Cover Page :

Document Title: Project Requirement Documentation

Project Title: Crowdsourced Disaster Relief Platform

Submitted Date: 3/8/2025

Due: 3/9/2025

Members: Casey, Kevin, Andy, Sawyer

University of Texas at Dallas, CS3354.004 , R. Feng

Document Purpose:

A guide focusing on the functional and non-functional part of the project. Helps whoever is reading this document to gain a clear understanding of the project and its underlayers. This document makes a clear understanding of the project scope, objectives, and the plan taken among all stakeholders, including developers, designers, testers, and clients.

Individual Contributions Breakdown :

Responsibility Matrix:

- Responsible
- Accountable
- Consultanted
- Informed

Activity	Casey	Kevin	Andy	Sawyer
Group Meetings	R	I	I	
Programming	R	R	R	
Section 1	R	C	I	
Section 2	A	C	R	
Section 3	A	R	I	
Section 4	A	R	I	
Section 5	R	C	I	
Section 6	A	C	R	
Section 7	R	C	I	

Responsibility Allocations Chart:

Tasks	Casey	Kevin	Andy	Sawyer
Management	✅	✅	✅	
Cover Page	✅			
Contributions Page	✅			
Section 1	✅			
Section 2			✅	
Section 3		✅		
Section 4		✅		
Section 5	✅			
Section 6			✅	
Section 7	✅			

Table of Contents :

Work Assignment:

Sub-Teams:

1. Kevin and Casey
2. Andy and Sawyer

Individual Competences:

- Casey: Cover Page, Section 1, 5 and 7
- Kevin: Sections 3 and 4.
- Andy: Sections 2 and 6
- Sawyer:

- Part 1 :

Section 1 : Customer Problem Statement

a. Problem Statement

Disasters such as earthquakes, floods, and hurricanes require a rapid and organized response to minimize damage and provide aid to those affected. However, coordinating relief efforts across various organizations, including government agencies, NGOs, local volunteers, and impacted individuals, presents significant challenges. These challenges include overlapping efforts, misallocated resources, ineffective communication, and slow response times. Often, the lack of efficient real-time communication is a primary reason for delays and inefficiencies in relief operations.

Our proposed Crowdsourced Disaster Relief Platform aims to streamline disaster response by providing a real-time communication hub that connects victims, volunteers, first responders, and relief organizations. This platform will facilitate instant updates on resource availability, areas in need of assistance, and overall disaster status, ensuring a more coordinated and effective response

Communities affected by disasters often struggle to access essential supplies and assistance. Victims may be ill informed of where to find help, while relief organizations may direct resources to the wrong areas due to incomplete or mismatching information. As a result, aid may not reach those who need it most, leading to inefficiencies and delays in disaster response. This also causes a waste of valuable donated resources and money. And most importantly of all, response time.

The main issue lies in the lack of real-time, accurate, and coordinated communication between relief efforts and affected individuals, leading to underutilized resources. Our centralized platform will serve as a communication and coordination hub that connects critical information, enabling stakeholders to respond effectively by directing aid where it is needed most.

The Need for a Centralized Solution

To address these challenges, a centralized, crowdsourced disaster relief platform is needed. This platform will serve as a communication hub where disaster victims can report their immediate needs, volunteers can offer assistance, and organizations can coordinate their efforts in real time. By leveraging technology to bridge the gap between those in need and those who can help, this system will create a more responsive and efficient disaster relief network.

The platform will enable real-time reporting, allowing affected individuals to share critical information about their location, needs, and safety status. Relief organizations and volunteers will have access to a dynamic map of the disaster situation, allowing them to direct aid to areas that need it most. By providing clear and updated information, the platform will improve decision-making and help ensure that resources such as food, water, medical supplies, and shelter are allocated effectively.

Stakeholders and Impact

A successful disaster relief platform must serve the diverse needs of multiple stakeholders, including:

- **Disaster Victims:** Individuals in affected areas who require immediate assistance, such as food, shelter, and medical aid.
- **Volunteers:** Individuals who are willing to provide support in rescue efforts, logistics, and distribution of resources.
- **NGOs & Government Agencies:** Organizations responsible for structured disaster response and management of relief efforts.
- **Donors:** Individuals and companies willing to contribute financial aid and material support.
- **First Responders:** Emergency personnel, including law enforcement and medical teams, who require up-to-date information to execute rescue operations effectively.

