FoodShare: A Surplus Food Donation Platform

Submitted By

Shahriar Arefin

221-35-1018

Supervised By

Mr. Nuruzzaman Faruqui

Assistant Professor

This project report has been submitted in fulfilment of the requirements for the degree of **Bachelor of Science in Software Engineering**

DAFFODIL INTERNATIONAL UNIVERSITY

DECLARATION OF THESIS AND COPYRIGHT

Author's Full Name : Shahriar Arefin
Date of Birth : 20/07/2000
Title : FoodShare: A surplus food donation platform
Academic Session : 2022 - 2025

I declare that this thesis is classified as:

☐ CONFIDENTIAL (Contains confidential information under the Official Secret Act 1997)*

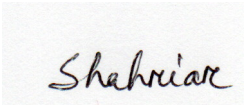
☐ RESTRICTED (Contains restricted information as specified by the organization where research was done)*

☐ OPEN ACCESS I agree that my project to be published as online open access (Full Text)

I acknowledge that Daffodil International University reserves the following rights:

1. The Project is the Property of Daffodil International University.
2. The Library of Daffodil International University has the right to make copies of the Project for the purpose of research only.
3. The Library of Daffodil International University has the right to make copies of the Project for academic exchange.

Certified by:



(Student's Signature)

221-35-1018

Student ID
Date: 24/12/2025

(Supervisor's Signature)

Mr. Nuruzzaman Faruqui

Name of Supervisor
Date: 24/12/2025

NOTE: * If the Project is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

PROJECT DECLARATION LETTER (OPTIONAL)

Librarian,
Daffodil International University,
Daffodil Smart City,
Ashulia, Dhaka, Bangladesh

Dear Sir,

CLASSIFICATION OF Project AS RESTRICTED

Please be informed that the following project is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.
Author's Name
Project Title

Reasons (i)

(ii)

(iii)

Thank you.

Yours faithfully,

_____
(Supervisor's Signature)

Date: 24/ 12/ 2025

Stamp:

Note: This letter should be written by the supervisor and addressed to the Librarian, *Daffodil International University* with its copy attached to the thesis.

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this project and in my opinion, this project is adequate in terms of scope and quality for the award of the degree of Bachelor of Science.

_____

(Supervisor's Signature)

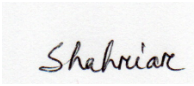
Full Name : **Mr. Nuruzzaman Faruqui**

Position : Assistant Professor

Date : 24/12/2025

STUDENT'S DECLARATION

I hereby declare that the work in this project is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Daffodil International University or any other institution.



_____

(Student's Signature)

Full Name : **Shahriar Arefin**
ID Number : 221-35-1018
Date : 24 December 2025

FoodShare: A Surplus Food Donation Platform

Shahriar Arefin

Project submitted in fulfillment of the requirements
for the award of the degree of
Bachelor of Science

Department of Software Engineering

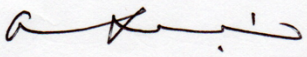
DAFFODIL INTERNATIONAL UNIVERSITY

NOVEMBER 2025

APPROVAL

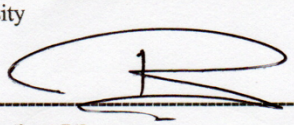
This thesis titled on "**FoodShare: A Surplus Food Donation Platform**", submitted by **Shahriar Arefin (ID: 221-35-1018)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS



Chairman

Dr. A. H. M. Saifullah Sadi
Professor
Department of Software Engineering
Faculty of Science and Information Technology Daffodil International University



Internal Examiner 1

Dr. Rubaiyat Islam
Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 2

Dr. Md. Abdul Kader
Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 3

Nuruzzaman Faruqui
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

External Examiner

Md. Mostafiz Khan
Managing Director
Tecognize Solutions Limited

ACKNOWLEDGEMENTS

I am expressing my sincere gratitude to the Almighty Allah, whose mercy and guidance have enabled me to carry this work from its initial idea to its final form.

I am deeply thankful to my supervisor, **Mr. Nuruzzaman Faruqui**, **Assistant Professor**, Department of Software Engineering, for his continuous encouragement and thoughtful insights throughout this project. The support I got from him consistently helped me understanding and shape the direction of this thesis and I am genuinely indebted for his commitment and support.

I am really blessed to have the respected faculty members of Software Engineering Department. They have guided me and make me motivated in these years that have significantly grew me academically and mentally and prepared me for the upcoming challenges.

Finally, I would like to convey my best regards to my parents as their support, patience and the belief they had in me have helped me a lot throughout this academic journey. Without them I would not be able to be this motivated and would not be able to complete this milestone.

DEDICATION

I therefore declare that I have done this project under the oversight of "**Mr. Nuruzzaman Faruqui," Assistant Professor,** Department of Software Engineering, Daffodil International University. I also declare that neither the entire record nor any portion of this record has been submitted somewhere else for my degree.

ABSTRACT

The worldwide problem of food waste generates massive amounts of discarded surplus food while numerous people struggle to access sufficient nutrition. The problem reaches its peak intensity in cities. The FoodShare project develops a complete web application that enables real-time food donation connections between surplus food providers and charitable organizations that distribute food to those in need. The platform is structured with React (Vite) for frontend development and Firebase for backend operations to build its modern serverless infrastructure and for easy maintaining. The system has a secure access control system that allows donors, NGOs, and admins to use the platform through their given roles by the system and admin. The platform enables users to authenticate through email or Google while providing a donation submission form that uses Google Maps APIs to add precise location information. The admin dashboard is the place for approving or rejecting donations after reviewing and moderation. The system operates through a protected donation process that starts with submission and ends with claiming and it is structured by Firestore rule structure. The project presents an affordable method to solve this essential social issue by using technological solutions to optimize food rescue operations

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Description
API	Application Programming Interface
CDN	Content Delivery Network
CSS	Cascading Style Sheets
ER	Entity-Relationship
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
NGO	Non-Governmental Organization
SDK	Software Development Kit
SQL	Structured Query Language
UI	User Interface
UML	Unified Modeling Language

UX	User Experience

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

All over the world food waste has become a more serious issue especially in Bangladesh and other rapidly developing nations. Households, restaurants, hotels, event spaces and food service enterprises throw away a significant amount of usable food every day. Meanwhile, a sizable section of the general population still faces food insecurity. This disparity draws attention to an imbalance between excess and scarcity that endures primarily because there are no formalized processes for redistributing food.

The issue is even worse in cities like Dhaka. A growing amount of surplus food is a result of fast-paced lifestyles, frequent social events, and growing commercial food services. Despite being safe to eat the majority of this food ends up in waste dumps harming the environment and increasing the emission of greenhouse gases. Many people and organizations say they want to donate the extra food and don't want to waste them. However, the process is inconsistent and ineffective because there is no current platform or any other things.

Several non-profit and charitable groups are working to eradicate this hunger as there are many food wasting daily in their areas. However, they often depend on informal communication methods such as phone calls, social media posts, and personal contacts. These methods are slow and not always reliable and not everyone is willing to take those hassles. They do not guarantee quick access to available food. Similarly, there is no structured way for donors to find trustworthy organizations that can safely collect, inspect, and distribute food.

Taking this challenges into account and also as the technology has proven as a valuable tool over the time for improving communication and problem solving between NGOs and food donors. A well-organized digital platform can facilitate real-time interactions, enhance coordination and promote transparency that relies in the food donation process. Furthermore, by verifying users donations and ensuring that the donated food reaches the correct peoples and the society benefits from it.

The necessity to reduceing food waste, boost the efficiency of food distribution and increases the capabilities of technology to aid social initiatives in Bangladesh serves as the foundation for this study.

1.1.1 Context and Relevance

The presence of huge food waste and overall food unstabilization is a grave social and economic problem in Bangladesh and also in some of the countries. Restaurants, event venues and even households also waste large amounts of perfectly safe and edible food. Ignoring the requirements of vulnerable people this waste contributes to environmental problems.

1.1.2 Problem Identification

The key issue is a disconnection between information and a proper channel. Donors who have excess food have no fast, simple, easy and reliable means of contacting organizations that can collect it. The charities and the organizations that work in this respected field on the other hand find it hard to find the sources and think of this are all hassles of food donation that are reliable and safe. Such informal methods as phone calls or posts on social media cannot be reliable, effective and lack an accountability and verification.

1.1.3 Purpose and Justification

FoodShare project through development of real time web application is expected to overcome this social problem. This is the platform that will play crucial role as a virtual connection between registered NGOs and identified donors. The project goal is to ease this process in order to reduce food waste and delivered by a trusted, dependable supply chain to eradicate hunger. Which will transform into a concrete social difference.

1.1.4 Scope

The scope of this project includes web application and development also the implementation of a full stack web application. It offers three types of user accounts. Includes NGOs who can claim the donations. Donors who submit a donation. Administrators who can moderate the site such as changing roles and approving donations or rejecting it. The main functionality includes user authentication, a donation submission form, an admin page, a running record of available donations to the NGOs that filters by location distance and profile pages allowing users to view their history.

1.2 Project Planning and Initiation

Feasibility Study (Step-by-Step)

Phase 1 Preliminary Analysis & Project Scope Definition:

- **Objective:** To create a web platform for handling the donations of surplus food.
- **Scope:** Three distinct user roles are—Donor, NGO and Admin—within the system. The Donor logs in for a donation and submits it for verification -> The Admin verifies it and either approves or rejects the donation -> The NGO then searches, views and claims the approved donation, which is the primary flow of the platform.

Phase 2 Market Feasibility Analysis (or Market Research):

- **Stakeholder Needs:** Contributors desire a straightforward, short time, low hassle and expedited submission process. NGOs require a trustworthy and up-to-date list of confirmed donations ideally in the area.
- **Market Gap:** The gap in the target market not yet have a specialized, moderated, real-time platform for this service. The current solution is both slow, unofficial and not digitalized.

Phase 3 Technical Feasibility Analysis:

- **System Architecture Overview:**
    - **Frontend:** Built with **React (Vite)**
    - **Backend:** Powered entirely by **Firebase**
    - **Database: Cloud Firestore**
    - **Authentication & Security: Firebase Authentication**
    - **Location Services:** The Google Maps Platform is implemented to provide an address autocomplete search box for donors to ease things and to enable location-based distance filtering for NGOs.

Phase 4 Financial Feasibility Analysis:

- **Development Cost:** Low. The main cost is this is a university project and the only cost is the development time. The software and tools used to develop this project are VS Code, React, Vite, Tailwind, Daisy UI, Firebase CLI and Git free to use.
- ● **Operational Expenses:** Really low. There is free plan called Spark. It provides a free tier for database to read and write operations, authentication and hosting. It gives monthly free credit of 200$ by the Google Maps Platform. For development, testing, and early user adoption, these free tiers are more than adequate.

1.3 Target User Profile and Tentative Elicitation Process

1.3.1 Target User

1. **Donors:** Individuals, restaurant managers and event organizers.
2. **NGOs:** Staffs those are registered, verified and vetted at non-profit organizations.
3. **Admins:** Platform managers responsible for quality control.

1.3.2 User profile

Table 1.1: User Profile for Donor

User Class	Note on Characteristics
Type of user	General User (e.g., Restaurant Manager, Individual)
Age range	20-50
Frequency of use	Occasionally (when surplus food is available)
Computer experience	Basic (can use web forms, Google Maps)
Education	Varies
Language skills	Bangla and English
Number of users	Many
Training	Minimal; must be intuitive
Way of working	Submits donation form, checks profile for status

Table 1.2: User Profile for NGO

User Class	User Profile for NGO
Type of user	Registered NGO Staff (Admin-vetted)
Age range	25-50
Frequency of use	Daily (monitoring for new donations)
Computer experience	Intermediate
Education	Varies
Language skills	Bangla and English
Number of users	Few to Many (depending on registered organizations)
Training	Minimal; requires understanding of the claiming process
Way of working	Browses, filters by location, and claims donations

Table 1.3: User Profile for Admin

User Class	User Profile for Admin
Type of user	Admin (Platform Manager)
Age range	30-50
Frequency of use	Frequently (daily checks)
Computer experience	Intermediate to Advanced
Education	Varies (likely technical or managerial)
Language skills	Bangla and English
Number of users	Very Few (1-3)
Training	Requires full system knowledge
Way of working	Reviews pending queue, manages user roles

1.3.3 Elicitation Process

- **Stakeholder Interviews:** To determine issues with the current (informal) donation process, informal interviews were conducted with volunteers at a nearby charity and managers of local restaurants.
- **Feedback and Prototyping:** React components were iteratively developed. In response to feedback regarding the ease and speed of use also the enhancements were made to the user interface.
- **Competitive Analysis:** Examined current (global) food waste applications to find best practices and common features.

1.4 Project Block Diagram

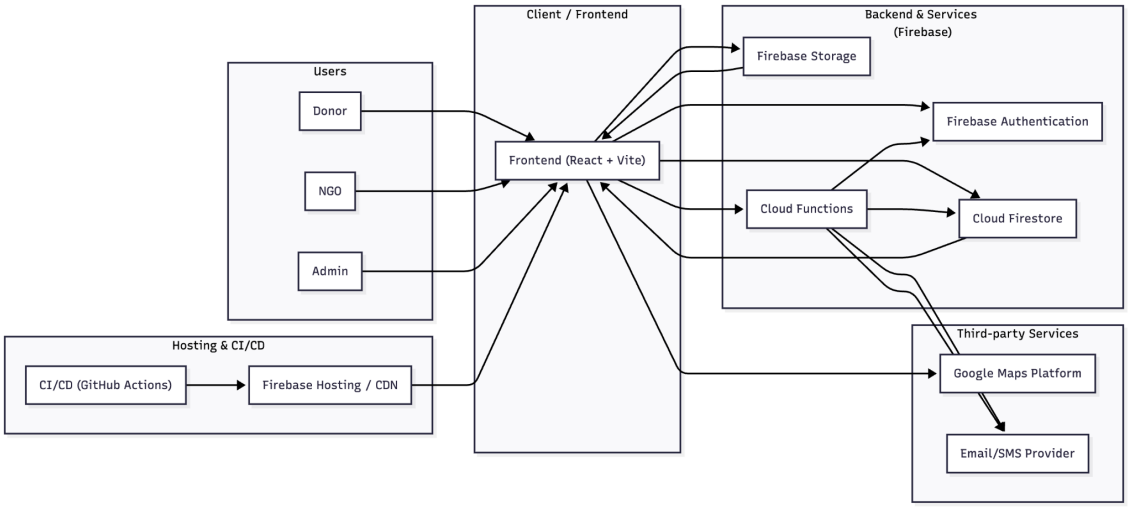


Figure 1.4.1: System Block Diagram

1.5 System Requirements

1.5.1 Hardware Requirements

- A computer (desktop, laptop) or mobile device (smartphone, tablet) with a modern web browser.
- An active and functioning internet connection to the device.

1.5.2 Software Requirements

- **Client-Side:** A contemporary web browser (e.g., Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge).
- **Development-Side:** Node.js, VS Code, Git, Firebase CLI.

1.5.3 Constraints and Dependencies

- The system is entirely dependent on an active internet connection for real-time database synchronization.
- Functionality is dependent on third-party services: Firebase (for backend) and Google Maps Platform (for location services).
- The system's integrity relies on the admin to properly vet and approve donations.

1.6 Project Scheduling

- **Time Frame/Gantt Chart:** The project was developed over an iterative 14-week timeline or in 3 to 4 month timeline (7 sprints).

Sprint 1: Foundation & Discovery

- **Goal:** Establish the project scope and create the technical foundation.
- **Tasks:**
    - A detailed project planning and feasibility studies.
    - Completely asses the technology required (React, Firebase, etc.).
    - Create first needed UI/UX for the entire application flow.
    - Install Firebase to the Project to set up Firestore and Firebase Authentication.
    - Create and install project with React Vite.
    - Complete code of header and navbar components that will have Login, Signup and Navbar.

Sprint 2: Core Authentication

- **Goal:** Develop a fully functional and secure authentication system.
- **Tasks:**
    - Use email and password for sign-up and login logic that connects the UI to Firebase Authentication.
    - Implement Google Sign-In (OAuth) functionality.
    - Creating the global AuthContext to bind user and manage user state across the app and to maintain unwanted access.
    - Structured a ProtectedRoute component file to access to a routes based on authentication status.
    - Code and install a logout function for logging out.

Sprint 3: Donor submits donations

- **Goal:** A donor inputs required fields and submits a donation that is sent for the review to admin.
- **Tasks:**
    - Code the donation submission form with all required fields.
    - Fullfill the form validation setup.
    - Install and create the Firestore database logic to write new submitted donation data to a donations collection that needs to be reviewed.
    - Dedicate a "pending" status to all new submissions.
    - The donor's profile page where they see a list of their own submissions and their current status (pending, approved, claimed).

Sprint 4: Admin Review

- **Goal:** Admin role is required to review and manage submitted donations.
- **Tasks:**
    - Admin page is protected by admin role protected route.
    - Fetch the submissions and display them from Firestore in the admin page.
    - Set up the approval and rejection logic.
    - Fix Firestore security rules to match to allow admins to perform these actions.

Sprint 5: NGO Claiming Donations

- **Goal:** NGO goes to available donations page. There they view and claim available listed donations.
- **Tasks:**
    - Create an "NGO" user role.
    - Fetch submitted donations to display them.
    - Code the claim button to claim the listed donations.
    - Update the status to claimed and removes from the list.

