SAHAAYIKHA

PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degree of Master of Computer Applications of A P J Abdul Kalam Technological University

Submitted by:

ATHULKRISHNA P P(SJC24MCA-2019)

MASTER OF COMPUTER APPLICATIONS

ST. JOSEPH'S COLLEGE OF ENGINEERING AND TECHNOLOGY, PALAI
(AUTONOMOUS)

CHOONDACHERRY P.O, KOTTAYAM
KERALA
OCTOBER 2025

ST. JOSEPH' S COLLEGE OF ENGINEERING AND TECHNOLOGY, PALAI
(AUTONOMOUS)

(An ISO 9001: 2015 Certified College)

CHOONDACHERRY P.O, KOTTAYAM KERALA

CERTIFICATE

This is to certify that the project work entitled **'SAHAAYIKHA'** submitted by **Athulkrishna P P,** student of **Third** semester **MCA** at **ST. JOSEPH'S COLLEGE OF ENGINEERING AND TECHNOLOGY**, **PALAI** in partial fulfillment for the award of Master of Computer Applications is a bonafide record of the project work carried out by him under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Dr. Fr. Jeethu Mathew	**Prof. Liz George**	**Dr. Rahul Shajan**
Assistant Professor	Assistant Professor	Associate Professor
(Project Guide)	(Project Coordinator)	(HoD-Dept. of CA)

Submitted for the Viva-Voce Examination held on ___________________________

Examiner 1: **Examiner 2:**

DECLARATION

I **Athulkrishna P P**, do hereby declare that the project titled **"SAHAAYIKHA"** is a record of work carried out under the guidance **Dr. Fr. Jeethu Mathew**, Asst. Professor, Department of Computer Applications, SJCET, Palai as per the requirement of the curriculum of Master of Computer Applications Programme of A P J Abdul Kalam Technological University, Thiruvananthapuram. Further, I also declare that this report has not been submitted, full or part thereof, in any University / Institution for the award of any Degree / Diploma.

Place: Choondacherry — Athulkrishna P P

Date: — (SJC24MCA-2019)

ACKNOWLEDGEMENT

The success of any project depends largely on the encouragement and guidelines of many others. I would like to take this opportunity to express my gratitude to those people who have been instrumental in the successful completion of this project.

First and foremost, I give all glory, honor and praise to **God Almighty** who gave me wisdom and enabled me to complete the project successfully.

I also express sincere thanks, from the bottom of my heart, to my parents for their encouragement and support in all my endeavors and especially in this project.

Words are inadequate to express my deep sense of gratitude to **Dr. V. P Devasia, Principal, SJCET, Palai** for allowing me to utilize all the facilities of our college and for his encouragement.

I extend my sincere gratitude to **Dr. Rahul Shajan, HOD, Department of Computer Applications**, **SJCET, Palai** who has been a constant source of inspiration and without his tremendous help and support this project would not have been materialized.

I owe a particular debt of gratitude to my project guide, **Dr. Fr. Jeethu Mathew, Asst.Professor, Department of Computer Applications, SJCET, Palai** for all the necessary help and support that he has extended to me. His valuable suggestions, corrections, and the sincere efforts to accomplish my project even under a tight time schedule were crucial in the successful completion of this project.

I extend my sincere thanks to all our teachers and non-teaching staff of SJCET, Palai for the knowledge they have imparted to me over the last three years.

I would also like to express my appreciation to all my friends for their comments, help and support.

DEPARTMENT OF COMPUTER APPLICATIONS

VISION

To emerge as a centre of excellence in the field of computer education with distinct identity and quality in all areas of its activities and develop a new generation of computer professionals with proper leadership, commitment, and moral values.

MISSION

- ❖ Provide quality education in Computer Applications and bridge the gap between the academia and industry.
- ❖ Promoting innovation research and leadership in areas relevant to the socio-economic progress of the country.
- ❖ Develop intellectual curiosity and a commitment to lifelong learning in students, with societal and environmental concerns.

PEO'S (Program Educational Objectives)

1. MCA Graduates will be able to progress career productively in software industry, academia, research, entrepreneurship pursuits, government, consulting firms and other IT enabled services.
2. MCA Graduates will be able to achieve peer-recognition as an individual or in a team by adopting ethics and professionalism, and communicate effectively to excel well in crisis and inter-disciplinary teams.
3. MCA Graduates will be able to continue life-long professional development in computing and in management that contributes in self and societal growth.

PROGRAMME OUTCOME

PO1(Foundation Knowledge): Apply knowledge of mathematics, programming logic, and coding fundamentals for solution architecture and problem-solving.

PO2 (Problem Analysis): Identify, review, formulate, and analyze problems primarily focusing on customer requirements using critical thinking frameworks.

PO3 (Development of Solutions): Design, develop, and investigate problems with an innovative approach for solutions incorporating ESG/SDG goals.

PO4 (Modern Tool Usage): Select, adapt, and apply modern computational tools, such as the development of algorithms, with an understanding of the limitations, including human biases.

PO5 (Individual and Teamwork): Function and communicate effectively as an individual or team leader in diverse and multidisciplinary groups, using methodologies such as agile.

PO6 (Project Management and Finance): Apply the principles of project management, including scheduling and work breakdown structure, and be knowledgeable about finance principles for profitable project management.

PO7 (Ethics): Commit to professional ethics in managing software projects, especially in financial aspects. Learn to use new technologies for cybersecurity and insulate customers from malware.

PO8 (Life-long Learning): Continuously enhance management skills and the ability to learn, keeping up with contemporary technologies and ways of working.

24SJMCA245 MINI PROJECT

Co No	COURSE OUTCOMES	K Level
CO1	Identify a real-life project which is useful to society / industry	K2
CO2	Interact with people to identify the project requirements	K2
CO3	Apply suitable development methodology for the development of the product / project	K3
CO4	Analyse and design a software product / project	K3
CO5	Test the modules at various stages of project development	K3
CO6	Build and integrate different software modules	K3
CO7	Document and deploy the product / project	K3

SYNOPSIS

Existing systems for community mutual aid and disaster relief are often fragmented and inefficient. Everyday community sharing relies on informal, untrusted social media groups, while disaster response is frequently hampered by uncoordinated and unneeded donations, creating significant logistical burdens for relief organizations. The design and implementation of **“SAHAAYIKHA”**, a web application that integrates peer-to-peer mutual aid with organized disaster relief, addresses this gap. The proposed system provides a unified, trusted platform for users to "Trade" or "Share" items, while simultaneously enabling *verified* organizations to post specific "Disaster Needs" and manage incoming donation offers, ensuring all aid is targeted, transparent, and efficient.

The frontend of the application is built using HTML, CSS, and Bootstrap, with server-side rendering powered by Jinja2, offering a stable and responsive user interface. The backend is powered by Python (Flask) and employs a robust three-role (User, Organization, and Admin) access control system. All user, item, and donation data is stored securely in an SQLite relational database, managed via the SQLAlchemy ORM. The system distinguishes itself from existing platforms by *integrating* everyday peer-to-peer sharing with a *verified* disaster relief module. The Admin role ensures platform integrity by vetting and approving all organizations, building a crucial layer of trust. Instead of a simple classifieds board, the platform features distinct dashboards for all roles, an integrated chat system for managing trades and donation logistics, and a notification system powered by Firebase to facilitate real-time communication.

.

TABLE OF CONTENTS

INTRODUCTION

1. INTRODUCTION

1.1 About The Project

"Sahaayikha" is an innovative, Python-based web application (using the Flask framework) developed to address the need for a centralized mutual aid and disaster relief platform. In today's world, community support and efficient disaster response are essential. This platform allows users to post items they wish to trade or share directly with other community members. Furthermore, it provides a dedicated portal for verified organizations to request and manage donations during emergencies, such as natural disasters, contributing to a more organized relief effort.

The platform supports three distinct user roles: Admin, User, and Organization, each playing a vital role in the system's functionality. The Admin manages system integrity, approves new organizations to ensure legitimacy, and oversees all activities. Users (individuals) can register to post items, search for items, request trades or shares via a built-in chat, and make specific donation offers to organizations. Organizations (verified bodies like NGOs or relief camps) can post specific disaster-related needs, review and accept donation offers from users, and manage the pickup and completion of aid. This user-based structure ensures an efficient workflow and transparency in both peer-to-peer sharing and large-scale disaster relief.

1.1.1 Objective of The Project

The primary objective of this project is to create an all-encompassing, web-based platform designed to provide seamless peer-to-peer mutual aid (trading and sharing) services, paired with a robust and transparent system for managing disaster relief donations. With the increasing need for community resilience, this platform aims to meet the growing demand for an accessible and efficient way to connect those who have items to spare with those who need them. This platform allows individual users to easily list, find, and exchange goods, promoting a culture of sharing and reducing waste. The platform also includes a dedicated disaster relief function, which facilitates targeted and efficient donation management. By allowing verified organizations to post specific needs, it ensures that users provide relevant aid, which helps to reduce the logistical challenges and environmental impact associated with uncoordinated donation drives.

1.1.2 Scope of The Project

The scope of this project includes the creation of a comprehensive online platform for mutual aid and disaster relief, designed to handle various user roles such as Administrators, Users (individuals), and Organizations (verified aid groups). This platform allows for the streamlined posting, searching, and management of items for trade or share, along with a systematic way for organizations to post disaster needs and manage incoming donation offers. Its functionality spans from user/organization registration and admin approval to detailed item management, donation tracking, and direct communication via a built-in chat, making it a versatile tool for communities focused on fostering mutual aid and supporting organized, transparent disaster relief.

Data entry screens (for posting items, making offers, and registering) are designed to be user-friendly, requiring minimum typing from the user where possible.

- Not much training is required.
- The new web application is more user-friendly.
- It aims for digital-first management of aid and donations.
- Fast access to information (via search, filters, and dashboards).
- Efficient traceability of item statuses, trade requests, and donation offers.
- Duplication of data will be avoided through a normalized database structure.
- A menu-driven interface provides ease of use.
- Availability of previous data for future reference (e.g., item history, completed donations).

INITIAL INVESTIGATION AND FEASIBILITY STUDY

2.1 INITIAL INVESTIGATION

The initial investigation for this project highlights the growing need for both accessible community-based mutual aid and organized, efficient disaster relief channels. In many communities, individuals often struggle to find trustworthy platforms to trade or share goods directly with neighbours, resorting to fragmented and unverified social media groups. Furthermore, during emergencies or natural disasters, aid organizations are frequently overwhelmed with un-requested or inappropriate donations, which creates significant logistical burdens. There is a clear demand for a solution that addresses both everyday community sharing and acute disaster-response needs.

A community needs analysis reveals that while many platforms exist for second-hand selling or informal giving, few effectively combine peer-to-peer trading and sharing with a dedicated, verified portal for organizational disaster relief. By integrating both services, this project fills a significant gap. It caters to users who value community-building and sustainability by facilitating a local circular economy. Simultaneously, it provides a critical tool for NGOs and relief bodies to clearly broadcast their specific needs and manage incoming donation offers, ensuring that aid is timely, relevant, and efficient. This unique dual-focus makes the platform valuable to both individual users and community organizations.

The project's technical requirements involve creating a robust system with three distinct user roles: Admin, User (individuals), and Organization. Key functionalities include item posting (with 'Trade' or 'Share' types), advanced search with location filters, a built-in chat system for negotiation, and a complete disaster relief module where organizations can post needs and users can make specific donation offers. Python, with the Flask framework, was chosen for its flexibility, extensive libraries, and scalability, ensuring the platform is responsive, secure, and easy to maintain. With attention to usability, scalability, and data integrity, the project aims to deliver a reliable platform that strengthens community ties and promotes efficient, transparent aid.

2.2 FEASIBILITY STUDY

System feasibility is a test or evaluation of the complete system plan. Such an evaluation is necessary to define the application area along with the extent and capability to provide the scope of computerization together with suggested output and input format and potential benefits. Feasibility study is a proposal according to the workability, impact on the organization, ability to meet user's needs and efficient use of resources. The feasibility study is conducted to determine if the proposed system is feasible or not. Feasibility analysis evaluates the candidate systems and determines the best system that needs performance requirements. The purpose of feasibility study is to investigate the present system, evaluate the possible application of computer-based methods, select a tentative system, evaluate the cost and effectiveness of the proposed system, evaluate impact of the proposed system on existing personnel and ascertain the need for new personnel.

All projects are feasible when given unlimited resources and infinite time. It is both necessary and prudent to evaluate the feasibility of a project at the earliest possible time. A feasibility study is not warranted for systems in which economic justification is obvious, technical risk is low, few legal problems are expected and no reasonable alternative exists. 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 the business point of view and if it can be developed in the given existing budgetary constraints. The feasibility study should be relatively cheap and quick. The result should inform the decision of whether to go ahead with a more detailed analysis. Feasibility study may be documented as a separated report to higher officials of the top-level management and can be included as an appendix to the system specification. Feasibility and risk analysis are related in many ways. If there is more project risk then the feasibility of producing the quality software is reduced.

The key combinations are involved in the feasibility study:

- Economic Feasibility.
- Technical Feasibility.
- Behavioral Feasibility.
- Operational Feasibility

2.2.1 Economic Feasibility

The "Sahaayikha" application demonstrates strong economic feasibility due to its low operational cost. Leveraging a completely open-source stack (Python, Flask) eliminates software licensing fees. The platform's return on investment is measured in social value and logistical savings for aid organizations, not profit. By automating aid matching and providing a free platform for a community circular economy (trading/sharing), it reduces administrative overhead and community waste, ensuring a positive social return.

2.2.2 Technical Feasibility

From a technical standpoint, "Sahaayikha" is highly feasible. The project is built using Python and the Flask framework, providing a robust, secure, and scalable backend infrastructure. Python's vast ecosystem of libraries (such as Flask-Login for authentication, SQLAlchemy for database management, and Flask-WTF for forms) streamlines development and ensures maintainability. The application is designed to be scalable with standard cloud hosting options, allowing it to expand as the user base grows. Key integrations, such as the Firebase Admin SDK for real-time push notifications, are already successfully implemented. The use of Alembic for database migrations ensures that the platform's database schema can be reliably updated and maintained, making the system stable, secure, and adaptable to future needs.

.

2.2.3 Behavioral Feasibility

"Sahaayikha" is expected to achieve high user acceptance, as it aligns with the public's desire for a convenient, centralized platform for mutual aid and disaster response. The user-friendly interface caters to its three distinct user roles—Users, Organizations, and the Admin—ensuring smooth onboarding and minimal training. Organizations and Admins can effectively oversee their functions using intuitive dashboards, leading to an anticipated smooth adaptation by all stakeholders.

2.2.4 Operational Feasibility

The operational feasibility of "Sahaayikha" is reinforced by its clear role-based workflows. The platform allows the Admin to manage the crucial task of organization approval, ensuring system integrity. Users and Organizations manage their respective activities (posting items, managing needs, reviewing offers) through dedicated dashboards. Designed to scale, Sahaayikha can accommodate a growing number of users, organizations, and listings. Its integration of a transparent donation management system aligns with best practices for efficient and accountable disaster relief.

SYSTEM ANALYSIS

3.1 System Analysis

Analysis is a structured method for identifying and solving problems. Analysis implies breaking something into its parts so that the whole may be understood. The definition of system analysis not only process analysis but also that of synthesis, which implies the process of putting parts together to form a new whole. All the activities relating to the life cycle phase must be performed, managed, and documented. To design a system, we need requirements of the system, and the specification document is prepared in this phase. The purpose of this document is to specify the functional requirement of the software that is to be built. The main thing is to find what is to be done to solve the problems with the current system. In this phase, the problems or drawbacks of the current system are identified and the necessary actions to solve these problems are recommended.

3.2 Existing System

The existing system for managing mutual aid and disaster relief is fragmented and often inefficient. Typically, individuals rely on informal social media groups or general classifieds, while disaster response is handled by separate, single-purpose applications.

- **Disaster-Specific Apps** (e.g., AmritaKripa, Rebuild Kerala): These platforms excel during emergencies but are single-purpose and do not support every day, user-to-user mutual aid or item sharing.
- **General Classifieds** (e.g., OLX, Quikr): These are primarily commercial platforms focused on buying and selling. They lack dedicated features for non-monetary trading, sharing, or structured disaster relief.
- **Barter Platforms:** While these systems facilitate item exchanges, they are not designed to support the complex needs of organized disaster relief, lacking distinct roles for verified organizations.

The core problem is this fragmentation. No single platform effectively unifies everyday peer-to-peer sharing and trading with a robust, organization-led disaster response system. This disconnection can lead to logistical challenges for relief organizations (receiving unneeded items) and missed opportunities for community sharing.

3.3 Proposed System

The proposed system, "Sahaayikha," is an integrated web platform designed to streamline both peer-to-peer mutual aid and organized disaster relief into a single, efficient solution. Individual users will be able to easily post items for "Trade" or "Share", search for items, and connect with other users via a built-in chat system to arrange exchanges. This fosters a community-based circular economy.
In addition to the mutual aid features, the platform incorporates a dedicated disaster relief module. Verified organizations can post specific "Disaster Needs," allowing users to make targeted "Donation Offers" of items that are actually required. Organizations can then review these offers, manage pickups, and track donations, ensuring a transparent and efficient aid process. The system supports three user roles—Admin (manages organizations), User (for trade/share/donations), and Organization (manages relief needs)—ensuring an efficient workflow and accountability for both everyday sharing and emergency response.

3.4 Software Requirement Specification (SRS)

This Software Requirements Specification (SRS) document provides a detailed description of the "Sahaayikha" Mutual Aid and Disaster Relief System. It outlines the system's functional and non-functional requirements, including user roles, interfaces, and overall system behavior.

A. Purpose

The purpose of this system is to provide a comprehensive web platform that allows users to post items for peer-to-peer trade or sharing and enables verified organizations to manage disaster relief donations. This system aims to enhance community resilience, promote sustainability through a circular economy, and streamline the management of disaster aid.

B. Scope

The scope of the project includes the development of an online platform with the following features:

- User and Organization registration and profile management.
- Posting of items for "Trade" or "Share" by users.
- Posting of "Disaster Needs" by verified organizations.
- System for users to make "Donation Offers" against specific needs.
- Role-based access for Admin, User (individuals), and Organization.
- Real-time status updates via in-app notifications and Firebase push notifications.
- Administrative management of users and organization approvals.
- Built-in chat system for user-to-user and user-to-organization communication.