Addressing Key Challenges

For the platform to be effective, it must address several key challenges:

1. **Reliable Functionality in Low-Connectivity Areas:**
    - The system must be designed to operate in areas with poor network coverage, utilizing offline functionality and optimized data transmission methods.
2. **Scalability and System Resilience:**
    - The platform must support a large number of simultaneous users and ensure uptime during peak disaster situations through cloud computing and load balancing.
3. **Data Security and User Verification:**
    - Secure handling of sensitive data is essential to protect disaster victims and donors.
    - Implementing role-based access control (RBAC) will help prevent unauthorized access to critical resources.
4. **Efficient Resource Allocation:**
    - The platform must provide accurate, real-time information to match available resources with the most urgent needs.
5. **Trust and Safety:**
    - Verifying volunteers and NGOs will prevent misuse of the system and ensure that only authorized individuals have access to critical relief operations.

Transforming Disaster Response

By focusing on communication, improving coordination, and maximizing the efficient use of resources, this platform has the potential to help the crisis in disaster relief efforts. The ability to quickly assess real-time information and connect those in need with those who can help will not only reduce response times but also save lives. Through crowdsourcing and technology, disaster response can become more effective, transparent, and resilient in the face of crises.

b. Decompositions into Sub-Problems

Real-Time Communication and Reporting

- Ensure that disaster victims can report their needs quickly and accurately.
- Provide a way for volunteers, NGOs, and first responders to access and share real-time updates.
- Maintain communication channels even in areas with poor connectivity.

Resource Allocation and Distribution

- Develop a system to match available resources (food, water, medical aid) with the most urgent needs.
- Optimize resource distribution based on proximity and priority.
- Prevent duplicate or misallocated relief efforts by different organizations.

User Identity and Role Management

- Implement a secure and safe registration system to verify victims, volunteers, and NGOs.
- Prevent unauthorized users from accessing sensitive information or manipulating resources.
- Assign role-based access controls to ensure different stakeholders have the appropriate permissions.

Scalability and System Reliability

- Ensure the platform can handle a large number of users during peak disaster situations.
- Implement cloud computing and load balancing for system time and performance.
- Design offline functionality to allow continued use in areas with unstable networks.

Data Security and Privacy

- Securely handle sensitive user data, including personal and financial information of donors.

- Implement encryption and authentication features to protect against unauthorized access.
- Address potential risks related to data breaches and fraudulent activities.

Trust and Safety Measures

- Develop features to verify the legitimacy of NGOs and volunteers before they access critical resources.
- Implement safeguards to prevent misinformation or fraudulent activities within the platform.
- Ensure accountability and transparency in disaster response efforts.

c. Glossary of Terms

1. **Disaster Victims** – Individuals directly affected by a disaster who require assistance such as shelter, food, or medical aid.
2. **Non-Governmental Organizations (NGOs)** – Independent organizations that provide disaster relief, including food distribution, medical aid, and shelter assistance.
3. **Relief Organizations** – Entities, including NGOs and government agencies, that coordinate disaster response and distribute aid to affected areas.
4. **Real-Time Reporting** – A system feature allowing users to provide and access live updates on disaster situations, including resource availability and urgent needs.
5. **Resource Matching** – The process of connecting available resources (e.g., food, medical supplies, shelters) with affected individuals or areas based on urgency and proximity.
6. **Role-Based Access Control (RBAC)** – A security mechanism that assigns different levels of access to users based on their roles (e.g., victims, volunteers, NGOs).
7. **Crowdsourced Data** – Information collected from a large number of users, including victims and volunteers, to create a complete overview of the disaster situation.
8. **Offline Functionality** – A feature enabling users to access critical information and report needs even when internet connectivity is limited or unavailable.
9. **Load Balancing** – A method to distribute user requests across multiple servers to ensure system stability and prevent overload during peak disaster situations.
10. **Two-Factor Authentication (2FA)** – A security feature requiring users to verify their identity using two independent methods to enhance data protection.
11. **Misinformation Control** – Measures taken to prevent the spread of false or misleading information within the platform to ensure accurate disaster response.
12. **Donors** – Individuals or organizations providing financial or material aid to support disaster relief efforts.

13. **System Uptime** – The percentage of time the platform remains operational and accessible to users during disaster events.
14. **Emergency Alerts** – Notifications sent to users to provide critical updates, warnings, or requests for assistance during a disaster.

