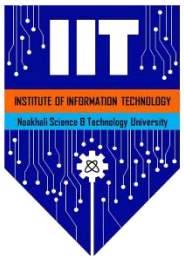
Institute of Information Technology, Noakhali Science and Technology University

**Bachelor of Science in Software Engineering**

**Course Code: SE 4100**




# Final Year Project Report

## Submitted by

*Ikra Chowdhury Nowkshi*

BFH1925019F

## Submitted to

*Final Year Project Committee*

Institute of Information Technology (IIT)
Noakhali Science and Technology University

**Internship Attended:** July 23, 2023 – January 22, 2024

**Date of Submission:** December 02, 2023

# LETTER OF TRANSMITTAL

# DEDICATION

d

# APPROVAL

This Internship report submitted by **Md. Armanur Rashid, ID No: *ASH1925013M*** to the Chairman of Internship Placement Office, Institute of Information Technology (IIT), Noakhali Science and Technology University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approved as to its style and contents. The presentation has been held on December 02, 2023.

## Internship Examination Committee:

**Dr. Md. Nuruzzaman Bhuiyan**

*Committee Member*

Assistant Professor
Institute of Information Technology (IIT)
Noakhali Science and Technology University

**Md. Auhidur Rahman**

*Committee Member*

Assistant Professor
Institute of Information Technology (IIT)
Noakhali Science and Technology University

**Dipok Chandra Das**

*Committee Member*

Assistant Professor
Institute of Information Technology (IIT)
Noakhali Science and Technology University

**Md. Iftekharul Alam Efat**

*Committee Member*

Assistant Professor
Institute of Information Technology (IIT)
Noakhali Science and Technology University

**Dr. Md. Kamal Uddin**

*External Member*

Associate Professor
Department of Computer Science and Telecommunication Engineering
Noakhali Science and Technology University

**Dr. Mohammad Salim Hossain**

*Chairman*

Director
Institute of Information Technology (IIT)
Noakhali Science and Technology University

# Acknowledgement

# Executive Summary

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# Chapter One: Introduction

## 1.1 Project Overview

Around a third of all the food produced globally each year is lost or wasted and is valued at almost $1 trillion dollars worldwide. This percentage is in the range of 25-30% of food every year that goes to waste just in Bangladesh. Food for Everyone is a web application which will connect entities with food surplus to those in need, like welfare organizations or even individual people. It is a user-friendly platform that helps near you to connected local and reduces food waste by food donation.

The process starts with users signing up and verifying their information to ensure the system is secure and trustworthy. Organization and rider accounts need admin approval to be created. After creating an account, a donor can then make a post for the type of food they have, including details such as how much and where it can be picked up, as well as attaching an image of the food. Donors can edit or cancel that post. People looking for food can browse these posts and request donations. After receiving requests from the donees, the donor can accept or reject them on the Request Page. If they accept the food request and select the delivery option, the system will automatically find a rider and send a request based on location and availability. Both donors and receivers are free to schedule mutually convenient pickup times or have the food delivered if desired. Donors can also donate money directly or donate food directly to the organization.

When the donee logs into their account, the dashboard will show available donation posts based on their location. In the food cart option, the donee's requested food donation posts will be displayed, and if the donee wishes, they can cancel the request. There is another option called "On Run Post," where donation requests accepted by the donor will be shown. The donee's history will be displayed on the history page.

There is another feature, a search facility. If a user wants to search for any post, it will show the available donated food for a particular area based on the user's location. The user can change their location from their profile page. There is another function, notifications, which will work for all users. Notifications are sent to keep users informed about updates related to their posts, such as when a food request is accepted or when food is ready for pickup. There is a chat feature. Donors, donees, and riders can all communicate with one another via our chat facility. They can contact each other before proceeding with a request. There is a rating system where users can rate the food they have received from the donor. This will help identify fraudulent donors. Based on the quality of the food, the donee will rate it. If donors provide rotten food, the donee can give a lower rating.

On the rider page, if the rider turns on the "available" button, the system will send the rider requests based on their location. The rider can accept or reject the request. If they reject it, the system will automatically search for another rider who is available within that particular area. Rider can see the location of the pickup point and the delivery point.

The main goal of this project is to create a platform to make a culture of sharing food and help the others. This platform will help to reduce the food waste and will help the needy people to get food.

## 1.2 Project Purpose

The main purpose of this project is to create a platform where donors can post about their extra food so that needy people can access it. This project will establish a connection between donors and those in need. In Bangladesh, a significant number of people struggle for food while others waste it. A large amount of food is wasted every day. The aim of this project is to address this issue by connecting food donors with needy individuals.

The platform is designed to help individuals and other organizations to easily donate their extra food. Donors can post the food they have available and those in need or welfare community groups might request it. The idea is to make the process of sharing food as simple as possible so that food that would normally go to waste can be given to someone who needs it.

To ensure a smooth running, the platform will include other features among which user verification, so only genuine users can participate, and a feedback system, where users can rate their experiences. This will help in to maintain the quality and ensure that people sharing food should be passing only good food material.

The project also seeks to educate about the notion of how breaking bread can benefit your local community. The ability to easily donate and receive food is hoped to foster a culture of helping others, reduction in uneaten waste and increased improvisation in life.

In the end, we hope "Food for Everyone" can lend a hand by putting some of this technology to social good, enabling kids to hack on both food access and hunger in their communities.

### 1.2.1 Background

Bangladesh is a country facing significant challenges with poverty and food insecurity. In 2022, 18.7% of the population lived below the national poverty line, and the extreme poverty rate is expected to rise to 5.1% in the coming year. While many people in Bangladesh struggle to get enough to eat, a large amount of food is still wasted. About 25-30% of the food produced in the country is thrown away each year, whether it’s from homes, restaurants, markets, or other sources. This situation creates a serious problem: food that could feed those in need is being wasted instead.

**"Food for Everyone"** was created as a response to this issue. The idea behind the project is to use technology to connect people who have extra food with those who need it. In this way, food that would otherwise be wasted can be shared with those who are hungry. With the

increasing use of the internet and smartphones in Bangladesh, it has become easier for people to connect with one another, and this project takes advantage of that.

### 1.2.2 Benefits & Beneficiaries

**Benefits:**

1. Reduces Food Waste: The project assists reduce thousands of tons of food that are wasted in this region by linking the donors to the users. Instead of discarding leftovers in food, processors, and distributors can pass them on to others in need of food.

2. Helps Fight Hunger: The users of the platform are people who have limited access to food and any individual who doesn't have his meal can get it for free. This definitely serves to tame hunger as it has determined excess food gets to be fed to those most deserving to be fed.

3. Protects the Environment: There is the benefit of reducing food waste, and therefore protecting the environment, hence effectively and efficiently executing the project. Less waste is produced and there is not so much food wasted ending up in the garbage disposal or requiring many resources required for food that has been dumped.

4. Builds a Sense of Community: The project used in practice is designed to mobilize people, coordinate them to share and help each other, thus increasing cohesion within communities. Giving back to society is the core message that is conveyed by this institution.

5. Convenient for Users: The platform is easy to use and efficiently helps people to find food and donate it. Elements such as searchable lists of donors, notifications about the received money, and the possibility to rate help to optimize the process for both, donors and receivers.

**Beneficiaries:**

1. Individuals Facing Food Insecurity: The needy that are unable to afford nutrient rich food will benefit by being availed direct, actual good quality donated extras.

2. Welfare Organizations and NGOs: These groups will ascertain the availability of regular source of food donations, increasing their capacities to help more needy people and families within our communities.
3. Food Donors: It is also easier for independent voters, restaurants, and supermarkets, for instance, to donate excess food which they would otherwise have let go to waste.
4. Local Communities: People will be empowered to share, and there will be strong bonding between families and friends since they are allowed to exchange food Articles in this course will also help reduce food wastage hence promoting a healthy environment.

### 1.2.3 Goals

The goals of the project are:

1. Prevent food waste
2. Create a platform connecting donors and needy people
3. Help the poor people to get food
4. Foster a culture of food donation
5. Showing technology can solve real-world problems

By addressing these goals, "Food for Everyone" assumes that it can significantly affect the food insecurity and environmental sustainability.

## 1.3 Stakeholders

Stakeholder is a person or group of people who are connected with the system. Stakeholder of this project are:

- **Donors**: Individuals, and organization that have surplus food to distribute to the needy. They are important in ensuring that they avail themselves the food requirements that serves the receivers needs.
- **Donee**: Individuals or welfare organizations who rely on food donations to manage food for their people. These stakeholders ensure that the food reaches the right people.
- **Rider**: Individuals offering their time and expertise to help operate and promote the platform would be stakeholders. Their contributions are vital for the day-to-day functioning and outreach efforts of the project. They can deliver the food as cost free or by taking little charge.
- **Admin**: These are individuals or teams responsible for managing and overseeing the day-to-day operations of the website. They ensure that the platform functions smoothly, handle user inquiries and support, monitor transactions, and address any technical issues that arise.

To achieve this, the project targets to involve these stakeholders to create an active network in order to fight both food waste and food insecurity in an effective and sustainable manner.

## 1.4 Scope and Limitations

**Scope:** The "Food for Everyone" project is aimed at developing a website for food sharing where food lenders can cooperate with those who need help. It will be formatted in a way that enables different classes of users; the individuals–sign up and verify their ID to post the available. It will allow users to look up food in certain locations, communicate with the donors and arrange collection/pick up. This Application will include rating systems where

users will rate and comment on various donations that have been made to make the page more credible and accountable, the users will be notified of all the activities made in relation to donation.

**Limitations:** While the project offers significant benefits, it also has some limitations:

1. **Dependence on Active User Participation**: This is because the functionality of the platform depends greatly on the continuous engagement of both 'donors' and 'receivers. If there are not enough food postings or active users the platform may not attain its goal.
2. **Limited Coverage Area**: First of all, the platform can be local, meaning that the number of supplied meals is relatively small in territories which have a weak connection and where food insecure people can be seen.
3. **Food Quality and Safety**: Despite giving the users of the platform a chance to give feedback of the quality of food being donated, the safety and freshness of food being donated still remain an issue. This policy may make it difficult for the platform to monitor and therefore regulate the quality of food being exchanged.
4. **Technology Access**: This essentially means that the platform is open to users who have Internet connection and own digital device that they can use, which in the current economy may not be available to some people such as low-income earners.

## 1.5 Significance of the Project

The "Food for Everyone" project is significant for several reasons:

1. **Reducing Food Waste**: This project directly addresses the growing problem of food waste, worldwide, by offering a platform through which excess food can be donated for consumption, instead of being trashed. Bypassing the garbage disposal, which contributes to producing greenhouse gases every time foodstuffs decay in the landfill, the project contributes to feeding those in need.
2. **Supporting Food Security**: In the upcoming days, the platform will help meet food the scarcity, particularly in the developing nations such as Bangladesh where a large number of populations suffers from nutritive meal. The organization arranges for surplus food to be delivered to welfare organizations or individuals who are in dire need through the project.
3. **Social Impact**: Over and above the varying benefits accrued to food redistribution, the project promotes togetherness in issue solving. 'Food for Everyone' helps people to open up and become active participants in their community, making donations and helping neighbors. The project has the potential to support the development of an

organizational culture that respects food as people's nourishment instead of throwing it away.

4. **Environmental Benefits**: This project has extreme environmental importance, and cannot be overstated. They stated that food production needs water, energy and human labor before it can be produced. In this way, the project reduces the waste of these materials, thereby lessening positive effects the food production sector has on the environment.

5. **Technology for Good**: The project shows how a concept in technology can be applied to address real life issues in the distribution of food without wastage. It plays the role of demonstrating possibilities of using digital platforms for solving key social and environmental challenges in stimulating such solutions in other industries.

All in all, the "Food for Everyone" initiative is a move toward the minimal management of food as well as a socially responsible means of addressing food waste and inequality.

# Chapter Two: Project Planning and Management

## 2.1 Project Charter and Goals

**Project Title:** Food for Everyone

**Project Start Date**: 11, February 2024

**Project End Date**: 25, September 2024

**Project Purpose**: To develop a web application that connects individuals and organizations with surplus food to those in need in Bangladesh, aiming to reduce food waste and alleviate hunger.

**Project Scope:**

- Create a user-friendly web platform for food donations.
- Facilitate user registration and verification.
- Enable food posting and searching within specific geographic areas.
- Implement communication features for donors and receivers.
- Include a feedback and rating system for quality assurance.

**Project Deliverables:**

- Software Requirements Specification (SRS)
- Database Design and Implementation
- Project Documentation
- Source Code

**Project Goal:**

- Reduce Food Waste
- Enhance Food Security
- Foster Community Engagement
- Create a platform connecting donors and needy people
- Help the poor people to get food

## 2.2 Project Timeline and Milestones

A project management timeline is a detailed schedule for the project. what should be done, whereas actions specify how it should be done, and outcomes keep track of how to determine whether it was done successfully.

### 2.2.1 Gnatt Chart

*Table 1: Project Duration*

| Task | Duration |
|---|---|
| Project Proposal & Summary | 3 weeks |
| Project Analysis | 3 weeks |
| SRS Documentation | 4 weeks |
| UI Design | 7 weeks |
| Database Design | 2 weeks |
| Integration | 8 weeks |
| Testing | 3 weeks |
| Demonstration & reporting | 2 weeks |


*Figure 1: Project Timeline*

### 2.2.2 Release Plan / Milestone

*Table 2: Release Plan*

| Phase | Deliverables | Purpose |
|---|---|---|
| System Requirement and Analysis | Requirement Gathering and analysis.<br>Functional Specification<br>Non-functional Specification | It gives an exact understanding of the user's requirements. |
| System Design | Use case Diagram<br>Activity Diagram<br>Data Flow Diagram<br>Sequence Diagram | It gives the logical structure that describes the system. |
| Implementation and Testing | The output obtained for the required functionality after implementing and doing various types of testing. | It makes the system robust and reliable. |

## 2.3 Resource Allocation and Budgeting

To make the "Food for Everyone" project work resource will be distributed across employees, equipment, promotion, and services.

**Human Resources:**

1. **Project Manager**: Monitors project and/or team so that time line and objectives are achieved.
2. **Developers**: Web application development which consists of frontend and backend development is the main responsibility.
3. **UI/UX Designer**: Defines the look and feel of the application, such as the layout and ease of use of the platform.
4. **Quality Assurance (QA) Tester**: Undertakes testing to determine these bugs and invariably check the overall working of the application.
5. **Marketing Specialist**: Creates marketing strategies to penetrate and market the platform.
6. **Community Outreach Coordinator**: Recruiting involves directly interacting with the local organizations and users so that they can participate.

**Technology:**

1. Web Development Tools: Tools and frameworks required to build an application.
2. Hosting Services: The assured web hosting services for the platform.
3. Database Management: Data storage and data management solutions.
4. Collaboration Tools: Official business communication platforms for team members.

*Table 3: Project Budget*

| Item | Estimated Cost |
|---|---|
| **Human Resources** | |
| Project Manager | xxxx |
| Developers | xxxx |
| UI/UX Designer | xxxx |
| QA Tester | xxxx |
| Marketing Specialist | xxxx |
| Community Outreach Coordinator | xxxx |
| **Technical Resources** | |
| Web Development Tools | xxxx |
| Hosting Services | xxxx |
| Database Management | xxxx |
| Collaboration Tools | xxxx |
| **Total Cost** | xxxx |

## 2.4 Risk Assessment and Mitigation Strategies

Identifying potential risks and developing mitigation strategies is crucial for the success of the "Food for Everyone" project. Below are the key risks, their potential impact, and strategies to mitigate them.

**1. Technical Risks**

- **Risk**: Software bugs and technical issues during development.
- **Impact**: Delays in project timelines and

- **Mitigation Strategies**: Implement a robust potential loss of user trust. Quality assurance (QA) process, including regular testing throughout development.

**2. User Engagement Risks**

- **Risk**: Low user participation from donors or receivers.
- **Impact**: Insufficient food donations and ineffective platform usage.
- **Mitigation Strategies**: Engage local communities and organizations through outreach programs to encourage sign-ups.

3. **Legal and Compliance Risks**

- **Risk**: Regulatory challenges related to food safety and donation laws.
- **Impact**: Legal issues that could halt operations or lead to penalties.
- **Mitigation Strategies**: Research local laws and regulations regarding food donations to ensure compliance.

## 2.5 Proposed System Model

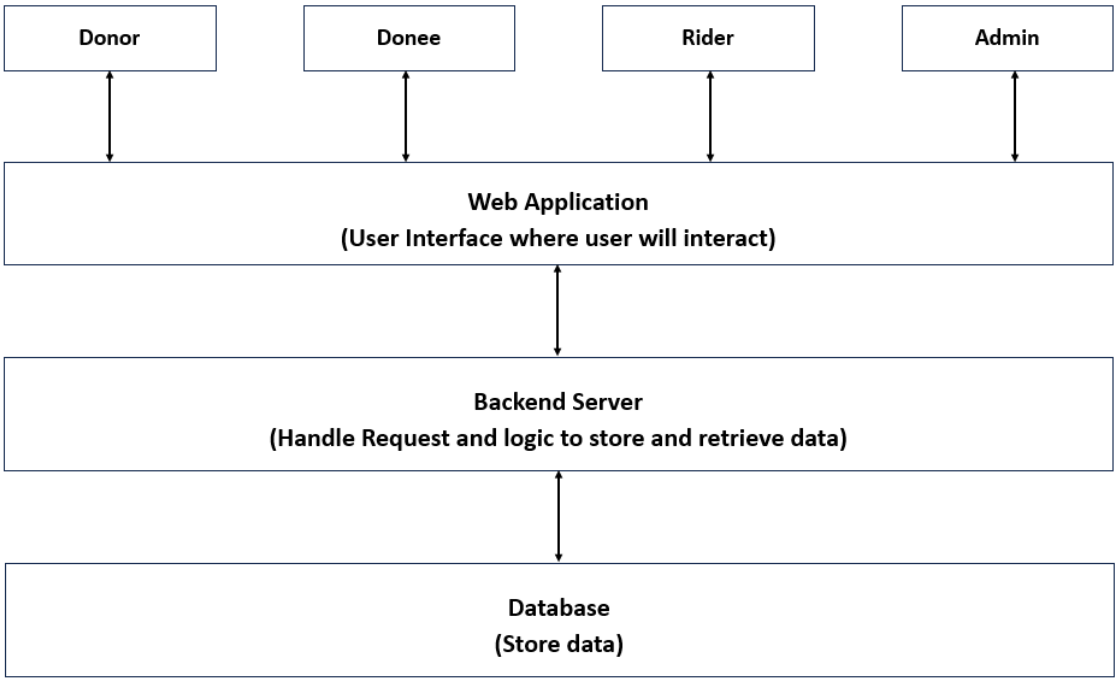



*Figure 2: Proposed System Model*

# Chapter Three: Software Requirement Specification

## 3.1 Functional Requirements

Functional requirement are those requirements that are used to illustrate the internal working nature of the system, the description of the system, and explanation of each subsystem. It consists of what task the system should perform. the processes involved, which data the system should holdand the interface with the user

*Table 4: User login and register*

| | | | |
|---|---|---|---|
| **FR-1** | User Registration and Login to a registered account | | |
| **Description** | User should register his/her account for the first time and be able to log in to the account which was registered once. Already registered users will not face this stage. | | |
| **Stakeholders** | Donors, Recipients, Volunteer | **Priority** | High |

*Table 5: Create new Food Donation Post*

| | | | |
|---|---|---|---|
| **FR-2** | Donor can create a new food donation post | | |
| **Description** | Registered users can easily post details of the surplus food they have available for donation. They can include information such as quantity, type of food, expiration date (if applicable), and pickup/delivery options | | |
| **Stakeholders** | Donor | **Priority** | High |

*Table 6: Edit Food Donation Post*

| | | | |
|---|---|---|---|
| **FR-3** | Edit Food Donation Post | | |
| **Description** | After the donor posts from his account, the donor can edit his/her post. | | |
| **Stakeholders** | Donor | **Priority** | Medium |

