Group Project Log

Note: *all information must be filled out. You must hand in the project log along with each group project deliverable for this course (e.g., milestones, proposals, reports). The percentage of work allocated to each group member must add up to 100%.*

Group Name:	Group no 6

Group Members:	Arpit Ribadiya, Jay Kania, Lav Patel, Neha Karkhanis, Viraj Joshi

Deliverable:	Project Report

Group Member Name	**Work Done (%)**
Arpit Ribadiya	20%
Jay Kania	20%
Lav Patel	20%
Neha Karkhanis	20%
Viraj Joshi	20%
Total:	**100%**

TECHNICAL REPORT

PROJECT GROUP 6

LASTSERVE:
LET'S END HUNGER TOGETHER

Members and Contributors

Arpit Ribadiya
B00932018
ar304626@dal.ca

Jay Kania
B00923785
jy440982@dal.ca

Lav Patel
B00910579
lv842182@dal.ca

Neha Karkhanis
B00903157
nh601176@dal.ca

Viraj Joshi
B00924759
viraj.joshi@dal.ca

Faculty of Computer Science
Dalhousie University

12, April, 2022

ABSTRACT

Every evening, restaurants with leftover food will update their public posting on our website with the amount of food that is still available. Anybody in need of food is welcome to do so by scheduling a time slot and arriving to pick up their meal for that evening. People won't go to bed hungry as a result, and restaurant food waste will be decreased.

KEYWORDS

Donation, Food-waste, Free-Food, Order, Posts, Restaurants, Students, Subscription, Time-slots, Volunteer.

CONTENTS

1. INTRODUCTION

There are disproportionately many students, people with limited resources, and those in need. We view having access to food as a fundamental human right, not a privilege. Nobody should have to struggle to get enough food each day or go to night hungry. Restaurants will be able to feed the hungry in their local area because of this effort.

This project aims to reduce the quantity of food that restaurants throw out after service. This includes food that hasn't been consumed yet or is soon to spoil. According to the most recent analysis, 1.3 billion tons of food are wasted annually. According to recent studies, there are $162 billion in annual expenditures associated with food waste and $2 billion in lost revenues. The second largest source of food waste is outside the household, which includes trash generated at cafes, restaurants, canteens, snack bars, and other establishments. 870 million people could be fed if only a quarter of the food lost or wasted annually in the world was saved. The objective of this effort is to help those in need by distributing food that would otherwise go to waste.

Additionally, we hope to inspire others to donate and make a difference through our website because we think the world has enough resources to meet everyone's needs but not their greed. With the help of LastServe, restaurants can easily register and distribute food that would otherwise go to waste.

1.1 Live Project URL

In this Section, you are expected to provide the URL from which your application can be accessed throughout the term:

https://lastserve-a3.netlify.app/

backend: https://csci5709-a3-backend.onrender.com

This Section, must also include the the Git Lab repository for your project's code:

https://git.cs.dal.ca/ribadiya/csci_5709_web_group6

Backend: https://git.cs.dal.ca/vjoshi/csci_5709_group6_backend

2. BACKGROUND

2.1 Competitive Landscape

All types of customers that need food on the fly can access the free website platform LastServe. LastServe acts as an intermediary between restaurants and clients, offering free services to both parties.

Our brand's current competitors are Food Rescue, Share the meal, Good to go, and Flash food.

- **Overall Layout and Design:** LastServe is user-friendly, with easily accessible features and simple registration procedures. All of the forms are created so that the user would feel comfortable entering their information without worrying that their privacy will be violated. Since we established a uniform layout and design elements, the user can easily navigate between each component. The colour design on our website is muted in order to relax a customer who is already hungry.

- **Speed:** In order to improve query response times, we are implementing a NoSQL database that will be kept on-site. The website's design also makes it possible for customers to complete the process in just three clicks, saving them time.

- **Mission Specific and service oriented:** Our goal is to reduce food waste produced by restaurants in order to combat world hunger. Unlike other food services, LastServe offers cooked food as opposed to raw ingredients. For those who are limited for time, this saves both time and money.

- **Posts are updated daily:** Our goal is to reduce food waste produced by restaurants to combat world hunger. Unlike other food services, LastServe offers cooked food as opposed to raw ingredients. For those who are limited for time, this saves both time and money.

2.2 Problem and Approach

Online, there are no open public platforms where local restaurants can register and donate all their excess food. As a result, food that may have been served as a nighttime meal for someone without access is wasted.

The primary reason we built this website was to give restaurants a free, public forum on which they donate all of their unsold food to those in need. A publicly available platform will entice additional businesses to donate, and this will lead to the development of a community of restaurants that support the underprivileged.

3. APPLICATION DETAILS

The importance of our organization’s support for the poor and reliance on restaurants to give food for the hungry is something that LastServe wants to emphasize. The landing page contains the majority of the information in order to ensure that customers are aware of our campaign. No in-depth reading would be required for a new user to comprehend our objectives. We've included pictures as well in order to visually convey our website's motto more effectively.

3.1 Target User Insights

This website primarily serves restaurant owners, volunteers, those on tight budgets, and donors. Our potential customers include people with restricted resources, including students, the unemployed, and those with little free time. The second category of users is restaurant owners or managers who sign up their restaurants on our website to enable themselves to post daily updates. The website would have a listing for the restaurant and daily updates would include the

amount of food remaining. An additional set of users would be those who have extra time on their hands and can help restaurants pack food and feed the needy voluntarily.

The site's users and material are managed by the administrators, who are LastServe coordinators. They will be given a special set of tools that will enable them to accomplish this by approving user registrations, reviewing or removing restaurant content as necessary, and controlling user involvement.

3.2 User-Centered Design Approach

Our website's major focus consists of consumers of food and restaurant owners and managers.

The first step we took to follow a user-centered design was to identify the intended end users of the product and to establish the usage context. The primary goal was to determine why and how these users might be interested in our website.

In order to ensure that the needs of the users are addressed, it was crucial to group our data in order to create a set of requirements and user goals.

When creating the website, particular consideration was made to prioritise the needs of our users in terms of navigational convenience, comprehension, enjoyment, clarity, and attention to detail. Users' wants and goals were given priority throughout the design process over commercial objectives.

Following efforts were taken to maintain a user-centred design approach.

- Analysed user personas, scenarios and use cases: A user persona was created at the beginning of the design phase. This persona was used throughout the design process and scenarios were built around it. Developing a user persona helped us identify the patterns in behaviours, attitude, goals, needs, skills of the end user.

- A thorough comprehension of the needs and tasks of the user: To follow the user-centered design approach we paid our main attention to what a user requires.

- Iterative development: We broke down the software in various parts throughout the development to ensure testing of the working software with keeping in mind the user requirements.

- Usability testing: We performed tests to see how user-friendly and simple our website is. We discovered faults in our approach to designing thanks to the ongoing testing, so we kept changing the layout until we had all we needed for the ideal website design.

Adopting these methods together helped us design our user-centred approach and achieve the following results

1. Transparency: maximising transparency enabled us to stay in touch with consumers and their needs
2. Cost Reduction: We were able to address early development errors by employing continuous usability testing and an agile development process, which ultimately helped us cut expenses significantly later on.

3. Higher Customer Satisfaction: Customer satisfaction was attained by delivering the finished product as promised with simple navigation and conveniently accessible compone

3.2.1 Information Architecture

Explanation:

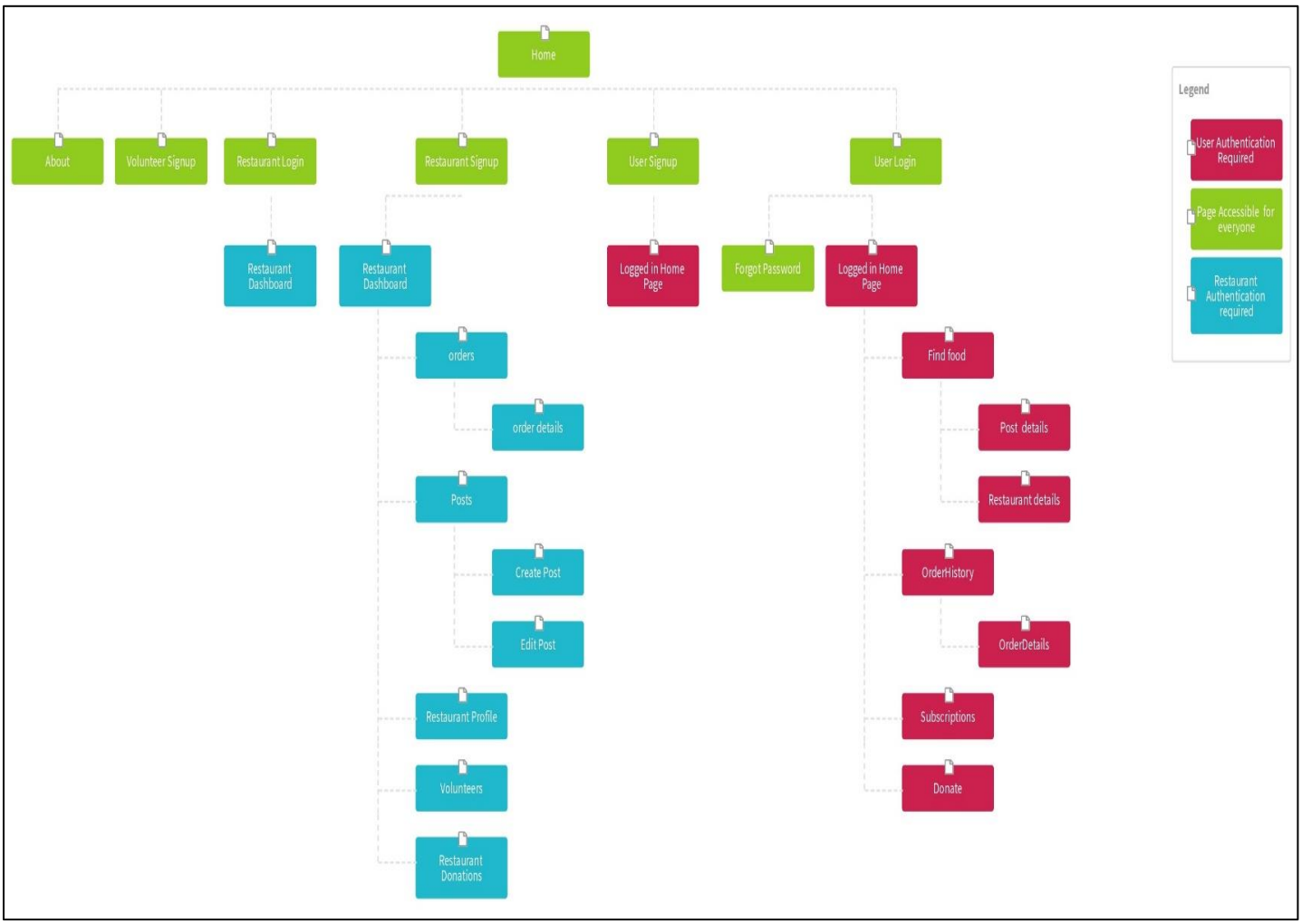

Figure 1: Information Architecture

3.2.2 Design and Layout

3.2.2.1 Proposed Wireframes

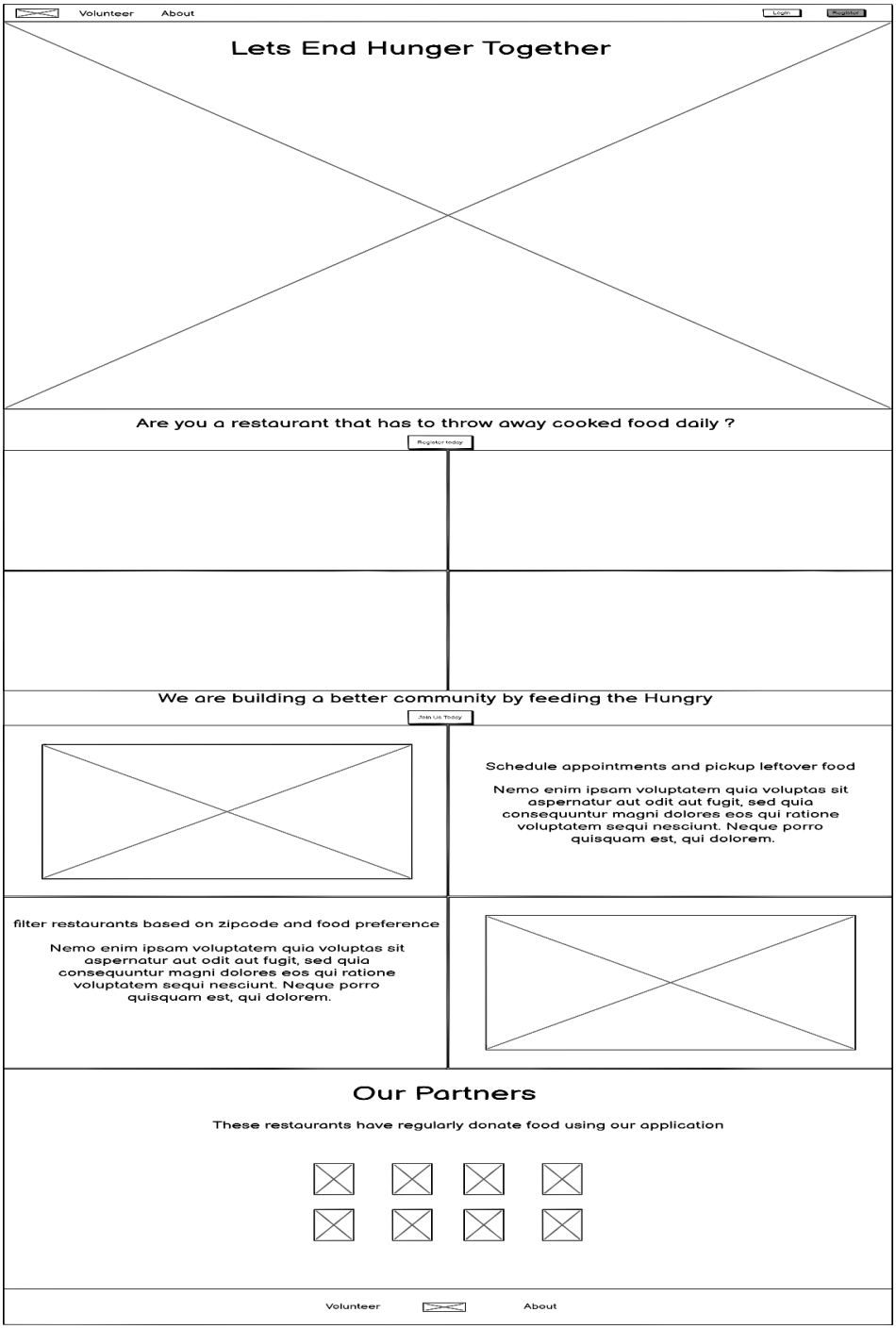

Figure 2: A wireframe for Home Page, Created using Balsamiq.cloud [4].

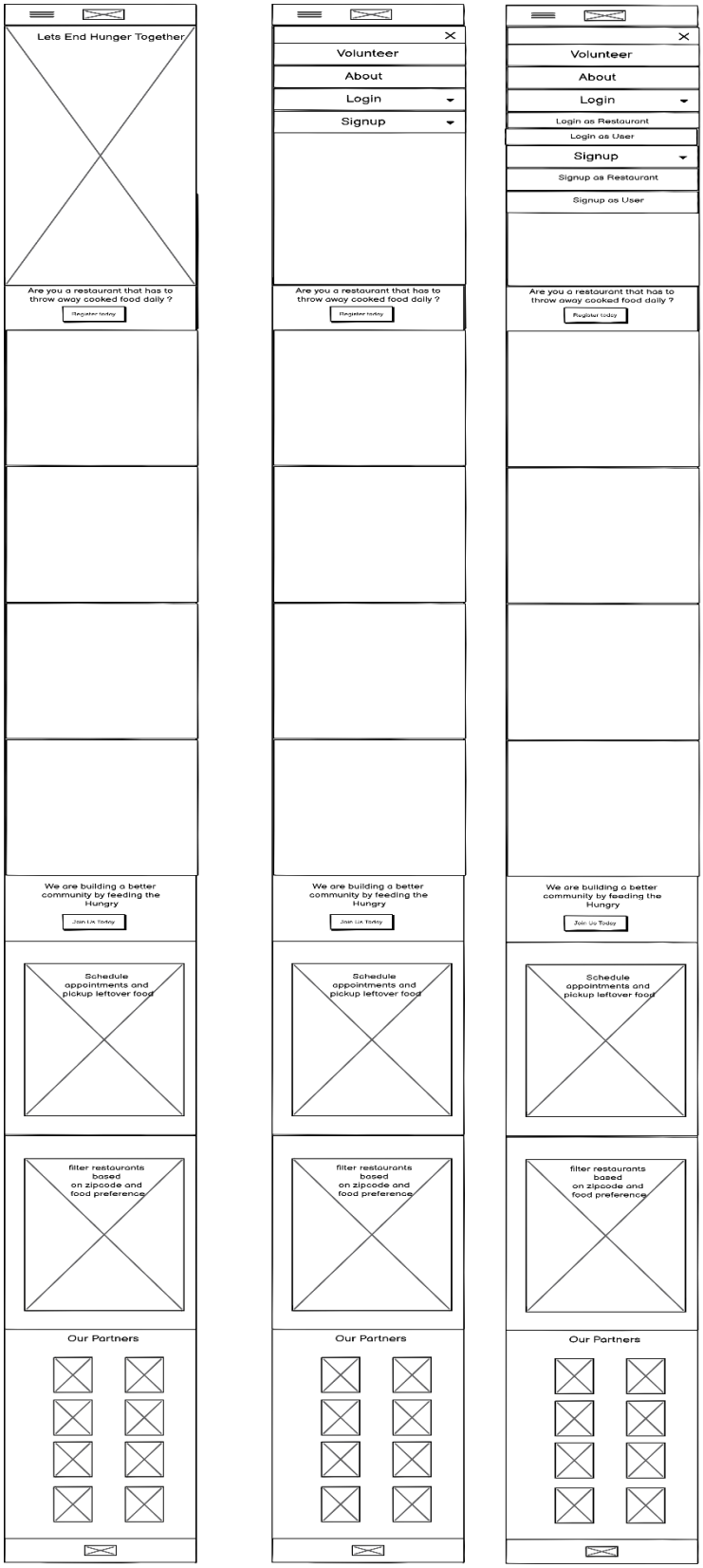

Figure 3: A wireframe for a Responsive mobile device, Created using Balsamiq.cloud [4].

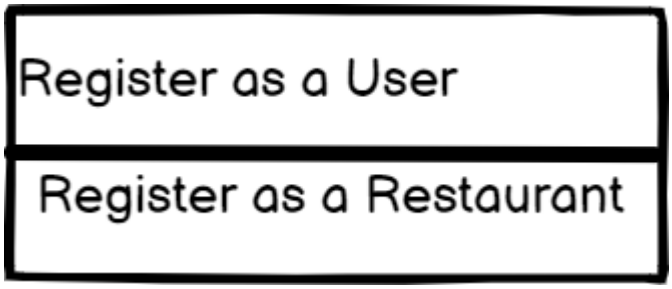

Figure 4: A wireframe for a user to choose which login do they want, Created using Balsamiq.cloud [4] .

Figure 5: A wireframe for User sign up, which a user lands on after clicking on "Register as a User", Created using Balsamiq.cloud [4].

Figure 6: A wireframe for Successful login after a user successfully signs up on LastServe, Created using Balsamiq.cloud [4].

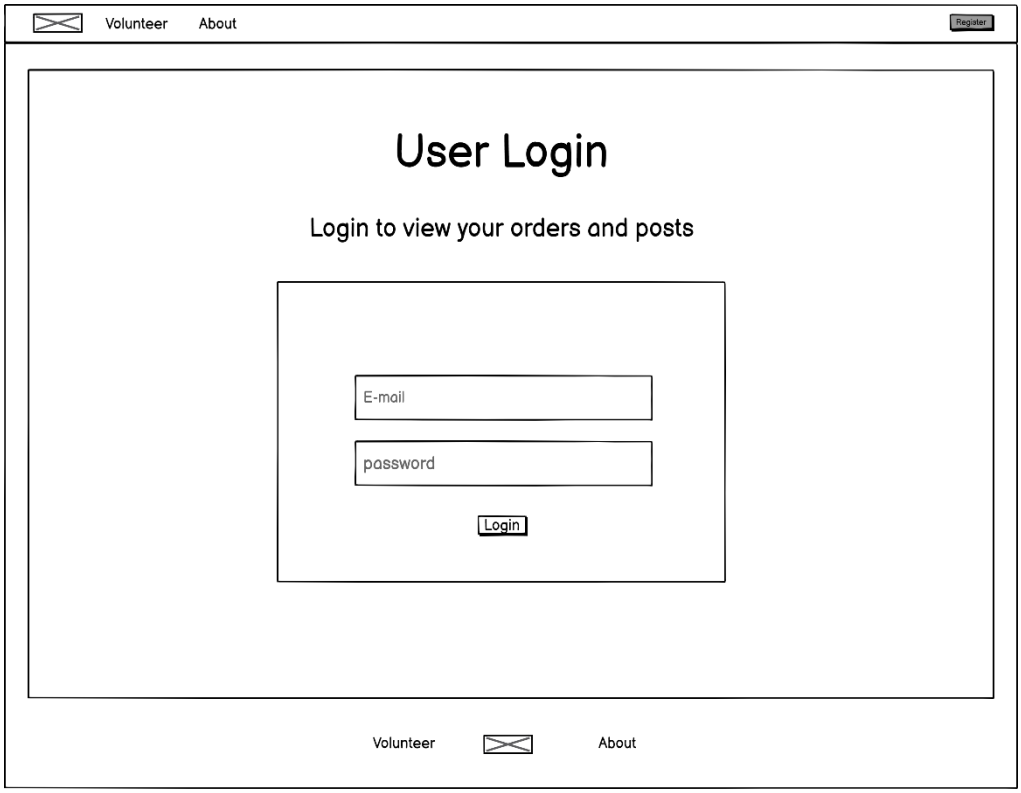

Figure 7: A wireframe for user to login in into LastServe after successful registration, Created using Balsamiq.cloud [4].

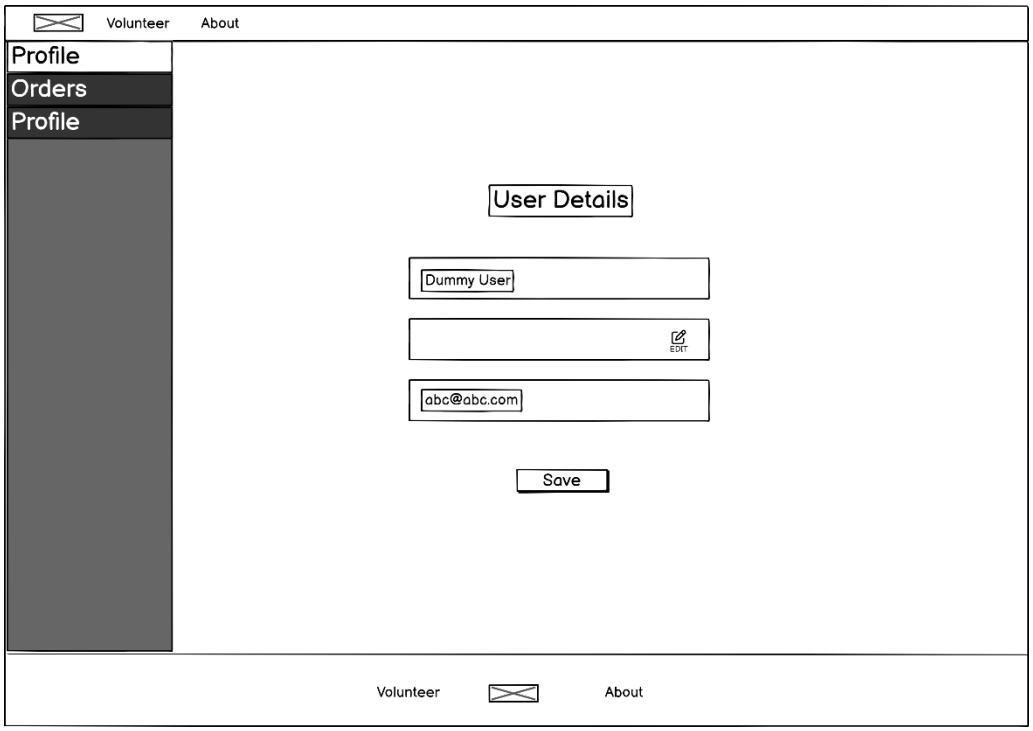

Figure 8: A wireframe for User details on the profile tab for user viewing, Created using Balsamiq.cloud [4].

