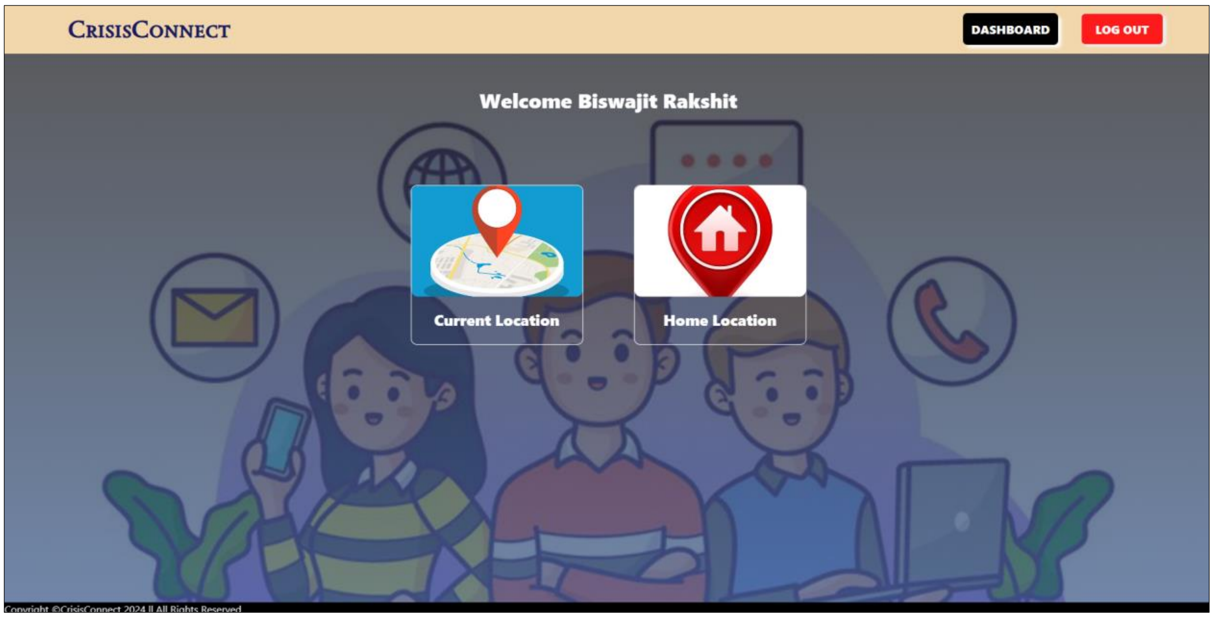


hello
heyy
happy testing
okay perfect

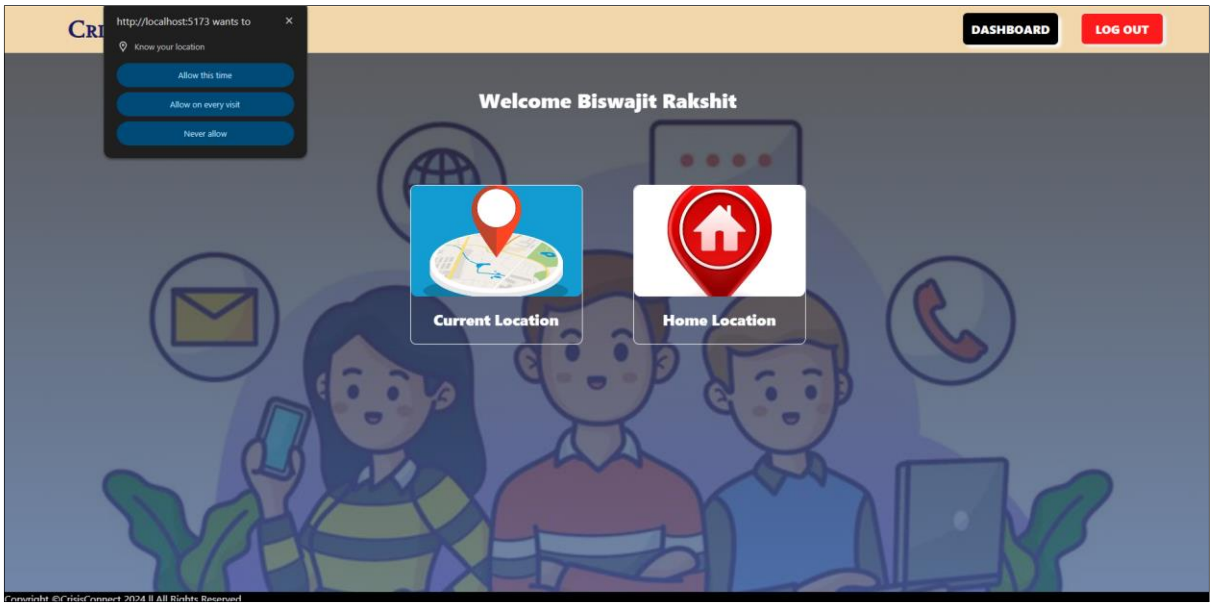
*Live Chat Support* system of **CrisisConnect**. User can directly communicate with admins via this features.