*Table 7: Delete Post*

| | | | |
|---|---|---|---|
| **FR-4** | Donor can delete his/her food donation post | | |
| **Description** | After the donor posts from his account, the donor can permanently delete his/her post. | | |
| **Stakeholders** | Donor | **Priority** | Low |

*Table 8: Quick Search*

| | | | |
|---|---|---|---|
| **FR-5** | Search post according to user preference | | |
| **Description** | A search option allows users to find food donations easily within a specified range or area. | | |
| **Stakeholders** | Donor, Recipients, Volunteer | **Priority** | Medium |

*Table 9: Rating System*

| | | | |
|---|---|---|---|
| **FR-6** | Rating the food | | |
| **Description** | Receivers can leave feedback through ratings. Donors providing poor-quality food may receive negative reviews. | | |
| **Stakeholders** | Recipients | **Priority** | Low |

*Table 10: Pickup and Delivery Service*

| | | | |
|---|---|---|---|
| **FR-7** | Pickup and delivery service for food | | |
| **Description** | Users can organize pickups or deliveries for the donated food based on mutual agreement. Donors and receivers can communicate through chat to arrange pickup details and timing. | | |
| **Stakeholders** | Donor, Recipients, Volunteer | **Priority** | high |

*Table 11: Notification*

| | | | |
|---|---|---|---|
| **FR-8** | Notification system for important event | | |
| **Description** | Recipients can enable notifications to receive alerts when someone wants to donate food within their specified range or area. Donor and volunteer can also enable notification system. | | |
| **Stakeholders** | Donor, Recipients | **Priority** | Medium |

*Table 12: Change Password*

| | | | |
|---|---|---|---|
| **FR-9** | Change user password | | |
| **Description** | Users can change their account password easily to maintain security and privacy | | |
| **Stakeholders** | Donor, Recipients, Volunteer, Admin | **Priority** | High |

*Table 13: Forget Password*

| | | | |
|---|---|---|---|
| **FR-10** | Forget Password | | |
| **Description** | In case users forget their password, this feature enables them to reset it and regain access to their account. | | |
| **Stakeholders** | Donor, Recipients, Volunteer, Admin | **Priority** | High |

*Table 14: Edit Profile*

| | | | |
|---|---|---|---|
| **FR-11** | Edit user profile | | |
| **Description** | Users have the flexibility to modify their profile information like name, contact details, and other information to keep their profile up to date. | | |
| **Stakeholders** | Donor, Recipients, Volunteer, Admin | **Priority** | Low |

*Table 15: Set Location*

| | | | |
|---|---|---|---|
| **FR-12** | Set location | | |
| **Description** | By setting location, users can customize their search for food donations or donors nearby, making it easier to find and donate food within their area | | |
| **Stakeholders** | Donor, Recipients, Volunteer | **Priority** | High |

*Table 16: Message*

| | | | |
|---|---|---|---|
| **FR-13** | Message | | |
| **Description** | This feature facilitates communication between users, allowing them to coordinate food donations, ask questions, or arrange pick-ups and deliveries effectively. | | |
| **Stakeholders** | Donor, Recipients, Volunteer | **Priority** | High |

*Table 17: Direct Donation*

| | | | |
|---|---|---|---|
| **FR-14** | Direct donation to the organization | | |
| **Description** | Users can contribute food directly to welfare organizations, ensuring that surplus food reaches those in need efficiently and directly supports charitable causes. | | |
| **Stakeholders** | Donor | **Priority** | Low |

*Table 18: Request Food Donation*

| | | | |
|---|---|---|---|
| **FR-15** | Request Food Donation | | |
| **Description** | Users can submit requests for food donations from available donors, indicating their needs and preferences. | | |
| **Stakeholders** | Recipients | **Priority** | High |

*Table 19: Cancel Request*

| | | | |
|---|---|---|---|
| **FR-16** | Cancel food donation request | | |
| **Description** | If users no longer require the food donation they requested, they can cancel their request to notify the donor. | | |
| **Stakeholders** | Recipients | **Priority** | High |

*Table 20: Accept Riding Request*

| | | | |
|---|---|---|---|
| **FR-17** | Accept riding request | | |
| **Description** | Users who wish to have donated food delivered to them; system automatically request a rider through the platform. Riders, registered on the platform, can view and accept these requests to facilitate the delivery of donated food to those in need. | | |
| **Stakeholders** | Volunteer | **Priority** | Medium |

*Table 21: Cancel donation*

| | | | |
|---|---|---|---|
| **FR-18** | Cancel donation | | |
| **Description** | If a donor wants, he/she can easily cancel the donation, notifying the recipient and avoiding any unnecessary arrangements | | |
| **Stakeholders** | Donor | **Priority** | High |

*Table 22: Admin Control*

| | | | |
|---|---|---|---|
| **FR-19** | Admin Control | | |
| **Description** | There will be admin panel, they can control everything after logging in admin panel. All action will be taken by admin. Admin can control and manage the whole system. | | |
| **Stakeholders** | Admin | **Priority** | High |

*Table 23: Log out*

| | | | |
|---|---|---|---|
| **FR-20** | User log out from their account | | |
| **Description** | The user will be able to log out of his/her account at the end of his need. Users will need to log in again for later use. | | |
| **Stakeholders** | Donor, Recipients, Volunteer, Admin | **Priority** | Medium |

## 3.2 Data Requirements

Data requirements refer to the specific data that is necessary for the project to function properly and meet its objectives. These requirements define the types of data that the software system needs to handle, process, store, and present in order to fulfill its intended functionality.

Data requirements for a software project typically include:

1. **Input Data**: The data that is required as input to the software system for it to perform its functions. Input data will store when user create account or will post donation about food.

2. **Output Data**: The data that the software system generates as output. Output data will come from database and will show in different pages as per user need.

3. **Data Structures**: The structure and format of the data required by the software system. This includes defining the data fields, data types, and relationships between different data elements. While user post the donation that data need to satisfied the required data format. And also, while uploading image it needs to matched the extension of the image.

4. **Data Sources**: Identifying the specific sources from which the software system will retrieve data. This can include databases, external APIs, files, or data entered manually by users. Main data sources is the API which retrieve data from the database.

5. **Data Validation Rules**: Defining the rules and constraints that need to be applied to the data to ensure its accuracy, integrity, and consistency. In the backend, there are set multiple data validation rules, while user upload any data these validation rules need to be filled.

6. **Data Storage**: Determining how the data will be stored and organized within the software system. For storing data, MySQL is used for this project.

7. **Data Integration**: Specifying how the software system will integrate and exchange data with other systems or databases, ensuring interoperability and data consistency.

## 3.3 Performance Requirements

The performance requirements define the expected performance characteristics of the system, including speed, accuracy, and capacity.

### 3.3.1 Speed and Latency Requirements

*Table 24: Fast Response for Search*

| | | | |
|---|---|---|---|
| **SLR** | Fast response for search | | |
| **Description** | When user search anything system will response fast | | |
| **Stakeholders** | Donors, Donee, Rider | **Priority** | Medium |

### 3.3.2 Precision or Accuracy Requirements

*Table 25: Precise and Accuracy data response for search*

| | | | |
|---|---|---|---|
| **PAR** | Precise and Accuracy data response for search | | |
| **Description** | System must response with accurate and precise data. When anything's are searching system must be give accurate data. | | |
| **Stakeholders** | Donors, Donee, Rider | **Priority** | Medium |

### 3.3.3 Capacity Requirements

*Table 26: Capacity Requirements*

| | | | |
|---|---|---|---|
| **CR** | The System will handle many user's data | | |
| **Description** | The system needs to handle data thousands of data every moment. | | |
| **Stakeholders** | Donors, Donee, Rider | **Priority** | Medium |

## 3.4 Dependability Requirements

Dependability requirements focus on ensuring that the "Food for Everyone" platform operates reliably, is consistently available, and can withstand failures while maintaining safety and integrity.

### 3.4.1 Reliability Requirements

Reliability requirement for the project aim to ensure that the system functions consistently, reliably, and securely, providing a dependable platform for users to manage their projects effectively.

*Table 27: Reliability Requirements*

| | | | |
|---|---|---|---|
| **RIR-1** | The System must keep user data secured | | |
| **Description** | The system should maintain the integrity of user data by ensuring accurate storage, retrieval, and modification of information. Data should be protected against loss, corruption, or unauthorized access | | |
| **Stakeholders** | Donors, Donee, Rider | **Priority** | Medium |

### 3.4.2 Availability Requirements

The system should have high availability to ensure uninterrupted access and operation. Downtime for maintenance or upgrade should be minimized and scheduled during low-usage periods to minimize disruption to users.

*Table 28: Availability Requirements*

| | | | |
|---|---|---|---|
| **AR** | The System must be available on 24 x 7 | | |
| **Description** | The system should maintain the integrity of user data by ensuring accurate storage, retrieval, and modification of information. Data should be protected against loss, corruption, or unauthorized access. | | |
| **Stakeholders** | Donors, Donee, Rider | **Priority** | Medium |

### 3.4.3 Robustness or Fault-Tolerance Requirements

Ensuring the robustness and fault-tolerance of the system is crucial to provide a seamless experience to end users. It is imperative to eliminate any instances of system crashes or failures, aiming for a 0% occurrence rate. By doing so, the system can operate reliably and consistently, minimizing disruptions and maximizing user satisfaction

*Table 29: Robustness Requirements*

| | | | |
|---|---|---|---|
| **RTR** | The system handles all user access without system errors. | | |
| **Description** | All stakeholder might hit our application system at a time. All their requests must be handled without any fault. | | |
| **Stakeholders** | Donors, Donee, Rider | **Priority** | Medium |

### 3.4.4 Safety-Critical Requirements

There are no safety-critical requirements in this project.

## 3.5 Maintainability and Supportability Requirements

### 3.5.1 Maintenance Requirements

*Table 30: Maintenance Requirements*

| | | | |
|---|---|---|---|
| **MR** | The system must Updates and Upgrades. | | |
| **Description** | To deal with any vulnerabilities and to take advantage of new features and improvements, System should always be up to date with the latest software versions and security patches | | |
| **Stakeholders** | Donors, Donee, Rider | **Priority** | Medium |

### 3.5.2 Supportability Requirements

Supportability requirements that our system meets, are-

- Testability
- Compatibility
- Configurability

### 3.5.3 Adaptability Requirements

- The web application should be designed to accommodate future changes and updates, allowing for seamless integration of new features and functionalities.
- It should support integration with external systems and APIs, enabling data exchange and interoperability with other software solutions.
- The application should be modular and well-structured, allowing for easy modification and extension of its components as needed.

### 3.5.4 Scalability or Extensibility Requirements

The ability of the Food for Everyone system to manage progressively heavier workloads, support expanding user bases, and enable the addition of additional features and functionalities is referred to as scalability or extensibility requirements. Below some this type of requirements are given:

*Table 31: Scalability Requirements*

| | | | |
|---|---|---|---|
| **SER-1** | The system should support load balancing. | | |
| **Description** | It should support load balancing techniques to evenly distribute user requests and workloads across multiple servers or resources. | | |
| **Stakeholders** | N/A | **Priority** | Medium |

## 3.6 Security Requirements

It's very important to include software security as a requirement. Its functional requirement should be the software security requirements. Software security ensures that the application system is secure. It is possible to directly test or observe the functionality of software security.

### 3.6.1 Access Requirements

There are still some authentication and authorization procedures for accessing our application system, and each module of our system will offer them.

*Table 32: Access Requirements*

| | | | |
|---|---|---|---|
| **ACR** | Provide security mechanism. | | |
| **Description** | Every module is designed in such a way that it only gives access to the authorized and authenticated users. | | |
| **Stakeholders** | Donors, Donee, Rider | **Priority** | High |

### 3.6.2 Integrity Requirements

The integrity requirements concern the security system that complies with data quality standards. It will also make sure that there is no possibility of unlawful alteration or accidental destruction of data saved in the system. We'll store the passwords of our users by encrypting a form that doesn't allow us to decrypt them. It's called a password hash as well.

### 3.6.3 Privacy Requirements

- The web application should comply with privacy regulations and protect the privacy of users' personal information.
- It should implement secure user authentication and access controls to ensure that only authorized individuals can access sensitive data.
- The application encrypts sensitive user data like password, both during transmission and storage, to prevent unauthorized access or interception.

## 3.7 Usability and Human-Interaction Requirements

These requirements focus on creating a user-friendly experience for the "Food for Everyone" platform, ensuring that it is accessible, understandable, and supportive for all users.

### 3.7.1 Ease of Use Requirements

*Table 33: Ease of Use Requirements*

| | | | |
|---|---|---|---|
| **EUR** | System must be usable for the end user | | |
| **Description** | This application is enough usable to the staff by which they can operate this system easily | | |
| **Stakeholders** | Donors, Donee, Rider | **Priority** | Medium |

### 3.7.2 Personalization and Internationalization Requirements

*Table 34: Personalization and Internationalization Requirements*

| | | | |
|---|---|---|---|
| **PIR** | Customizable User Profiles | | |
| **Description** | Users can build personalized customized profiles on the system. Users ought to be able to update and alter their profiles as necessary at any time. | | |
| **Stakeholders** | Donors, Donee, Rider | **Priority** | High |

### 3.7.3 Understandability and Politeness Requirements

The application that we are developing has been described as being so easy to understand. It provides users with an indication of whether a mistake has been made or not. By reading those error User may be able to operate the system easily.

*Table 35: Understandability Requirements*

| | | | |
|---|---|---|---|
| **UPR** | Clear and Simple Language | | |
| **Description** | Users of all educational levels and backgrounds should be able to quickly understand the system's language thanks to its clarity and conciseness. | | |
| **Stakeholders** | Donors, Donee, Rider | **Priority** | High |

### 3.7.4 Accessibility Requirements

*Table 36: Accessibility Requirements*

| | | | |
|---|---|---|---|
| **AR** | Application will provide role wise feature access mechanism | | |
| **Description** | Every module of the system will be designed such that for accessing it needs firstly to be a logged in and then user should be into the module as required role | | |
| **Stakeholders** | Donors, Donee, Rider | **Priority** | High |

### 3.7.5 User Documentation Requirements

There is no user documentation requirement in this system.

### 3.7.6 Training Requirements

Training requirements for a project are essential to ensure that individuals involved in the project, such as team members, end users, and stakeholders, have the necessary knowledge and skills to effectively utilize and interact with the project deliverables. These training needs may vary depending on the nature of the project and the target audience.

For the “Food for Everyone” project users need to understand it features, functionalities, and how to use them effectively. So, user training is necessary for the project. For different role in a project user will be provided different features and functionality based on user needs. So, role wise training may necessary for the user of “Food for Everyone”. The project is a solution for working with agile scrum framework, so the users who are not familiar with scrum may need to give training on Scrum.

## 3.8 Look and Feel Requirements

The looks and feel requirements refer in particular to how our system will appear, as well as the type of User Interface or Graphical User Interface on which it is displayed

### 3.8.1 Appearance Requirements

Appearance requirements of a project refer to the desired visual and aesthetic aspects that contribute to the overall look and feel of the project deliverables. These requirements focus on the design, layout, and graphical elements that create a visually appealing and cohesive user experience.

*Table 37: Appearance Requirements-1*

| | | | |
|---|---|---|---|
| **APR-1** | Label for mandatory field must be in UI form | | |
| **Description** | The fields which mandatory by business or database logic to a particular activity which is mandatory to place in the UI | | |
| **Stakeholders** | Donors, Donee, Rider | **Priority** | Low |

*Table 38: Appearance Requirements-2*

| | | | |
|---|---|---|---|
| **APR-2** | Error message will be in red colored | | |
| **Description** | Validation message for any action to any form field will be shown in red colored | | |
| **Stakeholders** | Donors, Donee, Rider | **Priority** | Low |

### 3.8.2 Style Requirements

Style requirements for “Food for Everyone” encompass the specific guidelines and considerations related to the visual style and presentation of the user interface.

*Table 39: Style Requirements*

| | | | |
|---|---|---|---|
| **STR** | Modern and Contemporary Style | | |
| **Description** | The system should adopt a modern, contemporary look that incorporates the most recent fashions in design. Clean lines, simple design components, and an emphasis on understatement and elegance across the visual presentation are examples of this. | | |
| **Stakeholders** | Donors, Donee, Rider | **Priority** | Low |

## 3.9 Operational and Environment Requirements

For “Food for Everyone” project to run smoothly, it is necessary to take into account a number of specific operational and environmental needs. Some examples of such requirements include the physical environment, interfacing with nearby systems, projecting, and releasing.

### 3.9.1 Expected Physical Requirements

Our system ought to be created to function in the typical physical setting, which typically consists of:

- Support for various platforms and devices, including desktop and laptop PCs.
- Accessibility across various OS systems is guaranteed by compatibility with widely used web browsers.
- Sufficient network connectivity to access the system from various university sites and outside contexts.

### 3.9.2 Requirements for Interfacing with Adjacent Systems

It could be necessary for our system to link with nearby systems or outside services. Some specifications in this regard could be:

- Integration with the already-in-place authentication system at the institution to give users secure access.
- To retrieve real-time results, sports news, player statistics, etc., integration with external APIs or data sources is required.
- Data consistency and integrity are guaranteed via compatibility with databases or data repositories utilized by other university systems.

### 3.9.3 Projectization Requirements

To facilitate a smooth development and deployment, “Food for Everyone” project may have special projectization criteria. These may consist of:

- Iterative development and quality assurance are facilitated by the availability of development and testing environments.
- For efficient project management and team coordination, use a version control system and collaboration tools.
- Venues for sharing expertise and documentation to guarantee project continuance.

### 3.9.4 Release Requirements

The system does not have any Release Requirements.

## 3.10 Legal Requirements

Legal requirements for a project refer to the laws, regulations, and legal obligations that must be adhered to throughout the project's lifecycle. These requirements ensure that the project is conducted in compliance with applicable laws and regulations to mitigate legal risks and maintain ethical practices. Legal requirements can vary depending on the nature of the project and the jurisdiction in which it operates. For our project we don’t have any requirable legal requirements which is must under consideration and has great importance directly to our system.

### 3.10.1 Compliance Requirements

Currently there are no compliance requirements to our system.

### 3.10.2 Standards Requirements

Currently there are no standard requirements for our project.

