

# Aid-Hive

(Blood Donation Management System)



*Under the supervision of*

Prof. Imran Shafique

Bachelor of Science in CS/IT (2021-2025)

Department of Information Technology,

Government Islamia College,

Gujranwala

# Aid-Hive

A project presented to  
Government Islamia College, Gujranwala

In partial fulfilment  
of the requirement for the degree of

Bachelor of Science in Information Technology (2021-2025)

By

Abdur Rehman	2021-ICG-663
Ammad Ahmad	2021-ICG-97
Ali Hassan	2021-ICG-120
Hafiz Maaz Ahmad Siddiqi	2021-ICG-117

Bachelor of Science in CS/IT (2021-2025)

Department of Information Technology,

Government Islamia College,

Gujranwala

## DECLARATION

We, the undersigned members of the project team, hereby confirm that the software system titled "Aid-Hive: Blood Donation Management System", along with its documentation, is our own original work. We also confirm that this project, in whole or in part, has not been plagiarized from any source, nor has it been submitted previously for any degree, diploma, or qualification at this or any other university or institution of learning. In the event any part of this work is determined to be a plagiarized work or copy of any other project, we take full responsibility for the same. This project and report have been designed and prepared by our team from scratch under the guidance of our revered supervisor.

Signature: -----

Abdur Rehman 073151

Signature: -----

Ammad Ahmad 056503

Signature: -----

Ali Hassan 056427

Signature: -----

Hafiz Maaz Ahmed Siddiqi 056430

## CERTIFICATE OF APPROVAL

It is to confirm that the final year design project (FYDP) of BS-IT "Blood Donation Management System" was prepared by **Abdur Rehman (2021-ICG-663)**, **Ammad Ahmad (2021-ICG-97)**, **Ali Hassan (2021-ICG-120)**, and **Hafiz Maaz Ahmad Siddiqi (2021-ICG-117)** under the guidance of "**Prof. Imran Shafique**" in my view; it is complete enough, in scope and quality, for the award of Bachelors of Science in Information Technology.

**Signature: -----**

**FYDP Supervisor: Prof. Imran Shafique**

**Signatures (Faculty Advisory Committee (FAC))**

<b>Signatures</b>			
<b>Name</b>			

**Signature: -----**

**Head of FYDP Coordination Office:**

**Signature: -----**

**Dated: \_\_\_\_\_**

**Chairperson, Department of Information Technology**

## **Executive Summary**

Our project Aid-Hive is an online portal through which patients and blood donors are matched. Patients can order blood through the portal, and donors in their vicinity are alerted. In case there is a matching donor, he can donate blood at a center within his vicinity. One of the most important features of the system is live tracking, which is utilized to track donors and Aid-Hive staff in real time. It makes it efficient and trustworthy. Aid-Hive is developed with the MERN stack with MySQL database. It offers a simple and user-friendly interface to donors, patients, and admins. The aim of our project is to save effort and time and ultimately save lives.

## Acknowledgement

First and foremost, we express our gratitude to Allah Almighty whose blessings have enabled us to accomplish our Final Year Project successfully. We are truly grateful to our honorable supervisor, **Prof. Imran Shafique**, for his constant support, guidance, and encouragement. We are thankful for the assistance of **Mr. Usman Ahmed** for this project. We thank our teachers, classmates, and everyone who offered suggestions and feedback. Lastly, we appreciate our friends and families for their encouragement and support, which enabled us to carry out this project.

Signature: -----

Abdur Rehman      073151

Signature: -----

Ammad Ahmad      056503

Signature: -----

Ali Hassan      056427

Signature: -----

Hafiz Maaz Ahmad Siddiqi      056430

## Table of Contents

<b>1. Introduction .....</b>	<b>2</b>
1.1 Problem Statement .....	2
1.2 Problem Solution .....	2
1.3 Objectives of the Proposed System.....	2
1.4 Scope .....	3
1.5 System Components .....	3
1.5.1 Pages of the System .....	3
1.5.2 Buttons of the System .....	3
1.5.3 Donor and Recipient Forms .....	3
1.5.5 Database (MySQL).....	4
1.6 Related System Analysis/Literature Review .....	4
1.7 Vision Statement.....	5
1.8 System Limitation and Constraints .....	5
1.8.1 System Limitations:.....	5
1.8.2 System Constraints:.....	5
1.9 Tools and Technologies.....	6
1.9.1 Frontend .....	6
1.9.2 Backend .....	6
1.9.3 Database .....	6
1.9.4 Development Tools .....	6
1.10 Project Planning.....	8
1.11 Summary.....	8
<b>2. Analysis .....</b>	<b>10</b>
2.1 User Functions and Characteristics .....	10
2.2 Requirement Identifying Technique .....	13
2.3 Functional Requirements.....	16
2.3.1 Functional Requirement 1 .....	16
2.3.2 Functional Requirements 2 .....	17
2.3.3 Functional Requirement 3 .....	17
2.3.4 Functional Requirement 4 .....	18
2.3.5 Functional Requirement 5 .....	18
2.3.6 Functional Requirement 6 .....	19
2.4 Non-Functional Requirements .....	20
2.4.1 Reliability.....	20
2.4.2 Useability .....	20

2.4.3	<i>Performance</i> .....	20
2.4.4	<i>Security</i> .....	20
2.5	<b>External Interface Requirements</b> .....	21
2.5.1	<i>User Interfaces Requirements</i> .....	21
2.5.2	<i>Software interfaces</i> .....	21
2.5.3	<i>Hardware interfaces</i> .....	22
3.5.4	<i>Communications interfaces</i> .....	23
2.6	<b>Summary</b> .....	24
3.	<b>System Design</b> .....	26
3.1	<b>Design considerations</b> .....	26
3.2	<b>Design Models</b> .....	28
3.2.2	<i>Interaction Diagram</i> .....	33
3.2.4	<i>Recipient State Transition</i> .....	37
3.3	<b>Architectural Design</b> .....	38
3.4	<b>Data Design</b> .....	41
3.4.1	<i>Data Dictionary</i> .....	43
3.5	<b>User Interface Design</b> .....	44
3.5.1	<i>Screen Images</i> .....	46
3.5.2	<i>Screen Objects and Actions</i> .....	51
3.6	<b>Behavioral Model</b> .....	54
3.6.1	<i>Recipient Registration</i> .....	54
3.6.2	<i>Donor Registration</i> .....	55
3.6.5	<i>Admin Panel</i> .....	58
3.6.6	<i>Login</i> .....	59
3.6.7	<i>Logout</i> .....	60
3.6.8	<i>Search for donor</i> .....	61
3.7	<b>Design Decisions</b> .....	61
3.8	<b>Summary</b> .....	62
4.	<b>Implementation</b> .....	64
4.1	<b>Code Repository</b> .....	64
4.2	<b>Summary</b> .....	65
5.	<b>Introduction</b> .....	67
5.1	<b>Unit Testing (UT)</b> .....	67
5.1.1	<i>Donor Registration</i> .....	68
5.1.2	<i>Recipient Registration</i> .....	69
5.1.3	<i>Admin Login</i> .....	70



<b>5.2 Functional Testing (FT).....</b>	<b>71</b>
<b>5.2.2 Recipient Registration .....</b>	<b>72</b>
<b>5.2.3 Admin Login.....</b>	<b>73</b>
<b>5.2.4 Admin Panel Controls .....</b>	<b>73</b>
<b>5.3 Integration Testing (IT).....</b>	<b>74</b>
<b>5.3.1 Registration → Request Submission Flow.....</b>	<b>74</b>
<b>5.3.2 Admin Action → User Login Blocked.....</b>	<b>75</b>
<b>5.3.3 Donor Sets Availability → System Updates Matching Pool.....</b>	<b>76</b>
<b>5.3.4 Admin Views Logs → Matches with User History.....</b>	<b>76</b>
<b>5.4 Performance Testing (PT).....</b>	<b>77</b>
<b>5.4.1 Login API Response Time .....</b>	<b>77</b>
<b>5.4.2 Request Creation Under Load .....</b>	<b>78</b>
<b>5.4.3 Admin Panel Performance .....</b>	<b>78</b>
<b>5.5 Summary .....</b>	<b>79</b>
<b>6. Introduction.....</b>	<b>81</b>
<b>6.1 Conversion Method .....</b>	<b>81</b>
<b>6.2 Summary .....</b>	<b>83</b>
<b>7. Introduction.....</b>	<b>85</b>
<b>7.1 Evaluation.....</b>	<b>85</b>
<b>7.2 Traceability Matrix.....</b>	<b>86</b>
<b>7.3 Conclusion .....</b>	<b>87</b>
<b>7.4 Future Work.....</b>	<b>87</b>
<b>References.....</b>	<b>88</b>
<b>Appendix A.....</b>	<b>89</b>
<b>Use case Description Template .....</b>	<b>89</b>

## List of Table

Table 1: Related System Analysis with proposed project solution.....	4
Table 2: Tools and Technologies for Proposed Project.....	7
Table 3: Project Planning Gantt Chart.....	8
Table 4: System Functions and User Characteristics.....	12
<i>Table 5: Use Case List.....</i>	<i>14</i>
Table 6 :Donor Registration .....	16
Table 7 :Recipient Registration .....	17
Table 8 : Search Donors .....	17
Table 9 : Admin login.....	18
Table 10 : Manage Records .....	18
Table 11 : Live Location Tracking.....	19
Table 12 : Database Management.....	19
Table 13: Data Dictionary Table.....	43
Table 14 : Functions and Function Parameters.....	44
Table 15: Use case 1 .....	51
Table 16: User Registration .....	68
Table 17: Recipient Registration .....	69
Table 18: Admin Login.....	70
Table 19: Find Blood.....	71
Table 20: Donor Registration .....	72
Table 21: Recipient Registration .....	72
Table 22: Admin login.....	73
Table 23: Admin Control.....	73
Table 24: Test Case 1 .....	74
Table 25: Test Case 2.....	75
Table 26: Test Case 3.....	76
Table 27: Test Case 6.....	76
Table 28: Test Case 1 .....	77
Table 29: Test Case 2.....	78
Table 30: Test Case 5.....	78
<i>Table 31 : Evaluation Table.....</i>	<i>85</i>
<i>Table 32 : Requirement Traceability Matrix.....</i>	<i>86</i>
<i>Table 33 : Correlation Table .....</i>	<i>87</i>
Table 34 : Table A- 1Show the detail use case template and example .....	90

## List of Figures

Figure 1: Use Case of Aid Hive.....	15
Figure 2: Component Diagram.....	29
Figure 3: Interaction diagram of Aid Hive .....	33
Figure 4: Donor State Transition Diagram of Aid Hive .....	35
<i>Figure 5 : Recipient State Transition Diagram .....</i>	<i>37</i>
Figure 6: Architectural Design of Aid Hive.....	40
<i>Figure 7 : ER Diagram.....</i>	<i>42</i>
<i>Figure 8 : Landing Page .....</i>	<i>47</i>
Figure 9 : Admin Login Form .....	48
Figure 10 : Donor Registration Form .....	49
Figure 11 : Recipient Registration Form .....	50
Figure 12: Recipient Registration.....	54
Figure 13: Donor Registration.....	55
Figure 14: Manage Profile.....	56
Figure 15: Search for Donor.....	57
Figure 16: Admin Panel.....	58
Figure 17 :Admin Login.....	59
Figure 18: Admin Logout .....	60
Figure 19: Search for Donor.....	61

# **Chapter 1**

## **Introduction**



# 1. Introduction

## 1.1 Problem Statement

In emergency cases, the majority of patients experience immense delays in accessing blood donations since there is no real-time system in place. The available platforms lack real-time tracking of donors' locations, priority notifications, and fast communication among donors and patients. Our project Aid-Hive aims to bridge this deficiency with a trusted platform that provides real-time features to link donors and patients efficiently.

## 1.2 Problem Solution

In order to address the challenge of delayed blood donation, our team came up with Aid-Hive, an online real-time platform that rapidly matches donors and patients. Our system includes live location tracking, emergency notifications, and safe communication to ensure that patients receive assistance immediately without any delay. Users can readily find donors around them, request blood, and get instant alerts. Donors are also able to respond quickly, and both can talk in real time. Aid-Hive also stores the history of donations and gives an easy-to-use simple dashboard. With these functionalities, our project bridges the loopholes of current systems and streamlines blood donation as faster, smarter, and more reliable, particularly in emergency situations.

## 1.3 Objectives of the Proposed System

Our project Aid-Hive is a blood donation management system built with the MERN stack and MySQL as the database. The primary goal of this system is to have a fast, simple, and secure means of connecting donors and recipients. Our goal is to enable patients to find compatible donors by blood type and geography easily. Donors and recipients can register in the system by filling out forms without undergoing a complicated login or signup procedure. The admin alone has the privileges for logging in to administer the website. The system also has live location tracking via Google Maps, whereby patients can easily find nearby donors or Aid-Hive workers. Together with this, emergency requests will be sent through real-time notifications and alerts, which will guarantee timely response and quicker communication.



### 1.4 Scope

The functionality of our project Aid-Hive is to offer an online blood donation management system that simplifies the process and speeds it up. The system is developed with the MERN stack, and MySQL serves as the database to save donor and recipient information. Our project has five primary pages: Home, Donate, Find Blood, Aid-Hive Workers, and About Us. It also includes two main buttons: Aid-Hive Admin login and Live Location Tracking. These features simplify the system so that it becomes very user-friendly and easy to handle

### 1.5 System Components

The Aid-Hive system is made up of different components that work together to provide blood donation management.

#### 1.5.1 Pages of the System

The system has five main pages:

- Home Page: Gives an overview and easy navigation.
- Donate Page: Donors fill a form with their details to register as donors.
- Find Blood Page: Recipients fill a form to request blood.
- Aid-Hive Workers Page: Shows information and live location of workers.
- About Us Page: Provides project and team information.

#### 1.5.2 Buttons of the System

The system has two main buttons:

- Aid-Hive Admin Button: Only for admin login.
- Live Location Tracking Button: To track donors and workers in real time.

#### 1.5.3 Donor and Recipient Forms

Donors and recipients can join by filling forms with their details. No login is required for them.

#### 1.5.4 Admin Login



Login is only available for the admin. The admin can manage donor/recipient data and oversee the system.

### 1.5.5 Database (MySQL)

The system uses MySQL database to store donor details, blood requests, and history records safely.

## 1.6 Related System Analysis/Literature Review

There are several online systems for donating blood that are already in use, but they do not cater to emergency requirements efficiently. In Pakistan and other countries, none of these available systems are well-equipped or user-friendly. The following is a brief overview of some of these existing systems and how our project Aid-Hive is better.

*Table 1: Related System Analysis with proposed project solution*

System / Website	Main Features	Limitations	Aid-Hive Improvements
<b>Sehat Kahani Blood Services</b>	Connects hospitals and patients for arranging blood donations.	Focused on hospitals only, no real-time donor location, and not open for all users.	Open for everyone and allows direct connection between donor and recipient.
<b>Red Cross Blood Services</b>	Large-scale international donor management system with hospital integration.	Complicated system, not user-friendly for local or individual needs, requires multiple steps.	Aid-Hive is lightweight, simple, and localized for quick use in Pakistan.
<b>Aid-Hive (Our System)</b>	Donor and recipient form filling, MySQL database, 5 web pages, and admin login.	Does not include automated push notifications or SMS alerts.	Focused on simplicity, visibility, and easy access.



## **1.7 Vision Statement**

Aid-Hive's vision is to develop a dependable, accessible, and technology-based blood donation management system that bridges donors and recipients in real time. By combining secure data storage space, live location tracking, and effective matching of donors and recipients, the system intends to minimize delays in the case of emergency, establish transparency, and foster a culture of voluntary blood donation. In the future, Aid-Hive plans to expand its platform to connect with hospitals, blood banks, and medical organizations, becoming a trusted online platform for lifesaving blood donation services.