Figure 9: A wireframe for Restaurant login, after the user chooses to login as a Restaurant owner, Created using Balsamiq [4].

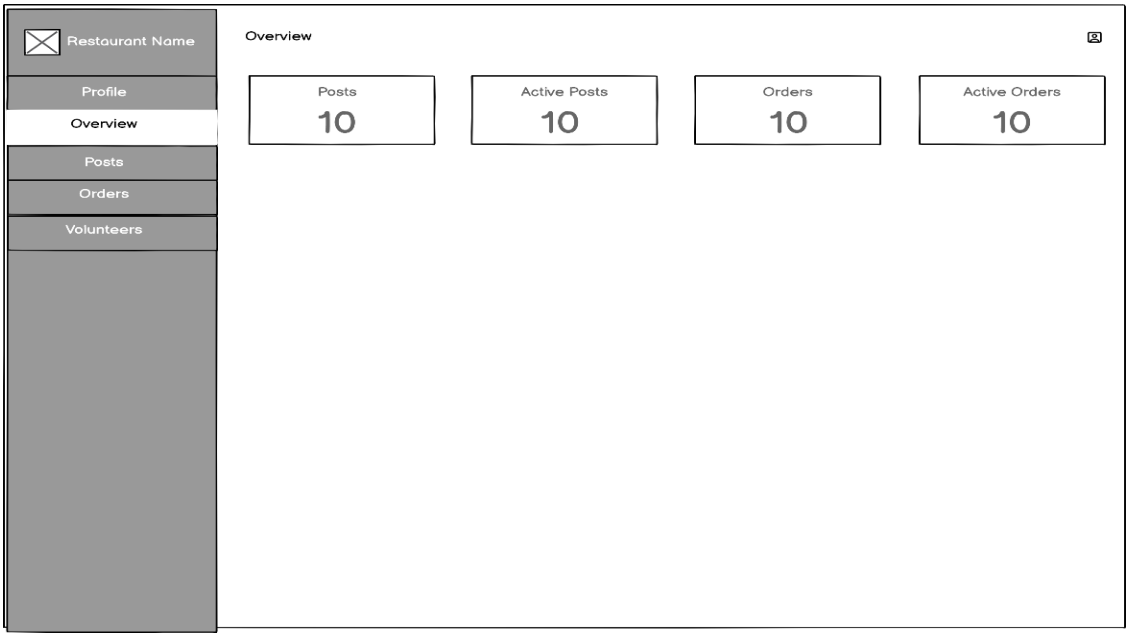

Figure 10: A wireframe for Restaurant Dashboard that provides an overview of all the details about active posts, orders and active orders, Created using Balsamiq.cloud [4].

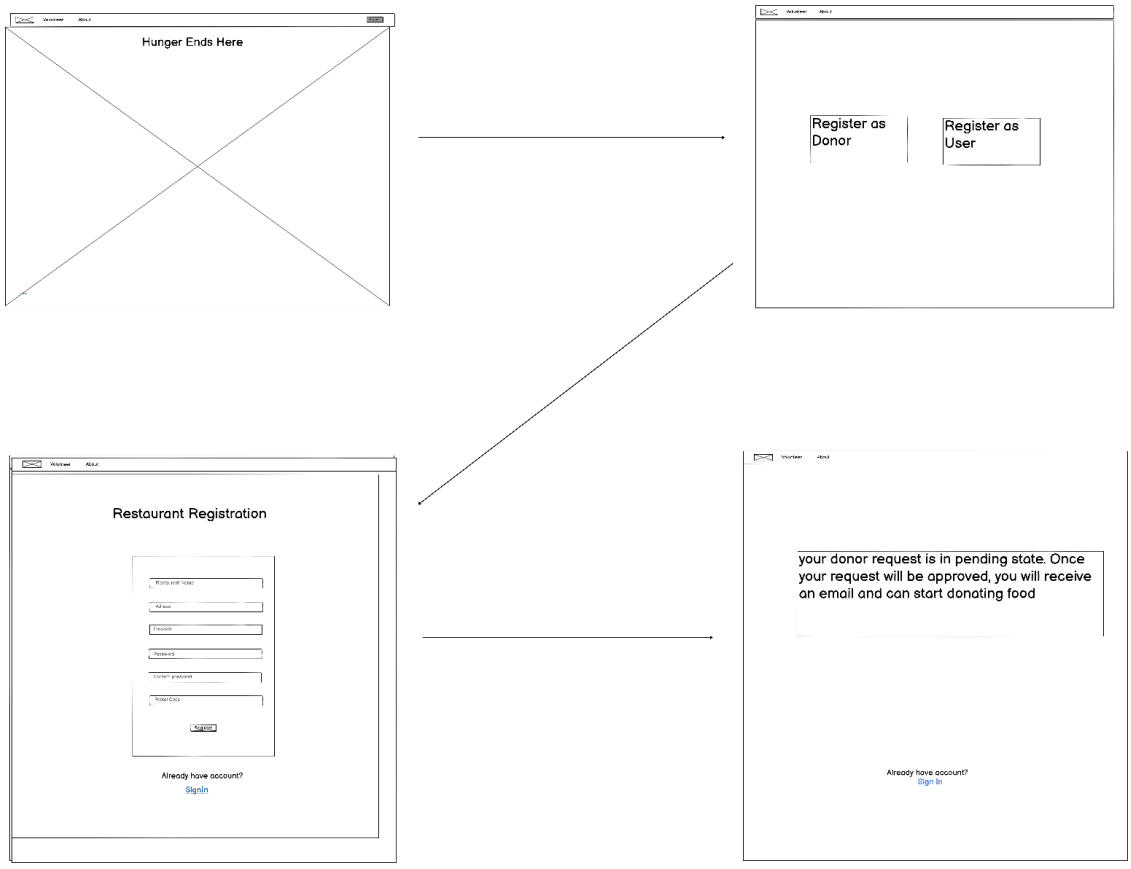

Figure 11: A wireframe for the entire registration process for a Restaurant, Created using Balsamiq.cloud [4].

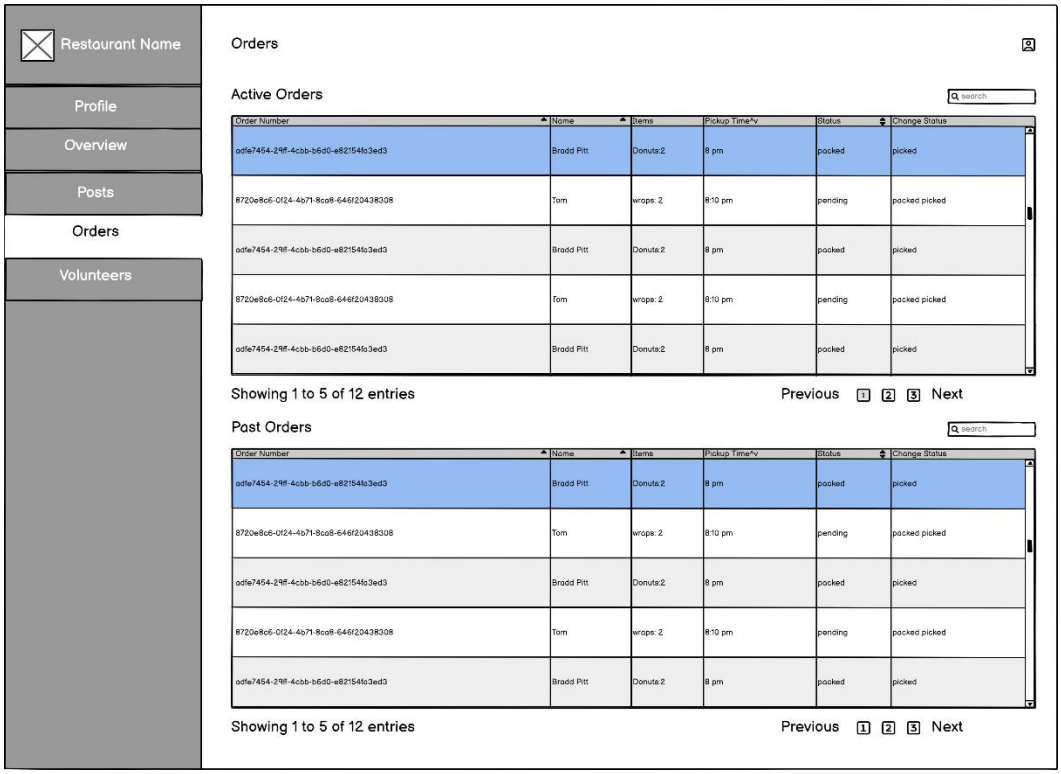

Figure 12: A wireframe for Active posts that can be viewed by the Restaurant owner after clicking on the post card, Created using Balsamiq.cloud [4].

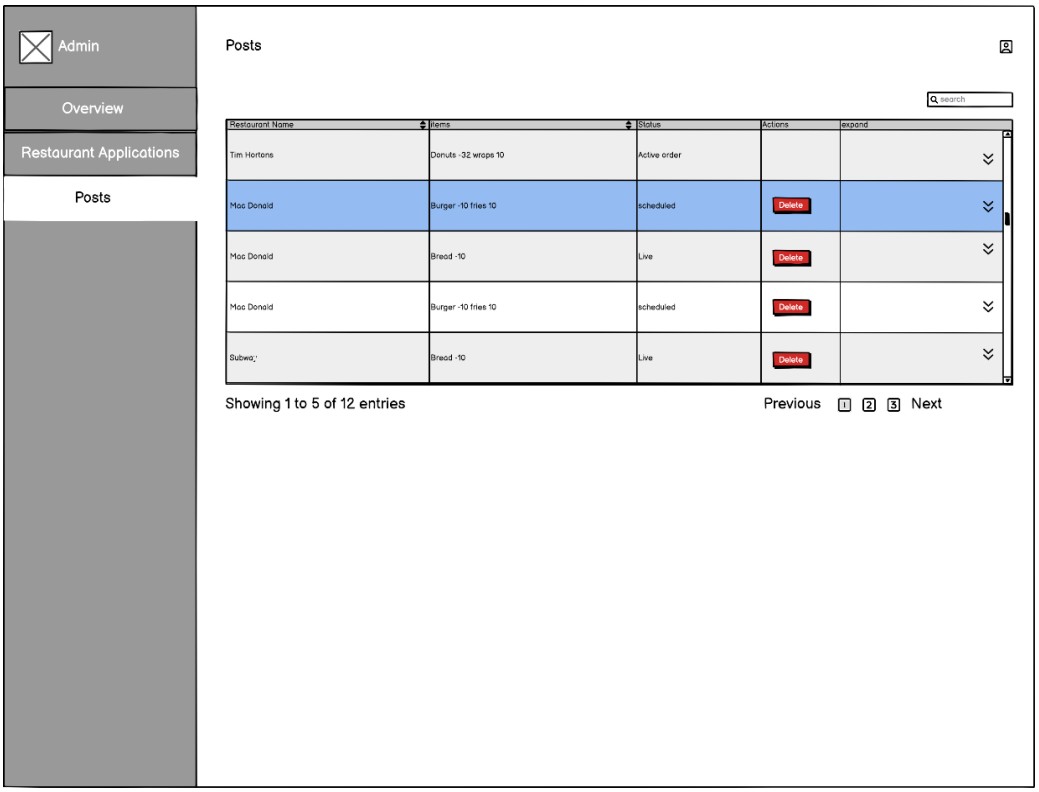

Figure 13: A wireframe of the admin viewing the posts on LastServe.ca, Created using Balsamiq.cloud [4].

Figure 14: A wireframe for subscription page, Created using Balsamiq.cloud [4].

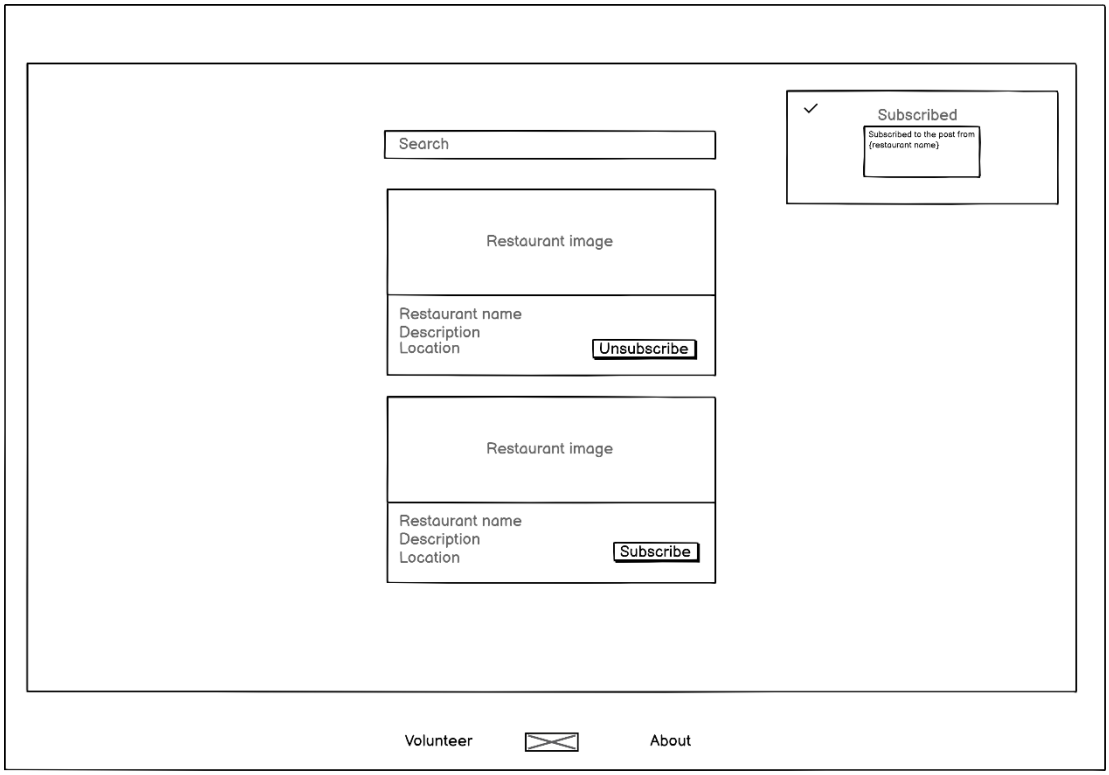

Figure 15: A wireframe for subscribed pages; a user can unsubscribe when required, Created using Balsamiq [4].

3.2.2.2 Website Design

The design we created was centered on enhancing user experience and making navigation simple. The layout is adaptable and takes a simple approach with a soft colour scheme. All form-filling procedures are clear and straightforward.

The design choices for several pages are listed below:

Home page: The fundamental idea of our website was intended to be easily communicated on the home page. The message is presented using large fonts, and the accompanying graphics help the reader understand our goal. It contains all the requisite links to point users in the direction of all important pages. A new user can establish trust by learning names of our partner restaurants. Our homepage is completely transparent to show what we do (see **Figure 22**).

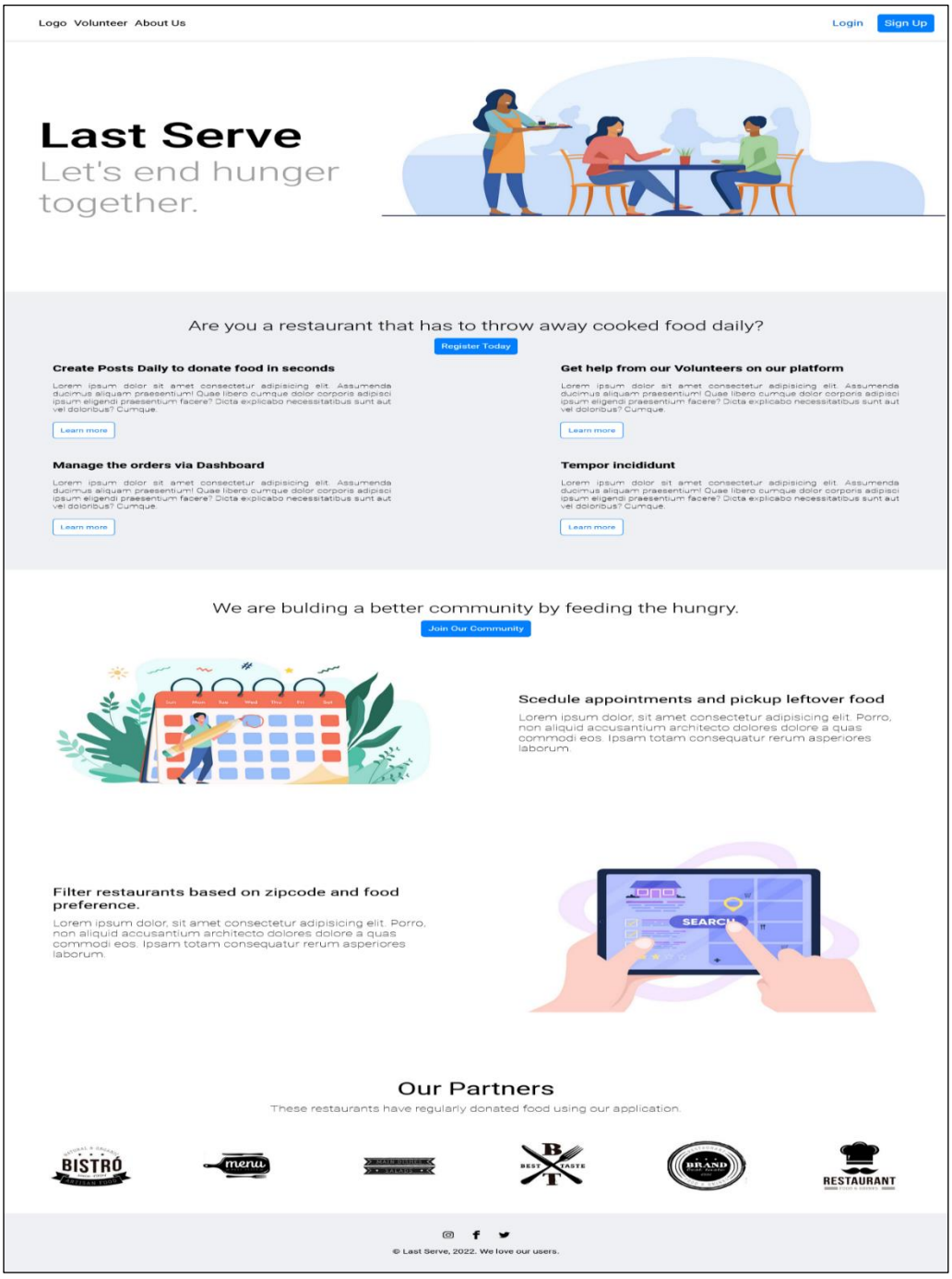

Figure 15: Screenshot for LastServe Home Page

User Registration: The user registration form is simple and simply requires the bare minimum of data. To prevent any problems in the database, it includes all the validations. A user can generate a unique password once and use it to log in each time

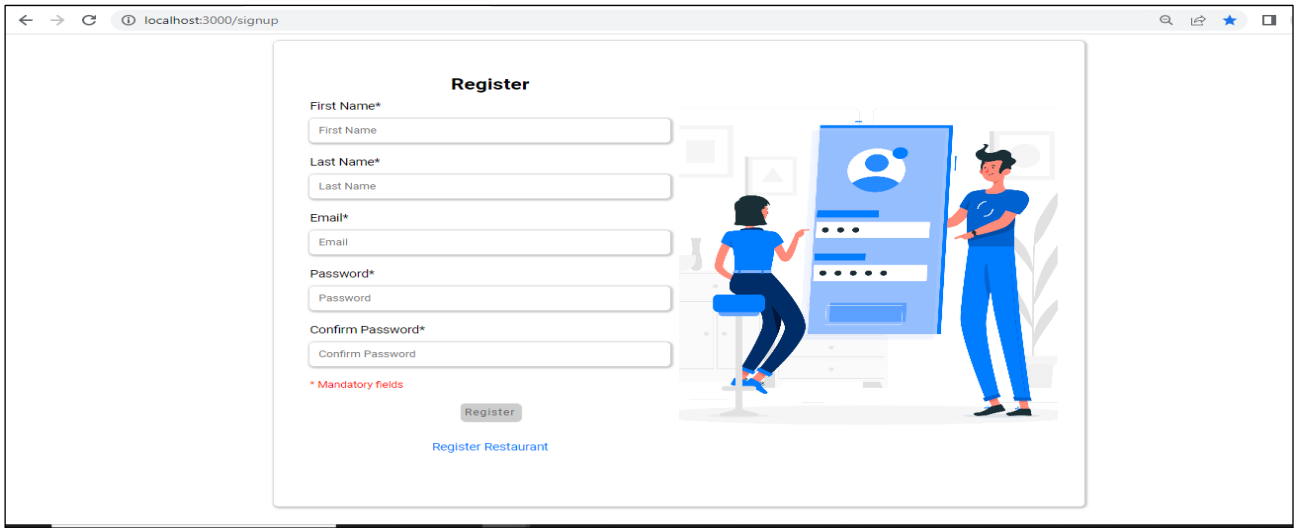

Figure 16: User registration

User Information

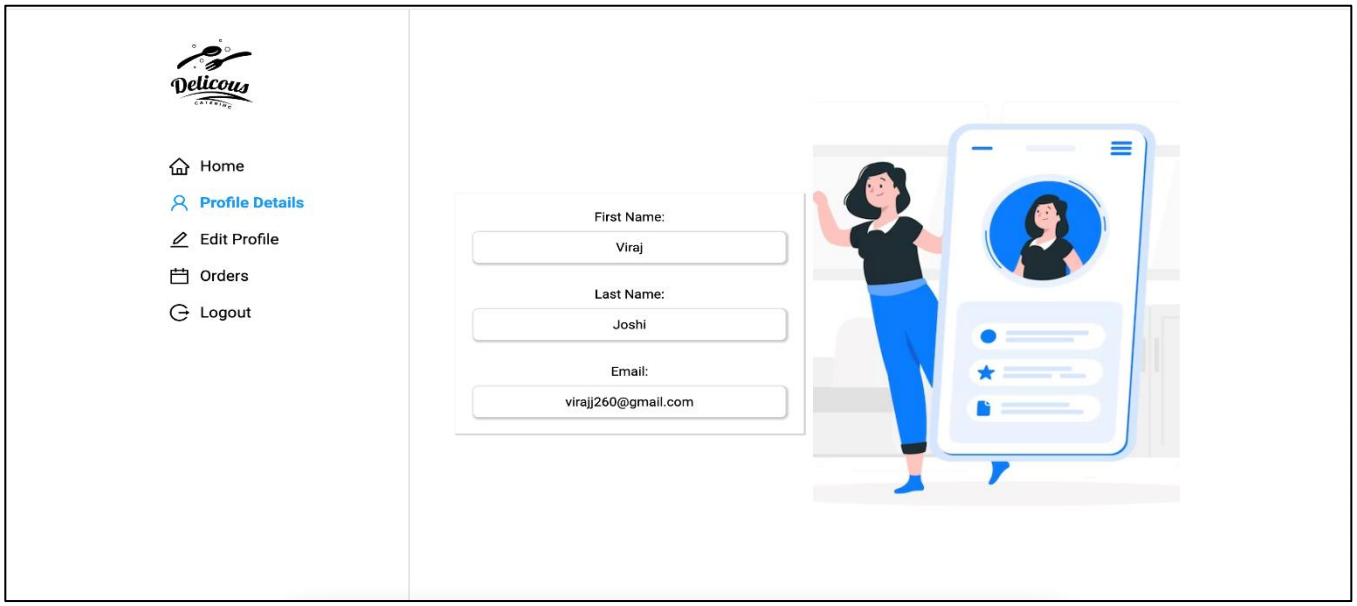

Figure 17: User Information

Password Reset:

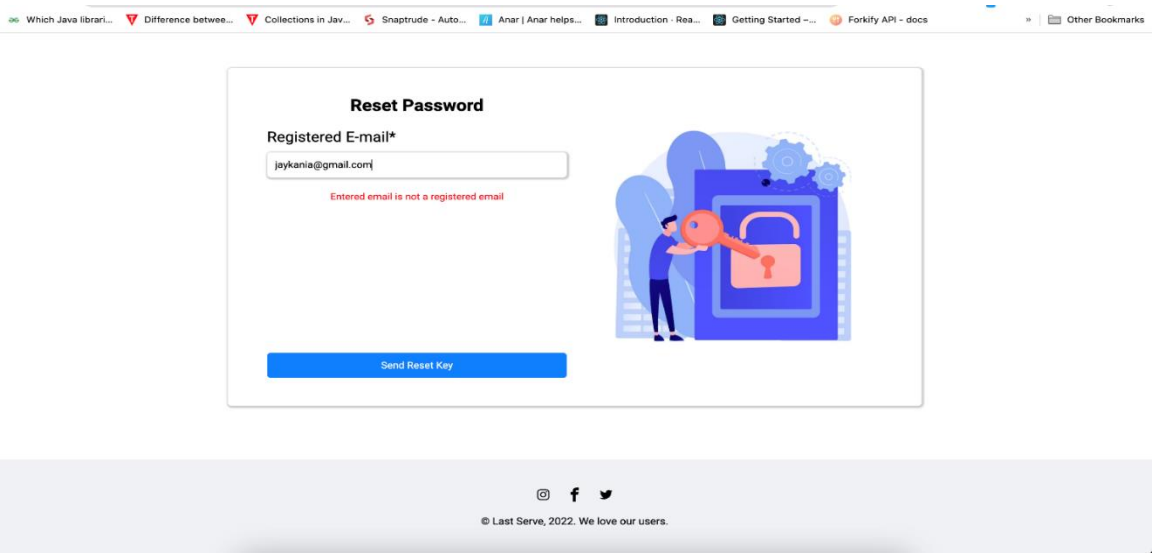

Figure 18: Password Reset

Restaurant Registration Page:

Restaurant Pending Approval Page: The restaurant will be sent to this page after successfully registering on our website. The user will see a notification about pending requests. The business

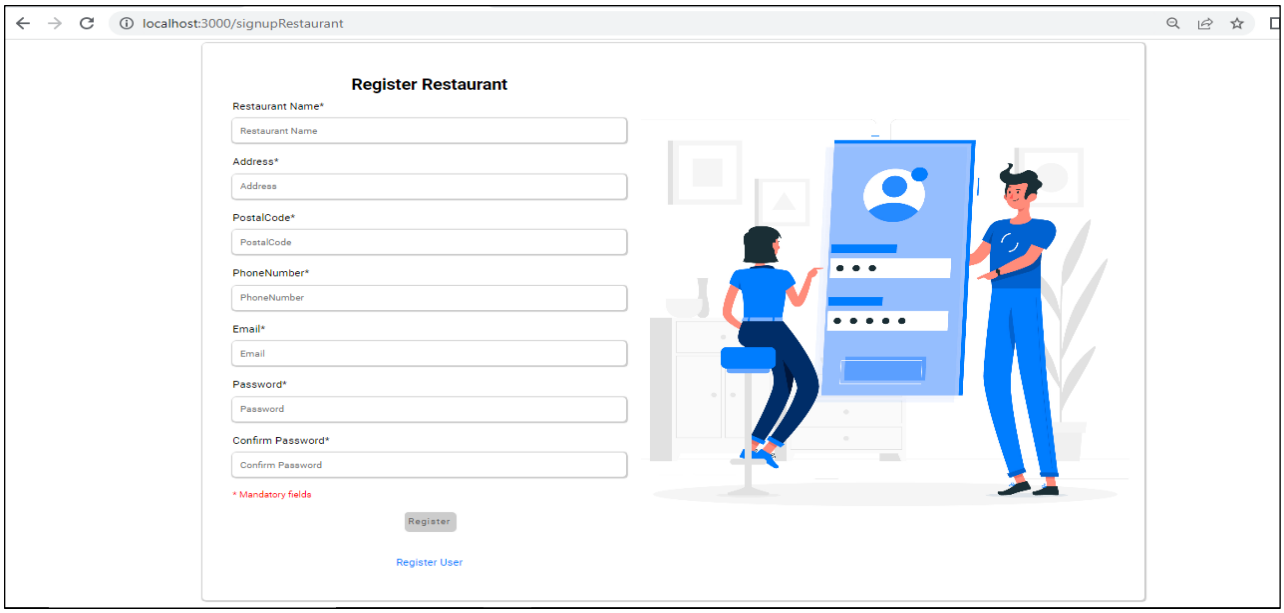

Figure 18: Screenshot for Restaurant Registration Page

will receive an email once their request has been granted, at which point they can start donating food through our website

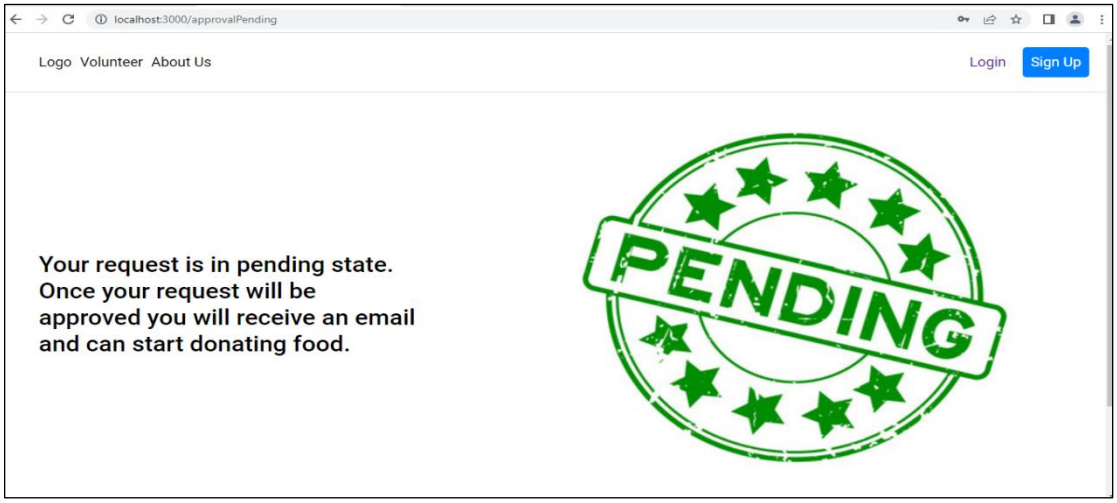

Figure 19: Screenshot for pending approval page

Restaurant Login: The restaurant manager will be able to log in to our website once their restaurant has been approved and carry out specific duties dependent on permission. He will see the above form after clicking the login link. After choosing the "Restaurant Login" radio button and entering his login information, he will be taken to the restaurant dashboard page once his login has been successful.

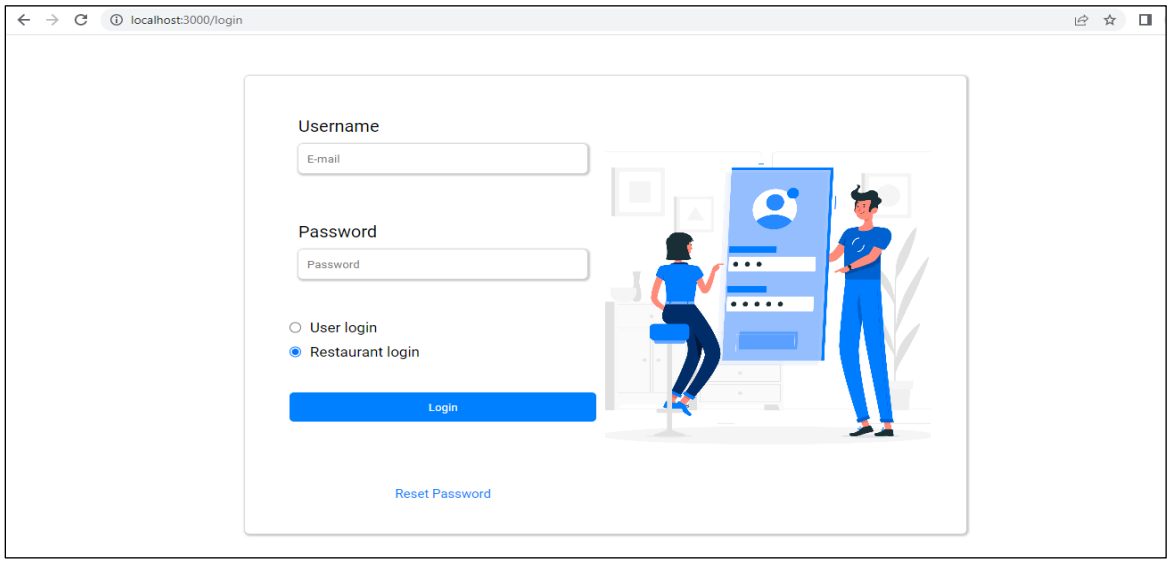

Figure 13: Screenshot for Restaurant Login

Restaurant View/Edit Profile: The manager can log in to the system and view and modify the restaurant's details once it has been approved. After logging in, the restaurant manager will be taken to the restaurant dashboard, where he can access the "Profile" option on the left. When he clicks on it, all of the most recent information about the restaurant, including the logo, is presented. He can adjust this information to suit his needs, and the system will update to reflect the changes successful.

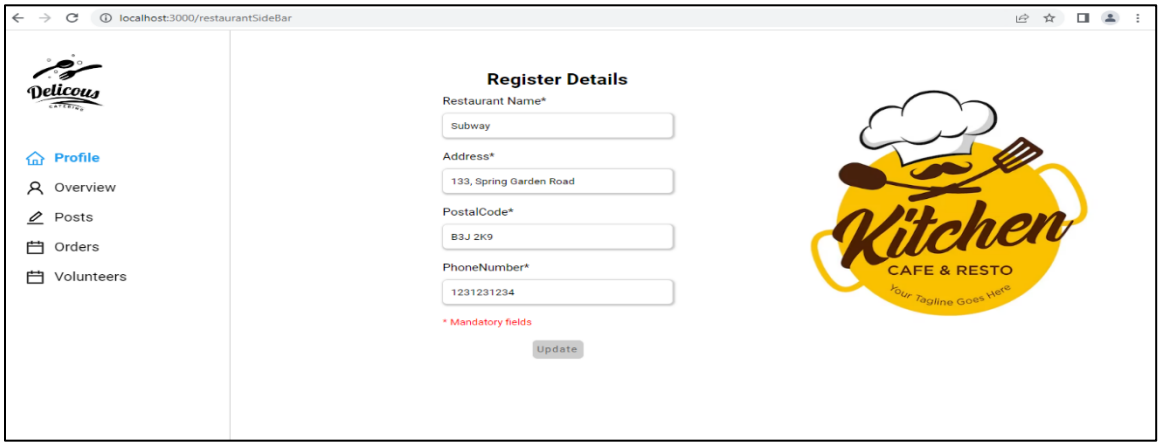

Figure 14: Screenshot for View/Edit Restaurant Profile

Restaurant Dashboard overview: As soon as the restaurant manager logs in, they are taken to the Overview page, where they can see a quick summary of all the restaurant's statistics, including the total number of orders, the number of active orders, the number of posts the restaurant has made, and the number of active posts. To make the dashboard's interface easy to use and understand, the color scheme has been kept minimalistic successful (see **Figure 30**).

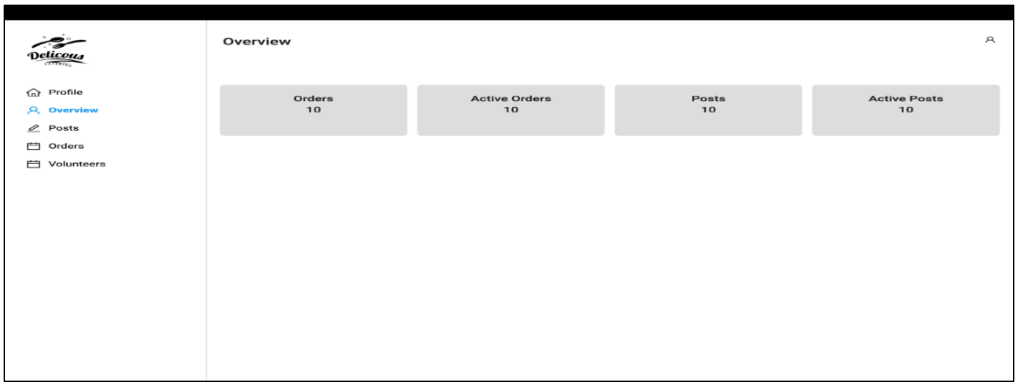

Figure 15: Screenshot for Restaurant Dashboard Overview Page

Volunteer Display: When the restaurant management needs assistance from a volunteer, they can go to the volunteers page to get a list of everyone in the system who is willing to help out and their availability. To help the restaurant identify a volunteer more quickly, the table can be arranged according to availability. The volunteer's phone number is also provided for the management to call. The tables' pagination will assist in breaking up the material into manageable portions successful.

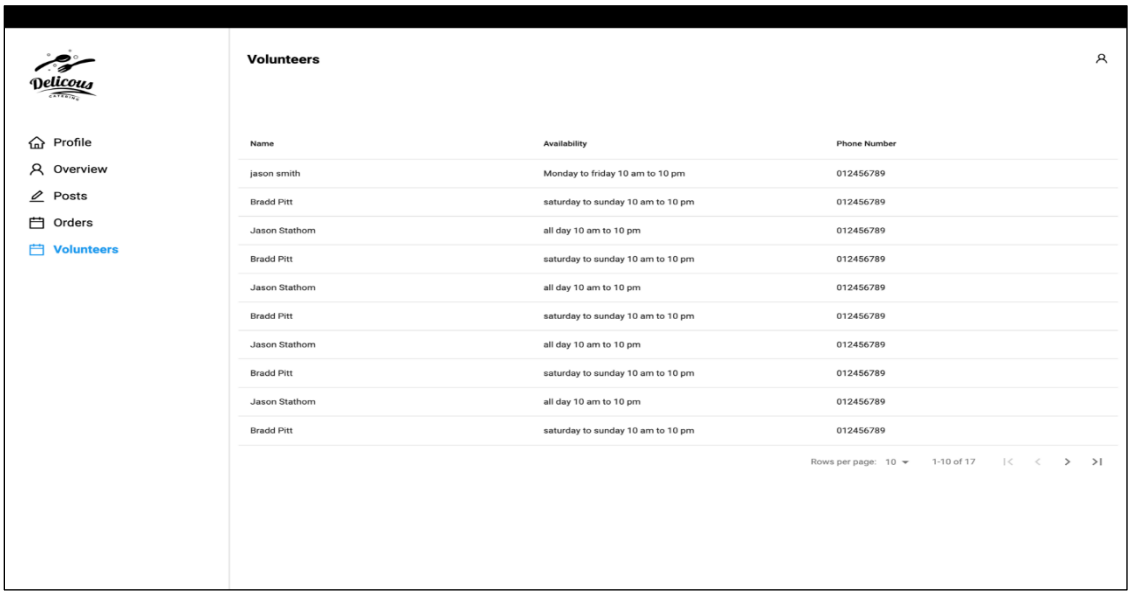

Figure 16: Screenshot for Volunteer viewing Page on Admin dashboard

Restaurant orders page: When the restaurant manager wishes to manage the orders on the system they can navigate to the orders screen. The page is divided into 2 table, active orders, and past orders, to organize the data into logical groups and make information retrieval easier for the restaurant manager. Search bars are provided on top of the table to easily search for an order. For active order buttons are provided to change the status of the order to packed and picked to manage the orders successful.

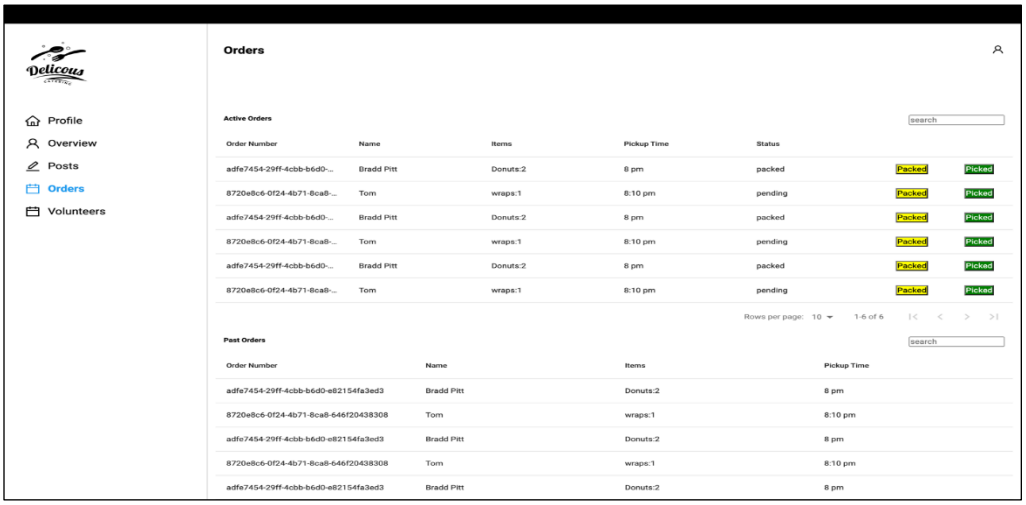

Figure 17: Screenshot for Volunteer display Page

Volunteer Registration: All of the fields needed for a volunteer to register are visible on the volunteer page. They must complete all the necessary fields so that a restaurant can find the best match. The form is kept straightforward and doesn't ask for a lot of personal data successful.

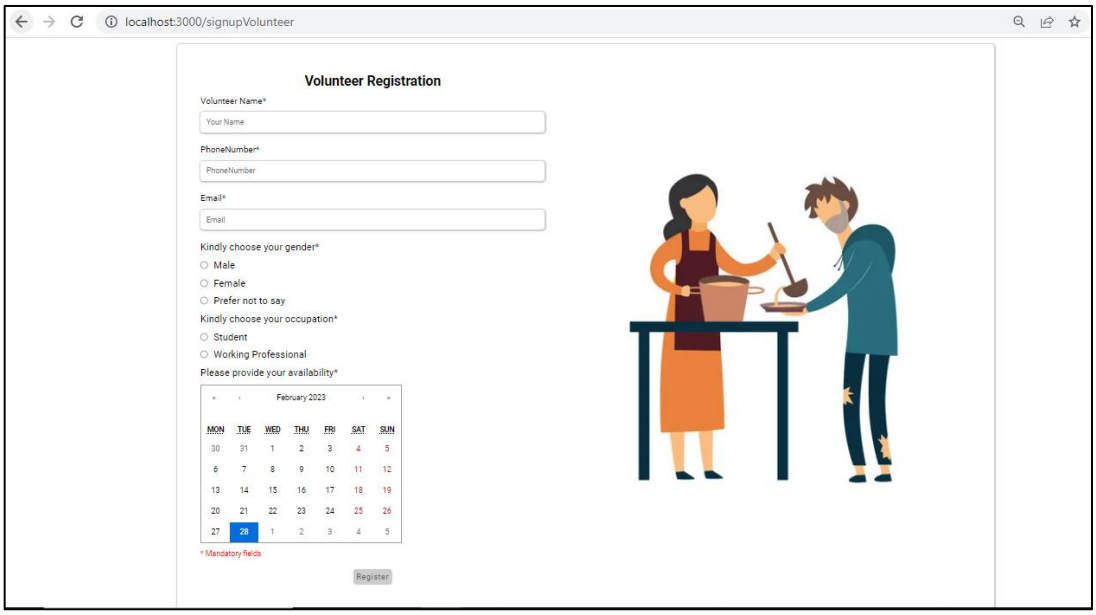

Figure 18: Screenshot for Volunteer registration

Appointment booking: Users will see a list of posts from restaurants that are now offering free food on the home page after logging into their accounts. Depending on the food category and whether they have a subscription to a certain restaurant, users can filter the establishments successful .

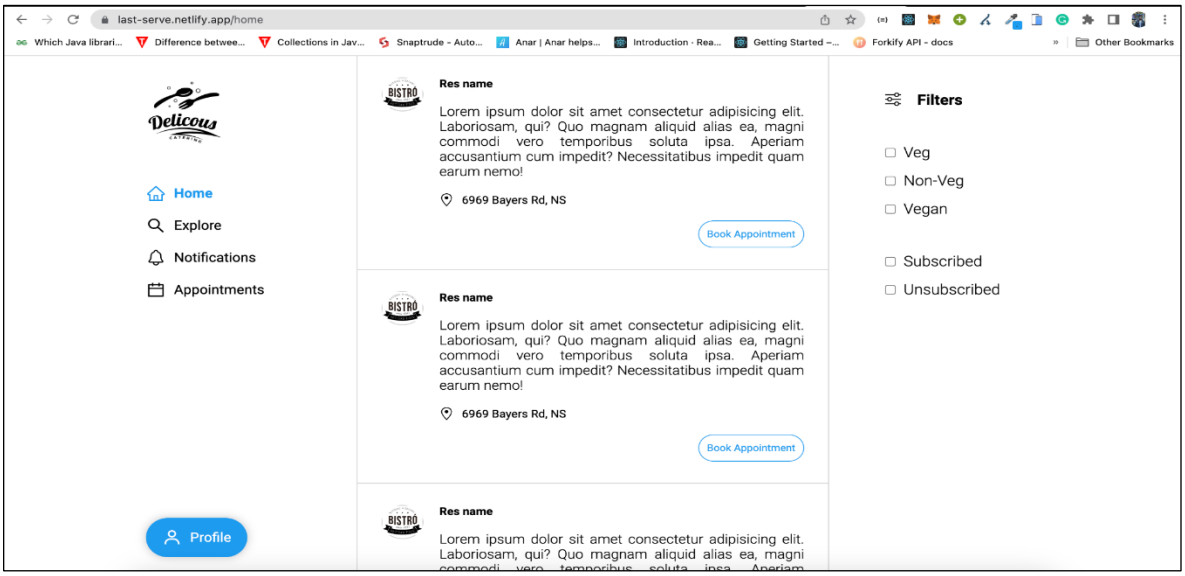

Figure 19: Screenshot for appointment booking Page

When users select the "Book Appointment" button on the main page, the above page will be displayed. On this page, a form asking for the user's name, email address, desired time slot, and whether they want a copy of their response via email will be present.