Section 2 : Goals, Requirements, and Analysis

2.1 Business Goals

Our platform aims to improve disaster response through real time communication, optimized resource allocation, and enhanced coordination among disaster victims, volunteers, and NGOs. THe key business goals are:

- **Faster Emergency Response**
  - Reduce response times for victims seeking aid
- **Efficient Resource Allocation**
  - Ensure that resources reach areas of need.
- **Enhanced Coordination**
  - Facilitate effective communication between relief organizations
- **Scalability & Reliability**
  - Support high numbers of users during critical times
- **Security & Data Protection**
  - Ensure secure handling of sensitive user and financial data
- **User Accessibility**
  - Provide a user friendly and mobile optimized interface

2.2 Enumerated Functional Requirements

Requirement Label	Priority	Requirement Description
REQ-1	High	The system should allow disaster victims to submit aid requests with geolocation data
REQ-2	High	The system should allow volunteers to register, provide skill details, and update their availability
REQ-3	High	The system should provide AI powered matching between aid requests and available volunteers

REQ-4	High	The system should provide a live map displaying aid requests and available volunteers
REQ-5	Medium	The system should allow NGOs to manage and update their inventory
REQ-6	Medium	The system should send real time notifications to users about critical updates
REQ-7	Medium	The system should implement role based access control for users (victims, volunteers, NGOs, donors)
REQ-8	Medium	The system should enable users to report and verify misinformation to prevent false reports
REQ-9	Medium	The system should provide a donation management module to track financial and supply contributions
REQ-10	Low	The system should provide multilingual support

2.3 Enumerated Non-Functional Requirements

Requirement Label	Priority	Requirement Description
NFR-1	High	**Scalability**: The system must support high volume of simultaneous users
NFR-2	High	**Security:** User authentication must include two factor authentication (2FA) Sensitive user information must be encrypted
NFR-3	High	**Reliability:** The system should maintain 99.9% uptime
NFR-4	Medium	**Performance:** The AI matching system should process requests in under 2 seconds
NFR-5	Medium	**Offline Support:** Key features must be accessible in low connectivity regions
NFR-6	Low	**User Experience:** The system should be accessible on both web and mobile devices and should be responsive on all devices

2.4 User Interface Requirements

- Simplicity & Useability
    - Minimal design to ensure that users under stress can quickly navigate the system
    - Accessible on both mobile and web
- Mobile First Design
    - Interface must be optimized for mobile users as victims will likely only have access to mobile devices
- Readability
    - Important information must be visually distinct (alerts, active aid requests, etc…)

Part 2 :

Section 3 : Use Cases

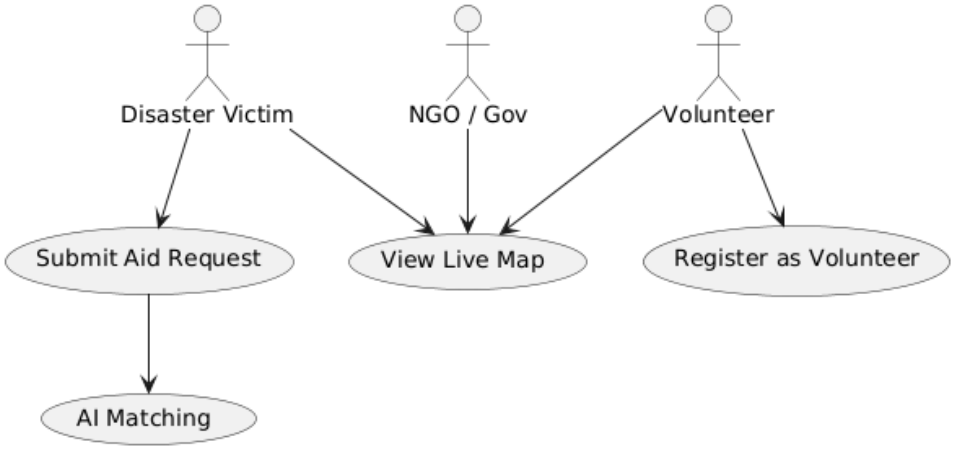
a. Stakeholders
    i. **Disaster Victims:** Need immediate aid.
    ii. **Volunteers:** Provide assistance based on skills and proximity.
    iii. **NGOs and Government Agencies:** Coordinate resource distribution and manage relief operations.
    iv. **Donors:** Provide financial or material support.

b. Actors and Goals
    i. **Victim Actor:** Wants to report an emergency and receive immediate help.
    ii. **Volunteer Actor:** Seeks to view and accept aid requests based on their skills.
    iii. **NGO Actor:** Requires real-time updates and data to manage resources efficiently.

c. Use Cases :
    1. Casual Description
        a. **Case 1:** Submit Aid Request: A disaster victim submits an aid request specifying type (medical, food, shelter) and location.
        b. **Case 2:** Register as Volunteer: A volunteer registers with personal details, skills, and current location.

c. **Case 3:** AI Matching: The system automatically matches an incoming aid request with the most suitable volunteers.
d. **Case 4:** View Live Map: All users can view a map that displays current aid requests and volunteer locations.