C. Overall Description

❖ Product Perspective

The "Sahaayikha" web application is a new, independent system designed to streamline mutual aid and disaster relief. It is a web-based platform that leverages modern web development technologies such as HTML, CSS, Bootstrap, and JavaScript. The backend uses Python, the Flask framework, and SQLAlchemy for database management with SQLite.

❖ Product Functions

- **Admin Functions:** Organization approval/rejection, user account management (block/unblock), system settings, viewing reports and feedback.
- **Organization Functions:** Post and manage disaster needs, review/accept/reject donation offers, update donation pickup status, and chat with donors.
- **User Functions:** Post items (trade/share), search/filter items, request items (via chat/trade), make donation offers, bookmark items, and follow categories for notifications.

❖ User Characteristics

- **Admin:** Experienced in managing web applications, capable of verifying organization credentials.
- **Organization:** Staff from a verified NGO or relief group, capable of managing online donation requests and logistics.

- **User:** General community members with basic knowledge of online platforms and posting items.

❖ **Constraints**

- The system must be compatible with modern web browsers on desktop and mobile.
- The application must ensure data security and privacy (e.g., hashed passwords).
- The system must handle multiple user requests simultaneously.
- Requires a configured Firebase project for push notifications to function.

❖ **Assumptions and Dependencies**

- The system assumes a stable internet connection for all users.
- The platform's integrity is dependent on the Admin to approve only legitimate organizations.
- Dependent on users and organizations to keep their posts and statuses updated.

D. Functional Requirements

❖ **User Registration and Authentication**

- Users and Organizations should be able to register using separate forms.
- All users must be able to log in and log out securely (using Flask-Login).
- Registration includes email verification via OTP.

❖ **Profile Management**

- Users/Orgs can view and update their profile information (name, location, photo)
- Users/Orgs can view their history of posted items, trade requests, and donation offers.

❖ **Item Posting (Trade/Share)**

- Users can post items with details: title, description, category/sub-category, condition, location, type (Trade/Share), and images.

❖ **Disaster Relief Management**

- Organizations can post "Disaster Needs" detailing required items and location.
- Users can make "Donation Offers" against a need, specifying items and quantities.
- Organizations can review, accept, or reject items within an offer and manage pickup status.

❖ **Role-Based Access Control**
- **Admin:** Manage organization accounts (approve/reject), manage user accounts (block/unblock).
- **Organization:** Full control over their own disaster needs and incoming donation offers.
- **User:** Full control over their own posted items, trade requests, and donation offers. Can chat, bookmark, and report other users/items.

❖ **Notifications and Updates**
- Users should receive in-app and Firebase push notifications for new chat messages, trade request updates, and donation offer status changes.

E. Non-Functional Requirements

❖ **Performance**
- The system should handle a reasonable number of concurrent users without performance degradation.
- Response time for user actions should not exceed 2-3 seconds under normal load.

❖ **Usability**
- The interface should be intuitive and user-friendly, allowing users to navigate easily.
- Forms should have clear validation and feedback.

❖ **Security**
- User passwords must be securely hashed (using Werkzeug security).
- The system must implement role-based access controls to prevent unauthorized actions.

❖ **Scalability**
- The system should be designed to accommodate future growth in users, items, and organizations.

F. User Interface Requirements

- The application should have a responsive design (using Bootstrap) suitable for both desktop and mobile devices.
- Clear and consistent navigation should be maintained across all pages.

G. System Architecture

❖ System Design

The system architecture follows a client-server model. The client interacts with the application through a web browser, and the server processes requests using Python and Flask.

- **Front-end**: HTML, CSS, Bootstrap, JavaScript (including Leaflet.js for maps and Tom-Select.js for tagging).
- **Back-end**: Python, Flask.
- **Database**: SQLite (managed via SQLAlchemy ORM)

❖ Data Flow

1. **User Interaction**: Users interact with the web application via a browser.
2. **Request Handling**: The Flask server processes the requests, interacting with the database via SQLAlchemy.
3. **Response Generation**: The server sends back the processed data (rendered Jinja2 templates) to the client for display.

H. Conclusion

This SRS document outlines the key requirements for the Sahaayikha Mutual Aid and Disaster Relief System. The successful implementation of these requirements will ensure that the system meets user needs, provides a reliable platform for community sharing and aid coordination, and promotes community resilience and efficient disaster response.

MODULES

4.1 MODULES

In a software project, a module is a collection of source files and build settings that divides a project into distinct units of functionality. Modules can be used to: Make software easier to use, Define program boundaries, Implement separate modules for different areas of a program, and Partition the system design or code.
Modules can be designed to be reusable and can be called by the program as needed. They can also be imported from others, which may be called a library.

4.1.1 Admin Module

The Administrator is the person who manages the software. He/she is the person who focuses on the data, integrity, and reports of the software. He/she is the person who manages all other users and, most importantly, verifies the legitimacy of organizations registering on the platform.

a) The admin has supreme power over the system.

b) The admin enters valid email id and password; log on to the admin home page.

c) The admin shall able to view all the registered users and Organizations.

d) The admin can block or unblock any User or Organization.

e) The admin should have the permission to view the details of pending Organization registrations and has the provision to approve or reject them.

f) The admin is responsible for maintaining and updating the whole system.

g) Admin should have the provision to view reports submitted by users (e.g., reports on items or other users)

h) Admin should have the provision to view feedbacks regarding the system.

i) The system should have the provision to logout.

4.1.2 Organization Module

The Organization module enables verified organizations (like NGOs or relief camps) to post their needs and manage donations. They can view and update their own information, post disaster needs, and review offers from users.

a) The system should have a provision for an organization to register by filling a registration form (which awaits admin approval).
b) The registered and approved organization can log into the system by entering an email id and password.
c) After logging in, the organization has permission to view and edit its profile.
d) The organization should have the permission to edit its password.
e) The organization should have the permission to post new Disaster Needs, detailing required items, descriptions, and locations.
f) The organization has the provision to review incoming Donation Offers from users that are made against their needs.
g) The organization should have the provision to accept or reject items within a donation offer.
h) The organization should have the provision to update the pickup status of accepted donations.
i) The organization can chat with users who have made donation offers. j) System should have the provision to logout.

4.1.3 User Module

The User module allows general community members to easily post items for trade/share and to offer donations to organizations. They are the main users of this application. They can post items, search for items, and connect with other users or organizations

a) The system should have a provision to register a new user by filling a registration form.
b) The registered users can log into the system by entering an email id and password.
c) After logging in, the user must have a provision to view their profile and edit their profile.
d) Users should have the permission to change their password.
e) There should be a provision for users to post items for "Trade" or "Share" by submitting a form with item details and images.

f) A user should have a provision to view their posted items and manage them (edit, delete, mark as completed).

g) A user should have the permission to search and filter for items posted by other users.

h) A user should have the provision to initiate a chat with another user to request a "Share" item or propose a "Trade

i) A user should have the provision to view "Disaster Needs" posted by organizations.

j) A user should have the provision to make a Donation Offer against a specific disaster need.

k) A user should have the provision to view their sent donation offers and their status (Pending, Accepted, Rejected).

l) A user can bookmark items, follow categories, and receive notifications.

m) A user should have the provision to give feedback about the web application or report other users/items.

n) System should have the provision to logout.

FLOW DIAGRAMS

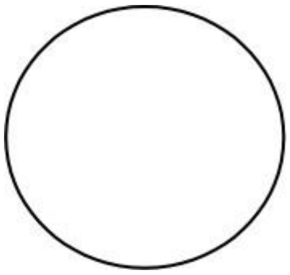
5.1 Data Flow Diagram (DFD)

DFD is the abbreviation for Data Flow Diagram. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart. Data Flow Diagram can be represented in several ways. The DFD belongs to structured-analysis modeling tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in software-system processes.

Data Flow Diagram Symbols

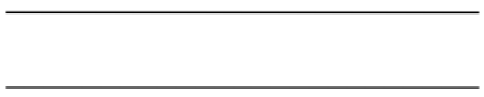
External entity: An outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.

Process: Input to output transformation in a system takes place because of process function. The symbols of a process are rectangular with rounded corners, oval, rectangle or a circle. The process is named a short sentence, in one word or a phrase to express its essence.



Data Flow: Data flow describes the information transferring between different parts of the systems. The arrow symbol is the symbol of data flow. A relatable name should be given to the flow to determine the information which is being moved. Data flow also represent material along with information that is being moved. Material shifts are modeled in systems that are not merely informative. A given flow should only transfer a single type of information. The direction of flow is represented by the arrow which can also be bi- directional.

Data Store: Also known as warehouse. The data is stored in the warehouse for later use. Two horizontal lines represent the symbol of the store. The warehouse is simply not restricted to being a data file rather it can be anything like a folder with documents, an optical disc, a filing cabinet. The data warehouse can be viewed independent of its implementation. When the data flow from the warehouse it is considered as data reading and when data flows to the warehouse it is called data entry or data updation.



Levels of DFD

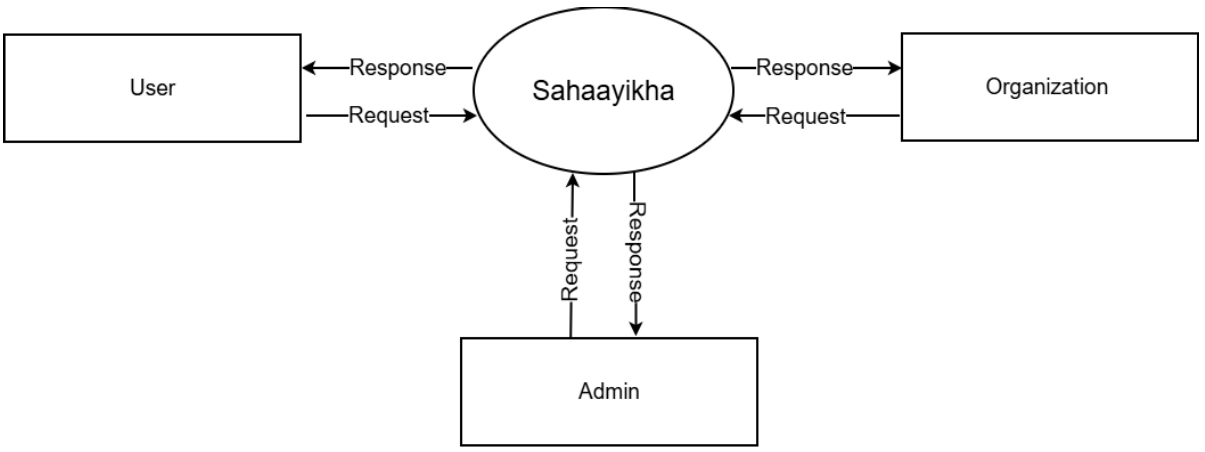
DFD uses hierarchy to maintain transparency thus multilevel DFD's can be created. Levels of DFD are as follows:

- 0-levelDFD
- 1-levelDFD
- 2-levelDFD

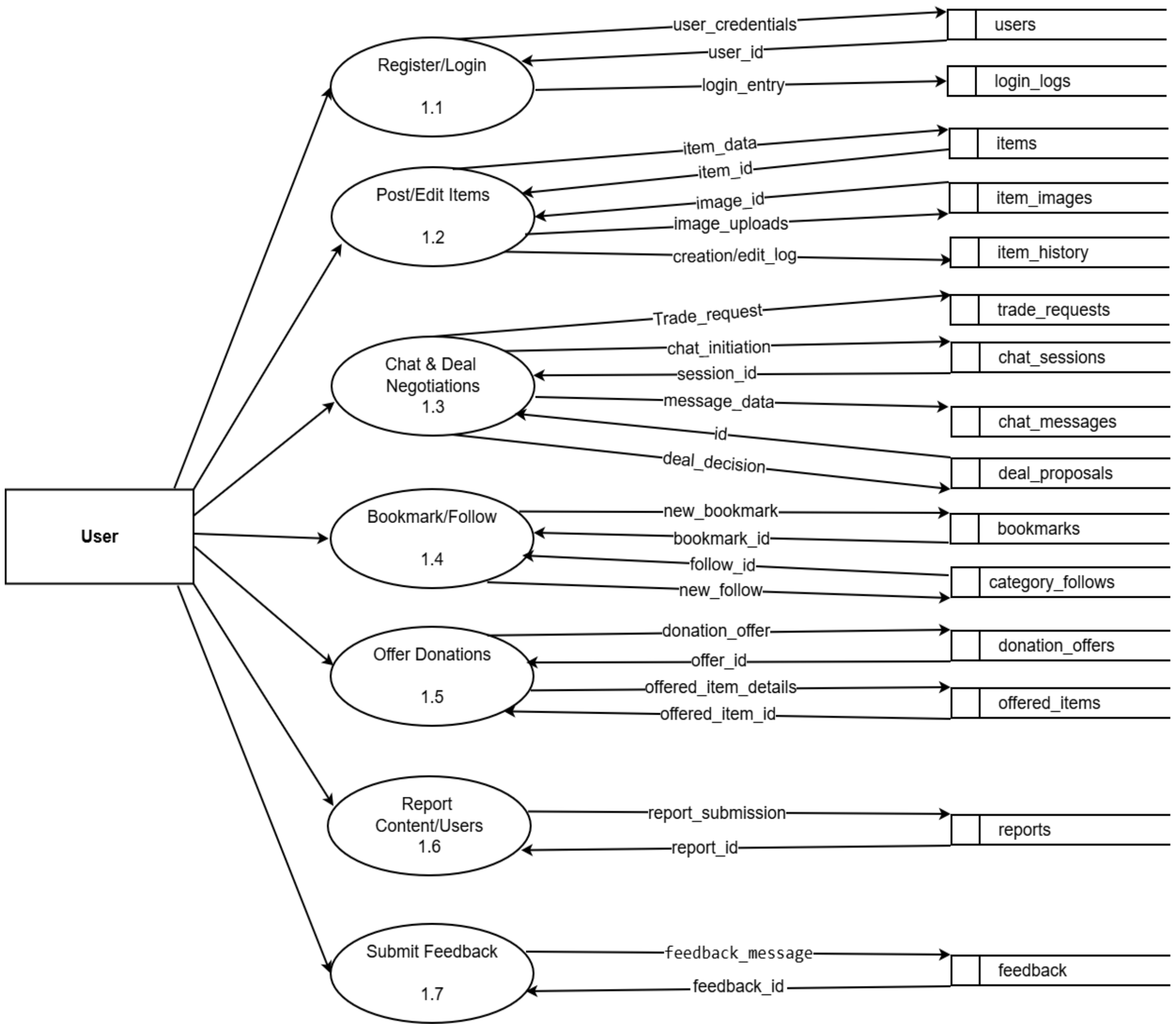
Rules for creating DFD

- ❖ The name of the entity should be easy and understandable without any extra assistance
- ❖ The processes should be numbered or put in ordered list to be referred easily.
- ❖ The DFD should maintain consistency across all the DFD levels.
- ❖ A single DFD can have maximum processes up to 9 and minimum 3 processes.

