CSE471: System Analysis and Design
Project Report
Project Title: All in One Booking System.

Group No: 04, CSE471 Lab Section: 07, Spring 2025	
ID	**Name**
24341069	Md. Irfan Hossain
24341209	Mahinoor Rahman
24341210	Sharmila Rani Saha
22101518	Md. Zobayer

Submission Date: 15/05/2025

Table of Contents

1. System Request 3
    Business need: 3
    Business requirements: 3
    Business value: 3
    Special issues or constraints: 3
2. Functional Requirements 4
3. Technology (Framework, Languages) 5
4. Backend Development 5
5. User Interface Design 12
6. Frontend Development 17
7. User Manual 31
8. Performance and Network Analysis 43
9. Github Repo [Public] Link 48
10. Link of Deployed Project 49
11. Individual Contribution 49
12. References 50

1. System Request

Business need:

The All in One Booking System helps users and businesses one stop service to book hotels, penthouse, transportation, even travel buddy, track bookings, and manage reservations through a centralized platform to make life easy and effective. Many businesses handle bookings manually, which creates confusion, mistakes. This booking system will help users and business owners with automated service without any kind of hassle in one place.

Business requirements:

To make the system easy and user friendly, it must meet the requirements:

- Secure user authentication for profile management and passwords should be stored safely using encryption to keep user data secure.
- Real-time booking management for all available services. This system should allow users to view and modify their booking and admin should be able to manage booking, editing and canceling reservations.
- The system should be able to handle increasing numbers of users and bookings. It should be flexible enough to accommodate future updates and new features.
- There should be a responsive and user-friendly interface. Users should be able to easily reach out for help for any issues and admins should be able to respond to customer inquiries.

Business value:

This All in one Booking System will provide many benefits to businesses and in daily life.. This system will make the booking processes easy to use for businesses and customers, improving efficiency and reducing human errors. It provides a competitive advantage with a centralized system that focuses more on customer service. This simple and easy-to-use booking system will make customers happy, leading to more bookings and repeat customers. Businesses can attract more people to use their services, ultimately increasing sales. The system can handle more bookings and users without any problems. The platform will also be able to add new features as customer needs change.

Special issues or constraints:

While building the system, there are a few challenges to consider:

- The system needs to be easy to use and keep things simple. If it is too complicated, customers will not use it.
- The system needs to ensure data security and user privacy. Since users will share personal information, the system must follow privacy rules to protect that information.
- After the system is built, it will need ongoing maintenance to fix bugs, improve features, and keep everything secure. Customer support should also be available to handle any issues that arise.

2. Functional Requirements

Module 1: System Setup & Admin Management

1. Admins can manage the entire branch network by adding new branches, updating existing ones, or removing outdated entries.

2. Admins can create bus routes and assign multiple buses with varied departure times to the same route for better service.

3. Admins can post rental cars of different types, sizes, and seating capacities, allowing users to select vehicles as per their needs.

4. Admins can add or remove hotel rooms of various types, which are then visible on the user dashboard.

5. Admins also have the ability to add penthouses, and these listings appear directly on the user dashboard.

6. The admin panel includes access to dashboard controls, profile management, and system display settings.

Module 2: User Access, Booking & Interaction

1. Users can sign up, log in, log out, and gain access to a personalized dashboard.

2. Users can manage and update profile details and see their information displayed on the profile page.

3. Users can access a flexible transportation system with both bus and car options, view schedules, and easily print their tickets online.