2. Use Case Diagram

a. 

3. Traceability Matrix

a.

Requirement Label	Priority	Use Case Addressed
REQ-1	High	Submit Aid Request (initiated by Disaster Victim)
REQ-2	High	Register as Volunteer (initiated by Volunteer)
REQ-3	High	AI Matching (system processes aid requests)
REQ-4	Medium	View Live Map (accessible by all actors)

b.

4. Fully-Dressed Description

a. **Title:** AI-Powered Volunteer Matching
b. **Primary Actor:** System (on behalf of victims and volunteers)

c. **Goal:** To automatically match an aid request with the best available volunteers based on skills and location.

d. **Preconditions:**

• A valid aid request exists in the system.

• At least three volunteers are registered and available.

e. **Postconditions:**

• The top three matching volunteers are returned and notified.

f. **Main Flow:**

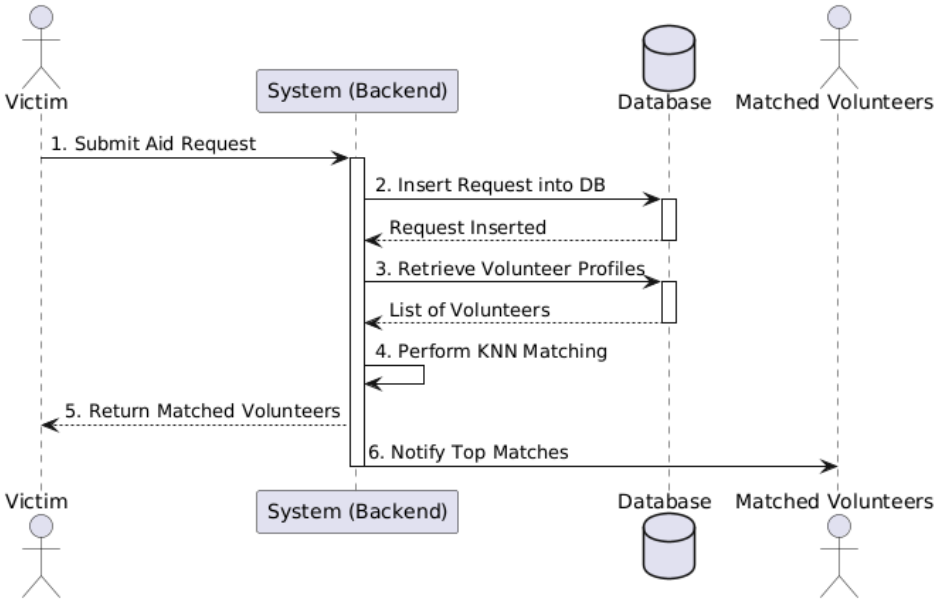
1. The system receives an aid request (with request ID).
2. The system queries the database for all registered volunteers.
3. Volunteer data is transformed into a numerical vector (using hash functions on skills and location).
4. The system transforms the aid request data similarly.
5. The K-Nearest Neighbors algorithm identifies the closest matches.
6. The system returns the top three matches and sends notifications.

g. **Alternative Flows:**

• If fewer than three volunteers are available, the system returns all available volunteers and a message indicating limited matches.

d. System Sequence Diagrams (Example: AI Matching Use Case)