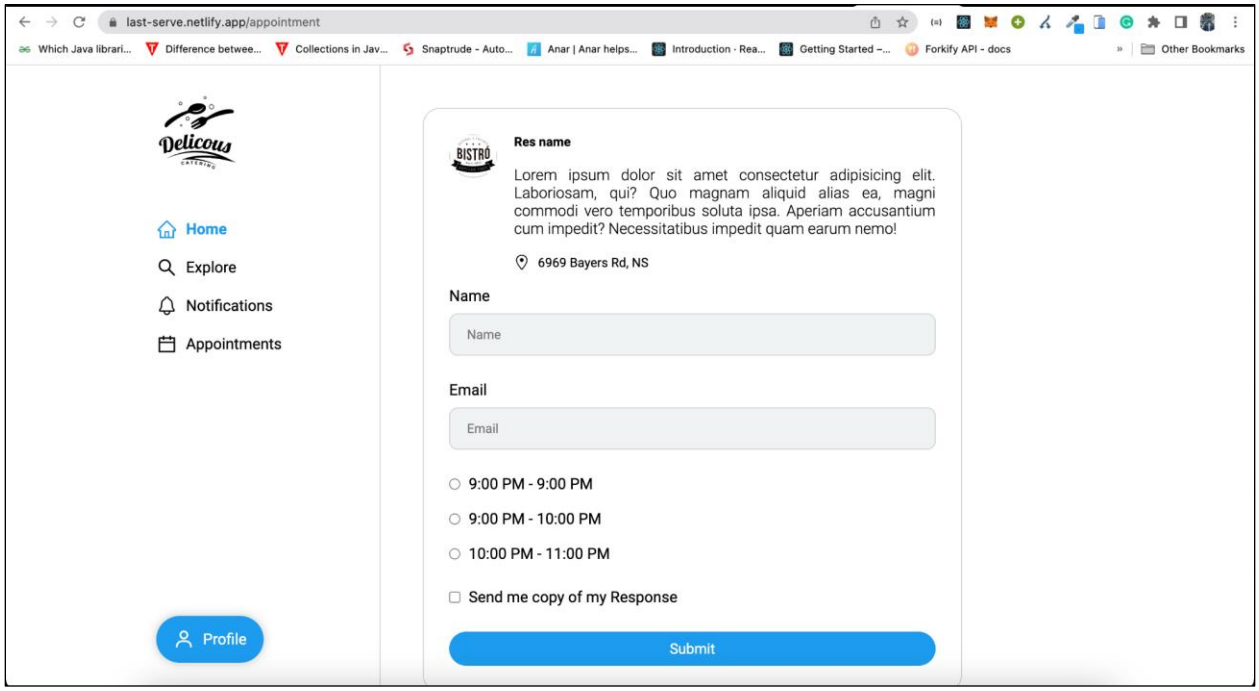

Figure 20: Screenshot for appointment booking screen

On the explore page users will be able to see a list of restaurants. Users can click on the subscribe button to get notifications whenever the restaurant posts something.

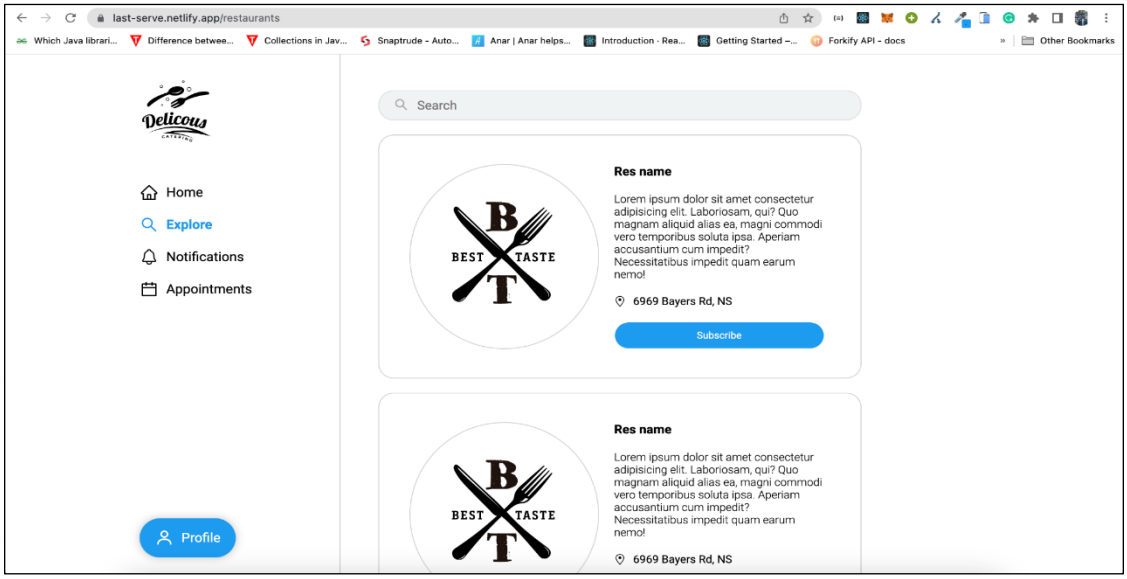

Figure 21: Screenshot for Subscription page

Donation Management

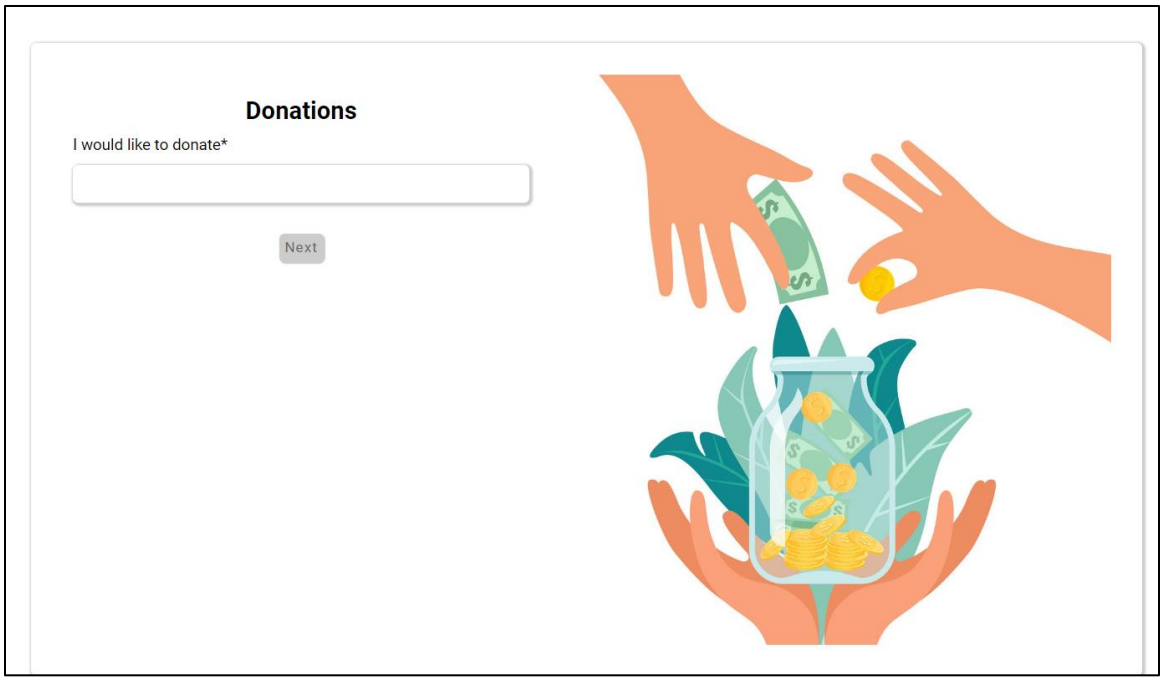

Figure 37: Donation Management

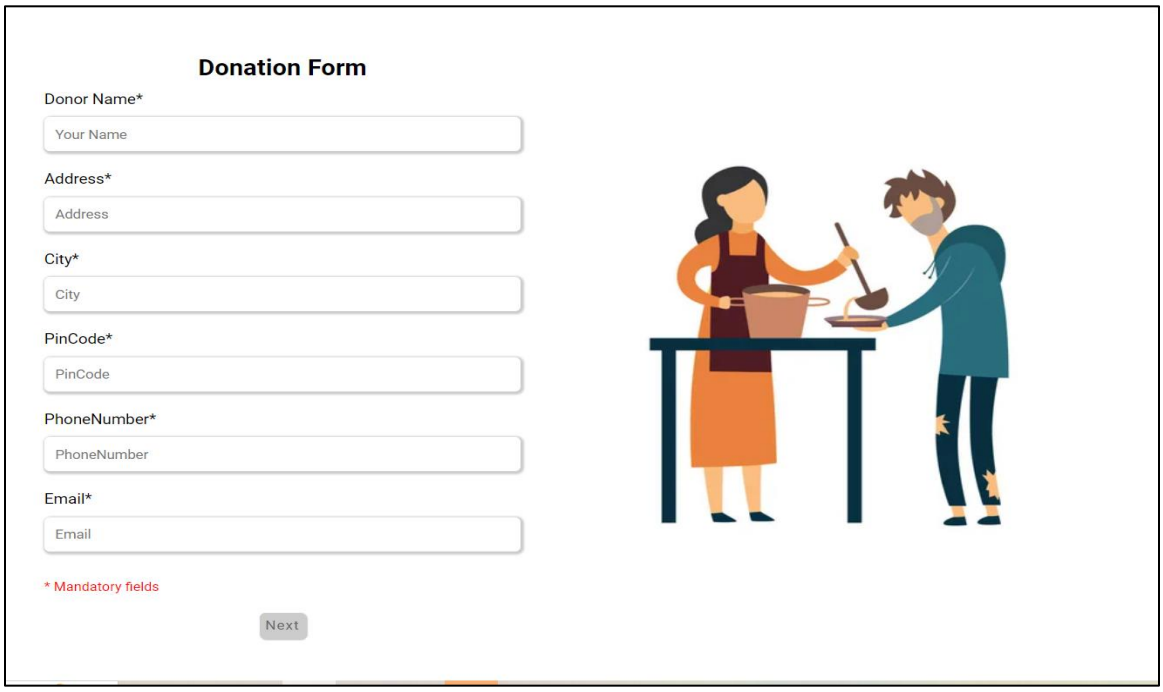

Figure 38: Donation Form

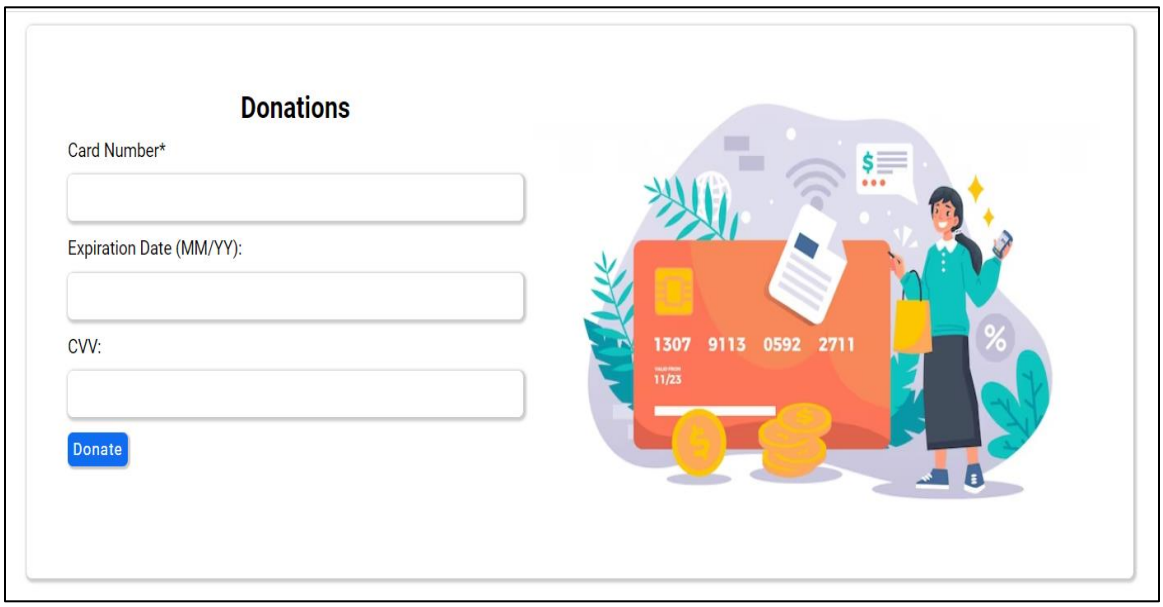

Figure 39: Card Details

4. APPLICATION WORKFLOW

The application boundaries are divided into three parts

1. Frontend Framework: React
   multiple libraries are used to support and speed up the development of frontend like
   a. **react-Data tables**: To provide pagination for tables.
   b. **react-toastify**: To add disappearing notifications on a page.
   c. **react-bootstrap**: To make the pages responsive.
   d. **Axios**: To interact with backend rest services.

2. Backend Language: NodeJs
   Pattern:MVC
   Multiple backend libraries are used like
   a. **Bcrypt3**: For hashing the password.
   b. **Mongo-client**:for establishingconnection and querying the database
   c. **Express**: To setup routing of the backend application
   d. **UUID**: To create unquie ids
   e. **NodeEmailer**: To send emails to users

3. Database:MongoDb

Multiple document collections are created to store the key value pair

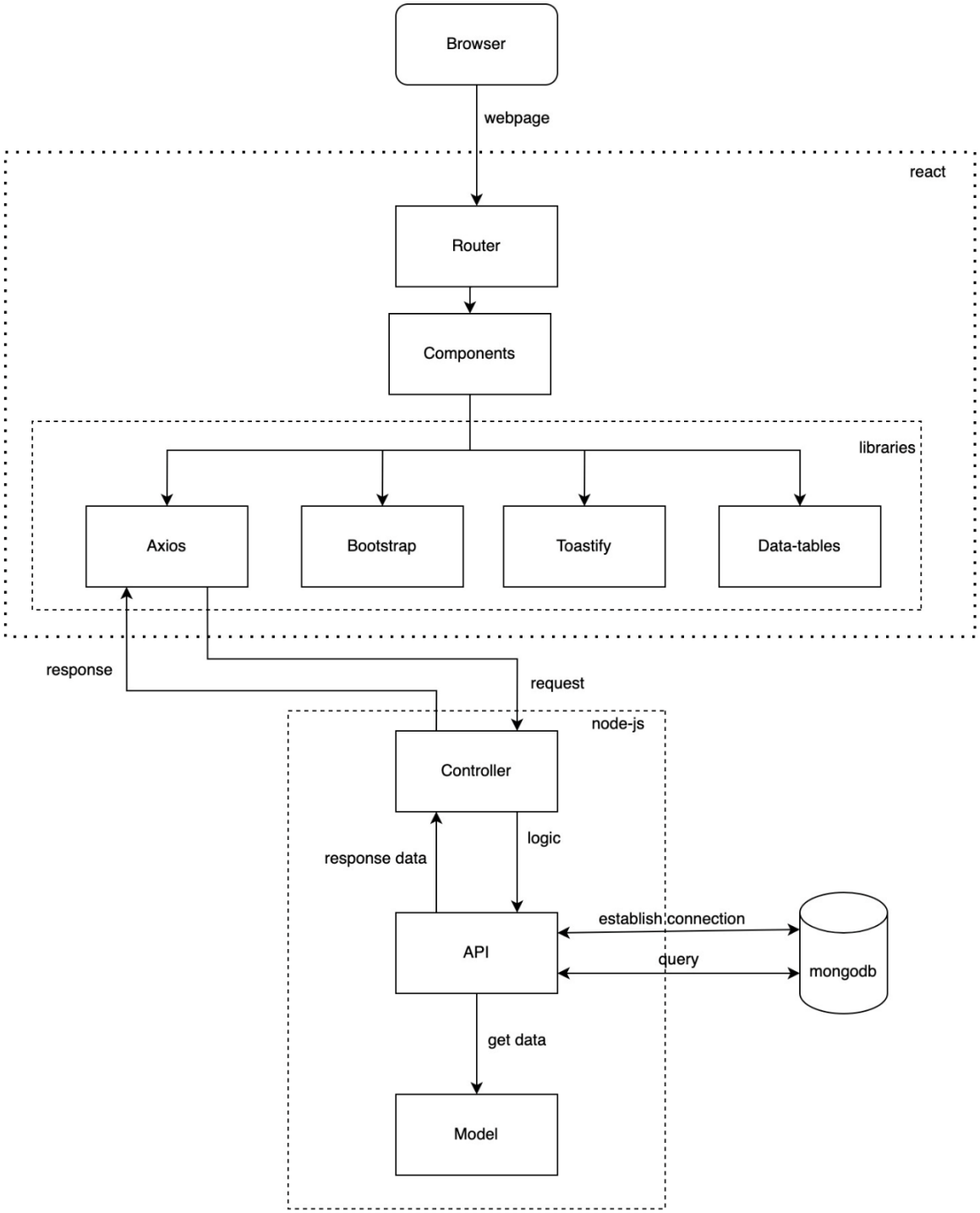

Figure 40: Architecture of the Application created using draw.io[]

4.1 Interaction Design

The frontend uses the following:

1. React components to render the DOM and use multiple react libraries
2. State to manage the data flowing between different components
3. UseEffect to trigger process like making call to frontend on page load or updated
4. JavaScript functions to trigger onclick and onupdate events

Working features and respective tasks on the Application

1. **User Profile Management**
   a. User Registration
   b. View User Profile
   c. Update User Profile
   d. Existing User Login
   e. Reset password

2. **Restaurant Profile Management**
   a. Restaurant Registration
   b. View Restaurant Profile
   c. Update Restaurant Profile
   d. Restaurant Login

3. **Food Donation Post Management**
   a. Create a new post
   b. View past Posts
   c. Edit active Posts

4. **Admin Content Moderation**
   a. Admin Dashboard visualization.
   b. Delete posts.
   c. Approve restaurants.

5. **Restaurant order management**
   a. View orders by user, their timeslots, and items.
   b. Change order status from pending to packed and picked.
   c. Get available volunteer information based on availability.

6. **User Order management**
   a. View all active donors with timings and items offered.
   b. Filter through all active donors based on location, restaurant name and type of food.
   c. Appointment slot booking for pickup.
   d. Email of confirmed order once an appointment is booked.

7. **Volunteer Management**
    a. Apply for volunteering with the application

8. **Subscription management**
    a. Users can subscribe to posts from a restaurant
    b. Users can unsubscribe to a restaurant.

9. **Donation Management**
    a. Users can donate money to restaurant

10. **User order history Management**
    a. Users can look through their previous order history
    b. Users can check status of current order
    c. Cancel active orders

Feature: Restaurant order management.
Task: View orders by user, their timeslots, and items.

User Scenario- View orders by user, their timeslots, and items: The information regarding the extra food offered at the LastServe has been posted by the restaurant manager. To pre-pack the orders and allocate resources for order collection, he wants to check the users and their appointment time slots to pick up the food, goods, and quantity they ordered on the application. He looks at the dashboard to see all orders that have been placed and their time slots.

1. The restaurant manager has successfully logged in.
2. Restaurant managers see the Restaurant dashboard with various functions
3. The restaurant manager clicks on the orders.
4. Two tables are displayed One for the active orders and completed past orders.
    4.1. There are no active orders user is displayed “there are no active orders”.
5. There are no past orders for the restaurant. The active orders table has the information on the order number, the Name of the user items ordered appointment time, the status of the order, and actions to be performed on the order.
    5.1. Order with status pending is the order that is just placed, in the actions two buttons are shown as packed, and picked. Order with status packed is the order that is packed but not picked up, in the actions one button is shown as picked.
    5.2. The restaurant manager wishes to see other orders than the ones displayed in the active orders table.
6. The restaurant manager clicks on the next page tab at the bottom of the table.
7. The restaurant manager wants to see the orders sorted based on appointment time, he clicks on the appointment time header.
8. The system sorts the orders and displays the sorted order.
9. The system displays past in the table below active order with the following columns order number, name, and items.[1]

Click stream diagram

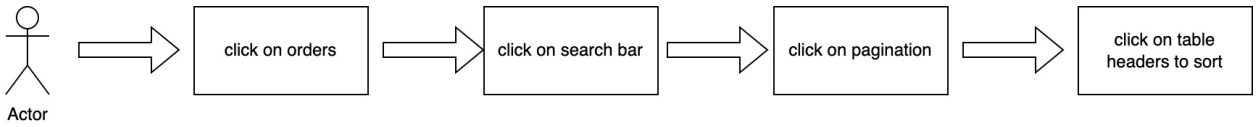

Figure 41: click stream diagram for viewing restaurant orders prepared using draw.io [5]

Task flow diagram

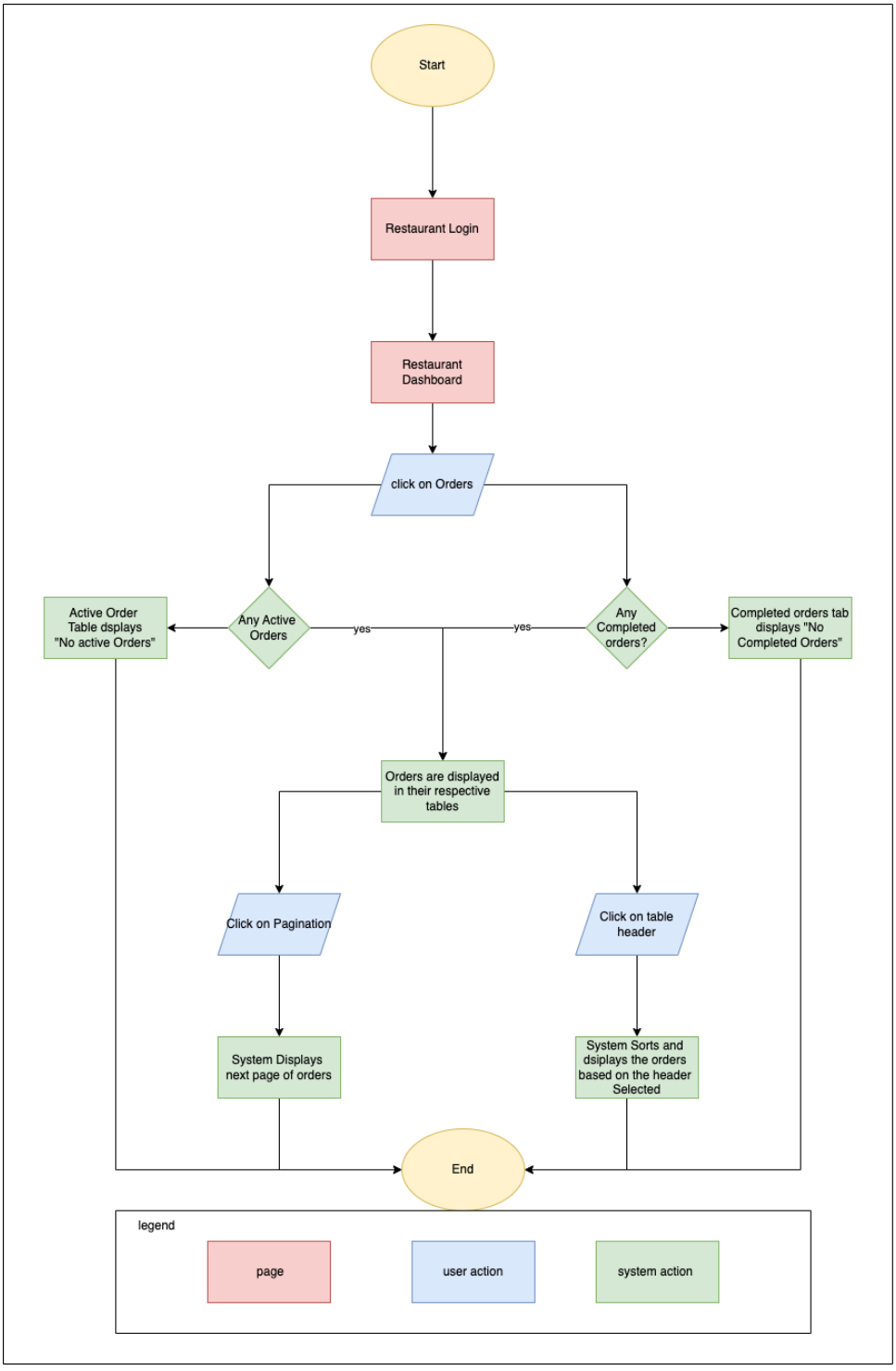

Figure 42: Taskflow diagram for viewing orders [5]

Task: Change order status

User Case- Change order status from pending to packed and picked: The restaurant can see every order that customers have placed. After packing one order, he wants to modify the order status so that other workers working on it are aware of how to pack additional orders. Additionally, he wants to mark an order as picked up after the user picks it up so that the order can be managed more easily. When it's done, he clicks the buttons for packaged and picked up.