4. Users can book a wide range of hotel rooms as per their preference.

5. Users have the ability to book any available penthouse accommodation.

Module 3: Advanced Admin Controls & Communication

1. Any user can create and publish travel buddy posts—either solo or as a group.

2. Admins can oversee and manage all travel buddy posts and penthouse listings from the backend.

3. Admins can approve or reject user transportation bookings, including both car and bus orders.

4. Admins can view all registered users, delete accounts if needed, and assign or update admin privileges.

5. Users can reach out through a “Contact Us” page and use a real-time chat box feature for support.

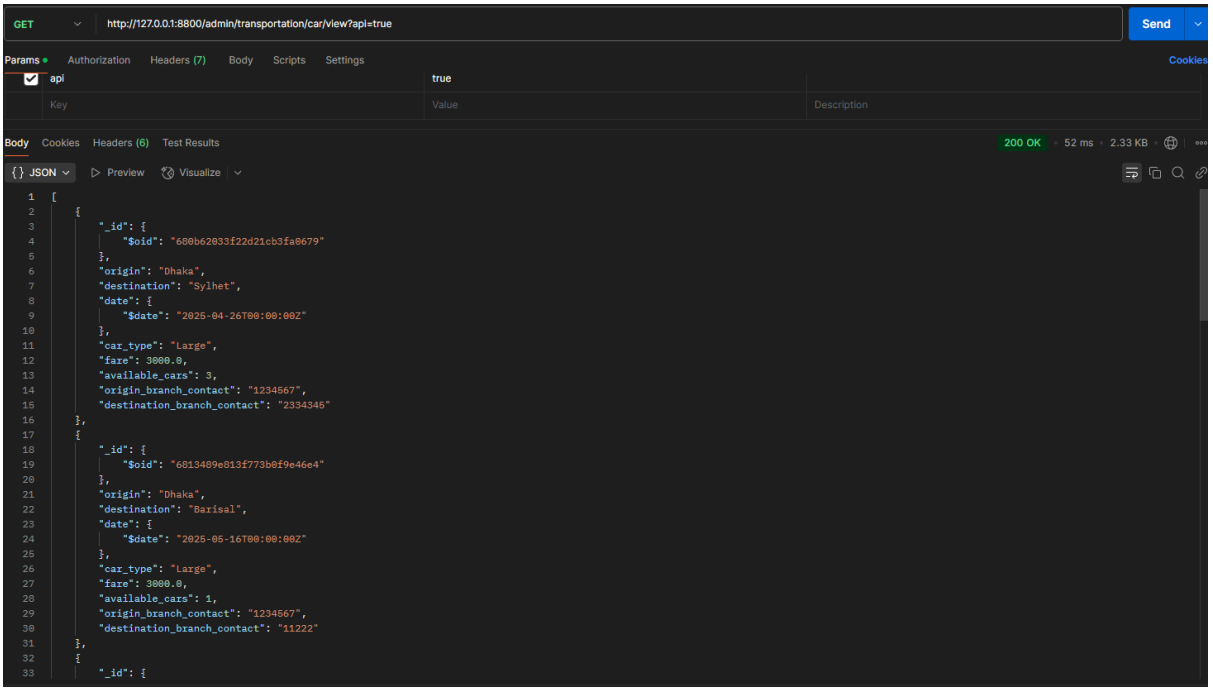
3. Technology (Framework, Languages)

- Backend: Flask (Python)
- Frontend: HTML, CSS, JavaScript
- Database: MongoDB
- Frameworks: Flask

4. Backend Development

Api 1: Get method, View Car routes from admins end.