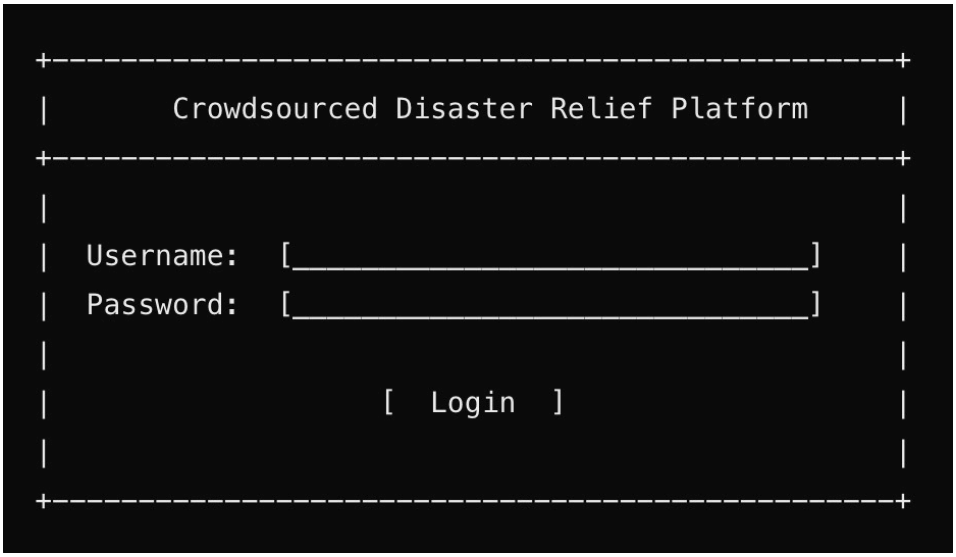
i.



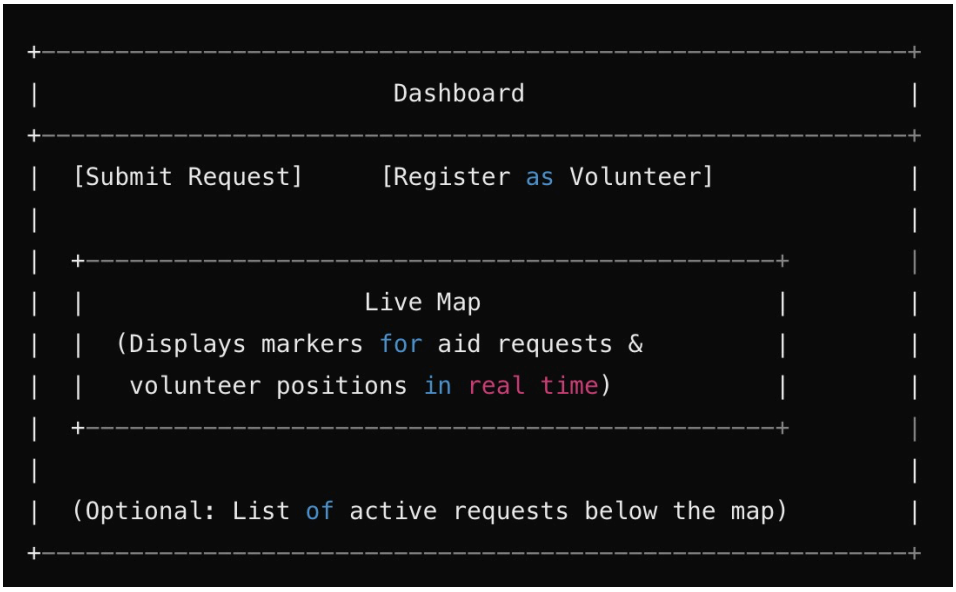
Section 4 : User Interface Specification