CrisisConnect
LOG OUT
Donation
Fire Brigade Support
Medicine Support
Scan the QR code to donate. Your donation will help to improve our services
Thank You From CrisisConnect
Close
Chat Now
Copyright ©CrisisConnect 2024 || All Rights Reserved

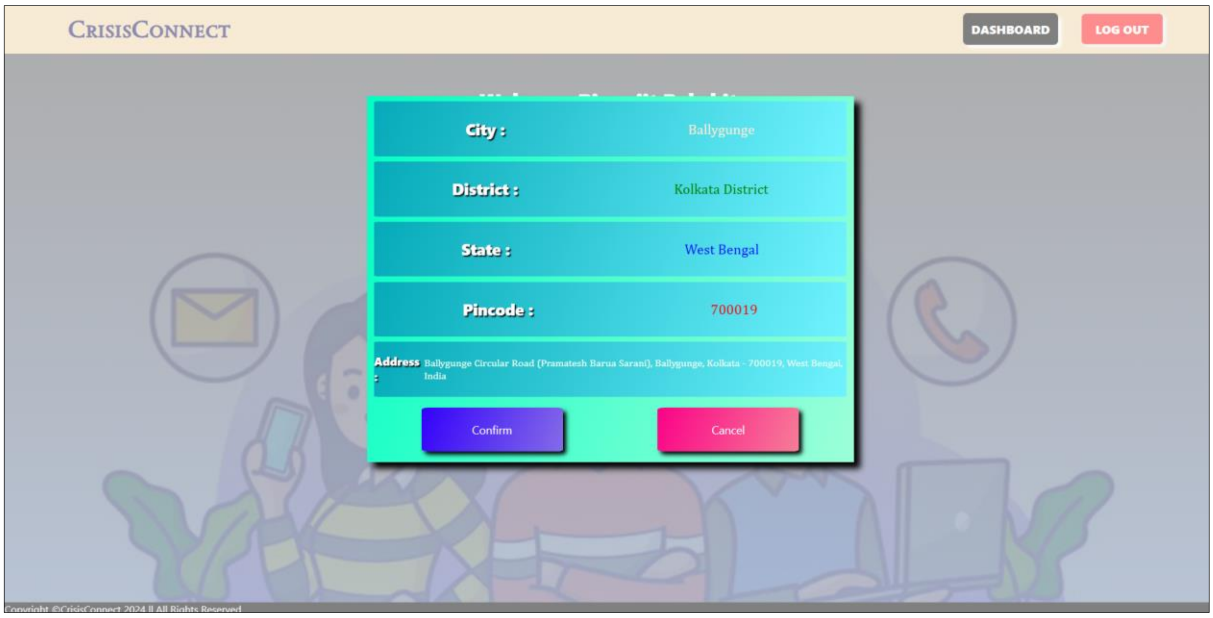
User's can donate to the **CrisisConnect** fund by scanning this UPI QR.



After selecting the help/support type user need to select their location preference. Either they need help in their current location or sending help to their home address.