# Chapter Four: System Analysis

## 4.1 Use Case Diagram

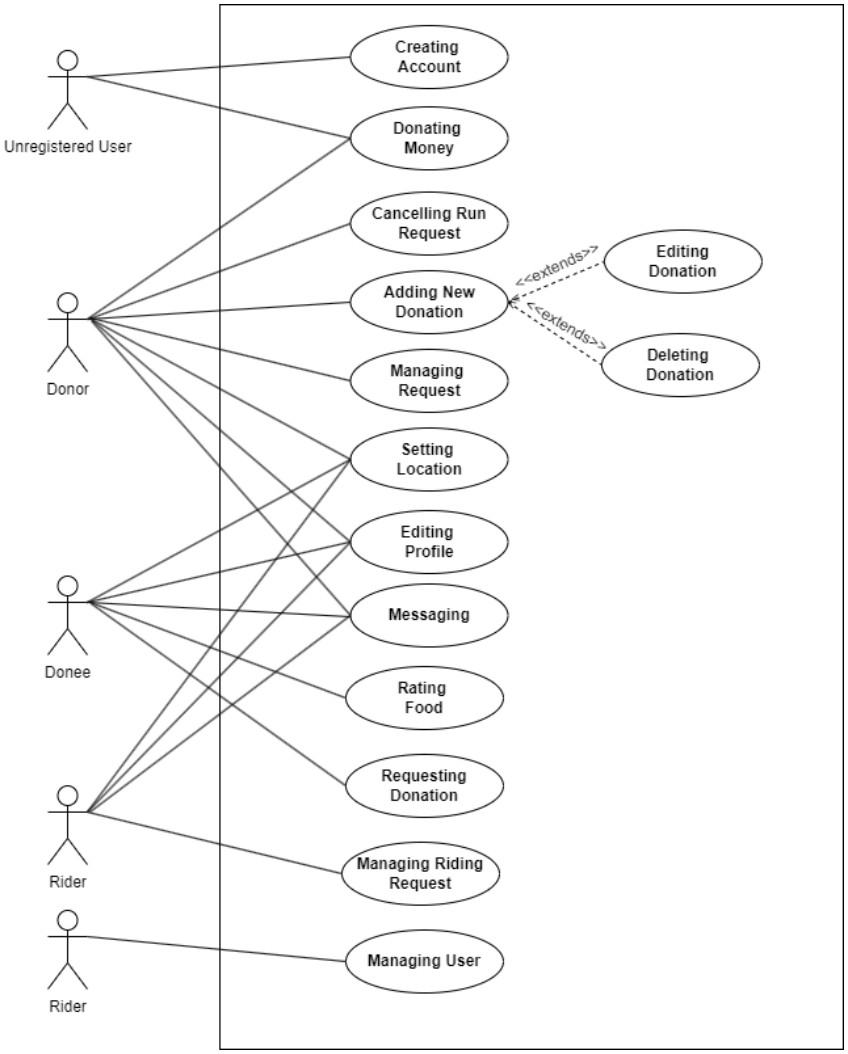



*Figure 3: Usecase-1 (Diagram)*

## 4.2 Use Case Description

*Table 40: Usecase-1 (Creating Account)*

| | | |
|---|---|---|
| **Use Case** | Creating Account | |
| **Goal** | User wants to create an account in the “Food for Everyone” platform. | |
| **Preconditions** | N/A | |
| **Success End Condition** | A user account is created. | |
| **Failed End Condition** | User account is not created. | |
| **Primary Actors:**<br><br>**Secondary Actors:** | New User<br><br>System | |
| **Trigger** | “Registration” button needs to be clicked. | |
| **Main Success Flows** | **Step** | **Action** |
| | 1 | User opens the “Food for Everyone” application. |
| | 2 | User clicks the “Registration” button. |
| | 3 | User provides required information to fill the registration form. |
| | 4 | User submits the registration form. |
| | 5 | Admin verifies and accepts the registration. |
| | 6 | User gets confirmation mail for new Account and activates the account. |
| **Alternative Flows** | **Step** | **Branching Action** |
| | 4a1 | User does not provide all information. |
| | 4a2 | Server shows that information invalid or used before. |
| | 6a | Admin gets “Account request rejected through verification process” mail. |
| **Quality Requirements** | **Step** | **Requirement** |
| | 6 | Server will respond within 3 to 5 seconds |

*Table 41: Usecase-2 (Adding Food Donation)*

| | | |
|---|---|---|
| **Use Case** | Adding Food donation | |
| **Goal** | Donors add a new food donation post to the platform. | |
| **Preconditions** | User is logged in | |
| **Success End Condition** | New food donation is posted | |
| **Failed End Condition** | Failed to add new donation post | |
| **Primary Actors:**<br><br>**Secondary Actors:** | Donor<br><br>Admin | |
| **Trigger** | "Add Food Donation" button needs to be clicked. | |
| **Main Success Flows** | **Step** | **Action** |
| | 1 | Donor selects the option to add a new food donation. |
| | 2 | Donor fills in the details of the donation (e.g., food type, quantity, expiry date). |
| | 3 | Donor saves the donation post. |
| | 4 | Donation post is added to the platform. |
| **Alternative Flows** | **Step** | **Branching Action** |
| | 4a | Failed to add new donation post. Donor does not provide all information |
| **Quality Requirements** | **Step** | **Requirement** |
| | 4 | Post will be added within 3 to 5 seconds |

*Table 42: Usecase-3 (Deleting Donation)*

| | | |
|---|---|---|
| **Use Case** | Deleting donation post | |
| **Goal** | Delete the previously donated post | |
| **Preconditions** | User is logged in | |
| **Success End Condition** | Post is deleted | |
| **Failed End Condition** | Failed to Delete the post. | |
| **Primary Actors:**<br>**Secondary Actors:** | Donor<br>System | |
| **Trigger** | "Delete" button needs to be clicked. | |
| **Main Success Flows** | **Step** | **Action** |
| | 1 | Select the post for delete. |
| | 2 | Post is deleted. |
| **Alternative Flows** | **Step** | **Branching Action** |
| | | N/A |
| **Quality Requirements** | | N/A |

*Table 43: Usecase-4 (Editing Donation)*

| | | |
|---|---|---|
| **Use Case** | Editing donation | |
| **Goal** | Food donation post details is updated | |
| **Preconditions** | User posted a donation | |
| **Success End Condition** | Post updated | |
| **Failed End Condition** | Failed to update the post. | |
| **Primary Actors:**<br><br>**Secondary Actors:** | Donor<br><br>System | |
| **Trigger** | "Edit" button needs to be clicked. | |
| **Main Success Flows** | **Step** | **Action** |
| | 1 | Donor selects the donation post and clicks edit. |
| | 2 | Changes some information and clicks update button. |
| | 3 | Post is updated. |
| **Alternative Flows** | **Step** | **Branching Action** |
| | | N/A |
| **Quality Requirements** | | N/A |

*Table 44: Usecase-5 (Setting Location)*

| | | |
|---|---|---|
| **Use Case** | Setting Location | |
| **Goal** | Users set their location preferences to find nearby donations or rider opportunities. | |
| **Preconditions** | User is logged in | |
| **Success End Condition** | Location selected | |
| **Failed End Condition** | Failed to set location | |
| **Primary Actors:**<br><br>**Secondary Actors:** | Donor, Donee, Volunteer Rider<br><br>System | |
| **Trigger** | "Set location" button needs to be clicked. | |
| **Main Success Flows** | **Step** | **Action** |
| | 1 | User visit profile page and click "Change location ". |
| | 2 | User enters their preferred location manually or allows the app to access their current location. |
| | 3 | User saves the location preferences. |
| | 4 | System updates the user's location preferences. |
| **Alternative Flows** | **Step** | **Branching Action** |
| | 2a | Could not find preferred given location |
| **Quality Requirements** | | N/A |

*Table 45: Usecase-6 (Requesting donation)*

| | | |
|---|---|---|
| **Use Case** | Requesting donation | |
| **Goal** | Request for getting donation on selected donation post. | |
| **Preconditions** | Post needs to be created | |
| **Success End Condition** | Request is proceeded | |
| **Failed End Condition** | Failed to request | |
| **Primary Actors:**<br><br>**Secondary Actors:** | Donee<br><br>System | |
| **Trigger** | "Request" button needs to be clicked. | |
| **Main Success Flows** | **Step** | **Action** |
| | 1 | Donee searches for available food donations or checks click food donations |
| | 2 | Donee selects a suitable donation post and sends a request to the donor. |
| | 3 | Donor receives the donation request. |
| **Alternative Flows** | **Step** | **Branching Action** |
| | | N/A |
| **Quality Requirements** | 3a | Donor receives the request within 1 to 5 seconds |

*Table 46: Usecase-7 (Managing request)*

| | | |
|---|---|---|
| **Use Case** | Managing request | |
| **Goal** | Accept or cancels donation request. | |
| **Preconditions** | N/A | |
| **Success End Condition** | Request is cancelled | |
| **Failed End Condition** | Cancellation failed | |
| **Primary Actors:**<br>**Secondary Actors:** | Donor<br>System | |
| **Trigger** | "Accept" or "Cancel" button needs to be clicked. | |
| **Main Success Flows** | **Step** | **Action** |
| | 1 | Donor goes to the requested page |
| | 2 | Clicks "See all request" button. |
| | 3 | Clicks "Accept" button |
| | 4 | Request is Accepted |
| **Alternative Flows** | **Step** | **Branching Action** |
| | 3a | Click "Reject" button and request is rejected |
| **Quality Requirements** | | N/A |

*Table 47: Usecase-8 (Donating money)*

| | | |
|---|---|---|
| **Use Case** | Donating money | |
| **Goal** | Contribute funds for the maintenance of the application. | |
| **Preconditions** | User must visit Food for Everyone. | |
| **Success End Condition** | Money is donated | |
| **Failed End Condition** | Donation failed | |
| **Primary Actors:**<br><br>**Secondary Actors:** | Any User<br><br>Admin | |
| **Trigger** | "Donate Money" button needs to be clicked. | |
| **Main Success Flows** | **Step** | **Action** |
| | 1 | User selects the option to donate money |
| | 2 | User specifies the amount they want to donate and provide the required information in the form. |
| | 3 | Payment option appears |
| | 4 | The system processes the donation |
| | 5 | The user gets a successful donation confirmation message. |
| **Alternative Flows** | **Step** | **Branching Action** |
| | 4a | Donation Payment failed. |
| **Quality Requirements** | **Step** | **Requirement** |
| | 4 | Server will process the donation within 3 to 5 seconds |

*Table 48: Usecase-9 (Messaging)*

| | | |
|---|---|---|
| **Use Case** | Messaging | |
| **Goal** | Communication between donor, donee and rider. | |
| **Preconditions** | User is logged in | |
| **Success End Condition** | Successfully communicated through messaging | |
| **Failed End Condition** | Communication failed | |
| **Primary Actors:**<br><br>**Secondary Actors:** | Donee, Donor and Rider<br><br>System | |
| **Trigger** | "Message" button needs to be clicked. | |
| **Main Success Flows** | **Step** | **Action** |
| | 1 | User clicks on message option and select the person he/she wants to communicate. |
| | 2 | Types message and clicks send |
| | 3 | Receivers' response and after all communication and delivery completion chat is ended |
| **Alternative Flows** | **Step** | **Branching Action** |
| | | N/A |
| **Quality Requirements** | | N/A |

*Table 49: Usecase-10 (Rating food)*

| | | |
|---|---|---|
| **Use Case** | Rating food | |
| **Goal** | Rate the donated food. | |
| **Preconditions** | User is logged in and got donation | |
| **Success End Condition** | Food is rated | |
| **Failed End Condition** | Failed to rating food | |
| **Primary Actors:**<br><br>**Secondary Actors:** | Donee<br><br>System | |
| **Trigger** | "Rate food" button is clicked | |
| **Main Success Flows** | **Step** | **Action** |
| | 1 | Donee click on Received |
| | 2 | User clicks rate. |
| | 3 | Rates the food and clicks done. |
| **Alternative Flows** | **Step** | **Branching Action** |
| | | N/A |
| **Quality Requirements** | | N/A |

*Table 50: Usecase-11 (Editing Profile)*

| | | |
|---|---|---|
| **Use Case** | Editing Profile | |
| **Goal** | Update information. | |
| **Preconditions** | User must be logged in | |
| **Success End Condition** | Successfully update information. | |
| **Failed End Condition** | Unable to update information. | |
| **Primary Actors:**<br><br>**Secondary Actors:** | User<br><br>System | |
| **Trigger** | "Edit Profile" Button needs to be clicked. | |
| **Main Success Flows** | **Step** | **Action** |
| | 1 | User click the "Edit Profile". |
| | 2a | User click the "Change username" |
| | 2a1 | System asks new username. User provide new username. |
| | 2b | User click the "Change password" |
| | 2b1 | System asks old and new password. User provide old and new password. |
| | 3 | User click the "Save" Button |
| | 4 | System updates the username. |
| **Alternative Flows** | **Step** | **Branching Action** |
| | 2b1 | Old password doesn't match .user is asked to add re-enter new password |
| | 4 | If the password doesn't include a minimum number, special character and letter and size of the password is less than 6 characters, then the system will again ask to enter a new password. |
| **Quality Requirements** | **Step** | **Requirement** |
| | | N/A |

*Table 51: Usecase-12 (Managing User)*

| | | |
|---|---|---|
| **Use Case** | Managing user | |
| **Goal** | Admin can handle user | |
| **Preconditions** | N/A | |
| **Success End Condition** | Manage User | |
| **Failed End Condition** | Action handling issue | |
| **Primary Actors:**<br>**Secondary Actors:** | Admin<br>System | |
| **Trigger** | Action button need to be clicked | |
| **Main Success Flows** | **Step** | **Action** |
| | 1 | Admin click on user option |
| | 2 | Click on an individual user for approve |
| | 3 | User Registration request approved |
| **Alternative Flows** | **Step** | **Branching Action** |
| | | N/A |
| **Quality Requirements** | | N/A |

*Table 52: Usecase-13 (Cancelling run post)*

| | | |
|---|---|---|
| **Use Case** | Cancelling run post | |
| **Goal** | Cancel previously accepted donation. | |
| **Preconditions** | User is logged in | |
| **Success End Condition** | Running donation is cancelled | |
| **Failed End Condition** | Running donation cancelation failed | |
| **Primary Actors:**<br><br>**Secondary Actors:** | Donee<br><br>System | |
| **Trigger** | "Cancel" button needs to be clicked. | |
| **Main Success Flows** | **Step** | **Action** |
| | 1 | Donee goes to the running post |
| | 2 | Clicks cancel button. |
| | 3 | Running Request is cancelled |
| **Alternative Flows** | **Step** | **Branching Action** |
| | | N/A |
| **Quality Requirements** | | N/A |

*Table 53: Usecase-14 (Managing Riding Request)*

| | | |
|---|---|---|
| **Use Case** | Managing Riding Request | |
| **Goal** | Accept or reject delivery request. | |
| **Preconditions** | Rider is logged in | |
| **Success End Condition** | Accept or Reject Request | |
| **Failed End Condition** | managing failed | |
| **Primary Actors:**<br><br>**Secondary Actors:** | Rider<br><br>System | |
| **Trigger** | "Accept or Reject" button needs to be clicked. | |
| **Main Success Flows** | **Step** | **Action** |
| | 1 | System sends a request to the rider based on the location and availability |
| | 2 | Rider accepts the request. |
| | 3 | Request set to the on running section |
| **Alternative Flows** | **Step** | **Branching Action** |
| | 2a | Rider Cancel the request |
| **Quality Requirements** | | N/A |