- NGO's profile page would show history and total counts of donations they claimed so far.

Sprint 6: Features and Iterations

- **Goal:** Better the core application with advanced admin control.
- **Tasks:**
  - Implemented the Google Maps API for avoiding confusion.
  - Executed geocoding to addresses into precise location or address while submitting donation.
  - The admin is for role manage.

Sprint 7: Final Test, Deploy and Documentation

- **Goal:** Ready the platform to be usable for public.
- **Tasks:**
  - Perform full test. Change roles and check their accessible options.
  - Fix any security rule causing error and do another test.
  - Cure any bugs in the code and fix any issues comes front during the test.
  - Write the project documentation that is called README file. Also write project setup guides or manual.
  - Split codes and optimize the lazy loadings to minimize time consumptions.
  - Finally final version of the platform is ready to deploy to Firebase Hosting.

- **Risk Management:**
    - **Risk:** Usage and invalid food giveaways or donations.
        - **Mitigation:** A compulsory approval required from admin. All the donations are registered as pending till they are verified by an admin.
    - **Risk:** Unauthorized access to data.
        - **Mitigation:** To avoid misuse, the Firestore Security Rules and client-side Protected Routes are made based on the user roles.
    - **Risk:** Inaccurate address or location data.
        - **Mitigation:** Added the Google Places Autocomplete API into the donation form to ease the support and false information. This ensures and asks the donors to use a valid geocoded address. Which assures that all the donations are correctly geocoded to be filtered and allow verification.

1.7 Summary

The project's outcome, scope and features were described in this chapter. It demonstrated the project's success from technical, financial, and commercial perspectives. Furthermore, it detailed the system requirements, intended users and the agile project timeline.

CHAPTER 2

DESIGN AND IMPLEMENTATION

2.1 Introduction

This chapter details the specific functional and non-functional requirements of the FoodShare platform. The text also illustrates the object-oriented system design via a collection of UML diagrams, which encompass use cases, activity diagrams, sequence diagrams and data models (including class and ER diagrams) that facilitated the implementation.

2.2 Functional Requirements

FR ID-01	**Registration**
Description	Allows new donors and NGOs to establish an account by supplying their details, creating a password and confirming their email address.
Stakeholder	Donor, NGO, Admin

FR ID-02	**Login**
Description	For users to log in securely with email and strong password or via Google Sign-In for easy and no password sign in.
Stakeholder	Donor, NGO, Admin

FR ID-03	Email Verification
Description	Sends a verification link to email at the time of registration. The system checks verification status when they logs in.
Stakeholder	Donor, NGO

FR ID-04	Forgot Password
Description	Allows a user to request a password reset link via their registered email.
Stakeholder	All Users

FR ID-05	Donation Submission
Description	Enables a logged-in donor to submit a donation form. The user must select a valid address from the Google Places Autocomplete suggestions. The system saves the address string and its GeoPoint coordinates.
Stakeholder	Donor

FR ID-06	Donation Approval
Description	Enables an admin to view a queue of pending donations and update their status to approved or rejected.
Stakeholder	Admin

FR ID-07	View Available Donations
Description	Enables a logged-in NGO to see a real-time list of all approved donations.
Stakeholder	NGO

FR ID-08	Claim Donation
Description	Enables an NGO to claim an approved donation, changing its status to claimed and recording their user ID and name.
Stakeholder	NGO

FR ID-09	**Location-Based Filtering**
Description	Enables an NGO to request their browser location and filter the available donations within a specified kilometer radius.
Stakeholder	NGO

FR ID-10	**Profile Page**
Description	Allows a user to view their donation history. Donors see their submitted donations; NGOs see their claimed donations.
Stakeholder	Donor, NGO

FR ID-11	**Manage Profile**
Description	Allows a logged-in user to update their own non-critical information (e.g., name, contact no.).
Stakeholder	Donor, NGO, Admin

FR ID-12	**Admin Role Management**
Description	Enables an admin to view a list of all users and change their role (e.g., promote a 'donor' to an 'NGO').
Stakeholder	Admin

2.3 Non-Functional Requirements

2.3.1 Performance

- Because application makes use of real-time onSnapshot listeners, clients don't need to refresh their pages in order to receive data updates (such as a donation being claimed).
- Even with a large number of donations, database queries are indexed (for example: to filter by status and sort by date) to guarantee quick load times.

2.3.2 Reliability

- The Firebase database on which the application is based provides automatic scaling and high performance.
- All database operations use client-side error handling which makes use of toast notifications and try...catch.

2.3.3 Portability

- It is completely portable and available on any device with a contemporary web browser because it is a web application (desktop, tablet, smartphone).

2.3.4 Security

- All data access is managed by strict Firestore Security Rules which guarantee that users can only read and write data that is allowed by their role (for example, only administrators can approve).

- Clien side ProtectedRoute components or code file is used to verify role and to safely authenticate the admin and profile pages.

2.3.5 Usability

- To create responsive and mobile friendly design both Tailwind CSS and DaisyUI are used for the users.

- Workflows are easy to understand; for example, Google Address Autocomplete makes entering addresses easier.

2.4 Object-Oriented System Design using UML

2.4.1 Use Case Diagram

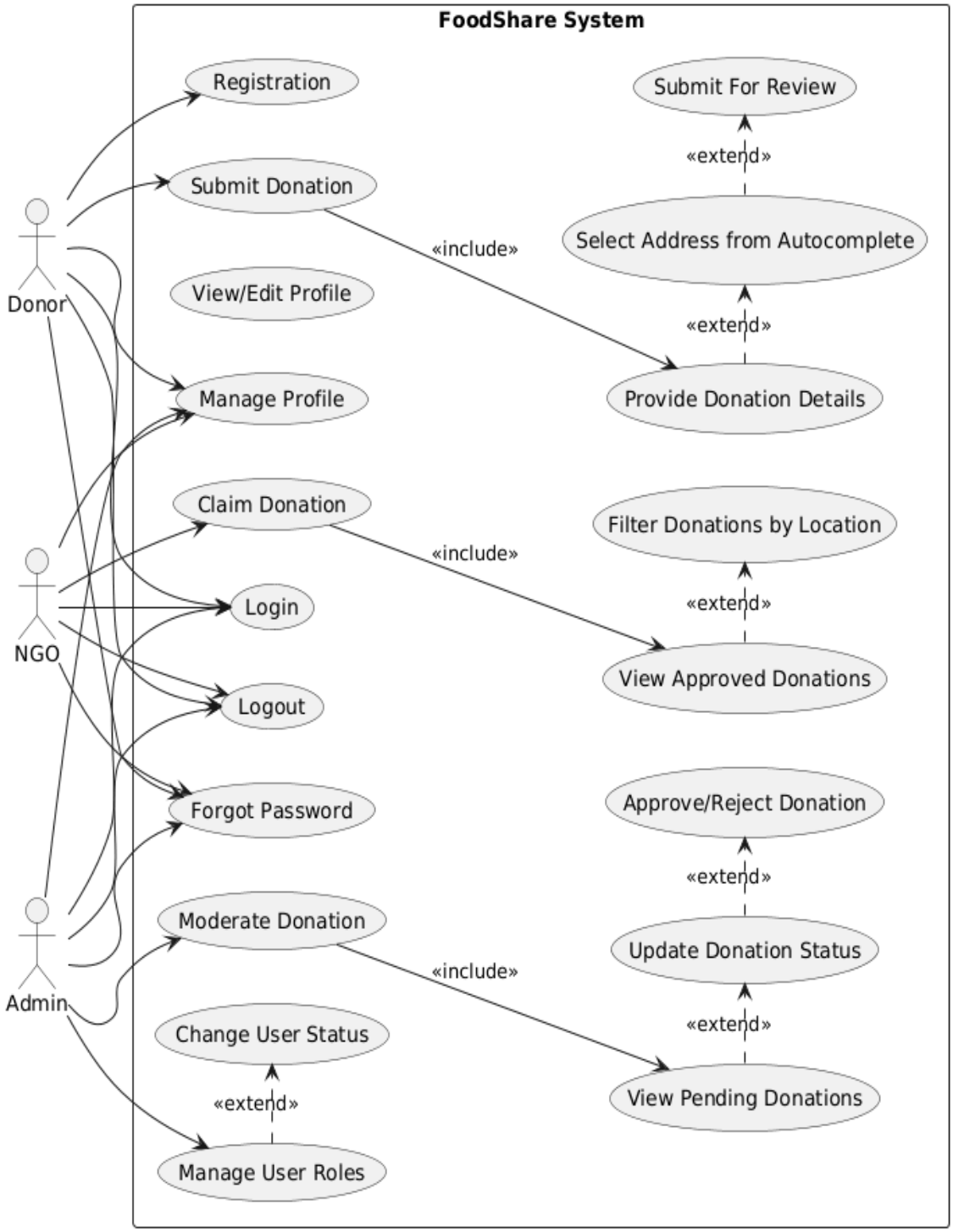


Figure 2.4.1: Use Case Diagram

2.4.2 Case Description

Case Description-01: Registration

<table>
<tr><td>Use Case</td><td>Registration</td></tr>
<tr><td>Goal</td><td>New user or viewers can create account to become a registered user.</td></tr>
<tr><td>Precondition</td><td>A web browser and internet is required for accessing the website.</td></tr>
<tr><td>Success End Condition</td><td>Notification: "Account has been created successfully!" (The user is now logged in and has been redirected).</td></tr>
<tr><td>Failed End Condition</td><td>"Email already exists" or "Submission Failed".</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Viewer
System</td></tr>
<tr><td>Trigger</td><td>The Viewer selects the "Sign Up" button from the Navbar.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>Viewer selects the "Sign Up" link.</td></tr>
<tr><td>2.</td><td>Present registration form</td></tr>
<tr><td>3.</td><td>Enter Information</td></tr>
<tr><td>4.</td><td>Selects “Sign Up” Button.</td></tr>
<tr><td>5.</td><td>Information saved</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>System Error</td></tr>
<tr><td></td><td>1.1.a. Please Try Again with Different Credentials!!</td></tr>
<tr><td>4.1</td><td>The user Did not fill up the details!</td></tr>
<tr><td></td><td>4.1.a. Checked By the system & Notify by “Please! Fill out the fields”.</td></tr>
<tr><td>5.1</td><td>The system did not respond</td></tr>
<tr><td></td><td>5.1.a. Show Error Message.</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>The user needs to fill the details in 5 minutes.</td></tr>
</table>

Case Description-02: Login

<table>
<tr><td>Use Case</td><td>Login</td></tr>
<tr><td>Goal</td><td>A new user logs in to the website and can view pages.</td></tr>
<tr><td>Precondition</td><td>The user must be registered.</td></tr>
<tr><td>Success End Condition</td><td>After the user has been verified, the homepage is displayed and the authorization context has been filled in. Alert Shown: "Logged in successfully!</td></tr>
<tr><td>Failed End Condition</td><td>Notification: "Failed to log in."</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Donor, NGO
Admin</td></tr>
<tr><td>Trigger</td><td>The user clicks the "Log In" button.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>Viewer clicks the "Log In" button.</td></tr>
<tr><td>2.</td><td>Provide credentials (or click Sign in with Google)</td></tr>
<tr><td>3.</td><td>System verifies credentials.</td></tr>
<tr><td>4.</td><td>System checks emailVerified == true</td></tr>
<tr><td>5.</td><td>System fetches user profile from Firestore and populates AuthContext</td></tr>
<tr><td>6.</td><td>System redirects to homepage.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>System Error</td></tr>
<tr><td></td><td>1.1.a. Please Try Again with Different Credentials!!</td></tr>
<tr><td>3.1</td><td>Email not registered.</td></tr>
<tr><td></td><td>3.1.a. Notification: "Account not found"</td></tr>
<tr><td>4.1</td><td>Email not verified.</td></tr>
<tr><td></td><td>4.1.a. Notification: "Please verify your email before logging in."</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>The user will fill up all the details in 1 minute.</td></tr>
</table>

Case Description-03: Manage Profile

<table>
<tr><td>Use Case</td><td>Manage Profile</td></tr>
<tr><td>Goal</td><td>Users update their information.</td></tr>
<tr><td>Precondition</td><td>The user must be logged in (authenticated).</td></tr>
<tr><td>Success End Condition</td><td>The user's profile information is validated and saved to the database (e.g., Firestore). The user sees their updated profile. Notification: "Profile updated successfully!"</td></tr>
<tr><td>Failed End Condition</td><td>The user's profile information is not saved.
Notification: "Failed to update profile."</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Donor, NGO

Admin</td></tr>
<tr><td>Trigger</td><td>The user navigates to the "Profile" page and edits name and contact information then clicks the "Save Changes" button.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>The user navigates to the profile page.</td></tr>
<tr><td>2.</td><td>The user's current information is pre-filled into an editable form that the system displays.</td></tr>
<tr><td>3.</td><td>The user makes changes as they want to keep their information valid.</td></tr>
<tr><td>4.</td><td>The user clicks save changes.</td></tr>
<tr><td>5.</td><td>The input data is verified by the system.</td></tr>
<tr><td>6.</td><td>The database for the user profile document is updated by the system.</td></tr>
<tr><td>7.</td><td>The new information is displayed when the system reroutes to the read-only profile page.</td></tr>
</table>
</td></tr>
</table>

Alternative Flows	<table><tr><td>4.1</td><td>The user reloads the page without clicking “Save Changes”.</td></tr><tr><td>5.1</td><td>Failures in Validation</td></tr><tr><td></td><td>5.1.a. The system indicates which form fields contain incorrect information.</td></tr></table>
Quality Requirements	The profile page must load existing user data and save changes within 3 seconds.

Case Description-04: Donation Submission

Use Case	Donation Submission
Goal	A donor posts a new surplus food donation for admin review.
Precondition	The user must be logged in with the "Donor" role.
Success End Condition	A new donation document is created in Firestore with status 'pending' and a valid location GeoPoint.
Failed End Condition	The donation is not created; an error notification is shown.
Primary Actors	Donor
Trigger	The user navigates to the "Donate" page.

<table>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>The donor selects "Donate."</td></tr>
<tr><td>2.</td><td>The donor completes the food type, quantity and contact form.</td></tr>
<tr><td>3.</td><td>The user makes changes to the desired fields (e.g., Name, Contact Number).</td></tr>
<tr><td>4.</td><td>After that the donor enters the address and choose a valid location from the Google Places Autocomplete option.</td></tr>
<tr><td>5.</td><td>The address string and its geographic coordinates are confirmed by the system.</td></tr>
<tr><td>6.</td><td>Submit for Review is clicked by the donor.</td></tr>
<tr><td>7.</td><td>The new donation document is saved to Firestore by the system.</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>4.1</td><td>Address not selected from suggestions: The system shows an error. "Please select a valid address from the suggestions."</td></tr>
<tr><td>7.1</td><td>Database Error: The system shows the error "Failed to submit donation."</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>Status updates must be reflected in real time for all users (< 3 seconds).</td></tr>
</table>

Case Description-05: Moderate Donation (Admin)

<table>
<tr><td>Use Case</td><td>Moderate Donation</td></tr>
<tr><td>Goal</td><td>Admin reviews pending donations and either approves them (making them available to NGOs) or rejects them.</td></tr>
<tr><td>Precondition</td><td>Admin must be logged in. There must be at least one donation with status 'pending.'</td></tr>
<tr><td>Success End Condition</td><td>The status of the donation is modified to either approved or rejected.</td></tr>
<tr><td>Failed End Condition</td><td>Donation status continues to be pending. Notification: "Failed to update donation."</td></tr>
<tr><td>Primary Actors</td><td>Admin</td></tr>
<tr><td>Trigger</td><td>The admin accesses the moderation dashboard to assess new submissions.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>The admin enters the moderation dashboard.</td></tr>
<tr><td>2.</td><td>The system <<includes>> View Pending Donations, which presents a compilation of all donations marked as "pending."</td></tr>
<tr><td>3.</td><td>The admin chooses a donation for reviewing.</td></tr>
<tr><td>4.</td><td>Selects the "Approve" button.</td></tr>
<tr><td>5.</td><td>System changes the status to approved.</td></tr>
<tr><td>6.</td><td>Firestore Rules confirms the status change.</td></tr>
<tr><td>7.</td><td>System saves new donation documentation to Firestore.</td></tr>
<tr><td>8.</td><td>The donation is removed instantly from the pending list.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Alternative Flows</td><td><table>
<tr><td>4.1</td><td>The admin presses the "Reject" button.</td></tr>
<tr><td>5.1</td><td>The system modifies the donation status to rejected.</td></tr>
<tr><td>8.1</td><td>The donation is eliminated from the pending list.</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Status updates must be reflected in real time for all users (< 3 seconds).</td></tr>
</table>

Case Description-06: Forgot Password

Use Case	Forgot Password
Goal	Allows a user to reset their password via email if they have forgotten it.
Precondition	User is logged out and has a registered account with a valid email.
Success End Condition	A password reset email is successfully sent to the user's email address.
Failed End Condition	No email is sent. Notification: "Failed to send reset email."
Primary Actors	Donor, NGO, Admin
Trigger	User clicks the Forgot Password from the Login page of the system.