1. The restaurant manager has logged in using his credentials.
2. The restaurant manager clicks on the orders.
3. The system displays all the active and past orders on tables.
   3.1. The system displays no active orders if there are no active orders for the restaurant.
4. The restaurant manager clicks on the packed button once an order is packed.
5. The system changes the state of the order to pack.
6. The system displays a message saying the order status is changed successfully.
   6.1. If there is an error while changing the state of the order "error something went wrong, please try again is displayed to the user".
7. The restaurant manager clicks on picked up once the order is picked up.
8. The system changes the state of the order.
9. The system displays saying "Order status changed successfully", System removes the order from active orders and adds it to the past orders table.
10. If there is an error while changing the state of the order user is displayed the error message "Something went wrong, please try again".

Click stream diagram

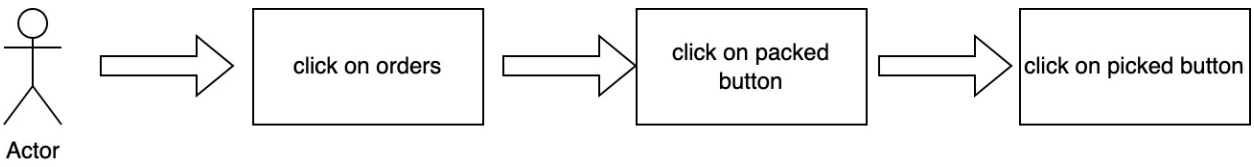

Figure43:clickstream diagram for order status change prepared using draw.io [5]

Task flow diagram:

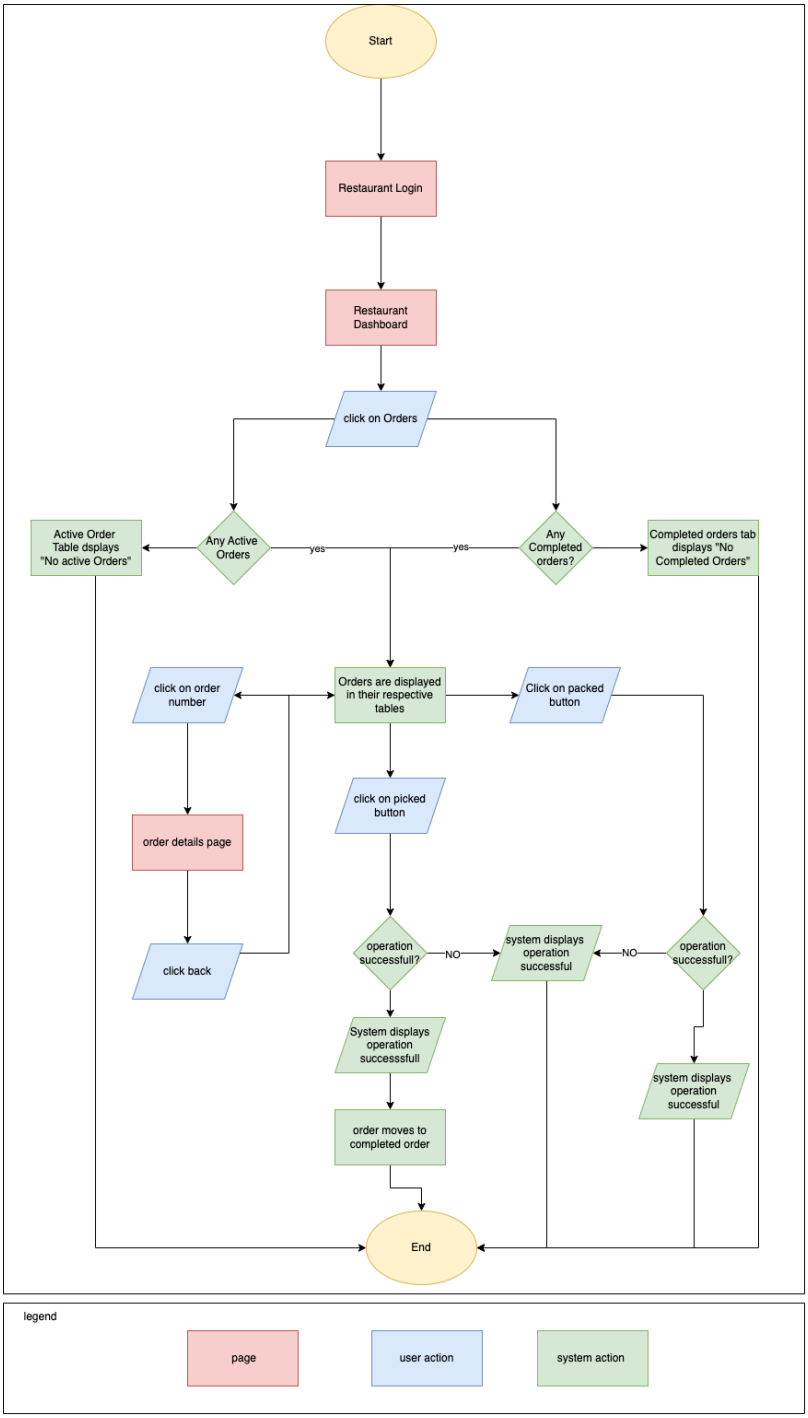

Figure 44: task flow diagram for order status change [4]

Task: View Volunteer information

Click stream diagram

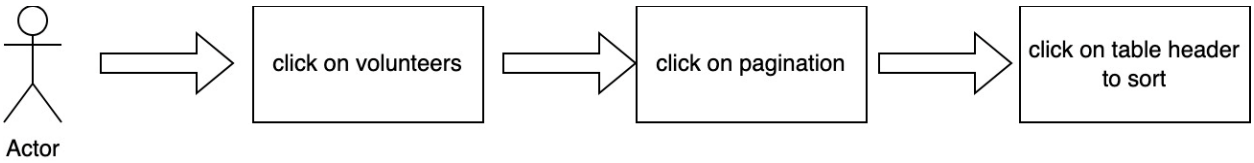

Figure 45:click stream diagram for viewing volunteers prepared using draw.io [5]

Task flow diagram

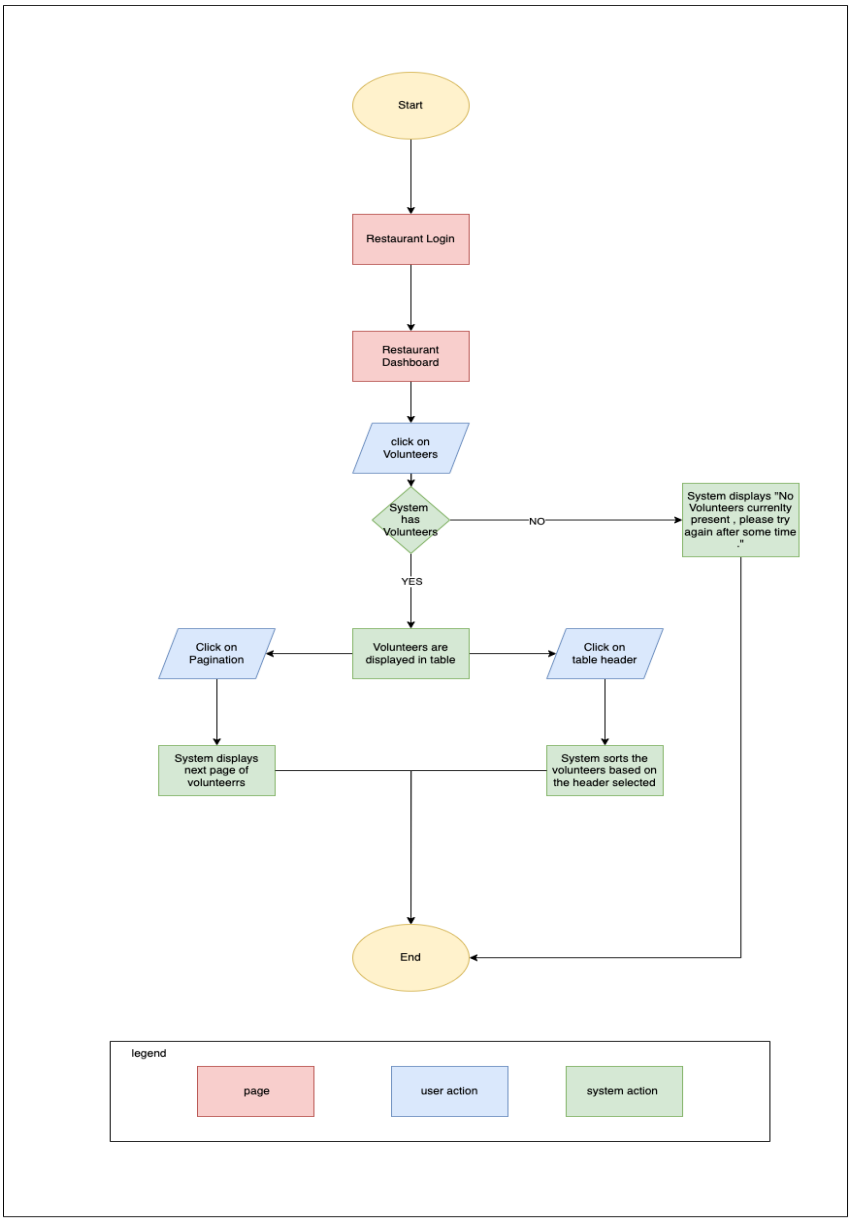

Figure 46:task flow diagram for viewing volunteers [5]

Feature: - Restaurant Profile Management
Task: restaurant registration

Below is a task flow diagram of restaurant registration. For using our application to post different food items, restaurant owners need to register their restaurant, and for that, on the register restaurant screen, they will enter all the required details and submit the form.

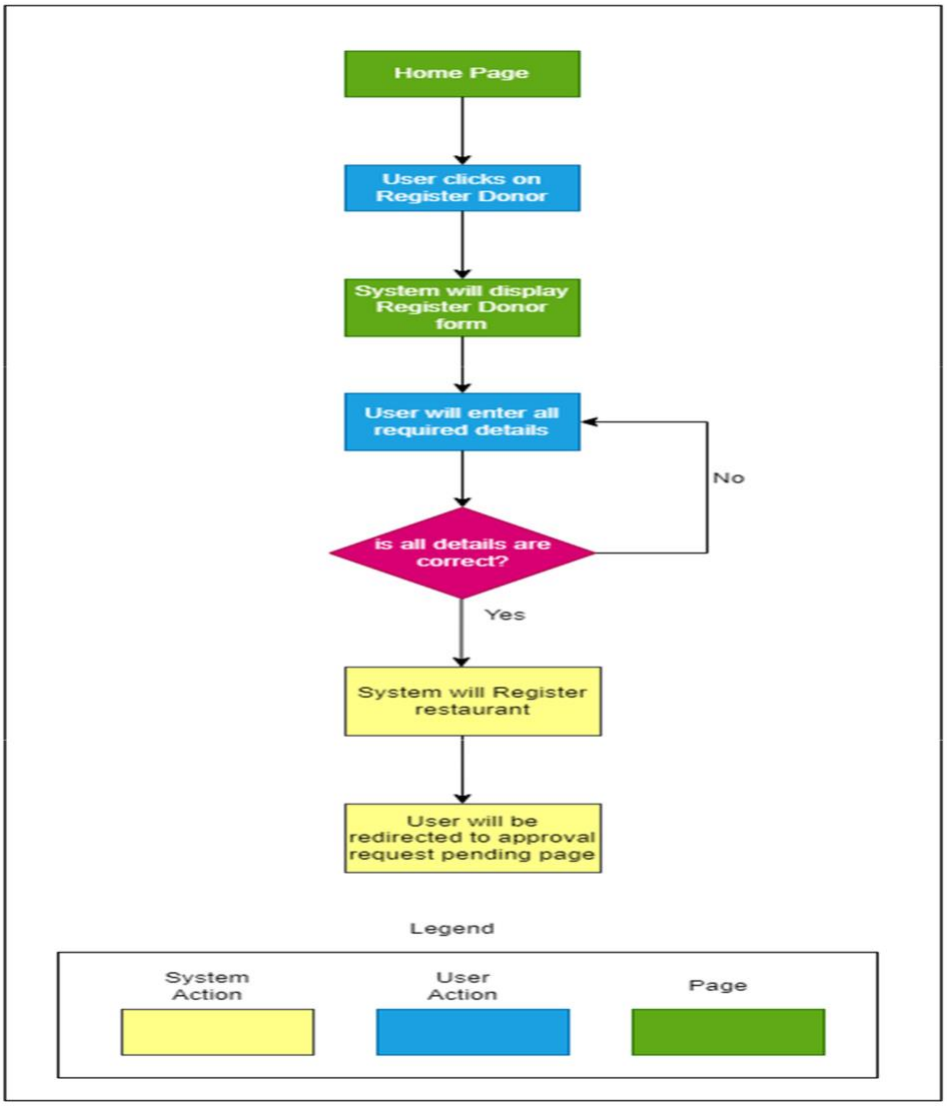

Figure 47 Task flow diagram for Restaurant registration created using draw.io [5]

Feature: - Restaurant Profile Management
Task: restaurant login

Below is a task flow diagram of a. Once restaurant registration has been approved by the admin, restaurant owners can login to our system using their username and password.

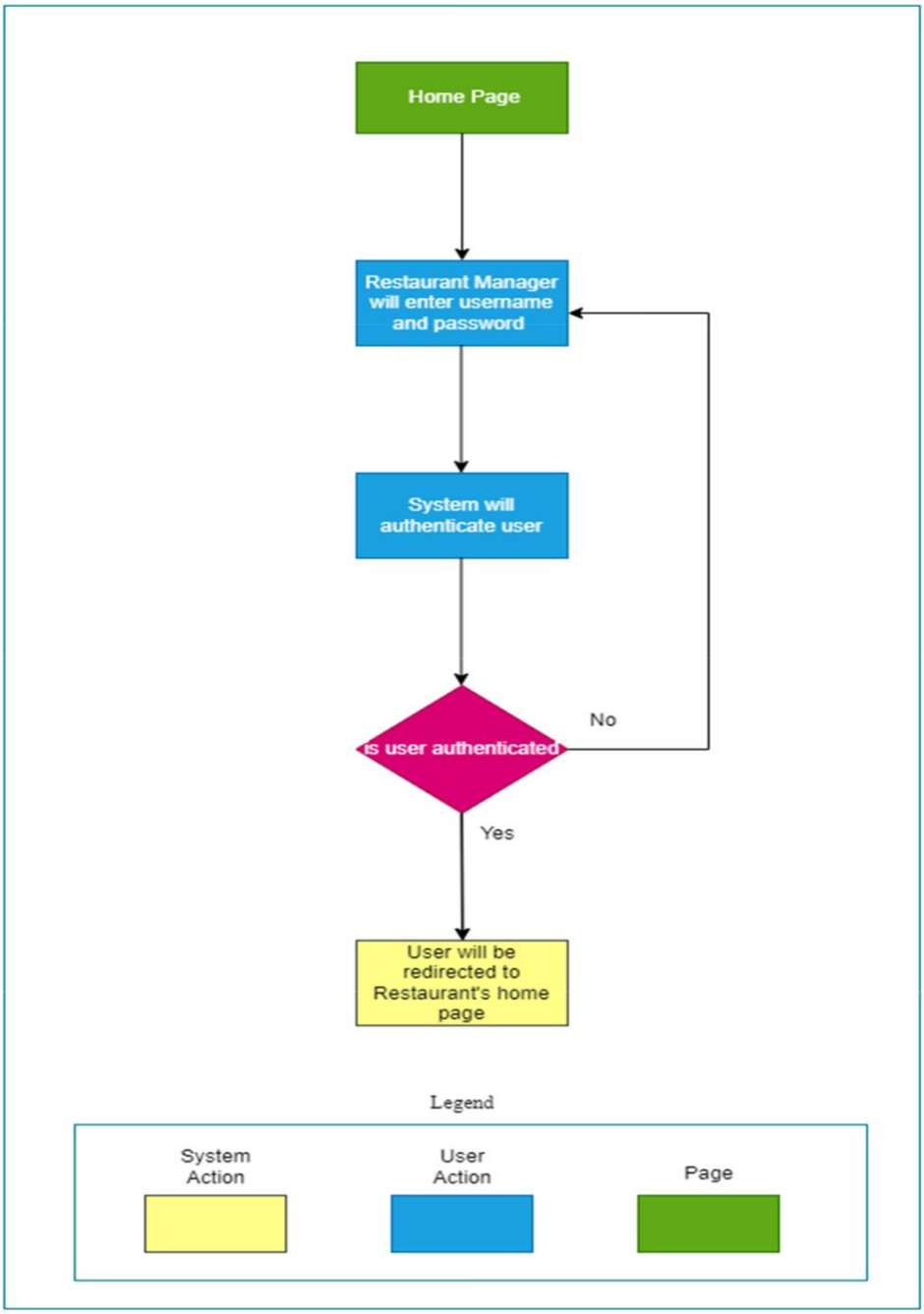

Figure 48: Task flow diagram for Restaurant Login, Created using Draw.io [5].

Feature: - Restaurant Profile Management
Task: View profile page

Below is a task flow diagram of a profile page for a restaurant. Once restaurant owners login to the system, they can view the restaurant details on their profile page. These details will be the same as those that will be provided by them during restaurant registration.

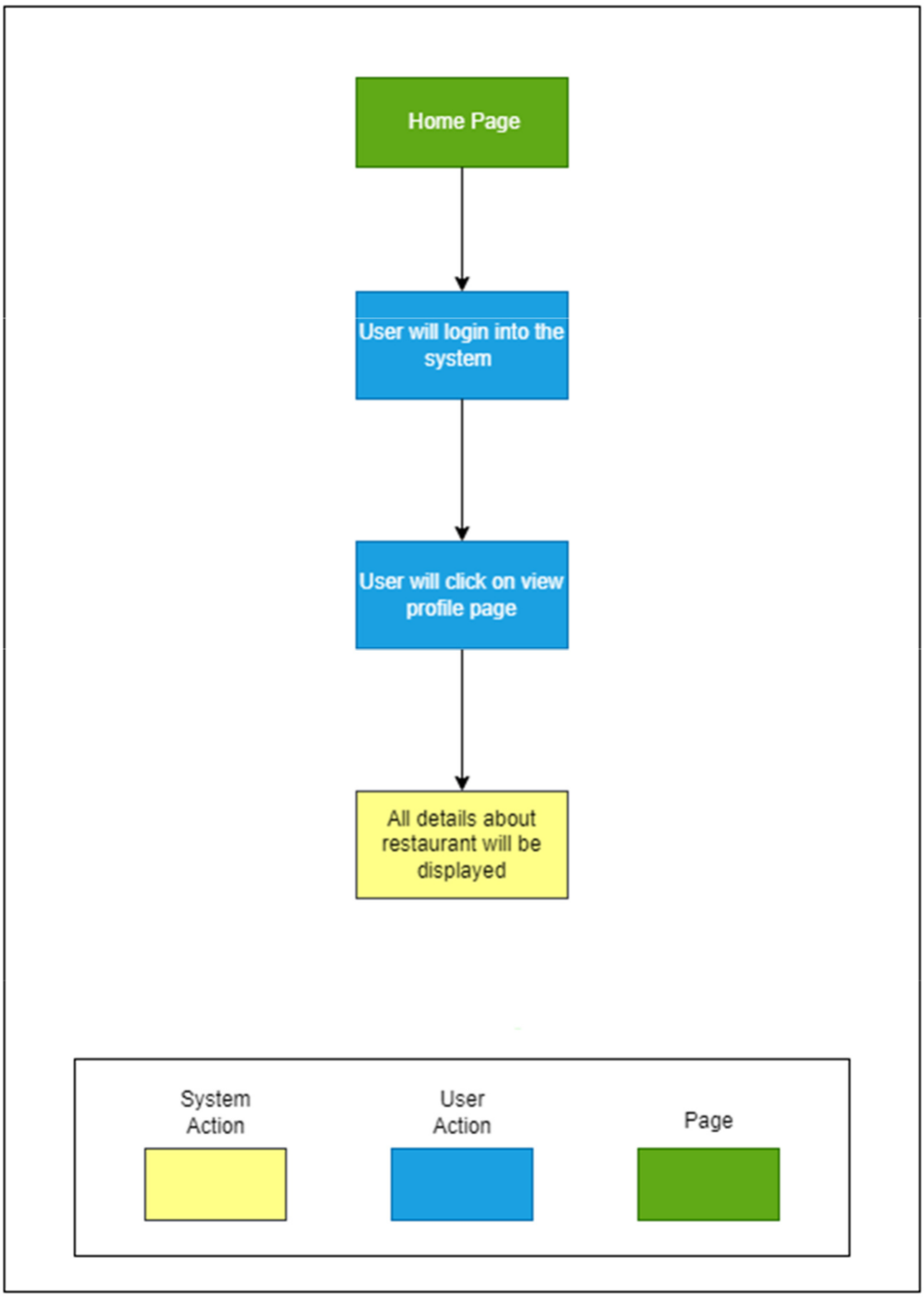

Figure 49:Task flow diagram to View Restaurant Profile, Created using Draw.io [5].

Feature: Restaurant Profile Management
Task: Update restaurant profile

Below is a task flow diagram for updating the profile of a restaurant. On the profile page, restaurant owners can view their restaurants details as well as update them based on their needs.