## 4.3 Activity Diagram

### Creating Account

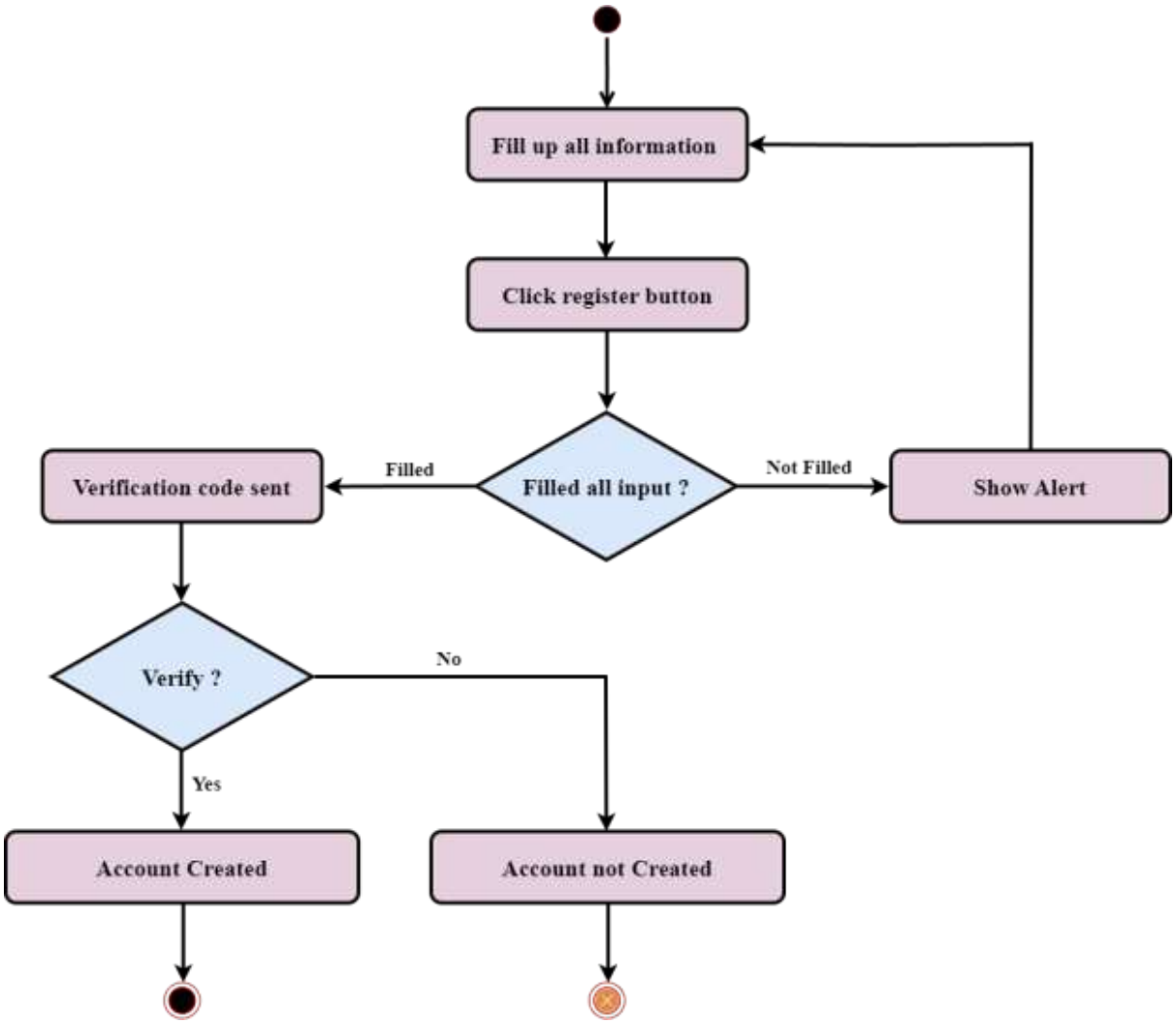



*Figure 4: Activity Diagram-1 (Creating Account)*

## Adding food donation

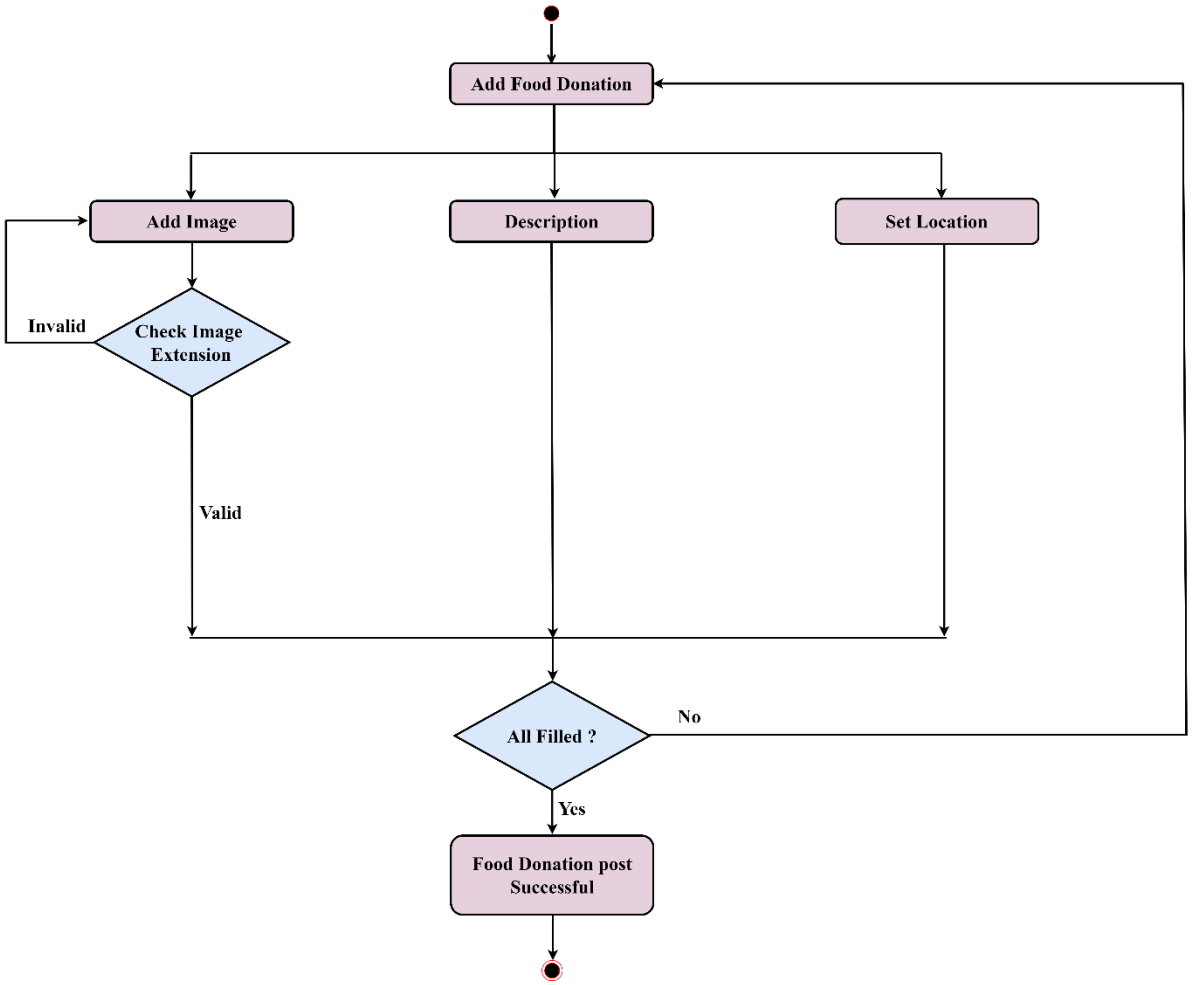



*Figure 5: Activity Diagram-2 (Adding food Donation)*

**Deleting Donation**

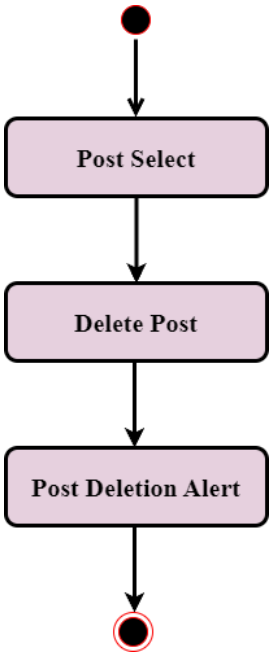



*Figure 6: Activity Diagram-3 (Deleting Donation)*

**Editing Donation**

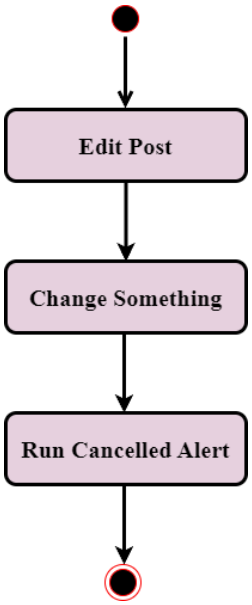



*Figure 7: Activity Diagram-4 (Editing Donation)*

### Setting Location

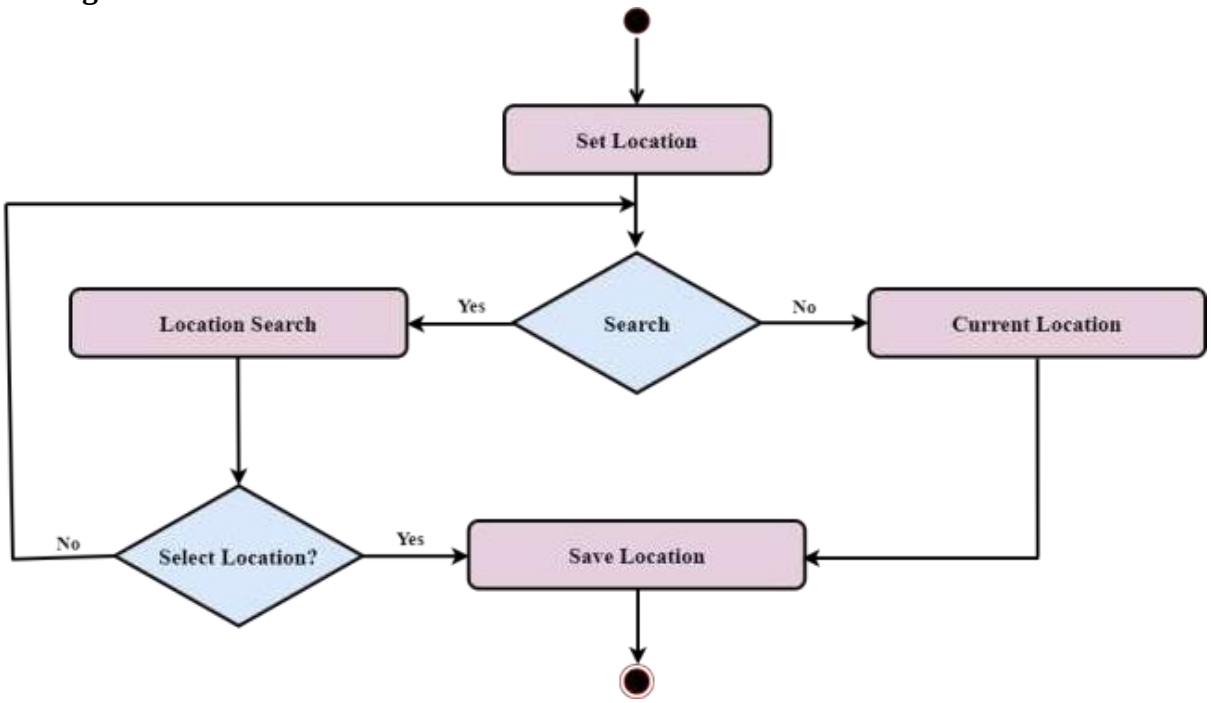



*Figure 8: Activity Diagram-5 (Setting Location)*

### Requesting Food

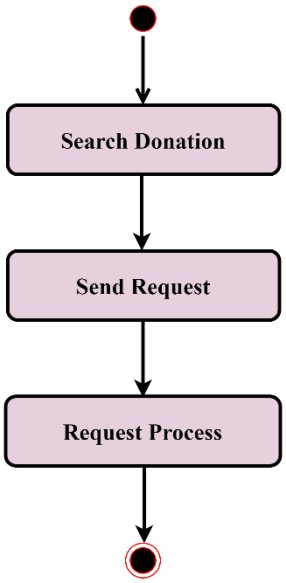



*Figure 9: Activity Diagram-6 (Requesting Food)*

**Managing Request**

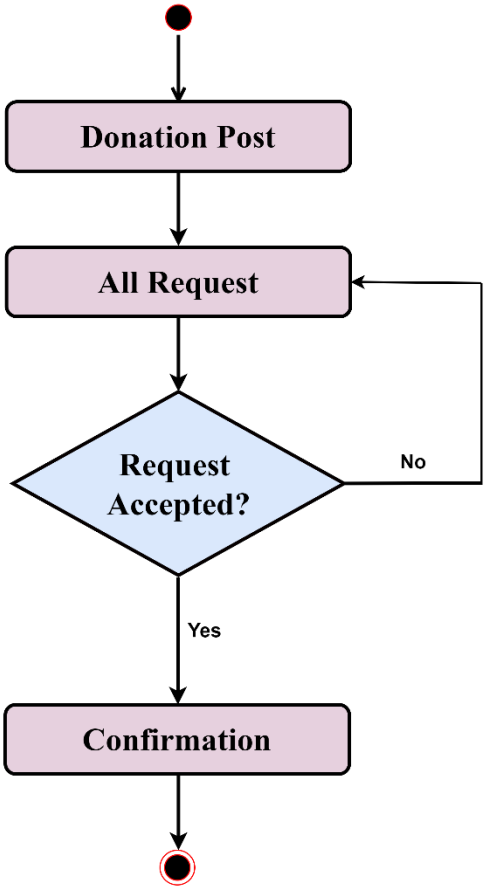



*Figure 10: Activity Diagram-7 (Managing Request)*

**Donating Money**

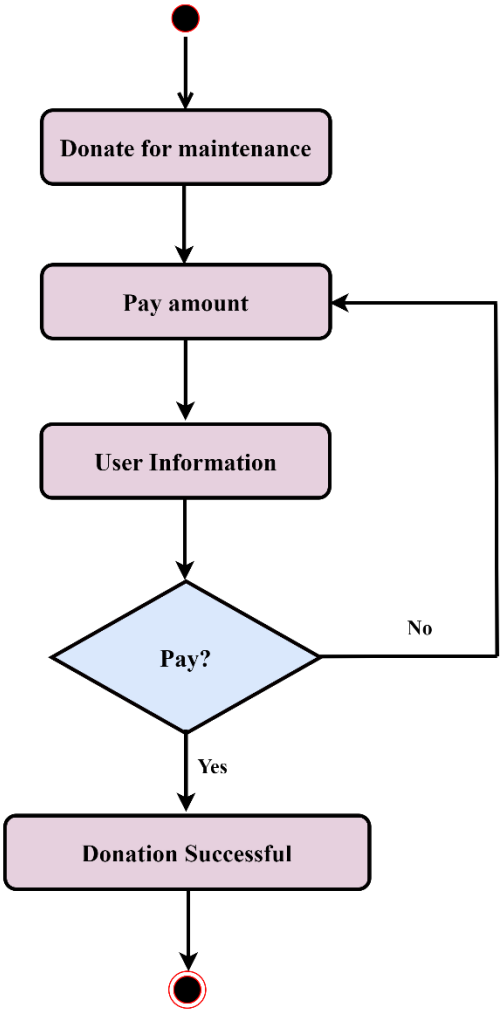



*Figure 11: Activity Diagram-8 (Donating Money)*

**Messaging**

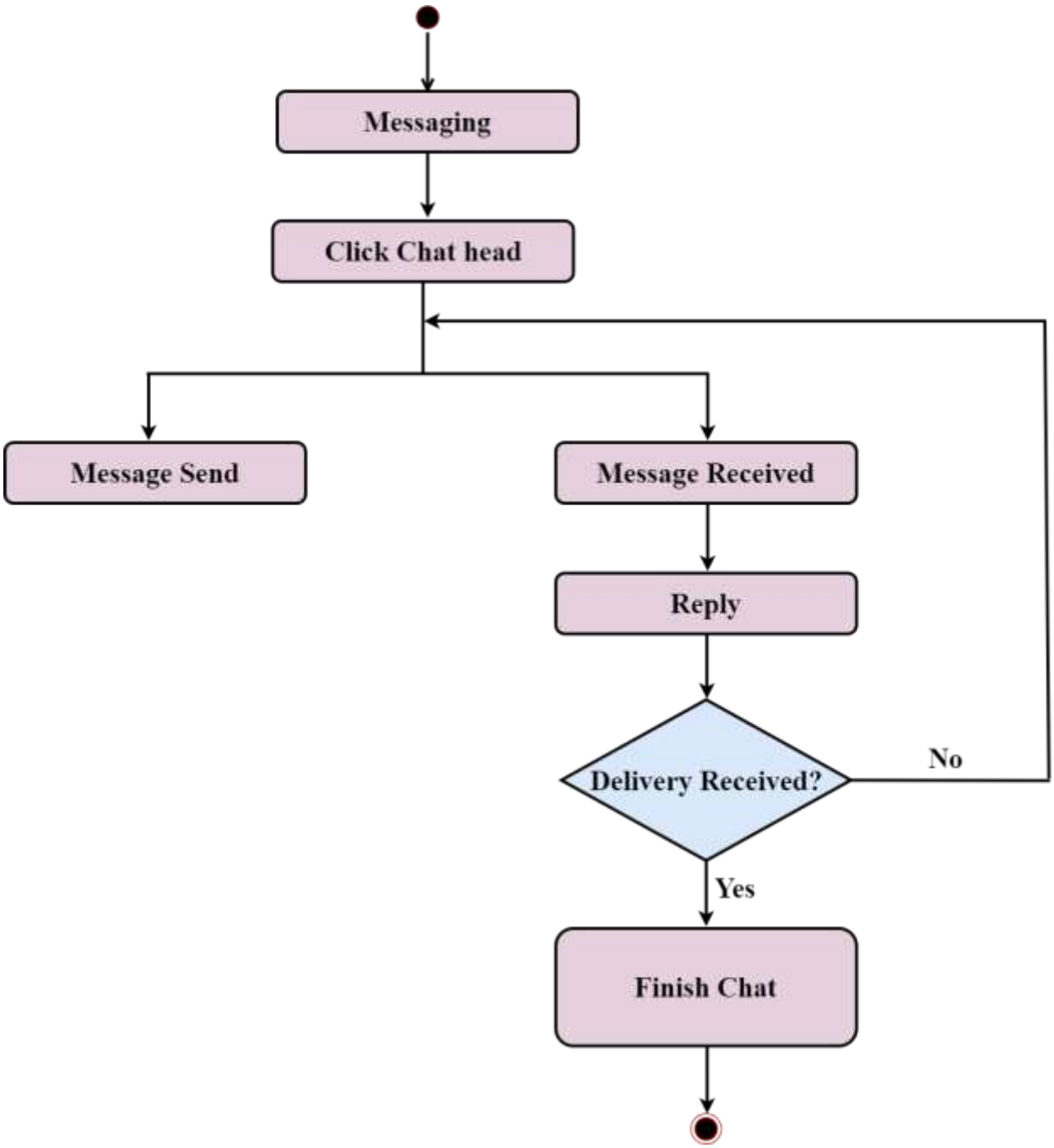



*Figure 12: Activity Diagram-9 (Messaging)*

### Rating Food

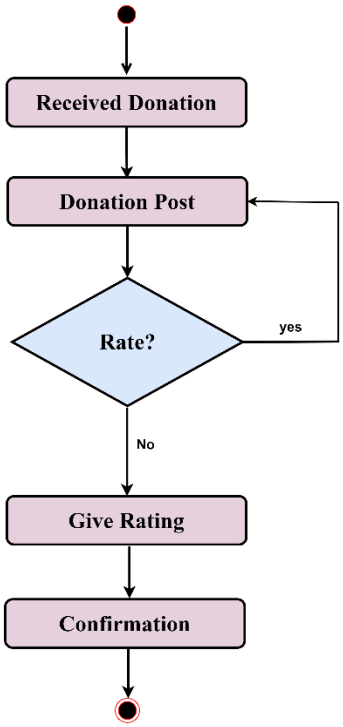



*Figure 13: Activity Diagram-10 (Rating Food)*

### Editing Profile

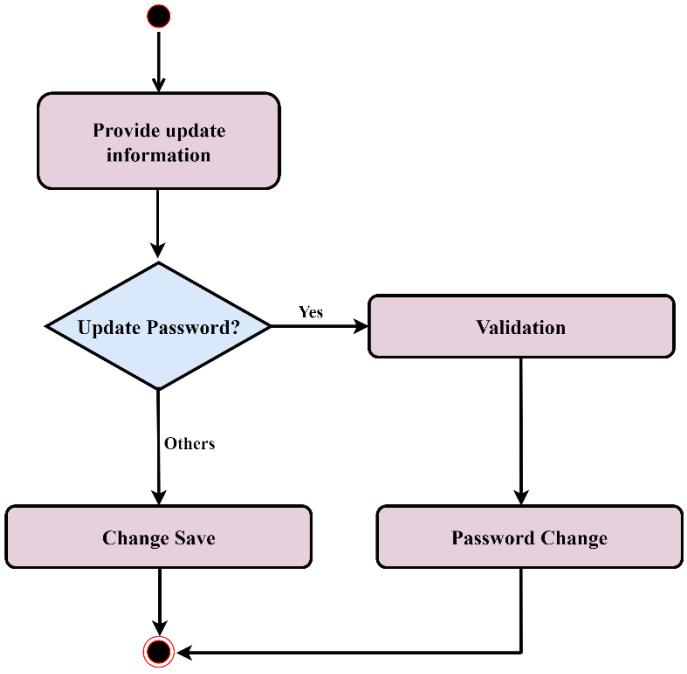



*Figure 14: Activity Diagram-11 (Editing Profile)*

**Managing User**

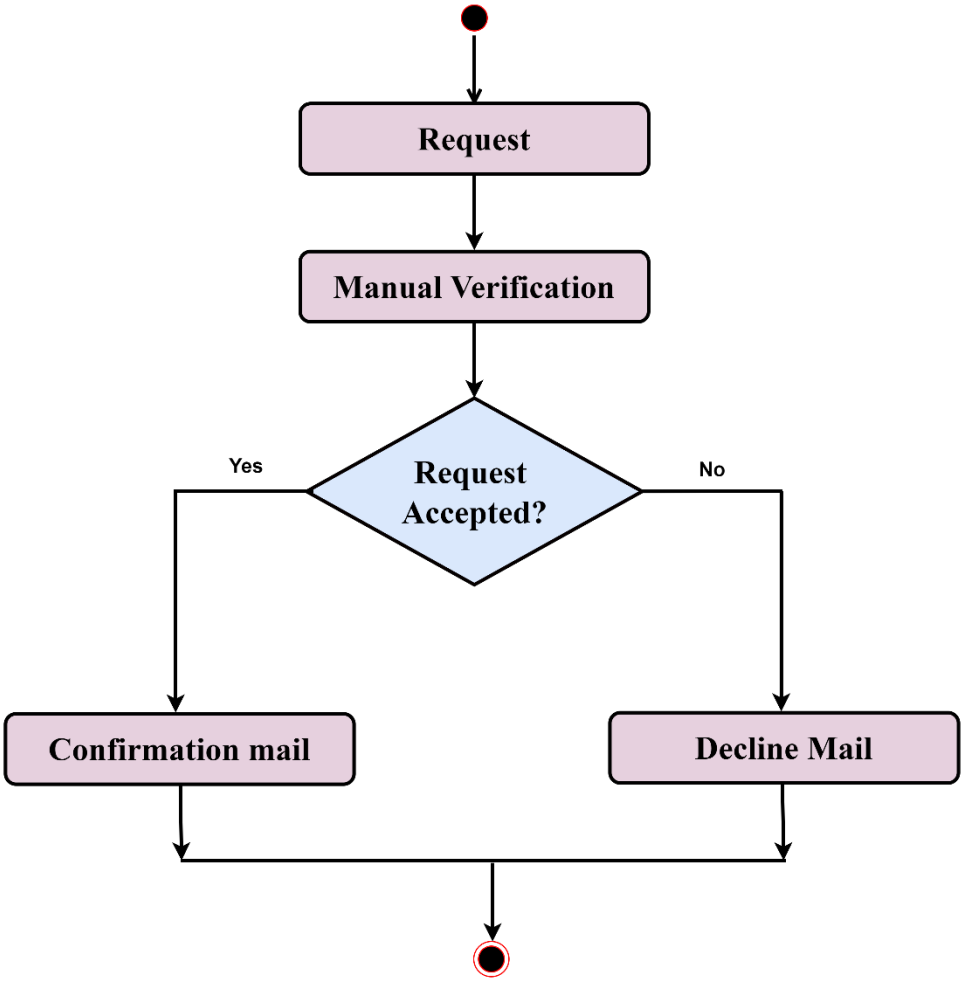



*Figure 15: Activity Diagram-12 (Managing User)*

### Cancelling running post

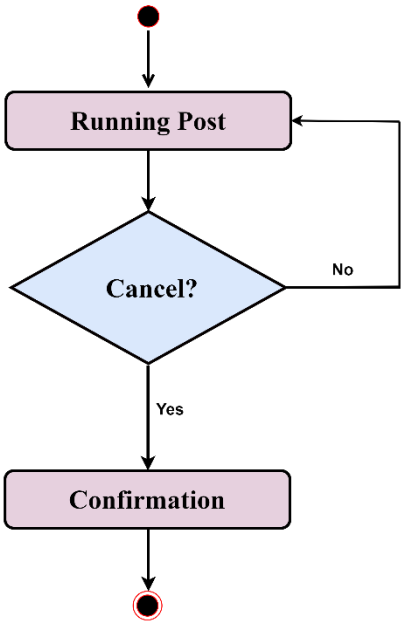



*Figure 16: Activity Diagram-13 (Cancelling running post)*

### Managing riding request

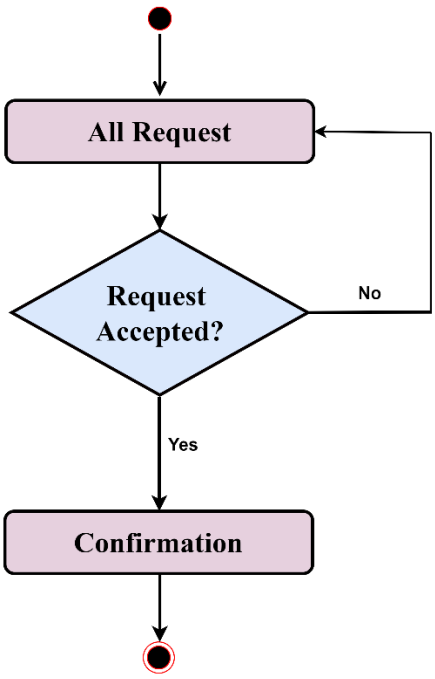



*Figure 17: Activity Diagram-14 (Managing riding request)*

## 4.4 System Sequence Diagram

### Creating Account

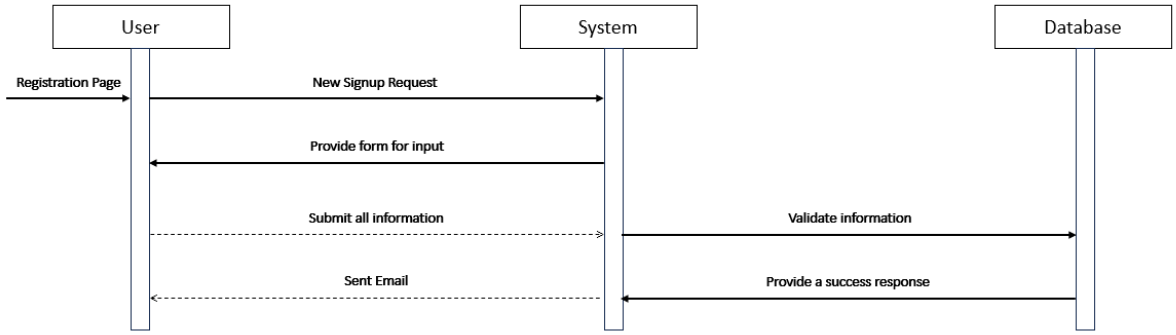



*Figure 18: Sequence Diagram-1 (Creating Account)*