## **1.8 System Limitation and Constraints**

### ***1.8.1 System Limitations:***

1. The system does not provide real-time alerts or notifications. Users must check the platform themselves.
2. Only the admin has a login system. Donors and recipients can only fill forms; no personal accounts are created for them.
3. The system works only with an active internet connection. Without internet, it cannot be used.
4. Location tracking accuracy depends on Google Maps and the user's device GPS, so it may not always be 100% precise.
5. The database is limited to MySQL, which may face challenges if the system is scaled to a very large number of users.
6. There is no direct integration with hospitals or blood banks at this stage.

### ***1.8.2 System Constraints:***

#### ***1. Technology Constraint:***

The system is built using the MERN stack with MySQL database, so it is limited to web-based usage only.

#### ***2. Connectivity Constraint:***

Continuous internet connection is required for form submission, database access, and live location tracking.





### 3. **Hardware Constraint:**

Users need a device (desktop, laptop, or smartphone) with a browser and GPS support for location tracking.

### 4. **User Constraint:**

Only the admin has login access. Donors and recipients can interact with the system only through form filling.

### 5. **Location Tracking Constraint:**

Accuracy of live tracking depends on Google Maps API and the device GPS, which may not always be fully precise.

### 6. **Data Constraint:**

The system stores information only in the MySQL database, which requires regular maintenance and optimization when data grows larger.

## 1.9 Tools and Technologies

The following tools and technologies were used to develop Aid-Hive (Blood Donation Management System):

### 1.9.1 Frontend

- **React.js** → For building the user interface.
- **CSS** → For styling and layout of web pages.

### 1.9.2 Backend

- **Node.js** → For server-side logic.
- **Express.js** → For connecting frontend with backend.

### 1.9.3 Database

- **MySQL** → To store and manage donor and recipient information.

### 1.9.4 Development Tools

- **Visual Studio Code (VS Code)** → Code editor.
- **npm (Node Package Manager)** → For installing and managing dependencies.



- **GitHub** → For version control and team collaboration.

*Table 2: Tools and Technologies for Proposed Project*

Category	Technology / Tool	Purpose
<b>Frontend</b>	React.js	To build the user interface of the system.
	CSS	For styling and layout of web pages.
	Framer Motion	For smooth animations and transitions.
<b>Backend</b>	Node.js	For server-side scripting and logic.
	Express.js	To handle APIs and connect frontend & backend.
<b>Database</b>	MySQL	To store and manage donor and recipient records.
<b>Libraries / APIs</b>	Axios	To handle HTTP requests between frontend and backend.
<b>Development</b>	Visual Studio Code (VS Code)	Code editor for development.
	npm (Node Package Manager)	To install and manage project dependencies.
	GitHub	For version control and collaboration.



## 1.10 Project Planning

Table 3: Project Planning Gantt Chart

Start Date					January	Feb	March	April	May	June
Task No.	Task	Supervisor	Start Date	End Date						
1	Project Proposal	Prof. Imran Shafique								
2	SRS and SDS	Prof. Imran Shafique								
3	RTM	Prof. Imran Shafique								
4	Frontend	Prof. Imran Shafique								
5	Backend	Prof. Imran Shafique								
6	DB Integration	Prof. Imran Shafique								
7	Test Plans	Prof. Imran Shafique								
8	Test Report	Prof. Imran Shafique								

## 1.11 Summary

The Aid-Hive Blood Donation Management System is designed to address the issues of conventional blood donation like delays, unverified data, and a lack of ease of finding proper donors. It offers a web-based system through which donors and recipients can register via forms, and the system can be securely managed by admins. The system has five primary pages (Home, Donate, Find Blood, Aid-Hive Workers, About Us) and two primary features (Admin Login and Live Location Tracking). Developed based on the MRN stack with MySQL as the database, Aid-Hive provides efficiency, accessibility, and reliability in blood donation activity management.



# **Chapter 2**

## **Requirements Analysis**



## **2. Analysis**

Analysis phase is concerned with grasping the problem with current blood donation procedures and outlining how the envisioned system will address them. It involves examining existing limitations, determining the needs of the users, and outlining the functional and non-functional specifications of the Aid-Hive Blood Donation Management System. This phase ensures that the system design is founded on precise objectives, users' requirements, and consideration of feasibility.

### **2.1 User Functions and Characteristics**

The Aid-Hive Blood Donation Management System is designed to serve three main categories of users: Donors, Recipients, and Admins. Each user type has specific functions and characteristics that define their role in the system.

#### **1. Donor**

- **Functions:**
  - Register on the website by filling out the donor registration form.
  - Provide required details such as name, age, gender, blood group, contact information, and location.
  - Become available in the system's database for recipients to search.
- **Characteristics:**
  - Donors are voluntary individuals who are willing to donate blood.
  - They require a simple and user-friendly registration interface.
  - Their data must be stored securely and made accessible to recipients when required.

#### **2. Recipient**

- **Functions:**
  - Register on the website by filling out the recipient form with necessary details (name, required blood group, contact number, and location).



- Search for available donors based on blood group and geographical location.
- Request blood through the system in urgent situations.
- **Characteristics:**
  - Recipients are usually patients or attendants in urgent need of blood.
  - They expect quick search results and accurate donor information.
  - Their requests must be handled efficiently to avoid delays in emergencies.

### 3. Admin

- **Functions:**
  - Log in securely through the admin login page.
  - Manage donor and recipient records in the MySQL database.
  - Approve, monitor, or delete any donor or recipient entries if needed.
  - Ensure that the data remains updated, accurate, and secure.
- **Characteristics:**
  - Admins are authorized personnel with full system access.
  - They are responsible for maintaining the privacy and integrity of the system.
  - They must have technical knowledge to handle system operations effectively.



*Table 4: System Functions and User Characteristics*

User Type	Functions	Characteristics
<b>Donor</b>	<ul style="list-style-type: none"> <li>• Fill out donor registration form</li> <li>• Provide details (Name, Age, Gender, Blood Group, Contact, Location)</li> <li>• Appear in system database for matching with recipients</li> </ul>	<ul style="list-style-type: none"> <li>• Voluntary individuals willing to donate blood</li> <li>• Require a simple and user-friendly interface</li> <li>• Information must be stored securely and accessed quickly</li> </ul>
<b>Recipient</b>	<ul style="list-style-type: none"> <li>• Fill out recipient registration form</li> <li>• Search donors by blood group and location</li> <li>• Submit request in urgent cases</li> </ul>	<ul style="list-style-type: none"> <li>• Usually patients/attendants in urgent need of blood</li> <li>• Expect fast response and accurate donor details</li> <li>• Depend on reliable donor availability</li> </ul>
<b>Admin</b>	<ul style="list-style-type: none"> <li>• Log in securely using admin credentials</li> <li>• Manage donor and recipient records</li> <li>• Approve, update, or remove entries</li> <li>• Monitor blood requests and system activity</li> </ul>	<ul style="list-style-type: none"> <li>• Authorized personnel with full system access</li> <li>• Responsible for data accuracy, privacy, and security</li> <li>• Must have technical knowledge to operate system</li> </ul>



## 2.2 Requirement Identifying Technique

Requirement identifying is the process of determining what the system must do to meet user needs effectively. For the Aid-Hive Blood Donation Management System, the following techniques were used to identify and gather requirements:

### 1. Interviews with Stakeholders

- Discussions were held with potential donors, recipients, and administrators to understand their expectations and challenges.
- Questions focused on difficulties in current blood donation processes, preferred features, and usability requirements.

### 2. Observation of Existing System

- Studied current blood donation practices in hospitals and blood banks.
- Identified delays in donor-recipient communication, lack of centralized data, and inefficiency in emergency cases.

### 3. Questionnaires/Surveys

- Distributed online and offline forms to gather inputs from volunteers and patients.
- Collected data on desired features, accessibility preferences, and common issues faced.

### 4. Brainstorming Sessions

- Group discussions among project members to decide which features are essential.
- Prioritized functionalities like admin login, live location tracking, and donor/recipient forms.

### 5. Review of Existing Literature & Systems

- Studied similar blood donation management systems and software to identify best practices.





- Noted features that were missing or needed improvement.

### 6. Use Case Technique

- The Use Case technique helps analyze how users interact with the system in real-life scenarios.
- It allows the team to define clear, user-focused features by relating each user action to the corresponding system behavior.
- This approach ensures that functional requirements are accurately specified, complete, and aligned with actual user needs.

#### Outcome:

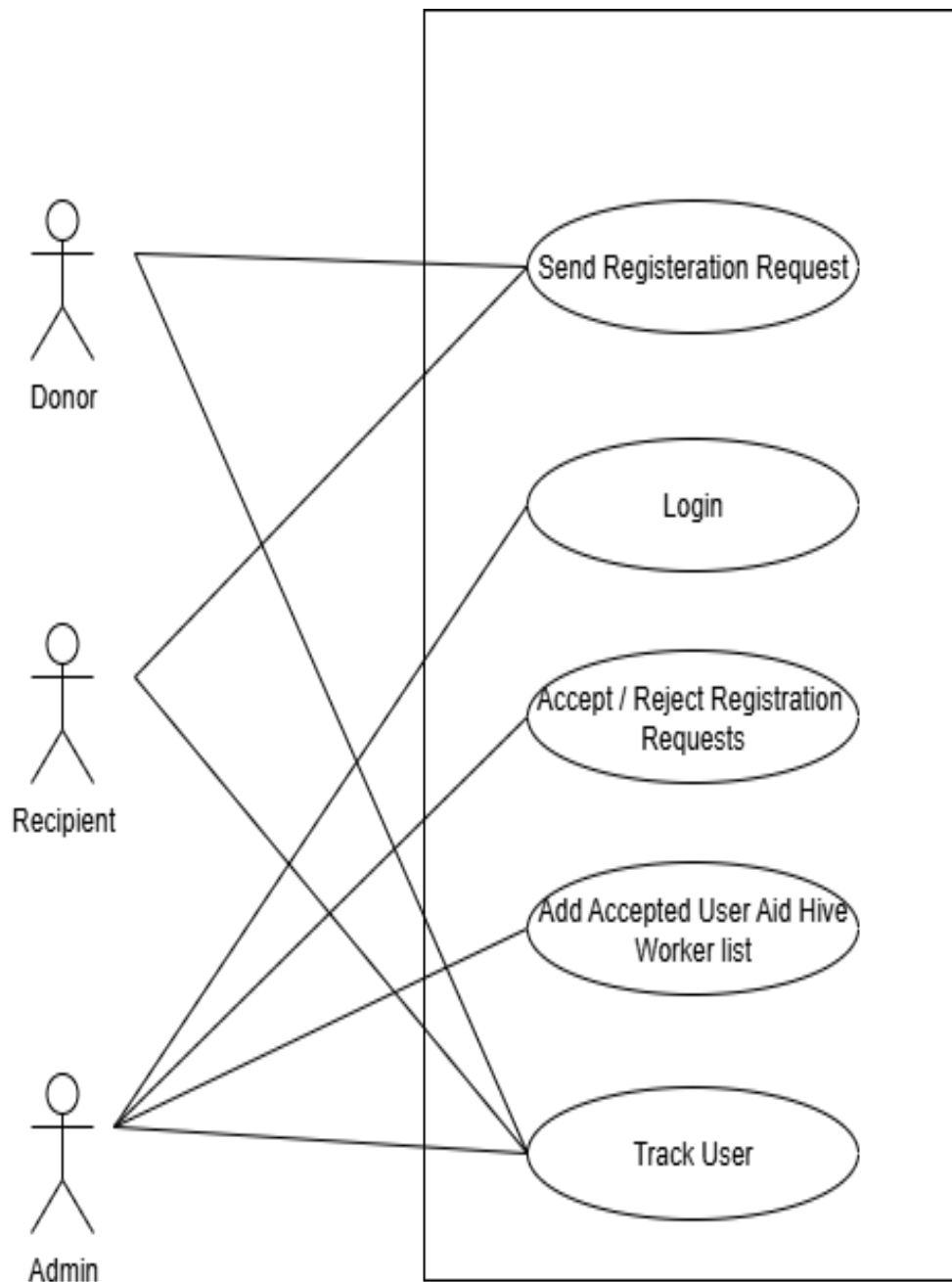
By using these techniques, the project team clearly defined functional and non-functional requirements, ensuring that the system effectively serves donors, recipients, and administrators.

*Table 5: Use Case List*

Use Case Name	Associated Requirement
Create Blood Request	Recipients must be able to submit requests with blood type, urgency, and location.
Receive Request Notification	Donors must be notified in real-time when a nearby matching request is created.
Accept/Decline Request	Donors must be able to respond to requests based on availability.
View Donation/Request History	Users must see their past activity in the dashboard.
Mark Emergency Request	Recipients should mark a request as “emergency” for priority matching.



Figure 1: Use Case of Aid Hive





### 2.3 Functional Requirements

The most important functional requirements of the Aid Hive system. Requirements are grouped by system functions or features with which the users interact. They are developed to address user requirements (donor and recipient) and allow for smooth, consistent, and secure deployment of the application.

#### 2.3.1 Functional Requirement 1

*Table 6 :Donor Registration*

<b>Identifier:</b>	FR-1
<b>Title:</b>	Donor Registration
<b>Description:</b>	Allow donors to register via an online form by providing personal and contact details
<b>Input Fields:</b>	Name, Age, Gender, Blood Group, Contact Number, Address/Location
<b>Processing:</b>	Validates form data and stores donor information securely in the MySQL database.
<b>Output:</b>	Confirmation message of successful registration; donor added to the database for recipient search.
<b>Dependencies:</b>	MySQL database for storing data; web form frontend for input; server-side backend for processing.
<b>Priority:</b>	High



### 2.3.2 Functional Requirements 2

Table 7 :Recipient Registration

<b>Requirement ID:</b>	FR-2
<b>Requirement name:</b>	Recipient Registration
<b>Description:</b>	Allows recipients to register their details and blood requests via an online form.
<b>User Type:</b>	Recipient
<b>Input Fields:</b>	Name, Required Blood Group, Contact Number, Address/Location, Request Date
<b>Processing:</b>	Validates form data and stores recipient information securely in the MySQL database.
<b>Dependencies:</b>	MySQL database for storing data; web form frontend for input; server-side backend for processing.
<b>Priority</b>	High

### 2.3.3 Functional Requirement 3

Table 8 : Search Donors

<b>Requirement ID:</b>	FR-3
<b>Requirement name:</b>	Search Donors
<b>Description:</b>	Enables recipients to search for available donors based on blood group and location.
<b>User Type:</b>	Recipient
<b>Input Fields:</b>	Blood Group, Location (City/Area)
<b>Processing:</b>	Matches recipient search criteria with donor records stored in the MySQL database.
<b>Output:</b>	List of matching donors with contact details and location information.



<b>Dependencies:</b>	MySQL database for storing donor data; frontend search interface; backend processing for query matching.
----------------------	--

#### 2.3.4 Functional Requirement 4

Table 9 : Admin login

<b>Requirement ID:</b>	FR-4
<b>Requirement name:</b>	Admin Login
<b>Description:</b>	Provides secure login access for administrators to manage system operations.
<b>User Tye:</b>	Admin
<b>Input Fields:</b>	Username, Password
<b>Processing:</b>	Validates admin credentials against records in the MySQL database and grants access to the admin dashboard.
<b>Output:</b>	Access to the admin panel for managing donors, recipients, and system activities.
<b>Dependencies:</b>	MySQL database for storing admin credentials; secure backend authentication; frontend login interface.

#### 2.3.5 Functional Requirement 5

Table 10 : Manage Records

<b>Requirement ID:</b>	FR-5
<b>Requirement name:</b>	Manage Records
<b>Description:</b>	Allows administrators to view, approve, update, or remove donor and recipient records in the system.
<b>User Tye:</b>	Admin
<b>Input Fields:</b>	Record ID, Action Type (Approve / Update / Delete), Updated Data (if applicable)
<b>Processing:</b>	Performs the selected action on the chosen record in the MySQL database and ensures data integrity.