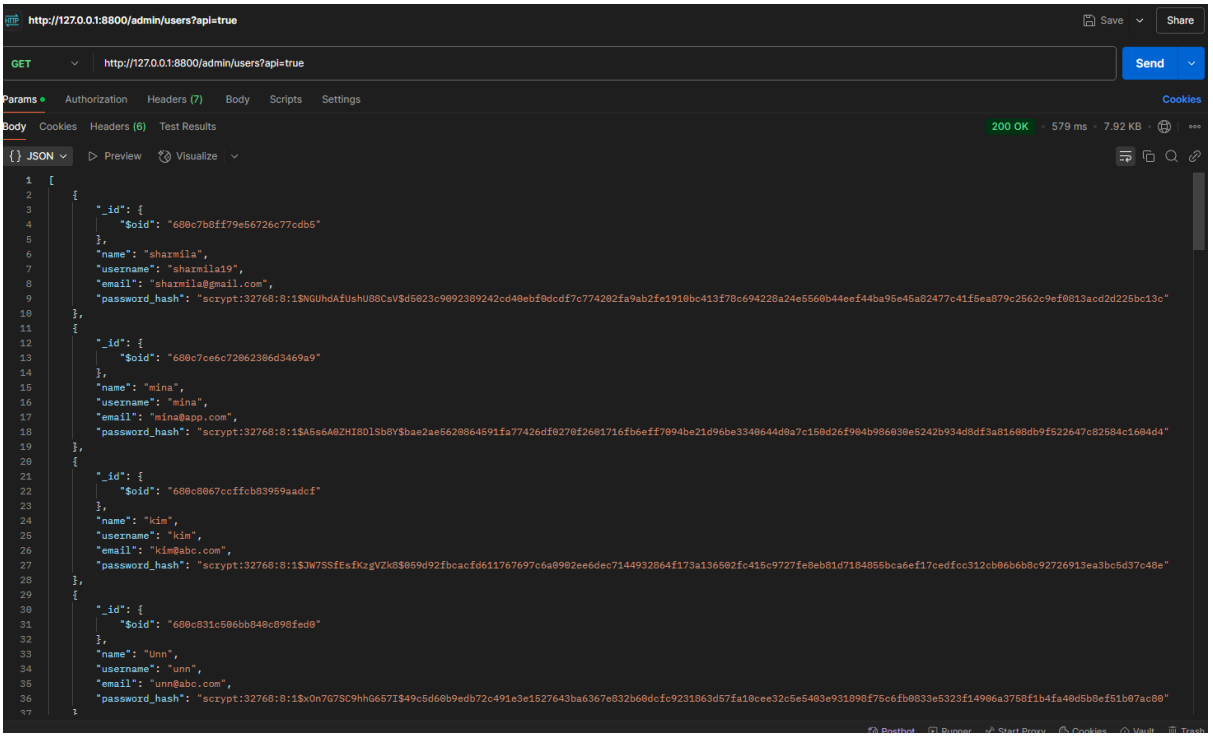
```python
# View Car Routes

@app.route("/admin/transportation/car/view", methods=["GET"])
def view_car_routes():
    car_routes = list(mongo.db.car_routes.find())  # convert cursor to list

    # Check for API usage (Postman, etc.)
    if request.headers.get("Accept") == "application/json" or request.args.get("api") == "true":
        return dumps(car_routes), 200  # return JSON

    # Else, render HTML
    return render_template("admin/transportation/car/view_car_routes.html", car_routes=car_routes)
    You, 2 weeks ago • car
```



The view_car_routes function allows admins to see a full list of all added car routes. It retrieves all entries from the car_routes collection in MongoDB and renders them in the view_car_routes.html template. Each route displays key details such as origin, destination, travel date, car type, fare, and number of available cars. It also includes contact information for both the origin and destination branches, making it easier for admins to coordinate and manage route logistics. This view helps in monitoring car rentals and updating route availability as needed.

Api 2: Get method, Admin can View Users list.