### Adding food donation

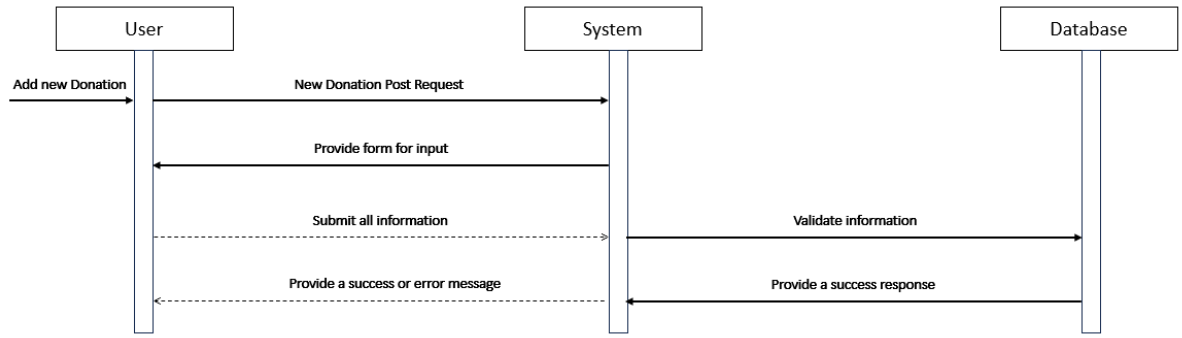



*Figure 19: Sequence Diagram- 2 (Adding food donation)*

### Deleting Donation

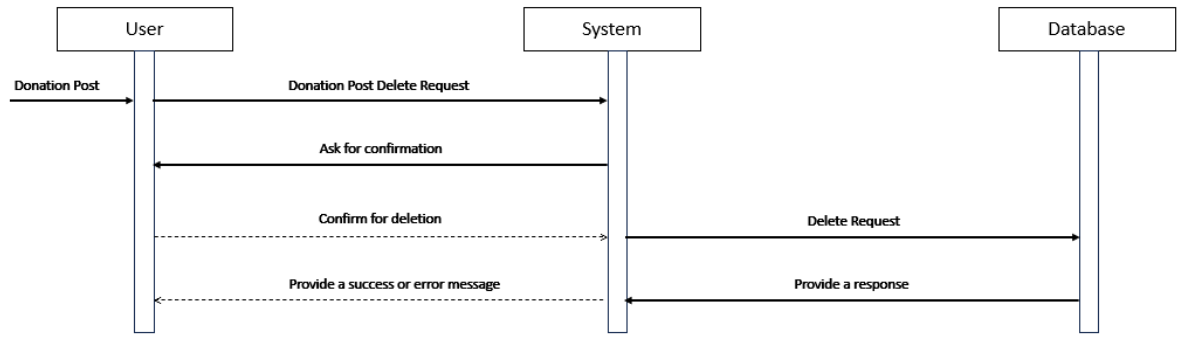



*Figure 20: Sequence Diagram- 3 (Deleting Donation)*

## Editing Donation

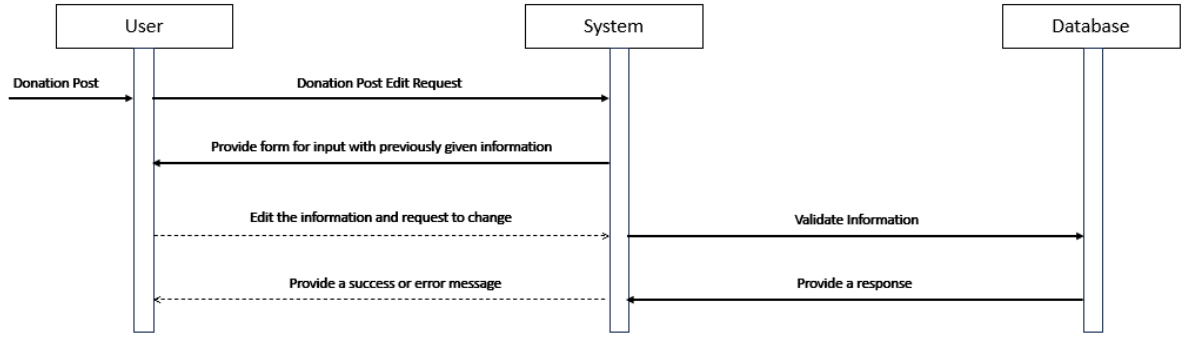



*Figure 21: Sequence Diagram- 4 (Editing Donation)*

## Setting Location

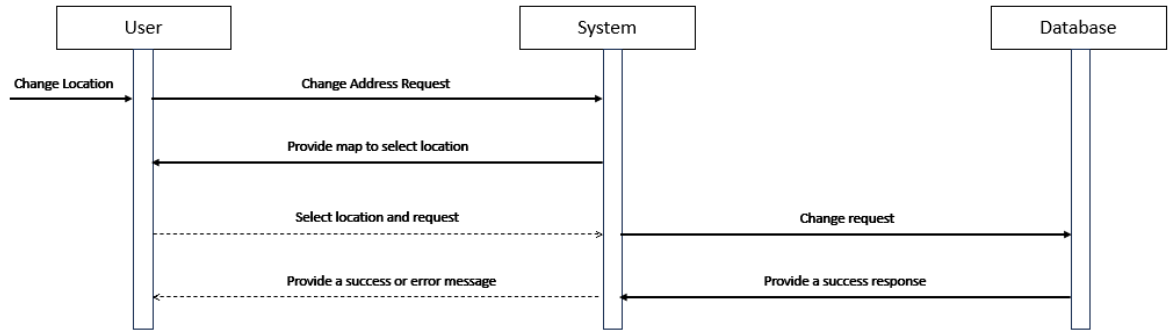



*Figure 22: Sequence Diagram- 5 (Setting Location)*

## Requesting Food

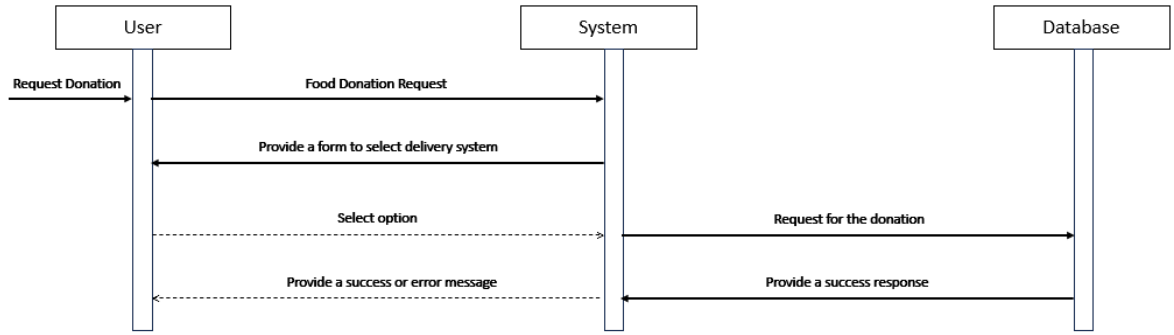



*Figure 23: Sequence Diagram- 6 (Requesting Food)*

## Managing Request

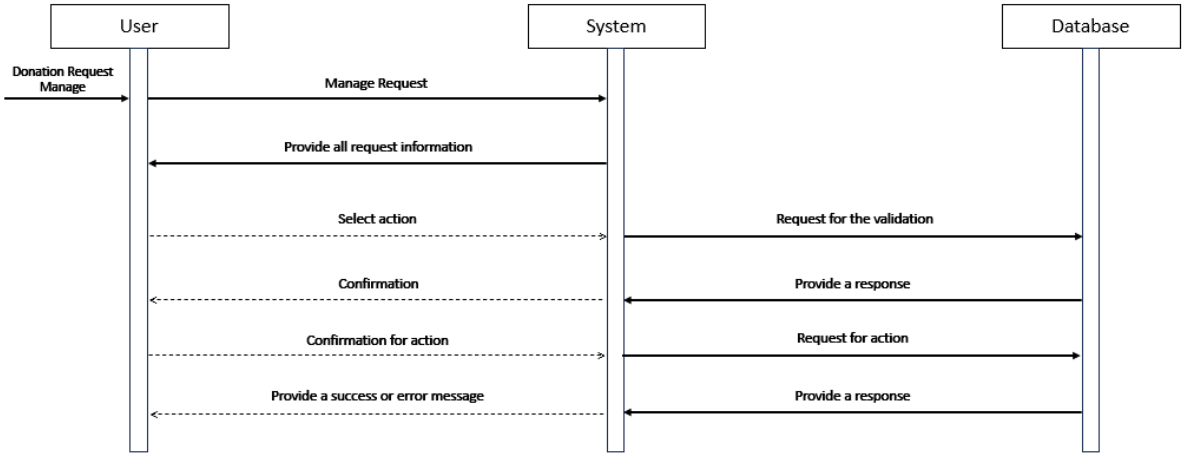



*Figure 24: Sequence Diagram- 7 (Managing Request)*

## Donating Money

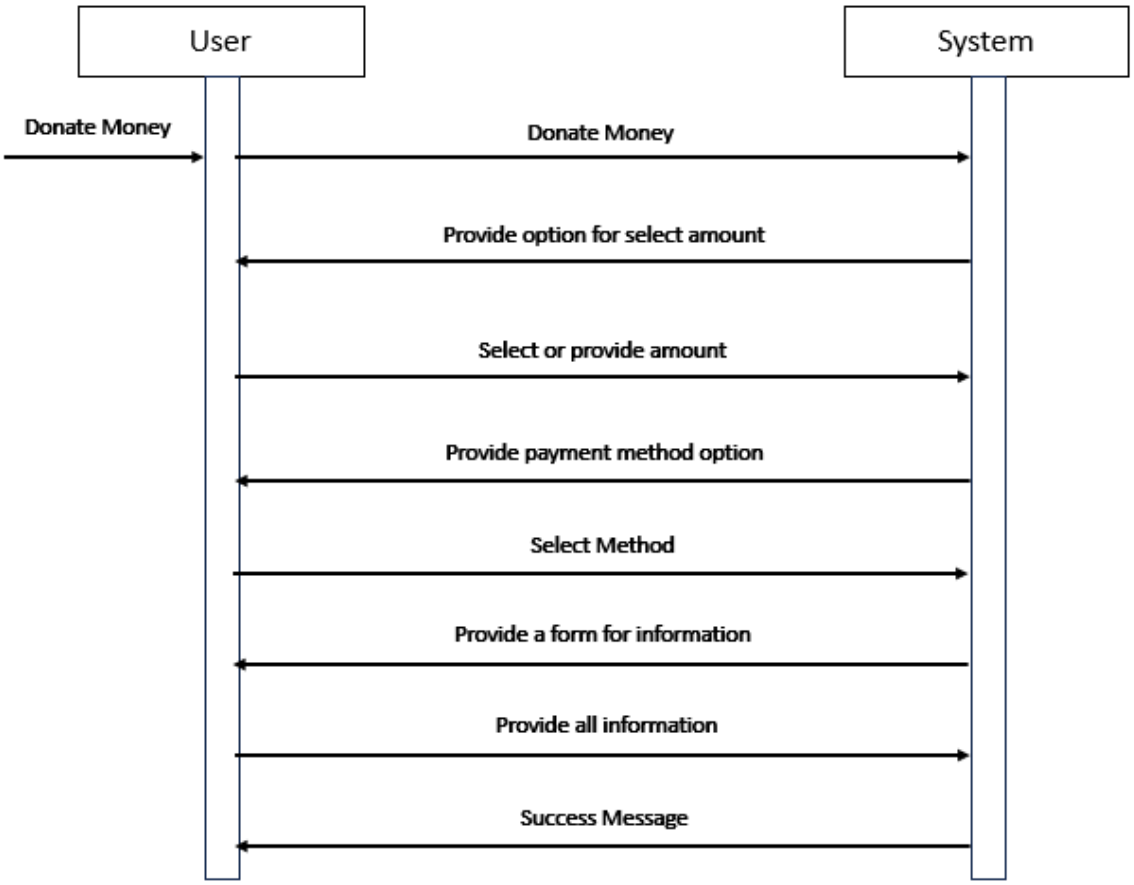



*Figure 25: Sequence Diagram- 8 (Donating Money)*

## Messaging

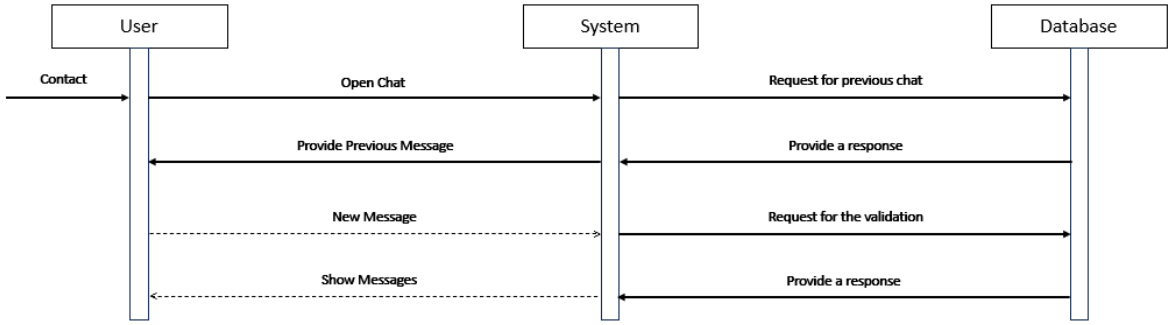



*Figure 26: Sequence Diagram- 9 (Messaging)*

## Rating Food

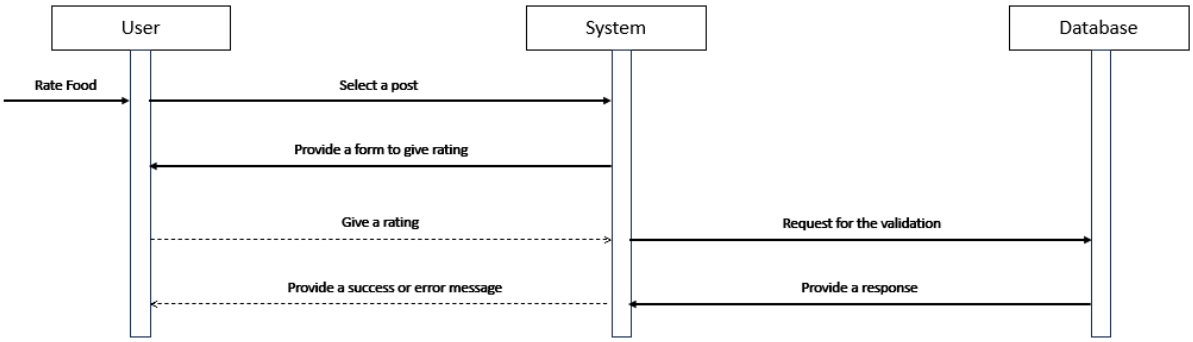



*Figure 27: Sequence Diagram-10 (Rating Food)*

## Editing Profile

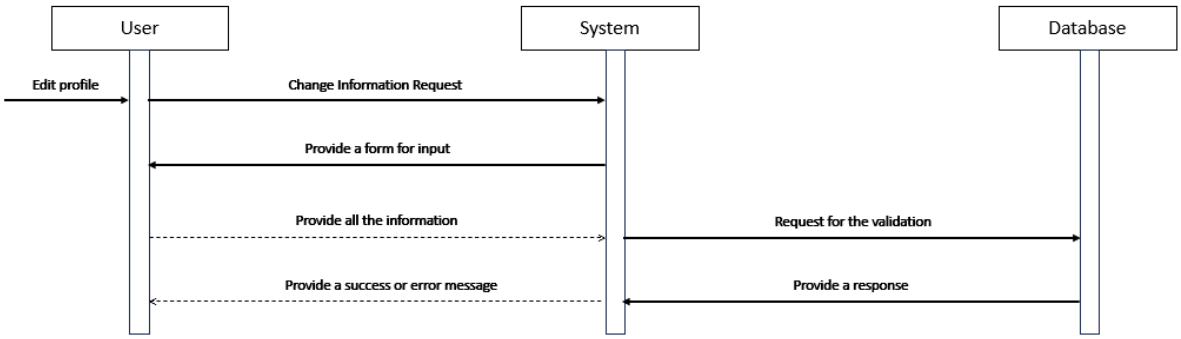



*Figure 28: Sequence Diagram- 11 (Editing Profile)*

## Managing User

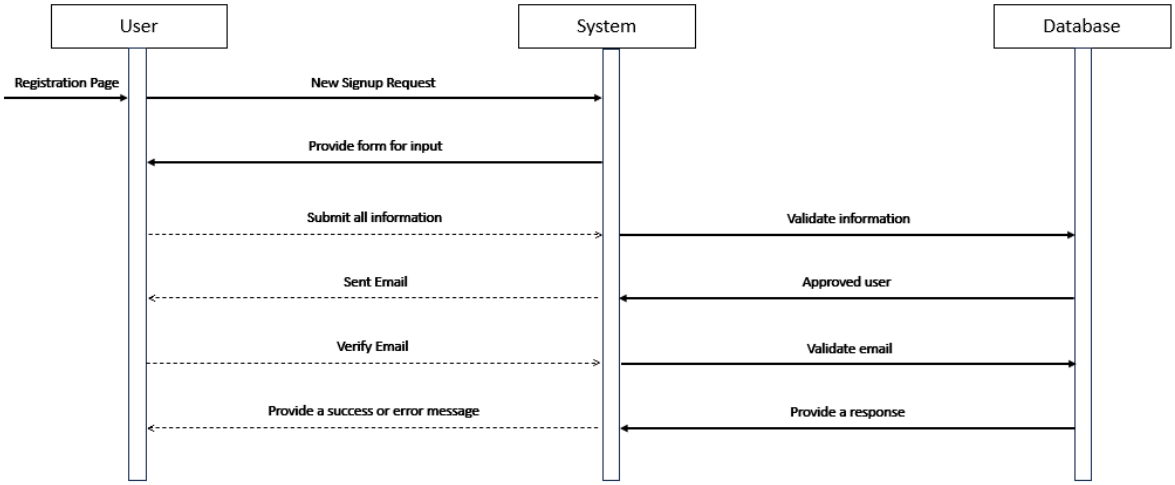



*Figure 29: Sequence Diagram- 12 (Managing User)*

## Cancelling running post

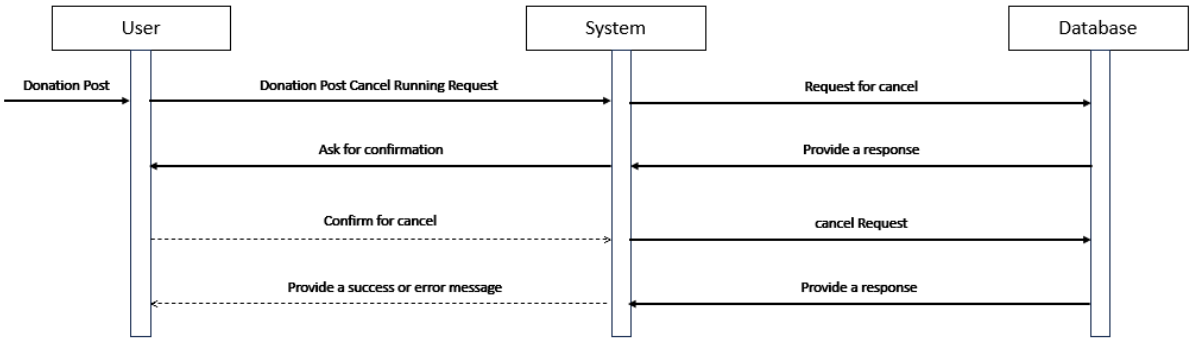



*Figure 30: Sequence Diagram- 13 (Cancelling running post)*

## Managing riding request

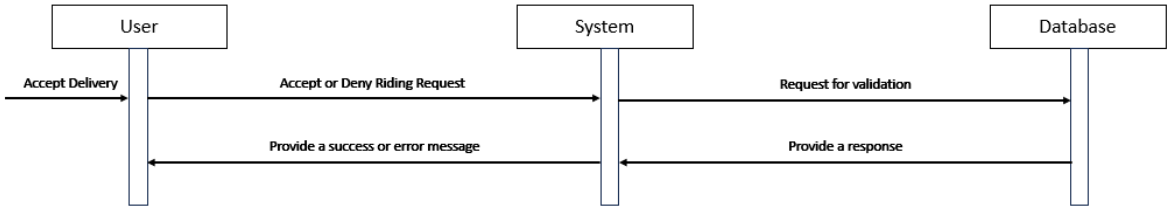



*Figure 31: Sequence Diagram-14 (Managing riding request)*

# Chapter Five: System Design and Architecture

## 5.1 System Architecture Overview

**1. Presentation Layer (Front-End):** This layer is the layer of the user interface and interaction, which is designed and implemented with the usage of the modern web technologies and utilizing such characteristics as responsiveness.