<b>Output:</b>	Confirmation of successful action (record approved, updated, or deleted).
<b>Dependencies:</b>	Admin login authentication; MySQL database; backend processing for CRUD operations.

### 2.3.6 Functional Requirement 6

Table 11 : Live Location Tracking

<b>Requirement ID:</b>	FR-6
<b>Requirement name:</b>	Live Location Tracking
<b>Description:</b>	Tracks the real-time location of donors and recipients to assist in urgent blood requests.
<b>User Tye:</b>	Donor / Recipient
<b>Input Fields:</b>	Location coordinates (latitude, longitude) or address
<b>Processing:</b>	Uses integrated maps/GPS service to track and update the location in real-time; matches nearby donors with recipient requests.
<b>Output:</b>	Display of donor/recipient location on the map interface; notification of nearby matches for urgent requests.
<b>Dependencies:</b>	GPS or Maps API integration; MySQL database for storing location data; backend processing for real-time updates.

### 2.3.7 Functional Requirement 7

Table 12 : Database Management

<b>Requirement ID:</b>	FR-7
<b>Requirement name:</b>	Database Management
<b>Description:</b>	Stores and manages all donor, recipient, and admin information securely in the MySQL database.
<b>User Tye:</b>	All Users
<b>Input Fields:</b>	Donor/Recipient/Admin details entered via forms or admin panel



<b>Processing:</b>	Validates, stores, retrieves, updates, and deletes records as per user actions; ensures data integrity and security.
<b>Output:</b>	Accurate storage and retrieval of user information; updates reflected in the system in real-time.
<b>Dependencies:</b>	MySQL database; backend server for database operations; secure access control.

## 2.4 Non-Functional Requirements

These are quality requirements for the extent to which the Aid Hive system functions and performs, not for what the system does. They are expectations of reliability, usability, performance, and security.

### 2.4.1 Reliability

It ought to operate without constant crashing. It ought to recover from abrupt errors gracefully without losing data. It ought to incorporate backup procedures so that critical operations like the posting of blood orders are not lost.

### 2.4.2 Useability

Aid Hive should be straightforward and easy to use for all, even non-technical people. The most important features should be available in two or three steps, and the interface should be based on established standards and guidelines with accessibility features.

### 2.4.3 Performance

The system should react to user activities in 2 seconds, and real-time messages like acceptance of requests should be received within 3 seconds. It should support 500 simultaneous users and remain responsive even on low-bandwidth cellular networks.

### 2.4.4 Security

All user data must be stored and transmitted securely with HTTPS. It must be role-based access (donor, recipient, admin), and the system must block common attacks like SQL injection and XSS. Sessions must timeout automatically if idle for 15 minutes.



## 2.5 External Interface Requirements

Here's some external interface requirements are as follows:

### 2.5.1 User Interfaces Requirements

The web application shall provide an intuitive, responsive, and accessible user interface for the following:

- **Donors can:**
  - Register/login
  - Update their profiles
  - See blood requests
- **Recipients can:**
  - Search/filter donors by blood group and location
  - Submit a blood request
  - Track request status
- **Admin Panel:**
  - Firstly, login then connect to Aid-Hive
  - Accept and reject requests
  - Manage users (donors and recipients)

UI will be built using HTML, Tailwind, JavaScript (React in MERN), and will be fully responsive for all screen sizes. Forms will include validation, tooltips, and error handling.

### 2.5.2 Software interfaces

The Aid-Hive Blood Donation Management System uses multiple software components to ensure smooth functioning and interaction between different modules. The key software interfaces are:

#### 1. Front-End Interface (MERN Stack – React & Node.js)

- Provides a user-friendly interface for donors and recipients to fill registration forms and search for blood.
- React ensures dynamic web pages and smooth navigation.





- Node.js handles requests and communicates with the backend.

### 2. Back-End Interface (Express.js)

- Acts as the middleware to process requests between the front-end and the database.
- Handles form submissions, validation, and API requests.

### 3. Database Interface (MySQL)

- Stores donor, recipient, and admin data.
- Interfaces with the backend to allow insertion, deletion, updating, and retrieval of records.
- Ensures data consistency and integrity.

### 4. Admin Panel Interface

- Allows only admins (through login) to manage donor and recipient records.
- Provides access to view, approve, or remove entries.

### 5. Maps / Location API Interface

- Connects with external GPS/Map service to provide live location tracking of donors and recipients.

#### 2.5.3 Hardware interfaces

Since Aid-Hive is a web-based application, it does not require specialized hardware. However, certain minimum hardware requirements are necessary for users (donors, recipients, and admins).

#### 1. User Side (Donors & Recipients)

- **Device Type:** Desktop computer, laptop, tablet, or smartphone.
- **Minimum Requirements:**
  - Processor: 1.5 GHz or higher
  - RAM: 2 GB or higher



- Storage: 200 MB free space for browser cache
- Internet: Stable connection (at least 1 Mbps)
- **Interface:** Users interact with the system through a web browser.

### 2. Admin Side

- **Device Type:** Desktop computer or laptop.
- **Minimum Requirements:**
  - Processor: 2.0 GHz Dual-Core or higher
  - RAM: 4 GB or higher
  - Storage: 500 MB free space
  - Internet: Stable broadband connection
- **Interface:** Admin interacts with the system through a secure login portal.

### 3.5.4 *Communications interfaces*

The Aid-Hive system relies on standard communication protocols to interact between users, admin, and the backend system.

#### 1. Web Communication (HTTP/HTTPS)

- Users (donors and recipients) and admins access the system through standard web browsers using HTTP/HTTPS protocols.
- HTTPS ensures that all data transmitted between users and the server is encrypted and secure.

#### 2. API Communication

- The front-end communicates with the back-end via RESTful APIs implemented in Express.js.
- APIs handle requests such as form submissions, donor searches, admin actions, and live location updates.

#### 3. Database Communication



- The back-end communicates with the MySQL database using standard SQL queries for inserting, updating, retrieving, and deleting records.

#### **4. Location Services (Optional/Maps API)**

- The system integrates with GPS/Map services for live location tracking of donors and recipients.
- Coordinates are transmitted securely to the website interface for display.

## **2.6 Summary**

In this chapter, we studied how the Aid-Hive Blood Donation Management System interacts with its users and interfaces. Donors and recipients register exclusively through simple online forms, while only the admin has login access to manage records. The system relies on standard web communication (HTTP/HTTPS), a user-friendly front-end, and secure MySQL database connections. Identifying requirements through use cases ensured that each feature supports real user needs efficiently. This analysis guarantees that the system remains easy to use, reliable, and responsive, especially in urgent blood donation scenarios where timely access to donors is critical.



# **Chapter 3**

## **Design and Architecture**



### 3. System Design

The system is built on a MERN stack with a MySQL database, designed to manage blood donations efficiently. It consists of a React.js frontend that provides five pages—Home, Donate, Find Blood, Aid Hive Workers, and About Us—along with two key buttons for Admin access and live location tracking. Donors and recipients can only interact with the system by filling out registration forms, while the admin must log in to access and manage the full system features. The backend, built with Node.js and Express.js, handles all API requests, user authentication, and communication with the MySQL database, ensuring secure data storage and real-time updates, including live tracking of Aid Hive workers.

#### 3.1 Design considerations

The design of the system takes into account usability, security, and scalability. User interfaces are kept simple and intuitive so that donors and recipients can easily register and access information, while admins have a secure login to manage the system. The backend is modular, with separate API routes for donations, blood requests, and live location tracking, ensuring maintainability and clear separation of concerns. Using MySQL provides reliable and structured data storage, and the system is designed to handle concurrent users efficiently. Additionally, real-time features like live location tracking are incorporated to improve responsiveness and operational efficiency.

#### *Assumptions and Dependencies:*

Here are the Assumptions and Dependencies in points format:

##### Assumptions:

- Donors and recipients have basic internet access.
- Users can accurately fill out registration forms.
- Admins are authorized personnel and manage the system responsibly.
- Users understand how to navigate the website.

##### *Dependencies:*

- MERN stack for frontend (React) and backend (Node.js + Express).



- MySQL database for structured and reliable data storage.
- Stable server environment for backend operations.
- Secure API endpoints for data communication.
- Continuous network connectivity for real-time features like live location tracking.
- Accurate user input and timely admin updates to maintain data integrity.

### ***Limitations:***

- Donors and recipients can only interact with the system through registration forms; no advanced user features are available.
- Admin access is required for most operations, so system functionality is limited without admin intervention.
- Real-time live location tracking depends on stable internet connectivity and may be affected by network issues.
- The system does not support offline usage.
- Scalability may be limited by server capacity and database performance under high user load.
- Data accuracy depends on correct input from users and timely updates from admins.

### ***Risks:***

- **Data Security Risk:** Unauthorized access to user or donor information if authentication is compromised.
- **Network Dependence:** Real-time features like live location tracking may fail during network outages.
- **Data Accuracy Risk:** Incorrect or incomplete information submitted by users may affect system reliability.
- **System Downtime:** Server or database failures can make the system temporarily unavailable.
- **Scalability Risk:** High user traffic may slow down the system or overload the server.



- **Admin Dependency:** Most critical operations depend on timely admin actions; delays may affect system efficiency.

### 3.2 Design Models

The system design is modeled using component-based and layered architecture to clearly define the interaction between frontend, backend, and database.

#### 1. Component Model:

- **Frontend (React):** Handles user interface, pages (Home, Donate, Find Blood, Aid Hive Workers, About Us), buttons (Admin, Live Location), and forms (Donor/Recipient registration).
- **Backend (Node.js + Express):** Processes API requests, manages authentication, handles business logic, and communicates with the MySQL database.
- **Database (MySQL):** Stores structured data including users, donations, blood requests, and worker information.

#### 2. Interaction Model:

- Donors/Recipients: Registration → Search/Donate → System Response.
- Admin: Login → Manage Users/Donations/Workers → System Response.

#### 3. Data Flow Model:

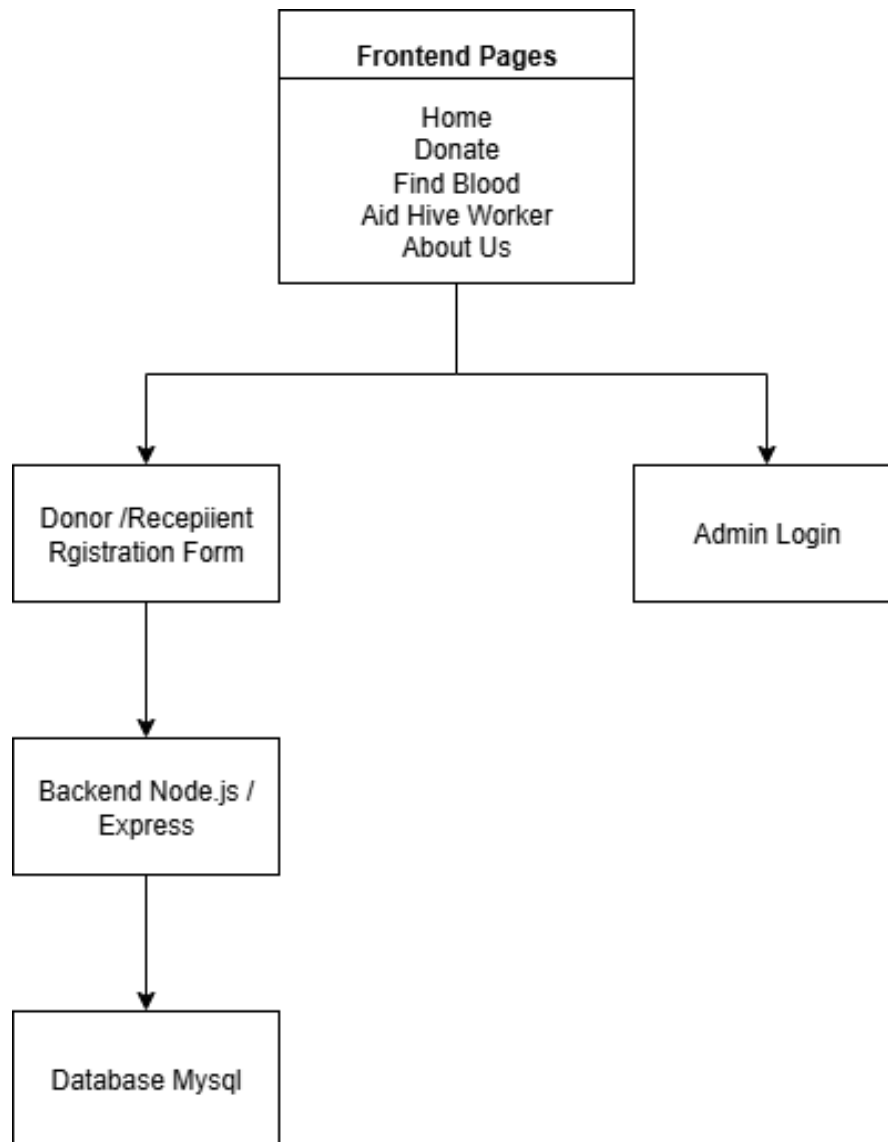
- Users submit forms on the frontend → Backend APIs process the requests → Data stored or retrieved from MySQL → Responses sent back to frontend.
- Admin actions (login, approvals, updates) follow the same flow with higher privileges.
- Live location tracking is updated in real-time via dedicated backend services.

This model ensures modularity, maintainability, and clear separation of concerns while supporting real-time tracking and secure database interactions.



### 3.2.1 Functional Component Diagram

Figure 2: Component Diagram







### 1. User Module

The User Module is designed to manage interactions for both donors and recipients through registration forms. Users can access services without the need for login, keeping the process simple and user-friendly.

- **Registration Form:**

- Donors and recipients provide personal details such as name, contact information, blood group, and location.

The form ensures all necessary information is collected to facilitate blood donation and requests.

- **Form Submission:**

- Once submitted, the backend processes the data and stores it securely in the MySQL database.

- **Interaction with System:**

- Donors can submit blood donation details.
- Recipients can request blood and view available options.
- Users cannot access administrative features; their interaction is limited to the forms and available search functionality.

- **Data Handling:**

- All user-submitted data is validated for accuracy and stored in the database for further processing by the admin.

This module emphasizes ease of use, secure data collection, and minimal user barriers, allowing donors and recipients to participate in the system quickly and efficiently.

### 2. Request Module

The Request Module handles blood requests from recipients and coordinates them with available donors. It ensures that blood requirements are efficiently captured and processed.



- **Request Form:**
  - Recipients fill out a form specifying their blood group, required quantity, and location.
  - Additional details such as urgency or special requirements can also be included.
- **Form Submission:**
  - The backend receives the request data and stores it in the MySQL database.
  - Data validation ensures that all required fields are completed correctly.
- **Processing Requests:**
  - Admin can view submitted requests and match them with available donors.
  - The system updates request status (e.g., pending, approved, completed) for tracking.
- **User Interaction:**
  - Recipients can submit multiple requests but cannot directly access donor information.
  - The module ensures that requests are processed securely and efficiently while keeping user interaction simple.

This module provides a structured and reliable way to manage blood requests while maintaining data integrity and supporting the overall blood donation workflow.

### 3. Admin Panel

The Admin Panel serves as the main control hub for managing the blood donation system. Access is restricted and requires login using a valid username and password. Only authorized admins can perform operations within the panel.

#### Key Features:

- **Login Authentication:**
  - Admin enters username and password to access the panel.

