

# Secure Public Transportation Booking System

Report

May 28, 2024

# Contents

<b>1</b>	<b>Overview</b>	<b>2</b>
<b>2</b>	<b>Features</b>	<b>2</b>
2.1	User Authentication . . . . .	2
2.2	Profile Management . . . . .	2
2.3	Booking Management . . . . .	2
2.4	Security Features . . . . .	2
<b>3</b>	<b>System Architecture</b>	<b>2</b>
3.1	Frontend . . . . .	2
3.2	Backend . . . . .	2
<b>4</b>	<b>Use Case Diagram</b>	<b>3</b>
<b>5</b>	<b>Installation</b>	<b>3</b>
5.1	Clone the repository . . . . .	3
5.2	Set up a virtual environment . . . . .	3
5.3	Install dependencies . . . . .	4
5.4	Generate certificate and key . . . . .	4
5.5	Initialize database . . . . .	4
5.6	Add buses and routes (admin) . . . . .	4
5.7	Run the application . . . . .	4
5.8	Access the application . . . . .	4
<b>6</b>	<b>Usage</b>	<b>4</b>
6.1	Add bus and route information (for admin) . . . . .	4
6.2	Run the application . . . . .	5
6.3	Index page . . . . .	5
6.4	Login page . . . . .	6
6.5	Registration page . . . . .	7
6.6	Home page . . . . .	8
6.7	Profile page . . . . .	9
6.8	Edit profile . . . . .	10
6.9	Book tickets . . . . .	10
6.10	View tickets/ Cancel tickets . . . . .	10
<b>7</b>	<b>API endpoints</b>	<b>11</b>
<b>8</b>	<b>Security Measures</b>	<b>13</b>

# 1 Overview

The Secure Public Transportation Booking System is a prototype designed to provide users with a seamless and secure way to book bus tickets online. The system ensures that all user data is handled securely, with robust mechanisms in place to protect sensitive information. Users can use the system to search bus routes and book tickets.

## 2 Features

### 2.1 User Authentication

- **Registration:** Users can register an account with their personal information.
- **Login:** Secure login using username and password.
- **Logout:** Users can securely log out of their session.

### 2.2 Profile Management

- **View Profile:** Users can view their profile details.
- **Edit Profile:** Users can update their personal information such as name, email, and phone number.
- **Delete Account:** Users can delete their account if necessary.

### 2.3 Booking Management

- **Search Buses:** Users can search for available buses based on their departure and destination cities and travel dates.
- **Book Tickets:** Users can book tickets for a selected bus.
- **View bookings:** Users can view their booked ticket information.
- **Cancel Tickets:** Users can cancel their booked tickets, with the system updating seat availability accordingly.

### 2.4 Security Features

- **Session Management:** Session handling with expiry validation to prevent unauthorized access.
- **Data Encryption:** Sensitive user data such as passwords are stored as hashed values to ensure privacy.
- **Input Validation:** All user inputs are validated to prevent SQL injection and other security threats.
- **HTTPS:** System uses HTTPS for secure communication between client and server.

## 3 System Architecture

### 3.1 Frontend

- **HTML/CSS:** For structuring and styling the web pages.
- **JavaScript:** For client-side scripting, AJAX requests, and form handling.

### 3.2 Backend

- **Python (SimpleHTTPServer):** The server handles HTTP requests and serves the necessary data.
- **SQLite:** The database used to store user details, bus information, and booking records.