**Technologies Used**: The front end of this application developed with use of HTML, CSS, Bootstrap and React JS.

- HTML & CSS: Elements that focus on web page architecture and design.
- Bootstrap: Bootstrap to make project's user interface visually beautiful and responsive
- ReactJS: In terms of the component design, React provides easy management of reusable UI components such as forms for food donations and listing available food.

**User Interaction & UI Components:**

- Dynamic Forms and Input Handling: Forms concerning users, foods, and requests are handled by react components.
- Responsive Layouts: Bootstrap makes the website have responsive design meaning that it adapts to the screen size being used.
- API Requests: React uses asynchronous calls to the Laravel back-end to get or send data, for instance, new posts of food donations or available foodstuffs.

**2. Business Logic Layer:** The business logic is built with Laravel framework and deals with the request object and performs business rules. It is a layer that allows for easy running and deals with managing the connection between the front end and the back end.

- Controllers: Laravel controllers process incoming API calls from the React interface and trigger the corresponding business logic, for example, managing posts with food donations and actions of users.
- Middleware: Controls the flow of authentication, authorization, and validation of requests.
- Services: Handle request, lookup foods, and perform actions related to user interactions and other application core logic.
- Security: Laravel takes care of the security aspects of the web application through such components as authentication, authorization, password hashing.

**3. Data Layer** This layer actually handles data storage, data retrieval, and data persistence using a relational database system (MySQL).

- Database: Designed for storing user data, donation posts, request records, and transaction history.

- File Storage: Laravel file system is used for uploading and retrieving food images from the user; cloud storage can also be integrated.

## 5.2 Class Responsibilities Collaboration (CRC) Cycle

A **Class Responsibilities Collaboration (CRC) cycle** is a simple and effective way to model object-oriented systems. It identifies classes in the system, their responsibilities, and their collaborations.

**Class: DonationPost**

- Responsibilities:
  - Manage donation post details such as post_name, post_description, serves, etc.
  - Link donation posts to requests using donationRequests().
- Collaborations:
  - Collaborates with DonationRequest to manage requests for food donations.
  - Collaborates with PostImage for managing images attached to donation posts.

**Class: DonationRequest**

- Responsibilities:
  - Handles requests for donation posts with fields like donation_id, donee_id, time, etc.
  - Track donation request status (run_status, accept_status).
  - Collaborate with the DonationPost model to get details about the post being requested.
- Collaborations:
  - Collaborates with DonationPost to associate the request with the post.
  - Could collaborate with Rider (not defined yet) for tracking delivery.

**Class: Notification**

- Responsibilities:
  - Manage notifications related to the donation process.
  - Handle notification status (e.g., read/unread) for a user.
- Collaborations:
  - Collaborates with User to send notifications to specific users.

**Class: orgInfo**

- Responsibilities:
    - Manages organization details such as org_name, office_time, and admin_approval
    - Ensures organizations are properly approved before creating donation posts.
- Collaborations:
    - Collaborates with User model for admin approval.

**Class: PostImage**

- Responsibilities:
    - Manage images associated with donation posts using image_path.
- Collaborations:
    - Collaborates with DonationPost to link images to the corresponding post.

**Class: Rating**

- Responsibilities:
    - Stores ratings for donation requests, linked to a specific request (req_id).
- Collaborations:
    - Collaborates with DonationRequest to link ratings to requests.

**Class: RequestDonation**

- Responsibilities:
    - Manages donation requests and stores request status (accept_status, run_status).
    - Tracks delivery (delivery) for each request.
- Collaborations:
    - Collaborates with DonationPost and User for managing food donation requests.

**Class: RiderApproval**

- Responsibilities:
    - Manages approval for riders who will handle food deliveries.
- Collaborations:
    - Collaborates with Rider and Admin (if created) for approval workflow.

**Class: RiderAvailability**

- Responsibilities:
  - Manages rider availability for pickup and delivery services.
- Collaborations:
  - Collaborates with Rider to track rider availability for requests.

**Class: User**

- Responsibilities:
  - Manage user authentication (login, register, change password, etc.).
  - Handle user profile data (name, email, password).
- Collaborations:
  - Collaborates with Notification for sending and receiving notifications.
  - Collaborates with DonationPost and DonationRequest for user-donation interactions.

## 5.3 Class Diagram

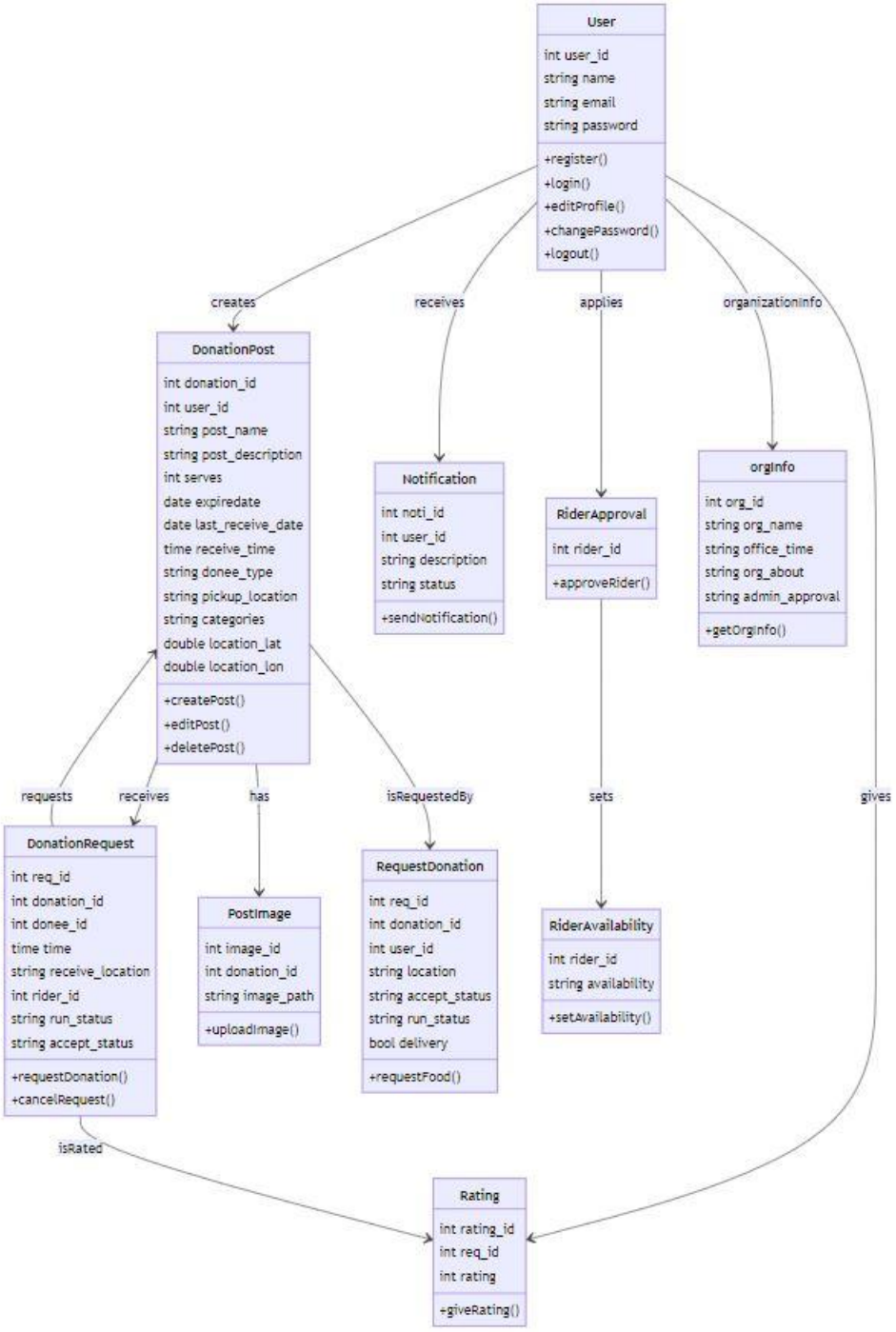



*Figure 32: Class Diagram*

## 5.4 Detailed Design and Component Section

Here is a high-level overview of the Detailed Design and Component Selection:

**1. Front-end Development:**

- I selected React.js as the front-end framework to build the user interface of the web application.
- I utilized HTML, CSS and Bootstrap to create the visual elements, layout, and interactive components.
- I implemented responsive design techniques to ensure the application is accessible on various devices and screen sizes.

**2. Back-end Development:**

- I chose Laravel as the runtime environment for the server-side development.
- I integrated MySQL as the database to store and manage the application's data.

**3. Authentication and Authorization:**

- I implemented user authentication using JWT (JSON Web Tokens) for secure access to the application.
- I utilized ‘hash’ for password hashing to ensure user credentials are securely stored.

**4. External APIs and Integrations:**

- To display the donee or donors or both locations, I integrated MapBox MAP API.
- To handle upload and data processing some react libraries and packages are used in this project.

## 5.5 Database Design Diagram

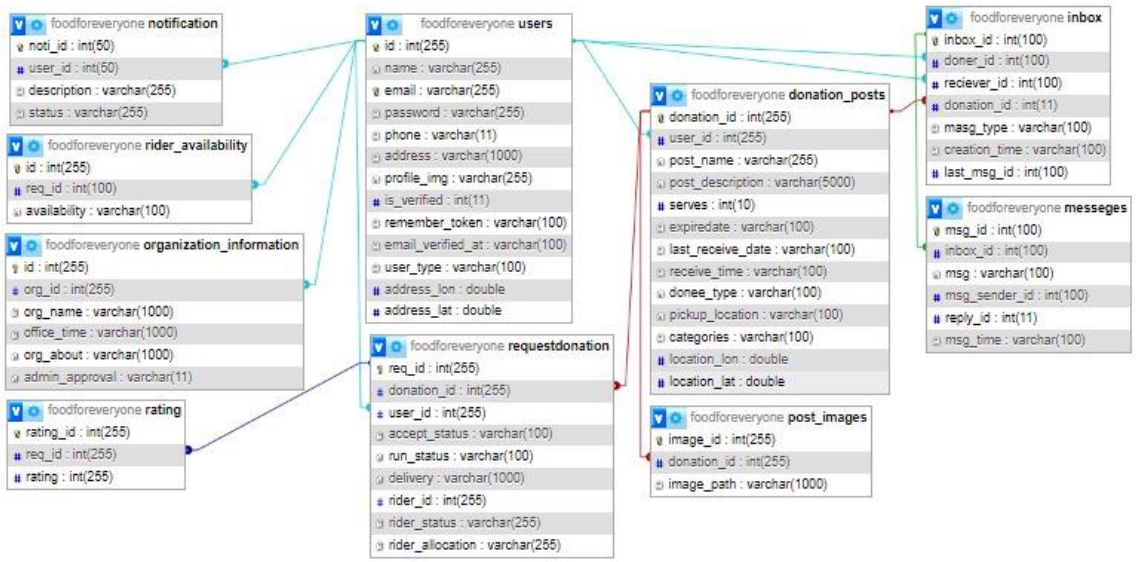



*Figure 33: Database Design*

## 5.6 User Interface Design

Some methods and strategies that can be employed when implementing the Food for Everyone System's User Interface (UI) design is-

User-Centered Design: The user interface (UI) of the system should be designed with the users' needs, and preferences, and behavior as the top priority. To ensure that the interface meets their requirements, the following must be done: The users must be studied, their opinions must be sought and they should be involved in the designing of the interface.

Consistent and Intuitive Navigation: The act of ensuring that the navigational components like menu bars, buttons and links in the system bear resemblance.

Error Handling and Feedback: Ensure system returns informative error messages that guide users in resolving challenges easily. Perform input validation in real-time and return appropriate feedback.

Interactive Elements: I used buttons, forms, dropdowns, tooltips – it gave users more options for engagement and offered friendly interactions.

Visual Design and Branding: In my work, I used the same color up and typeface throughout the session, and icons and illustrations that helped me to leave a lasting impression.