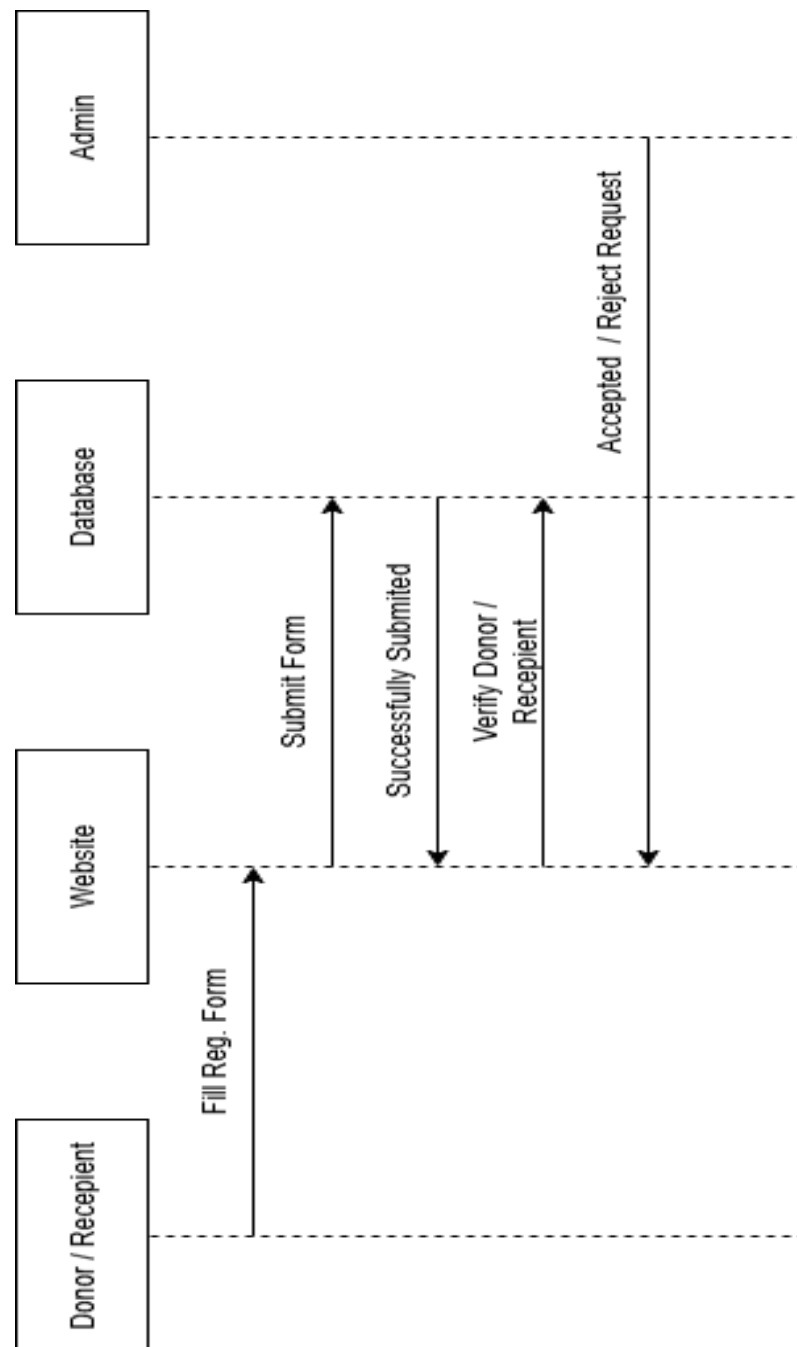


- Ensures that only authorized personnel can manage the system.
- **User Management:**
  - View donor and recipient details submitted via registration forms.
  - Approve, update, or remove user records as necessary.
- **Request Management:**
  - Monitor and process blood requests submitted by recipients.
  - Assign donors to fulfill requests and update request status (pending, approved, completed).
- **Donation Tracking:**
  - Maintain records of donations submitted by donors.
  - Ensure accurate and secure storage in the MySQL database.
- **Live Location Monitoring:**
  - Track real-time locations of Aid Hive workers to manage field operations efficiently.



### 3.2.2 Interaction Diagram

Figure 3: Interaction diagram of Aid Hive



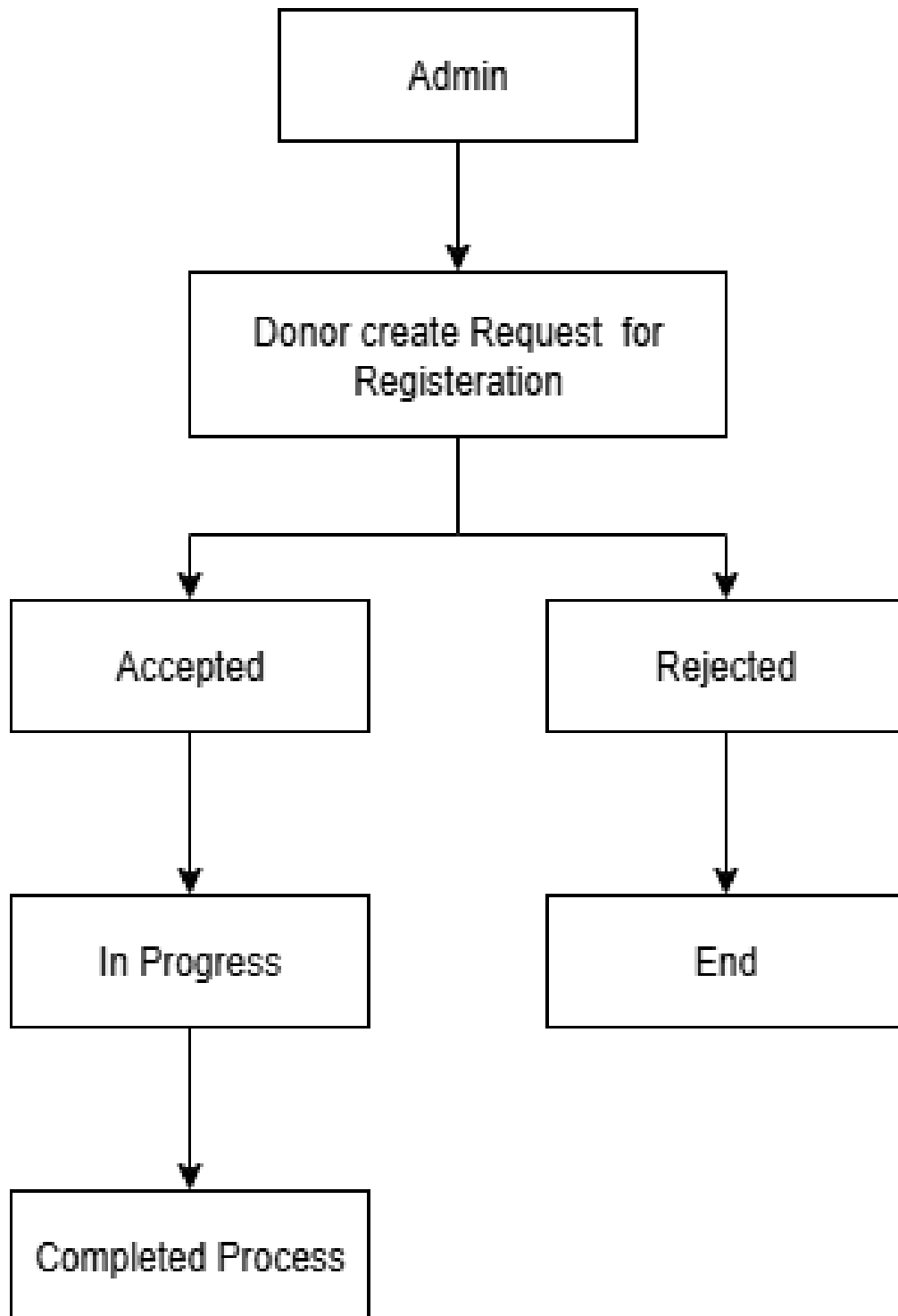


1. The user recipient/donor registers on the platform through the Frontend UI using the registration form.
2. Upon logging in, the recipient fills in the required details (blood type, urgency, and location) to request blood.
3. The request is submitted to the Request Module, which stores it in the MySQL database for processing.
4. The Donor Matching Module searches for suitable donors in the database based on blood group and city/location.
5. Once matching donors are found, the Notification Module (using Node mailer) sends confirmation emails and alerts to them.
6. Donors/Recipient can check requests via the platform interface; if available and willing, they accept the request.
7. The Google Maps API (v3) enables live location tracking to help recipients and donors connect efficiently.
8. The admin, through login access, monitors overall system activities (requests, donors, recipients).



### 3.2.3 Donor State Transition Diagram

Figure 4: Donor State Transition Diagram of Aid Hive





### **Explanation:**

#### **1. Admin (Initial State)**

The process begins with the admin logged into the system.

#### **2. Donor Creates Request for Registration**

A donor fills the registration form on the platform.

The request is forwarded to the admin for verification/approval.

#### **3. Decision State: Accepted or Rejected**

**If accepted** → the donor's details are stored in the MySQL database, and the donor can now participate in the system (donating/responding to blood requests).

**If rejected** → the process ends (the donor cannot access the system).

#### **4. In Progress (For Accepted Requests)**

Once accepted, the donor's status moves to In Progress, meaning they are now an active donor in the system.

#### **5. Completed Process**

When all verification steps are finalized, the donor registration is marked as Completed, and the donor is fully onboarded.

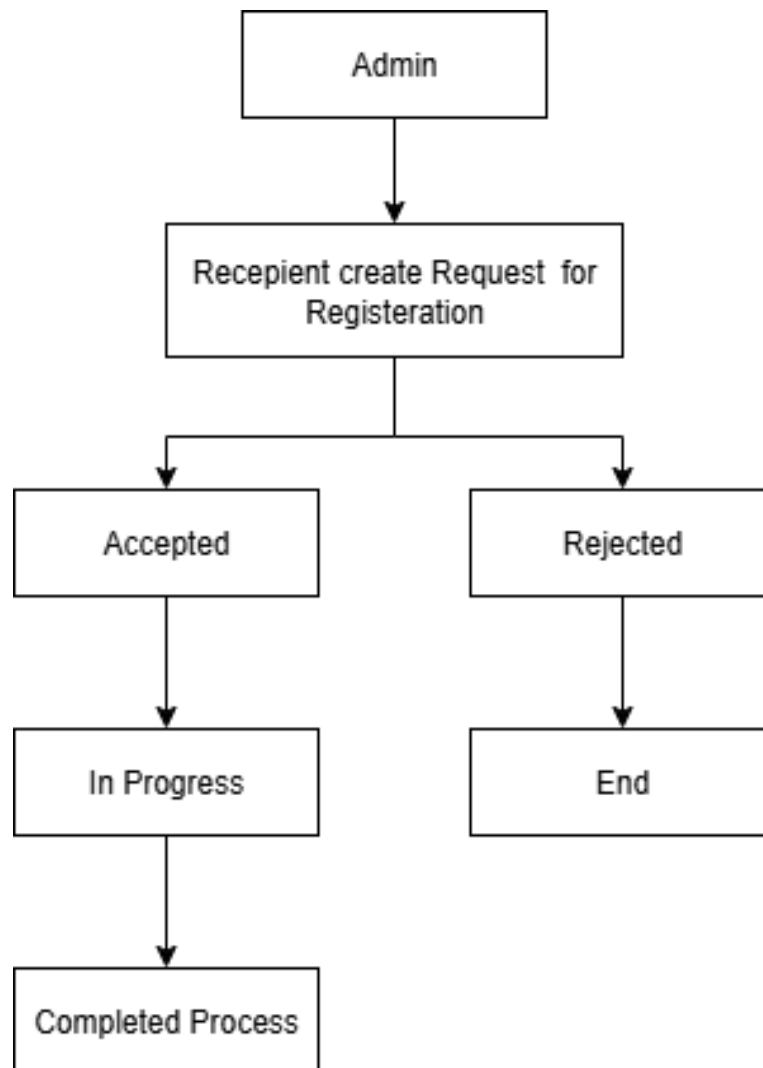
#### **6. End (For Rejected Requests)**

If the request is not valid, the process terminates without registration.



### 3.2.4 Recipient State Transition

Figure 5 : Recipient State Transition Diagram







### Explanation:

#### 1. Recipient (Start State)

A recipient (patient or person in need of blood) starts the process by accessing the system via the frontend UI.

#### 2. Aid Hive Worker

The request is forwarded to the Aid Hive Worker module, which handles filtering and management of requests.

#### 3. Listed Donor

The system retrieves and displays a list of donors from the MySQL database who match the required blood type and selected city.

#### 4. Check Status – Available or Not

The system checks the availability status of listed donors:

- **Available** → Donor can be contacted for donation.
- **Not Available** → Donor cannot respond at the moment.

### 3.3 Architectural Design

The system follows a three-tier architecture consisting of frontend, backend, and database layers to ensure modularity, maintainability, and clear separation of concerns.

#### 1. Frontend Layer (React.js):

- Provides the user interface for donors, recipients, and admins.
- Donors and recipients interact through registration forms and search functionality.
- Admin accesses feature only after login using username and password.
- Pages include Home, Donate, Find Blood, Aid Hive Workers, and About Us, with buttons for Admin and Live Location Tracking.

#### 2. Backend Layer (Node.js + Express.js):

- Handles all API requests from the frontend.
- Processes form submissions, user requests, and donation tracking.
- Performs authentication and authorization for the admin.
- Manages real-time live location updates of Aid Hive workers.



### 3. Database Layer (MySQL):

- Stores structured data including users (donors, recipients, admin), blood requests, donations, and worker information.
- Provides secure and reliable storage to support system operations.

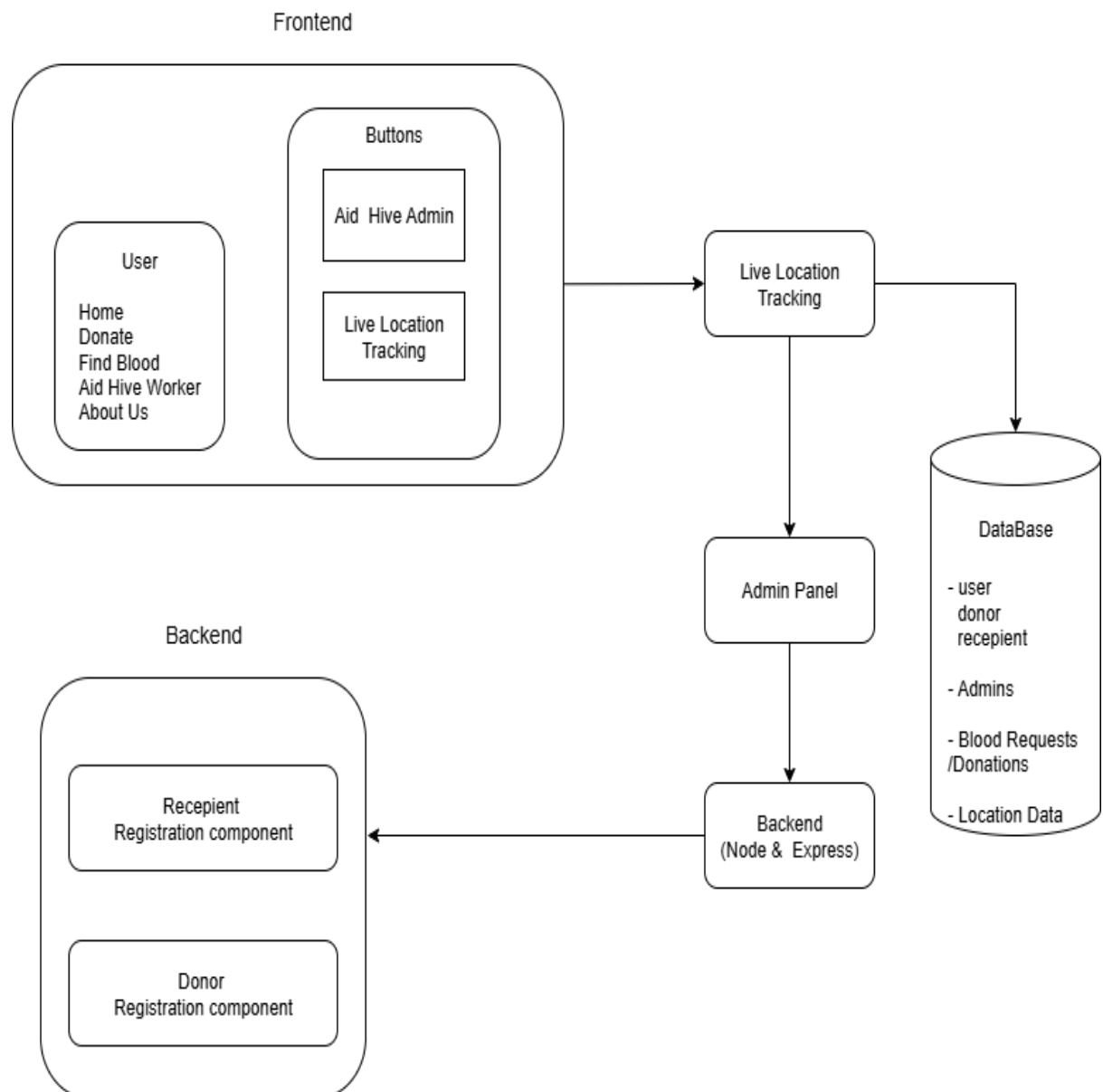
#### Data Flow:

- Users submit forms → Backend validates and stores data → Database updates → Admin monitors and manages requests.
- Live location data from workers is sent to the backend and updated in real time for monitoring.