## 4 Use Case Diagram

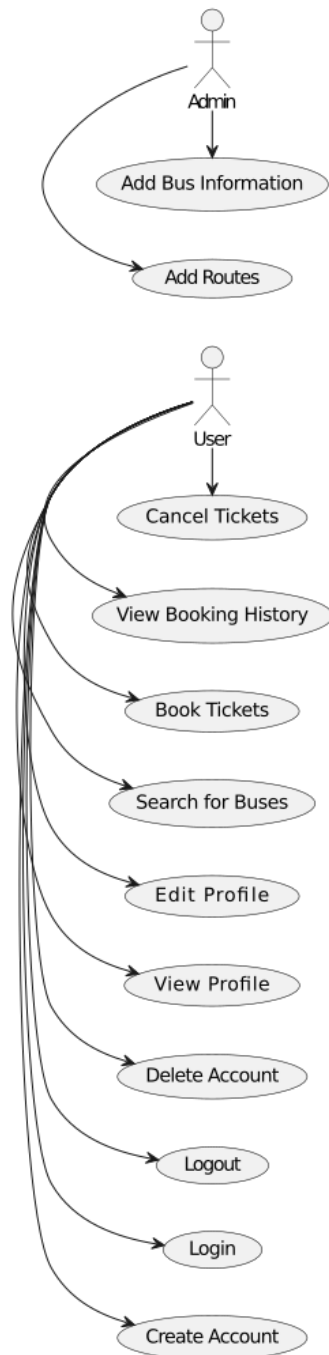


Figure 1: Use case diagram

## 5 Installation

### 5.1 Clone the repository

```
git clone https://github.com/ashtapadhi/SPT-Ticket-Booking.git
```

### 5.2 Set up a virtual environment

```
#setup virtual environment:
```

```
python -m venv venv
```

```
#activate virtual environment:  
source venv/bin/activate
```

```
#On Windows, use:  
venv\Scripts\activate
```

### 5.3 Install dependencies

```
pip install -r requirements.txt
```

### 5.4 Generate certificate and key

```
openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365
```

### 5.5 Initialize database

```
python init_db.py
```

### 5.6 Add buses and routes (admin)

```
#run:
```

```
python admin.py
```

```
#select option from menu and enter details  
#enter 3 once completed
```

### 5.7 Run the application

```
python server.py
```

### 5.8 Access the application

```
#Open a web browser and go to:  
https://localhost:443
```

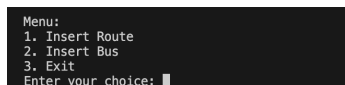
```
#port 443 requires root privileges
```

## 6 Usage

### 6.1 Add bus and route information (for admin)

```
python admin.py
```

Run the above command to start a menu-driven program. Choose the option and proceed to enter

A screenshot of a terminal window showing a menu for the admin program. The menu lists three options: 1. Insert Route, 2. Insert Bus, and 3. Exit. Below the menu, it prompts the user to 'Enter your choice:' followed by a cursor.

```
Menu:  
1. Insert Route  
2. Insert Bus  
3. Exit  
Enter your choice: █
```

Figure 2: Admin program

details. After completing, enter 3 to exit.

## 6.2 Run the application

After running the application, it will ask to enter PEM pass phrase and enter the passphrase given while creating the certificate.

## 6.3 Index page

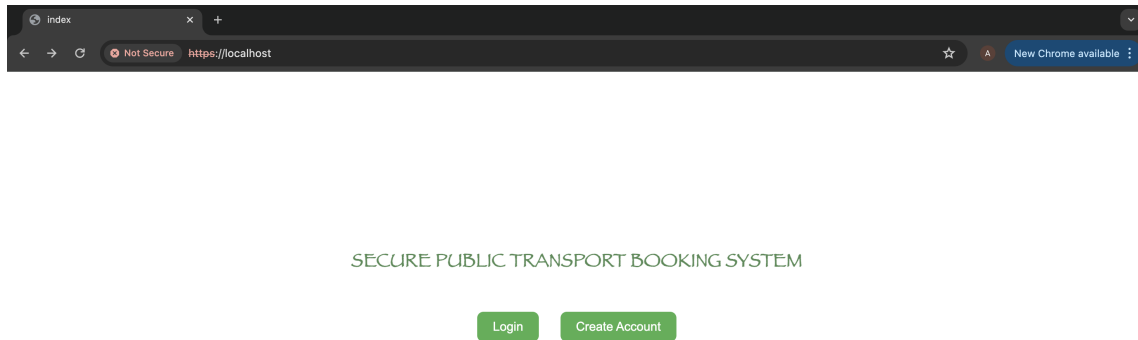


Figure 3: Index page

Users land on the index page initially, where they can either log in if they are existing users or create an account if they are new users.