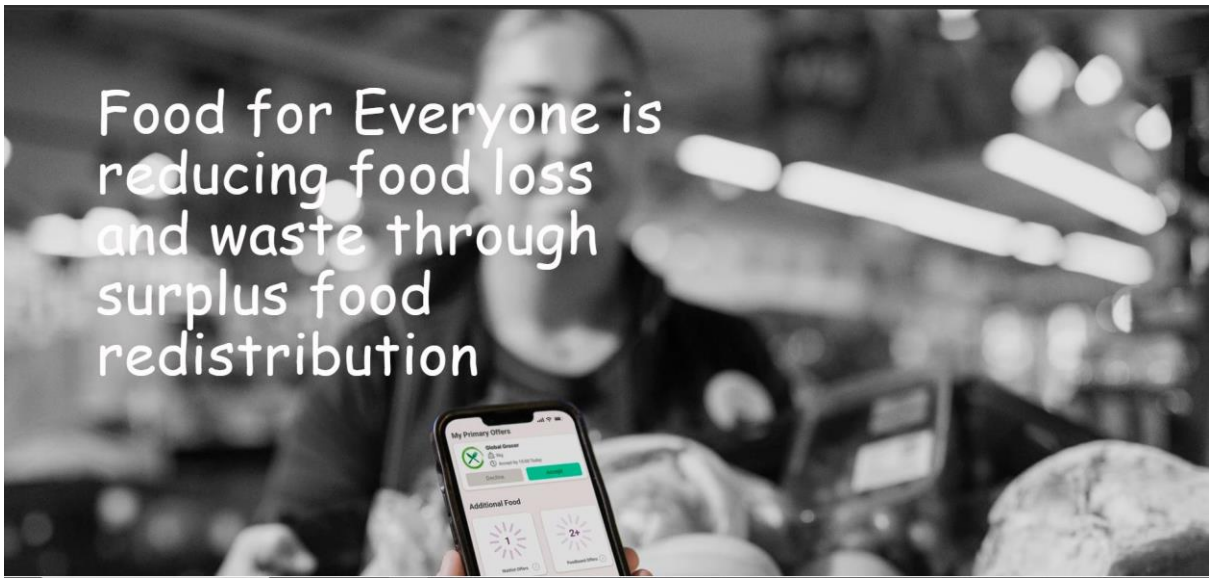



*Figure 34: Home Page Design 1*

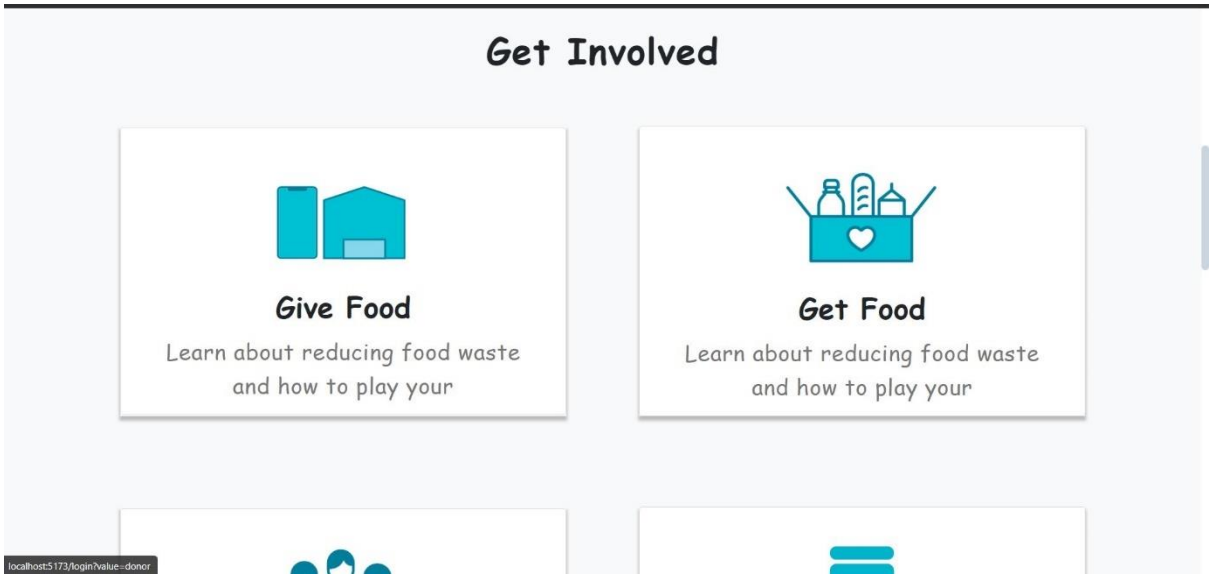



*Figure 35: Home Page Design 2*

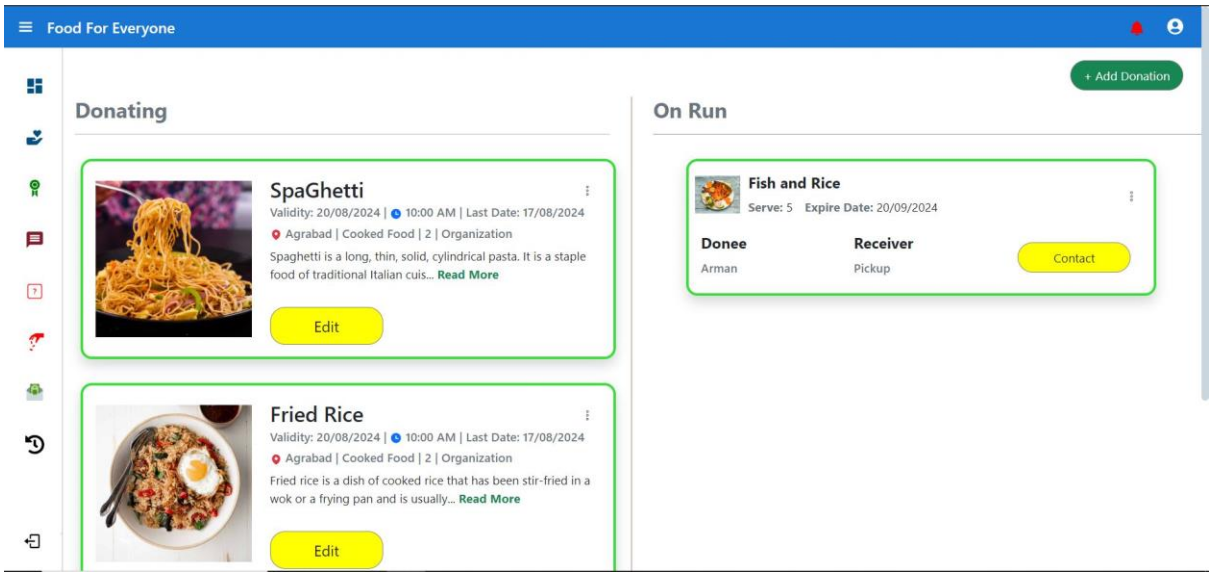



*Figure 36: Donor Dashboard Design*

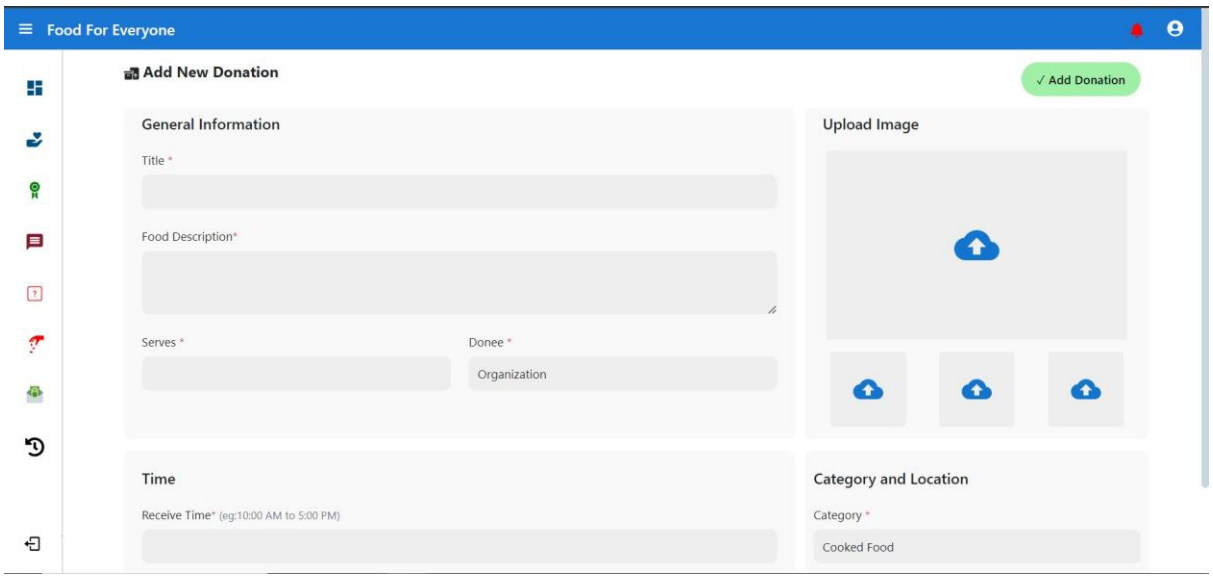



*Figure 37: Add New Donation Design*

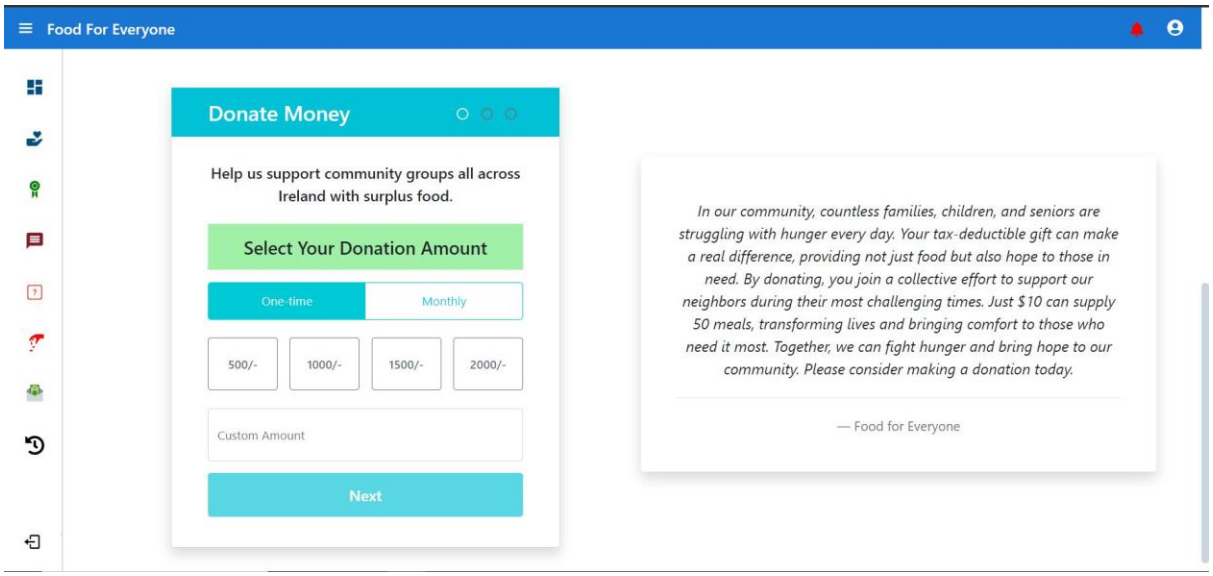



*Figure 38: Donate Money Design*

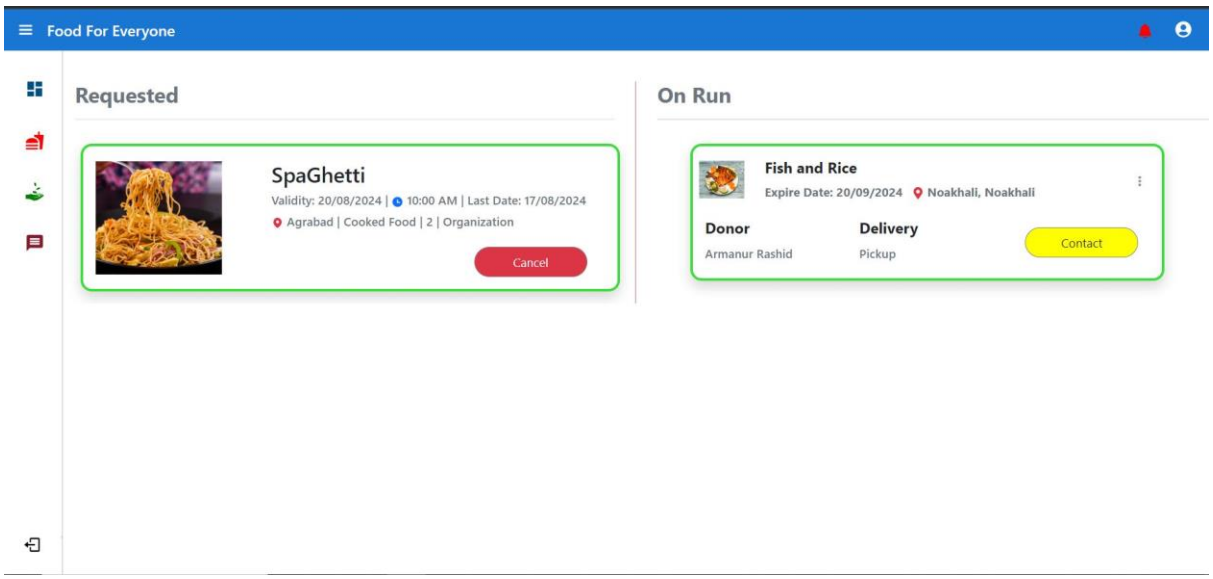



*Figure 39: Donee Food Cart Design*

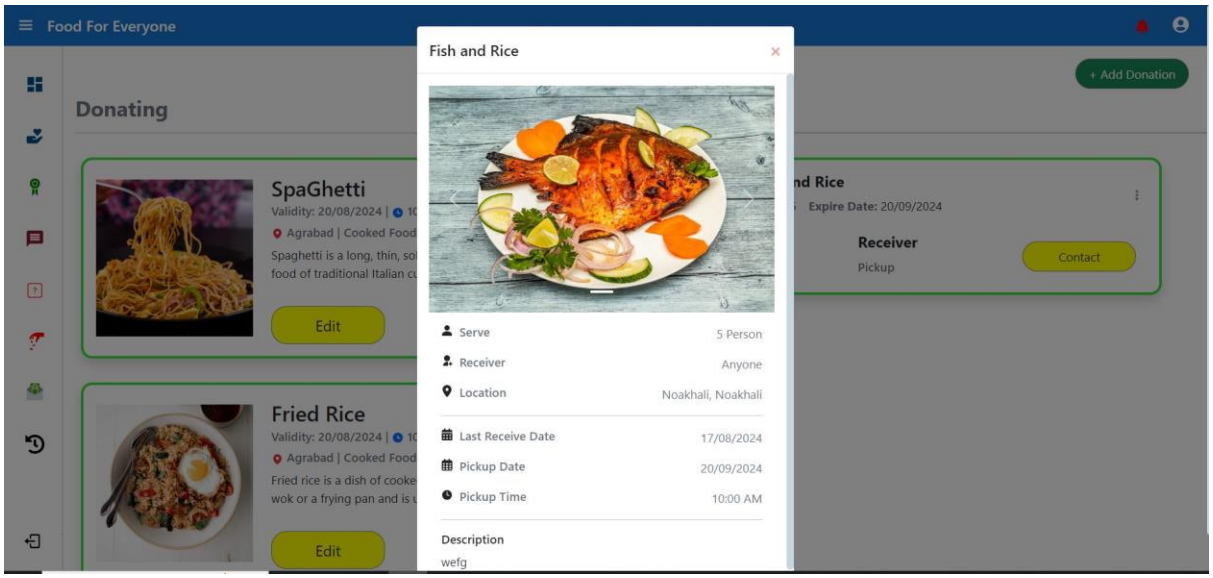



*Figure 40: Donation Details*

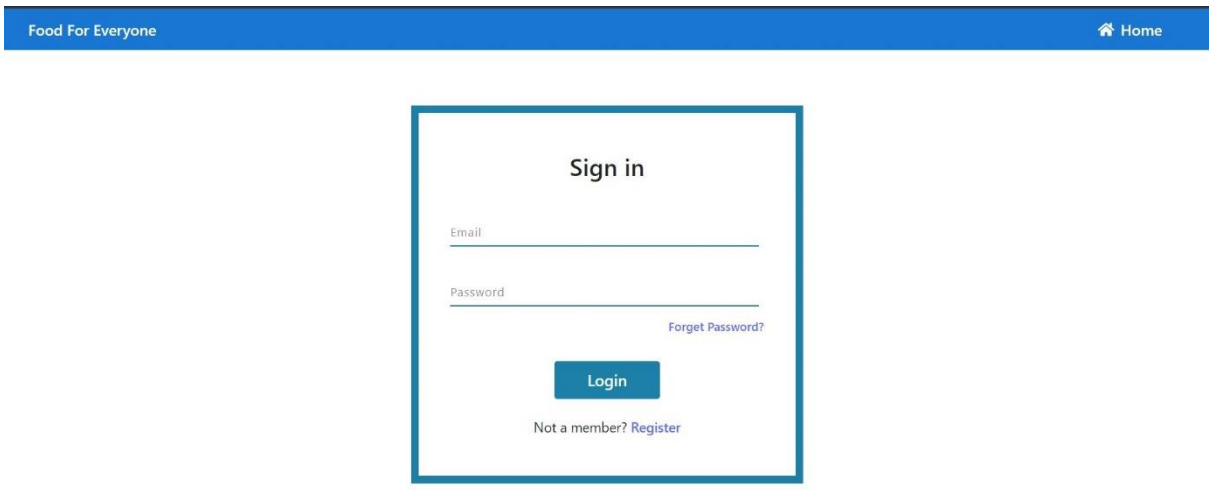



*Figure 41: Login Page Design*

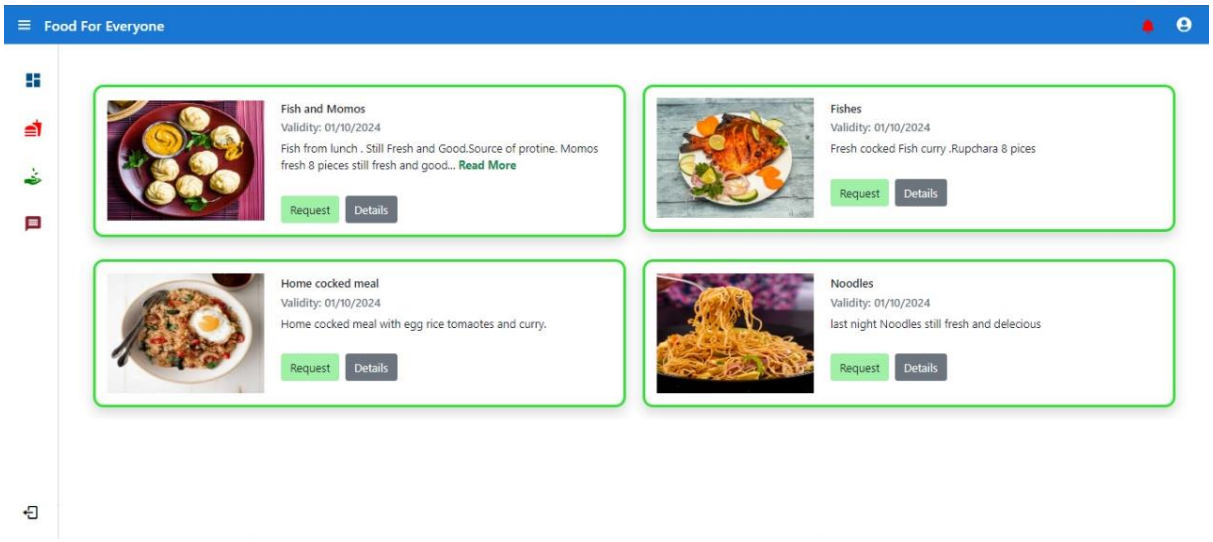



*Figure 42: Donee Dashboard Design*

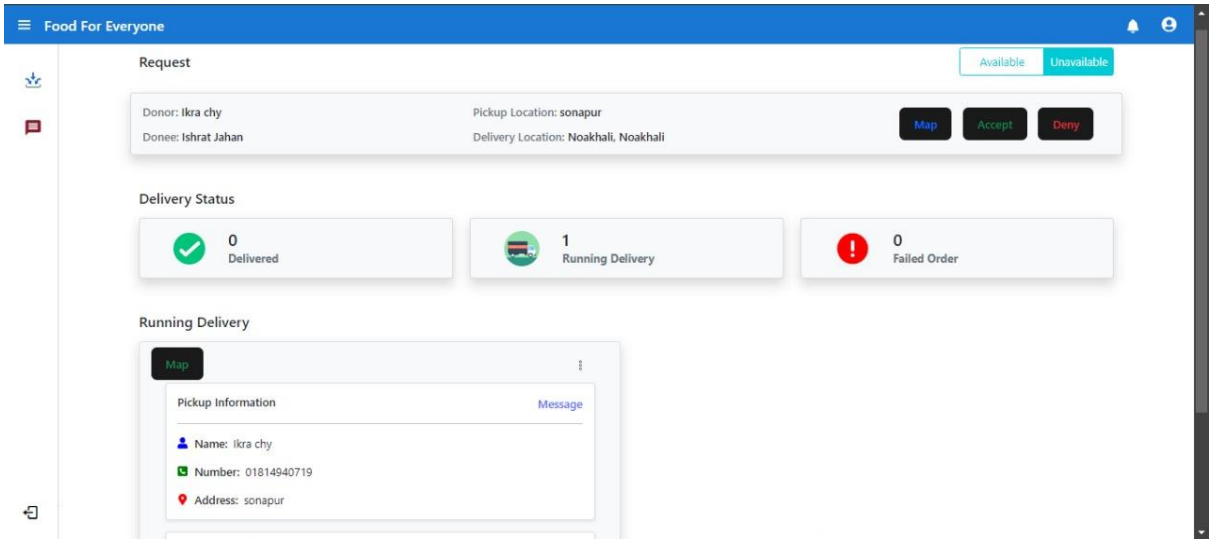



*Figure 43: Rider Dashboard*

# Chapter Six: Implementation and Development

## 6.1 Development Tools & Technology

The “Food for Everyone” was installed and developed using a range of tools and technologies to produce a robust and effective online solution. The development tools and technologies utilized for the project are summarized in this section. The language and tools that have been used to develop this project are as follows:

- Visual Studio Code
- PHP
- HTML
- Cascading Style Sheets (CSS)
- ReactJS
- Xampp
- MYSQL
- Laravel and Bootstrap as a framework

### 6.1.1 User Interface Technology

For the development of this application, I utilized popular development tools and technologies such as VS Code, React, React Router, CSS, Bootstrap, MySQL. These tools and technologies provided a robust foundation for building the application's frontend components, ensuring efficient development and seamless functionality.

#### 6.1.1.1 Laravel Framework

The web application is built using the Laravel Framework. An open-source PHP framework called Laravel uses the Model-View-Controller (MVC) design principle. Routing, Eloquent ORM for database interactions, Blade templating engine for separating presentation logic, middleware for filtering HTTP requests, authentication and authorization systems, caching for improved performance, task scheduling for automating recurring tasks, key features and concepts of Laravel.

#### 6.1.1.2 CSS Framework or Bootstrap

For the project I used CSS for user interface designing. It is the standard stylesheet language used to define the visual appearance of HTML documents. It allows me to control the layout, colors, fonts, spacing, and other visual aspects of web pages.
I also use Bootstrap to make my project's user interface visually beautiful and responsive. I can easily prototype and develop a beautiful layout using the pre-built CSS and JavaScript components in Bootstrap. Its typography, forms, buttons, grid system, and other UI elements make it simple for me to design an interface with a clean appearance. Because of Bootstrap's

comprehensive documentation and active community, I can modify and improve the design to meet the needs of my project.

#### 6.1.1.3 React Icon

React Icon was utilized in the " Food for Everyone " web application to enhance the visual appeal and user experience by incorporating a wide range of scalable vector icons. These icons were employed across various components and features to provide intuitive navigation, improve information representation, and enhance overall aesthetics.

### 6.1.2 Implementation Tools & Platforms

The implementation technologies and platforms used for the project will be determined by a number of factors, including project needs, team competence, scalability, and budget.

#### 6.1.2.1 Visual Studio Code

Microsoft created Visual Studio Code (VS Code), a simple open-source source code editor. It supports a large number of programming languages. It provides various features and extensions that improve my development process. I can efficiently develop, edit, and debug my code using this. It is appropriate for working on my Laravel and React based project.

#### 6.1.2.2 MySQL Server

MySQL was utilized as the relational database management system (RDBMS) for the "Food for Everyone" web application. It provided a robust and scalable database solution for storing and managing various data related to students, supervisors, project committees, notices, and project progress.

#### 6.1.2.3 Apache HTTP Server

A popular open-source web server known for its performance, scalability, and security is Apache HTTP Server. It offers customizable configuration options, supports a variety of operating systems, and integrates with modules for greater functionality.

## 6.2 Coding Practice and Standards

- **Consistent Code Formatting**: When developers follow a consistent code formatting style, the code is easier to read, understand, and maintain. In order to improve code readability, use consistent indentation. Indentation is usually done with spaces or tabs.

- **DRY (Don't Repeat Yourself) and modular Code**: Modularizing the code's components encourages reuse and maintainability. Avoid writing monolithic code that combines unrelated topics or performs many activities.