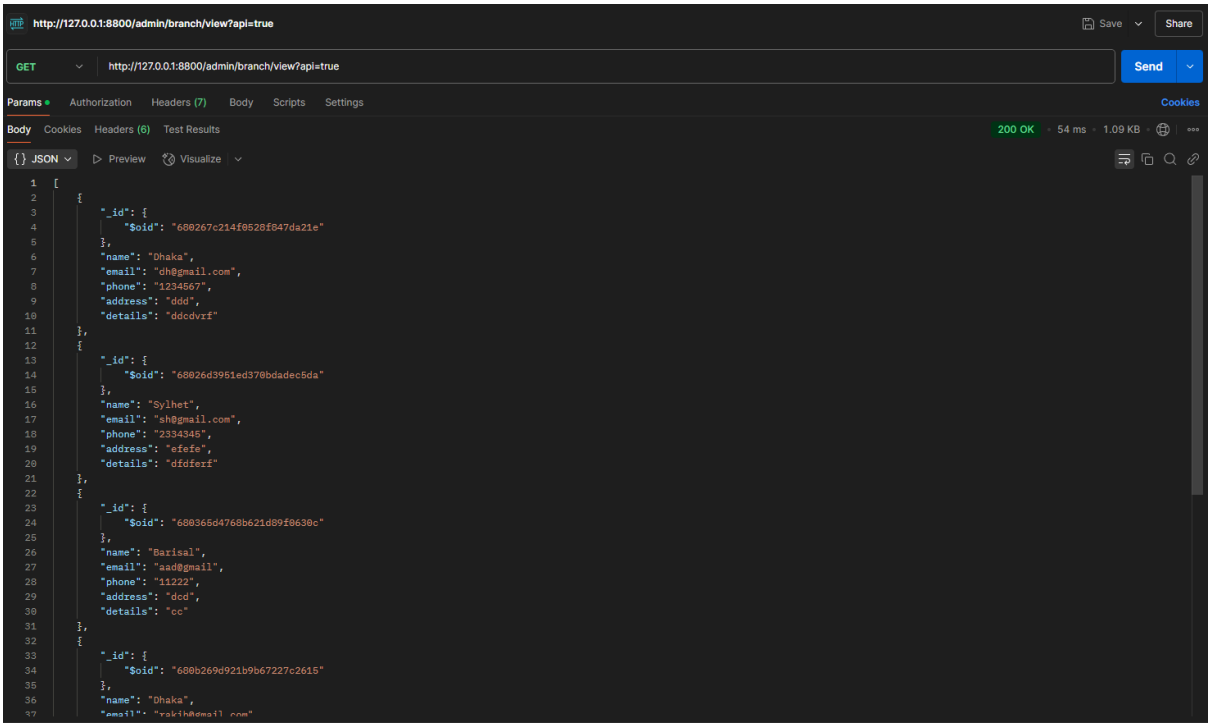
```python
# ------------------- View Users -------------------
@app.route('/admin/users', methods=["GET"])
def view_users():
    users = list(mongo.db.users.find())

    # Check if request is from API client (Postman or frontend)
    if request.headers.get("Accept") == "application/json" or request.args.get("api") == "true":
        return dumps(users), 200

    # Otherwise return the HTML page
    return render_template('admin/view_users.html', users=users)      You, 3 weeks ago • adm
```



The view_users function is responsible for displaying all registered users in the system to the admin. It fetches all documents from the users collection in MongoDB and passes the data to the view_users.html template. In the interface, each user is shown in a tabular format with key information such as Name, Nickname, Hash Password and Email.

Api 3: Get Method, view Branches from Admins end.

```python
@app.route("/admin/branch/view")
def view_branches():
    # Allow API access without session, if requested
    if request.headers.get("Accept") == "application/json" or request.args.get("api") == "true":
        branches = list(mongo.db.branches.find())
        return dumps(branches), 200

    # For browser access, require login session
    if "admin_id" not in session:
        flash("Login required.", "danger")
        return redirect(url_for("admin_login"))

    branches = list(mongo.db.branches.find())
    return render_template("admin/branch/view_branches.html", branches=branches)
```