## 6.4 Login page

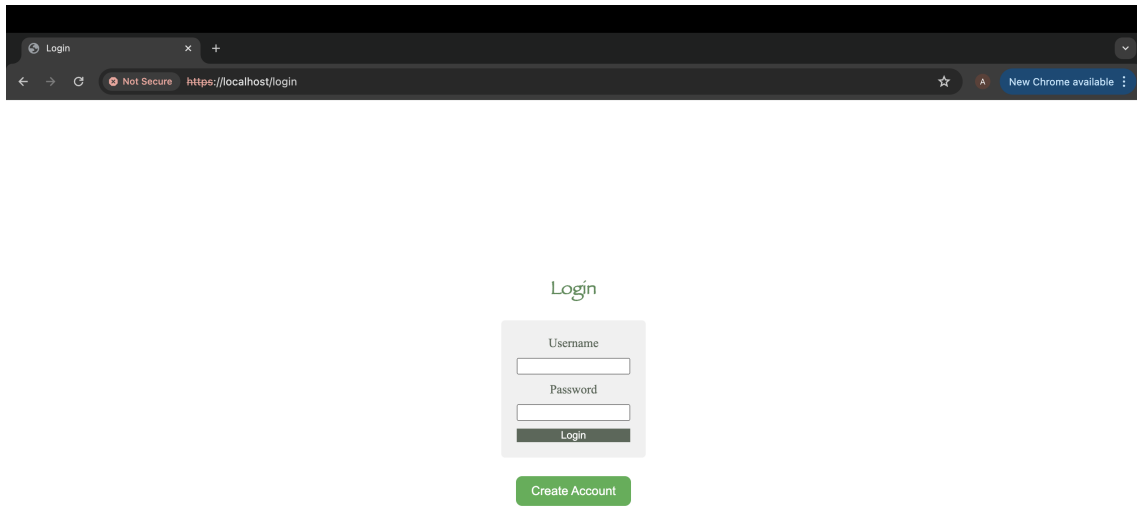
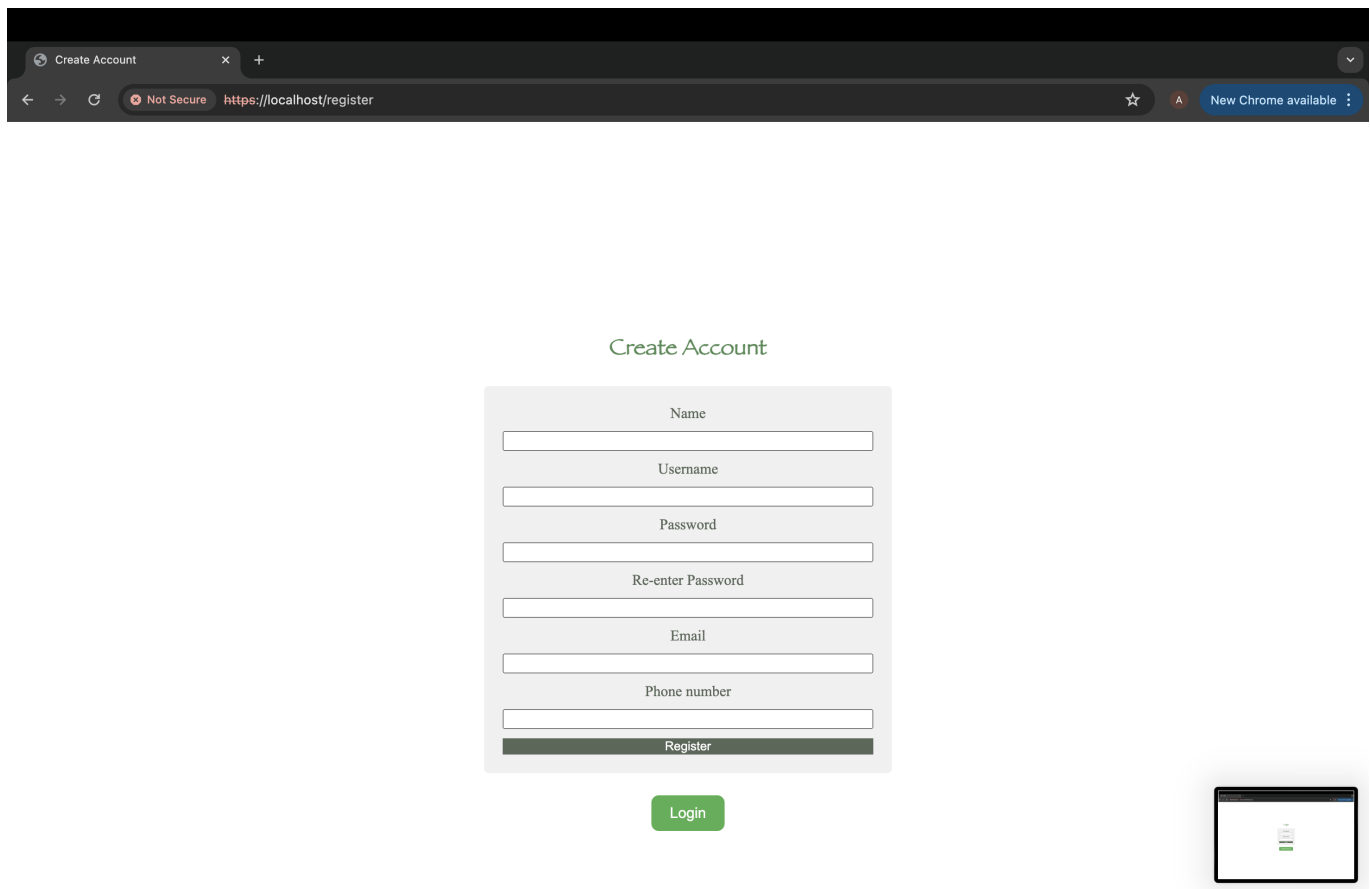