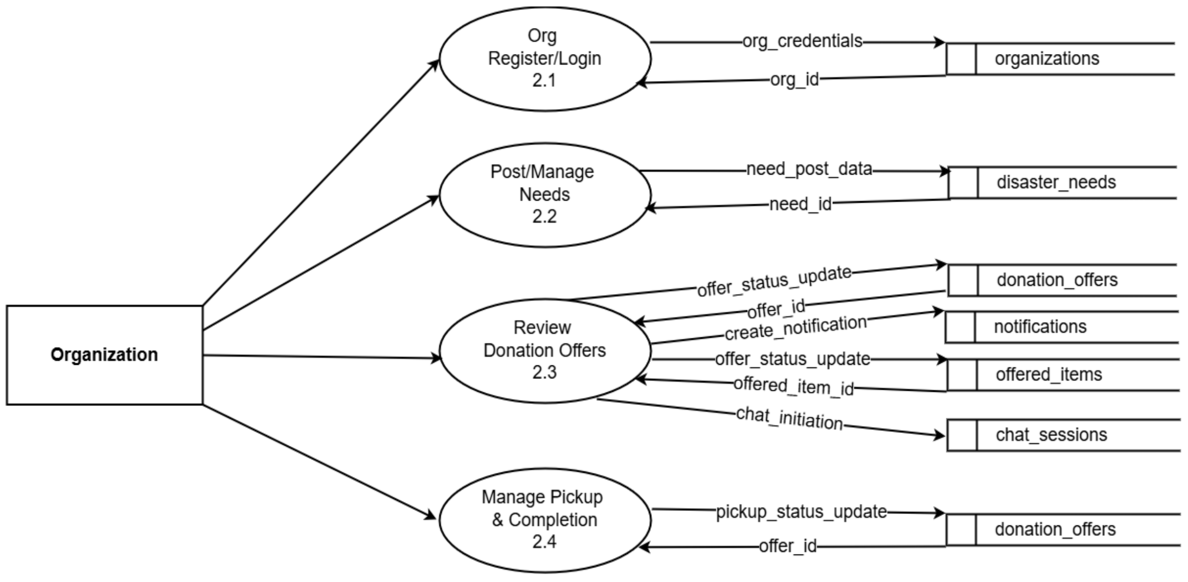
CONTEXT LEVEL-LEVEL 0 DFD



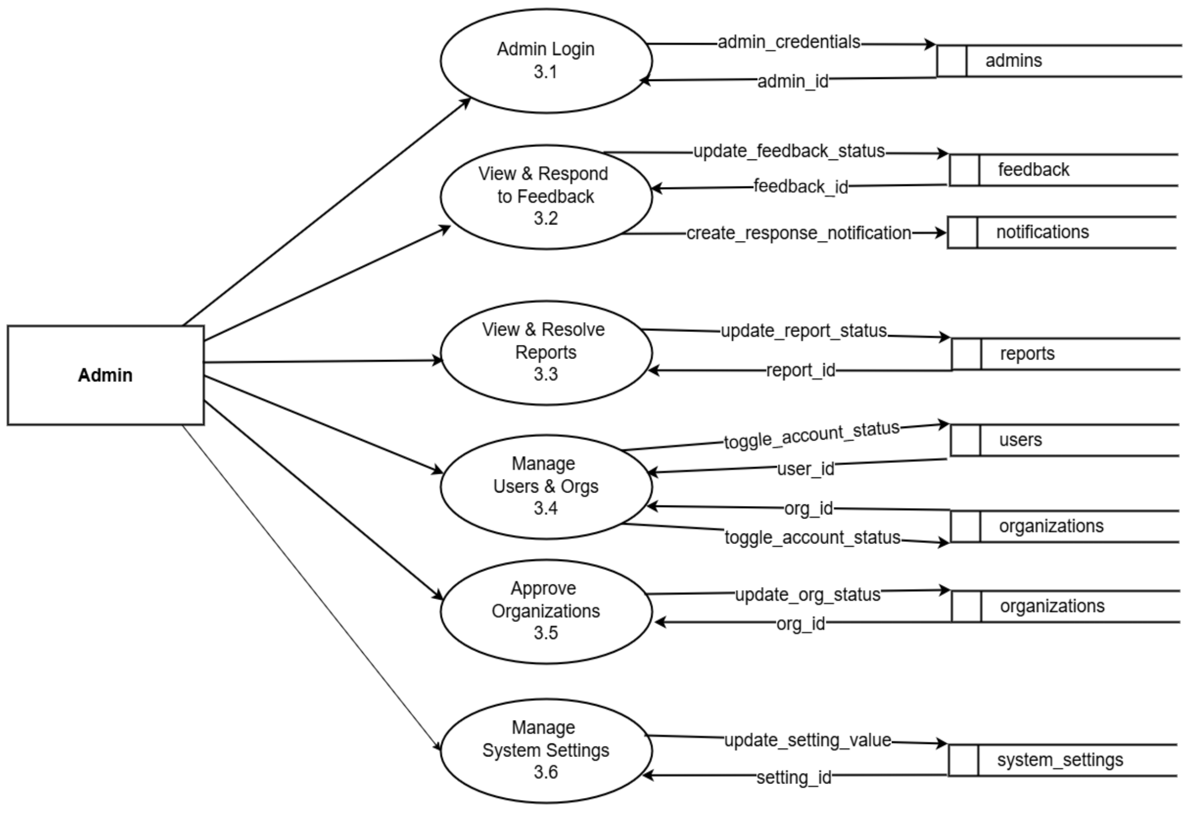
LEVEL 1-USER



LEVEL 1-ORGANIZATION



LEVEL 1- ADMIN



SYSTEM DESIGN

6.1 SYSTEM DESIGN

System designing is the process of defining the architecture, components, modules, interfaces and data for a system to satisfy specified requirements. it is a solution to a “how to” approach compared to system analysis which is a “what is” orientation. it translates the system requirements into ways of making them operational. The design phase focuses on the detailed implementation of the system recommended in the feasibility study. The system which is in making is developed by working on two different modules and combining them to work as a single unit. That single unit is the one which is known as the new software. We go through the different design strategies to design the system we are talking about. In the input design we decide which type of input screens are going to be used for the system in making. In the output design we decide the output screens and the reports that will be used to give the output and in the database design we decide what all tables will be required and what all fields will be there in those tables. Each of them discussed briefly below.

6.2 Input Design

Input design converts user-oriented inputs to computer-based formats, which requires careful attention. The collection of input data is the most expensive part of the system in terms of the equipment used and the number of people involved. In input design, data is accepted for computer processing and input to the system is done through mapping via a map support or links. Inaccurate input data is the most common cause of errors in data processing. The input screens need to be designed more carefully and logically. A set of menus is provided which help for better application navigation. While entering data in the input forms, proper validation checks are done and messages will be generated by the system if incorrect data has been entered. The objective of input design is to create an input layout that is easy to follow and prevent operator errors. It covers all phases of input from creation of initial data into actual entry of the data to the system for processing. The input design is the link that ties the system into world of its users. The user interface design is very important for any application. The interface design defines how the software communication within itself, to system that interpreted with it and with human who use it. The input design requirements such as user friendliness, consistent format and interaction dialogue for giving the right message and help for the user at right time are also considered for the development of the project.

6.3 Output Design

Outputs are the most important and useful information to the user and to the department. Intelligent output designs will improve systems relationships with the user and help much in decision-making. Outputs are also used to provide a permanent hard copy of the results for later use. The forms used in the system are shown in the appendix. The outputs also vary in terms of their contents, frequency, timing and format. The users of the output, its purpose and

sequence of details to be printed are all considered. The output forms a system in the justification for its existence. If the outputs are inadequate in any way, the system itself is inadequate. The basic requirements of output are that it should be accurate, timely and appropriate, in terms of content, medium and layout for its intended purpose. Hence it is necessary to design output so that the objectives of the system are met in the best possible manner.

6.4 Table Design

he efficiency of an application using SQLite (via the SQLAlchemy ORM) is mainly dependent upon the database tables, the fields in each table and joined using the fields contained in them to retrieve the necessary information. A table is a set of data elements that is organized using a model of vertical columns and horizontal rows. A table has a specified number of columns, but can have any number of rows. Each row is identified by the values appearing in a particular column subset which has been identified as a unique key index. The primary objective of a database design is fast response time to inquiries, more information at low cost, control of redundancy, clarity and ease of use, accuracy and integrity of the system fast recovery and availability of powerful end-user language.
There are mainly 20 tables in the project. They are,

1. users
2. admins
3. organizations
4. login_logs
5. items
6. trade_requests

7. deal_proposals
8. item_images
9. item_history
10. chat_sessions
11. chat_messages
12. disaster_needs
13. donation_offers
14. offered_items
15. feedback
16. reports
17. bookmarks
18. category_follows
19. notifications
20. system_settings

1. Table: users

Description: This table is used to store the details of individual users.

Sl.no	Field Name	Data type	Size	Constraints
1	user_id	Integer		Primary Key
2	first_name	String	255	Not Null
3	last_name	String	255	
4	email	String	255	Not Null, Unique
5	password	String	255	Not Null
6	phone	String	50	
7	location	String	255	
8	profile_picture	String	255	
9	status	String	50	Default 'Active'
10	created_at	DateTime		Default now
11	fcm_token	String	255	
12	otp	String	6	
13	otp_expiry	DateTime		
14	is_verified	Boolean		Default False
15	latitude	Float		
16	longitude	Float		
17	search_radius	Integer		Default 20

2. Table: admins

Description: Description: This table is used to store the details of administrators.

Sl.no	Field Name	Data type	Size	Constraints
1	admin_id	Integer		Primary Key
2	first_name	String	255	
3	last_name	String	255	
4	email	String	255	Not Null, Unique
5	password	String	255	Not Null
6	status	String	50	Default 'Active'
7	created_at	DateTime		Default now

3. Table: organizations

Description: This table is used to store details of verified organizations.

Sl.no	Field Name	Data type	Size	Constraints
1	org_id	Integer		Primary Key
2	name	String	255	Not Null
3	email	String	255	Not Null, Unique
4	password	String	255	Not Null
5	phone	String	100	
6	location	String	255	
7	profile_picture	String	255	
8	status	String	50	Default 'Pending'
9	registered_at	DateTime		Default now
10	description	Text		
11	otp	String	6	
12	otp_expiry	DateTime		
13	is_verified	Boolean		Default False

4. Table: login_logs

Description: his table is used to log user login attempts.

Sl.no	Field Name	Data type	Size	Constraints
1	log_id	Integer		Primary Key
2	user_id	Integer		Not Null, Foreign Key
3	login_time	DateTime		Default now
4	ip_address	String	100	

5. Table: items

Description: This table is used to store details of items posted for trade or share.

Sl.no	Field Name	Data type	Size	Constraints
1		Integer		Primary Key
2	user_id	Integer		Not Null, Foreign Key
3	title	String	255	Not Null
4	description	Text		
5	category	String	255	
6	sub_category	String	255	
7	type	String	50	
8	condition	String	50	
9	urgency_level	String	50	
10	expected_return_category	String	255	
11	expected_return_sub_category	String	255	
12	location	String	255	
13	status	String	50	Default 'Active'
14	created_at	DateTime		Default now
15	deal_finalized_at	DateTime		
16	latitude	Float		
17	longitude	Float		

6. Table: trade_requests

Description: This table is used to manage trade requests between users for items.

Sl.no	Field Name	Data type	Size	Constraints
1	id	Integer		Primary Key
2	item_offered_id	Integer		Foreign Key
3	item_requested_id	Integer		Not Null, Foreign Key
4	requester_id	Integer		Not Null, Foreign Key
5	owner_id	Integer		Not Null, Foreign Key
6	status	String	50	Default 'pending'
7	created_at	DateTime		Default now

7. Table: deal_proposals

Description: This table manages the deal confirmation status within a chat.

Sl.no	Field Name	Data type	Size	Constraints
1	id	Integer		Primary Key
2	chat_session_id	Integer		Not Null, Foreign Key, Unique
3	proposer_status	String	50	Not Null, Default 'pending'
4	owner_status	String	50	Null, Default 'pending'
5	updated_at	DateTime		Default now

8. Table: item_images

Description: This table is used to store image URLs for items.

Sl.no	Field Name	Data type	Size	Constraints
1	image_id	Integer		Primary Key
2	item_id	Integer		Not Null, Foreign Key
3	image_url	String	255	Not Null
4	uploaded_at	DateTime		Default now

9. Table: item_history

Description: his table stores a log of actions performed on an item.

Sl.no	Field Name	Data type	Size	Constraints
1	history_id	Integer		Primary Key
2	item_id	Integer		Not Null, Foreign Key
3	user_id	Integer		Foreign Key
4	action	String	255	
5	timestamp	DateTime		Default now

10. Table: chat_sessions

Description: This table is used to define a chat room between participants.

Sl.no	Field Name	Data type	Size	Constraints
1	session_id	Integer		Primary Key
2	trade_item_id	Integer		Foreign Key
3	disaster_need_id	Integer		Foreign Key
4	donation_offer_id	Integer		Foreign Key, Unique
5	user_one_id	Integer		Not Null, Foreign Key
6	user_two_id	Integer		Foreign Key
7	participant_org_id	Integer		Foreign Key
8	status	String	50	Default 'Active'
9	started_at	DateTime		Default now

11. Table: chat_messages

Description: This table is used to store individual chat messages.

Sl.no	Field Name	Data type	Size	Constraints
1	message_id	Integer		Primary Key
2	session_id	Integer		Not Null, Foreign Key
3	sender_type	String	50	Not Null
4	sender_id	Integer		Not Null
5	message	Text		
6	image_url	String	255	
7	timestamp	DateTime		Default now
8	is_read	Boolean		Default False
9	deleted_at	DateTime		

12. Table: disaster_needs

Description: This table is used to store disaster needs posted by organizations.

Sl.no	Field Name	Data type	Size	Constraints
1	need_id	Integer		Primary Key
2	org_id	Integer		Not Null, Foreign Key
3	title	String	255	
4	categories	Text		
5	description	Text		Not Null
6	location	String	255	
7	posted_at	DateTime		Default now

13. Table: donation_offers

Description: This table is used to store user donation offers against disaster needs.

Sl.no	Field Name	Data type	Size	Constraints
1	offer_id	Integer		Primary Key
2	user_id	Integer		Not Null, Foreign Key
3	need_id	Integer		Not Null, Foreign Key
4	org_id	Integer		Not Null, Foreign Key
5	status	String	50	Default 'Pending Review'
6	created_at	DateTime		Default now
7	verified_at	DateTime		
8	picked_up_at	DateTime		
9	completed_at	DateTime		
10	pickup_retries	Integer		Default 0
11	proof_image_url	String	255	

14. Table: offered_items

Description: This table stores the specific items within a single donation offer.

Sl.no	Field Name	Data type	Size	Constraints
1	offered_item_id	Integer		Primary Key
2	offer_id	Integer		Not Null, Foreign Key
3	title	String	255	Not Null
4	category	String	255	
5	description	Text		
6	quantity	Integer		Not Null, Default 1
7	condition	String	50	
8	image_url	String	255	
9	manufacture_date	Date		
10	expiry_date	Date		
11	status	String	50	Default 'Pending'

15. Table: feedback

Description: This table is used to store user-submitted feedback.

Sl.no	Field Name	Data type	Size	Constraints
1	feedback_id	Integer		Primary Key
2	user_id	Integer		Not Null, Foreign Key
3	message	Text		Not Null
4	submitted_at	DateTime		Default now
5	status	String	50	Default 'Open'

16. Table: reports

Description: This table is used to store user-submitted reports on items, orgs, etc.

Sl.no	Field Name	Data type	Size	Constraints
1	report_id	Integer		Primary Key
2	reported_by	Integer		Not Null, Foreign Key
3	item_id	Integer		Foreign Key
4	chat_session_id	Integer		Foreign Key
5	reported_org_id	Integer		Foreign Key
6	donation_offer_id	Integer		Foreign Key
7	reason	Text		Not Null
8	reported_at	DateTime		Default now
9	status	String	50	Default 'Pending'

17. Table: bookmarks

Description: his table is used to store items bookmarked by users or organizations.

Sl.no	Field Name	Data type	Size	Constraints
1	bookmark_id	Integer		Primary Key
2	user_id	Integer		Foreign Key
3	org_id	Integer		Foreign Key
4	item_id	Integer		Not Null, Foreign Key
5	saved_at	DateTime		Default now

18. Table: category_follows

Description: This table stores which categories a user is following.

Sl. no	Field Name	Data type	Size	Constraints
1	follow_id	Integer		Primary Key
2	user_id	Integer		Not Null, Foreign Key
3	category	String	255	
4	followed_at	DateTime		Default now

19. Table: notifications

Description: This table stores in-app notifications for users.

Sl.no	Field Name	Data type	Size	Constraints
1	notification_id	Integer		Primary Key
2	user_id	Integer		Not Null, Foreign Key
3	item_id	Integer		Foreign Key
4	message	Text		Not Null
5	sent_at	DateTime		Default now
6	Status	String	50	Default 'Unread'

20. Table: system_settings

Description: This table is used to store key-value system settings for the admin.

Sl.no	Field Name	Data type	Size	Constraints
1	setting_id	Integer		Primary Key
2	key	String	255	Not Null, Unique
3	value	String	255	
4	updated_at	DateTime		Default now

TOOLS AND PLATFORM

7. TOOLS AND PLATFORM

7.1 **Front End:** HTML, CSS, JavaScript, Bootstrap

The front end of the website is what users directly interact with, built using HTML, CSS, JavaScript, and Bootstrap. HTML (rendered via Jinja2 templates) provides the structure of web pages. CSS (style.css) styles these elements, adding colors, fonts, and layouts. JavaScript enables interactivity, allowing users to engage with dynamic features like the chat, map (using Leaflet.js), and notifications (using Firebase JS SDK). Finally, Bootstrap 5 offers pre-designed components and styles for creating responsive layouts across various devices, making development faster.

7.2 Back End: Python

he back end of the "Sahaayikha" application is developed using Python, a versatile and easy-to-read programming language. In this project, Python handles all server-side logic, including managing requests, processing data, managing user sessions, and interacting with the database. Python's readability and extensive libraries (like Flask, SQLAlchemy, Flask-Login) make it ideal for creating robust and scalable web applications, providing a smooth, efficient back-end experience.

7.3 FrameWork: Flask

Flask is a high-level micro-framework for building backend applications using Python. It simplifies web development by providing built-in tools for handling URL routing, request handling, and template rendering. Flask is highly extensible, allowing the project to use libraries like Flask-SQLAlchemy for database management, Flask-Login for authentication, and Flask-Migrate for database schema changes. It promotes organized and maintainable code, making it easy to build secure applications quickly.

7.4 Platform: Visual Studio Code

Visual Studio Code (VS Code) is a lightweight and powerful code editor used as the primary platform for developing and managing the project. With support for multiple programming languages, extensions, and features like debugging, syntax highlighting, and Git integration, VS

Code makes coding, testing, and deploying applications convenient and efficient. It’s especially helpful for Flask development, as it provides tools to streamline coding in Python and managing web files in one workspace

7.1 Database: SQLite

SQLite is a lightweight, self-contained database engine used by the application. It stores data in a single file (sahaayikha.db) on disk, making it simple to set up and ideal for development and small-to-medium projects. SQLite requires minimal configuration and provides all essential SQL features, which are managed through the SQLAlchemy ORM (Object Relational Mapper). This makes it highly effective for this application's data management needs.

TESTING

8. TESTING

8.1 Unit Testing

Unit Testing Here we test each module individually and integrated the overall system. Unit testing focuses verification efforts even in the smallest unit of software design in each module. This is known as” module testing”. The modules of the “Sahaayikha” are tested separately. This testing is carried out in the programming style itself. In this testing each module is focused to work satisfactorily as regard to expected output from the module. There are some validation checks for the fields. Unit testing gives stress on the modules independently of one another, to find errors. Different modules are tested against the specifications produced during the design of the modules. Unit testing is done to test the working of individual modules with test servers. Program unit is usually small enough that the programmer who developed it can test it in a great detail. Unit testing focuses first on that the modules to locate errors. These errors are verified and corrected and so that the unit perfectly fits to the project.

8.2 Integration Testing

Integration Testing Data can be lost across an interface, one module can have an adverse effect on the other sub functions, when combined they may not perform the desired functions. Integrated testing is the systematic testing to uncover the errors within the interface. This testing is done with simple data and the developed system has run successfully with this simple data. The need for integrated system is to find the overall system performance. After splitting the programs into units, the units were tested together to see the defects between each module and function. It is testing to one or more modules or functions together with the intent of finding interface defects between the modules or functions. Testing completed at as part of unit or functional testing, integration testing can involve putting together of groups of modules and functions with the goal of completing and verifying meets the system requirements.

8.3 System Testing

System testing focuses on testing the system as a whole. System Testing is a crucial step in Quality Management Process. In the Software Development Life Cycle, System Testing is the first level where the System is tested as a whole. The application/System is tested in environment that closely resembles the production environment where the application will be finally deployed. The perquisites for System Testing are: -

• All the components should have been successfully Unit Tested.
• All the components should have been successfully integrated.
• Testing should be completed in an environment closely resembling the production environment. When necessary, iterations of System Testing are done in multiple environments.

8.3.1 Acceptance Testing

Acceptance Testing The system was tested by a small client community to see if the program met the requirements defined the analysis stage. It was fond to be satisfactory. In this phase, the system is fully tested by the client community against the requirements defined in analysis and design stages, corrections are made as required, and the production system is built. User acceptance of the system is key factor for success of the system.