<table>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>User or Viewer visits the Login page.</td></tr>
<tr><td>2.</td><td>Click on to "Forgot Password?" link.</td></tr>
<tr><td>3.</td><td>The Platform redirects user to the password reset form which requests an email address to enter.</td></tr>
<tr><td>4.</td><td>User enters their registered or verified email address.</td></tr>
<tr><td>5.</td><td>User clicks "Send Reset Link" button.</td></tr>
<tr><td>6.</td><td>System checks email verified or not.</td></tr>
<tr><td>7.</td><td>System sends a password reset link to designated email.</td></tr>
<tr><td>8.</td><td>The system shows notification saying a success message: "Check your email to reset password."</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>6.1</td><td>Invalid Email Format: The user inputs text that lacks "@" or a domain.</td></tr>
<tr><td></td><td>The system displays an error and instructs user to enter a valid email address.</td></tr>
<tr><td>7.1</td><td>Email Not Found: The entered email is not associated with any account.</td></tr>
<tr><td></td><td>7.1.a System displays error: "Account not found" (or generic success message for security).</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>The reset email should be triggered within 30 sec.</td></tr>
</table>

Case Description-07: Location-Based Filtering (NGO)

<table>
<tr><td>Use Case</td><td>Location-Based Filtering</td></tr>
<tr><td>Goal</td><td>NGO filters available donations to find those nearby</td></tr>
<tr><td>Precondition</td><td>Users must be logged in with an NGO role.</td></tr>
<tr><td>Success End Condition</td><td>The list of donations is filtered to show only those within the specified distance.</td></tr>
<tr><td>Failed End Condition</td><td>The filter is not applied; an error notification is shown.</td></tr>
<tr><td>Primary Actors</td><td>NGO</td></tr>
<tr><td>Trigger</td><td>The NGO proceeds to the available donations.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>The NGO puts a distance (for example, "4") into the radius field.</td></tr>
<tr><td>2.</td><td>NGO pressed the "Filter" button for filtering.</td></tr>
<tr><td>3.</td><td>Browser wants the user location permission.</td></tr>
<tr><td>4.</td><td>System gets the NGO's current location.</td></tr>
<tr><td>5.</td><td>The system filters and calculates distance for every approved donation.</td></tr>
<tr><td>6.</td><td>Interface displays donations within a 4 km radius.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Alternative Flows</td><td><table>
<tr><td>3.1</td><td>The user Declines location permission.</td></tr>
<tr><td></td><td>The system presents the error message "Please enable location services."</td></tr>
<tr><td>6.1</td><td>There are no donations that meet the criteria.</td></tr>
<tr><td></td><td>6.1.a The system displays "No available donations match your criteria."</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Status updates must be reflected in real time for users (< 3 seconds).</td></tr>
</table>

Case Description-08: Donation Claiming (NGO)

Use Case	Donation Claiming
Goal	NGO filters available donations to find those nearby
Precondition	Users must be logged in with an NGO role.
Success End Condition	The list of donations is filtered to show only those within the specified distance.
Failed End Condition	The filter is not applied; an error notification is shown.
Primary Actors	NGO
Trigger	NGO navigates to available donations

<table>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>NGO puts a distance (e.g., "10") into the radius.</td></tr>
<tr><td>2.</td><td>NGO clicks the "Filter" button.</td></tr>
<tr><td>3.</td><td>Browser asks the user for accessing location permission.</td></tr>
<tr><td>4.</td><td>The system gets the NGO's current location.</td></tr>
<tr><td>5.</td><td>The system filters the distance of approved donation internally.</td></tr>
<tr><td>6.</td><td>UI renders to show only donations within a 10 km radius.</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>3.1</td><td>The user denies location permission.</td></tr>
<tr><td></td><td>3.1.a The system shows the error "Please enable location services."</td></tr>
<tr><td>6.1</td><td>No donations match the criteria.</td></tr>
<tr><td></td><td>6.1.a The system displays "No available donations match your criteria."</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>Status updates must be reflected in real time for users (< 3 seconds).</td></tr>
</table>

Case Description-09: Manage User Roles (Admin)

<table>
<tr><td>Use Case</td><td>Manage User Roles</td></tr>
<tr><td>Goal</td><td>Allows an admin to change the permission level or role of a registered user (e.g., promote a 'Donor' to an 'NGO').</td></tr>
<tr><td>Precondition</td><td>The administrator must be logged into the FoodShare system successfully.</td></tr>
<tr><td>Success End Condition</td><td>The role of the target user is updated in the database successfully.</td></tr>
<tr><td>Failed End Condition</td><td>The role of the user remains the same. Notification: "Failed to update user role."</td></tr>
<tr><td>Primary Actors</td><td>Admin</td></tr>
<tr><td>Trigger</td><td>The admin opens the admin page and selects a user for modification.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>The administrator visits to the "admin" page.</td></tr>
<tr><td>2.</td><td>System shows list of the registered users and their current roles with role change dropbar.</td></tr>
<tr><td>3.</td><td>The administrator chooses a specific user to manage.</td></tr>
<tr><td>4.</td><td>The system provides an option to alter the selected user's role (for instance, a dropdown menu featuring 'Donor,' 'NGO,' and 'Admin').</td></tr>
<tr><td>5.</td><td>System updates the user's record in the database with the new role.</td></tr>
<tr><td>6.</td><td>System presents a success message: "User role updated successfully."</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Alternative Flows</td><td><table>
<tr><td>5.1</td><td>Database Error</td></tr>
<tr><td></td><td>5.1.a The system is unable to record the role change in the database.</td></tr>
<tr><td>6.1</td><td>No donations match the criteria.</td></tr>
<tr><td></td><td>6.1.a The system shows "No available donations match your criteria."</td></tr>
<tr><td></td><td>6.1.b Notification: "Failed to update user role."</td></tr>
<tr><td>6.2</td><td>6.2.a The administrator tries to remove their own 'admin' role, being the last admin in the system.</td></tr>
<tr><td></td><td>6.2.b The system blocks the action. Notification: "Cannot remove the last administrator."</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>The role change should take effect immediately or upon the user's next session refresh.</td></tr>
</table>

Case Description-10: Logout

<table>
<tr><td>Use Case</td><td>Logout</td></tr>
<tr><td>Goal</td><td>The user securely ends their authenticated session and is returned to a public-facing page.</td></tr>
<tr><td>Precondition</td><td>The user must be logged in as an verified user.</td></tr>
<tr><td>Success End Condition</td><td>The user's session is terminated, AuthContext is cleared, and the user is redirected to the login page. Notification: "Logged out successfully!"</td></tr>
<tr><td>Failed End Condition</td><td>The user's session is not terminated. Notification: "Logout failed. Please try again."</td></tr>
<tr><td>Primary Actors</td><td>Donor, NGO, Admin</td></tr>
<tr><td>Trigger</td><td>The user clicks the "Log Out" button.</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>The user clicks the "Logout" button.</td></tr>
<tr><td>2.</td><td>The system clears the authentication state (e.g., AuthContext, local storage/session tokens).</td></tr>
<tr><td>3.</td><td>The system cancels the session on the server.</td></tr>
<tr><td>4.</td><td>System redirects the user to the Login page.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Alternative Flows</td><td><table>
<tr><td>3.1</td><td>Network Error</td></tr>
<tr><td></td><td>3.1.a The system fails to communicate with the server to invalidate the session.</td></tr>
<tr><td></td><td>3.1.b The system proceeds to clear the local authentication state and redirects to Login to ensure the user is logged out from the client's perspective.</td></tr>
<tr><td></td><td>3.1.c Notification: "You have been logged out locally. The session on the server may still be active."</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Logout must be processed and the user redirected within 3 seconds.</td></tr>
</table>