Figure 4: Login page

Users can log in using their username and password.

## 6.5 Registration page



The screenshot shows a web browser window with the title 'Create Account' and the address bar displaying 'https://localhost/register'. The page content features a central registration form with the following fields: Name, Username, Password, Re-enter Password, Email, and Phone number. Below these fields is a 'Register' button. A green 'Login' button is positioned below the 'Register' button. To the right of the main form, there is a small, faint thumbnail of the same page.

Create Account

Name

Username

Password

Re-enter Password

Email

Phone number

Register

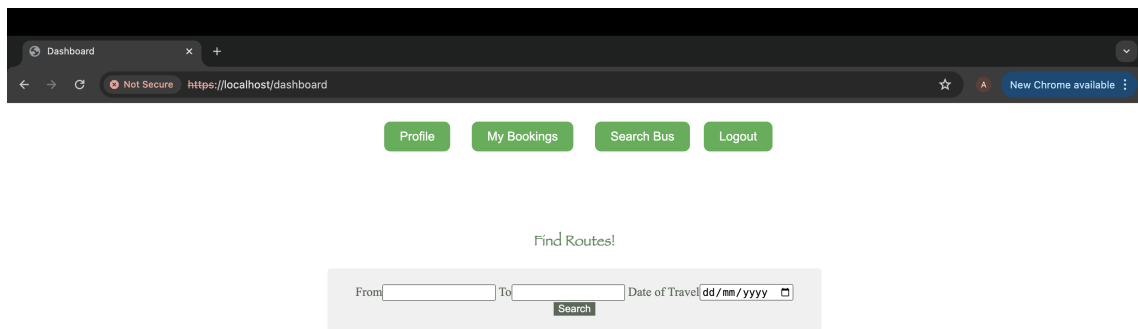
Login

Figure 5: Registration page

Users can create new accounts by providing their details. Each user's username must be unique. If the username provided by a user already exists in the database, an alert will be shown indicating that the username is already taken. Additionally, the password and confirm password fields must match, with the password having a minimum length of 8 characters and including at least one number. The email address should be in a valid format, and the phone number must contain exactly 10 digits.



## 6.6 Home page



The screenshot shows a web browser window with a single tab titled "Dashboard". The address bar displays "Not Secure" and the URL "https://localhost/dashboard". Below the browser window, there is a horizontal navigation bar with four green buttons: "Profile", "My Bookings", "Search Bus", and "Logout". In the center of the page, the text "Find Routes!" is displayed above a search form. The form contains three input fields: "From", "To", and "Date of Travel" with a date picker set to "dd/mm/yyyy". A "Search" button is positioned below the "To" field.

Figure 6: Home page

After successfully logging in, users will be directed to the home page, which offers multiple options. They can visit their profile, search for bus routes using the "from" and "to" locations along with the date, view their booking history, and log out.