A. Preliminary Design

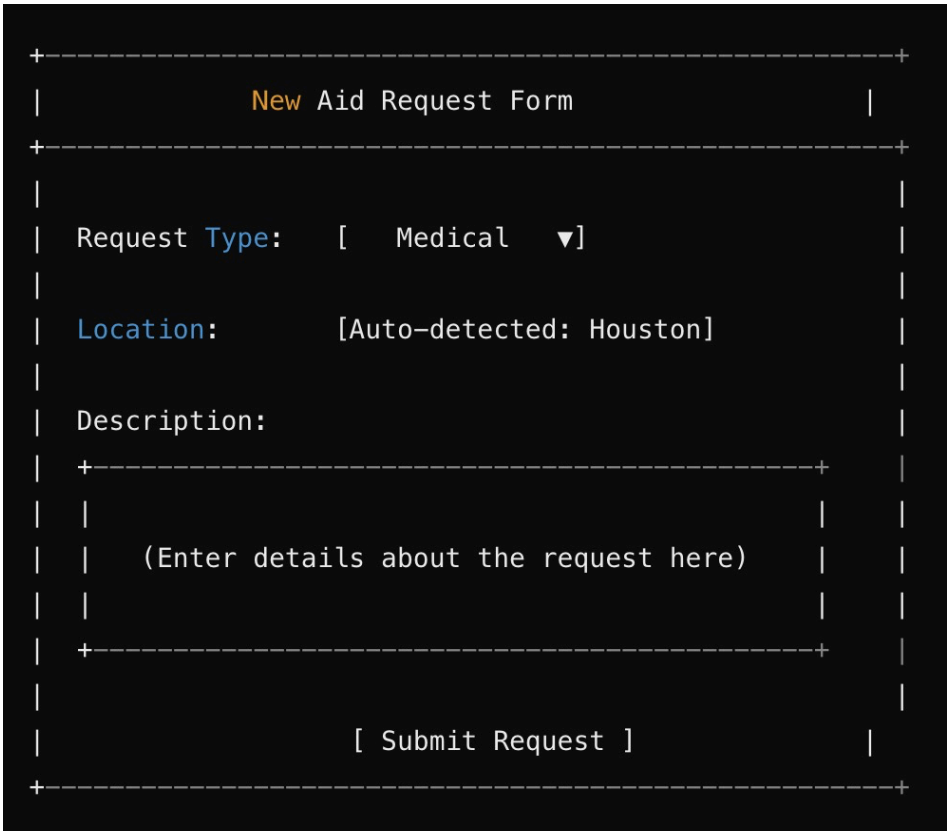
a. For the AI Matching feature (accessible via mobile or web):
b. Login Screen: Fields for user credentials; a “Login” button.
c. Dashboard:
    i. A “Submit Request” button for victims.
    ii. A “Register as Volunteer” button for volunteers.
    iii. A live map area showing current aid requests and volunteer positions.
d. Request Submission Screen:
    i. Input fields for request type and location (auto-geotagged if possible).
    ii. A “Submit” button that triggers the matching algorithm.
e. Volunteer Match Results Screen:
    i. A list displaying volunteer names, skills, and distance from the request location.
    ii. Option to view detailed profiles or accept a match.
f. (Include wireframes or sketches as images if available.)
    i. Login Screen:



ii. Dashboard:



iii. Request Submission Screen:



iv. Volunteer Match Results Screen:

ID	Name	Skills	Location	Distance
1	John Doe	Medical, Rescue	Houston	1.2 mi
2	Jane Smith	Food Logistics, Safety	Austin	2.5 mi
3	Mike Brown	Shelter Management, Aid	Dallas	3.0 mi

B. User Effort Estimation

a. **Login:** 2–3 keystrokes/clicks.
b. **Submit Aid Request:** Approximately 5–6 interactions (typing details, confirming geolocation, clicking “Submit”).
c. **Volunteer Registration:** Around 8–10 interactions including data entry and confirmation.
d. **Viewing Matches:** 1–2 clicks to refresh or view details from the list.

Part 3 :

Section 5 : System Architecture

a. Identifying Subsystems:

- **Frontend Subsystem:**

  Developed in Flutter (mobile) and/or React (web).

  Handles user interactions, displays data, and sends requests to the backend.

- **Backend API Subsystem:**

  Built with FastAPI, it acts as the core processing unit.

  Handles user authentication, processes requests, and manages business logic.

  Connects to both the database and the AI Matching subsystem.

- **Database Subsystem:**

Uses PostgreSQL for persistent storage.

Stores user data, aid requests, volunteer registrations, and matching results.