This architecture ensures secure, efficient, and scalable operations while keeping user interactions simple and admin operations centralized.



Figure 6: Architectural Design of Aid Hive





### 3.4 Data Design

Aid Hive uses MongoDB, a NoSQL document-based database, to store system entities in collections (like tables in SQL). Main Collections in MongoDB

#### 1. Users Collection

Stores donor & recipient profiles (name, contact, blood type, location, etc.).

#### 2. Donor Collection

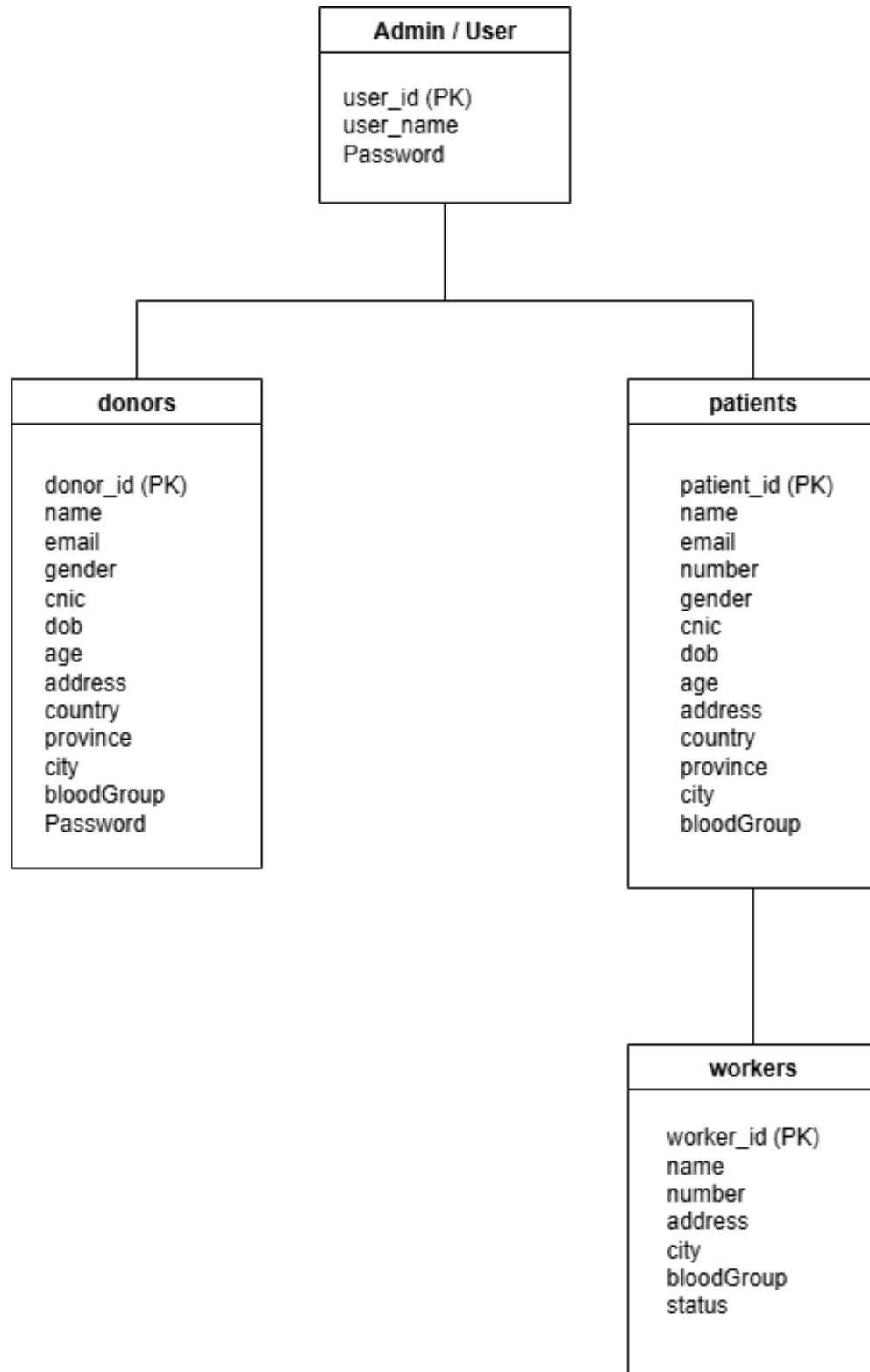
Keeps track of available donors, their donation history, and status.

#### 3. Recipient Collection

Stores blood request details, including urgency and location.

#### 4. Authentication Collection

Manages user credentials & security tokens for login.

*Figure 7 : ER Diagram*



### 3.4.1 Data Dictionary

Alphabetical List of System Entities or Major Data with Types and Descriptions:

Table 13: Data Dictionary Table

Terminology Description	Terminology Description
Blood Request	MySQL table representing a blood donation request
Blood Type	VARCHAR – Type of blood (A+, A-, B+, B-, O+, O-, AB+, AB-)
Donor	MySQL table representing a blood donor
Donor ID	INT (Primary Key, Auto Increment) – Unique identifier for each donor
Location	VARCHAR – Stores city/address details of donor/recipient (can also use DECIMAL for latitude/longitude if GPS is stored)
Recipient	My SQL Table representing a person requesting blood
Request Status	Enum - Status of blood request (Pending, Approved, Completed)
User	My SQL Table storing general user information



### Structured Approach: Functions and Function Parameters:

*Table 14 : Functions and Function Parameters*

API Endpoint	Description	HTTP Method	Parameters
/Api/donors	Retrieves available donors based on location & blood type	GET	(bloodType: String, location: Object)
/Api/requests/create	Creating a new blood donation request	POST	(recipientID: ObjectId, bloodType: String, location: Object)
/Api/requests/update	Updates request status	PUT	(requestID: ObjectId, status: String)

### 3.5 User Interface Design

The User Interface (UI) of the system is designed to be simple, intuitive, and user-friendly for donors, recipients, and admins. It ensures that users can perform their tasks efficiently while keeping navigation straightforward.

Key Pages and Features:

#### 1. Home Page:

- Displays system overview, purpose, and quick navigation links to other pages.
- Accessible to all users without login.

#### 2. Donate Page:



- Donors can fill out a donation registration form including name, blood group, contact, and donation details.
- Form submission sends data to the backend for storage.

### **3. Find Blood Page:**

- Recipients can submit blood request forms specifying blood group, quantity, and location.
- Requests are sent to the backend for processing by admin.

### **4. Aid Hive Workers Page:**

- Displays information about workers involved in blood donation and aid activities.
- Admin can monitor live location of workers.

### **5. About Us Page:**

- Provides general information about the platform and its objectives.

### **6. Buttons:**

- Aid Hive Admin Button: Redirects to login page for admin authentication.
- Live Location Tracking Button: Shows real-time location of workers on a map (accessible via admin panel).

### **7. Forms:**

- Donor and Recipient forms are simple and mandatory fields are validated to ensure complete and accurate data collection.
- No login is required for donors or recipients to interact with forms.

### **Design Principles:**

- Minimalist and clear layout to reduce confusion.
- Forms with proper labels and placeholders for easy data entry.
- Responsive design for accessibility on desktops, tablets, and mobile devices.





## Aid Hive

---

- Admin interface includes authentication, dashboards, and real-time monitoring for effective management.

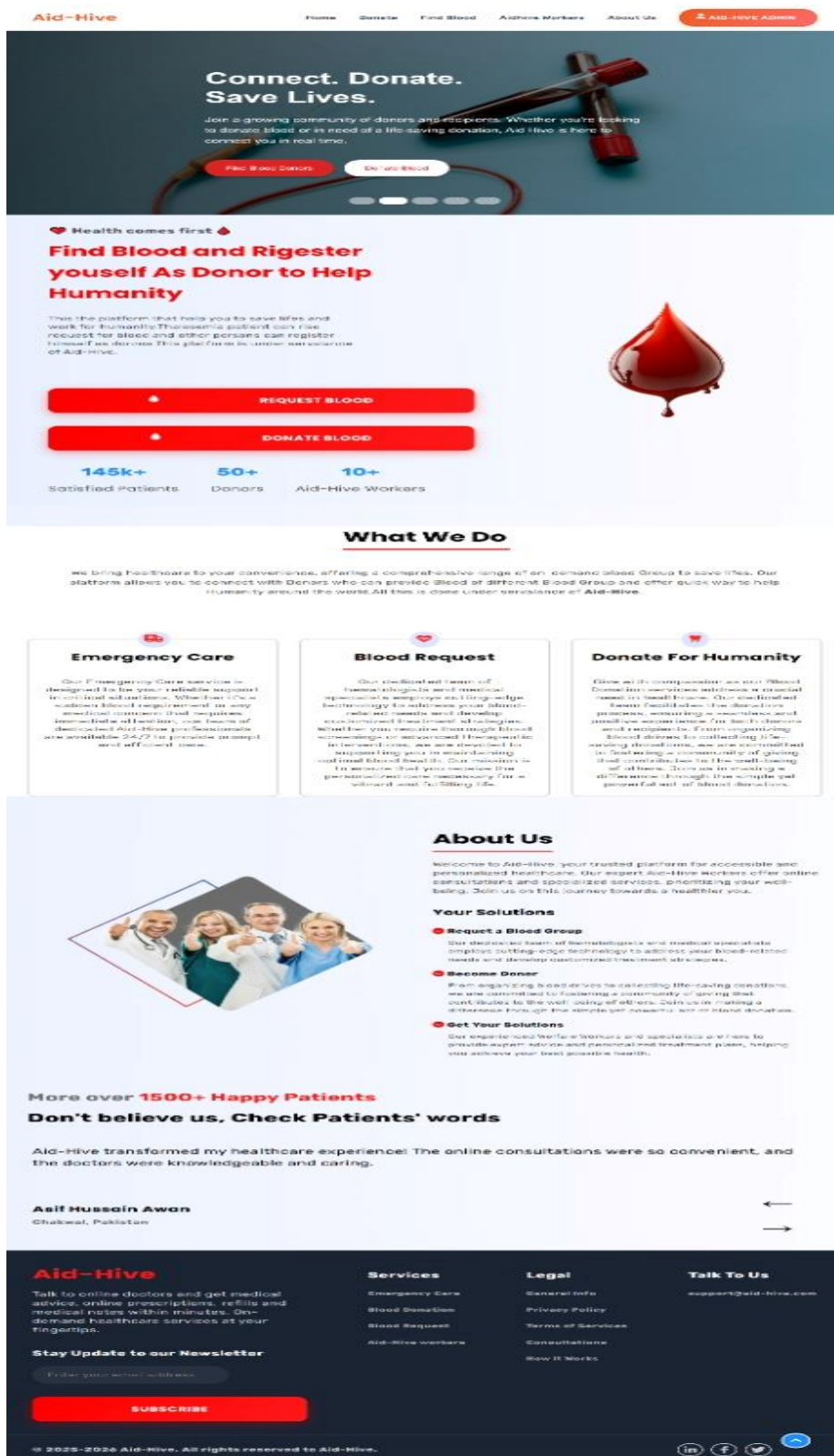
### ***3.5.1 Screen Images***

Following is the list of screen shots of UI mockups of Aid Hive illustrating the visual design of different components of the Aid Hive.

The subsequent images illustrate the landing page of the Aid Hive. That is the page that the user finds when he opens the app.



Figure 8 : Landing Page





The following image show case the Admin Login Form up of the Aid Hive.

*Figure 9 : Admin Login Form*

The screenshot displays the Admin Login Form on the Aid-Hive website. The page has a dark blue background with a blurred image of medical equipment. At the top left is the 'Aid-Hive' logo in orange. At the top right are navigation links: 'Home', 'Donate', 'Find Blood', and 'Aidhive Workers'. The login form is a semi-transparent white box in the center. It contains the following elements:

- Welcome Back**: A heading in bold black text.
- Username\***: A label for the first input field.
- : A text input field with a light blue border.
- Password\***: A label for the second input field.
- : A password input field with a light blue border.
- Login**: A blue button with white text.
- [Forgot Password?](#): A link below the login button.



The following image show case the Donor Registration Form up of the Aid Hive.

*Figure 10 : Donor Registration Form*

Home Donate Find Blood aidhive workers

### Blood Donation Form


\* Full Name:

\* Email:

\* Phone Number:

\* Gender:

\* CNIC Number

\* Date Of Birth:  
 

\* Age:

\* Address:



The following image show case the Recipient Registration Form up of the Aid Hive.

*Figure 11 : Recipient Registration Form*

The image shows a web form titled "Blood Request Form" with a red underline. The form is set against a light blue background with a purple border. It contains several input fields, each preceded by a red asterisk indicating a required field. The fields are: "Full Name:" with a placeholder "Enter your Name"; "Email:" with a placeholder "Enter your email"; "Phone Number:" with a placeholder "Enter your Phone Number"; "Gender:" with a dropdown menu showing "Select"; "CNIC Number" with a placeholder "Enter your CNIC"; "Date Of Birth:" with a placeholder "mm/dd/yyyy" and a calendar icon; "Age:" with a placeholder "Enter your age"; "Address:" with a placeholder "Enter your address"; and "Country:" which is partially visible at the bottom.



### 3.5.2 Screen Objects and Actions

#### *Use Case 1: Donor Registration*

*Table 15: Use case 1*

<b>Use Case ID</b>	UC-01
<b>Actor</b>	Donor
<b>Description</b>	Allows a donor to register and provide their details for blood donation.
<b>Preconditions</b>	Donor has internet access and opens the Donate page.
<b>Postconditions</b>	Donor's information is stored in the MySQL database.

#### **Main Flow:**

1. Donor navigates to the Donate page.
2. Donor fills out the registration form.
3. Donor submits the form.
4. Backend validates and stores the information in the database.
5. System displays a confirmation message.

#### **Alternate Flow:**

- If the form is incomplete or contains invalid data, the system shows an error message and prompts the donor to correct it.

**Use Case 2: Recipient Registration***Table 16: Use case 1*

<b>Use Case ID</b>	UC-02
<b>Actor</b>	Recipient
<b>Description</b>	Allows a Recipient to register and provide their details for blood donation.
<b>Preconditions</b>	Recipient has internet access and opens the find blood page.
<b>Postconditions</b>	Recipient's information is stored in the MySQL database.

**Main Flow:**

1. Recipient navigates to the find blood page.
2. Recipient fills out the registration form.
3. Recipient submits the form.
4. Backend validates and stores the information in the database.
5. System displays a confirmation message.

**Alternate Flow:**

- If the form is incomplete or contains invalid data, the system shows an error message and prompts the Recipient to correct it.

***Use Case 3: Admin Processing Blood Requests****Table 17: Use case 2*

<b>Use Case ID</b>	UC-02
<b>Actor</b>	Admin
<b>Description</b>	Allows the admin to view, manage, and approve blood requests submitted by recipients.
<b>Preconditions</b>	Admin must be logged in using username and password.
<b>Postconditions</b>	Blood request status is updated in the database and donor assignment is recorded.