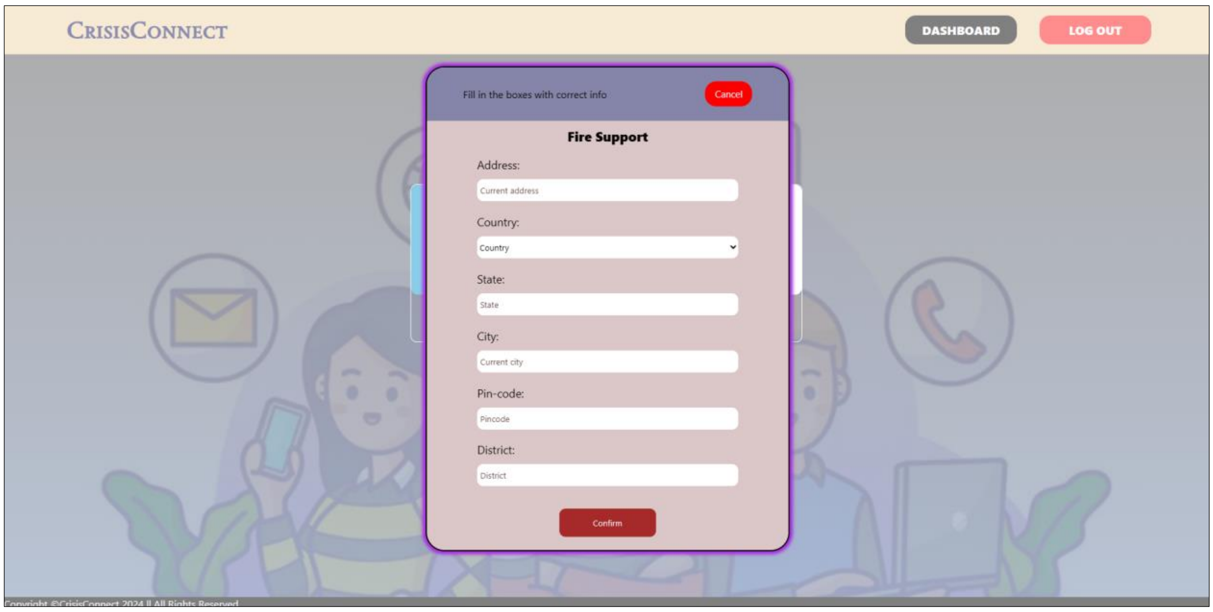


If  a user need help in their current location , they can available it by clicking on the current location option. After clicking that option browser will ask for location permission of their respective device. User must need to allow that permission to available services.



| | |
|---|---|
| City : | Ballygunge |
| District : | Kolkata District |
| State : | West Bengal |
| Pincode : | 700019 |
| Address : | Ballygunge Circular Road (Pramatesh Barua Sarani), Ballygunge, Kolkata - 700019, West Bengal, India |