8.3.2 Validation Testing

Data validation is the process of testing the accuracy of data. A set of rules we can apply to a control to specify the type and range of data that can enter. It can be used to display error alert when users enter incorrect values in to a form. Now performing validation testing in system Centralized Social Welfare by undergoing validation for each tool and the validation succeeded when the software function in a manner that can be reasonably accepted, by the user.

8.3.3 Black box Testing

Testing Knowing the specified function that a product has been designed to perform, test can be conducted that demonstrates each function that is fully operational, at the same time searching for errors in each function. Black Box testing focuses on functional requirement of the software.

8.3.4 White box Testing

Testing Knowing the internal working of a product test can be conducted to ensure that "all gears mesh" that is internal operation performs according to specification and all internal components have been adequately exercise

8.4 Sample Test Cases

TC No.	Test Steps	Expected Result	Actual Result	Status
1	Run application and navigate to login screen	Login screen is displayed. A field for entering email address, a field for entering password and a button to submit should be present	Login screen has been displayed, fields for entering email address and password together with a log in button is available.	Pass
2	Enter an invalid email address and invalid password and press the button	A message should be displayed stating that email address and password are invalid	A message has been displayed stating that email address and password are invalid	Pass
3	Enter a valid email address and password and press the button	User must successfully login to the webpages.	A message has been displayed stating that the login successful and navigate into home page	Pass
4	Enter a valid email address and leave password and press the button	A message should be displayed stating that please enter the email address and password	A message has been displayed stating that please enter the email address and password	Pass
5	Leave email address and password and press the button	A message should be displayed stating that please enter the email address and password	A message has been displayed stating that please enter the email address and password	Pass
6	Leave email address and enter a valid password and press the button	A message should be displayed stating that please enter the email address and password	A message has been displayed stating that please enter the email address and password	Pass

SYSTEM IMPLEMENTATION

9.SYSTEM IMPLEMENTATION

9.1 Implementation

Implementation includes placing the system into operation and providing the users and operation personnel with the necessary documentation to use and maintain the new system. Implementation includes all those activities that take place to convert from the old system to the new. The new system may be totally new, replacing an existing system. Proper implementation is essential to provide a reliable system to meet the organizational requirements. Successful implementation may not guarantee improvement in the organization using the new system, as well as, improper installation will prevent. There are four methods for handling a system conversion. The Implementation Plan describes how the information system will be deployed, installed and transitioned into an operational system. The plan contains an overview of the system, a brief description of the major tasks involved in the implementation, the overall resources needed to support the implementation effort, and any site-specific implementation requirements. The plan is developed during the Design Phase and is updated during the Development Phase the final version is provided in the Integration and Test Phase and is used for guidance during the Implementation Phase. The implementation phase ends with an evaluation of the system after placing it into operation of time. The validity and proper functionality of all the modules of the developed application is assured during the process of implementation. Implementation is the process of assuring that the information system is operational and then allowing user to take over its operation for use and evaluation. Implementation is the stage in the project where the theoretical design is turned into a working system. The implementation phase constructs, installs and operated the new system. The most crucial stage in achieving a new successful system is that it works effectively and efficiently.

9.2 Problem Statement

In today's interconnected communities, significant challenges exist in efficiently coordinating mutual aid and disaster relief. Individuals and families in need often struggle to make their specific needs known, resulting in hardship and a lack of access to essential goods. Conversely, people who wish to donate or share surplus items often find the process difficult, with no clear, centralized way to see what is needed and where.

Additionally, this disconnect leads to significant waste, as usable items (like furniture, clothes, or books) are often discarded simply because finding a new home for them is too inconvenient. During disaster situations, this problem is magnified: relief organizations are inundated with unneeded items while critical supplies remain scarce, all due to a lack of a coordinating platform. Existing solutions are fragmented, forcing users to navigate multiple social media groups or opaque charity channels, which leads to inefficiencies and a mismatch between supply and demand. Thus, there is an urgent need for an integrated platform that streamlines the process of sharing, trading, and donating goods, offering convenience, direct communication, and targeted support for disaster relief, ultimately promoting sustainability and strengthening community resilience.

9.3 Problem Definition

The problem at hand is the inefficiency and fragmentation of traditional mutual aid and donation systems. While community generosity is high, the lack of a centralized, trusted platform creates significant challenges in connecting item-specific needs with available resources. Individuals frequently encounter difficulties in finding or requesting specific items, facing issues such as a lack of a trusted channel, inconvenient exchange processes, and inadequate communication with potential donors or recipients.

Furthermore, the improper disposal of still-usable goods poses environmental concerns, with many people lacking awareness of or access to easy redistribution options. The fragmented nature of existing solutions (like unorganized social media posts or local donation drives) exacerbates these challenges, resulting in wasted resources, logistical burdens for relief organizations, and unmet needs within the community. Thus, there is a critical need for a comprehensive platform that integrates peer-to-peer sharing, trading, and organized disaster relief, simplifying the user experience and fostering a more efficient and sustainable community support network

.

CONCLUSION

10. CONCLUSION

The project was successfully completed within the time span allotted. All the modules are tested separately and put together to form the main system. Finally, the modules are tested with real data and it worked successfully. Thus, the system has fulfilled the entire objective defined.
This project will help the community to connect users who need items with those who can provide them, facilitating mutual aid, donations, and trades in Sahaayikha. Our goal of developing this “Sahaayikha” has come to get a good result without many defects.
The main motive for developing this system is for the welfare of the society by providing a centralized platform for mutual aid, reducing waste by redistributing usable goods, and offering coordinated support during disaster relief. By connecting neighbours and organizations, the system aims to strengthen community resilience and promote a more sustainable, circular economy

FUTURE ENHANCEMENTS

The system has been designed in such a way that it can be modified with very little effort when such needs arise in the future. New features can be added with slight modifications of software which make it easy to expand the scope of this project. Though the system is working on various assumptions, it can be modified easily to any kind of requirements.
Even though we have tried our best to present the information effectively and efficiently, yet there can be further enhancement in the application. We have taken care of all the critical aspects, which were needed to be taken care of. Because of fast changes in the world of programming this system will gradually get outdated and less effective. For the time being it’s possible to overcome problems by amendments and minor modifications to acknowledge the need of fundamental design.
Though the new system provides base for improving the efficiency of operations, there are a lot of future enhancements that can be added to this project. Keeping this in view, the following new applications and modules are planned:

- Integrated Volunteer Delivery Network To solve the "last-mile" problem of exchanging goods, a new module will be created. This will introduce a new "Volunteer Rider" role. When two users confirm a deal, they can opt for "Community Delivery." This request is added to a dispatch board, which registered volunteers can accept, creating a full logistics chain (Pickup -> In Transit -> Delivered) within the platform, managed by trusted community members.

- Service & Volunteer Exchange Marketplace This enhancement expands Sahaayikha from a platform for *goods* to a platform for *services*. A new application module will be built allowing users to post requests and offers for "mutual aid services" such as "Help with moving," "Free tuition for 1 hour," "Gardening help," or "Errand running for elders." This directly broadens the "mutual aid" mission of the project.
- Scheduled Collection Drive Management (for Orgs) This feature is a new application for the Organization role. Organizations (like disaster relief groups or e-waste recyclers) could create and publicize "Scheduled Collection Drives" for specific dates and locations (e.g., "E-Waste Collection Drive - Kochi, Oct 25" or "Winter Blanket Drive - Main Hall"). Users could then see these drives on a map and "pledge" items, helping organizations plan logistics for large-scale collection efforts.
- Secure Payment Gateway & Escrow for Trades To make the "monetary offer" feature for Trade items robust and secure, a payment gateway (like Razorpay or Stripe) will be integrated. When a buyer's monetary offer is accepted, the funds will be held in an in-app escrow system. The money will only be released to the item's owner after both parties confirm the exchange is complete, preventing fraud and building trust.
- Native Mobile Application To improve accessibility, notifications, and location-based features, a dedicated native mobile application for Android and iOS will be developed. This application will use the same Flask backend API but will provide a much richer user experience, especially for the real-time chat, delivery tracking, and push notification features.

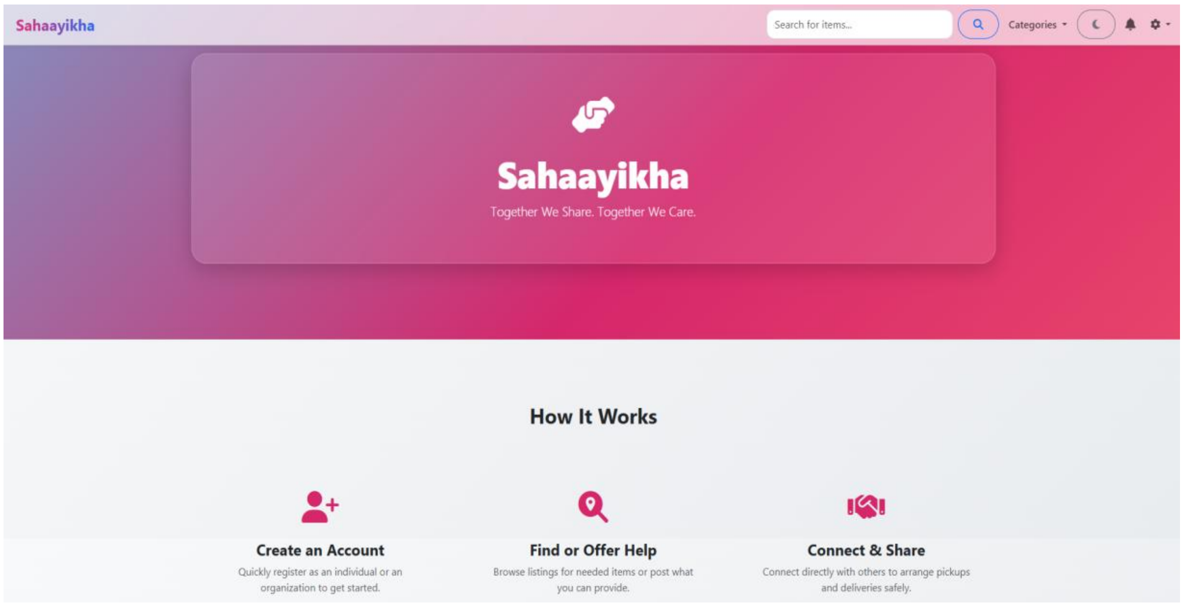
A provision has been made in the system to facilities easy modification updating in the future. Any modification will not affect the normal working of the system.

APPENDICES

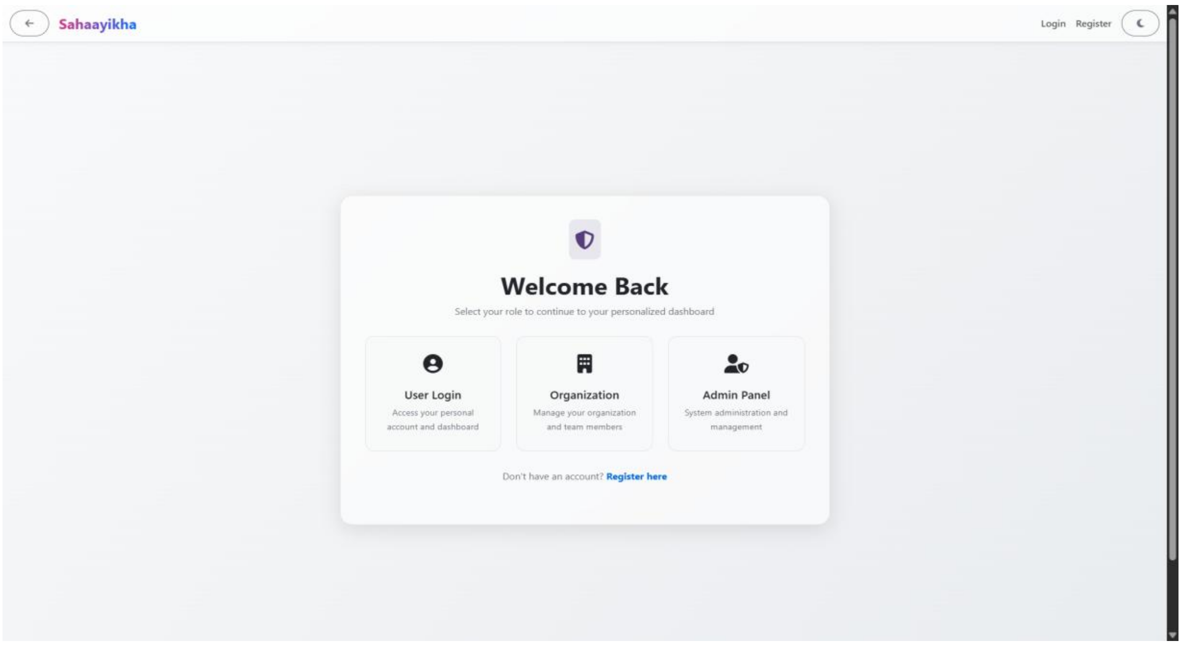
11. APPENDICES

APPENDICES A: Sample Screens

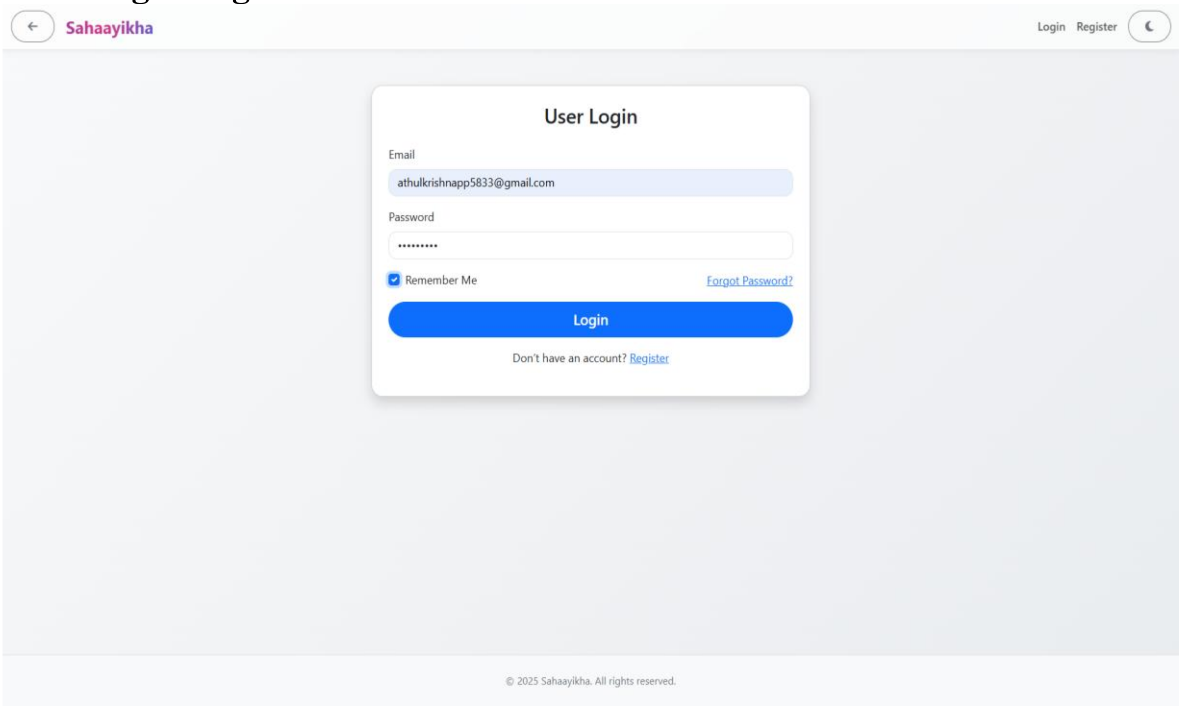
Home Page


Sahaayikha
Search for items...
Categories
Sahaayikha
Together We Share. Together We Care.
How It Works
Create an Account
Quickly register as an individual or an organization to get started.
Find or Offer Help
Browse listings for needed items or post what you can provide.
Connect & Share
Connect directly with others to arrange pickups and deliveries safely.