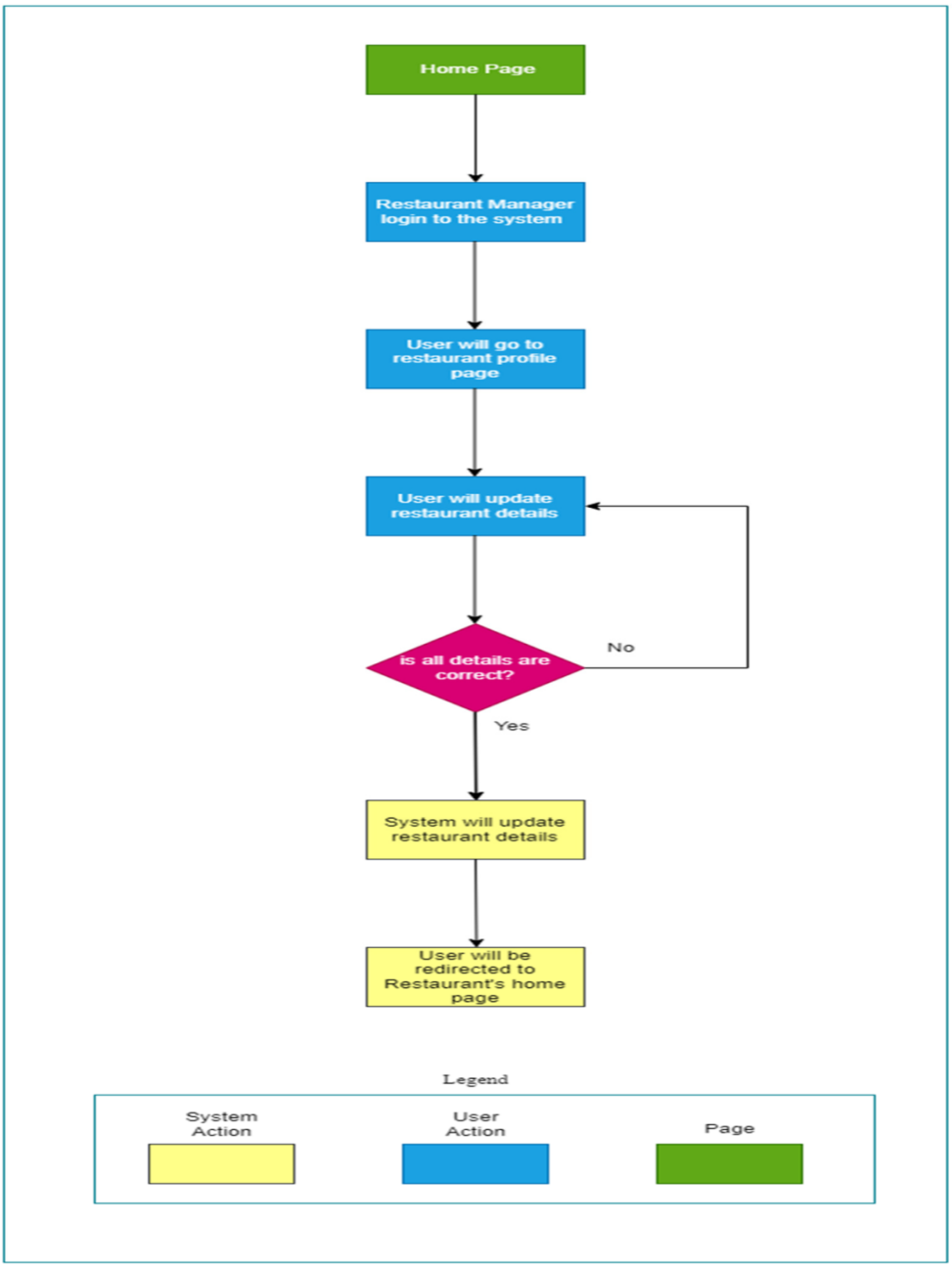

Figure 50:Task flow diagram to update Restaurant Profile, Created using Draw.io [5].

Click Stream diagram

Feature: - Restaurant Profile Management
Task: restaurant registration

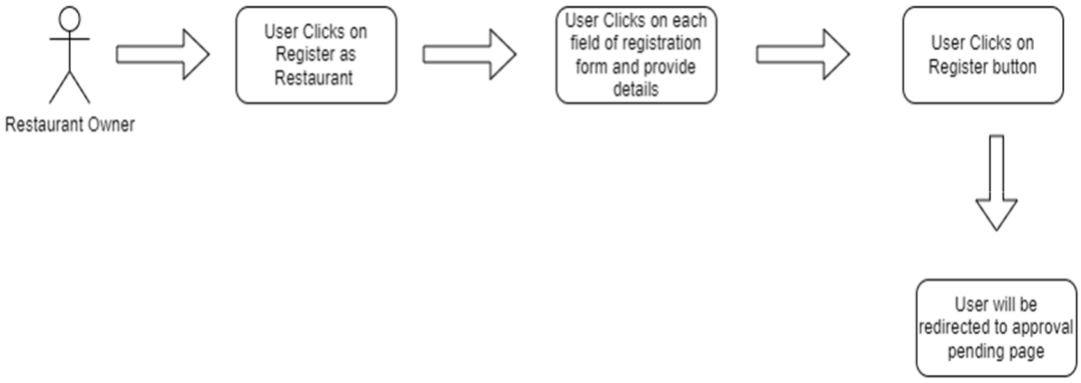

Figure 22:click stream of restaurant registration created using draw.io[]

Feature: - Restaurant Profile Management
Task: restaurant login

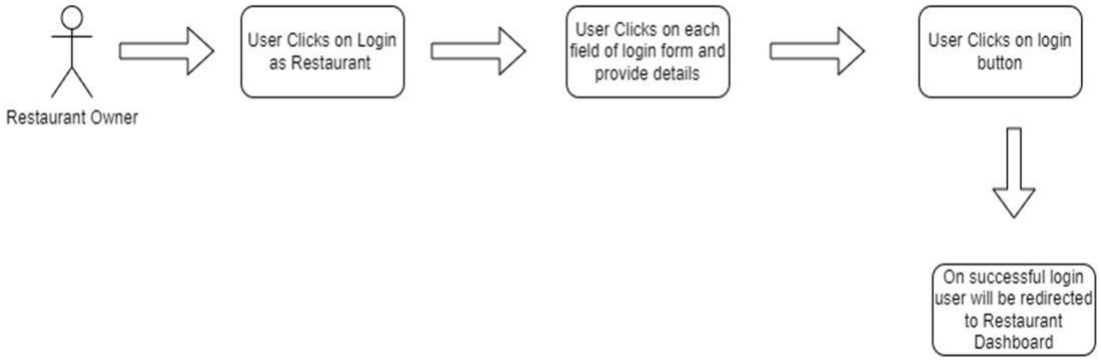

Figure 51:click stream of restaurant login created using draw.io [5]

Feature: - Restaurant Profile Management
Task: View profile page

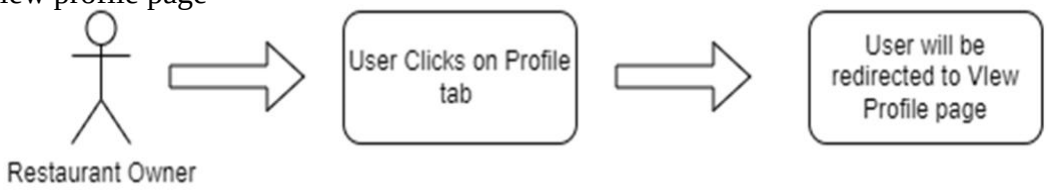

Figure 52 :Click stream of view restaurant profile created using draw.io[5]

Feature: Restaurant Profile Management
Task: Update restaurant profile

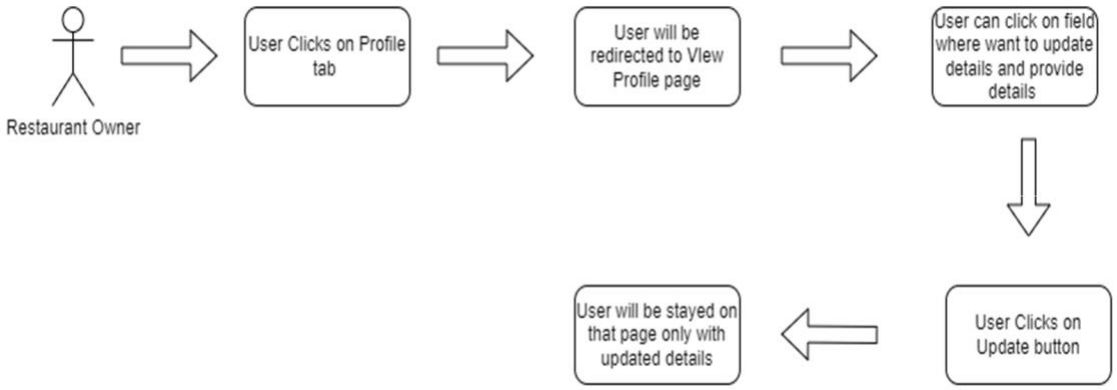

Figure 53: Click stream of update restaurant profile created using draw.io[5]

Feature: User Profile Management
Task: User Registration

User Scenario - At midnight, a student is starving but too worn out to prepare supper. Due to a limited budget, the student does not want to order food at a restaurant. The student wishes to register with a website that lists local restaurants that provide leftovers from the day at no charge. To finish the procedure, he will do the following.

Figure 53 and **Figure 54** below shows the task flow and click stream diagram for the user registration process.

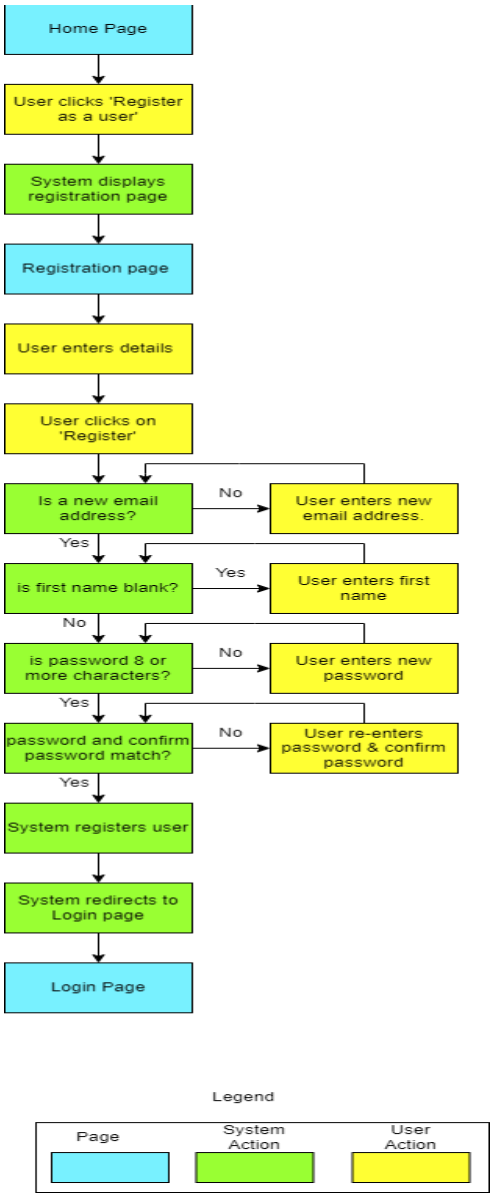

Figure 54: Task flow diagram for user registration [5]

Click Stream

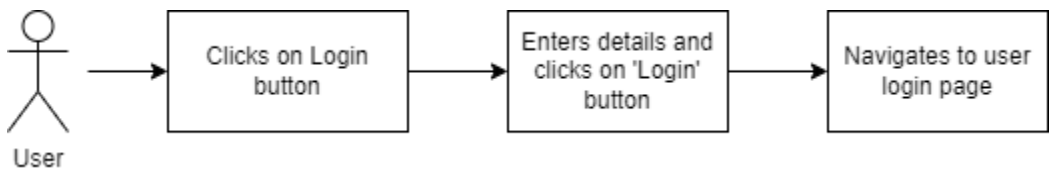

Figure 55: Click stream for user registration process[5]

Feature: User Profile Management
Task: Existing User Login

User Scenario - Login into account: A student is hungry and wants to check restaurants in the vicinity offering free leftover food on the LastServe website.

Figure 56 and **Figure 57** show the task flow and click stream diagram for the login process.

Task flow Diagram

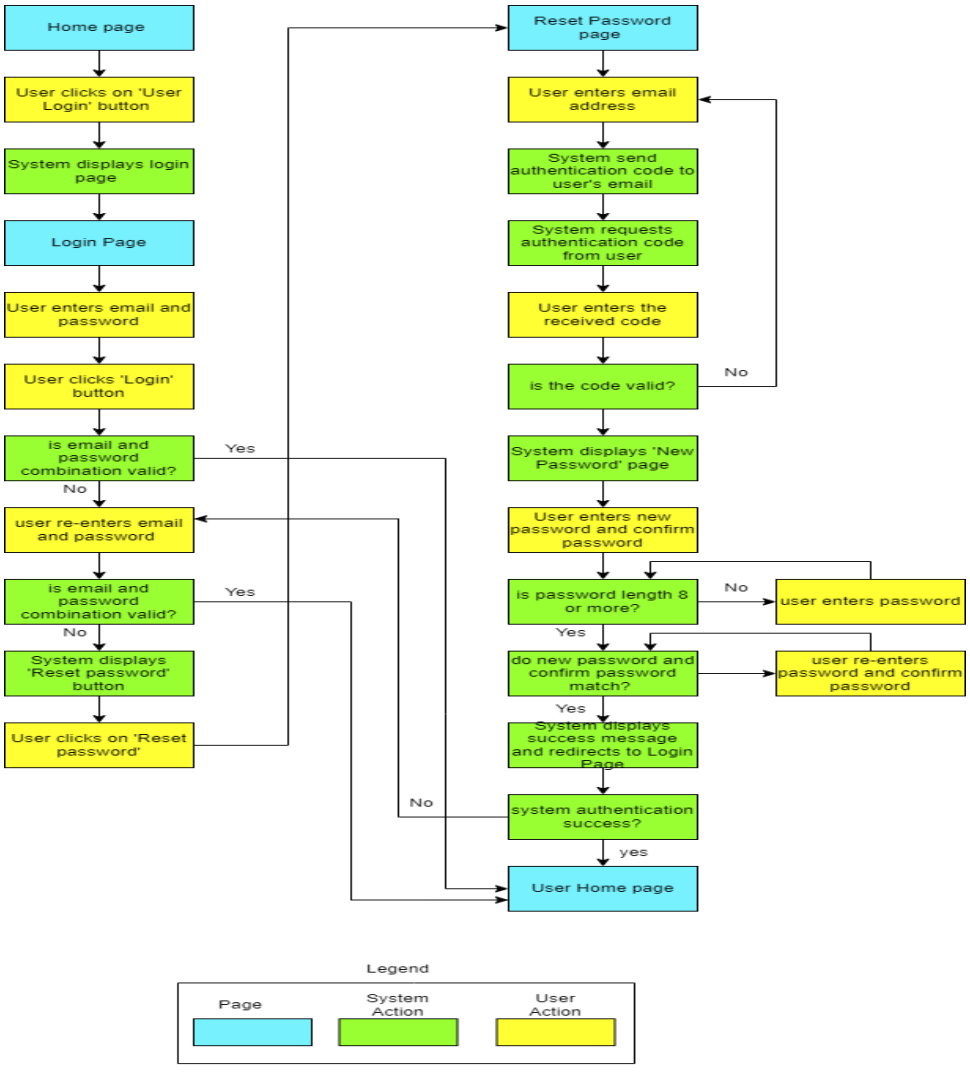

Figure 56: Task flow diagram for user login[5]

Click stream diagram

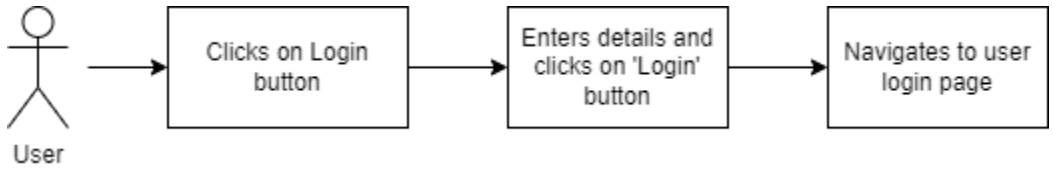

Figure 23: Click stream for user login process [5]

Feature: User Profile Management
Task: View User Profile

User Scenario - The student needs to check to the data entered while creating the account.

Figure 58 and **Figure 59** depict the task flow and click stream diagram for view user profile task.

Task flow diagram

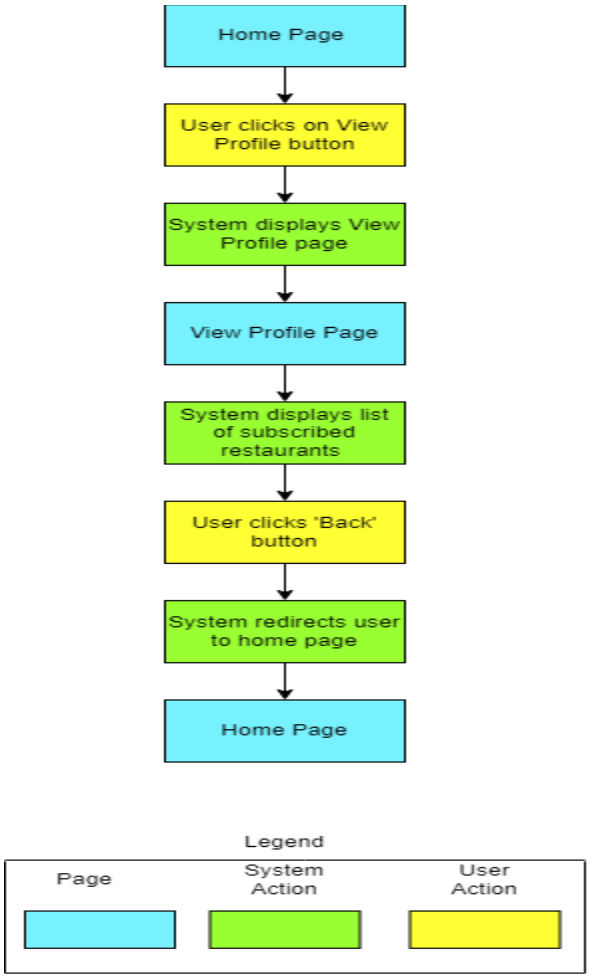

Figure 24: Task flow diagram for viewing profile details

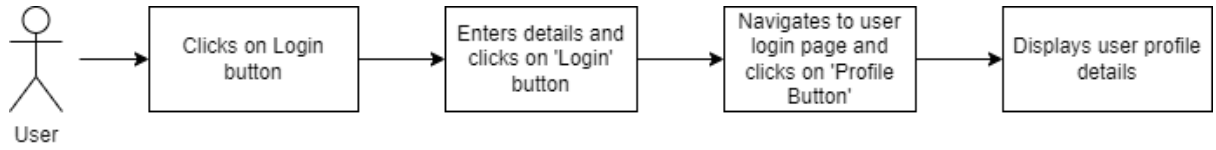

Figure 25: Click stream for view user profile details [5]

Feature: User Profile Management
Task: Edit User Profile

User Scenario - The student typed his name incorrectly when setting up an account. Also, even though it was an optional field, the student omitted the last name. The last time he visited a restaurant, he was denied access to pick up food since the name on the reservation was different from the name on his ID. To prevent future misunderstandings, the student wishes to modify his first name and add a last name.

Figure 60 and **Figure 61** shows the task flow and click stream diagram for edit details task.

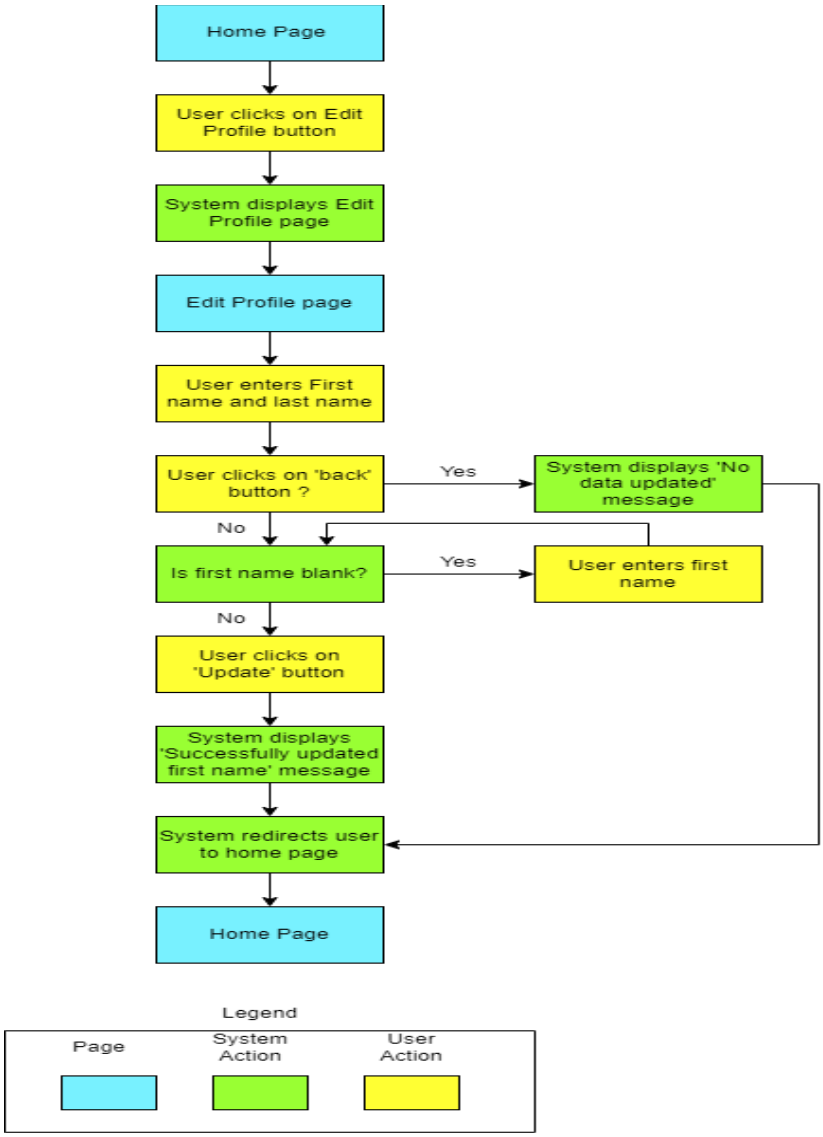

Figure 26: Task flow diagram to edit user details [5]

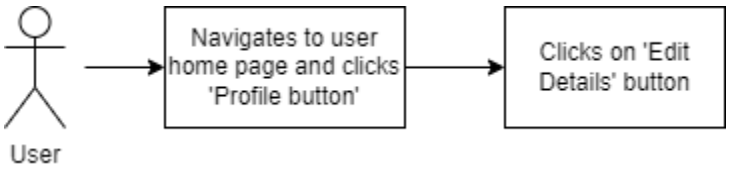

Figure 27: Click stream for edit user details task [5]

Feature: User Order Management
Task: View all active donors with timings and items offered.

User Scenario - A student is looking for restaurants that currently offer free food as they have crossed their monthly expense limit.