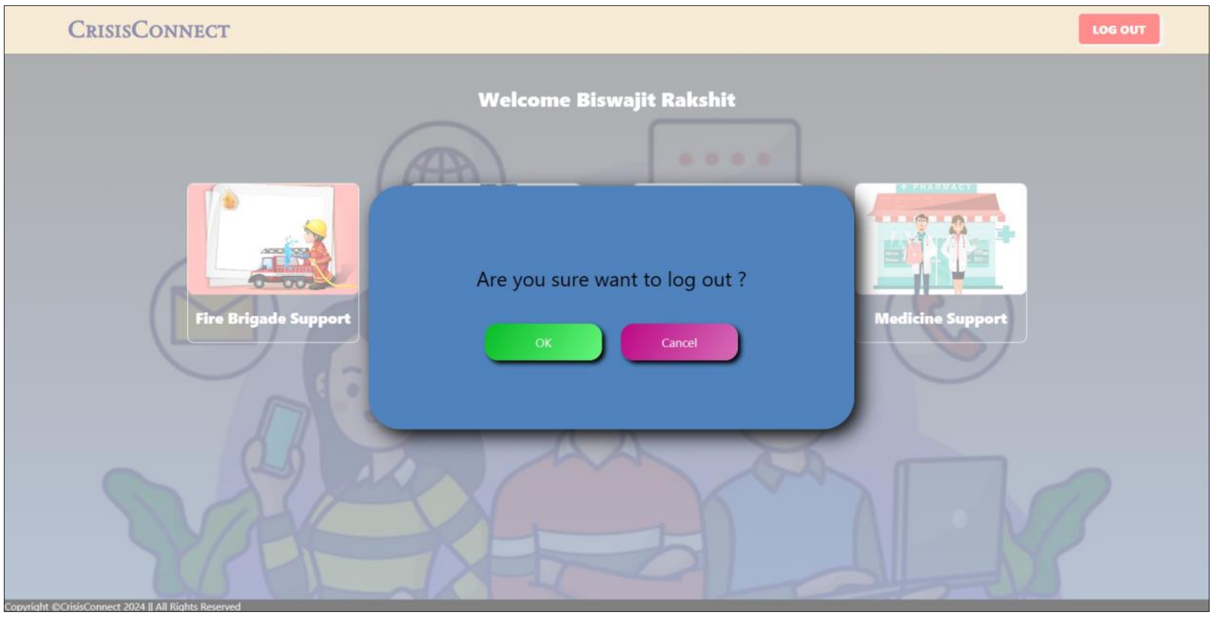
If the user allow the location permission this *Confirmation Page* will pop-up. It shows the user current location details and ask user to Confirm or Cancel the service request. To available services user need to click on confirm option.


CrisisConnect
DASHBOARD
LOG OUT
Fill in the boxes with correct info
Cancel
Confirm
Copyright ©CrisisConnect 2024 || All Rights Reserved

**Fire Support**

| | |
|---|---|
| Address: | Current address |
| Country: | Country |
| State: | State |
| City: | Current city |
| Pin-code: | Pincode |
| District: | District |

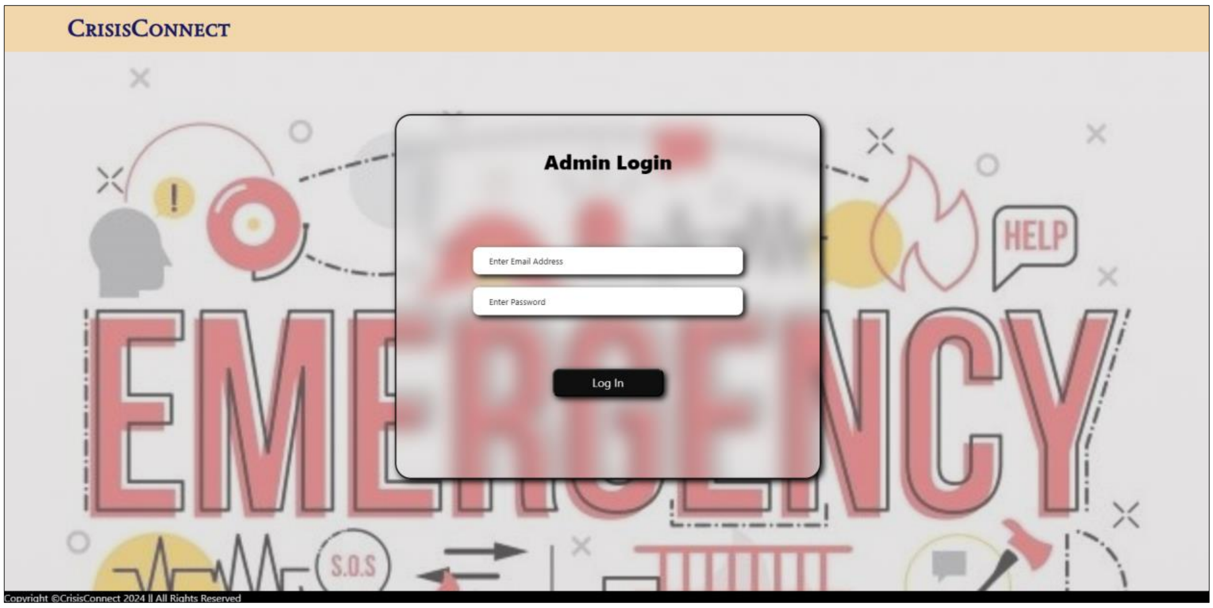
If  a user need help in their home location , they available it by clicking on the home location option. After clicking that option a form will open. User must need to fill that form with correct location information to available services.