## 6.7 Profile page

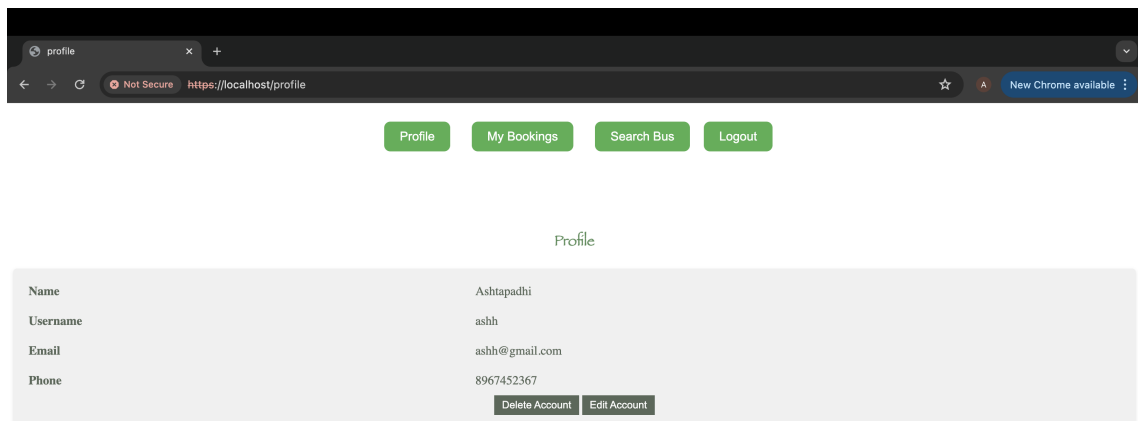


Figure 7: Profile page

On the profile page, users can view their user information. Additionally, there are options available to delete their account or edit their account details.

## 6.8 Edit profile

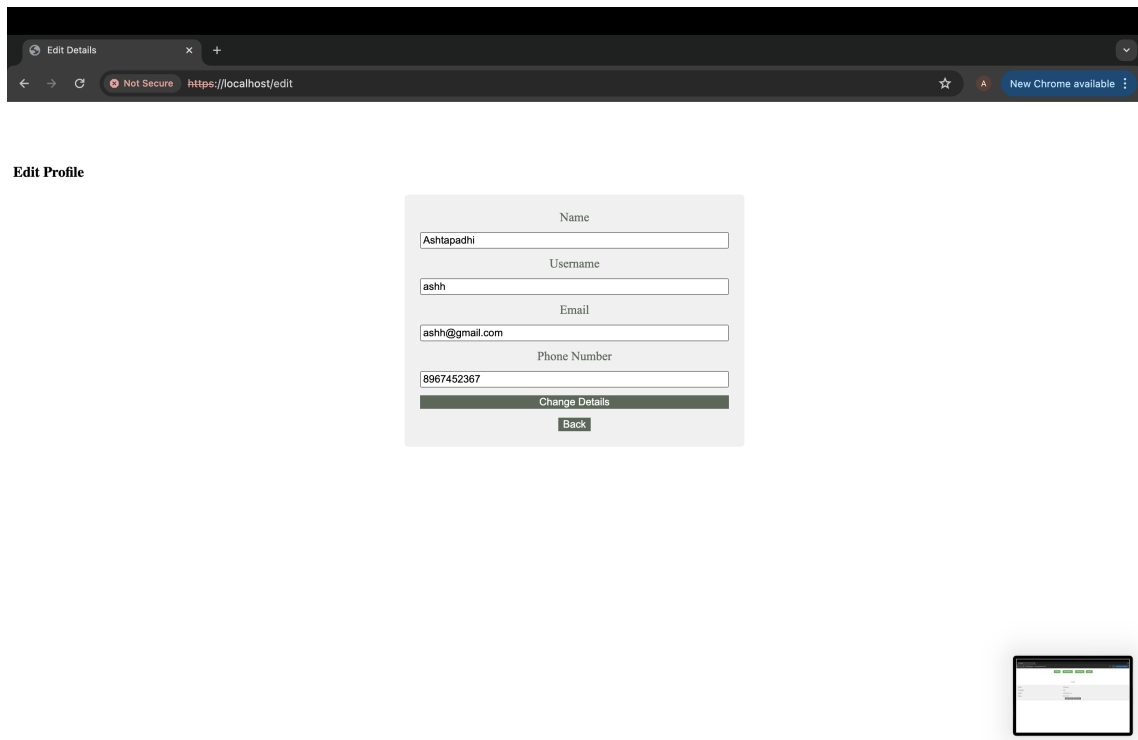


Figure 8 shows a web browser window with the address bar displaying 'https://localhost/edit'. The page title is 'Edit Profile'. The form contains the following fields and values:

Field	Value
Name	Ashtapadhi
Username	ashh
Email	ashh@gmail.com
Phone Number	9967452367

At the bottom of the form, there are two buttons: 'Change Details' and 'Back'.

Figure 8: Edit profile

Users have the ability to modify their details such as their name, username, email, and phone number. If a new username is already in use, an alert will be triggered.