- **Meaningful Variable and Function Names**: Giving variables, functions, and classes names that are descriptive and meaningful increases code readability. Avoid using names that are cryptic or unclear, as these may make the code difficult to understand.

- **Comment**: For making anyone understand, comment has been used in views and controller files.

- **Handling Errors and Exceptions**: Implement effective error handling strategies to elegantly handle predicted and unexpected problems.

## 6.3 Version Controlling

Version control is a crucial component of software development. Git enables you to communicate with other developers, track and manage changes to your codebase, and quickly roll back to earlier versions if necessary. To manage the source code and track changes of this project, Git (GitHub) is used for this project.

## 6.4 Integration and Development Strategies

**1. API Integration**:

- MapBox API: Use the services of MapBox in the process of location-based services that will enable easy pickup and delivery services. Make sure that the application can identify long locations and give the right directions to riders.

- Made necessary API in the backend using Laravel. Which all are integrated in the frontend using ReactJS to functionable the system.

**2. User Verification Systems**:

- Implement a secure verification system using email or mail verification to ensure that all users are legitimate.

- Set up encryption for important information and comply with local data privacy.

**3. Data Management and Security**:

- Use a strong database management system to store user profiles, donation posts, requests, and other information.

- Implement encryption for sensitive data and ensure compliance with local data protection rules.

**Development Strategies**

**1. Agile Development Approach**: Applied Agile development approach to allow for flexibility, regular feedback, and iterative improvements based on user input.

**2. User-Centered Design:**

Focus on creating a user-friendly interface by conducting user research and usability testing. Gather feedback from potential users to refine the design and functionality.

**3. Feature Prioritization:**

Prioritize essential features that directly support the core functionality of the platform. Some of the interactions are as follows:

- User registration
- Food posting
- Requesting for a service
- Notification

# Chapter Seven: Testing and Quality Assurance

## 7.1 Testing Features

The various modules of this system at time stated their performance and operation statuses. Key modules that are frequently examined in such systems include the following:

### 7.1.1 Features to be tested

- User Registration and Login
- Mail Verification
- Create Food Donation Post
- Edit Food Donation Post
- Delete Post
- Search
- Rating
- Pickup and Delivery Service
- Notification
- Change Password
- Change Address
- Change Phone Number
- Forget Password
- Message
- Set Location
- Direct Donation
- Request Food Donation
- Cancel Request
- Rider Request
- Accept Riding Request
- Cancel donation
- Admin Control
- Log out

### 7.1.2 Features not to be tested

- **Libraries from Third Parties**: I think it's possible that other libraries or frameworks such JavaScript, Bootstrap, and Django have each been carefully reviewed by the developers that created them. I can therefore omit testing these libraries' fundamental functionality.
- **External Services**: I might not need to test the functionality of external services because my project communicates with their APIs. Testing the integration points and making sure my app can handle responses and errors well are still necessary, though.

## 7.2 Testing Strategies

It is therefore important to conduction testing on an “Food for Everyone” to determine the current state to ascertain whether it is fully functional, reliable and performs to its optimum level or not. Different approaches and techniques for ensuring complete testing of systems may be included in the project test strategies:

### 7.2.1 Test Approach

Both Blackbox and Whitebox testing methodologies will be required to conduct a test procedure for this project. Based on the testing approach, the testing of Blackbox will conspire of elements such as system testing and user acceptance testing. It means that principles of Whitebox testing also implies checking internal structure and code of the system to guarantee its purity and compliance with coding standards. For my project, take into account the following strategy:

**Unit testing**: Involves analyzing specific components and processes to verify their functionality.

**Integration testing**: Checking the unity and the interaction of modules and their parts with each other.

**System testing**: Perform tests on the entire system to check the system’s performance and its behavior.

### 7.2.2 Pass/Fail Criteria

Given the anticipated actions and outcomes outlined in the project specifications. We will determine pass or fail benchmarks. In each test scenario there should be a definition set out for evaluation purposes. A test scenario is deemed successful when it runs smoothly without any issues or straying from the projected outcomes. If significant issues arise or the expected outcomes are not met as planned out in the scenario guidelines then it will be classified as unsuccessful.

### 7.2.3 Suspension and Resumption

It will be necessary to note that there could be conditions during the test cycle when tests have to be put on hold because of their dependency or issues. Otherwise, sufficient documentation shall be maintained to support recording of additional information in relation to the suspension including reasons for suspension, expected resumption date and any pending activity.

### 7.2.4 Testing Schedule

Planning and allotting time for various testing tasks is necessary when creating a testing schedule for the project. Depending on the project's duration, available resources, and the

system's complexity, the precise testing schedule could change. However, the project's testing plan is outlined generally as follows:

**Testing Planning Phase**

- Define the goals, parameters, and requirements for the test.
- Describe the testing methodology, tactics, and strategies.
- Make the test plan and get the required permissions.

**Test preparation Phase**

- Determine which test scenarios and test cases should come first.
- Produce or collect test data.
- Create the testing infrastructure and environment.
- Create any necessary automation frameworks or test scripts.

**Test execution Phase**

- Run test cases in accordance with the top-priority test scenarios.
- Maintain a defect log and status update.
- Perform performance testing, integration testing, functional testing, and other sorts of testing that are pertinent.
- Work together with the development team to remedy issues and test remedies again.

**Phase of Test Reporting**

- Review test findings and provide a report on the system's overall quality.
- Any unresolved problems or dangers should be documented and shared.
- Give stakeholders test metrics and progress reports.
- Conduct test closing activities, including final documentation and lessons learned.

### 7.2.5 Traceability Matrix

| Project Name: Ikra Chowdhury | | **Requirement Traceability Matrix** | | Business Area: Social Welfare Project | |
|---|---|---|---|---|---|
| Project Manager: Ikra Chowdhury | | | | Business Analysist Lead: Ikra Chowdhury | |
| QA Lead: Ikra Chowdhury | | | | Implementation Date: September 26, 2024 | |
| **Requirements** | **Priority** | **Requirement description** | **Test Case Reference** | **Validation** | **Comments** |
| FR-1 | High | User Registration and Login to a registered account | TC01<br>TC02<br>TC03<br>TC21 | Verified | Covered |
| FR-2 | High | Donor can create a new food donation post | TC05 | Verified | Covered |
| FR-3 | Medium | Edit Food Donation Post | TC06 | Verified | Covered |
| FR-4 | Low | Donor can delete his/her food donation post | TC07 | Verified | Covered |
| FR-5 | Medium | Search post according to user preference | TC08 | Verified | Covered |
| FR-6 | Low | Rating the food | TC10 | Verified | Covered |
| FR-8 | Medium | Notification system for important event | TC11 | Verified | Covered |
| FR-7 | High | Pickup and delivery service for food | TC16 | Verified | Covered |
| FR-9 | High | Change user password | TC09 | Verified | Covered |
| FR-10 | High | Forget Password | TC18 | Verified | Covered |
| FR-11 | Low | Edit user profile | TC19 | Verified | Covered |
| FR-12 | High | Set Location | TC26 | Verified | Covered |
| FR-13 | High | Message | TC27 | Verified | Covered |
| FR-14 | Low | Direct donation | TC28 | Verified | Covered |
| FR-15 | High | Request Food Donation | TC24 | Verified | Covered |
| FR-16 | High | Cancel food donation request | TC14 | Verified | Covered |

| | | | | | |
|---|---|---|---|---|---|
| FR-17 | Medium | Accept riding request | TC25 | Verified | Covered |
| FR-18 | High | Cancel donation | TC17 | Verified | Covered |
| FR-19 | High | Admin Control | TC15<br>TC23 | Verified | Covered |
| FR-20 | Medium | Log out | TC20 | Verified | Covered |

### 7.3 Testing Environment

**Hardware Requirements**

- CPU: Intel Core i5 or equivalent
- RAM: 8 GB or Higher
- Storage: 256 GB Hard Disk

**Software Requirements**

- Operating System: Windows 10
- Python: Version 3.9
- Web Browser: Google Chrome

## 7.4 Test Cases

| Test Case ID | Test Case Description | Test Steps | Input Data | Expected Result | Actual Result | Pass\ Fail |
|---|---|---|---|---|---|---|
| **TC01** | User Registration | 1. Open registration page<br>2. Enter valid details<br>3. Click register | Name: Ikra<br>Email: ikra@gmail.com<br>Password: 123456 | User registered | As expected | Pass |
| **TC02** | Invalid User Registration (Missing Fields) | 1. Open registration page<br>2. Leave required fields blank<br>3. Click register | Email: ikra@gmail.com | Error message: "All fields are required" | As expected | Pass |
| **TC03** | User Login | 1. Open login page<br>2. Enter valid credentials<br>3. Click login | Email: ikra@gmail.com<br>Password: 123456 | User logged in and redirected to dashboard | As expected | Pass |
| **TC04** | Forgot Password | 1. Click "Forgot Password" | Email: ikra@gmail.com | Password reset link | As expected | Pass |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | 2. Enter email<br>3. Submit | | sent to email | | |
| **TC05** | Create Donation Post | 1. Log in as donor<br>2. Navigate to donation post page<br>3. Fill in details<br>4. Submit post | Post Name: Surplus Bread<br>Description: 10<br>Expiry Date: 2024-09-28 and other information | New donation post created and visible on the feed | As expected | Pass |
| **TC06** | Edit Donation Post | 1. Log in<br>2. Navigate to donation post<br>3. Click edit<br>4. Change details<br>5. Submit | Change description to 8 | Post updated with new details | As expected | Pass |
| **TC07** | Delete Donation Post | 1. Log in<br>2. Navigate to donation post<br>3. Click delete | Post ID: 12 | Post successfully deleted | As expected | Pass |
| **TC08** | Search Donation Posts | 1. Navigate to search page<br>2. Enter search criteria<br>3. Click search | Search: "bread" | Relevant posts are shown | As expected | Pass |
| **TC09** | Change Password | 1. Log in<br>2. Navigate to profile<br>3. Click "Change Password"<br>4. Enter new password<br>5. Submit | New Password: 7891011<br>Confirm Password: 781011 | Password successfully changed | As expected | Pass |
| **TC10** | Rating System | 1. Log in as receiver<br>2. Navigate to completed requests<br>3. Leave rating | Rating: 5 stars | Rating recorded for the donation | As expected | Pass |
| **TC11** | Notification for New Request | 1. Log in as donor<br>2. Submit donation post<br>3. Wait for | Donation ID: 12 | Notification received for new request | As expected | Pass |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | request<br>4. Check notifications | | | | |
| **TC12** | Accept Donation Request | 1. Log in as donor<br>2. Go to request page<br>3. Accept a request | Request ID: 9 | Request accepted | As expected | Pass |
| **TC13** | Rider Request | 1. Donor accepts donation request<br>2. System assigns rider<br>3. Check rider status | Donation ID: 12 | Rider assigned for delivery | As expected | Pass |
| **TC14** | Cancel Donation Request | 1. Log in as receiver<br>2. Navigate to active request<br>3. Click cancel | Request ID: 9 | Request successfully canceled | As expected | Pass |
| **TC15** | Admin Control: Approve Organization | 1. Log in as admin<br>2. Go to organization requests<br>3. Approve request | Org ID: 2 | Organization approved and notified | As expected | Pass |
| **TC16** | Set Pickup Location | 1. Log in as donor<br>2. Create new donation post<br>3. Set pickup location on the map<br>4. Submit | Location: Latitude: 23.8103, Longitude: 90.4125 | Pickup location set | As expected | Pass |
| **TC17** | Cancel Donation | 1. Log in as donor<br>2. Navigate to active donations<br>3. Click cancel on active donation | Donation ID: 12 | Donation post successfully canceled | As expected | Pass |
| **TC18** | Forgot Password with Invalid | 1. Click "Forgot Password"<br>2. Enter invalid | Email: ikr@gmail.com | Error message: "Email not | As expected | Pass |

| | | | | | | |
|---|---|---|---|---|---|---|
| | Email | email<br>3. Submit | | found" | | |
| **TC19** | Edit Profile Information | 1. Log in<br>2. Go to profile page<br>3. Update personal details<br>4. Submit | New Name: Jane Doe | Profile updated successfully | As expected | Pass |
| **TC20** | Log Out | 1. Log in<br>2. Click log out button | | User successfully logged out | As expected | Pass |
| **TC21** | Failed Login (Incorrect Password) | 1. Open login page<br>2. Enter valid email but incorrect password<br>3. Click login | Email: ikra@gmail.com<br>Password: 132457655 | Error message: "Invalid credentials" | As expected | Pass |
| **TC22** | View Donation Post Details | 1. Log in as receiver<br>2. Navigate to donation feed<br>3. Click on a donation post | Donation ID: 1 | Donation post details displayed | As expected | Pass |
| **TC23** | Admin Control: Approve Rider Request | 1. Log in as admin<br>2. Go to rider approval page<br>3. Approve pending rider | Rider ID:7 | Rider account approved | As expected | Pass |
| **TC24** | Request Donation as donee | . Log in as receiver<br>2. Search for donation posts<br>3. Request food donation | Donation ID: 23<br>Receiver ID: 9 | Food donation request successfully submitted | As expected | Pass |
| **TC25** | Rider Accepts Delivery Request | 1. Log in as rider<br>2. Navigate to delivery requests<br>3. Accept delivery | Delivery ID: 5 | Delivery request accepted and status updated | As expected | Pass |
| **TC26** | Set Location | 1. Navigate to the profile page. | Latitude: 23.8103, | location successfully | As | Pass |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | 2. Click on the "Change Address" option.<br>3. Select a location | Longitude: 90.4125 | set | expected | |
| **TC27** | Message | 1. Select a user<br>2. write message<br>3. click send button | “Hello” | Message Send Successfully | As expected | Pass |
| **TC28** | Direct donation | 1. Visit direct donation page<br>2. Select Organization<br>3. Select Contact | Organization ID: 2 | Direct Donation to the organization | As expected | Pass |

# Chapter Eight: User Manual

## 8.1 User Manual (Donor)

### 1. Add New Donation

- After login Navigate to the Add New Donation page
- Fill up the form with all information
- Click the "Submit" button to post

### 2. Edit Donation

- Navigate to the dashboard
- Find the post wants to edit
- Click on “Edit” Button
- Change the information
- Click on “Save” Button

### 3. Delete Donation post

- Navigate to the dashboard
- Find the post wants to delete
- Click on “Menu” Icon
- Click on “Delete” Button
- After getting confirmation modal click on “Delete” Button

### 4. Message

- After login Navigate to the Message page
- Select the person want to send Message
- Write message
- Click the "Send" button

### 5. Donate Money

- Navigate to the Donate Money page
- Select the amount or write the amount
- Select the payment method
- Provide the necessary information
- Click the “Donate” Button

### 6. Edit Profile

- After login Navigate to the Edit Profile page
- Select the field wants to change
- Provide the necessary information

- Click the “Change” Button

**7. Manage Donation**

- After login Navigate to the Request page
- Click on the post “See all request” button
- Accept the request donor wants to.

## 8.2 User Manual (Donee)

**1. Request for Donation**

- After login Navigate to the Dashboard
- Find the post want to send request
- Click on “Request” button

**2. Cancel Request Donation**

- After login Navigate to the Food Cart page
- Find the post want to cancel request
- Click on “Cancel” button

**3. Message**

- After login Navigate to the Message page
- Select the person want to send Message
- Write message
- Click the "Send" button

**4. Give Rating**

- After received the food navigate to Received Donation Page
- Select the post want to give rate
- Click on “Rate” Button
- Give rating and click “Submit” button

## 8.3 User Manual (Rider)

**1. Manage Riding Request**

- Getting riding request from the system
- Select the Accept or Reject button

**2. Mark as Delivered**

- Navigate to the dashboard
- Select the “Mark as delivered” button

### 3. Message

- After login Navigate to the Message page
- Select the person want to send Message
- Write message
- Click the "Send" button

## 8.4 User Manual (Admin)

### 7. Manage User

- After login Navigate to the Request
- Check the user request
- Take the action of approve or reject.

# Chapter Nine: Project Summary

## 9.1 GitHub Link

https://github.com/Ikrachowdhury/food_for_everyone

## 9.2 Critical Evolution

The “Food for Everyone” project focuses on food waste problems in Bangladesh with an online food donation system where food contributors can find those in need. The key features of the application make the donation process much easier, allowing people and companies to donate the excess stock easily. We have received a number of positive comments from users: they have noticed how easy it was to use the platform, and the difference it makes in engaging the community. Thus, the following evaluation of the project’s weakness and limitations: The first aspect which should be improved within the given project is the data accuracy. The second aspect where further improvement should be made is the addition of new features to make the usage experience richer and more engaging. Since the platform is to be permanently populated with users who will be targeted at promoting the different products and services, it should be stressed that constant interaction with users and their feedback will be upcoming key objectives that will define the further improvements of the platform.

## 9.3 Limitations

Nevertheless, the given topic and problem reveal some limitations within its framework, although the project indicates a high potential for development. One limitation is that the application uses the MapBox API to access location functionalities. While using MapBox, one may not get updated information about certain areas, making it difficult for a user to locate where food donations are made. Further, for the current version of the application, there is no paid API integration applying in the use of the application that mean monetary donations or financially related transactions between the users and organizations. It means that these features may constraints the effectiveness and the range of possibilities of this platform.

## 9.4 Obstacles & Achievements

During the development of the “Food for Everyone” several hurdles were experienced; these include technical difficulties in the interconnection of various technologies, and the matter of user privacy. Nevertheless, these are challenges rated experienced and they were exhaustively dealt with. Some great accomplishments are user interface; easy-to-implement user verification procedures, and designing a ReactJS front-end system connected to a back-end built in Laravel. The project has gained some attention from organizations in the targeted geographical areas as evidenced by its applicability in solving real life problems.

## 9.5 Future Scope

The “Food for Everyone” project has several opportunities for future growth and improvement in the following directions: An innovative future scope includes creating a mobile application to enhance usability and interaction, more precisely to create an application for its mobile version to handle food donations. Moreover, the inclusion of a payment API could also enable users to make monetary donations making the platform more effective in supporting users and organizations. Increased use of location services as a general feature with the addition of enhanced mapping tools will also be added at the helm among other general features which may include improved APIs in helping users locate the donated foods effectively. All in all, steady expansion and feature addition will be vital for reaching the project’s methodological purpose of minimizing food waste in Bangladesh.

# Chapter Ten: Conclusion

## 9.1 Conclusion

Thus, the “Food for Everyone” project can be a great step toward fighting the problem of food waste in the country, as well as helping needy individuals and families in Bangladesh. This is based on the idea that any food that exists in excess should not be wasted but taken to those who need it. By making a web application as friendly as possible for users, we are promoting the sharing and support needed to help others, understanding, and finding some connection even when giving or receiving help.

As a foundation for itself, the platform brings together potential food donors—from individuals to restaurants or other organizations—with potential receivers of the products, such as local welfare groups and people in need of food. It also plays a major role in reducing food waste while ensuring that needy people receive the nourishment they need. When people give extra food to those who can use it, we are also doing our part to create a better and cleaner environment.

As much as we have come a long way, we understand the importance of change and growth. These are priceless gems that have helped us as a team improve on the parts of the platform that users want. We know our current limits, including restrictions in the MapBox API to offer details of the entire geographical world and the lack of a payment API to allow users to donate. Recognizing these factors calls for finding unique ways to further enhance the application and its performance.

To sum up, the “Food for Everyone” project is a new online platform connecting food donors and hungry people and bringing only positive changes to society. Our goal is to end food waste and create a world where everyone who needs it has something to eat. Our way to change the public perception of food waste is to turn it into our product because it’s more valuable than you think. Only through cooperation can we build a better world with a more responsible community that will help every single person get the food they need to live. It only takes the unity of people, creativity, and their desire to make the world a better place to bring positive change to many and promote sustainability.

# References


- IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998
- Laravel 8 documentation https://laravel.com/docs/8.x/readme
- PHP 8.1 documentation https://www.php.net/releases/8.0/en.php
- Bootstrap 5.0 documentation https://getbootstrap.com/docs/5.0/getting-started/introduction
- https://stackoverflow.com