Are you sure want to log out ?

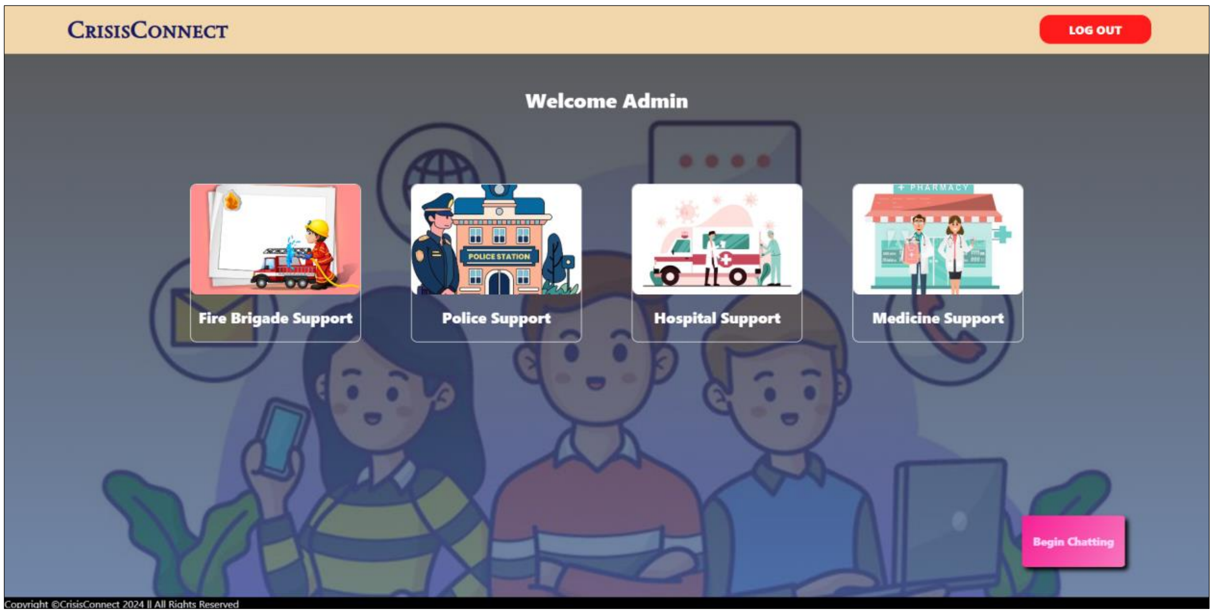
To logout from **CrisisConnect** user need to click on *Log Out* option on the top right side of their dashboard.