Figure 62 and **Figure 63** shows the task flow diagram and click stream diagram for this task

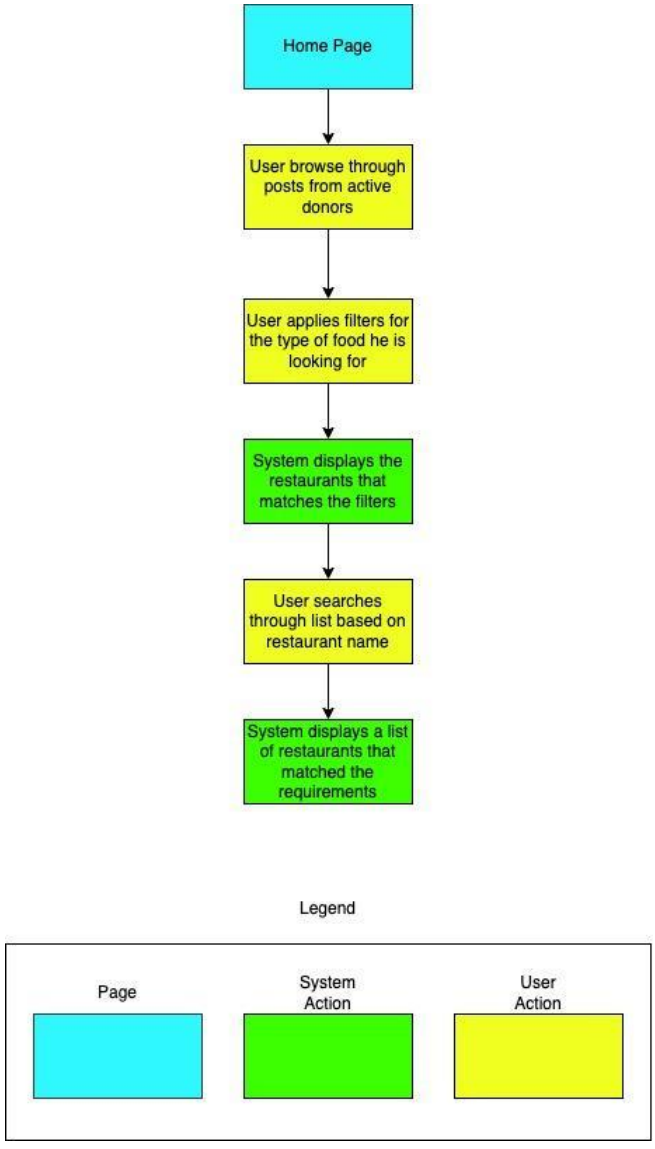

Figure 28*: Task flow diagram for viewing active donors* [5]

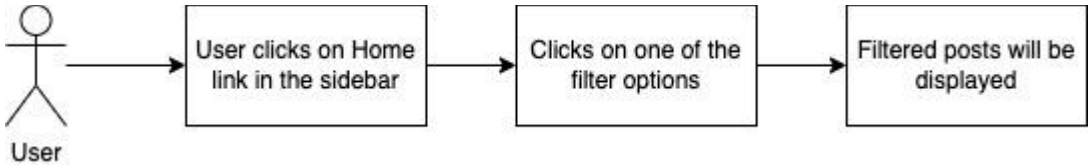

Figure 29: Click Stream diagram for viewing active donors [5]

Feature: User Order Management
Task: Appointment booking for pickup

User Scenario - The student has found a restaurant that is offering free food at a particular time and now is trying to book an appointment to pick up the food items.

Figure 63 and **Figure 64** shows the task flow diagram and click stream diagram for this task

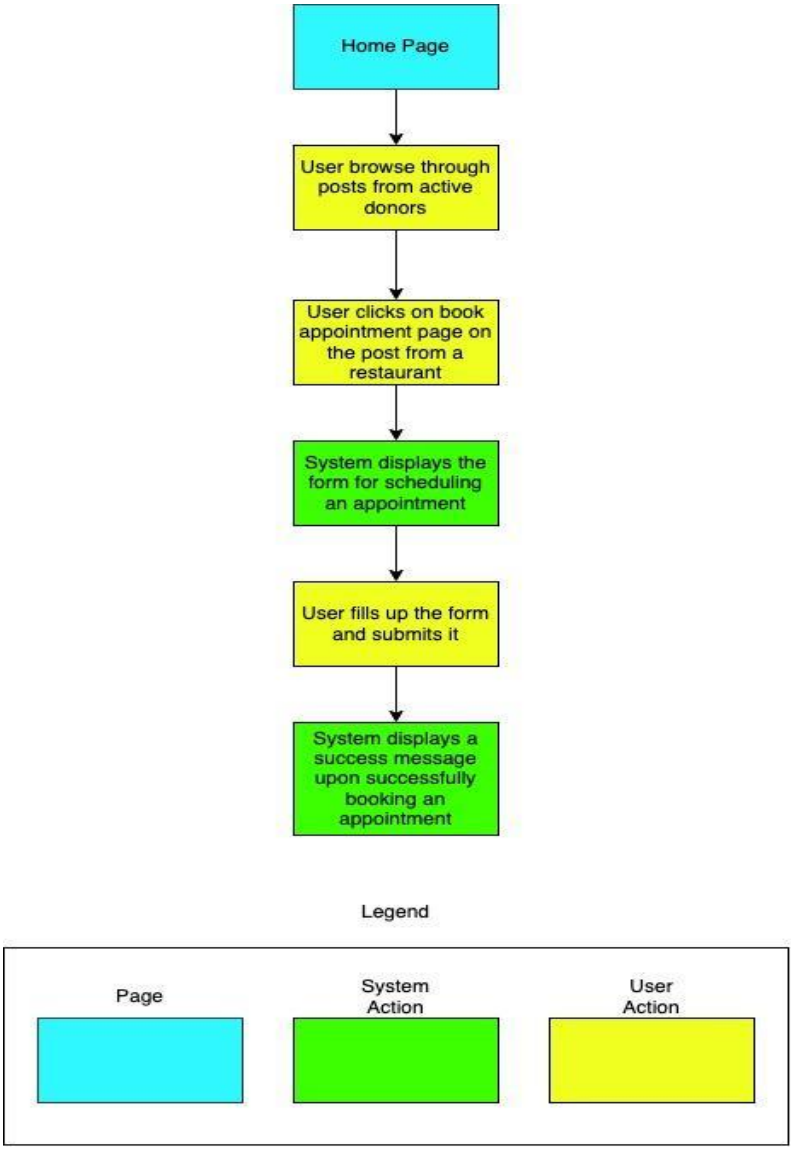

Figure 30: Task Flow diagram for appointment booking [5]

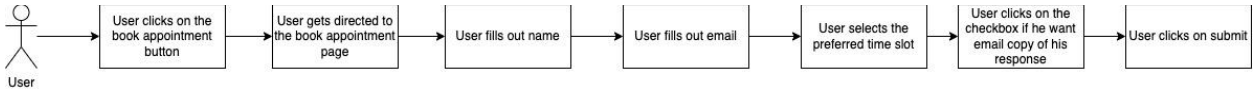

Figure 31: Click Stream diagram for appointment booking [5]

Feature: User Order Management
Task: Subscribe to a restaurant

User Scenario - The student likes food from a particular restaurant and eagerly waits for the restaurant to post about leftover food. At times, he gets too busy completing assignments and misses posts from the restaurant. He frequently checks his inbox for important emails from the university or his manager. The student wants to receive a notification whenever his favourite restaurant posts on LastServe.

Figure 65 and **Figure 66** shows the task flow diagram and click stream diagram for this task

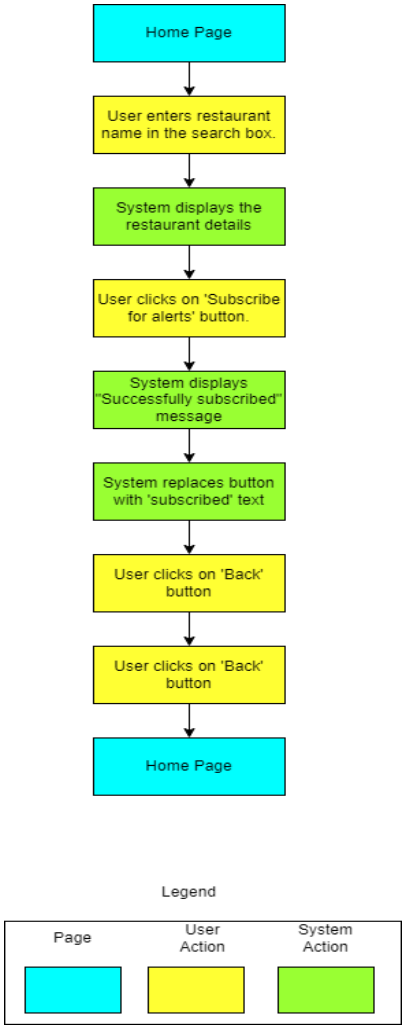

Figure 32: Task Flow diagram for subscribing to a restaurant [5]

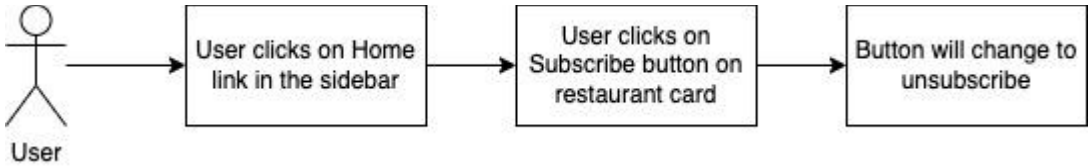

Figure 33: Click Stream diagram for subscribing to a restaurant [5]

4.2.2 Process and Service Workflow

Backend is built on NodeJs .Restful services are developed to serve as an interface between frontend and backend. Overview of the backend's working is a s follows

1. Request first hits the backend routes through express.
2. This request data is then forward to the specific controller responsible for that route.
3. Business logic resides in the controller
4. Controller establishes connection using the singleton connection class using mongo-client library.
5. Controller fetches data from database.
6. Output is converted into JSON format and sent to the client.

Process workflow diagram

Chosen feature : **Restaurant order management.**
The following workflow will be used for the given feature and tasks

Feature: Restaurant order management.
Task: View orders

API's used

1. View active orders: get domain/restaurantorders/active
    a. Get request forwards the request to restarauntordercontroller.get_active_posts() function
    b. Once the request is received the headers are checked if the user is a restaurant admin by checking the restaurant ids in database.
    c. All the orders for the restaurant are fetched using async query calls to database
    d. These orders are filtered for active orders and returned in JSON format

2. View past orders: domain/restaurantorders/past
    a. Get request forwards the request to restarauntordercontroller.get__posts() function
    b. Once the request is received the headers are checked if the user is a restaurant admin by checking the restaurant ids in database.
    c. All the orders for the restaurant are fetched using async query calls to database
    d. These orders are filtered for active orders and returned in JSON format

Both backend processes can be explained via sequence diagram (**figure 67**) and process workflow diagram (**figure 68**).

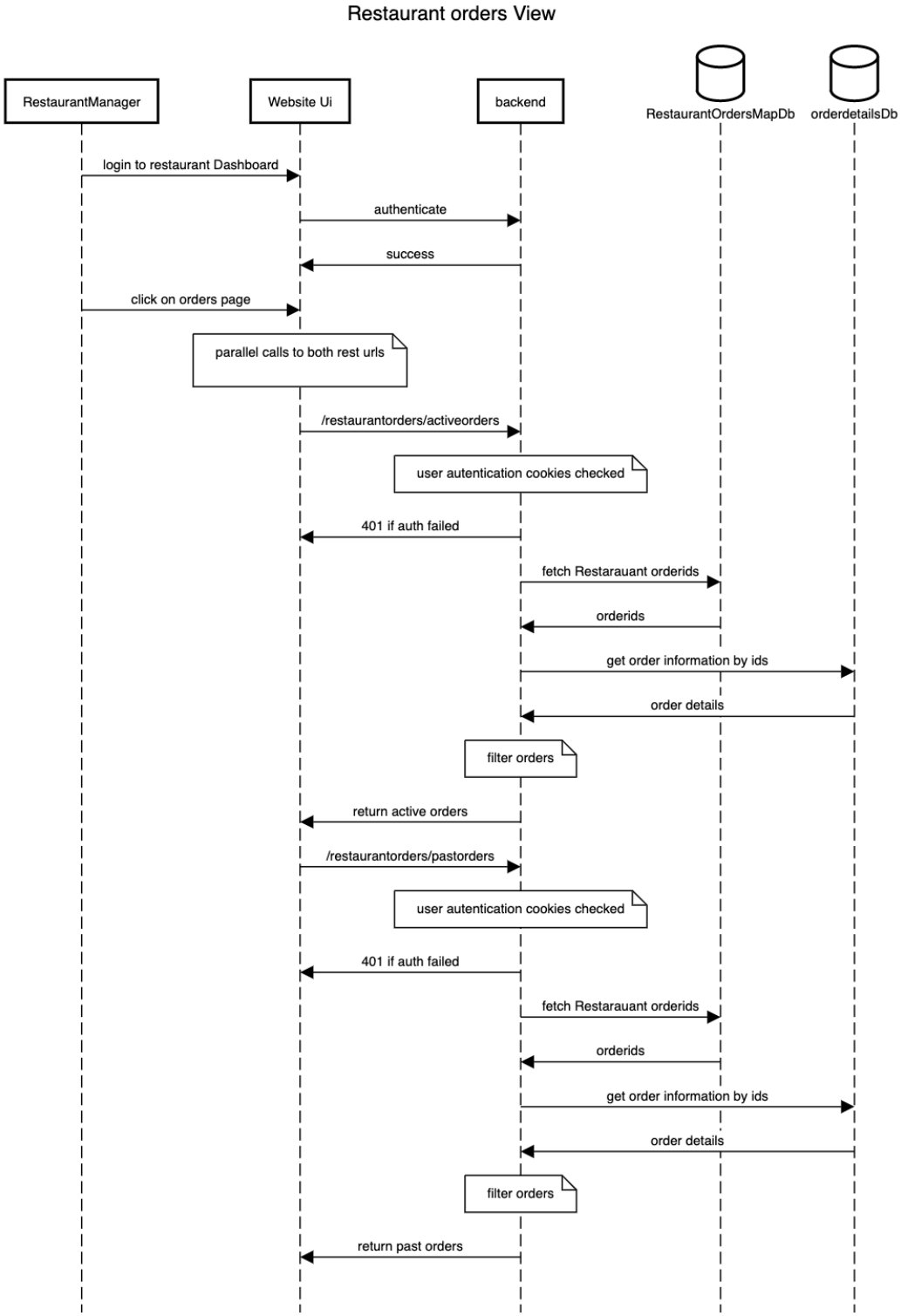

Figure 34:Sequance diagram to view orders [5]

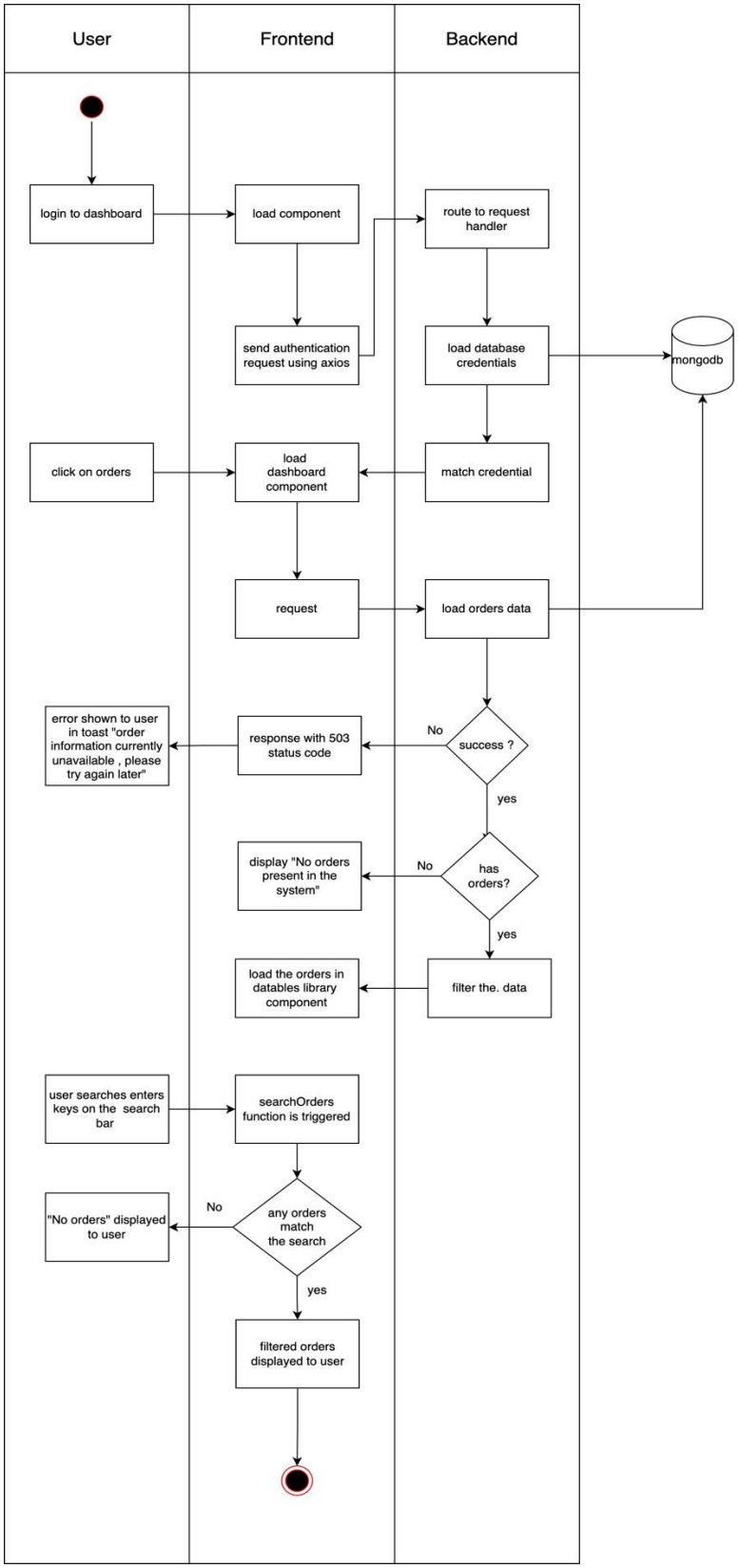

Figure 35:Process flow diagrams to view order created using draw.io [5]

User Scenario- View orders by user, their timeslots, and items:

Restaurant Manager has posted the information on excess food available on the last serve. He wants to check the users and their appointment time slots to collect the food, and items and quantity ordered by them on the application so he can pre-pack the orders and allocate resources for order collection. He checks through the dashboard of all orders placed and their time slots.

Persona: Restaurant Manager

Feature: Restaurant order viewing

Need: Check the appointment and better manage the time and food

Context: They have placed the post and want to see the information on users who have ordered.

User Persona

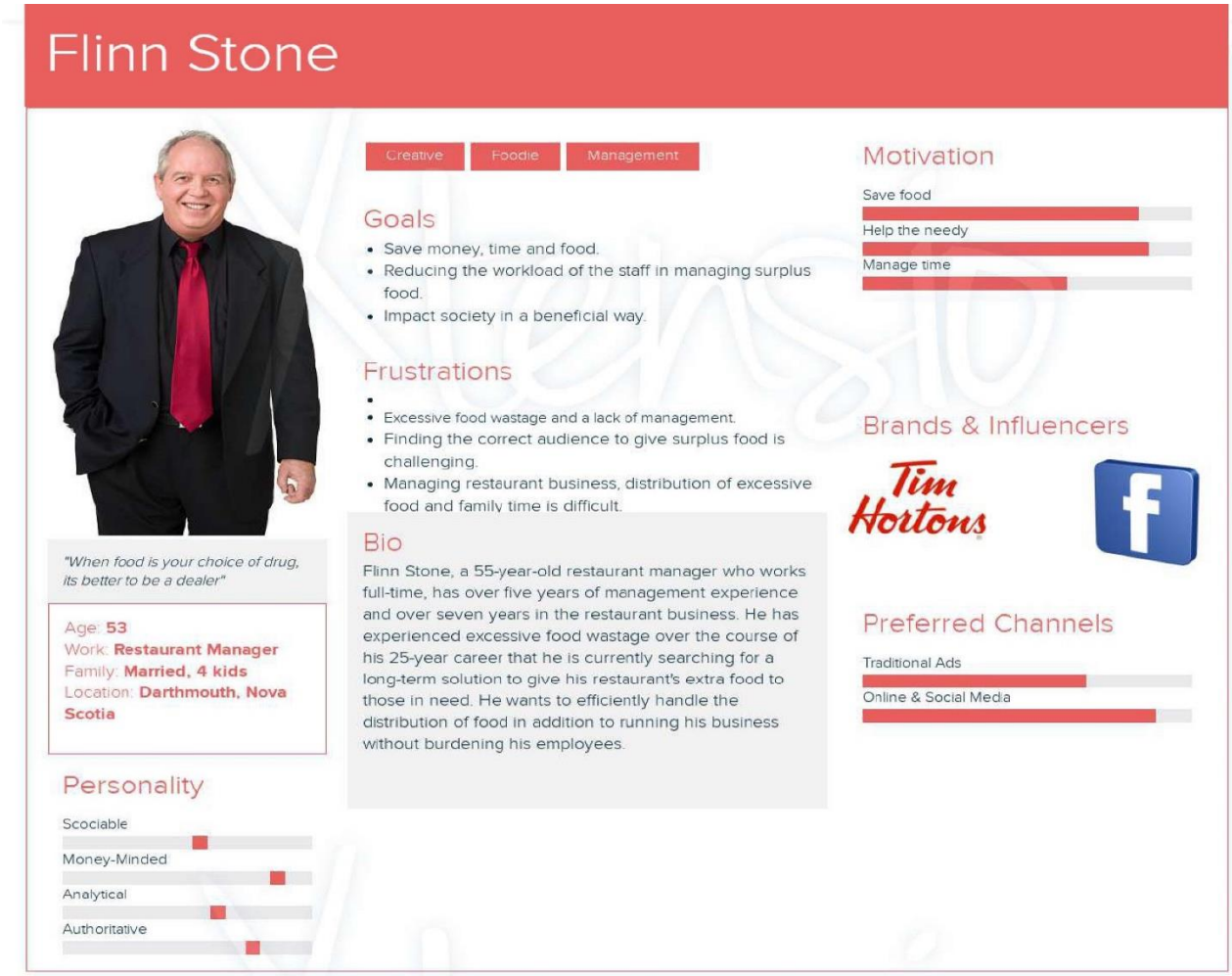

Figure 36:User persona of restaurant manager prepared using xtensio.com [6]

Use Case

1. The restaurant manager has successfully logged in.
2. Restaurant managers see the Restaurant dashboard with various functions
3. The restaurant manager clicks on the orders.
4. Two tables are displayed One for the active orders and completed past orders.
   4.1. There are no active orders user is displayed “there are no active orders”.
   4.2. The backend call fails user is displayed “order information not available currently please try again later”
5. There are no past orders for the restaurant. The active orders table has the information on the order number, the Name of the user items ordered appointment time, the status of the order, and actions to be performed on the order.
   5.1. Order with status pending is the order that is just placed, in the actions two buttons are shown as packed, and picked. Order with status packed is the order that is packed but not picked up, in the actions one button is shown as picked.
   5.2. The restaurant manager wishes to see other orders than the ones displayed in the active orders table.
6. The restaurant manager clicks on the next page tab at the bottom of the table.
7. The restaurant manager wants to see the orders sorted based on appointment time, he clicks on the appointment time header.
8. The system sorts the orders and displays the sorted order.

The system displays past in the table below active order with the following columns order number, name, and items.[1]

Feature: Restaurant order management.
Task: Change order status

The given task changes the order status from pending, picked and packed

API's used

1. change order status: post domain/restaurantorders/changeorderstatus/:id
    a. Get request forwards the request to restarauntordercontroller.post_change_orderstatus() function
    b. Once the request is received the headers are checked if the user is a restaurant admin by checking the restaurant ids in database.
    c. The status of the given order is updated in database using updateOne() function to the one provided in the order
    d. If the status change is successful response status code 200 is returned or else 503 is returned.

Process workflow diagram (**figure 69**) that depicts the interaction of frontend and the above process.