2.4.3 Activity Diagram

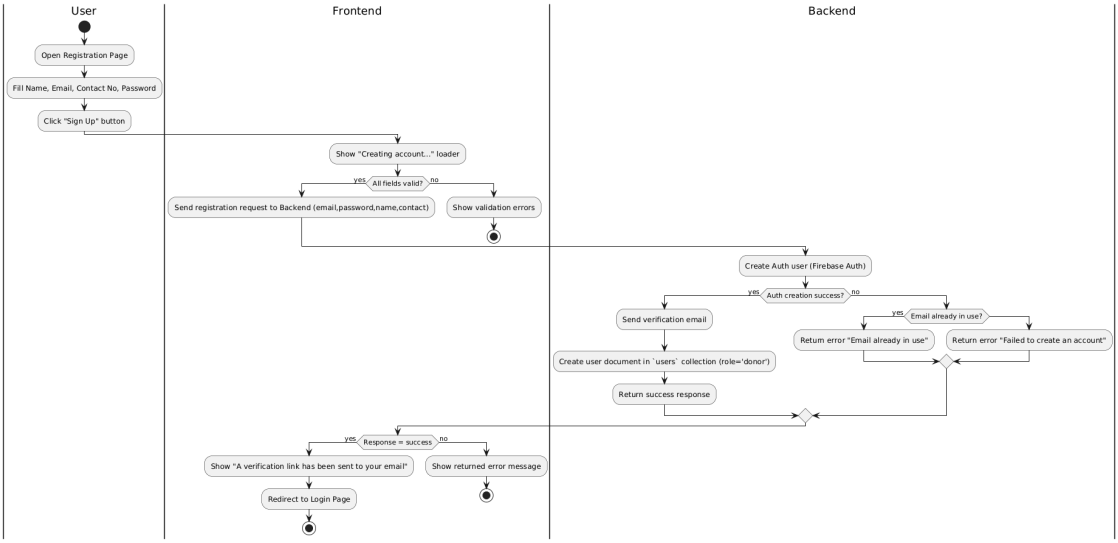


Figure 2.4.3.1: Registration

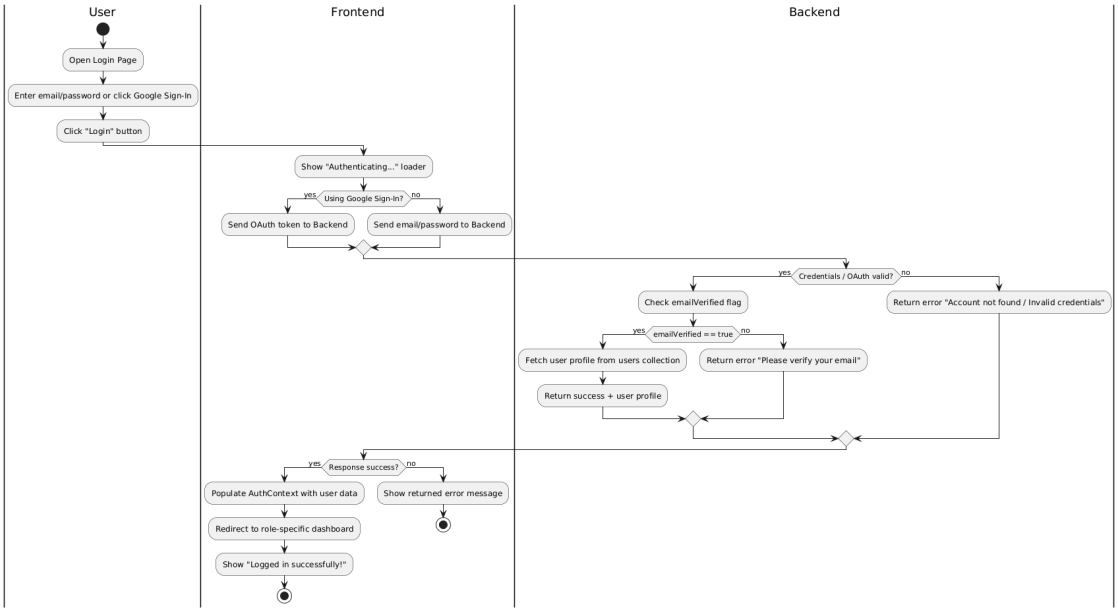


Figure 2.4.3.2: Login

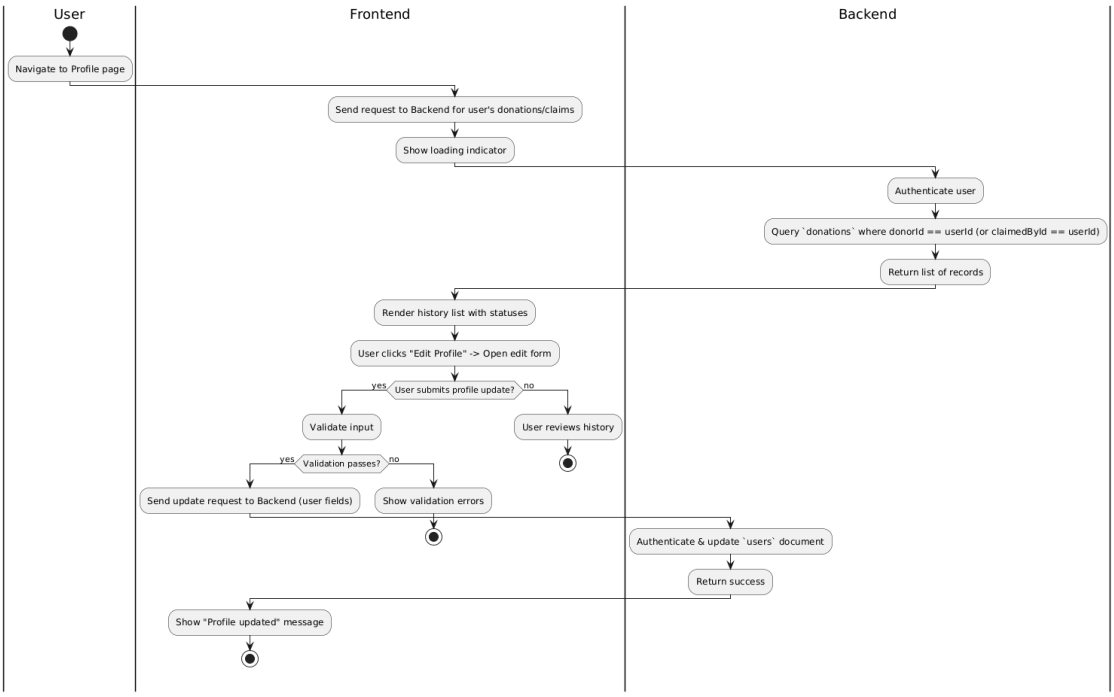


Figure 2.4.3.3: Manage Profile

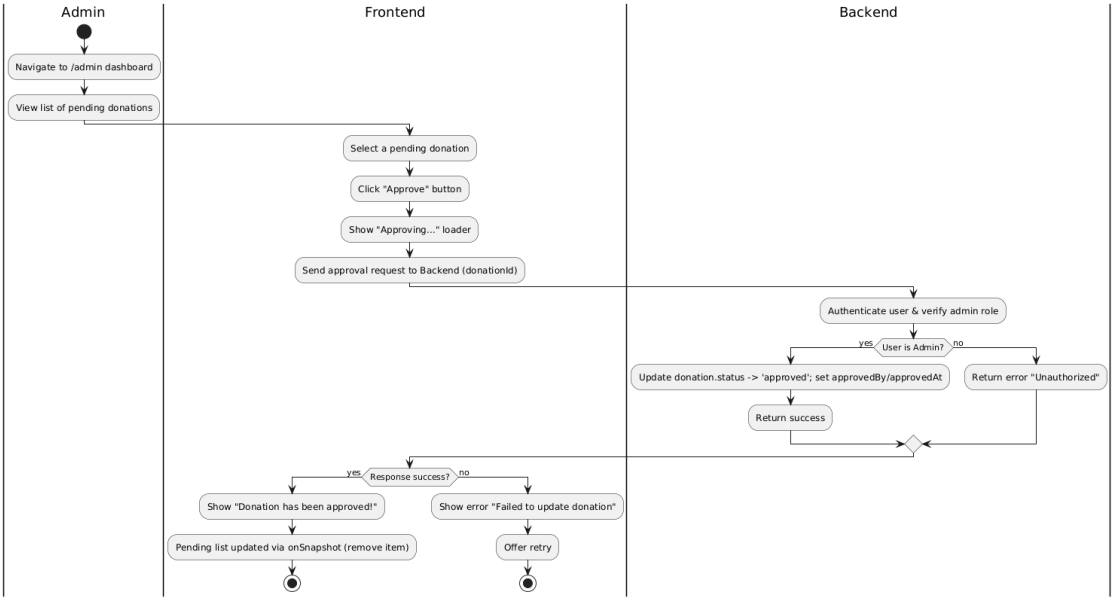


Figure 2.4.3.4: Moderate Donation

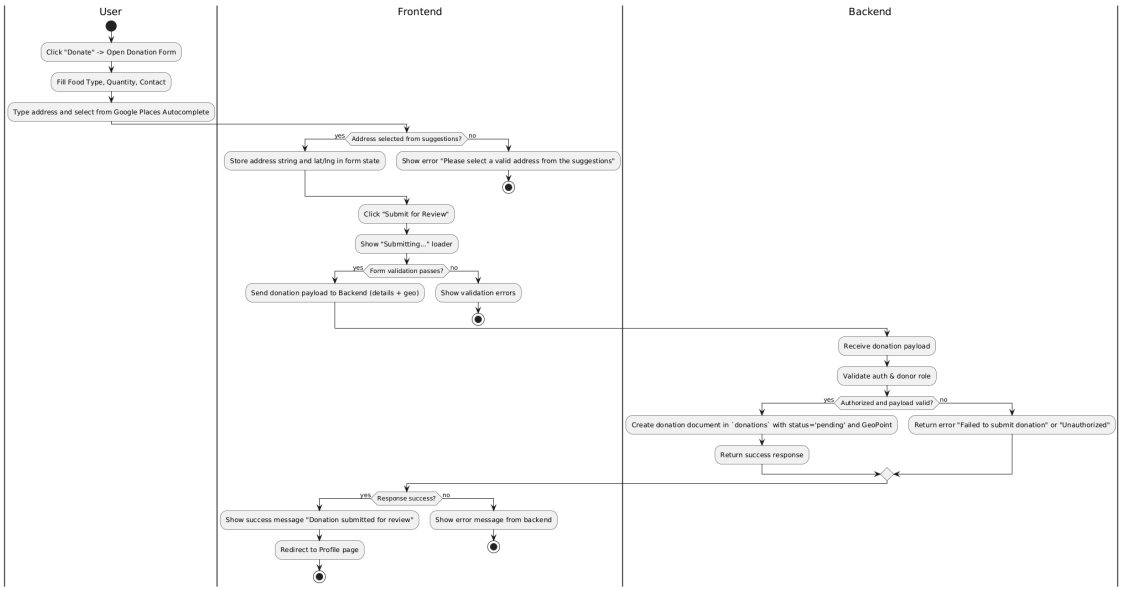


Figure 2.4.3.5: Donation Submission

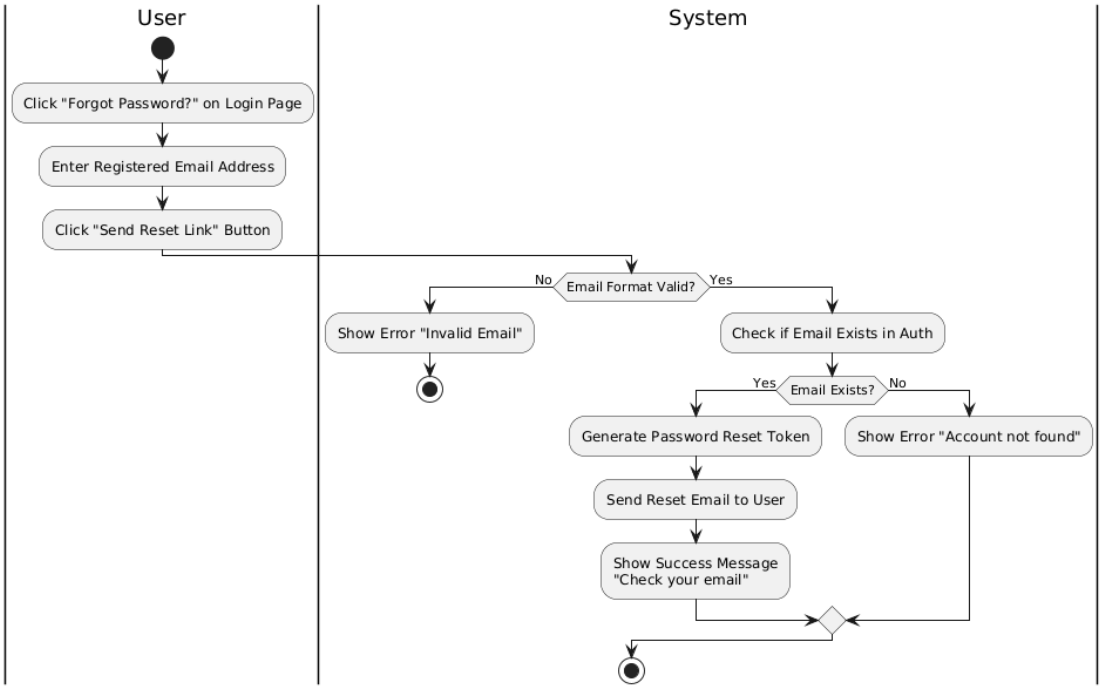


Figure 2.4.3.6: Forgot Password

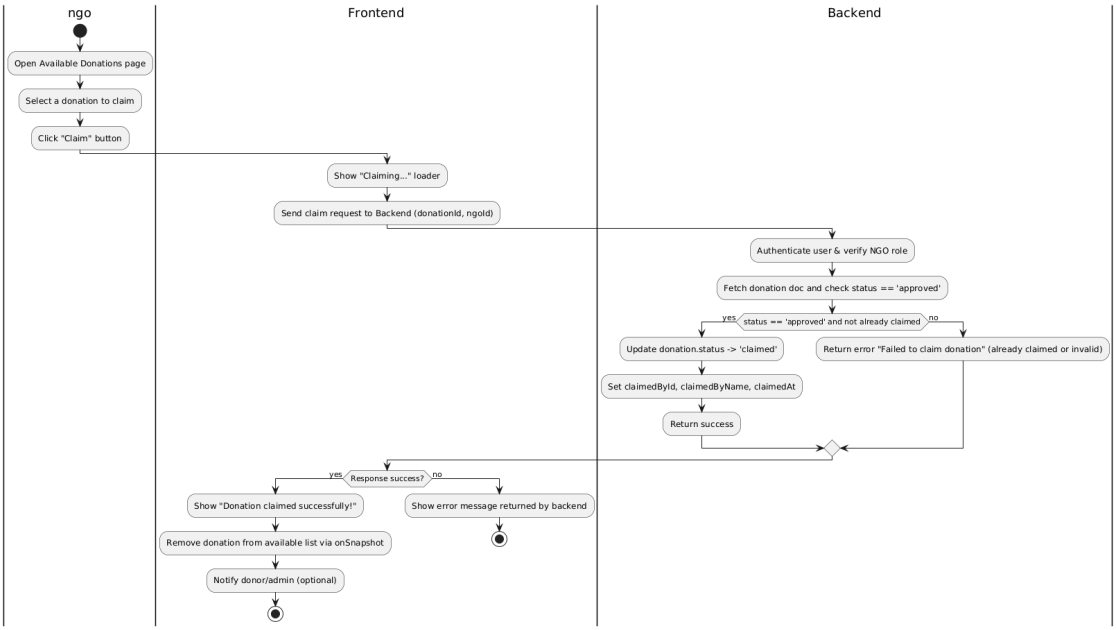


Figure2.4.3.7: Claim Donation

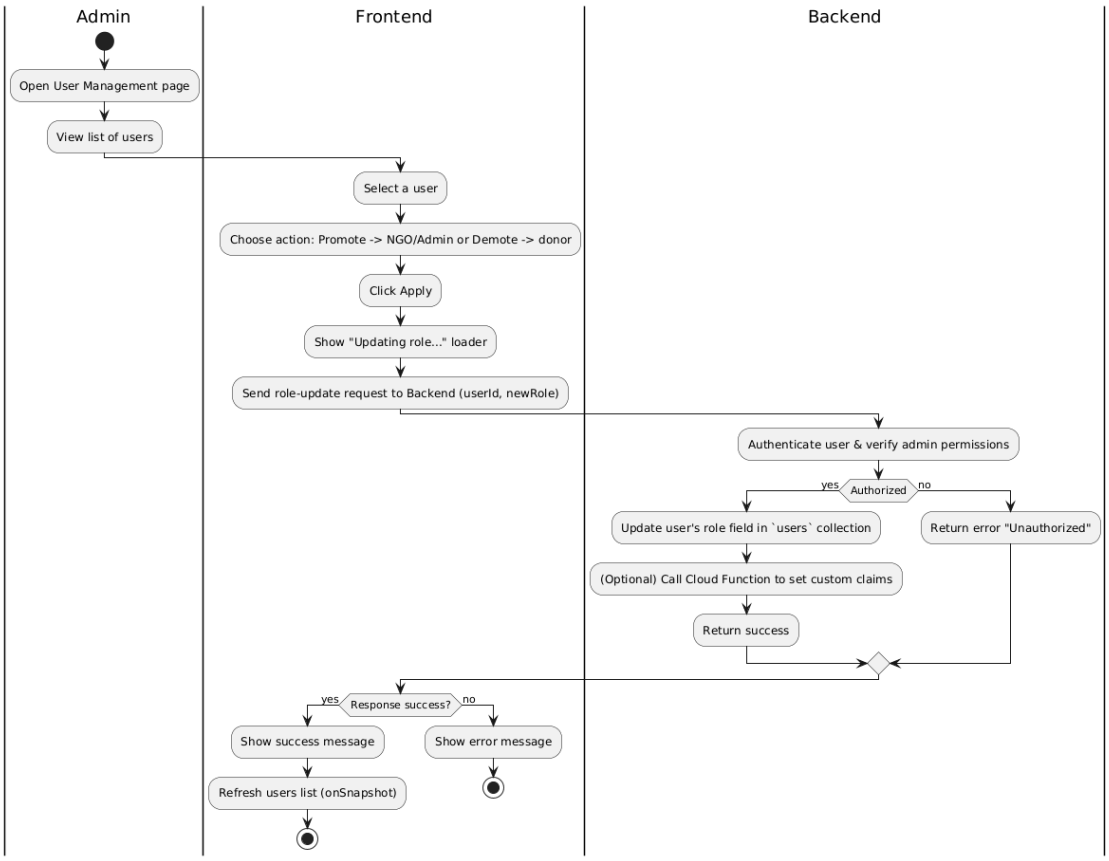


Figure 2.4.3.8: Manage User Role

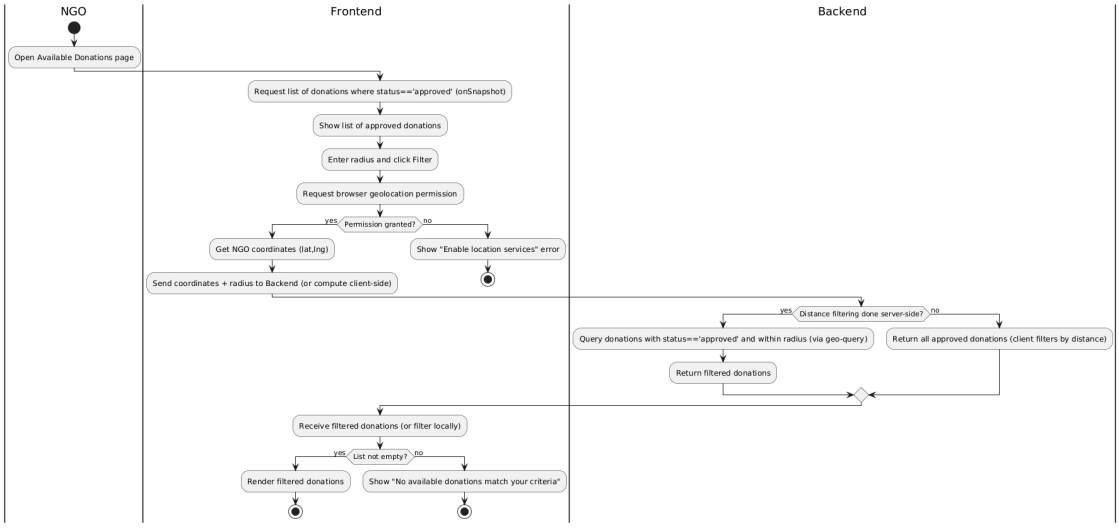


Figure 2.4.3.9: View Available Donation

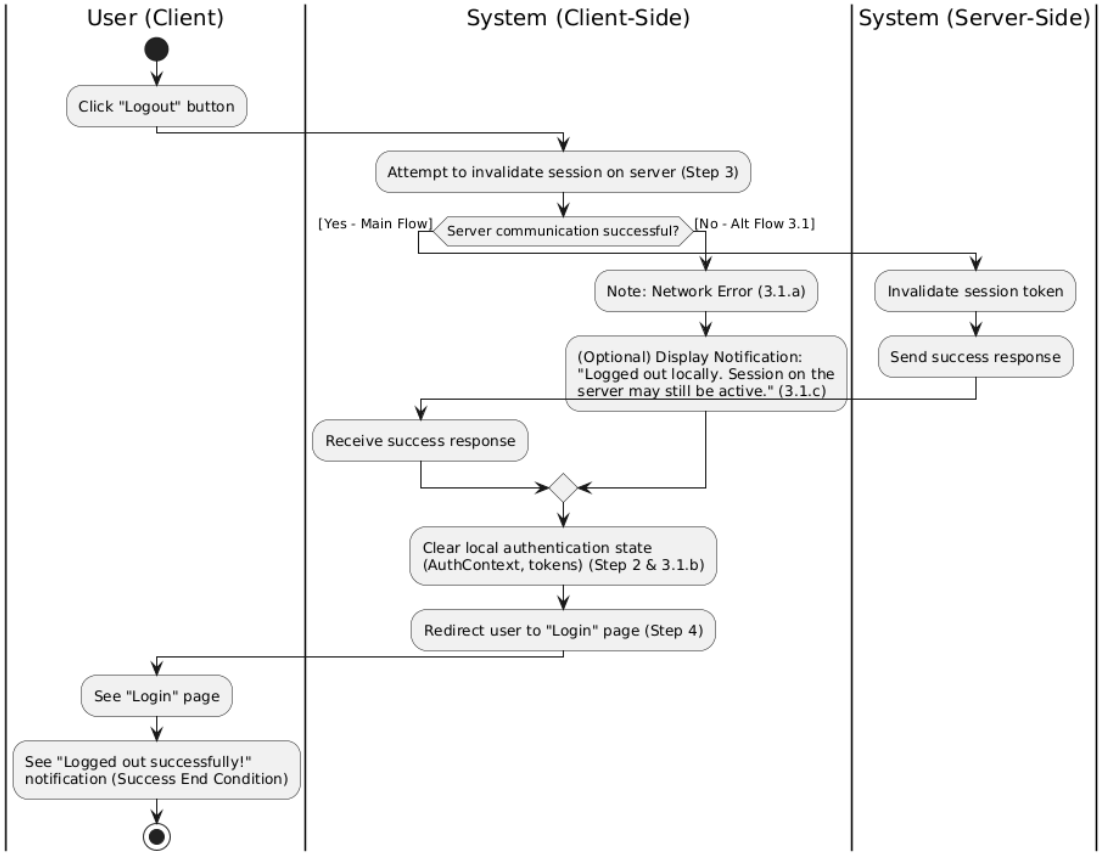


Figure 2.4.3.10: Logout

2.4.4 Sequence Diagram

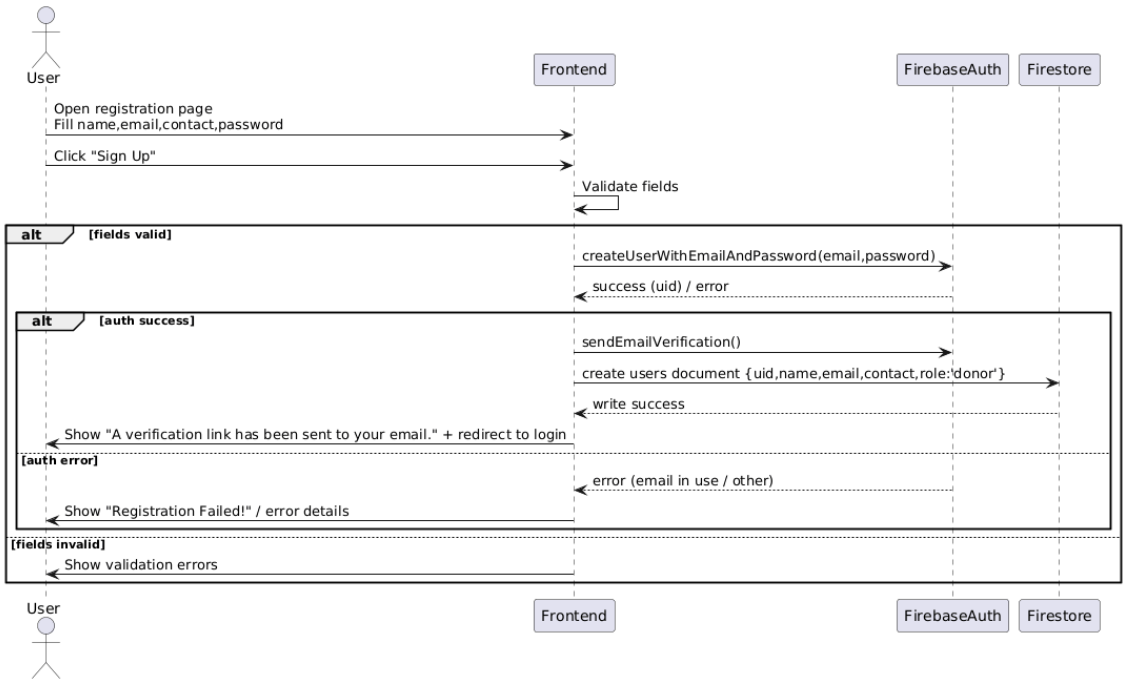


Figure 2.4.4.1: Registration

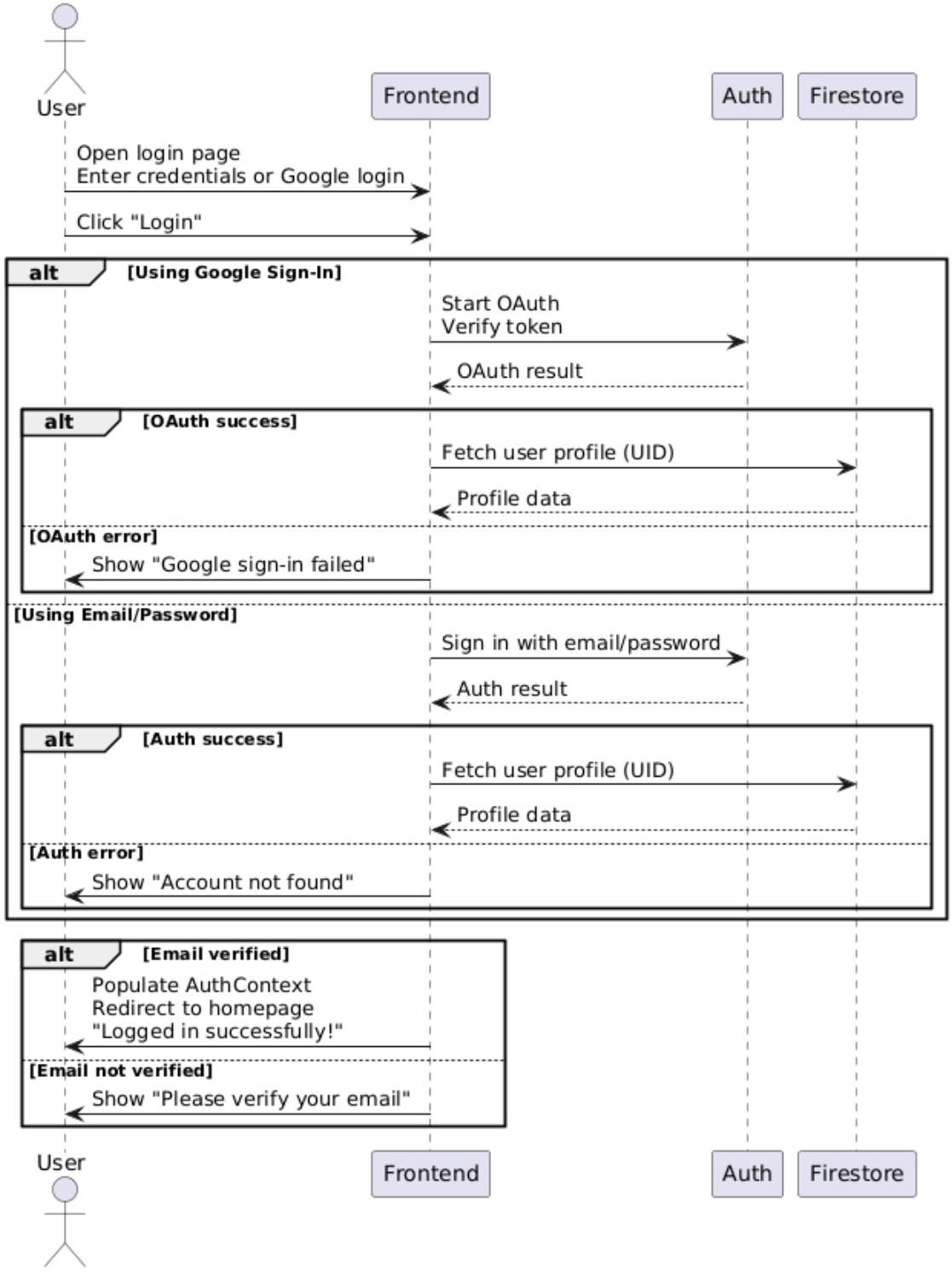


Figure 2.4.4.2: Login

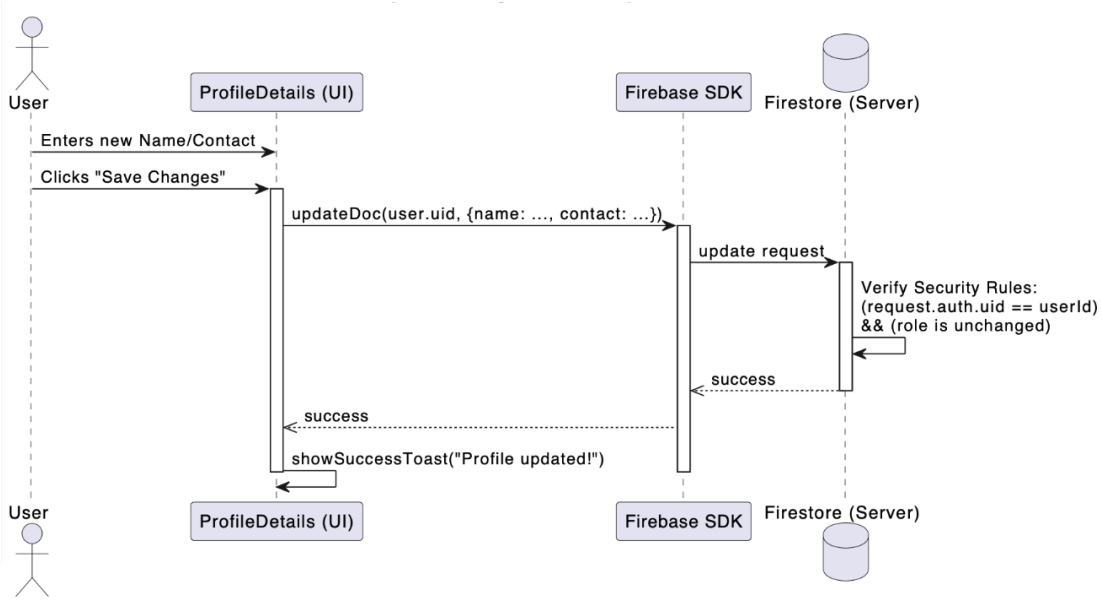


Figure 2.4.4.3: Manage Profile

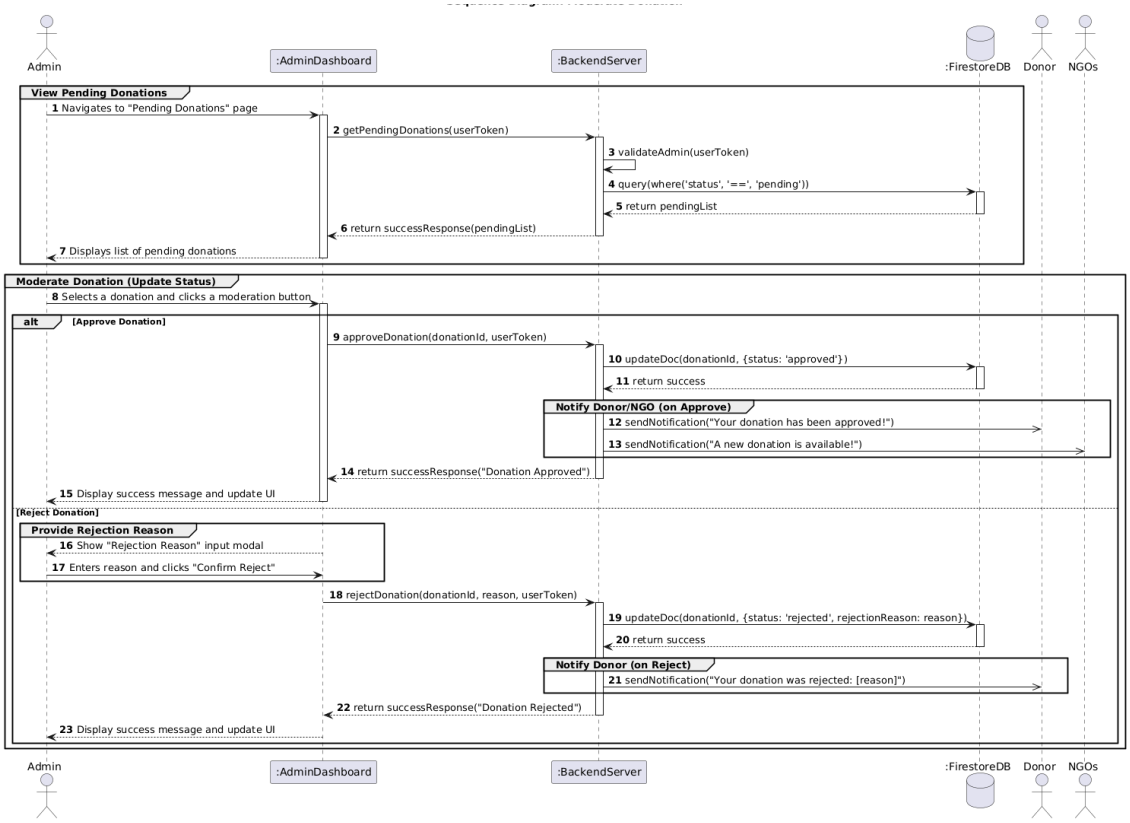


Figure 2.4.4.4: Moderate Donation

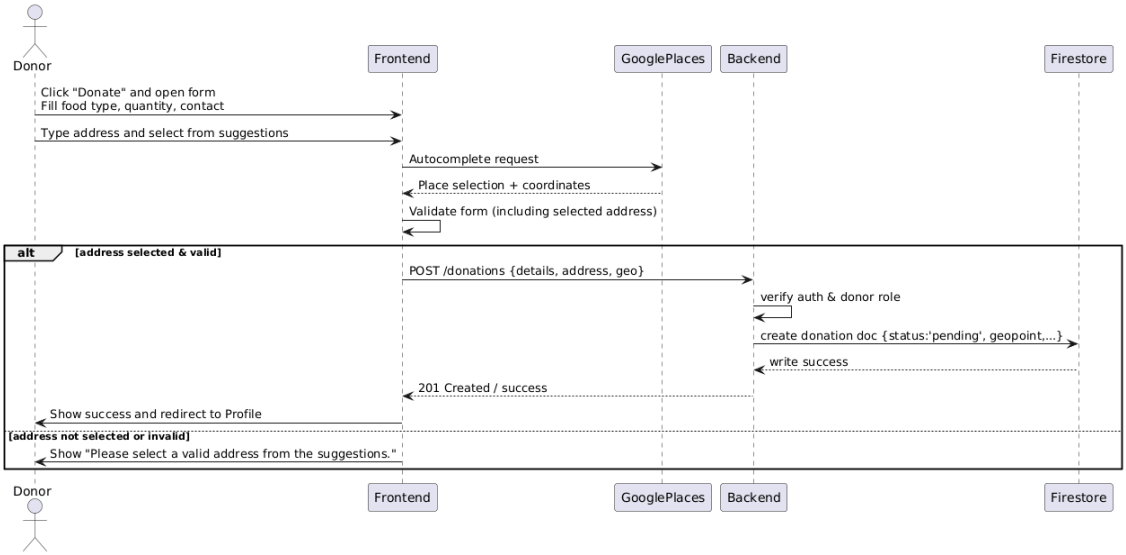


Figure 2.4.4.5: Donation Submission

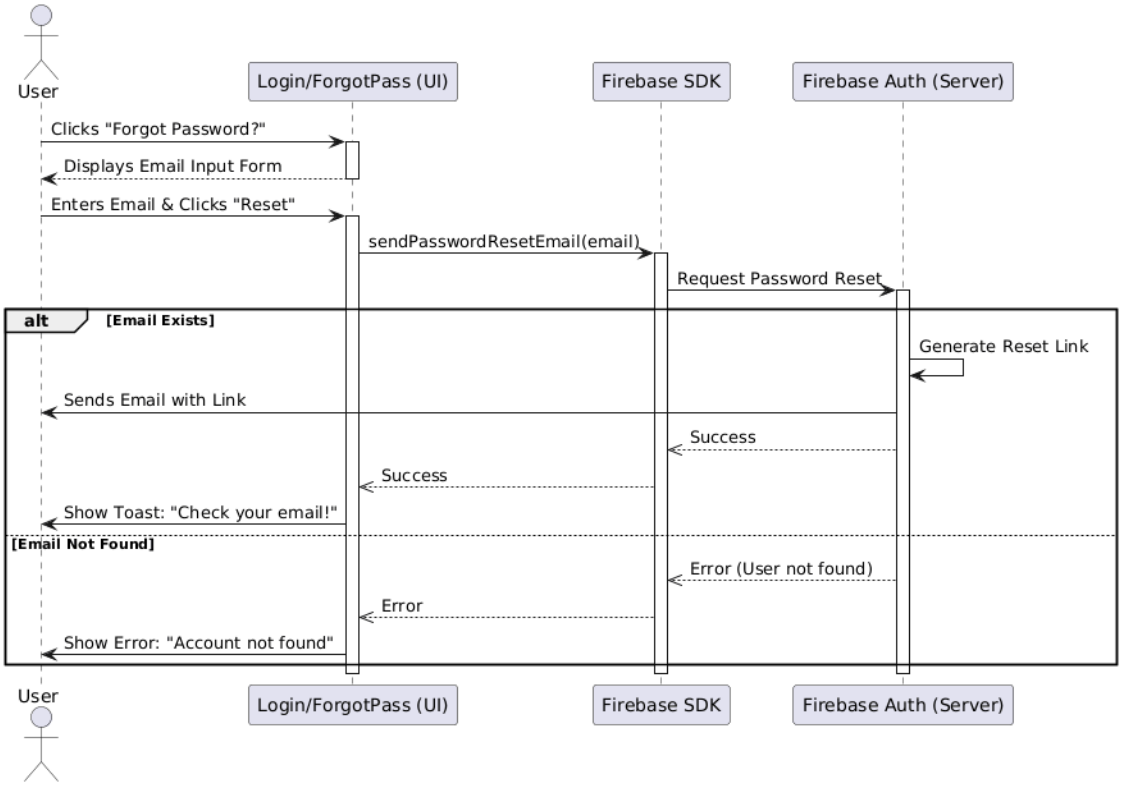


Figure 2.4.4.6: Forgot Password

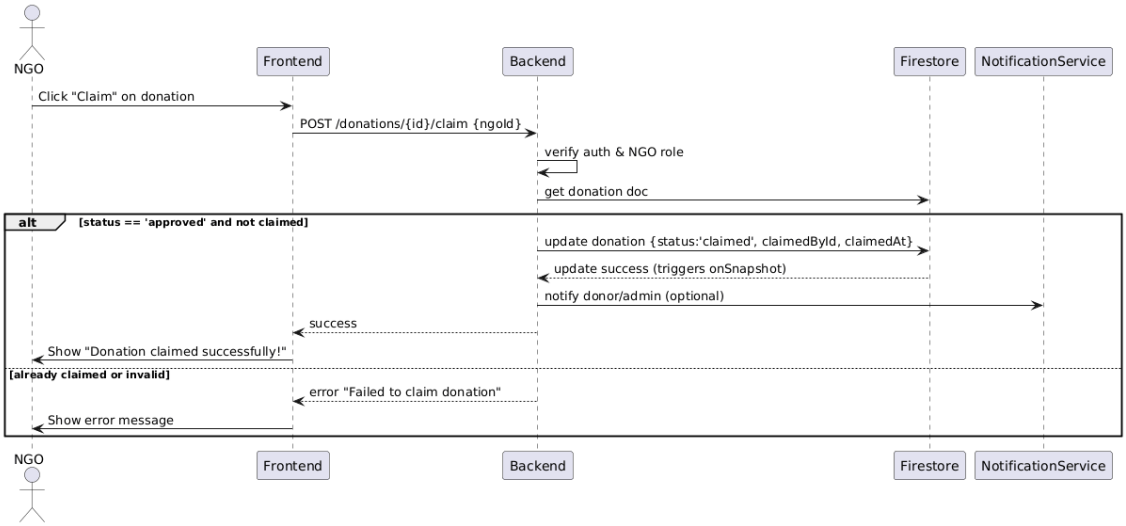


Figure 2.4.4.7: Claim Donation

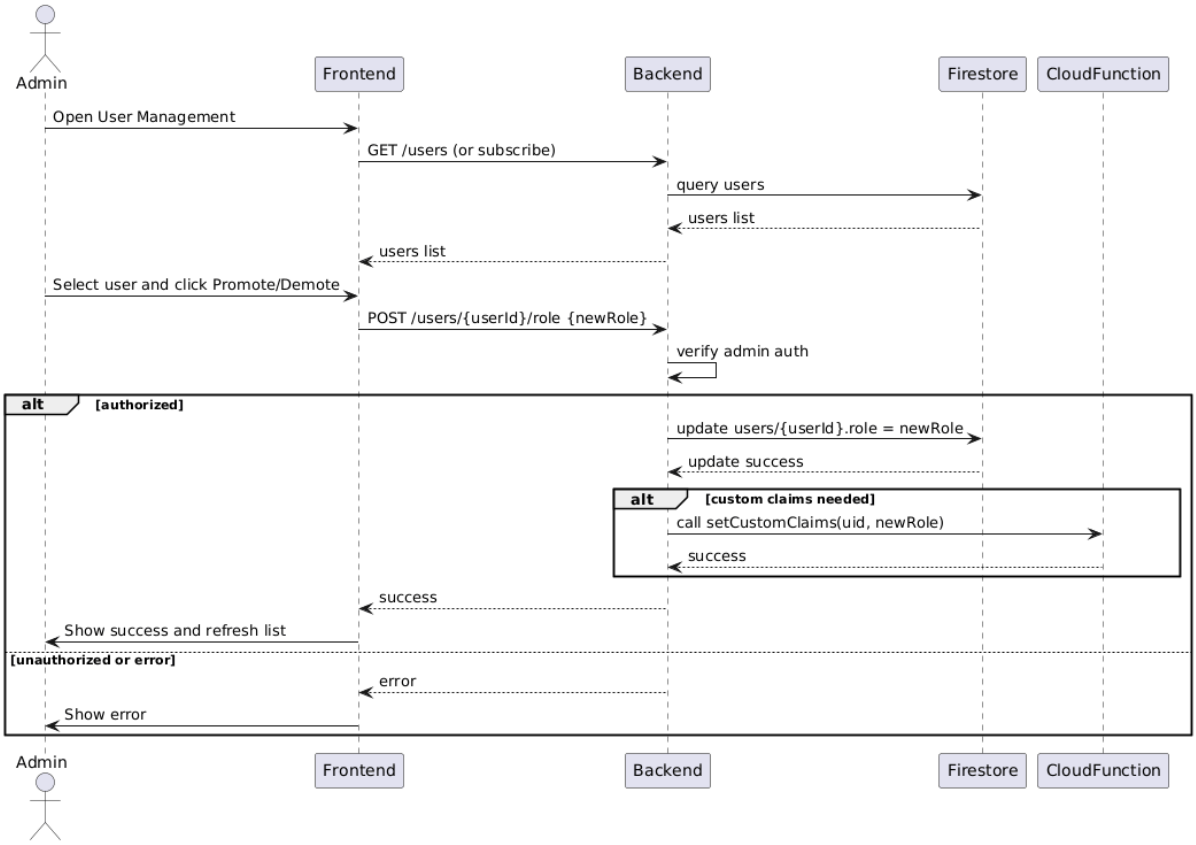


Figure 2.4.4.8: Manage User Role

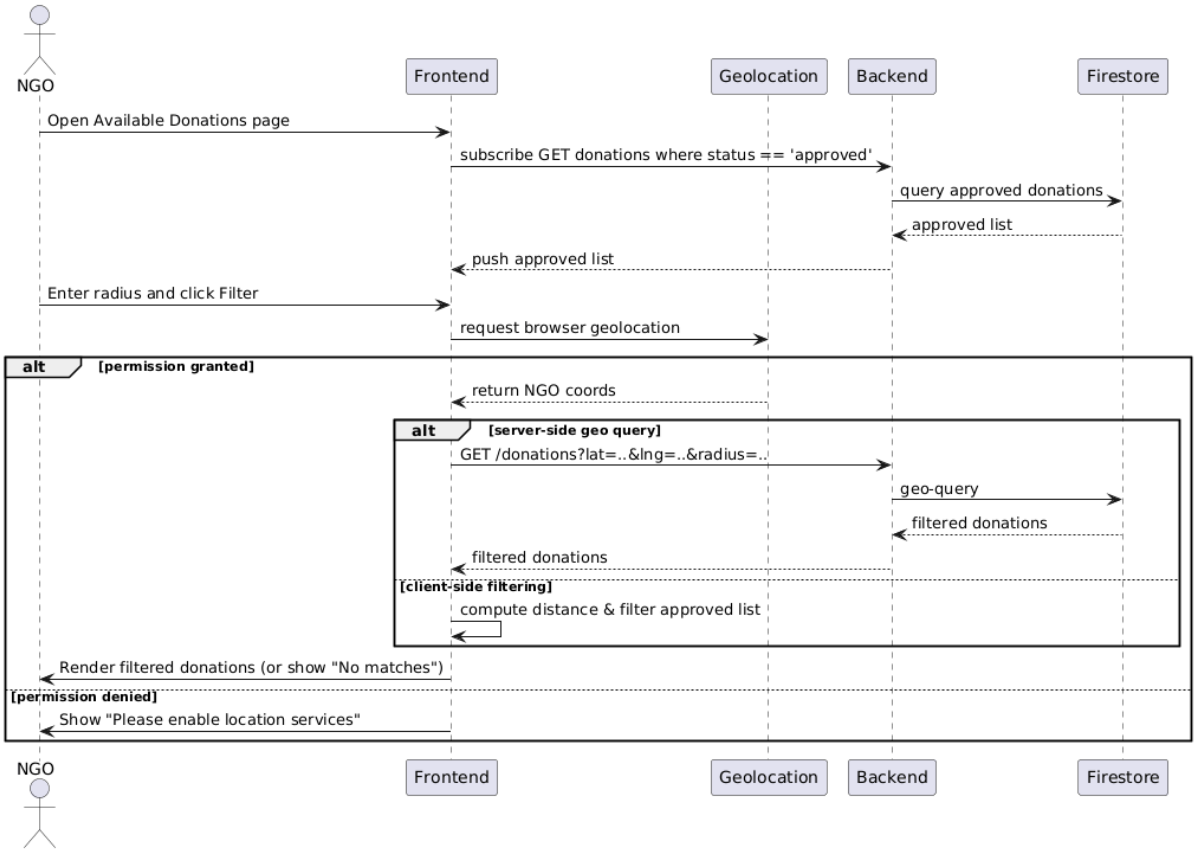


Figure 2.4.4.9: View Available Donation

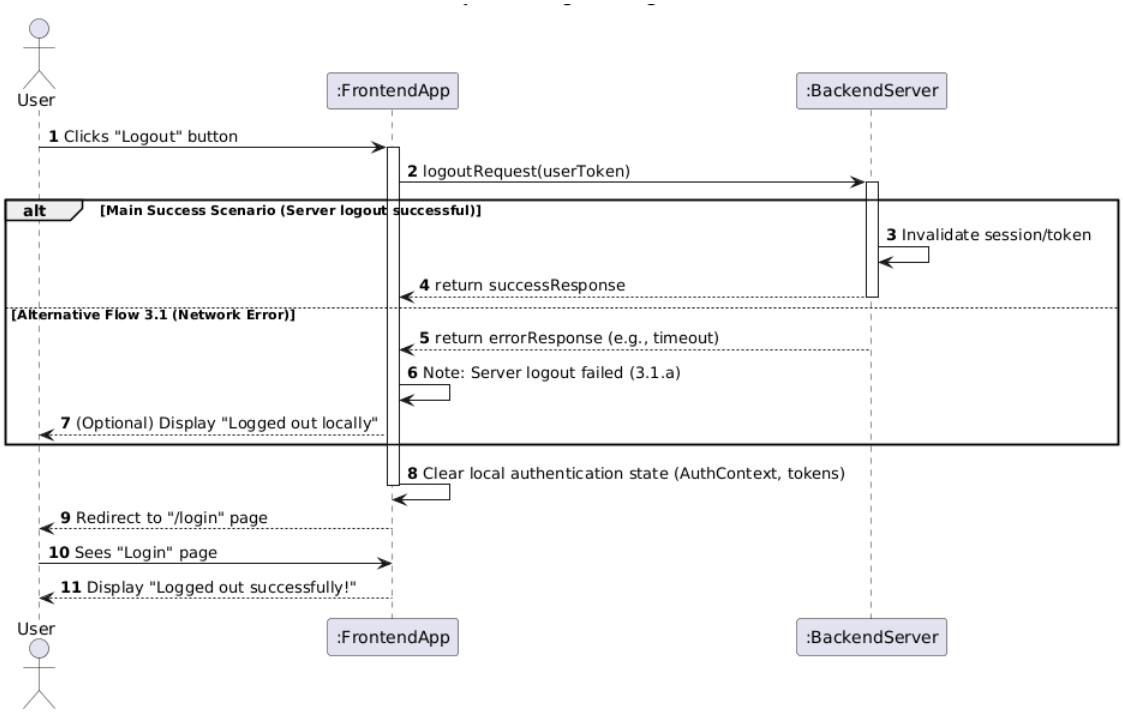


Figure 2.4.4.10: Logout

2.4.5 Class Diagram

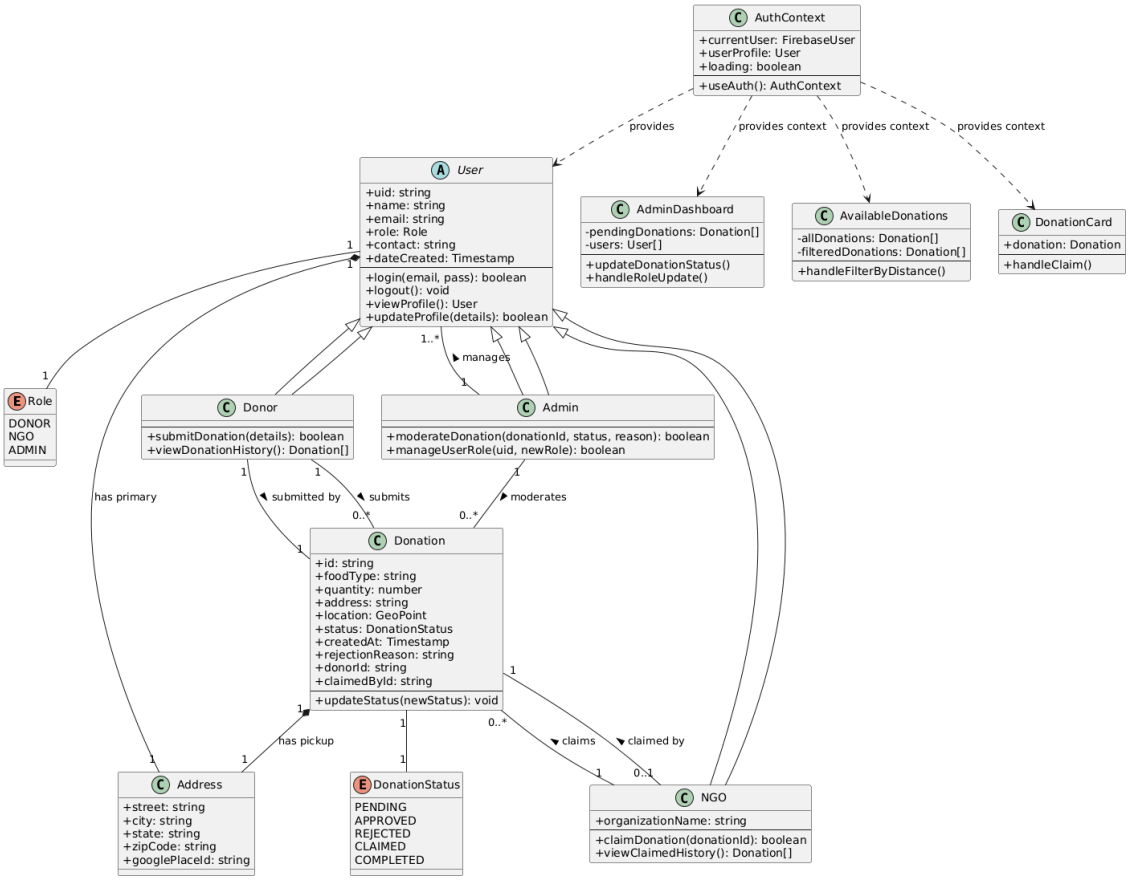


Figure 2.4.5: Class Diagram

2.4.6 ER Diagram

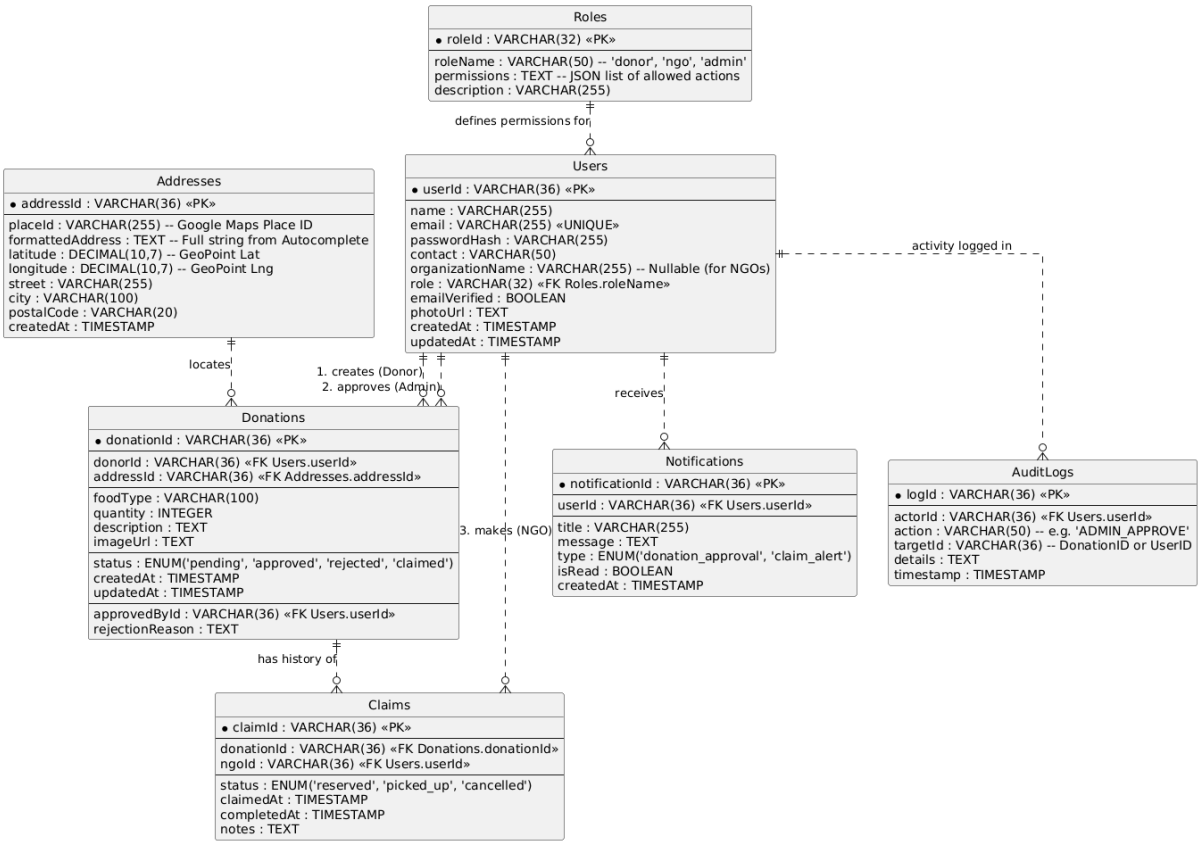


Figure 2.4.6: ER Diagram

2.5 Coding: Appendix A

All project source code (React components, context, configuration files, and Firebase rules) is attached in Appendix A.

Technology Stack Frontend Framework: React (via Vite)