**Main Flow:**

1. Admin logs in to the Admin Panel.
2. Admin navigates to the Request Management section.
3. Admin views pending blood requests submitted by recipients.
4. Admin assigns available donors to requests and updates status (approved/pending/completed).
5. Backend updates the database and sends confirmation to the recipient.

**Alternate Flow:**

- If no matching donors are available, the system keeps the request in pending status and notifies the admin.

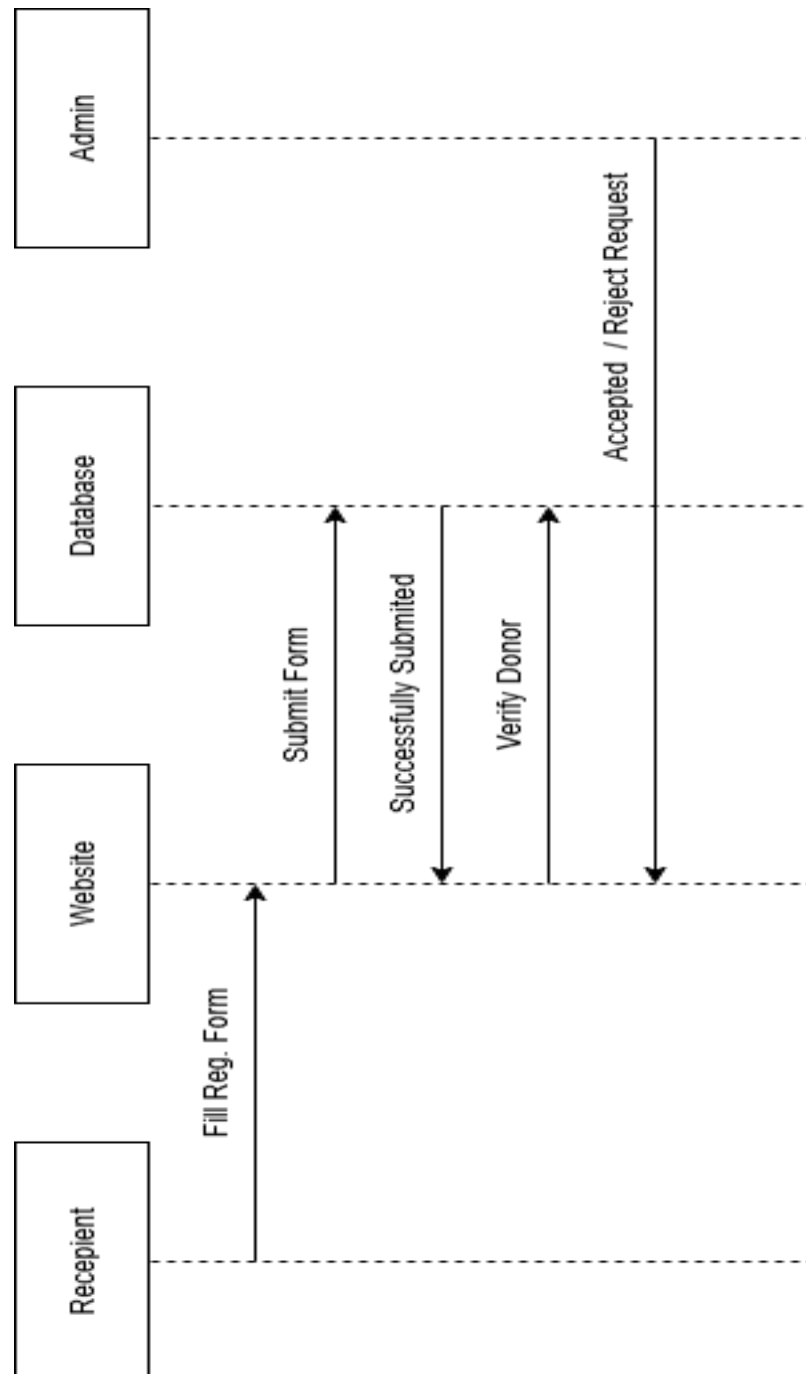


### 3.6 Behavioral Model

Interaction Diagrams:

#### 3.6.1 Recipient Registration

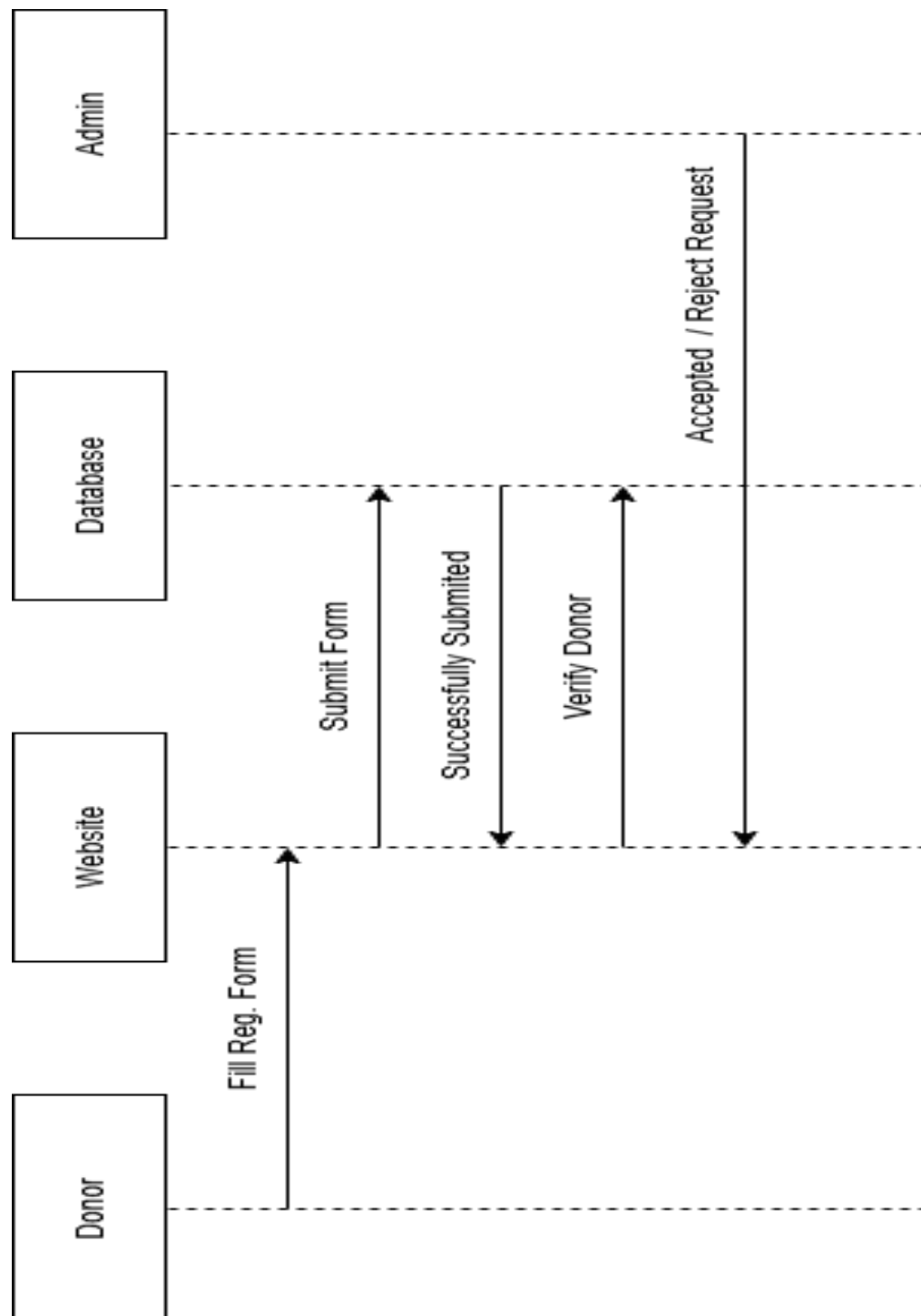
Figure 12: Recipient Registration





### 3.6.2 Donor Registration

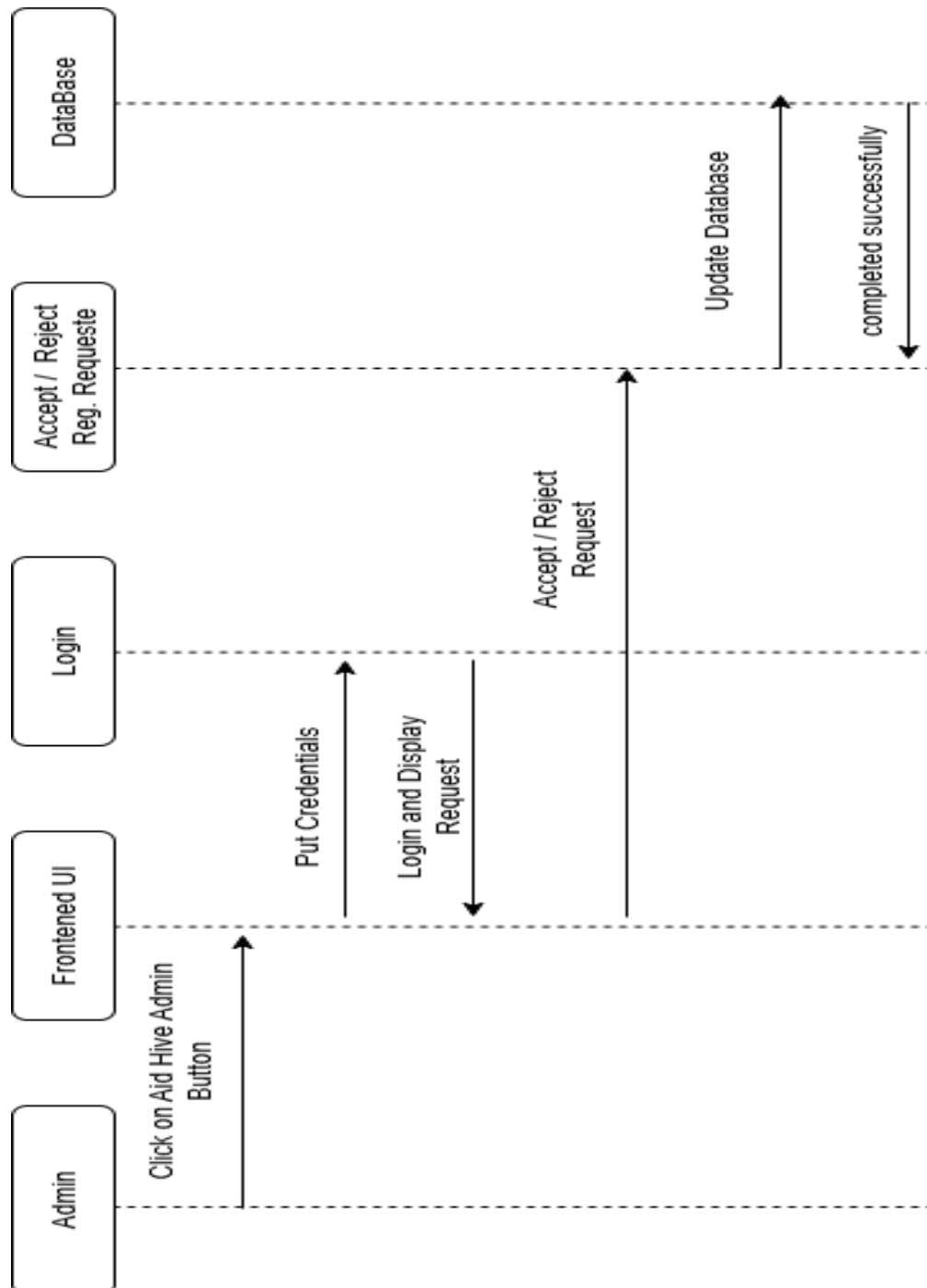
Figure 13: Donor Registration





### 3.6.3 Manage Profile

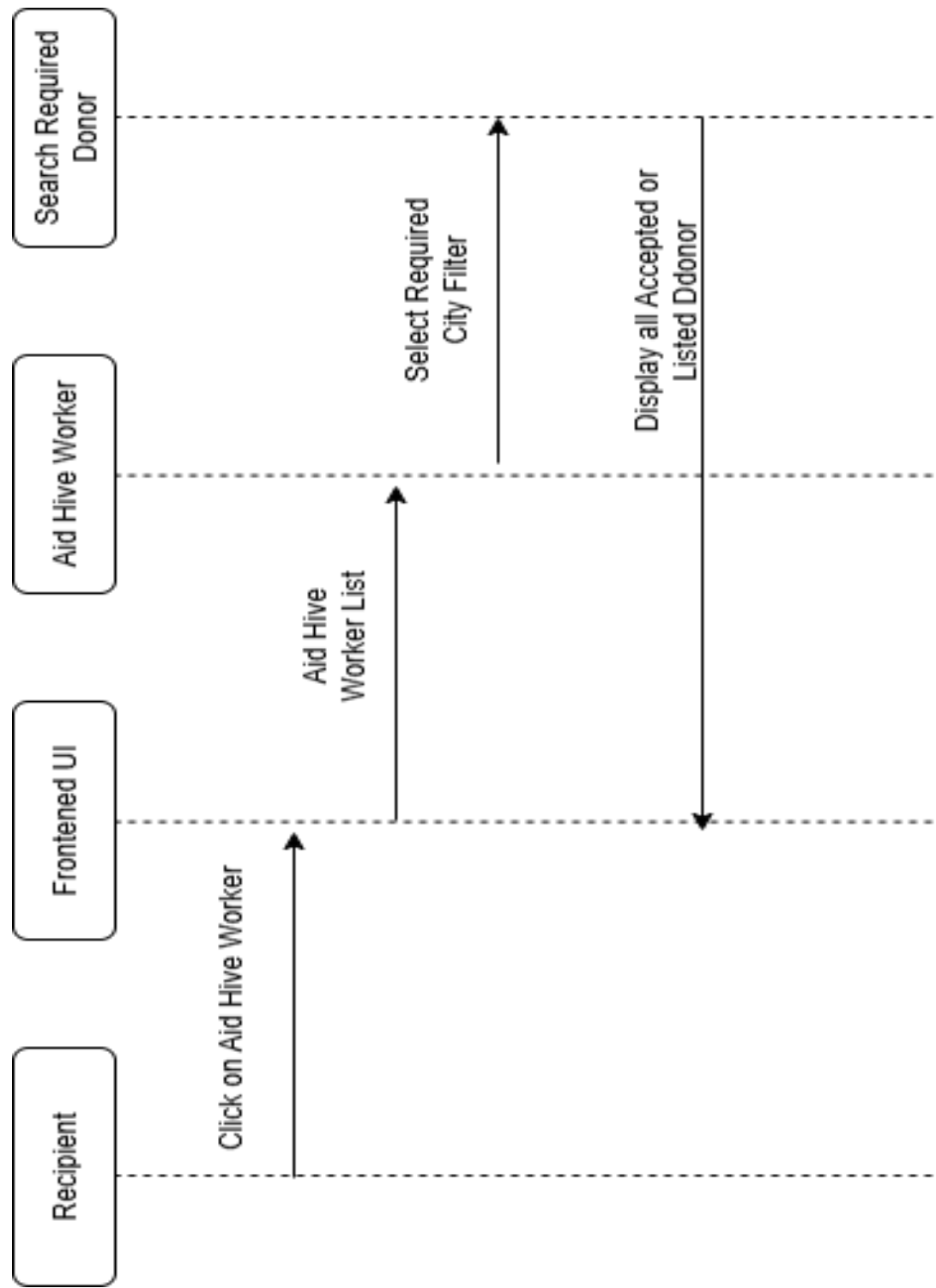
Figure 14: Manage Profile





3.6.4 Search for Donor

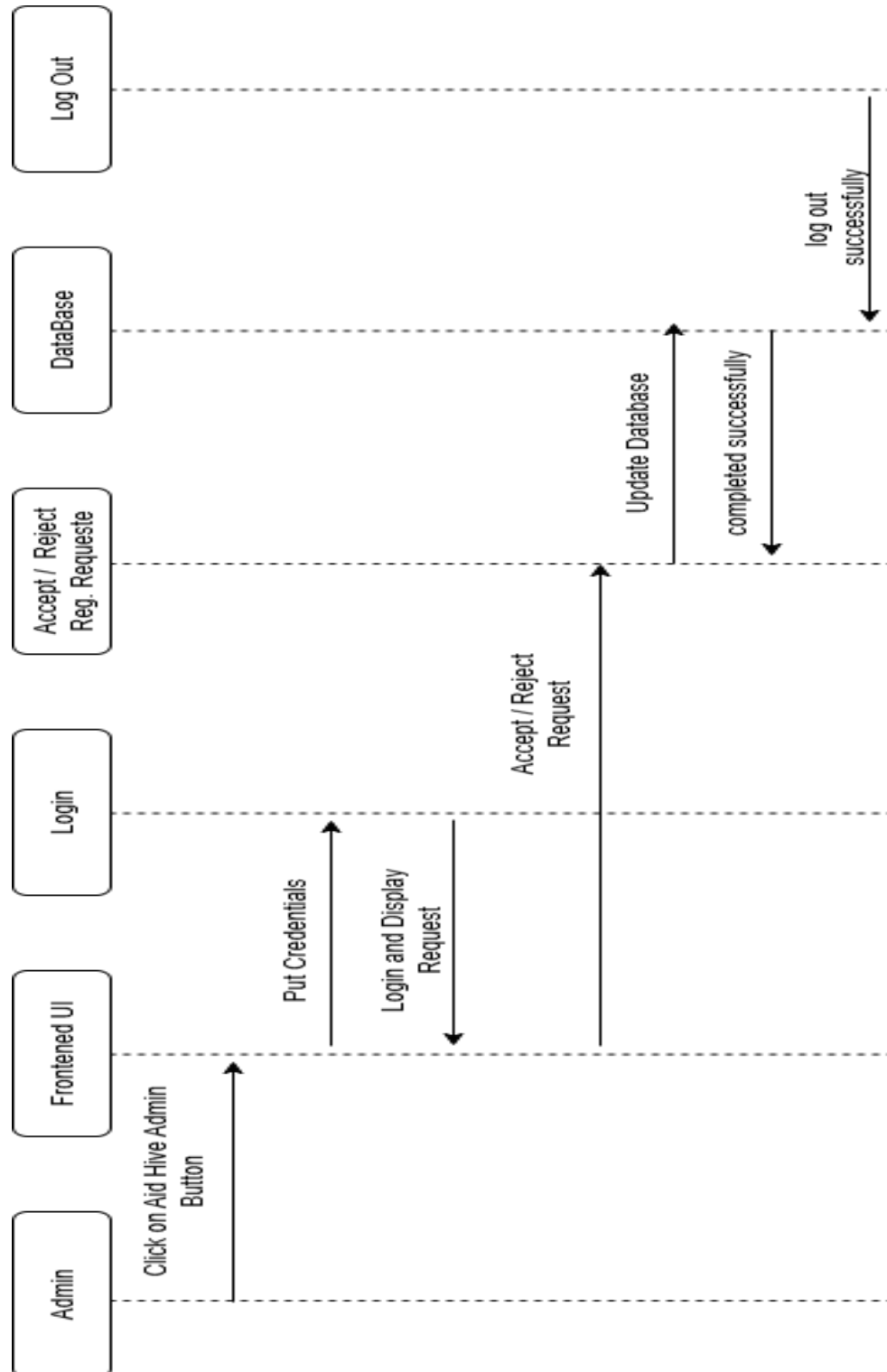
Figure 15: Search for Donor





### 3.6.5 Admin Panel

Figure 16: Admin Panel

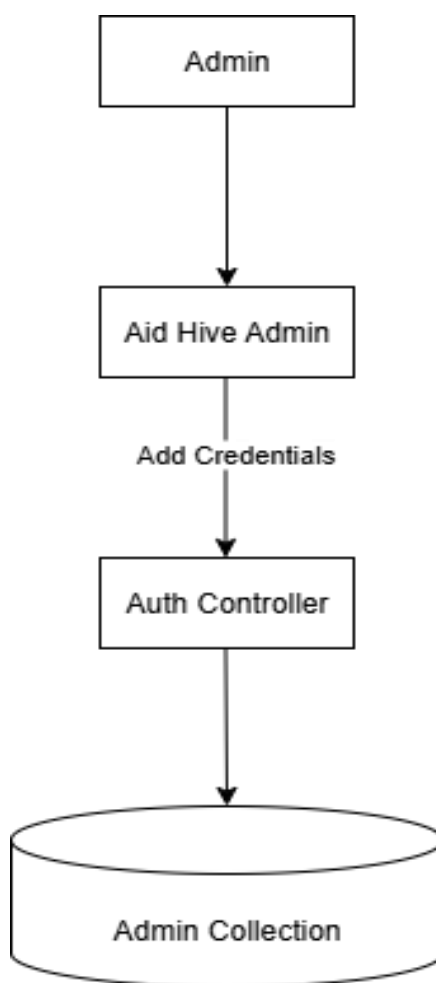




### Collaboration Diagrams

#### 3.6.6 *Login*

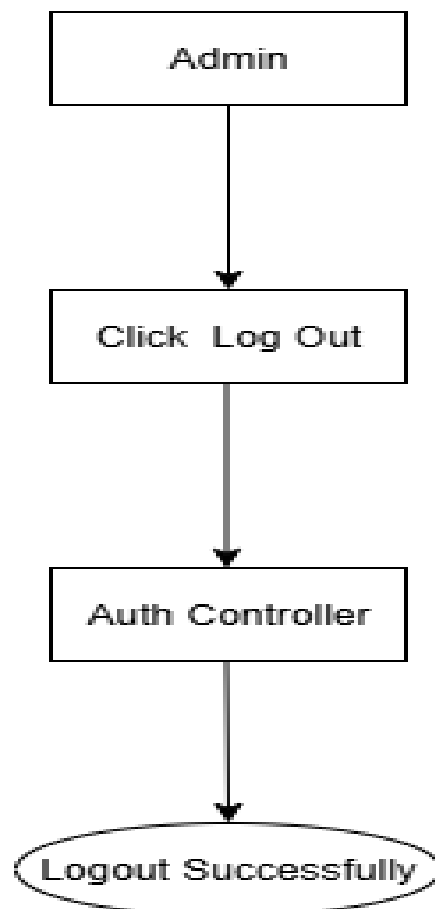
*Figure 17 :Admin Login*





### 3.6.7 Logout

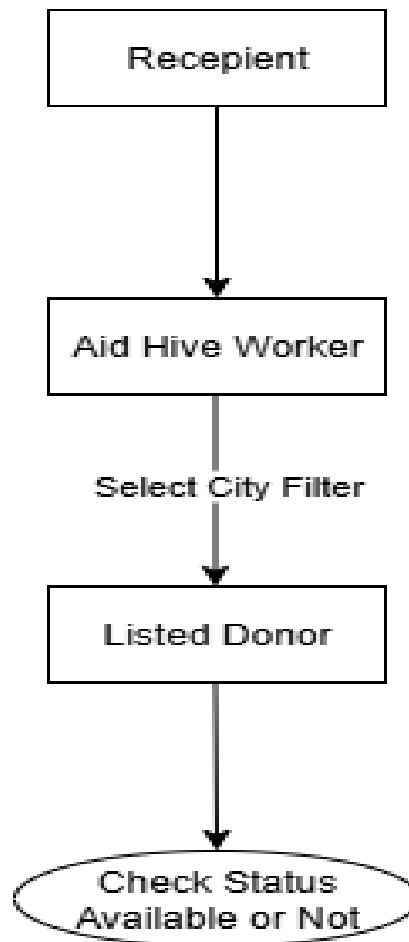
Figure 18: Admin Logout





### 3.6.8 Search for donor

Figure 19: Search for Donor



## 3.7 Design Decisions

The design decisions focus on simplicity, security, and proper role-based access:

### 1. MERN Stack with MySQL:

- Frontend: React.js
- Backend: Node.js + Express.js
- Database: MySQL for structured data storage





### 2. Form-Based Interaction for Users:

- Donors and recipients interact only through registration and blood request forms.
- No login is required for users to submit data.

### 3. Admin Login Authentication:

- Admin can access the system only via username and password.
- Ensures secure and authorized management of the system.

### 4. Three-Tier Architecture:

- Separation of frontend, backend, and database layers improves maintainability and scalability.

### 5. Live Location Tracking:

- Admin can monitor real-time locations of Aid Hive workers.
- Users do not have access to live location tracking.

### 6. User-Friendly Interface:

- Simple navigation and clear forms make donor and recipient interactions easy.
- Admin panel is designed for efficient management and monitoring.

### 7. Data Validation and Security:

- Form inputs are validated before submission.
- Sensitive data is secured through backend and database protocols.

## 3.8 Summary

The System Design of the Aid Hive platform focuses on creating a secure, efficient, and user-friendly system. Donors and recipients interact solely through registration and blood request forms, while admins access the system via username and password login. The system uses a MERN stack with MySQL for structured data storage and follows a three-tier architecture separating frontend, backend, and database layers. Features like live location tracking for Aid Hive workers are accessible only to admins, and all user inputs are validated to ensure data integrity. Overall, the design supports real-time coordination between donors and recipients while maintaining centralized control and operational efficiency.



# **Chapter 4**

## **Implementation**



## **4. Implementation**

### **4.1 Code Repository**

For version control, collaborative development, and efficient project management, the Aid Hive blood donation web application used Git as the main version control system. The repository included all source code, frontend and backend components, documentation, and related project resources. Using Git allowed the team to track changes, resolve conflicts, and collaborate effectively while working with the MERN stack. The repository served as a single source of truth, ensuring that all team members worked with the latest version of the system and enabling smooth coordination throughout the development process.

#### **Git Repository Link**

[https://github.com/Abdur-Rehman153/Aid-Hive.git.](https://github.com/Abdur-Rehman153/Aid-Hive.git)

<https://github.com/A-Hassan001/Aid-Hive.git>

<https://github.com/madiii-01/Aid-Hive.git>

<https://github.com/Maaz177-beep/Aid-Hive.git>



### 4.2 Summary

The Implementation phase of Aid Hive focused on developing a functional and secure blood donation portal using the MERN stack with MySQL as the database. The frontend was built with React.js and designed to be responsive and user-friendly, while the backend using Node.js and Express.js handled data operations, API routing, and admin functionalities. Core features such as donor registration, recipient blood requests. Admin access is secured via username and password login, and the system validates all form submissions to maintain data integrity. All modules were tested for performance, security, and compliance with system requirements, ensuring a reliable and efficient platform for connecting donors and recipients in real time.



# **Chapter 5**

## **Testing and Evaluation**



## **5. Introduction**

The Testing and Evaluation of Aid Hive ensures that the system operates smoothly and meets user expectations. During development, testing verifies that all functionalities from user registration and login to donor matching and blood requests perform correctly under various conditions.

While automated testing is preferred for faster validation, manual test cases are still applied for functions that require user interaction. All test cases are developed following best practices to ensure accuracy, reliability, and maintainability.

Key principles guiding the testing process include:

- Conducting independent testing that does not rely on other test results.
- Writing short and clear statements to validate expected outcomes.
- Using test data properly with setup and teardown methods.
- Avoiding hardcoded values to make tests adaptable.
- Ensuring tests are repeatable and reliable each time they are executed.

This approach ensures that Aid Hive is robust, reliable, and fully functional for both donors, recipients, and administrators.

### **5.1 Unit Testing (UT)**

Unit testing entails testing separate components or functions of the Aid Hive system to ensure that each component works as it is supposed too independently. These are done early in development to catch small issues before they become big issues.



### 5.1.1 Donor Registration

Table 16: User Registration

Testcase ID	UT-1
Requirement ID	FR-1 (Donor-Registration)
Title	Register New Donor
Description	Tests if a new donor can register successfully
Objective	Ensure system stores new donor data correctly
Driver/precondition	User is not already registered
Test steps	1. Open registration form 2. Enter valid info 3. Submit 4. Check database
Input	Ali Hassan, contact.abhassan@gmail.com, Gujranwala, Hafizabad Road Sharukh colony, B+, 1234
Expected Results	Form loads, data accepted, success message, new record exists
Actual Result	Form loads Data accepted Redirected with success User found in database
Remarks	Pass



### 5.1.2 Recipient Registration

Table 17: Recipient Registration

Testcase ID	UT-2
Requirement ID	FR-2 (Recipient Registration)
Title	Register New Recipient
Description	Tests if a new recipient can register successfully
Objective	Ensure system stores new recipient data correctly
Driver/precondition	User is not already registered
Test steps	<ol style="list-style-type: none"> <li>1. Open registration form</li> <li>2. Enter valid info</li> <li>3. Submit</li> <li>4. Check database</li> </ol>
Input	Sara Khan, sara.khan@gmail.com, Lahore, Gulberg, O-, 5678
Expected Results	Form loads, data accepted, success message, new record exists
Actual Result	Success + redirect Error shown Error shown
Remarks	Pass





### **5.1.3 Admin Login**

*Table 18: Admin Login*

Testcase ID	UT-3
Requirement ID	FR-2 (Admin Login)
Title	Admin Login
Description	Tests if admin can login with correct credentials
Objective	Ensure admin access works
Driver/precondition	Admin account exists
Test steps	1. Open admin login 2. Enter credentials 3. Submit
Input	admin@aidhive.com, password123
Expected Results	Login successful, dashboard opens
Actual Result	Successful login
Remarks	Pass



### 5.1.4 Find Blood

Table 19: Find Blood

Testcase ID	UT-4
Requirement ID	FR-4 (Find Blood)
Title	Search Blood Donors
Description	Tests if the system returns correct donors based on blood group & city
Objective	Ensure blood search works accurately
Driver/precondition	Donors exist in database
Test steps	1. Open Find Blood page 2. Enter blood group & city 3. Submit search
Input	Blood Group: B+, City: Gujranwala
Expected Results	List of matching donors displayed
Actual Result	Blood Matched
Remarks	Pass

## 5.2 Functional Testing (FT)

Functional testing is performed to ensure that all the important functionalities and user-centric modules of Aid Hive work as expected. The objective is to ensure that the system is according to the functional specifications and that the system can be used by all the stakeholders (Donor, Recipient, Admin) effectively.



### 5.2.1 Donor Registration

*Table 20: Donor Registration*

Testcase ID	FT-1
Requirement ID	FR-1 (Donor Registration)
Title	Verify Donor Registration
Description	Ensures a donor can register successfully with valid inputs
Objective	Confirm donor data is stored correctly
Driver/precondition	User is not already registered
Test steps	Open form → Enter valid info → Submit → Verify database
Input	Ali Hassan, contact.abhassan@gmail.com, Gujranwala, Hafizabad Road Sharukh colony, B+, 1234
Expected Result	Form submits, success message, new record created
Actual Result	Works as expected
Remarks	Pass

### 5.2.2 Recipient Registration

*Table 21: Recipient Registration*

Testcase ID	FT-2
Requirement ID	FR-3 (Recipient Registration)
Title	Verify Recipient Registration
Description	Ensures a recipient can register successfully with valid inputs
Objective	Confirm recipient data is stored correctly
Driver/precondition	User is not already registered
Test steps	Open form → Enter valid info → Submit → Verify database
Input	Sara Khan, sara.khan@gmail.com, Lahore, Gulberg, O-, 5678
Expected Result	Form submits, success message, new record created
Actual Result	Successfully registered
Remarks	Pass



### 5.2.3 Admin Login

Table 22: Admin login

Testcase ID	FT-3
Requirement ID	FR-4 (Admin Login)
Title	Verify Admin Login
Description	Ensures admin can login with correct credentials
Objective	Confirm admin access works properly
Driver/precondition	Admin account exists
Test steps	Open login → Enter credentials → Submit
Input	admin@aidhive.com, password123
Expected Result	Admin dashboard opens
Actual Result	Successfully Login
Remarks	Pass

### 5.2.4 Admin Panel Controls

Table 23: Admin Control

Testcase ID	FT-5
Requirement ID	FR-5 (Admin Dashboard)
Title	View Users and Deactivate Account
Description	Admin views users/requests and deactivates accounts
Objective	Ensure administrative functions perform correctly
Driver/precondition	Admin logged in
Test steps	Login → View → Deactivate
Input	User ID = 1098
Expected Result	User status changes to inactive
Actual Result	Successfully updated
Remarks	Pass



### 5.3 Integration Testing (IT)

Integration testing was done after all the separate unit tests were successfully finished. Integration testing consisted of testing interactions of sets of modules (e.g., request creation, donor matching, and notifications) to ensure that the overall flow of the features generates the desired outputs.