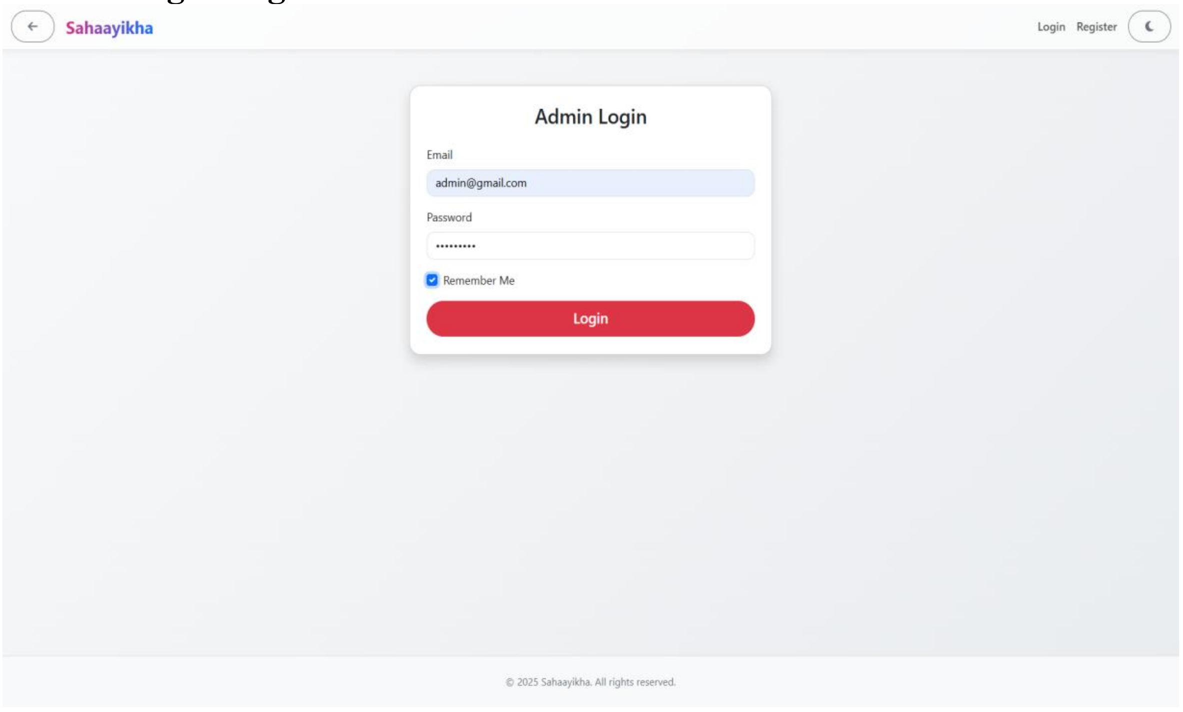
Login Page


Sahaayikha
Login
Register
Welcome Back
Select your role to continue to your personalized dashboard
User Login
Access your personal account and dashboard
Organization
Manage your organization and team members
Admin Panel
System administration and management
Don't have an account? Register here

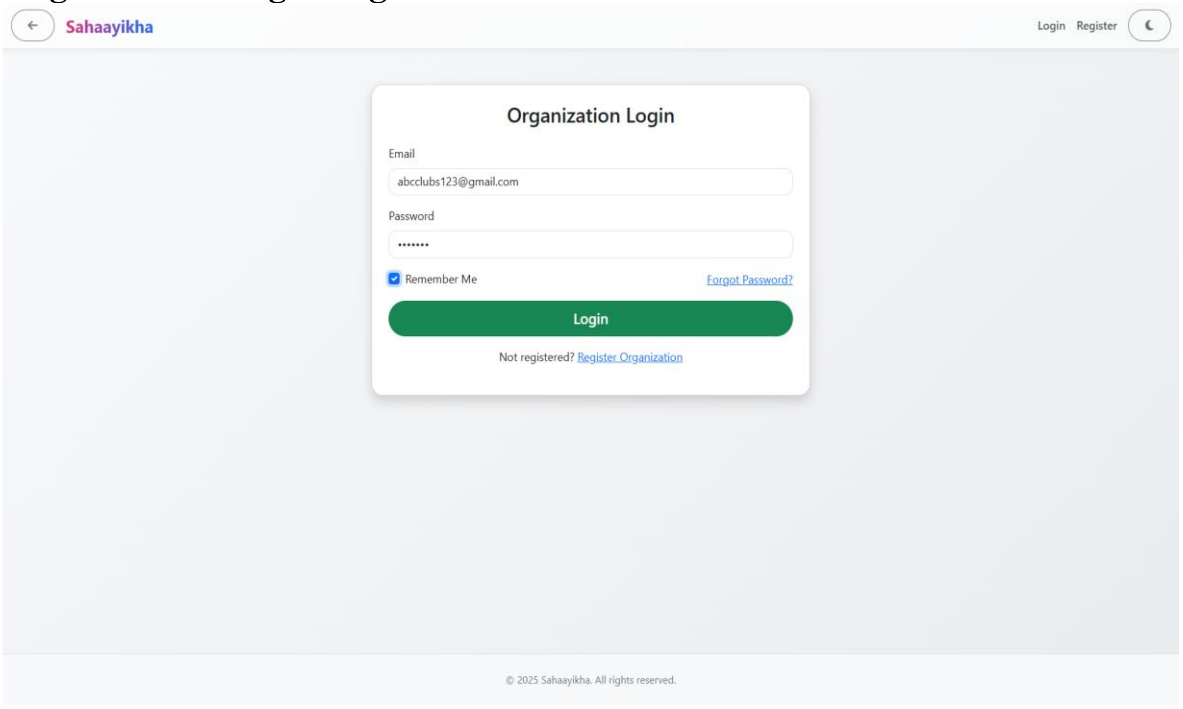
User Login Page


Sahaayikha
Login
Register
User Login
Email
athulkrishnapp5833@gmail.com
Password
•••••••••
Remember Me
Forgot Password?
Login
Don't have an account? Register
© 2025 Sahaayikha. All rights reserved.

Admin Login Page



Organization Login Page



Sahaayikha

Login Register

Organization Login

Email

abcclubs123@gmail.com

Password

•••••••

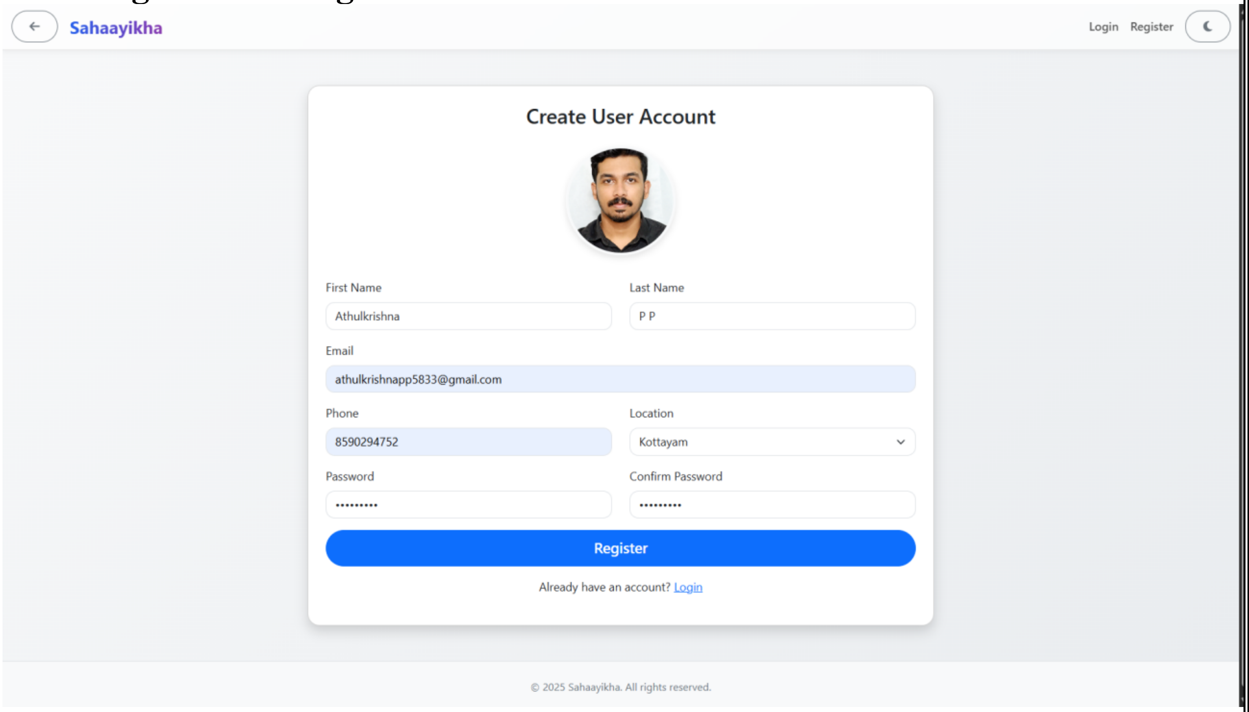
☑ Remember Me Forgot Password?

Login

Not registered? Register Organization

© 2025 Sahaayikha. All rights reserved.

User Registration Page



Sahaayikha

Login Register

Create User Account

First Name

Athulkrishna

Last Name

P P

Email

athulkrishnapp5833@gmail.com

Phone

8590294752

Location

Kottayam

Password

•••••••••

Confirm Password

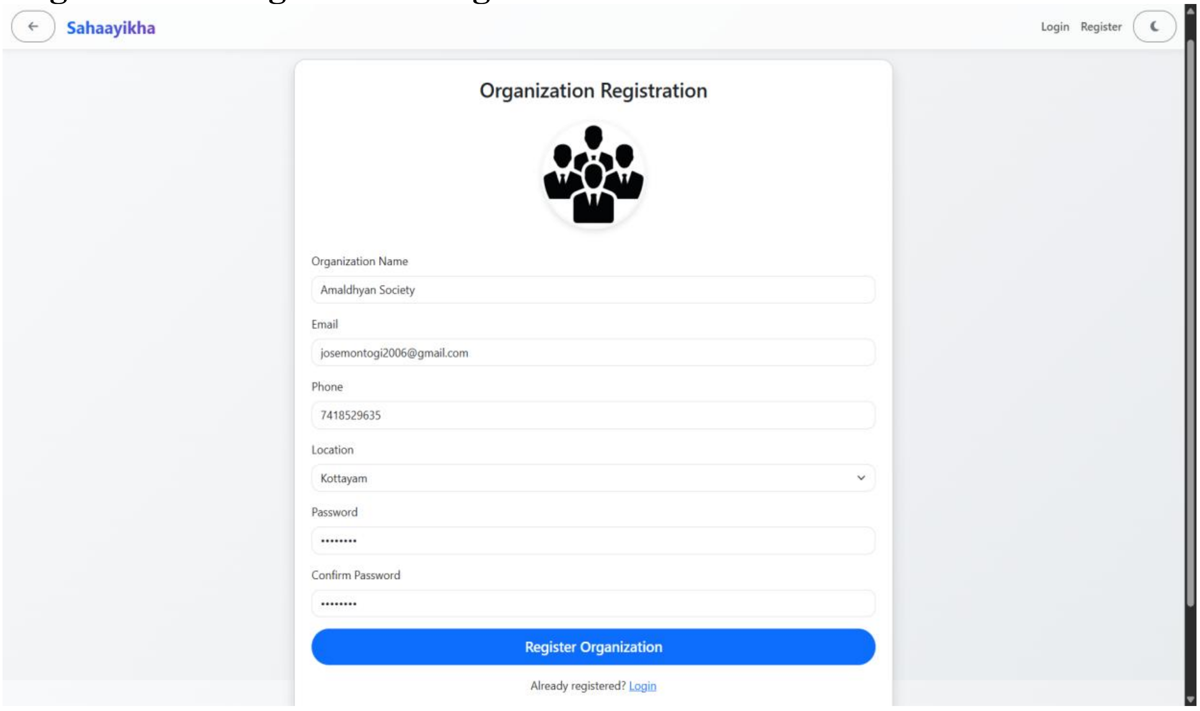
•••••••••

Register

Already have an account? Login

© 2025 Sahaayikha. All rights reserved.

Organization Registration Page



Sahaayikha

Login Register

Organization Registration

Organization Name

Amaldhyan Society

Email

josemontogi2006@gmail.com

Phone

7418529635

Location

Kottayam

Password

••••••••

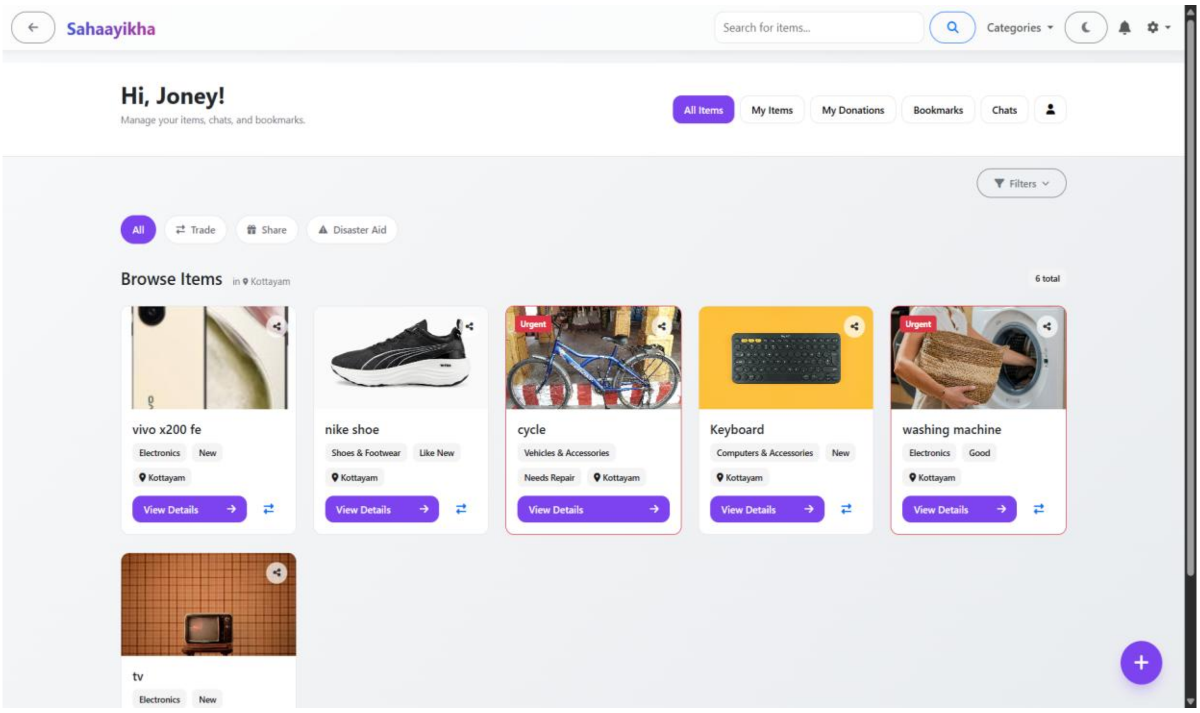
Confirm Password

••••••••

Register Organization

Already registered? Login

User Dashboard



Sahaayikha

Search for items... Categories

Hi, Joney!

Manage your items, chats, and bookmarks.

All Items My Items My Donations Bookmarks Chats

Filters

All Trade Share Disaster Aid

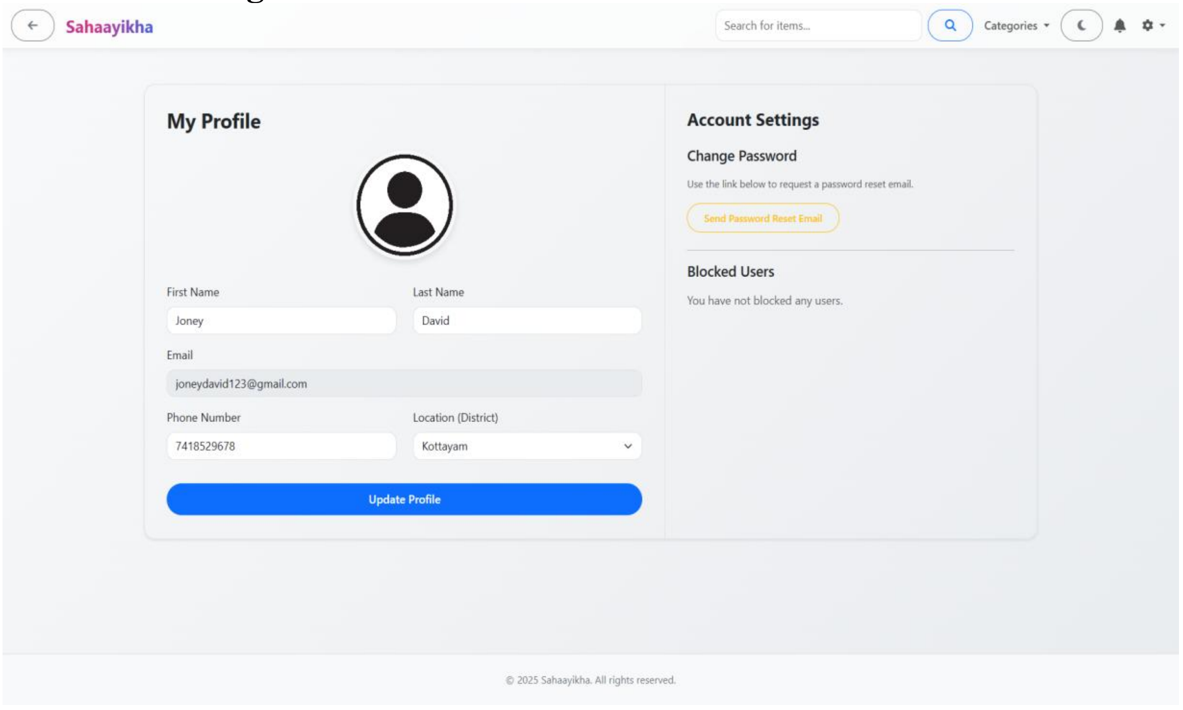
Browse Items in Kottayam 6 total

vivo x200 fe	nike shoe	Urgent cycle	Keyboard	Urgent washing machine
Electronics · New	Shoes & Footwear · Like New	Vehicles & Accessories · Needs Repair	Computers & Accessories · New	Electronics · Good
Kottayam	Kottayam	Kottayam	Kottayam	Kottayam
View Details	View Details	View Details	View Details	View Details

tv

Electronics New

User Profile Page



My Profile

First Name: Joney
Last Name: David
Email: joneydavid123@gmail.com
Phone Number: 7418529678
Location (District): Kottayam

Account Settings

Change Password

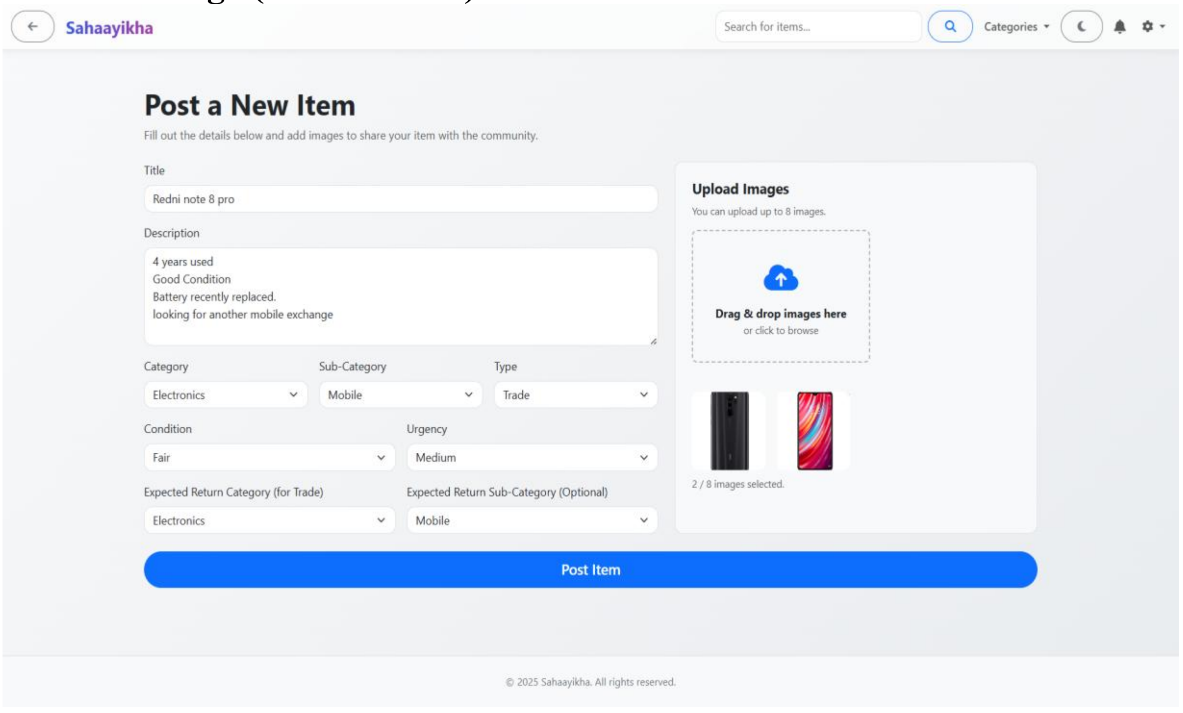
Use the link below to request a password reset email.