- **AI Matching Subsystem:**

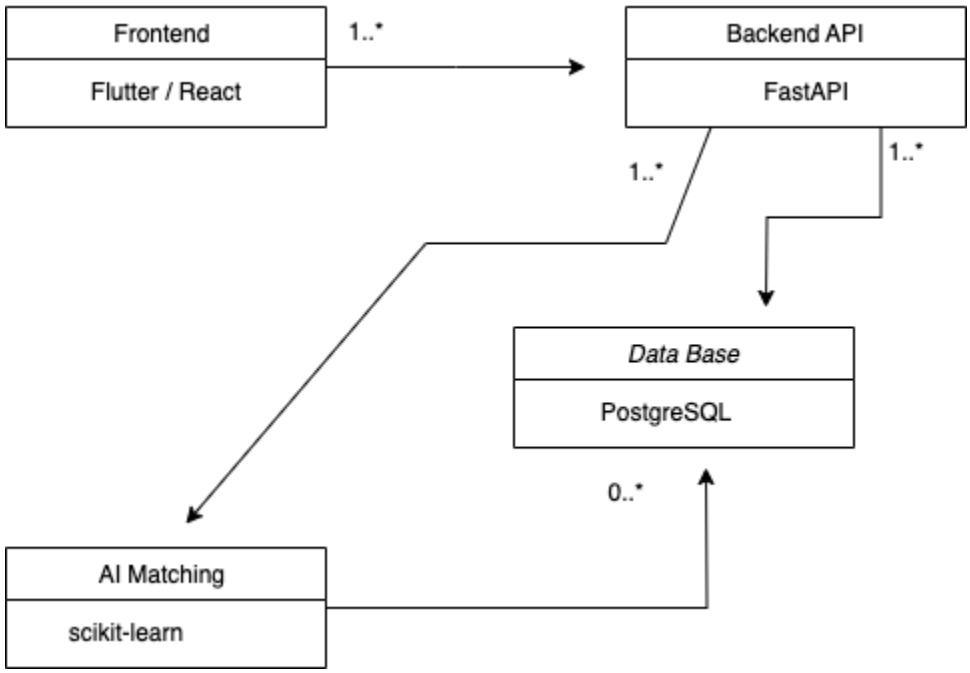
Integrated within the backend, utilizing scikit-learn for KNN-based matching.

Processes data to find the best volunteer-user matches.

UML Package Diagram:

Diagram Explanation:

- The Frontend Subsystem communicates with the Backend API via HTTP requests.
- The Backend API Subsystem interacts with the Database Subsystem using SQLAlchemy ORM (JDBC).
- The AI Matching Subsystem receives data from the Backend API, processes matches, and returns results.



b. Mapping Subsystems to Hardware

Distributed Architecture:

Subsystem	Hardware

Frontend	Runs on user devices (smartphones, tablets, or PCs).
Backend API	Hosted on a server (cloud-based or on-premise).
Database	Runs on the same server as the backend or a separate database server for scalability.
AI Matching	Integrated into the backend server. If scaling is required, it can be used separately.

c. Connections and Network Protocols

The system runs across multiple devices, requiring network communication.

- **Frontend ↔ Backend API:**
  - Protocol: HTTP/HTTPS
  - Reason: Standard, secure communication for RESTful API requests.
- **Backend API ↔ Database:**
  - Protocol: JDBC/ORM (SQLAlchemy)
  - Reason: Efficient and structured interaction with PostgreSQL.
- **Backend API ↔ AI Matching:**
  - Protocol: Internal function calls (as it's integrated into the backend).
- **Real-Time Communication:**
  - Protocol: WebSockets for push notifications.

d. Global Control Flow

Execution Order:

- The system follows an event-driven architecture.
- It waits for user actions (e.g., submitting a request, registering as a volunteer) and processes them asynchronously.
- Users can perform actions in any order, as the system does not enforce a strict sequence.

Time Dependency:

- The system is not a real-time system.
- Actions are processed as they arrive, with no strict timing constraints.
- Notifications may be sent based on triggers (e.g., successful matching).

e. Hardware Requirements

Server Requirements

The backend and database servers should meet the following minimum specifications:

- **CPU**: 2+ cores
- **RAM**: 2 GB minimum (expandable based on load)
- **Storage**: 20 GB disk space (expandable)
- **Network**: Stable internet connection

Client Requirements

Users need:

- A modern web browser (for web-based access)
- A smartphone/tablet (for the Flutter app)
- Minimum network bandwidth: 56 Kbps for basic usage, higher for real-time updates.