Every integration test verifies that the system's components are communicating effectively with each other, data is transferred between modules without any problems, and end-to-end processes provide results as intended from the end-user perspective.

#### 5.3.1 Registration → Request Submission Flow

Table 24: Test Case 1

Testcase ID	IT-1
Requirement ID	FR-1, FR-2
Title	User Registration to Request Flow
Description	Ensures system returns accurate donors based on blood group & city
Objective	Confirm blood search works correctly
Precondition	Donor exists in database
Test steps	Open Find Blood → Enter blood group & city → Submit
Input	Blood Group: B+, City: Gujranwala and further fields
Expected Result	Matching donors displayed
Actual Result	Works as expected
Remarks	Pass



### 5.3.2 Admin Action → User Login Blocked

Table 25: Test Case 2

Testcase ID	IT-2
Requirement ID	FR-1, FR-5
Title	Admin Deactivates User → Prevent Login
Description	Ensure admin block prevents user access via login
Objective	Validate integration of admin actions and authentication logic
Precondition	User exists; admin has access
Test steps	Admin deactivates → User tries login
Input	User ID = 1099
Expected Result	Login fails with “account disabled” message
Actual Result	As expected,
Remarks	Pass

**5.3.3 Donor Sets Availability → System Updates Matching Pool***Table 26: Test Case 3*

Testcase ID	IT-3
Requirement ID	FR-3, FR-4
Title	Donor Availability Status Integration
Description	Ensure system updates donor availability status and adjusts future matching
Objective	Confirm that availability flag reflects in donor matching
Precondition	Donor shows activeness in aid-hive workers
Test steps	Donor sets available → Recipient sends request
Input	Donor: ON status; Recipient: Blood Type = O+, Location = City Lahore
Expected Result	Donor appears in match results
Actual Result	Donor matched correctly
Remarks	Pass

**5.3.4 Admin Views Logs → Matches with User History***Table 27: Test Case 6*

Testcase ID	IT-6
Requirement ID	FR-5, FR-2
Title	Admin Review of User History
Description	Admin checks if user request logs match with system-wide records
Objective	Ensure transparency and data syncing between user and admin views
Precondition	Admin and recipient both logged in
Test steps	User submits → Admin reviews logs
Input	User ID = 3012, Request ID = 1204
Expected Result	Admin sees same request data as recipient
Actual Result	Matched perfectly
Remarks	Pass



## 5.4 Performance Testing (PT)

Following the testing of each module and integration, performance testing was done to determine the behavior of the system under stress, heavy load, and concurrent usage states. These were performed on response time, reliability, and stability.

### 5.4.1 Login API Response Time

Table 28: Test Case 1

Testcase ID	PT-1
Requirement ID	FR-1
Title	Login API Performance
Description	Test login response time with multiple simultaneous users
Objective	Ensure system handles authentication quickly under load
Precondition	4 users attempt login (valid & invalid credentials)
Test steps	Simulate login requests → Measure response time
Input	Email/password for 4 users
Expected Result	Response time < 1 second
Actual Result	~350ms average
Remarks	Pass





#### 5.4.2 Request Creation Under Load

Table 29: Test Case 2

<b>Testcase ID</b>	PT-2
<b>Requirement ID</b>	FR-2
<b>Title</b>	Request for Submission Load Performance
<b>Description</b>	Assess how request creation behaves under simultaneous usage
<b>Objective</b>	Ensure the system remains responsive with multiple users
<b>Driver/precondition</b>	100 recipients submit blood requests at once
<b>Test steps</b>	Simulate load → Submit requests → Record time
<b>Input</b>	Request payloads with blood type, location, urgency
<b>Expected Result</b>	< 2 seconds per request
<b>Actual Result</b>	~1.5 seconds
<b>Remarks</b>	Pass

#### 5.4.3 Admin Panel Performance

Table 30: Test Case 5

<b>Testcase ID</b>	PT-5
<b>Requirement ID</b>	FR-5
<b>Title</b>	Admin Dashboard Load Time
<b>Description</b>	Test how fast admin panel loads user and request data
<b>Objective</b>	Confirm UI responsiveness and database query performance
<b>Driver/precondition</b>	Admin logged in
<b>Test steps</b>	Access dashboard → View user & request tables
<b>Input</b>	Admin access to system database
<b>Expected Result</b>	Load time < 1.5 seconds
<b>Actual Result</b>	~910ms
<b>Remarks</b>	Pass



### 5.5 Summary

This chapter put the Aid Hive system through unit, functional, integration, and performance testing. Each individual module authentication, requests, donor matching, notifications, admin, and optional chat were tested in isolation and collectively. The system performed reliably, responded quickly under load, and met all functional requirements. These results confirm that Aid Hive is stable, efficient, and deployment-ready.



# **Chapter 6**

## **System Conversion**



## **6. Introduction**

The System Conversion process for Aid Hive transforms the traditional, manual blood donation procedures into a centralized, web-based system. Key operations, such as donor registration, blood request management, donor-recipient matching, and notifications, are migrated to a streamlined online platform, making the process faster, more efficient, and user-friendly.

During the conversion, all user data, roles, and workflows are seamlessly transferred to ensure continuity and accuracy. This chapter outlines the deployment of the system, adoption by stakeholders, and integration into real-world use cases, highlighting how the web-based Aid Hive platform replaces legacy methods and supports smooth, real-time coordination between donors, recipients, and administrators.

### **6.1 Conversion Method**

For the Aid Hive system, a direct conversion method is used. This means the manual and standalone processes of blood donation (such as paper-based donor lists, offline request handling, and manual matching) are replaced immediately by the new centralized web-based system.

The direct conversion approach was chosen because:

- The system is relatively new and not dependent on heavy legacy infrastructure.
- Real-time donor–recipient matching and notifications require an instant shift to avoid duplication or delays.
- It ensures all users (donors, recipients, and admin) interact with a single updated platform without relying on outdated methods.

Although direct conversion carries some risk, careful data migration, testing, and training for stakeholders minimizes errors and ensures a smooth transition to the Aid Hive platform.

#### ***Direct Conversion***

The Aid Hive system adopts a Direct Conversion method, where the existing manual blood donation process is completely replaced by the new web-based system in a single step. Once the system is deployed, all stakeholders—including donors, recipients, and administrators—will begin using the new platform exclusively, without relying on old manual methods.



This approach was selected because it provides:

- Immediate transition to a centralized system for blood donation.
- Elimination of duplicate processes, ensuring everyone uses one updated platform.
- Faster adoption by all users, as the old paper-based procedures are discontinued.

### ***Parallel Conversion***

The Aid Hive system uses a Parallel Conversion method, in which the new web-based platform runs side by side with the existing manual process for a limited period of time. During this phase, blood donation requests, donor registrations, and recipient matching are handled both manually and through the Aid Hive portal.

This method was chosen because it:

- Provides a safety net, as the old system continues to function until the new system is fully reliable.
- Helps detect and fix any technical issues in the Aid Hive system without interrupting ongoing blood donation activities.
- Allows users to gradually adapt to the new system, reducing training challenges and resistance.

### ***Pilot Conversion***

The Aid Hive system can also be deployed using a Pilot Conversion method, where the new web-based blood donation platform is first introduced to a small group of users or a specific region before being rolled out organization-wide. In this approach, a limited number of donors, recipients, and administrators are given access to the Aid Hive system to perform real-world tasks such as donor registration, blood request posting, and location-based donor searches.

### ***Suitability for Aid Hive***

For the Aid Hive system, the most suitable approach is a Pilot Conversion, where the platform is first introduced to a limited group of users such as a single hospital, a specific blood bank, or a small community of donors and recipients. In this pilot phase, the selected group will use the Aid Hive system for donor registration, blood requests, and live location-based matching, while the rest of the users continue with the manual process.



### ***Phased Conversion***

In the Phased Conversion approach, the Aid Hive system is implemented gradually in stages, rather than all at once. Specific modules such as Donor Registration, Blood Request Posting, Live Location Tracking, and Admin Panel are rolled out one after another, allowing each component to be tested and stabilized before moving on to the next.

### ***Suitability for Aid Hive***

The Phased Conversion method is suitable for Aid Hive because it allows the system to be introduced step by step instead of all at once. Since Aid Hive contains multiple critical modules such as Donor Registration, Blood Request Management, Live Location Tracking, and Admin Panel, rolling them out in phases ensures that each feature is properly tested, validated, and accepted by users before moving forward.

## **6.2 Summary**

The Aid Hive system was converted from a manual blood donation process into a centralized, web-based platform using a structured conversion strategy. Different methods such as Direct, Parallel, Pilot, and Phased Conversion were considered, with Phased Conversion identified as the most suitable due to its low risk, gradual rollout, and ease of user adoption. This ensured that modules like Donor Registration, Blood Requests, Live Location Tracking, and Admin Panel were introduced step by step, minimizing errors and disruptions. The conversion process successfully established a reliable, user-friendly, and efficient system, enabling real-time interaction between donors, recipients, and admins.



# **Chapter 7**

## **Conclusion**



## 7. Introduction

The Aid Hive Blood Donation Management System, built on the MERN stack with MySQL, streamlines blood donation through five pages Home, Donate, Find Blood, Aid Hive Workers, and About Us with two key features: admin panel and live location tracking. Donors and recipients only interact by filling registration forms, while data management and approvals are handled securely by the admin. This simple yet effective structure makes the system user-friendly, reliable, and well-organized for managing blood donation activities.

### 7.1 Evaluation

*Table 31 : Evaluation Table*

Objectives	Status
Allow donors to update their availability status directly from their dashboard.	Completed
Keep user data secure and private, complying with standard security practices.	Completed
Ensure the system is fully responsive and works well on desktops, laptops, and mobile devices.	Completed
Provide a clear and user-friendly interface using React.js and Tailwind CSS.	Completed
Offer an emergency mode to prioritize urgent blood donation requests.	Completed
Provide role-based dashboards for both donors and recipients with personalized features.	Completed
Store and manage donation and request history for all registered users.	Completed





## 7.2 Traceability Matrix

Following is a table of all the requirements for Aid Hive with their brief description and design specifications.

Table 32 : Requirement Traceability Matrix

Requirement ID	Requirement Description	Design Specification	Code	Test ID
R1	The system must allow users to store and retrieve blood donor/recipient profiles and request data.	Component: <i>User Authentication, Profile Management</i>	auth.js, user.js, profileRoutes.js	T01
R2	The system must recommend blood donors based on location and blood type.	Component: <i>AI-Powered Recommendation Engine</i>	matchEngine.js, donorFilter.js	T02
R3	The system must track the status of blood requests (pending, fulfilled, etc.).	Component: <i>Request Status Tracking</i>	requestStatus.js, requestRoutes.js	T04
R4	The system must allow users to view their donation history and track past activities.	Component: <i>Donation History Management</i>	history.js, userDashboard.js	T05
R5	The system must provide a user-friendly interface for donation and request submission.	Component: <i>User Interface Design</i>	Form Component.js, requestForm.css	T06
R6	History management for tracking past donations.	Design: <i>Database Logging (My sql)</i>	mysql.js, donationLogs.js	T07



### 7.3 Conclusion

In conclusion, Aid Hive provides a simple yet effective platform for connecting blood donors and recipients, while keeping the system secure and manageable through the admin panel. Donors and recipients interact only by filling registration forms, and the admin oversees all records and approvals through a secure login. This structured approach ensures user-friendliness, reliability, and proper organization of blood donation activities, making the system a practical solution for managing the donation process.

#### *Correlation Table: Objectives vs. Functionality*

Table 33 : Correlation Table

Defined Objective	Functionality Provided
Location-based donor recommendations	Google Maps API integration with geolocation-based donor filtering
Easy access and usability across all devices	Fully responsive frontend using React.js and Tailwind CSS
Track and manage blood donation history	Profile management with past donation forms stored in My SQL
Simplified blood request process	Structured request forms with validation and user-friendly submission interface

### 7.4 Future Work

Even though the Aid Hive system has fulfilled its original objectives, there is always potential for future growth and expansion. Among the proposed advancements are:

- **Mobile Application Development:**

To improve accessibility and convenience for users while they are on the go, a cross-platform mobile application version will be developed.



- **AI-Based Donor Matching:**

This method uses computer programs with artificial intelligence to suggest the best donors based on urgency and historical data.

- **Multilingual Support:**

Providing multilingual support will make the platform more accessible to users from various geographical locations.

- **Hospital Integration:**

Enabling real-time donor data viewing and mass donation campaign management for hospital accounts.

- **Offline Alert System:**

In places with poor internet connectivity, use SMS-based alert systems.

## References

MySQL Documentation. Retrieved from

<https://www.mysql.com/docs/manual/>

React. (n.d.). React.js Documentation. Retrieved from

<https://react.dev/>

Tailwind CSS. (n.d.). Tailwind CSS Documentation. Retrieved from

<https://tailwindcss.com/>

Express. (n.d.). Express.js Documentation. Retrieved from

<https://expressjs.com/>

Node.js. (n.d.). Node.js Documentation. Retrieved from

<https://nodejs.org/docs/latest/api/documentation.html>

Google. "OAuth 2.0 for Web Applications." Internet:

<https://developers.google.com/identity/protocols/oauth2/>



# **Appendix A**

## **Use case Description Template**



## Appendix-A Use Case Description (Fully Dressed Format)

The Table A-1 below indicates a comprehensive use case template

Table 34 : Table A- 1 Show the detail use case template and example

<b>Use Case ID</b>	UC-1
<b>Use Case Name</b>	Register as a Blood Donor
<b>Actors</b>	Donor (User), System (Aid Hive Web App)
<b>Description</b>	This use case allows a new user to register as a blood donor by filling in their personal information, blood type, and availability. Once registered, they can be matched with recipients.
<b>Trigger</b>	The donor clicks on the "Register" button on the homepage or navbar.
<b>Preconditions</b>	The donor must have internet access and a valid email address.
<b>Postconditions</b>	The donor's data is saved in MySQL, and they receive a confirmation email via Node mailer. They are redirected to the donor dashboard.
<b>Normal Flow</b>	Donor navigates to the "Donor Registration" page. System displays the registration form. Donor enters full name, email, password, blood type, location, and availability. Donor clicks the "Submit" button. Frontend (React) performs validation on all fields (e.g., required fields, email format, password strength) If valid, form data is sent via Axios to the Express.js backend. Backend hashes the password using crypt, stores the data in My SQL. System sends a confirmation email using Node mailer. System responds with a success message and redirects to the dashboard.
<b>Alternative Flows</b>	<p>AF-1: <i>(At Step 3)</i> If the user enters an already registered email address, the system displays an error message: "Email already exists."</p> <p>AF-2: <i>(At Step 5)</i> If any field is invalid (e.g., empty or wrong format), the system highlights the error and disables submission.</p>
<b>Business Rules</b>	Only users 18 years and above can register. Blood type must be selected from a fixed dropdown list. Donors must confirm availability status (Yes/No). Email format must match RFC 5322 standards. Password must have at least 8 characters including uppercase, lowercase, and numbers. Backend checks for existing accounts to prevent duplicates.