Blocked Users

You have not blocked any users.

© 2025 Sahaayikha. All rights reserved.

Post Item Page (Share/Trade)



Post a New Item

Fill out the details below and add images to share your item with the community.

Title: Redni note 8 pro

Description:
4 years used
Good Condition
Battery recently replaced.
looking for another mobile exchange

Category: Electronics
Sub-Category: Mobile
Type: Trade
Condition: Fair
Urgency: Medium
Expected Return Category (for Trade): Electronics
Expected Return Sub-Category (Optional): Mobile

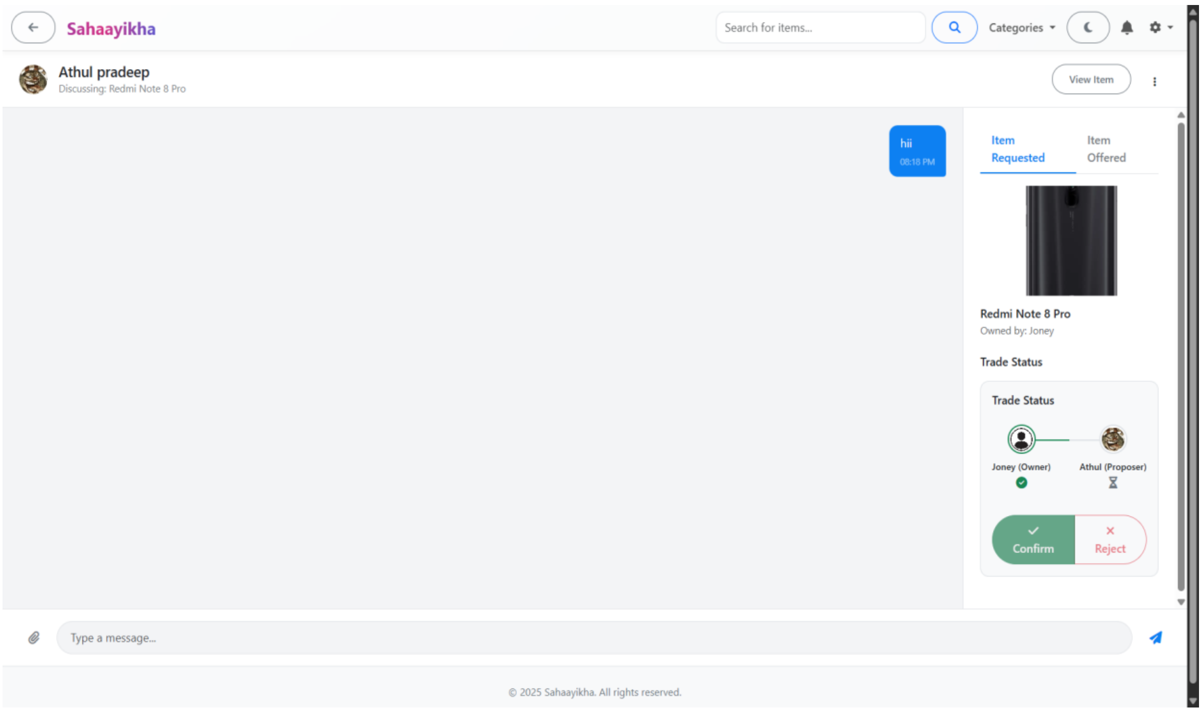
Upload Images

You can upload up to 8 images.

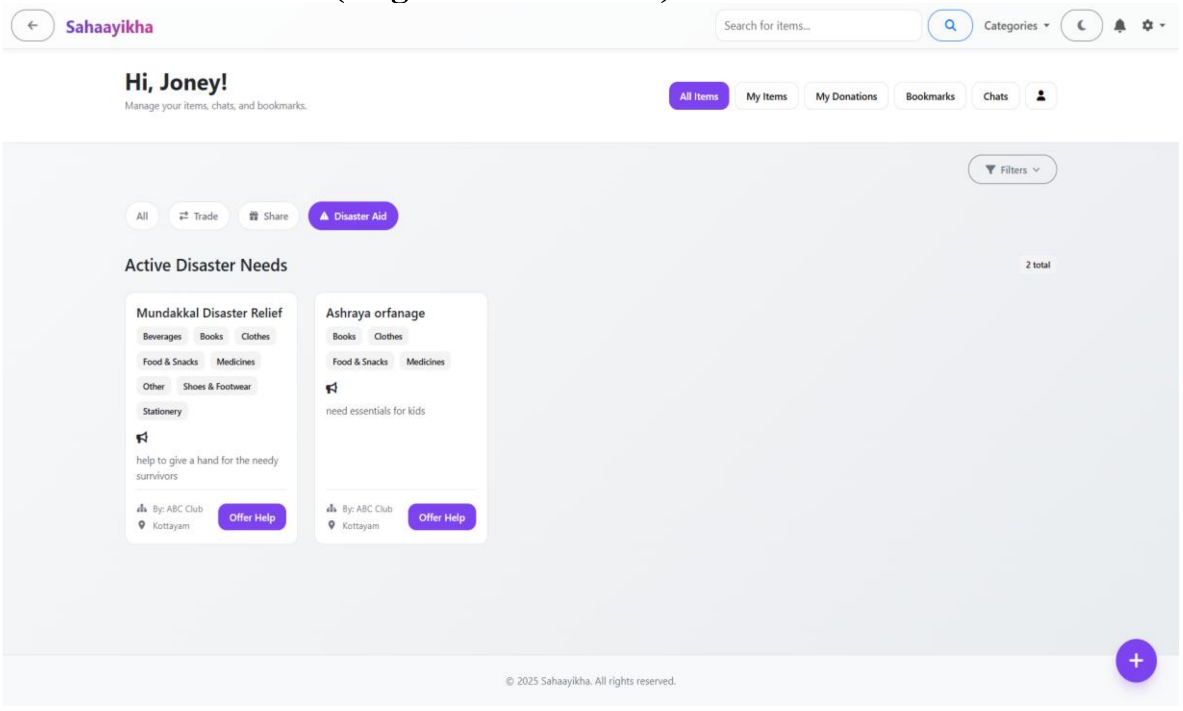
2 / 8 images selected.

© 2025 Sahaayikha. All rights reserved.

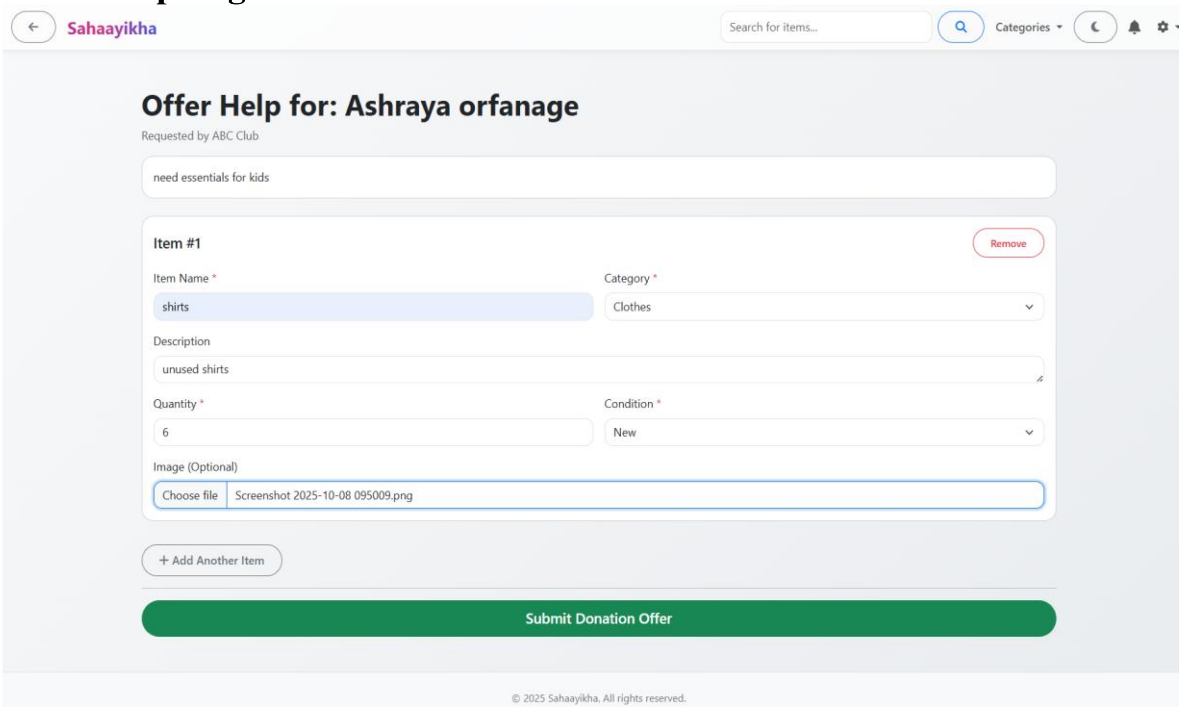
Chat & Deal Proposal Page



Disaster Relief Feed (Organization Needs)



Offer Help Page



Sahaayikha

Search for items... Categories

Offer Help for: Ashraya orfanage

Requested by ABC Club

need essentials for kids

Item #1 Remove

Item Name *: shirts

Category *: Clothes

Description: unused shirts

Quantity *: 6

Condition *: New

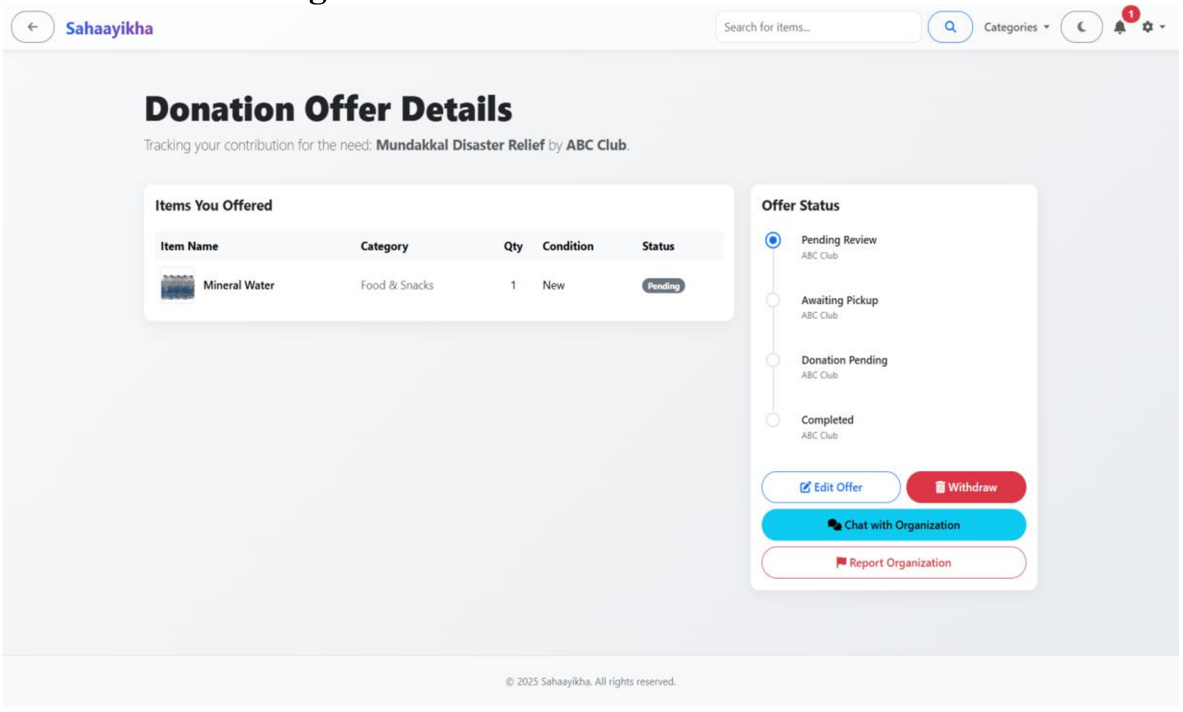
Image (Optional): Choose file Screenshot 2025-10-08 095009.png

+ Add Another Item

Submit Donation Offer

© 2025 Sahaayikha. All rights reserved.

Track Donation Page



Sahaayikha

Search for items... Categories

Donation Offer Details

Tracking your contribution for the need: **Mundakkal Disaster Relief** by **ABC Club**.

Items You Offered

Item Name	Category	Qty	Condition	Status
Mineral Water	Food & Snacks	1	New	Pending

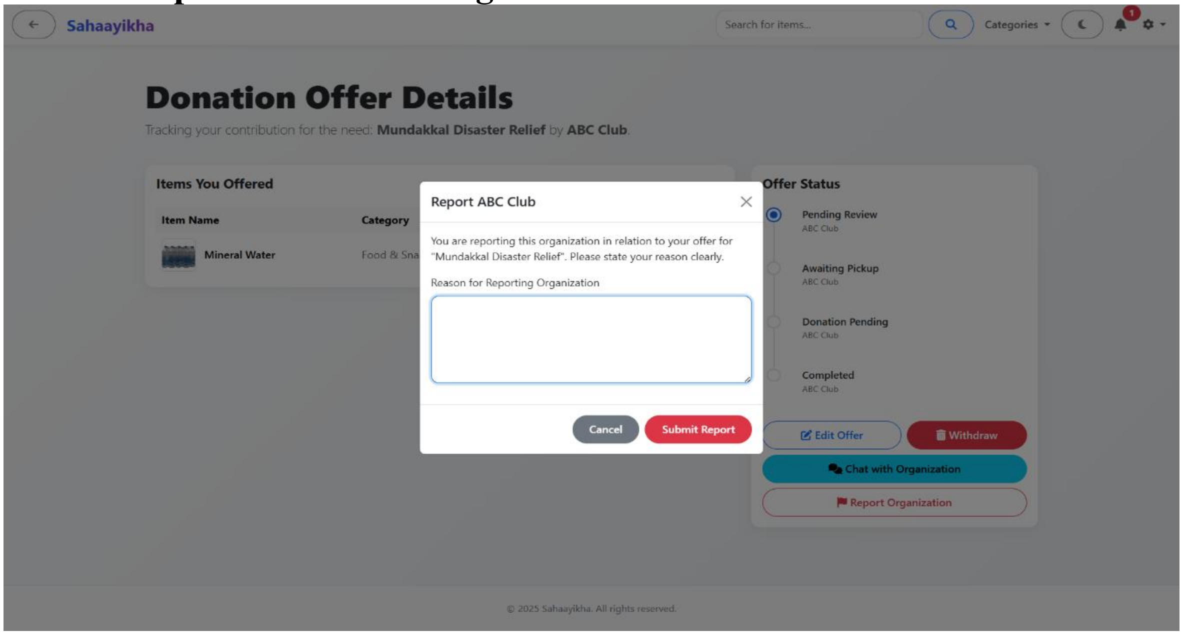
Offer Status

- Pending Review — ABC Club
- Awaiting Pickup — ABC Club
- Donation Pending — ABC Club
- Completed — ABC Club

Edit Offer | Withdraw | Chat with Organization | Report Organization

© 2025 Sahaayikha. All rights reserved.

Submit Report / Feedback Page



Donation Offer Details

Tracking your contribution for the need: **Mundakkal Disaster Relief** by **ABC Club**

Items You Offered

Item Name	Category
Mineral Water	Food & Sna

Offer Status

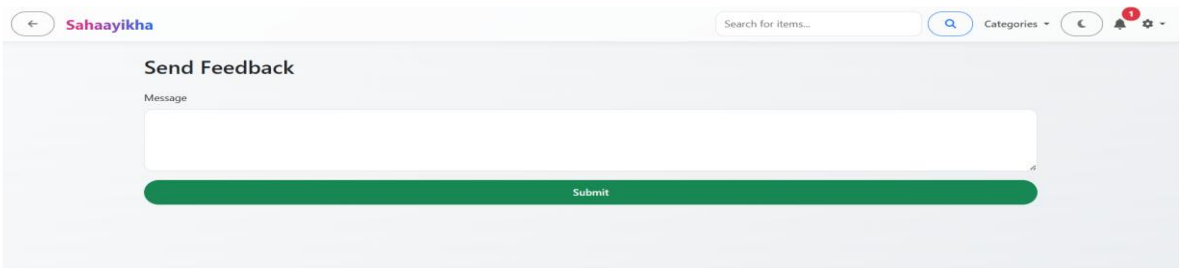
- Pending Review — ABC Club
- Awaiting Pickup — ABC Club
- Donation Pending — ABC Club
- Completed — ABC Club

Report ABC Club

You are reporting this organization in relation to your offer for "Mundakkal Disaster Relief". Please state your reason clearly.