**Admin Panel :**

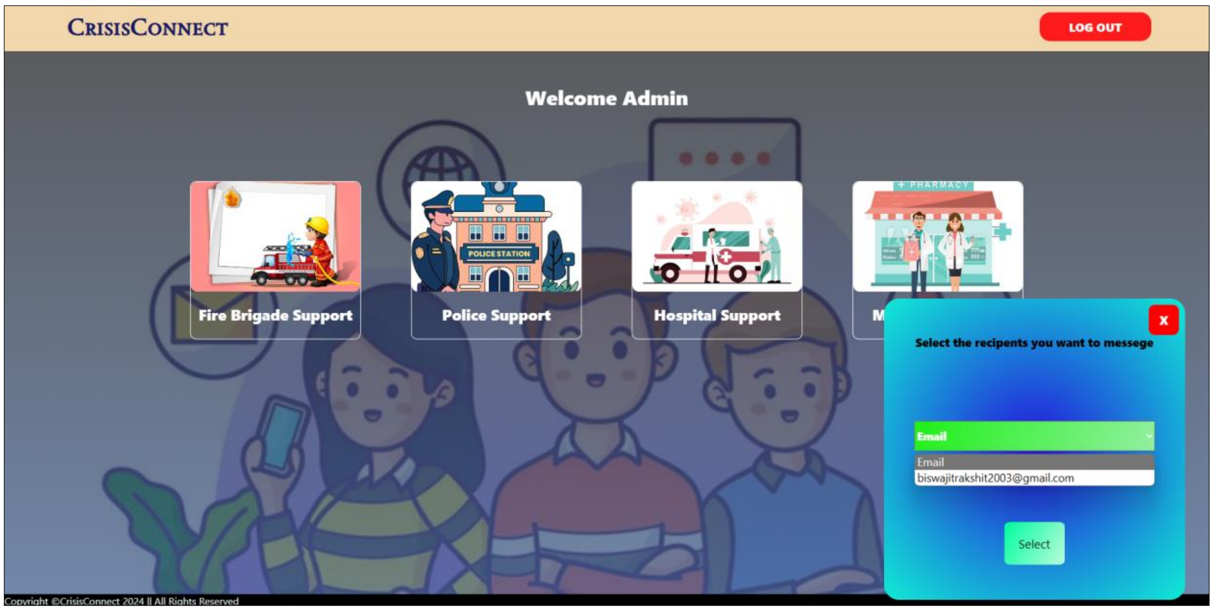


The *Log In* page of admin panel. Admins need to login with the registered admin email id and password to enter the admin panel.

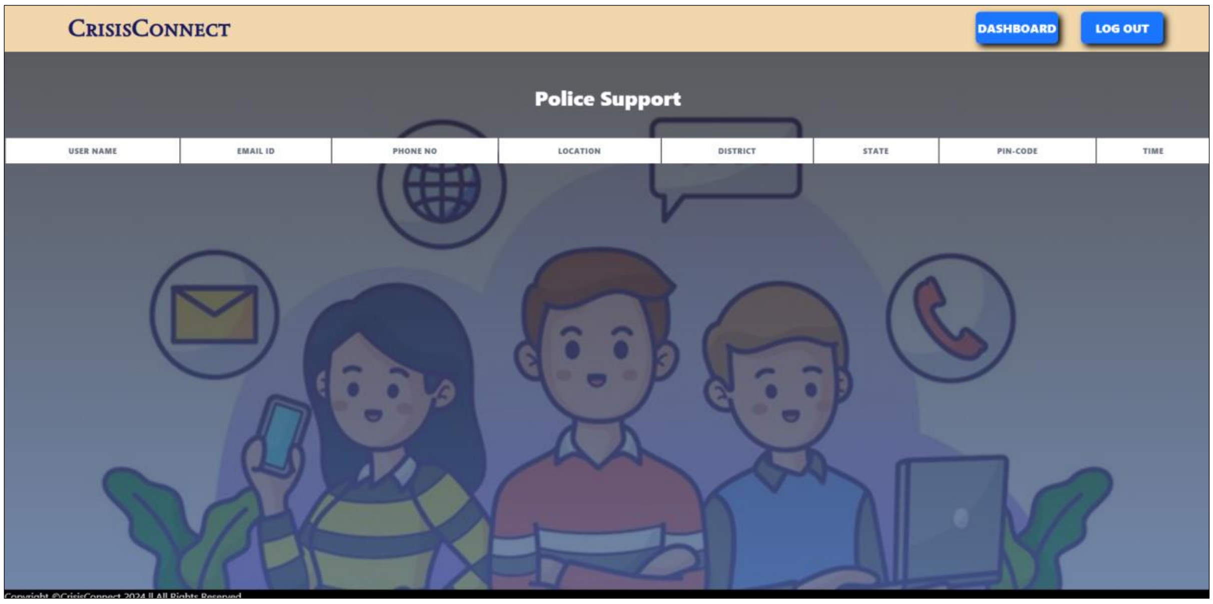
*( The admin login page is not accessible from the* ***CrisisConnect*** *landing page due to privacy issues. It can be accessible by changing the url as "<websiteurl>/admin" )*



After successful login admins will navigate to the *Admin Dashboard* of **CrisisConnect**. Here admin can view all types of requested services by clicking on these tabs. And to check incoming chats from users admin should click on the *Chat* option at the bottom corner.