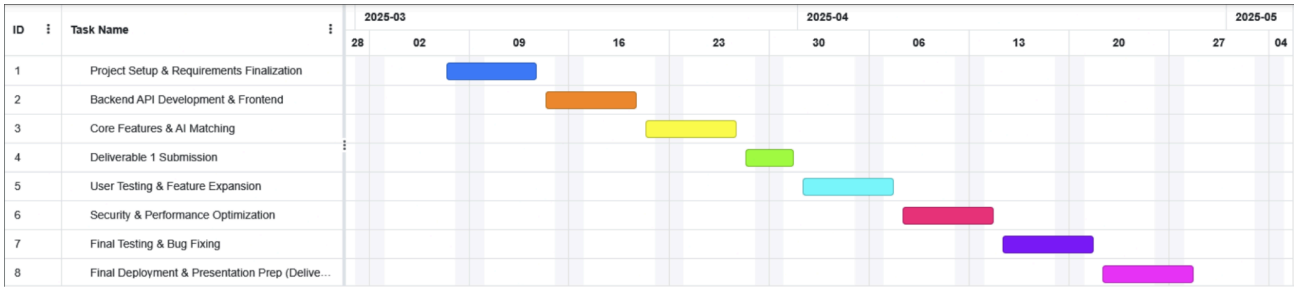
Section 6 : Plan of Work

6.1 Timeline and Milestones

Week	Task	Backend(Kevin, Casey)	Frontend (Andy, Sawyer)
Week 1 Mar 7 - Mar 13	**Project Setup & Requirements Finalization** ● Define API contracts & database schema ● Create UI wireframes and design layout ● Set up project repo,	Define user roles, authentication, database setup	Design wireframes for login, dashboard, request submission

	documentation		
Week 2 Mar 14 - Mar 20	**Backend API Development & Frontend Prototype** ● Implement user authentication (RBAC) ● Develop API for submitting aid requests ● Frontend builds static UI with mock data	Set up database, start API routes	Develop login, aid request form, and dashboard UI
Week 3 Mar 21 - Mar 27	**Core Features & AI Matching** ● Implement AI matching algorithm ● Develop live map API for geolocation tracking ● Frontend integrates basic API calls	Implement AI matching, finalize core API endpoints	Connect frontend to APIs, refine UI components
Week 4 Mar 28 - Mar 30	**Deliverable 1 Submission** ● Fully functional login, aid request, and AI matching system ● Frontend connected to backend ● Initial bug fixes & documentation	Backend deployed & tested	UI integrated with backend
Week 5 Apr 1 - Apr 7	**User Testing & Feature Expansion** ● Implement real-time push notifications ● Improve volunteer & NGO dashboard ● Conduct initial stress tests	Add real-time notifications	Refine volunteer & NGO interfaces
Week 6 Apr 8 - Apr 14	**Security & Performance Optimization** ● Implement data encryption & secure authentication (2FA) ● Optimize AI matching efficiency ● Improve offline support for low-connectivity areas	Backend security updates, performance tuning	Frontend testing, UI bug fixes

Week 7 Apr 15 - Apr 21	**Final Testing & Bug Fixing** ● Stress test high user loads ● Fix UI inconsistencies ● Conduct integration testing	Ensure backend scalability, optimize DB	Ensure smooth user experience, finalize UI
Week 8 Apr 22 - Apr 27	**Final Deployment & Presentation Prep** ● Deploy final version of the platform ● Prepare for final presentation ● Conduct last-minute testing & debugging	Final backend deployment	UI/UX polish, final testing



6.2 Product Ownership and Responsibility Breakdown

- **Casey Nguyen:** Oversee project management; contribute to both front and back ends.
- **Kevin Pulikkottil:** Lead backend development and AI matching implementation.
- **Sawyer:** Handle front-end development and database-related UI integration.
- **Andy Jih:** Assist in front-end development and user documentation.

Section 7 : References

UML Diagrams - UML.drawio : for designing UML Diagrams. The tool is useful for designing and documenting different aspects of the disaster relief platform.

ScienceDirect – Crowdsourcing in Disaster Relief: Database and research. A large article about the uses of Crowdsources and how it is helpful. Discusses how crowdsourcing

technologies can be leveraged to enhance disaster response. It provides insights into best practices, challenges, and case studies that inform the development of the proposed disaster relief platform.

Global Disaster Preparedness Center – Case Studies: The Global Disaster Preparedness Center (GDPC) is an initiative focused on improving disaster response and preparedness worldwide. Case studies from GDPC provide real-world examples of how different regions and organizations have managed disaster response, offering valuable insights into best practices for developing an effective relief platform.

Google Maps API Documentation: The Google Maps API allows developers to integrate interactive maps, geolocation services, and real-time tracking into applications. This documentation provides guidelines on how to use mapping services effectively, which is crucial for pinpointing affected areas, locating resources, and guiding volunteers and first responders.

Twilio SMS API Documentation: Twilio provides cloud-based communication services, including SMS, voice, and video messaging. The Twilio SMS API allows the disaster relief platform to send real-time alerts, emergency messages, and updates to victims, volunteers, and organizations. This documentation explains how to implement SMS-based notifications.

FastAPI Documentation: FastAPI is a modern web framework for building APIs with Python. It is known for its speed, efficiency, and ease of use. The FastAPI documentation provides details on creating and managing APIs, which are essential for connecting different components of the disaster relief system, such as mobile apps, databases, and external services.