Reason for Reporting Organization

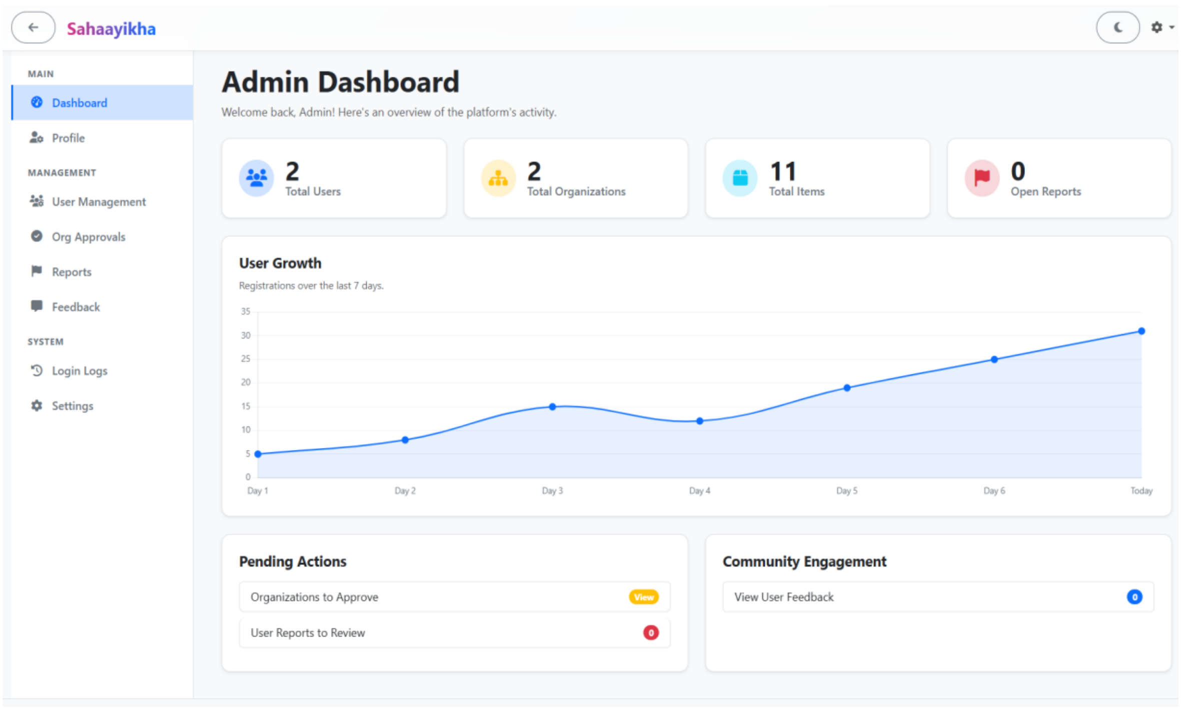
© 2025 Sahaayikha. All rights reserved.



Send Feedback

Message

Admin Dashboard



Admin Dashboard

Welcome back, Admin! Here's an overview of the platform's activity.

2 Total Users	2 Total Organizations	11 Total Items	0 Open Reports

User Growth

Registrations over the last 7 days.

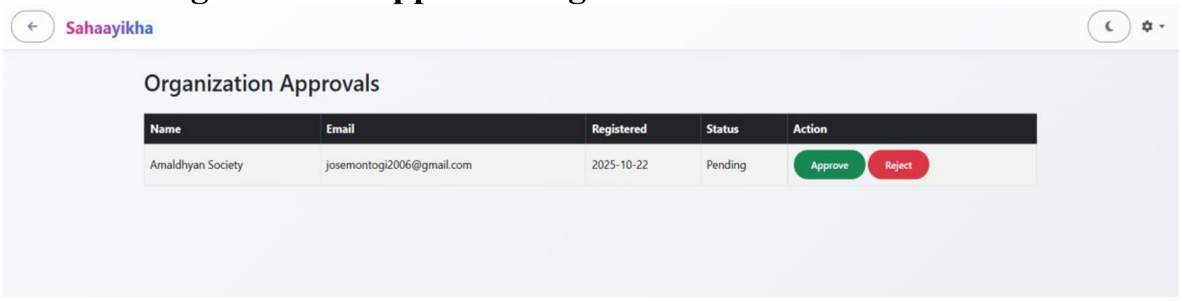
Pending Actions

- Organizations to Approve
- User Reports to Review

Community Engagement

- View User Feedback

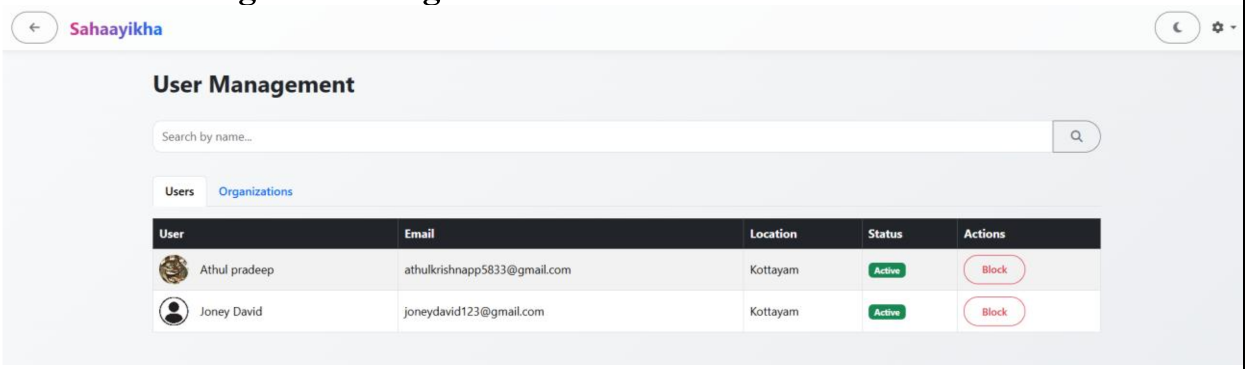
Admin - Organization Approval Page



Organization Approvals

Name	Email	Registered	Status	Action
Amaldhyan Society	josemontogi2006@gmail.com	2025-10-22	Pending	Approve Reject

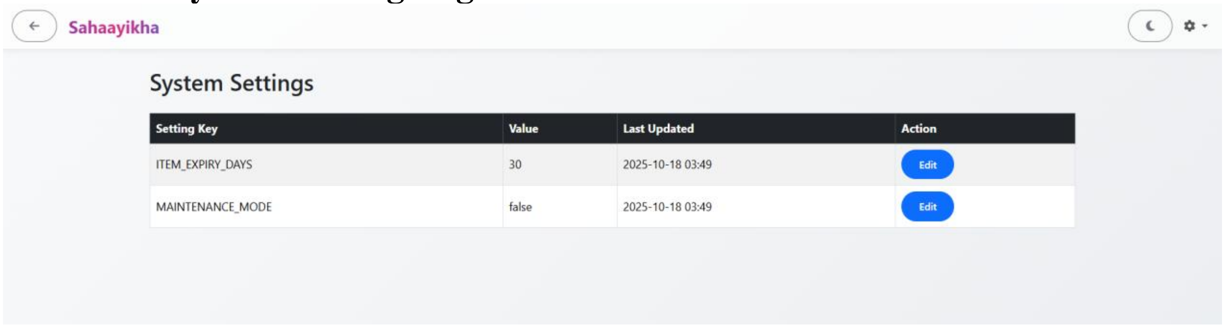
Admin - Manage Users Page



User Management

User	Email	Location	Status	Actions
Athul pradeep	athulkrishnapp5833@gmail.com	Kottayam	Active	Block
Joney David	joneydavid123@gmail.com	Kottayam	Active	Block

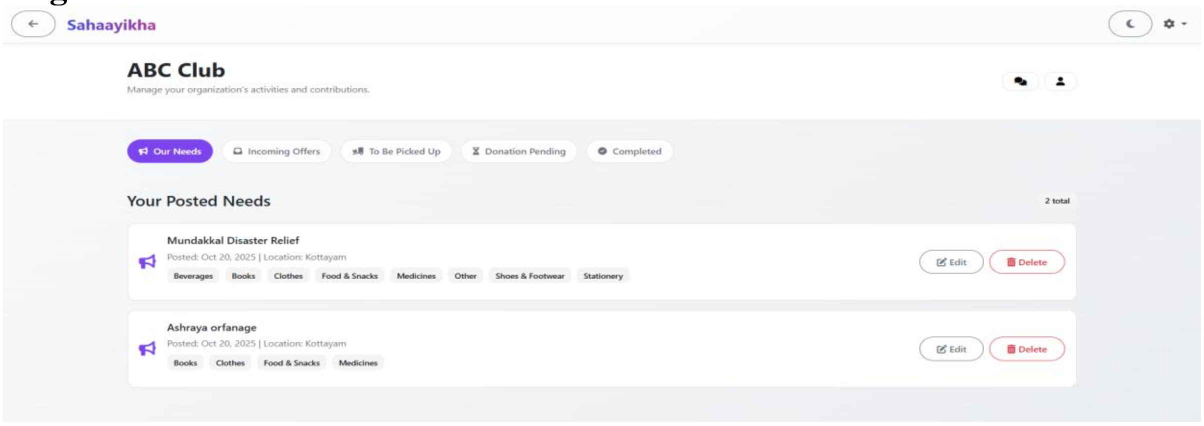
Admin – System Setting Page



System Settings

Setting Key	Value	Last Updated	Action
ITEM_EXPIRY_DAYS	30	2025-10-18 03:49	Edit
MAINTENANCE_MODE	false	2025-10-18 03:49	Edit

Organization Dashboard



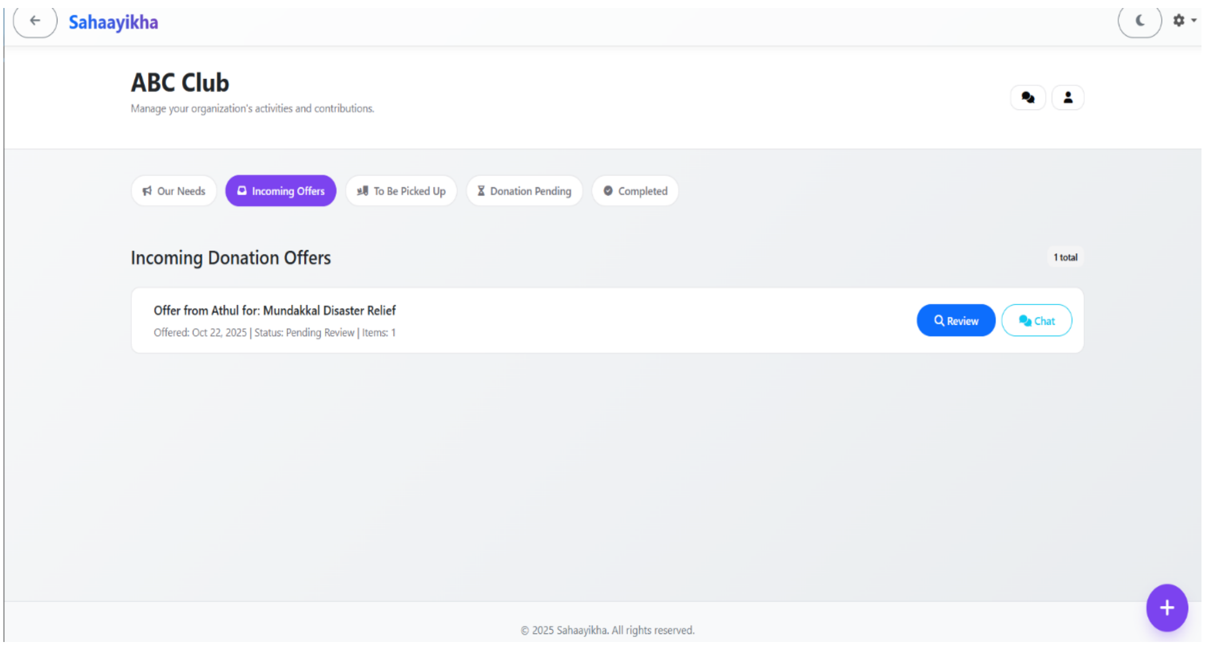
ABC Club

Manage your organization's activities and contributions.

Your Posted Needs 2 total

Mundakkal Disaster Relief
Posted: Oct 20, 2025 | Location: Kottayam
Beverages Books Clothes Food & Snacks Medicines Other Shoes & Footwear Stationery

Ashraya orfanage
Posted: Oct 20, 2025 | Location: Kottayam
Books Clothes Food & Snacks Medicines



ABC Club

Manage your organization's activities and contributions.

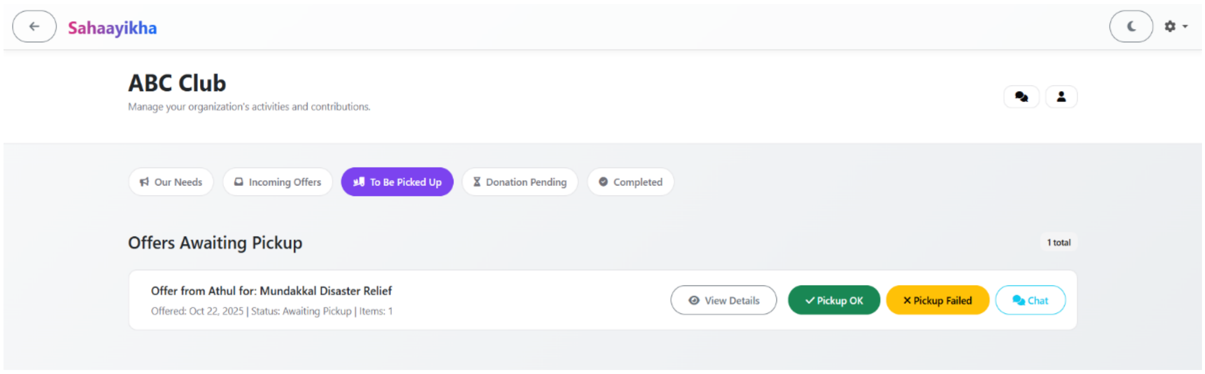
Incoming Donation Offers

1 total

Offer from Athul for: Mundakkal Disaster Relief

Offered: Oct 22, 2025 | Status: Pending Review | Items: 1

© 2025 Sahaayikha. All rights reserved.



ABC Club

Manage your organization's activities and contributions.

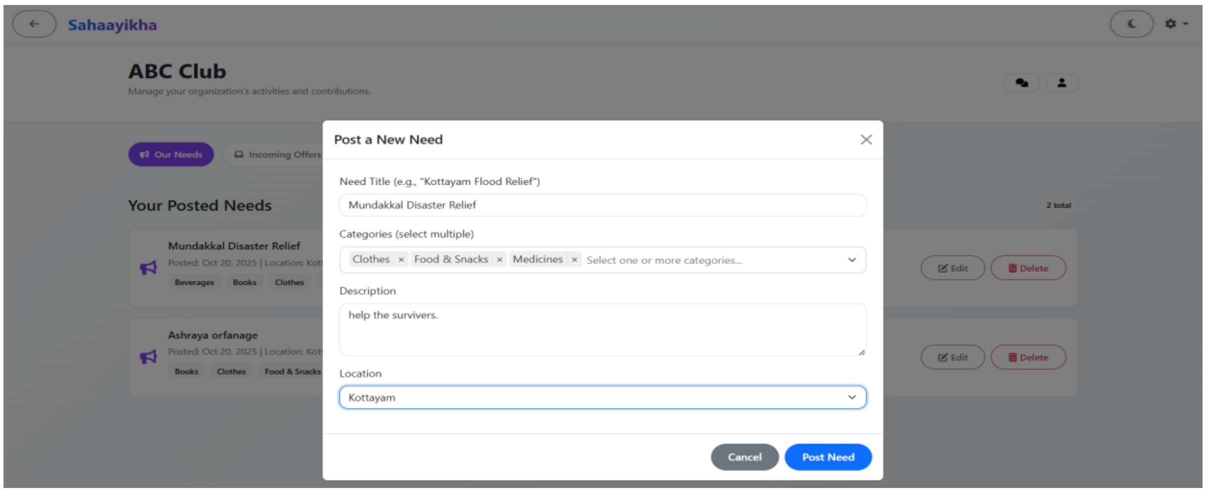
Offers Awaiting Pickup

1 total

Offer from Athul for: Mundakkal Disaster Relief

Offered: Oct 22, 2025 | Status: Awaiting Pickup | Items: 1

Organization - Post Disaster Need Page



ABC Club

Manage your organization's activities and contributions.

Your Posted Needs

2 total

Mundakkal Disaster Relief

Posted: Oct 20, 2025 | Location: Kot

Beverages Books Clothes

Ashraya orfanage

Posted: Oct 20, 2025 | Location: Kot

Books Clothes Food & Snacks

Post a New Need

Need Title (e.g., "Kottayam Flood Relief")

Mundakkal Disaster Relief

Categories (select multiple)

Clothes × Food & Snacks × Medicines × Select one or more categories...

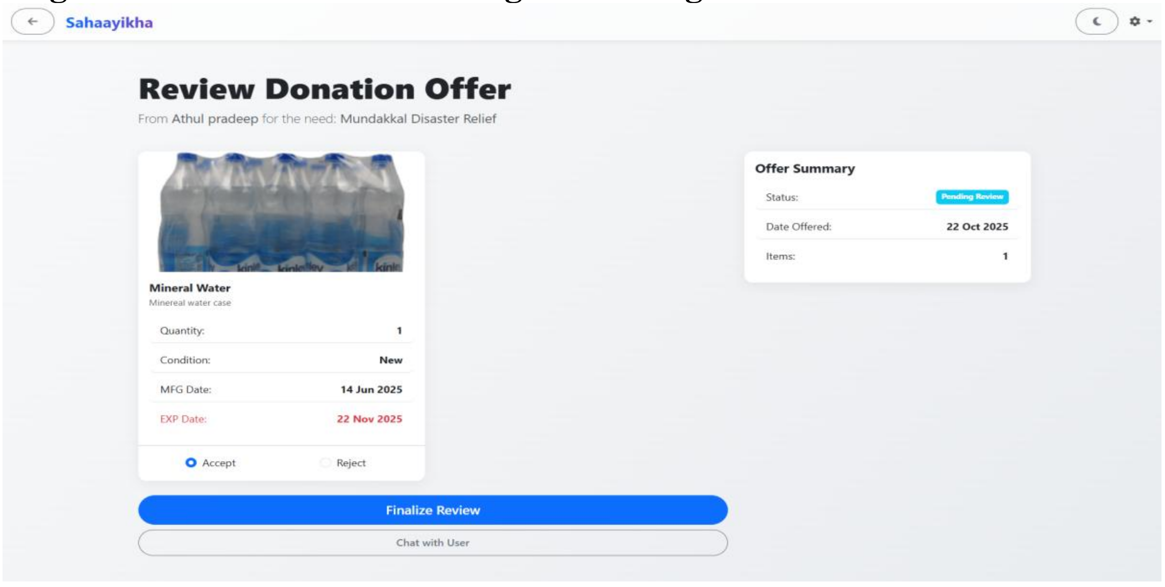
Description

help the survivers.