Routing: React Router

Styling: Tailwind CSS with the DaisyUI component library

Notifications: React Toastify

Backend & Database Platform: Firebase

Database: Cloud Firestore (NoSQL)

Authentication: Firebase Authentication

Where the code lives:

URL: https://github.com/shahriararefin/FoodShare

2.6 Summary

The functional and non-functional requirements that influenced the creation of the FoodShare platform have been covered in detail in this chapter. It offered a thorough system design that used UML diagrams to show the actors, behaviors, and data structure of the system, from high-level use cases to the particular Firestore data schema.

Chapter 3

Software Testing

3.1 Introduction

This chapter outlines the testing strategies to validate the functionality of the project, security, and usability of the FoodShare platform. The objective was to make sure that all features and function are in accordance with the specified requirements. As the system is both robust and secure.

3.2 Testing Features

3.2.1 Feature to Be Tested

- **a.** Registration (Email/Password)
- **b.** Login (Email/Password, Google)
- **c.** Email Verification Check
- **d.** Forgot Password
- **e.** Donation Submission (with Geocoding)
- **f.** Admin Dashboard Access & Role Protection
- **g.** Donation Approval (Admin)
- **h.** Donation Rejection (Admin)
- **i.** User Role Management (Admin)
- **j.** Available Donations Page Access
- **k.** Donation Claiming (NGO)
- **l.** Location-Based Filtering (NGO)
- **m.** Profile Page Data Loading (Donor & NGO)
- **n.** User Logout

3.3 Testing Strategies

3.3.1 Test Approach

Manual End-to-End Testing: Manual Testing: This is one of the primary method of testing. It involved the creation of test accounts for each of those roles and executing the complete user flows from beginning to end like this. The Donor signs up -> Admin promotes Donor to NGO -> Donor logs in as NGO -> Admin approves a donation -> NGO claims the donation.