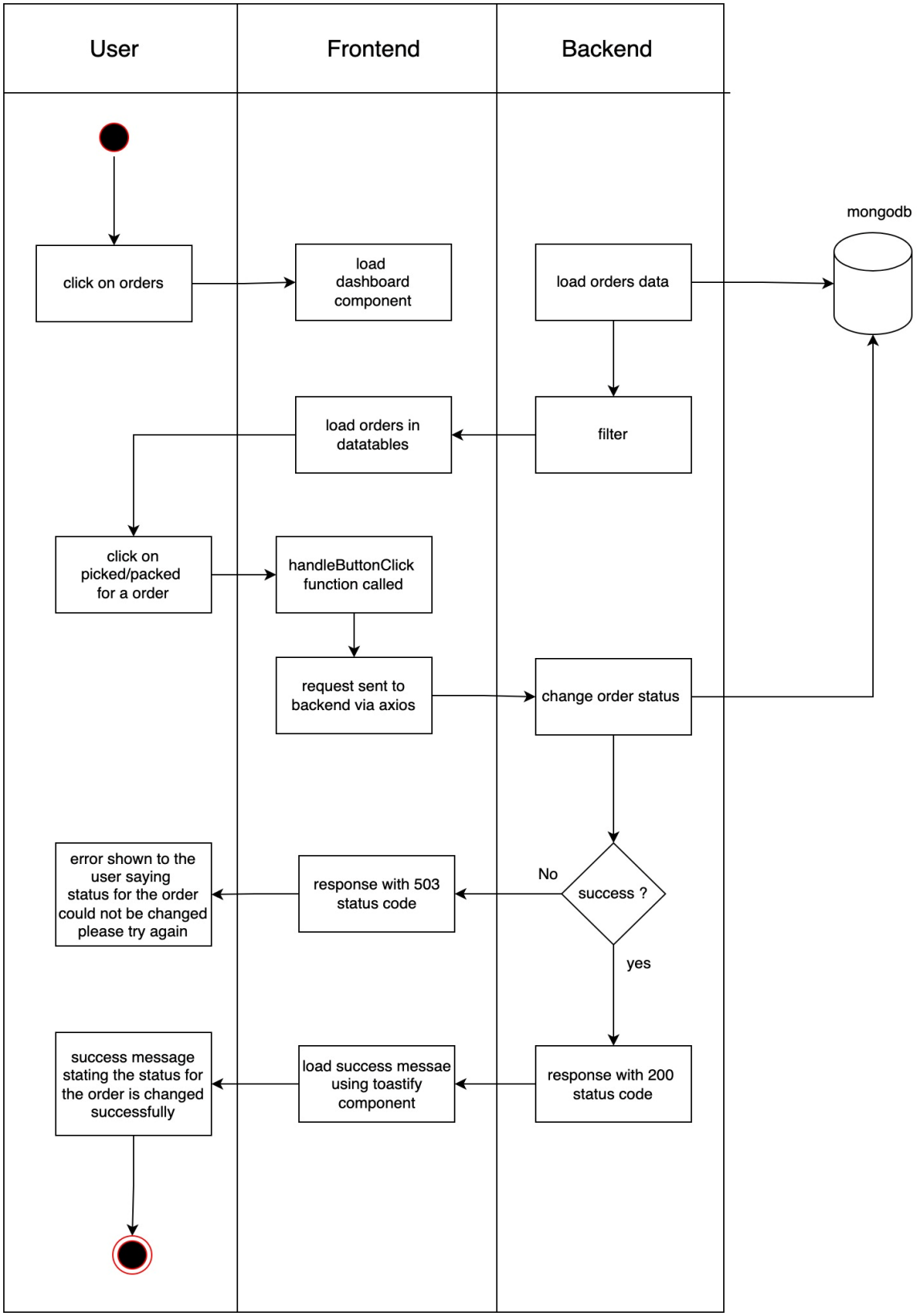

Figure 37:Process work flow diagram to change order status created using draw.io [5]

User scenario

Task: Change order status from pending to packed and picked

The restaurant is viewing all the orders placed by users. He wants to change the order status once he has packed an order so other employees working on it can know to pack other orders. he also wants to mark an order as picked up once it is picked up by the user so it is easier to manage the order. He clicks on the packed and picked-up buttons once it is done.

> ***Persona:*** Restaurant Manager
>
> ***Feature:*** Change order status from pending to packed and picked
>
> ***Need:*** keep track of orders that are packed and picked, so the pending orders can be worked on.
>
> ***Context***: Once the order is placed by the user Restaurant manager wants to track the order through different statuses.

Use case

1. The restaurant manager has logged in using his credentials.
2. The restaurant manager clicks on the orders.
3. The system displays all the active and past orders on tables.
   3.1. The system displays no active orders if there are no active orders for the restaurant.
4. The restaurant manager clicks on the packed button once an order is packed.
5. The system changes the state of the order to pack.
6. The system displays a message saying the order status is changed successfully.
   6.1. If there is an error while changing the state of the order “error something went wrong, please try again is displayed to the user”.
7. The restaurant manager clicks on picked up once the order is picked up.
8. The system changes the state of the order.
9. The system displays saying “Order status changed successfully”, System removes the order from active orders and adds it to the past orders table.
10. If there is an error while changing the state of the order user is displayed the error message “Something went wrong, please try again”.

Feature: Restaurant order management.
Task: View Volunteer available for order management

API's used

1. change order status: get domain/restaurant/volunteers
    a. express route forwards the request to restarauntVolunteerController.get_volunteers () function
    b. Once the request is received the headers are checked if the user is a restaurant admin by checking the restaurant ids in database.
    c. All the volunteers are fetched from collection "Volunteers".
    d. If fetching is successful volunteer details are returned in JSON format else response status code 503 is returned.

Process workflow diagram (**figure 71**) that depicts the interaction of frontend and the above process.

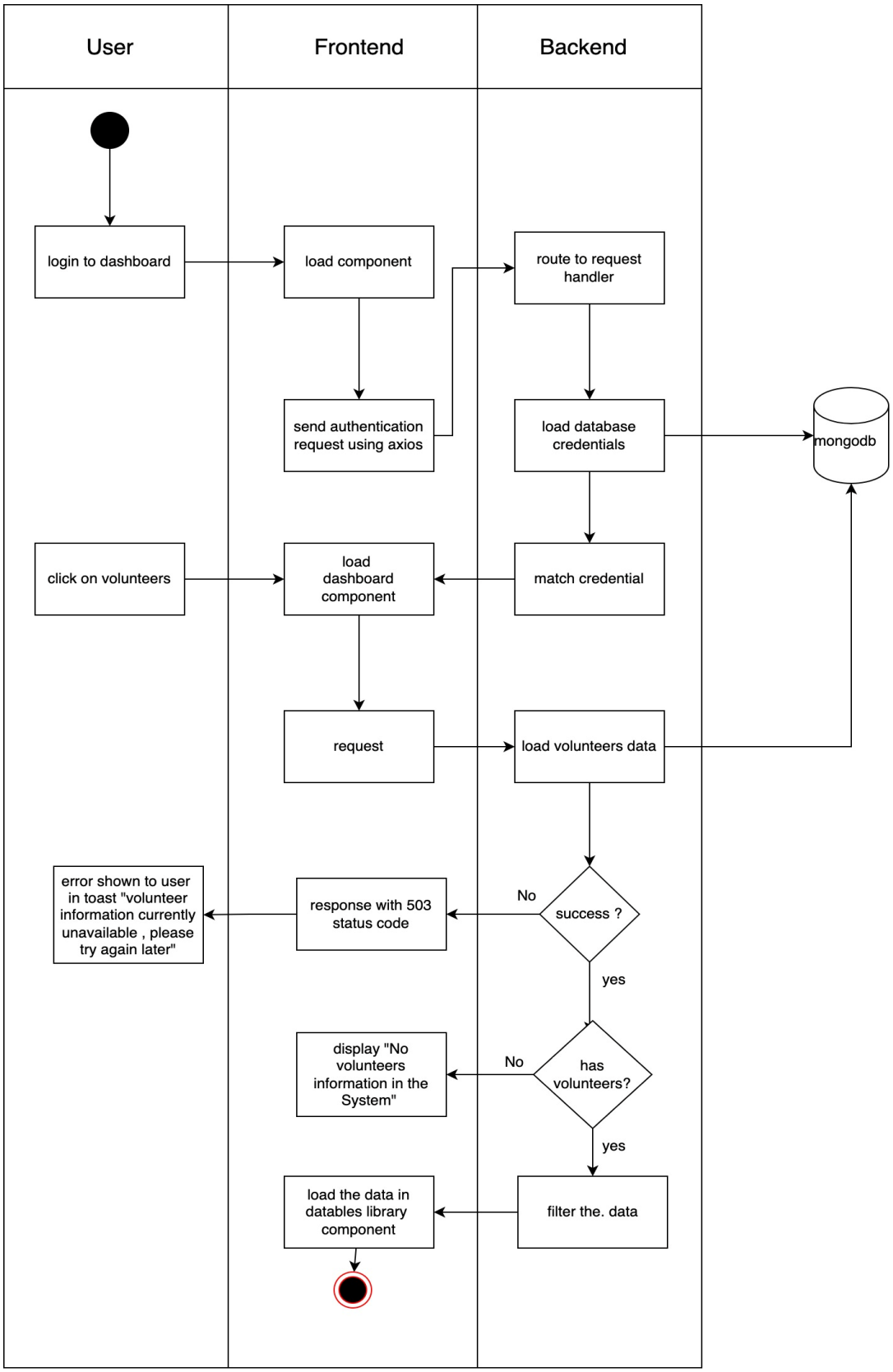

Figure 38: Process work flow diagram to change order status created using draw.io [5]

User Scenario
Task: View Volunteer information by their availability

The restaurant manager is viewing all the orders placed by users. He finds the the volume of orders from last serve are more than usual and number of employees scheduled to work currently may face difficulty fulfilling the orders. He checks the portal if there are any volunteers available currently to help him with the orders.

> ***Persona:*** Restaurant Manager, Volunteer
>
> ***Feature:*** Change order status from pending to packed and picked
>
> ***Need:*** keep track of orders that are packed and picked, so the pending orders can be worked on.
>
> ***Context***: Once the order is placed by the user Restaurant manager wants to track the order through different statuses.

Use Case

1. The restaurant manager has successfully logged in.
2. Restaurant managers see the Restaurant dashboard with various functions
3. The restaurant manager clicks on the Volunteer.
4. Table is displayed with volunteer information based on name, phone number and availability.
   4.1. There are no volunteers in the system , user is displayed “there are no volunteers”.
   4.2. The backend call fails user is displayed “volunteer information not available
5. The restaurant manager clicks on the next page tab at the bottom of the table.
6. The restaurant manager wants to see the volunteer sorted based on availability, he clicks on availability header.
7. The system sorts the volunteer and displays the sorted volunteers to the manager.

8. **User Persona** : Volunteer

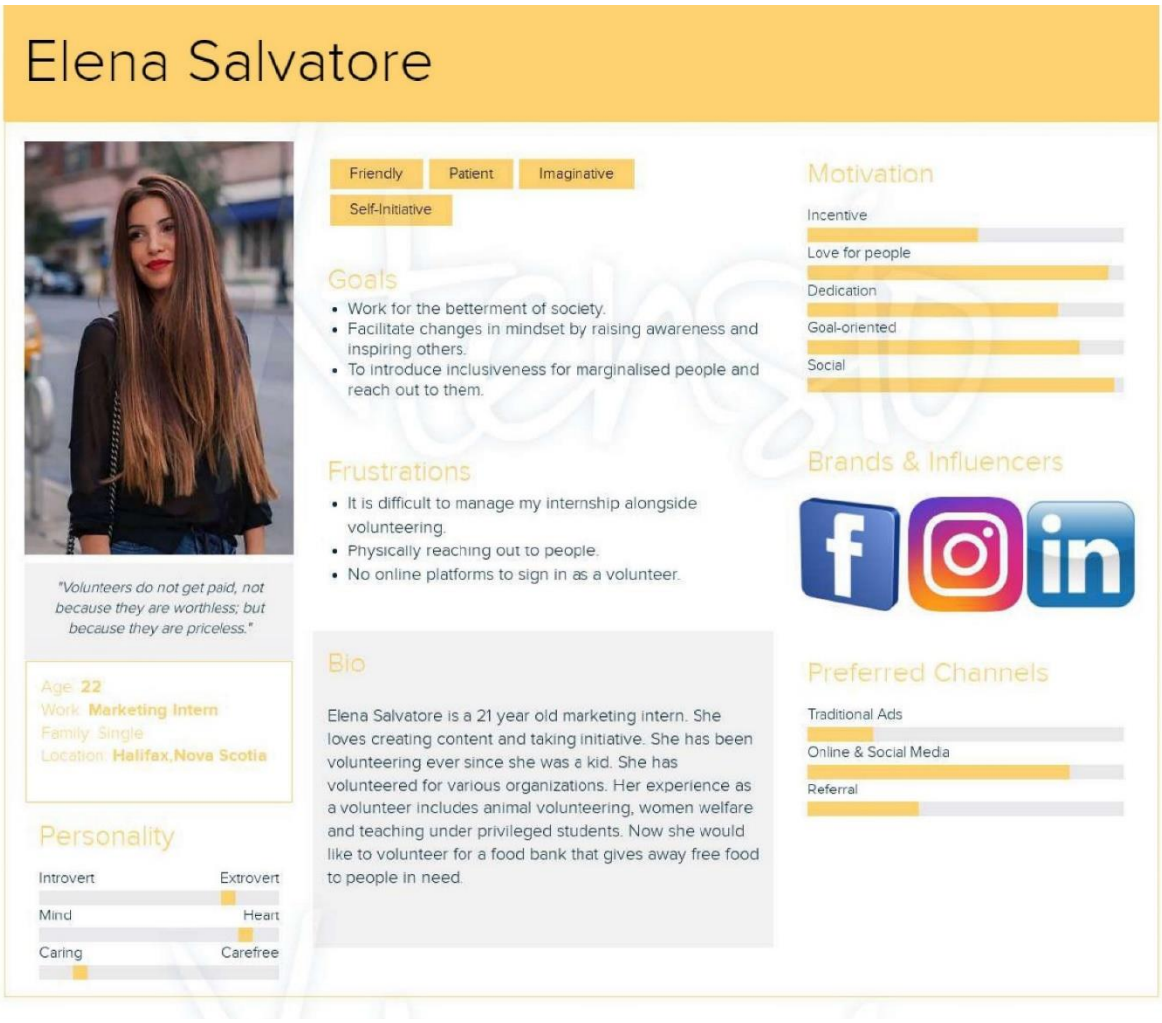

Figure 39: User persona of Volunter created using xtensio.com [6].

Folder Structure

Frontend

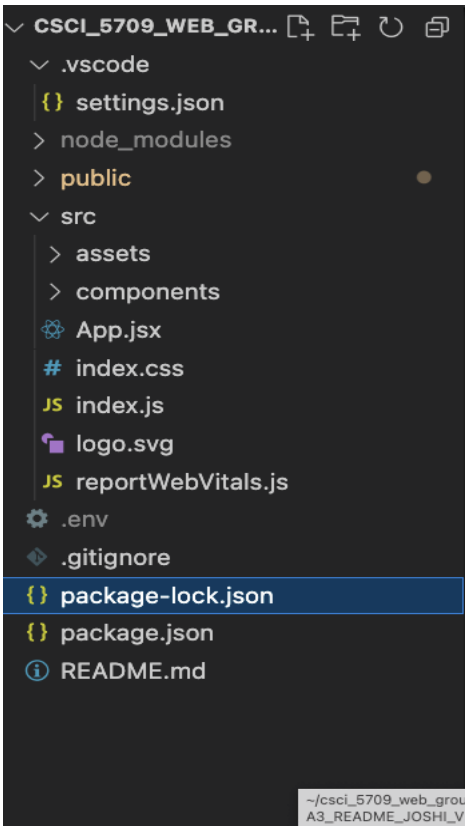

Figure 40:Frontend folder structure screenshot

Backend

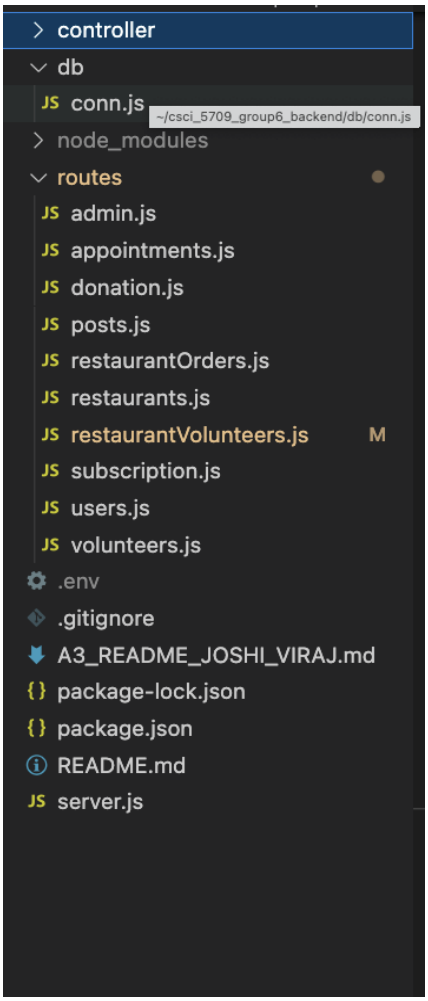

Figure 41: folder structure of backend

5. Search engine optimization

Various approach have been followed to enhance the SEO(Search engine optimization) of the application

5.1 Implemented robots.txt

- Allowed all user agents and search engine crawlers
- Added paths in disaalowed which require a login so the crowaler does not reduce the ranking of the page based on the paths availability

```
# https://www.robotstxt.org/robotstxt.html
User-agent: *
Disallow: /admin
Disallow: /restaurantSideBar
Disallow: /adminDashboard
Disallow: /pastPosts
Disallow: /createPost
Disallow: /updatePost
Disallow: /restaurantSideBar
```

Figure 42: screenshot of robots.txt

.

5.2 Added Meta information

- Added Appropriate titles to pages so better SEO
- Added relevant Meta description for the pages so the crawler can associate the pages based on the description

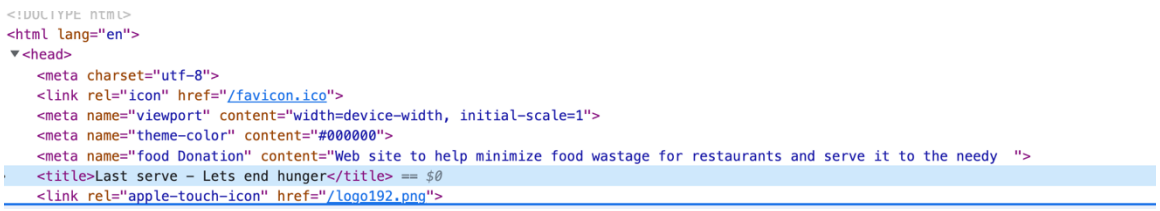

Figure 43: Screenshot of homepage source code as rendered on browser

5.3 Added relevant content on pages with proper headings

- Homepage and other landing pages have appropriate contents for the website crawler to get grouped into relevant keywords

Schedule appointments and pickup leftover food

By providing a convenient and easy-to-use system for scheduling pickups, more organizations and individuals may be willing to donate their surplus food, ultimately benefiting the community and reducing food waste.

Filter restaurants based on zipcode and food preference.

By using filters based on zip code and food preference, users can quickly and efficiently narrow down their options, saving time and potentially discovering new dining options they may not have otherwise considered.

Figure 44: Screenshot of homepage content

6. Optimizations

Various approach have been followed to optimize the performance of the Website

1. The Larger CSS stylesheets were split into smaller css files , only common css properties were loaded for all pages.
2. Each component has its own styled div.
3. Use effect in react has been given appropriate dependencies wherever required this reduces the number of times the component is reloaded
    a. For example the src/components/RestaurantOrder/RestaurantOrders.jsx calls backend api to get active and past orders of the restaurant, whenever a change is made to the order status only the components dependent on the values are reloaded rest components of the

pages do not reload.

```
useEffect(() => {
  const getActiveOrders=async () => {
  const result = await axios.get(`${process.env.REACT_APP_BACKEND_
  if(result.status===200){
    setActiveOrders(result.data)
    setTempActiveOrders(result.data)
    setActiveOrdersPending(false)
    setUpdateOrders(false)
  }
}
getActiveOrders();

},[updateOrders]);
```

Figure 45: Screenshot of file RestaurantOrders.jsx

4. Clubbed necessary backend calls into single service this reduces the load on backend and the network latency between frontend and backend.
    a. The api /admin/overview : provides count of multiple data points , rather then using multiple services to get the data
5. Optimized the database by searching the NoSQL documents via primary key rather then using search via any key in mongo, this provides the response in O(1) time complexity as opposed to O(n) time complexity of the earlier approach
    a. The restaurant document in the restaurant collection stores all of the order-Ids as array and then based on this array the details are fetched rather then searching in orders collection for the restaurant

7. Security

Various approach has been used to enhance the security of the application

1. Protecting data in flight: Https protocol is used to encrypt the data in flight. This prevents password sniffing attacks that trace the packets and try to exploit any passwords or sensitive information sent in plain text through the network.
2. Protecting passwords: any stored password in the database are hashed using Bcrypt , any security breach will only reveal the hash of the passwords .Hashes are one way , to derive the password from a hash is next to impossible
3. Externalizing configuration: All secret keys used are externally added to the code via env file , this prevents any hardcoded config to reveal essential secret keys to the hacker
4. Route protecting: all react routes are protected that require user authentication.

.

8. CONCLUSION

LastServe is an intermediary website that allows restaurants to sign up and donate food to those in need.
We created this website to take an initiative to provide a platform for restaurant who would like to come forward to donate food from their restaurants to those in need; but do not have the means to do so.
LastServe not only works towards those in need of food but also makes sure restaurant have an online medium to manage their new initiative which involves giving away food for free.
LastServe is a business that not only focuses on user experience but also pays attention to details such as privacy, integrity, authentication and security.

Technical Colclusion:

So as a part of this project, we have completed almost all the features that we mentioned in our group proposal. While implementing all the functionality, we try to achieve it in the best possible way so that the end user likes our product. We made our website (LastServe) in such a way that it meets all the below criteria.

- User Experience: LastServe is easy to navigate and use. The user will be able to find what they are looking for quickly and easily.
- Visual Appeal: Last Serve is visually appealing, with a clean and modern design. It uses a color scheme that is easy on the eyes and not overwhelming.
- Mobile Responsiveness: LastServe is mobile-friendly and responsive. More and more users are accessing the internet through their mobile devices, and a website that is not mobile-responsive can be frustrating to use, so LastServe will be good for users in terms of responsiveness.
- Page Load Time: LastServe loads quickly. We have made sure that all the pages get loaded immediately when the user opens them, and if it takes time to load, it should give a proper message to the end user so that they know what is going on.
- Content: LastServe has high-quality, relevant, and engaging content. We made the content easy to read and understand. Moreover, we have placed only relevant images throughout the website.
- Security: LastServe is a very secure website, and we protect our users' personal information. One such example is when we store sensitive data of a user, we always store it in encrypted format so that the users' privacy is not compromised.

9. REFERENCES

[1] “Food Waste Statistics in Canada for 2023 - Made in CA,” *Made in CA*, Sep. 15, 2022. https://madeinca.ca/food-waste-canada-statistics/ (accessed Feb. 28, 2023).

[2] MOONSHOT COMPOSTING, “Restaurant Food Waste – How Much Food Is Thrown Away by Restaurants & Why?,” *MOONSHOT COMPOSTING*, Jun. 03, 2020. https://www.moonshotcompost.com/restaurant-food-waste-how-much-why/ (accessed Feb. 28, 2023).

[3] writemaps.com, “WriteMaps | Create Sitemaps Online,” *Writemaps.com*, 2019. https://writemaps.com/ (accessed Feb. 28, 2023).

[4] “Balsamiq. Rapid, Effective and Fun Wireframing Software | Balsamiq,” *Balsamiq.com*, 2023. https://balsamiq.com/ (accessed Feb. 28, 2023).

[5] “Flowchart Maker & Online Diagram Software,” *Diagrams.net*, 2023. https://app.diagrams.net/ (accessed Feb. 28, 2023).

[6]“Create powerful business content together with Xtensio,” *Xtensio.com*, Oct. 02, 2022. https://xtensio.com/ (accessed Feb. 28, 2023).