Location

Kottayam

Organization - Review Incoming Offers Page



Sahaayikha

Review Donation Offer

From Athul pradeep for the need: Mundakkal Disaster Relief

Mineral Water
Minereal water case

Quantity:	1
Condition:	New
MFG Date:	14 Jun 2025
EXP Date:	22 Nov 2025

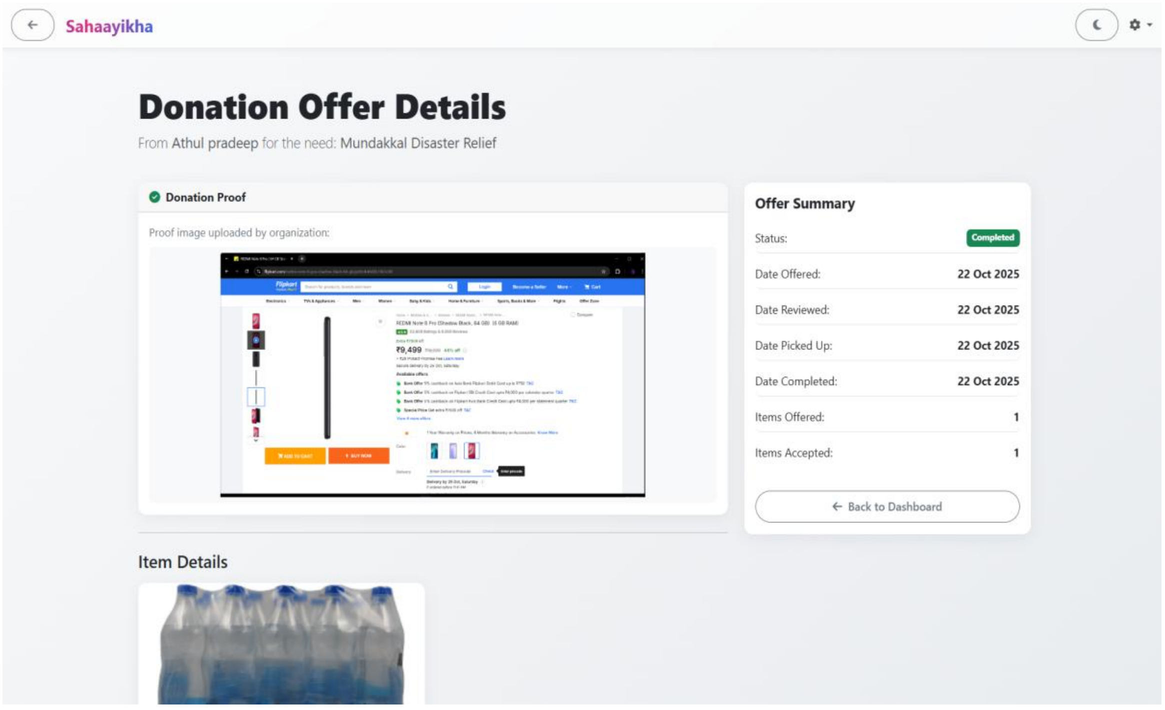
[✓ Accept] Reject

Finalize Review
Chat with User

Offer Summary

Status:	Pending Review
Date Offered:	22 Oct 2025
Items:	1

Completed Deals Page



Sahaayikha

Donation Offer Details

From Athul pradeep for the need: Mundakkal Disaster Relief

Donation Proof

Proof image uploaded by organization:

Offer Summary

Status:	Completed
Date Offered:	22 Oct 2025
Date Reviewed:	22 Oct 2025
Date Picked Up:	22 Oct 2025
Date Completed:	22 Oct 2025
Items Offered:	1
Items Accepted:	1

← Back to Dashboard

Item Details

APPENDICES B: Sample Code

1. Sample Python Code (app/routes.py - Flask Route)

```
from flask import render_template, flash, redirect, url_for, request, session

from app import app, db

from app.forms import (LoginForm, RegistrationForm, ItemForm, OrgRegistrationForm,

                  OrgLoginForm, AdminLoginForm, SearchForm, ...)

from app.models import (User, Item, Organization, Admin, ChatSession, ChatMessage,

                    TradeRequest, Bookmark, Notification, ...)

from flask_login import current_user, login_user, logout_user, login_required

from sqlalchemy import or_, and_, desc, func, not_

from sqlalchemy.orm import joinedload as orm_joinedload

from app.utils import geocode_location, haversine_distance, GEOCODE_DATA

from app.main import main

from app.decorators import role_required

import os

import secrets

import json

from datetime import datetime, timedelta


@main.route("/dashboard")

@login_required

@role_required("user")

def dashboard():

    """Displays the user dashboard with different views and integrated search/filters."""
```

```
view = request.args.get("view", "all")
form_data = request.args.copy()
if view == 'all' and 'location' not in form_data and current_user.location:
    form_data['location'] = current_user.location
form = SearchForm(form_data)

items, disaster_needs, my_offers = [], [], []
regular_chats, disaster_chats = [], []
unread_session_ids, has_unread_regular, has_unread_disaster = set(), False, False
has_unread_chats = False
items_json = "[]"
map_center_coords = None
map_radius_km = None
current_location_filter = None
active_search_term = request.args.get('search', '').strip()

# --- Query User's Chats ---
user_sessions_query = ChatSession.query.filter(
    or_(ChatSession.user_one_id == current_user.user_id, ChatSession.user_two_id ==
      current_user.user_id)
).options(
    orm_joinedload(ChatSession.user_one),
    orm_joinedload(ChatSession.user_two),
    orm_joinedload(ChatSession.participant_org),
    orm_joinedload(ChatSession.trade_item),
    orm_joinedload(ChatSession.disaster_need)
) # Eager load participants and subjects


# --- *** MODIFICATION START: Fetch latest message times and unread status *** ---
sessions_with_latest_message = []
unread_session_ids = set()
all_user_sessions = user_sessions_query.all() # Fetch all sessions first


if all_user_sessions:
    session_ids = [s.session_id for s in all_user_sessions]


    # Query for unread messages *not* sent by the current user
    unread_messages_query = ChatMessage.query.filter(
        ChatMessage.session_id.in_(session_ids),
        ChatMessage.is_read == False,
        not_(and_(ChatMessage.sender_id == current_user.user_id, ChatMessage.sender_type ==
      'user'))
    ).options(db.load_only(ChatMessage.session_id)) # Optimize query


    unread_session_ids = {msg.session_id for msg in unread_messages_query.all()}
    has_unread_chats = len(unread_session_ids) > 0


    # Query for the latest message timestamp in each session
    latest_messages_subquery = db.session.query(
        ChatMessage.session_id,
        func.max(ChatMessage.timestamp).label('latest_timestamp')
    ).filter(ChatMessage.session_id.in_(session_ids))\
     .group_by(ChatMessage.session_id)\
     .subquery()
    # Join sessions with their latest message timestamp
    sessions_with_time = db.session.query(
        ChatSession, latest_messages_subquery.c.latest_timestamp
    ).outerjoin(latest_messages_subquery,                    ChatSession.session_id                    ==
      latest_messages_subquery.c.session_id)\
     .filter(ChatSession.session_id.in_(session_ids))\
     .all() # Returns list of tuples (ChatSession, latest_timestamp | None)

    # Create list of dictionaries for easier sorting
    for session, latest_timestamp in sessions_with_time:
        sessions_with_latest_message.append({
            'session': session,
            'latest_activity': latest_timestamp or session.started_at, # Fallback to start time
            'is_unread': session.session_id in unread_session_ids
        })

    # Sort: Unread first, then by latest activity timestamp descending
    sessions_with_latest_message.sort(key=lambda  x:  (not  x['is_unread'],  x['latest_activity']),
      reverse=True)
# --- Fetch Data Based on View/Filter ---
if view == "chats":
    # Use the sorted list
    all_sorted_sessions = [s_data['session'] for s_data in sessions_with_latest_message]
    regular_chats = [s for s in all_sorted_sessions if s.trade_item_id is not None]
    disaster_chats = [s for s in all_sorted_sessions if s.disaster_need_id is not None or
      s.donation_offer_id is not None]
    # Unread flags are now based on the direct query result
    has_unread_regular = any(s.session_id in unread_session_ids for s in regular_chats)
    has_unread_disaster = any(s.session_id in unread_session_ids for s in disaster_chats)


elif view == "mine":
    # ... (rest of the 'mine' view logic remains the same) ...
    query = Item.query.filter_by(user_id=current_user.user_id, status='Active')
    filter_type = request.args.get('filter')
    if filter_type in ["Trade", "Share"]:
        query = query.filter_by(type=filter_type)
    mine_categories = form.categories.data
    mine_urgency = form.urgency.data
    mine_condition = form.condition.data
    mine_sort_by = form.sort_by.data or 'newest'
    if mine_categories: query = query.filter(Item.category.in_(mine_categories))
    if mine_urgency: query = query.filter_by(urgency_level=mine_urgency)
    if mine_condition: query = query.filter_by(condition=mine_condition)
    if mine_sort_by == 'oldest':
        items = query.order_by(Item.created_at.asc()).all()
    else:
        items = query.order_by(Item.created_at.desc()).all()
```

```
elif view == "bookmarks":
    items = Item.query.join(Bookmark, Item.item_id == Bookmark.item_id)\
                    .filter(Bookmark.user_id == current_user.user_id)\
                    .order_by(Bookmark.saved_at.desc()).all()
elif view == "donations":
    my_offers                                                          =
      DonationOffer.query.filter_by(user_id=current_user.user_id).order_by(DonationOffer.cre
      ated_at.desc()).all()

else: # 'all' view
    # ... (rest of the 'all' view logic remains the same) ...
    query = Item.query.filter(Item.status == "Active", Item.user_id != current_user.user_id)
    current_location_filter = form.location.data
    search = active_search_term
    if search:
        search_term_like = f"%{search}%"
        query            =            query.filter(or_(Item.title.ilike(search_term_like),
      Item.description.ilike(search_term_like)))
    location_filter = form.location.data
    radius_str = form.radius.data
    categories = form.categories.data
    urgency = form.urgency.data
    condition = form.condition.data
    sort_by = form.sort_by.data or 'newest'
    active_categories = categories if categories else []
    if not active_categories and request.args.get('categories'):
         active_categories = [request.args.get('categories')]
    if active_categories:
        query = query.filter(Item.category.in_(active_categories))
    if urgency: query = query.filter_by(urgency_level=urgency)
    if condition: query = query.filter_by(condition=condition)
    filter_type = request.args.get('filter')
    if filter_type == 'Disaster':
        disaster_needs                                                    =
      DisasterNeed.query.filter_by(status='Active').order_by(DisasterNeed.posted_at.desc()).all
      ()
        items = []
    else:
        if filter_type in ["Trade", "Share"]:
            query = query.filter_by(type=filter_type)
        all_items = query.order_by(Item.created_at.desc()).all()
        results = []
        user_lat, user_lon = None, None
        base_location_name = None
        items_with_distance = []
        if location_filter:
            base_location_name = location_filter
            map_center_coords = geocode_location(location_filter)
        else: map_center_coords = None
        if radius_str:
            try: map_radius_km = float(radius_str)
          except ValueError: map_radius_km = None
        else: map_radius_km = None
        radius_active = map_radius_km is not None
        sort_by_distance_active = sort_by == 'distance'
        center_coords_valid = map_center_coords and map_center_coords[0] is not None
        if (radius_active or sort_by_distance_active) and center_coords_valid:
            user_lat, user_lon = map_center_coords
            radius_km_filter = map_radius_km if radius_active else float('inf')
            for item in all_items:
                item_lat, item_lon = item.latitude, item.longitude
                if item_lat is None or item_lon is None:
                    coords = geocode_location(item.location)
                    if coords and coords[0] is not None:
                        item_lat, item_lon = coords
                if item_lat is not None and item_lon is not None:
                    dist = haversine_distance(user_lat, user_lon, item_lat, item_lon)
                    if dist <= radius_km_filter: items_with_distance.append({'item': item, 'distance':
      dist})
                elif not radius_active:
                    items_with_distance.append({'item': item, 'distance': float('inf')})
            if sort_by_distance_active: items_with_distance.sort(key=lambda x: x['distance'])
            results = [item_dist['item'] for item_dist in items_with_distance]
        else:
            results = all_items
            if radius_active and not center_coords_valid: flash(f"Could not find coordinates for
        '{base_location_name or 'selected location'}'. Cannot filter by radius.", "warning")
            if sort_by_distance_active and not center_coords_valid: flash(f"Could not find
        coordinates for '{base_location_name or 'selected location'}'. Cannot sort by distance.",
        "warning"); sort_by = 'newest'
         if sort_by == 'oldest' and (not sort_by_distance_active or not center_coords_valid):
            results.sort(key=lambda item: item.created_at)
         elif sort_by == 'newest' and (not sort_by_distance_active or not center_coords_valid):
            results.sort(key=lambda item: item.created_at, reverse=True)
         items = results
         items_for_map = [
            {"item_id": item.item_id, "title": item.title, "latitude": item.latitude, "longitude":
        item.longitude}
            for item in items if item.latitude and item.longitude
         ]
         items_json = json.dumps(items_for_map)
return render_template(
   "dashboard/user_dashboard.html",
   items=items,
   view=view,
   disaster_needs=disaster_needs,
   my_offers=my_offers,
   regular_chats=regular_chats,
   disaster_chats=disaster_chats,
   unread_session_ids=unread_session_ids, # Pass the set of IDs
   has_unread_regular=has_unread_regular,
   has_unread_disaster=has_unread_disaster,
     has_unread_chats=has_unread_chats,
          form=form,
          items_json=items_json,
          map_center_coords=map_center_coords,
          map_radius_km=map_radius_km,
          all_locations_coords=json.dumps(GEOCODE_DATA),
          current_location_filter=current_location_filter,
          active_search_term=active_search_term
        )
```

2. Sample HTML Code (app/templates/auth/user_register.html - Jinja2 Template)

```
{# templates/auth/user_register.html #}
{% extends "base.html" %}
{% block content %}
<div class="container my-5">
  <div class="row justify-content-center">
    <div class="col-lg-8">
      <div class="card shadow p-4">
        <h3 class="mb-4 text-center">Create User Account</h3>

        {% with messages = get_flashed_messages(with_categories=true) %}
          {% if messages %}
            <div class="mb-3">
              {% for category, message in messages %}
                <div class="alert alert-{{ category }}">{{ message }}</div>
              {% endfor %}
                </div>
    {% endif %}
  {% endwith %}

  <form method="POST" enctype="multipart/form-data" novalidate>
    {{ form.hidden_tag() }}

    {% set form_field = form.profile_picture %}
    {% set placeholder_img = 'images/users_placeholder.png' %}
    {% include 'auth/_profile_uploader.html' %}

    <div class="row">
      <div class="col-md-6 mb-3">
        {{ form.first_name.label(class="form-label") }}
        {{ form.first_name(class="form-control") }}
      </div>
      <div class="col-md-6 mb-3">
        {{ form.last_name.label(class="form-label") }}
        {{ form.last_name(class="form-control") }}
      </div>
    </div>
    <div class="mb-3">
      {{ form.email.label(class="form-label") }}
      {{ form.email(class="form-control") }}
    </div>
    <div class="row">
      <div class="col-md-6 mb-3">
        {{ form.phone.label(class="form-label") }}

{{ form.phone(class="form-control") }}
```

```
          </div>
          <div class="col-md-6 mb-3">
            {{ form.location.label(class="form-label") }}
            {{ form.location(class="form-select") }}
          </div>
        </div>
        <div class="row">
          <div class="col-md-6 mb-3">
            {{ form.password.label(class="form-label") }}
            {{ form.password(class="form-control") }}
          </div>
          <div class="col-md-6 mb-3">
            {{ form.confirm_password.label(class="form-label") }}
            {{ form.confirm_password(class="form-control") }}
          </div>
        </div>
        <div class="d-grid">
          {{ form.submit(class="btn btn-primary btn-lg") }}
        </div>
        <p class="mt-3 text-center">
          Already have an account? <a href="{{ url_for('main.login') }}">Login</a>
        </p>
      </form>
    </div>
   </div>
 </div>

</div>
{% endblock %}
```

APPENDICES C: Hardware and Software Requirements

- Hardware Requirements

Processor	:	Intel Core i3
Random Access Memory	:	4GB or above
Hard Disk/SSD	:	240GB
Monitor	:	Color Monitor
Keyboard	:	Standard
Video	:	800X600 256 colors

- Software Requirements

Operating System	:	Windows 11 or Higher
Front End	:	HTML, CSS, JavaScript, Bootstrap 5
Environment	:	Python 3.x, Flask
Database	:	SQLite
Key Libraries	:	Flask-SQLAlchemy, Flask-Migrate (Alembic), Flask-Login, Flask-WTF, Firebase-Admin (for notifications)
Operating System	:	Windows 11
Documentation Tool	:	MS Word

APPENDICES D: Bibliography

Book References:

- Taming Python by Programming by Dr. Jeeva Jose

Publications:

1. **Grinberg, M. (2018).** Flask Web Development: Developing Web Applications with Python. O'Reilly Media**.**
2. **Aggarwal, S. (2015).** Flask Framework Cookbook. Packt Publishing.
3. Spade, D. (2020). Mutual Aid: Building Solidarity During This Crisis (and the Next). Verso Books. | Google Scholar
4. **Sundararajan, A. (2016).** The Sharing Economy: The End of Employment and the Rise of Crowd-Based Capitalism. MIT Press. | Google Schola

5. **Palen, L., & Liu, S. B. (2007).** Citizen communications in crisis: anticipating a future of ICT-supported public participation. Proceedings of the SIGCHI conference on Human factors in computing systems. | Google Scholar
6. **Stallings, R., & Ragsdale, J. (2002).** A unified platform for mutual aid and emergency management. Disaster Prevention and Management, 11(5), 364-374. | Google Scholar

Website References:

- Retrieved from www.stackoverflow.com
- Retrieved from www.w3schools.com
- Retrieved from www.gihub.com
- Retrieved from www.python.org
- Retrieved from https://flask.palletsprojects.com/
- Retrieved from www.tutorialspoint.com