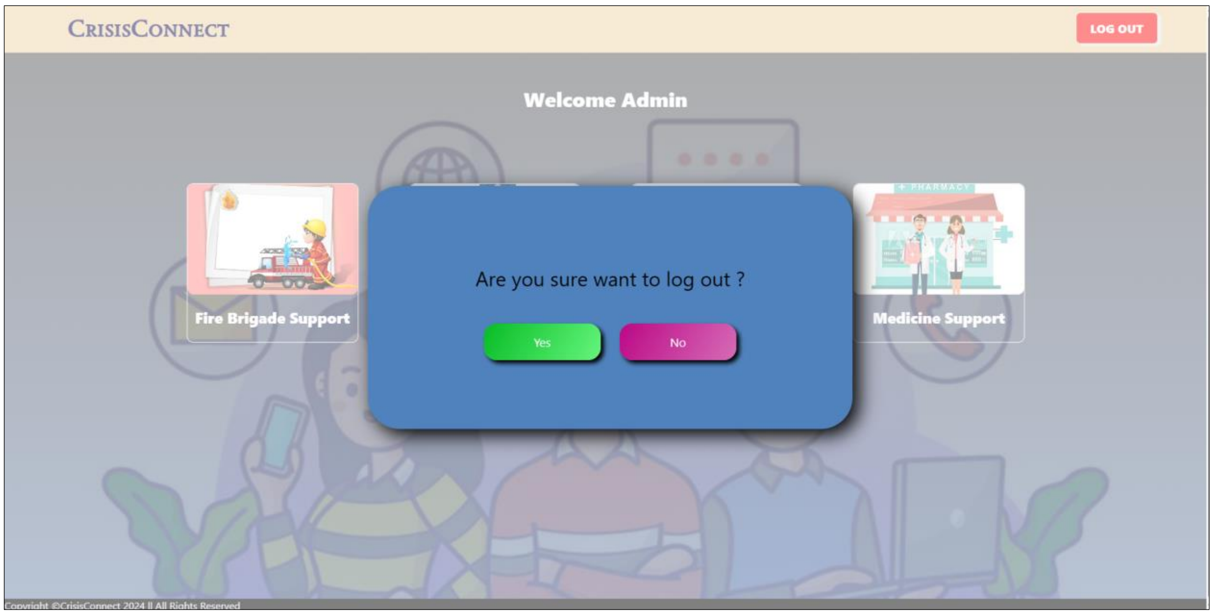

CrisisConnect
LOG OUT
Welcome Admin
Fire Brigade Support
Police Support
Hospital Support
x
Select the recipents you want to messege
Email
Email
biswajitrakshit2003@gmail.com
Select
Copyright ©CrisisConnect 2024 || All Rights Reserved

To begin view incoming messages admin need to select the email id. And then can reply their messages via *Realtime-Chat* features.



| USER NAME | EMAIL ID | PHONE NO | LOCATION | DISTRICT | STATE | PIN-CODE | TIME |
|---|---|---|---|---|---|---|---|

Emergency requests are displayed on this page. Admins can contacts with the nearest fire brigade, police station, hospital, medical store to provides service to users. *( The table on this page is showing blank because it contains users personal informations )*


CrisisConnect
LOG OUT
Welcome Admin
Fire Brigade Support
Medicine Support
Are you sure want to log out ?
Yes
No
Copyright ©CrisisConnect 2024 || All Rights Reserved

To logout from **CrisisConnect** Admins need to click on *Log Out* option on the top right side of their dashboard.

# Limitations

1. **Dependency on Internet Connectivity:**

- CrisisConnect relies heavily on internet connectivity for users to access the platform and request emergency services. In areas with poor or no internet access, users may be unable to use the application effectively.

2. **Accuracy of Location Services:**

- The effectiveness of CrisisConnect in providing timely assistance is dependent on the accuracy of the location data provided by users or determined by GPS. Inaccuracies in location data can lead to delays or misrouting of emergency services.