## 6.9 Book tickets



Figure 9 shows a web browser window displaying the 'Book tickets' page. The page contains the following information:

Field	Value
From	Mumbai
To	Pune
Date	2023-10-10
Fare	1000

At the bottom, there is a 'Book' button.

Figure 9: Book tickets

When users enter their travel details, including the "to" and "from" locations and the travel date, on the home page, a list of available buses will be displayed. This list includes details such as the bus name, fare, and available seats. Users can click the "Book" button next to each bus to book tickets for that particular bus. They will then be redirected to the booking page, where the bus information will be available. Users can select the number of seats, and the fare will be automatically calculated. After clicking the "Book" button to finalize their booking, a confirmation alert will notify them of a successful booking, and they will be redirected to their booking history.

## 6.10 View tickets/ Cancel tickets

Users can view their booking history on this page. They also have the option to cancel their tickets if necessary. When they click "Cancel," an alert will appear stating that this action cannot be undone. If they choose to proceed with the cancellation, the specific booking details will be removed from their history, and the availability of seats for the particular bus will be updated accordingly.



Figure 10: View/cancel tickets

## 7 API endpoints

1. **Endpoint:** /login
  - **Method:** POST
  - **Usage:** Authenticates a user with a username and password. If successful, creates a session and returns a session cookie.
2. **Endpoint:** /register
  - **Method:** POST
  - **Usage:** Registers a new user with name, username, password, email, and phone number. Creates a session for the new user and returns a session cookie.
3. **Endpoint:** /buses
  - **Method:** POST
  - **Usage:** Searches for buses based on the provided `from_city`, `to_city`, and `travel_date`. Returns a list of available buses and their details.
4. **Endpoint:** /book
  - **Method:** POST
  - **Usage:** Books a bus ticket for a logged-in user with provided bus details, number of passengers, total fare, and travel date. Updates seat availability accordingly.
5. **Endpoint:** /logout
  - **Method:** POST
  - **Usage:** Logs out the current user by deleting their session. Invalidates the session cookie.
6. **Endpoint:** /
  - **Method:** GET
  - **Usage:** Serves the homepage of the application.
7. **Endpoint:** /login
  - **Method:** GET
  - **Usage:** Serves the login page.
8. **Endpoint:** /register
  - **Method:** GET
  - **Usage:** Serves the registration page.
9. **Endpoint:** /dashboard
  - **Method:** GET
  - **Usage:** Serves the dashboard page for logged-in users. Returns unauthorized error if the user is not logged in.

10. **Endpoint:** /bookpage
  - **Method:** GET
  - **Usage:** Serves the booking page with bus details. Requires `bus_id` and `travel_date` query parameters.
11. **Endpoint:** /profile
  - **Method:** GET
  - **Usage:** Serves the profile page for logged-in users. Returns unauthorized error if the user is not logged in.
12. **Endpoint:** /busdetails
  - **Method:** GET
  - **Usage:** Returns detailed information about a specific bus including available seats. Requires `bus_id` and `travel_date` query parameters.
13. **Endpoint:** /menu.html
  - **Method:** GET
  - **Usage:** Serves the menu page.
14. **Endpoint:** /static
  - **Method:** GET
  - **Usage:** Serves static files like CSS. The exact path should be specified.
15. **Endpoint:** /userdetails
  - **Method:** GET
  - **Usage:** Returns the details of the logged-in user. Returns unauthorized error if the user is not logged in.
16. **Endpoint:** /editdetails
  - **Method:** GET
  - **Usage:** Returns the details of the logged-in user for editing. Returns unauthorized error if the user is not logged in.
17. **Endpoint:** /edit
  - **Method:** GET
  - **Usage:** Serves the edit user details page for logged-in users. Returns unauthorized error if the user is not logged in.
18. **Endpoint:** /mybookings
  - **Method:** GET
  - **Usage:** Returns the booking details of the logged-in user. Returns unauthorized error if the user is not logged in.
19. **Endpoint:** /bookings
  - **Method:** GET
  - **Usage:** Serves the bookings page for logged-in users. Returns unauthorized error if the user is not logged in.

## 8 Security Measures

1. HTTPS: Provides secure communication between the client and server.
2. Password Hashing: Hashes user passwords before storing them in the database.
3. Session Expiry: Checks sessions for expiry time during requests.
4. Data Sanitization: Sanitizes data to remove unwanted elements from the input.
5. Data Validation: Validates data entered by users.
6. Authentication: Users are authenticated before using the system.