Security Rule Testing: The Firebase Rules Playground was used to write unit tests for all security rules. Queries for getting list, creating and updating were done for each user role to ensure permissions were correctly in place.

Cross-Browser Testing: The application was informally tested on Google Chrome and Brave with the help of Inspect.

3.3.2 Pass/Fail Criteria

Pass: The features behave the way it is described in the functional requirements sector. Data is correctly updated and stored in the database and the UI shows the change in real time. Security rules correctly allow or deny the action.

Fail: The feature generates an error (either in the user interface or console), the data is not updated accurately, the user interface fails to display the change, or a user is able to execute an action that their role should not be permitted to access.

3.4 System Testing (Test Cases with Report)

Test Case 01: Registration

Test Case			**Test Case Name:** Registration				
System: FoodShare			**Subsystem:** Authentication				
Designed by: Shahriar Arefin			**Design Date:** 22/08/2025				
Executed by: Shahriar Arefin			**Execution Date:** 12/10/2025				
Description: A new user signs up for the system by submitting accurate information.							
Pre-Condition: The user is currently on the registration page (/register)							
Step	**Name**	**Email**	**Password**	**Confirm Password**	**Response**	**Pass/Fail**	**Comment**
1	Test Donor	donor@test.com	123456	123456	Redirect to Login; Toast: "Registration Successful! Please confirm your email."	Pass	Successful registration.
2	Test Donor	donor@test.com	123456	123456	Error: "Name is required."	Pass	Validation check working.
3	Test user	donor@test.com	123	123	Error: "Password should be at least 6 characters."	Pass	Password length check working.