3. **Scalability Challenges:**

- While the MERN stack is scalable, handling a sudden surge in users during a large-scale emergency or disaster could strain system resources and impact performance. Ensuring the system remains responsive under high load conditions requires significant investment in infrastructure and optimization.

4. **User Adoption and Training:**

- Widespread user adoption is essential for the success of CrisisConnect. However, some users may be resistant to adopting new technologies, and others may require training to use the application effectively, especially during stressful emergency situations.

5. **Language and Accessibility Barriers:**

- To be universally accessible, CrisisConnect must support multiple languages and accommodate users with disabilities. Developing and maintaining these features can be resource-intensive and may still not fully address the needs of all users.

6. **Resource Allocation:**

- The process of coordinating with and dispatching the nearest available emergency services involves multiple variables and uncertainties. Ensuring optimal resource allocation without overburdening specific service providers can be challenging.

By acknowledging these limitations, the CrisisConnect project team can proactively address potential issues, mitigate risks, and enhance the platform's effectiveness and reliability.

# Future Planning

Planning for the future of **CrisisConnect** involves identifying opportunities for expansion, improvement, and sustainability to enhance its effectiveness in addressing emergency situations and serving the needs of users and stakeholders. Here are some key considerations for future planning:

1. **Mobile Application Development:**

- Develop a mobile application version of CrisisConnect to enhance accessibility and reach, particularly for users in areas with limited internet connectivity or those who prefer mobile devices over desktop computers.

- Optimize the mobile app for offline functionality, allowing users to submit emergency requests and access critical information even in the absence of internet connectivity.

2. **Geographical Expansion:**

- Expand the geographical coverage of CrisisConnect to reach underserved or remote areas that may lack access to traditional emergency services.

- Collaborate with local authorities, non-profit organizations, and community leaders to customize the platform to meet the unique needs and challenges of different regions.

3. **Partnerships and Collaborations:**

- Forge partnerships with emergency service providers, government agencies, healthcare organizations, and technology companies to enhance the capabilities and reach of CrisisConnect.

- Collaborate with academic institutions and research organizations to leverage data analytics, machine learning, and predictive modeling for improving emergency response strategies.

4. **Community Engagement and Awareness:**

- Launch community outreach campaigns to raise awareness about CrisisConnect and educate the public on the importance of emergency preparedness and response.

- Engage with local communities through workshops, seminars, and social media campaigns to promote adoption, participation, and feedback gathering.

5. **Continuous Improvement and Iteration:**

- Establish a culture of continuous improvement and iteration within the CrisisConnect development team, encouraging regular feedback collection, experimentation, and adaptation.

- Implement agile development methodologies and DevOps practices to streamline development workflows, accelerate feature delivery, and maintain high standards of quality and reliability.

# Bibliography & References

1. Johnson, M. (2022). "Emergency Response Systems: Challenges and Opportunities." Journal of Emergency Management, 10(2), 45-62.

2. Smith, A. (2023). "User-Centered Design Principles for Emergency Service Platforms." Proceedings of the International Conference on Human-Computer Interaction.

3. React.js Documentation. (n.d.). Retrieved from https://reactjs.org/docs/getting-started.html

4. Node.js Documentation. (n.d.). Retrieved from https://nodejs.org/en/docs/

5. MongoDB Documentation. (n.d.). Retrieved from https://docs.mongodb.com/

6. "Best Practices for Designing RESTful APIs." (2022). Whitepaper, CrisisConnect Development Team.

7. "Emergency Management and Preparedness Guidelines." (2023). Report, Federal Emergency Management Agency (FEMA).

8. "Data Privacy and Security Regulations for Emergency Service Platforms." (2023). Legal Brief, CrisisConnect Legal Team.

9. Smith, J. (2022). "Community Engagement Strategies for Crisis Response Platforms." Proceedings of the International Conference on Community Development.

10. "Best Practices for Agile Development in Emergency Response Projects." (2023). Internal Document, CrisisConnect Development Team.