4	Test user	donor@test.com	123456	654321	Error: "Passwords do not match."	Pass	Confirmation check working.
Post- Condition: For Step 1, a new user document is created in Firestore, an Auth user is created in Firebase, and a verification email is sent. For Steps 2-5, no user is created.							

Test Case 02: User Login (Email and Google)

Test Case			**Test Case Name:** User Login (Email and Google)		
System: FoodShare			**Subsystem:** Authentication		
Designed by: Shahriar Arefin			**Design Date:** 22/08/2025		
Executed by: Shahriar Arefin			**Execution Date:** 13/10/2025		
Description: A user who is registered tries to log in utilizing either Email/Password or Google Sign-In.					
Pre-Condition: The user is registered and has confirmed their email address.					
Step	**Name**	**Email**	**Response**	**Pass/Fail**	**Comment**
1	Email Login	donor@test.com	Redirects to Homepage. "Logged in successfully!"	Pass	Valid credentials.
2	Email Login	donor@test.com	Error Notification: "Failed to log in."	Pass	Invalid password check.
3	Email Login	unknown@test.com	Error Notification: "Account not found."	Pass	Unregistered email check.
4	Email Login	verified@test.com	Error Notification: "Please verify your email before logging in."	Pass	Email verification check.

5	N/A	N/A	Pop-up opens; User selects account; Redirect to Homepage.	Pass	Google Auth provider works.
Post- Condition: User is authenticated, the AuthContext is populated with user data, and the user is redirected to the dashboard.					

Test Case 03: Email Verification Check

Test Case			**Test Case Name:** Email Verification Check		
System: FoodShare			Subsystem: User Authentication		
Designed by: Shahriar Arefin			Design Date: 25/08/2025		
Executed by: Shahriar Arefin			Execution Date: 12/10/2025		
Description: It verifies the system does email verification before allowing a user to log in.,					
Precondition: A new user account has just been registered but the verification link has not yet been clicked.					
Step	**Action**	**Input Data**	**Response**	**Pass/Fail**	**Comment**
1	Attempt Login	unverified@test.com	Error Notification: "Please verify your email before logging in."	Pass	System correctly blocks unverified access.
2	Check Email Inbox		Email with subject "Verify your email for FoodShare" is received.	Pass	Firebase sent the email correctly.
3	Click Verification Link		Browser opens a page saying "Your email has been verified."	Pass	Verification process works.
Post- Condition: The user's emailVerified status in Firebase Authentication is updated to true, and they are granted access to the dashboard.					

Test Case 04: Forgot Password

Test Case			**Test Case Name:** Forgot Password		
System: FoodShare			**Subsystem:** User Authentication		
Designed by: Shahriar Arefin			**Design Date:** 26/07/2025		
Executed by: Shahriar Arefin			**Execution Date:** 12/10/2025		
Description: A user requests a password reset link via their registered email.					
Post-Condition: The user logged out and on the Login page.					
Step	**Name**	**Input Data**	**Response**	**Pass/Fail**	**Comment**
1	Click "Forgot Password ?"		Redirect to/Show Password Reset form.	Pass	Link works.
2	Submit Empty Form	(empty)	Error Notification: "Please enter your email."	Pass	Validation check works.
3	Enter Invalid Format	invalid-email	Error Notification: "Invalid email format."	Pass	Format check works.
4	Enter Unregistered Email	unknown@test.com	Error Notification: "Account not found."	Pass	System correctly identifies non-users.
5	Enter Valid Email	donor@test.com	Success Message: "Check your email for password reset instructions."	Pass	Reset email triggered.
Post- Condition: A password reset email containing a secure link is sent to the user's inbox.					

Test Case 05: User Logout

Test Case	**Test Case Name:** User Logout
System: FoodShare	Subsystem: Logging out
Designed by: Shahriar Arefin	Design Date: 25/08/2025
Executed by: Shahriar Arefin	Execution Date: 12/10/2025
Description: It is for cancelling or terminating the session.	
Precondition: User has completed tasks in the platform and wants to leave by logging out	

Step	**Name**	**Input Data**	**Response**	**Pass/Fail**	**Comment**
1	Click "Logout" Button		User is redirected to the Login page. Toast: "Logged out successfully."	Pass	Session terminated correctly.
2	Check Auth State		Verify AuthContext.currentUser is null.	Pass	Internal state updated.
3	Access Protected Route	Navigate to /profile	User is blocked and redirected to Login page.	Pass	ProtectedRoute component correctly blocks unauthenticated access.
4	Browser Back Button	Click Back	User is shown Login page (or cannot access previous session).	Pass	Session history is secure.
Post- Condition: The user's session is destroyed, local storage tokens are cleared, and they must log in again to access any private features.					

Test Case 06: Donation Submission

Test Case			**Test Case Name:** Donation		
System: FoodShare			**Subsystem:** Donation Lifecycle		
Designed by: Shahriar Arefin			**Design Date:** 09/08/2025		
Executed by: Shahriar Arefin			**Execution Date:** 12/10/2025		
Description: A donor submits a donation using Google Places Autocomplete to tag their location and fills other forms as well.					
Pre-Condition: Logged in with role: 'donor' and is on the "Donate" page.					
Step	**Name**	**Input Data**	**Response**	**Pass/Fail**	**Comment**
1	Fill Food Details	Food Type: Rice Quantity: 50	Input fields accept data.	Pass	Form inputs working.
2	Enter Contact Info	Contact: 01700000000	Input field accepts numeric string.	Pass	Validation accepts contact.
3	Address Autocomplete	Type Dhanmondi in Address field.	Google Maps suggestions dropdown appears (e.g., "Dhanmondi, Dhaka").	Pass	Places API is active.
4	Select Address	Click on a suggestion.	The input field is populated with the full address. Internally, lat and lng are saved.	Pass	Geocoding successful.
5	Submit Form	Click "Submit for Review".	Redirect to Profile page. Toast: "Donation submitted successfully!"	Pass	Database write successful.

6	Negative Test: Manual Address	Type Unknown Place but do not select from dropdown.	Error Notification: "Please select a valid address from the suggestions."	Pass	System enforces valid geocoding.
7	Negative Test: Empty Fields	Leave Quantity empty.	Error Notification: "Please fill in all fields."	Pass	HTML5/React validation working.
Post- Condition: A new document is created in the donations Firestore collection with status: 'pending' and a valid location GeoPoint.					

Test Case 07: Profile Page Data Loading

Test Case			**Test Case Name:** Profile Page Data Loading		
System: FoodShare			**Subsystem:** User Profile & History		
Designed by: Shahriar Arefin			**Design Date:** 02/09/2025		
Executed by: Shahriar Arefin			**Execution Date:** 12/10/2025		
Description: Verifies that the Profile page correctly fetches user details and the specific donation history based on the user's role.					
Pre-Condition: User is logged in. Database contains donations submitted by the Donor and claimed by the NGO.					
Step	**Name**	**Input Data**	**Response**	**Pass/Fail**	**Comment**
1	Navigate to Profile	Click "Profile" in Navbar.	Page loads without errors. User Name, Email, and Role are displayed correctly at the top.	Pass	User details fetch successful.

2	Test as Donor: Check History	Login as Donor with existing donations.	Section "My Donations" is visible. List shows donations with correct statuses (pending, approved, rejected).	Pass	Query where("donorId", "==", uid) is correct.
3	Test as NGO: Check History	Login as NGO with claimed donations.	Section "Claimed Donations" is visible. List shows donations the NGO has successfully claimed.	Pass	Query where("claimedById", "==", uid) is correct.
4	Check Empty State	Login as a new user (no history).	"My Donations" / "Claimed Donations" section displays "No donations found" (or similar empty state message).	Pass	Empty state handled gracefully.
5	Verify Real-Time Updates	(In a separate tab) Change a donation status from pending to approved as Admin.	The Donor's profile list updates instantly to show "Approved" without refreshing the page.	Pass	Firestore onSnapshot listener is working.
Post- Condition: The user interface accurately reflects the current state of the database regarding the user's personal activity.					

Test Case 08: Admin Role-Based Access

Test Case			**Test Case Name:** Admin Role-Based Access		
System: FoodShare			**Subsystem:** Authentication & Routing		
Designed by: Shahriar Arefin			**Design Date:** 28/08/2025		
Executed by: Shahriar Arefin			**Execution Date:** 12/10/2025		
Description: Verifies that the Admin Dashboard and associated routes are inaccessible to users without the 'admin' role.					
Pre-Condition: User is currently logged in with role: 'donor' (or 'ngo').					
Step	**Name**	**Input Data**	**Response**	**Pass/Fail**	**Comment**
1	Direct URL Access	Type http://.../admin in the browser address bar.	User is redirected to the Home page or Login page. Toast/Error: "Access Denied" or "Admins only".	Pass	ProtectedRoute component correctly blocks the route.
2	Check Navbar UI	Observe the Navigation Bar.	The "Admin Dashboard" link is not visible.	Pass	Conditional rendering works based on user.role.
3	Positive Control	Log out and Log in as Admin. Click "Admin Dashboard".	User has the access and dashboard.	Pass	Role verification allows correct access.
4	Database Rule Test	(Console/Code) Attempt to write to donations collection with status: 'approved' as a Donor.	Firebase returns: Permission Denied (insufficient permissions).	Pass	Firestore Security Rules are enforcing role-based writes.
Post- Condition: Unauthorized users remain on public or user-specific pages, preserving the security of administrative functions.					

Test Case 09: Donation Approval

Test Case			**Test Case Name:** Donation Approval		
System: FoodShare			**Subsystem:** Donation Management		
Designed by: Shahriar Arefin			**Design Date:** 08/09/2025		
Executed by: Shahriar Arefin			**Execution Date:** 12/10/2025		
Description: An Admin can approve a 'pending' donation.					
Pre-Condition: 1. Admin is logged in. 2. A donation exists with status: 'pending'.					
Step	**Name**	**Input Data**	**Response**	**Pass/Fail**	**Comment**
1	Go to Dashboard	Click "Admin ".	The "Pending Donations" list loads, displaying the test donation.	Pass	Data fetching works.
2	Verify Donation Details	Click "View Details" on the pending item.	Modal/Page opens showing correct Food Type, Quantity, and Location map.	Pass	Read operation works.
3	Approve Donation	Click the "Approve" button.	System asks for confirmation (if implemented) or processes immediately.	Pass	Button is interactive.
4	Check UI Feedback	Confirm action.	Toast message: "Donation Approved Successfully." The item is removed from the "Pending" list (or status updates to "Approved").	Pass	UI updates correctly.
5	Verify Database State	Check Firestore Console (or Donor Profile).	The specific document's status field is now approved.	Pass	

					Write operation successful.
6	Verify NGO Visibility	Log out and Log in as NGO.	The donation now appears in the "Available Donations" feed.	Pass	Approval correctly exposes data to NGOs.
Post- Condition: The donation status is approved and triggers visibility for NGOs and potentially sending a notification to the Donor.					

Test Case 10: NGO Donation Claiming

Test Case			**Test Case Name:** NGO Donation Claiming		
System: FoodShare			**Subsystem:** Donation Lifecycle		
Designed by: Shahriar Arefin			**Design Date:** 13/09/2025		
Executed by: Shahriar Arefin			**Execution Date:** 12/10/2025		
Description: Verifies that a NGO can successfully claim an 'approved' donation.					
Pre-Condition: 1. NGO is logged in. 2. A donation exists with status: 'approved'.					
Step	**Name**	**Input Data**	**Response**	**Pass/Fail**	**Comment**
1	Navigate to "Find Donations"	Click "Donations" link.	List of all 'approved' donations is displayed.	Pass	Query where("status", "==", "approved") works.
2	Select a Donation	Click "View" on a specific item.	Detail page opens showing pickup address	Pass	Details loaded correctly.

			and food details.		
3	Claim Donation	Click "Claim This Donation".	System asks for confirmation: "Are you sure you want to claim this?"	Pass	Confirmation dialog appears.
4	Confirm Claim	Click “Confirm".	Toast: "Donation claimed successfully!" Redirect to Profile or update UI to show "Claimed".	Pass	Claim logic executed.
5	Verify Database State	Checkingthe do Firestore Console.	The document's status updates to claimed, and claimedById is set to the NGO's user ID.	Pass	Database write verified.
6	Negative Test: Double Claim	(Simulate 2nd NGO) Try to claim the same ID.	Button is disabled OR Error: "This donation has already been claimed."	Pass	Race condition/Status check works.
Post- Condition: The donation is not visible in the public "Available" list. The claiming NGO sees it in their "Claimed History".					

Test Case 11: Donation Rejection

Test Case	**Test Case Name:** Donation Rejection
System: FoodShare	**Subsystem:** Donation Management
Designed by: Shahriar Arefin	**Design Date:** 16/09/2025
Executed by: Shahriar Arefin	**Execution Date:** 12/10/2025
Description: Verifies that an Admin can reject a 'pending' donation, preventing it from being displayed to NGOs.	
Pre-Condition: 1. Admin is logged in. 2. A donation exists with status: 'pending'.	

Step	**Name**	**Input Data**	**Response**	**Pass/Fail**	**Comment**
1	Navigate to Dashboard	Click "Admin Dashboard".	The "Pending Donations" list loads.	Pass	Admin view accessible.
2	Select Donation	Click "View Details" on a pending item.	Modal displays donation details (Photo, Food Type).	Pass	Details loaded.
3	Reject Donation	Select the "Reject" button.	A popup appears asking for a reason (optional but recommended).	Pass	Reject flow initiated.
4	Enter Reason	Reason: "Image is unclear" or "Safety concern".	System accepts the text input.	Pass	Rejection reason capture.
5	Confirm Rejection	Click "Confirm".	Toast: "Donation Rejected." Item is removed from the "Pending" list.	Pass	UI updates immediately.
6	Verify Database State	Check Firestore Console.	Document status updates to rejected. (Optional:	Pass	Database write successful.

			rejectionReason field is populated).		
7	Verify Donor View	Log in as the Donor and check "My Donations".	The specific donation shows a "Rejected" badge/status.	Pass	
Post- Condition: The donation status is rejected. It is not visible to NGOs in the "Available Donations" feed.					

Test Case 12: User Role Management

Test Case			**Test Case Name:** User Role Management		
System: FoodShare			**Subsystem:** User Management		
Designed by: Shahriar Arefin			**Design Date:** 28/09/2025		
Executed by: Shahriar Arefin			**Execution Date:** 12/10/2025		
Description: Verifies that an Admin can update a registered user's role (e.g., changing a 'Donor' to an 'NGO' or 'Admin').					
Pre-Condition: 1. Admin is logged in. 2. A target user exists with role: 'donor'.					
Step	**Name**	**Input Data**	**Response**	**Pass/Fail**	**Comment**
1	Navigate to "Manage Users"	Click "Users" in Admin Sidebar.	A table listing all registered users (Name, Email, Role) is displayed.	Pass	Data fetching successful.
2	Select Target User	Locate donor@test.com.	The user row is visible with current role "Donor".	Pass	User found.
3	Modify Role	Click "Edit" or use Role Dropdown. Select "NGO".	The dropdown updates to	Pass	UI component active.

			reflect the new selection.		
4	Confirm Update	Click "Save Changes".	Notification: "User role has been updated successfully." The table now shows "NGO" for this user.	Pass	Update action triggered.
5	Verify Database Current State	Visit and check Firestore Console.	The individual users document's role field is now 'ngo'.	Pass	Database write successful.
6	Verify User Access	Log out Admin. Log in as the Target User.	The user now sees the "Find Donations" (NGO) link instead of "Donate Food".	Pass	Permissions effectively changed.
Post- Condition: The user's role is updated and stored in the database and their access rights (navbar links, protected routes) are immediately permitted upon their next login or refresh.					

Test Case 13: Available Donations Page Access

Test Case	**Test Case Name:** Available Donations Page Access
System: FoodShare	**Subsystem:** Routing & Data Fetching
Designed by: Shahriar Arefin	**Design Date:** 01/10/2025
Executed by: Shahriar Arefin	**Execution Date:** 12/10/2025
Description: Verifies that the "Available Donations" page is accessible to NGOs, protected from guests, and correctly filters out non-approved items.	
Pre-Condition: 1. Database contains donations with statuses: pending, approved, and claimed. 2. User is logged out initially.	

Step	**Name**	**Input Data**	**Response**	**Pass/Fail**	**Comment**
1	Guest Access	Navigate URL: /donations	Redirect to Login Page. Error: "You must be logged in."	Pass	Protected route works.
2	Donor Access	Log in as Donor. Try to access page.	Redirect to Donor Dashboard OR Access Denied message. (Depending on policy).	Pass	Role-based restriction works.
3	NGO Access	Log in as NGO. Click "Find Donations".	Page loads successfully.	Pass	Authorized access granted.
4	Verify Content Filter	Inspect the list of donations.	Only donations with status: 'approved' are visible.	Pass	Filter where('status', '==', 'approved') is active.
5	Negative Data Test	Search for a specific pending or claimed donation ID.	The item is not displayed in the list.	Pass	Pending/Claimed items are correctly hidden.

6	Pagination/Load	Scroll down (if list is long).	Additional approved donations load (Lazy Loading).	Pass	UI handles data efficiently.
Post- Condition: The user is served the correct view based on their authentication state and role permissions.					

Test Case 14: Location-Based Filtering

Test Case		**Test Case Name:** Location-Based Filtering			
System: FoodShare		**Subsystem:** Search & Discovery			
Designed by: Shahriar Arefin		**Design Date:** 26/09/2025			
Executed by: Shahriar Arefin		**Execution Date:** 12/10/2025			
Description: User selects filtering and inputs distance.					
Pre-Condition: User is in the available donation page.					
Step	**Name**	**Input Data**	**Response**	**Pass/Fail**	**Comment**
1	Click to "Find Donations"	Click Navbar link.	All approved donations are displayed (regardless of location).	Pass	Loads successfully.
2	Apply Location Filter	Enter "Dhanmondi" in the "Filter by Location" input/search bar.	The list refreshes.	Pass	Filter input accepts text.
3	Verify Filter Results	Inspect the displayed items.	Donations address searched contains "Dhanmondi" are visible. Donations from	Pass	Filtering logic works correctly.

			"Gulshan" are hidden.		
4	Negative Test: No Match	Enter a location with no donations (e.g., "Sylhet").	The list becomes empty. Message: "No donations found in this area of selected distance."	Pass	Empty state handled gracefully.
5	Clear Filter	Click "Clear" or delete text.	The list reverts to showing all approved donations.	Pass	Reset functionality works.
6	Partial Match	Enter "Dha".	Donations from "Dhaka", "Dhanmondi", etc., are displayed.	Pass	String matching is flexible/inclusive.
Post- Condition: The user interface displays a filtered subset of data without altering the actual database records.					

3.5 Summary

The FoodShare system testing phase needs to complete the implementation of 14 different test cases that were separated into four main modules, i.e. Authentication, Donor Functionality, Admin Management, and NGO Operations. The main aim was to verify the logic of the system, role base security and lastly integrations of the database.

Chapter 4

Deployment and Maintenance

4.1 Introduction

This chapter discusses the implementation procedure of the FoodShare application and how it gets deployed and maintained.

4.2 Software Release Life Cycle (SRLC)

The project was an iterative approach similar to Agile.

1. **Planning:** The first requirement and feature were determined and it is the feasibility.
2. **Development:** Small, logical sprints were used to develop features. Firstly, auth then forms then dashboards and then location.
3. **Testing:** After completing each sprint, manual testing and security rule were performed.
4. **Release:** Application is built using React and Vite application using the npm run build command and this is a production-ready command.
5. **Deployment:** Product has been deployed to **Firebase Hosting** using the Firebase CLI command firebase deploy --only hosting. Firebase Hosting was selected because of its global CDN, automatic SSL and smooth going with the Firebase backend.
6. **Maintenance:** After deployment the maintenance starts monitoring the Firebase Console for database performance check and errors. The bugs are solved by updating the code and redeployed.

Chapter 5

User Manual

5.1 Introduction

This manual here described the fundamental functioning processes and shows the UI of the FoodShare application.

5.2 Project Functionalities

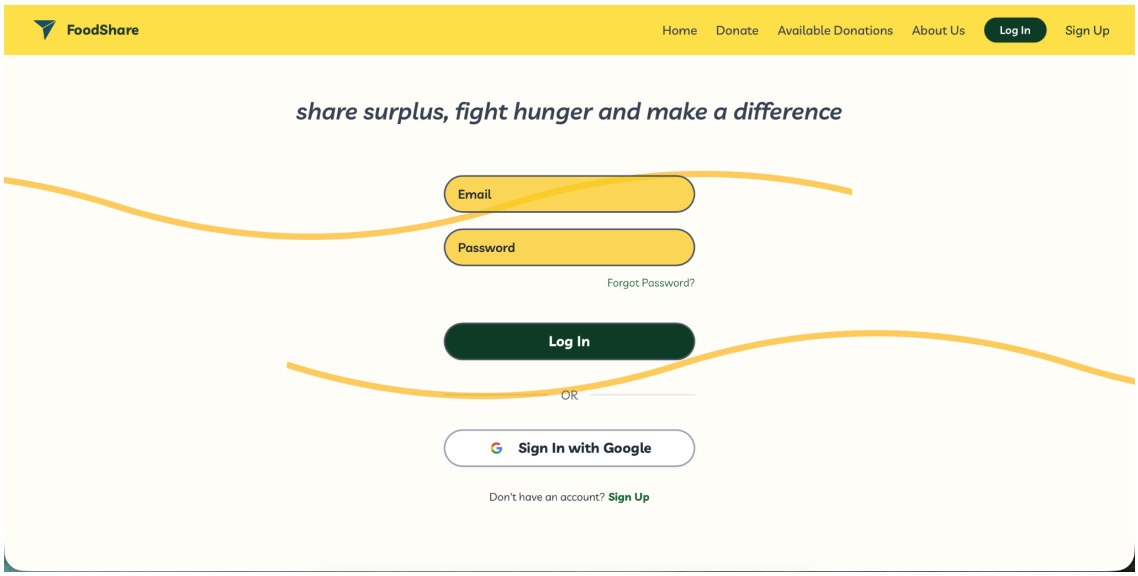


Figure: User/Admin Login

Submit Donation

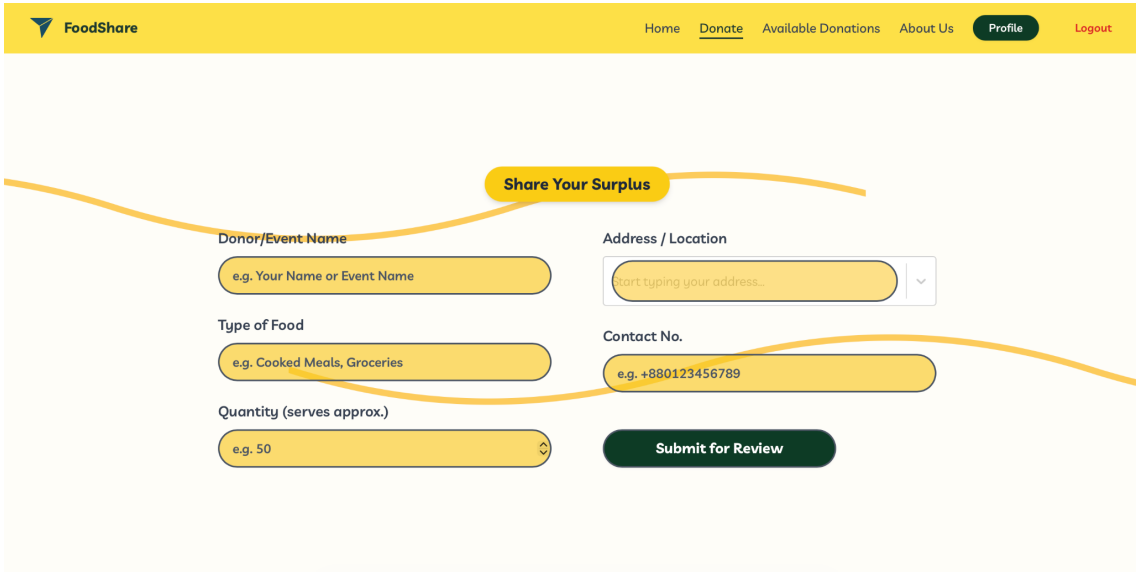


Figure: Submit Donation

Claim Approved Donation

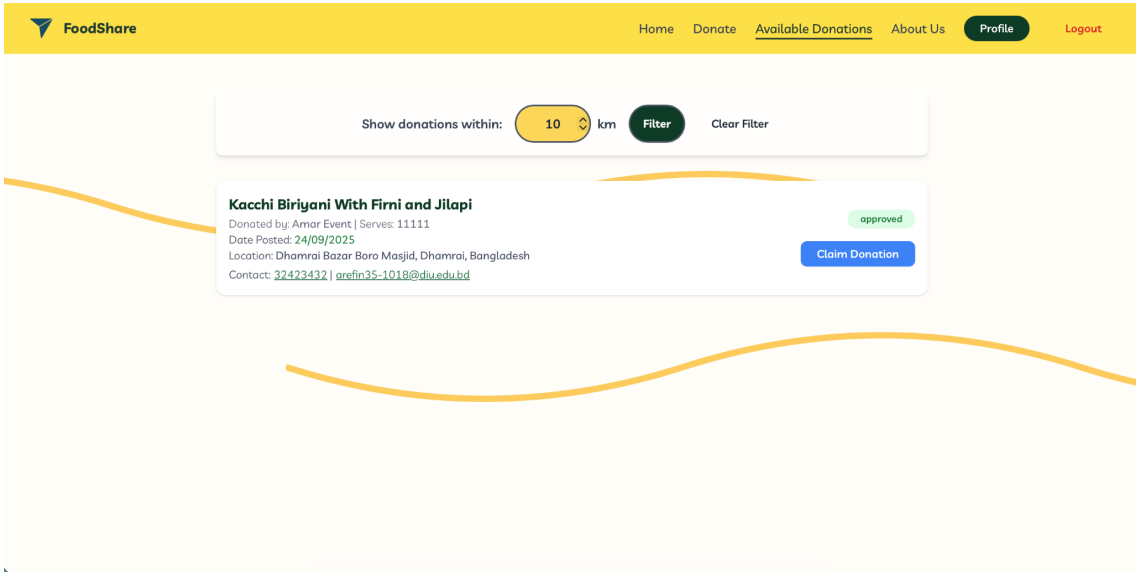


Figure: Claim Approved Donation

Home Page

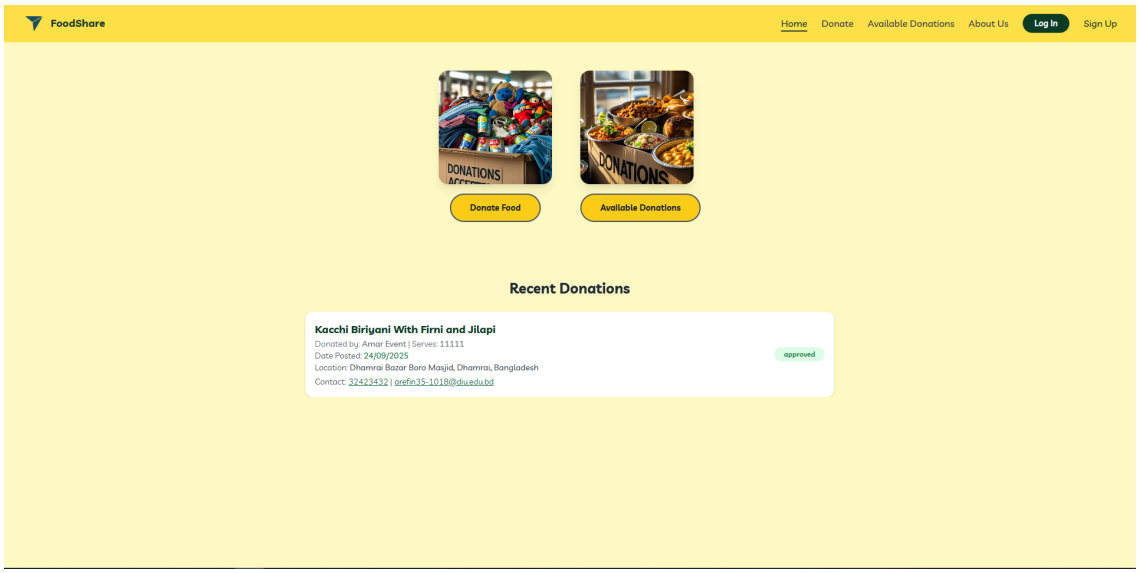


Figure: Home Page

Admin Dashboard

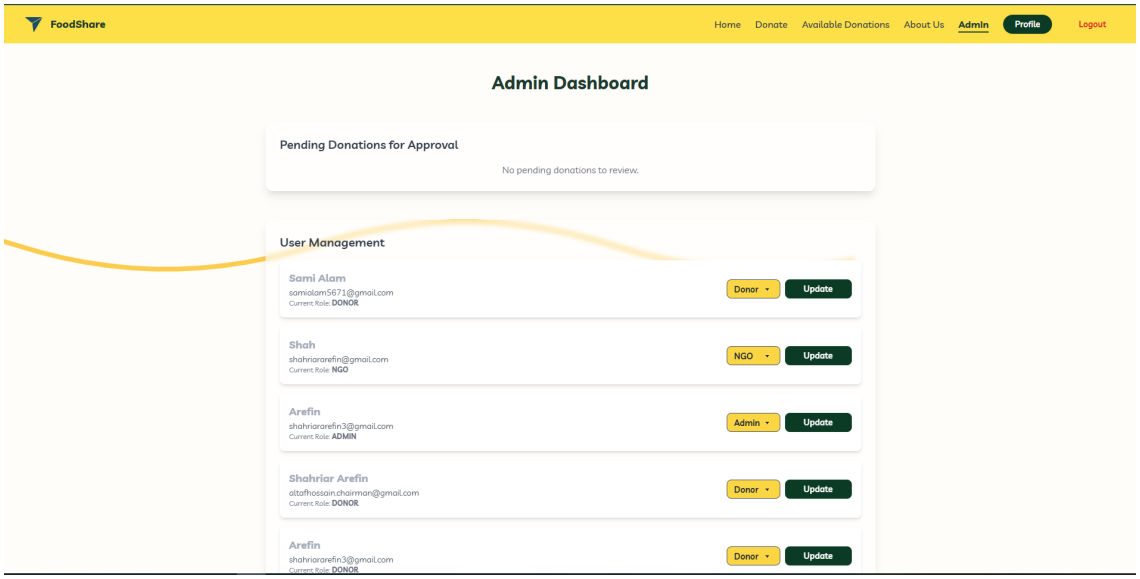


Figure: Admin Dashboard

5.3 Summary

This chapter has perfectly outlined the processes for the FoodShare application. One needs to follow these steps to complete the operation. Donor can submit food and quantity, Admins can maintain system integrity through moderation and approves or rejects submitted donations after reviewing and verification and NGOs can efficiently locate and claim foods.

Chapter 6

Project Summary

6.1 Introduction

This project "FoodShare" has successfully demonstrated the creation of a full-stack, real-time web version application to solve and eradicate the social issue of surplus food waste and remove hunger.

6.2 Project Limitation

- The platform can be accessed via web only not a mobile app.
- A working internet connection is needed for the platform to operate functionally for all users.
- The system is significantly influenced by the careful reviewing of donations by the admin.
- Right now, the notifications are not automated. NGOs and admins are required to check regularly for new donations.

6.3 Scope

The project scope has been properly completed. It covers the complete lifecycle of a food donation, from submission for donation to claiming by NGO. However, the project does not include features such as real-time chat, a rating system and automated push notifications.

6.4 Future Work

- **Email Notifications:** Utilizing firebase functions that automatically dispatches mails for all NGOs when approval occurs for a new donation.
- **Rating System:** Adding a review and rating for donors and NGOs to rate each other after successfully complete a donation. It will help building platform trust.
- **Real-time Chat:** Establish a fundamental chat feature for NGOs to coordinate pickup details directly with donors and share delivery status.
- **Native Mobile App:** Develop a cross-platform or native mobile app to improve accessibility for users on the go and to perform task with more ease.

6.5 Conclusion

The FoodShare platform is an effective proof as a prototype and a very powerful base to a complete social impactful platform. It is an effective use of a current technology stack to provide an easy to maintainable, scalable and cost-effective solution. The project achieves all its main factors and in a safe and effective way through a connection of food donors to NGOs which gives a clear channel to reduce food wastage and help the needy people around us.

REFERENCES

1. Jayaveeran, M. K., Reena, E. A. P., Muthukumarasamy, M. S., & Sriram, M. (2025). Home2Home: A Community-Driven App for Sustainable Living. INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT. https://doi.org/10.55041/ijsrem52834
2. Authentication | DashCode. https://dashcode-doc.codeshaper.tech/development/authentication.html
3. Safuan, N., Anuar Kamal, A., & Hassan, R. (2024). A data-driven approach to unveiling mental health realities among undergraduate students at the International Islamic University Malaysia (IIUM) using machine learning: A case study. https://core.ac.uk/download/639867717.pdf
4. Norazlina, R. (2023). Waste To Product: Potential Of Concrete Waste Additive For Strength Improvement Of Peat Soil. https://core.ac.uk/download/613052121.pdf
5. (2024). Advancement in ICT: Exploring innovative solutions (AdICT) Series 2/2024. https://core.ac.uk/download/647831047.pdf
6. React. (n.d.). *React—A JavaScript library for building user interfaces*. Retrieved October 31, 2024, from https://reactjs.org/
7. Firebase. (n.d.). *Firebase Documentation*. Retrieved October 31, 2024, from https://firebase.google.com/docs
8. Google. (n.d.). *Google Maps Platform Documentation*. Retrieved October 31, 2024, from https://developers.google.com/maps
9. Tailwind CSS. (n.d.). *Tailwind CSS—Rapidly build modern websites without ever leaving your HTML*. Retrieved October 31, 2024, from https://tailwindcss.com/
10. DaisyUI. (n.d.). *DaisyUI—Tailwind CSS Components*. Retrieved October 31, 2024, from https://daisyui.com/