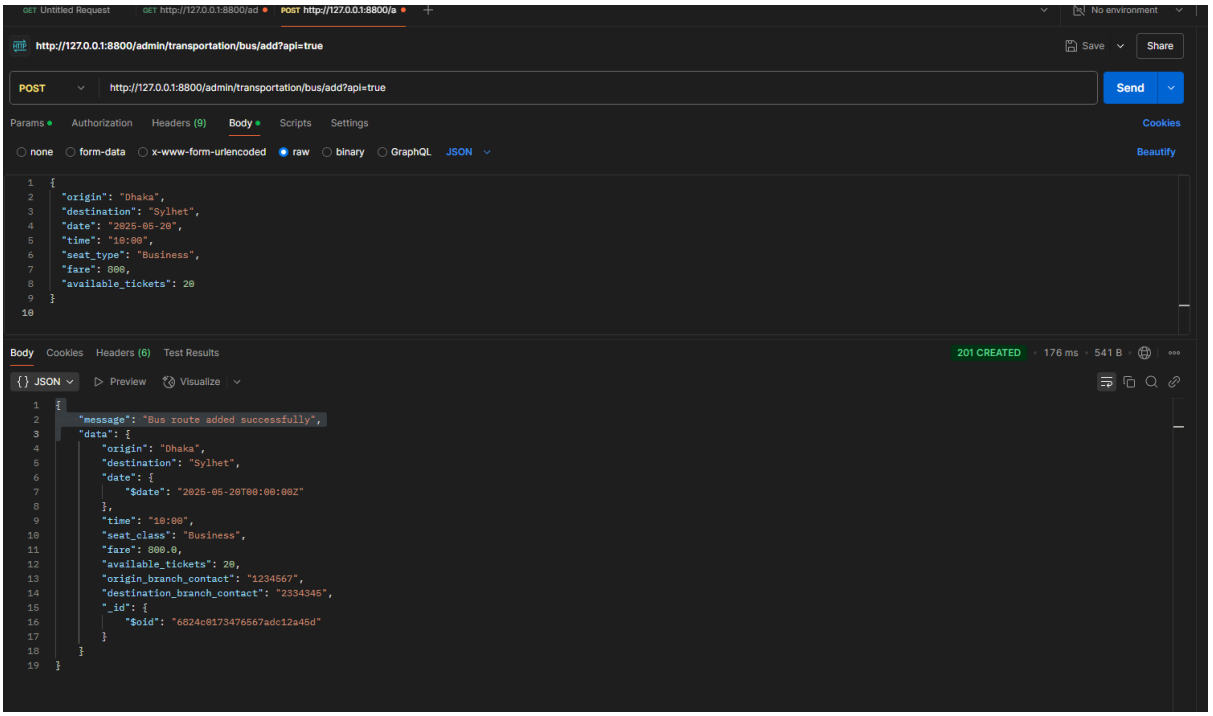
The view_branches function displays a detailed list of all branches that have been added by the admin. Each branch entry includes essential information such as the branch name (which corresponds to a division of Bangladesh), branch email, contact phone number, physical address, and optional details or description provided during branch creation. This view helps the admin keep track of all operational or service points from where transportation services originate or terminate, ensuring accurate route and booking associations.

Api 4: Post Method, Add bus routes from Admin.

```python
# ------------ Transportation System - Bus Routes ------------


@app.route("/admin/transportation/bus/add", methods=["GET", "POST"])
def add_bus_route():
    is_api = request.headers.get("Content-Type") == "application/json" or request.args.get("api") == "true"

    # Check session only for web form access
    if not is_api and "admin_id" not in session:
        flash("Login required.", "danger")
        return redirect(url_for("admin_login"))

    if request.method == "POST":
        if is_api:
            data = request.get_json()
            origin = data.get("origin")
            destination = data.get("destination")
            date = data.get("date")
            time = data.get("time")
            seat_class = data.get("seat_type")
            fare = data.get("fare")
            available_tickets = data.get("available_tickets")
        else:
            origin = request.form.get("origin")
            destination = request.form.get("destination")
            date = request.form.get("date")
            time = request.form.get("time")
            seat_class = request.form.get("seat_type")
            fare = request.form.get("fare")
            available_tickets = request.form.get("available_tickets")
```

```python
        # Basic validations
        if not all([origin, destination, date, time, seat_class, available_tickets]):
            if is_api:
                return jsonify({"error": "Missing required fields"}), 400
            flash("All fields except fare are required.", "danger")
            return redirect(url_for("add_bus_route"))

        # Auto-fare
        if not fare:
            fare = {"Economy": 400, "Business": 800, "Regular": 600}.get(seat_class)
            if not fare:
                if is_api:
                    return jsonify({"error": "Invalid seat class."}), 400
                flash("Invalid seat class.", "danger")
                return redirect(url_for("add_bus_route"))

        origin_branch = mongo.db.branches.find_one({"name": origin})
        destination_branch = mongo.db.branches.find_one({"name": destination})

        if not origin_branch or not destination_branch:
            if is_api:
                return jsonify({"error": "Origin or destination branch does not exist."}), 404
            flash("Origin or destination branch does not exist.", "danger")
            return redirect(url_for("add_bus_route"))

        bus_data = {
            "origin": origin,
            "destination": destination,
            "date": datetime.strptime(date, "%Y-%m-%d"),
            "time": time,
```

```
        "seat_class": seat_class,
        "fare": float(fare),
        "available_tickets": int(available_tickets),
        "origin_branch_contact": origin_branch.get("phone"),
        "destination_branch_contact": destination_branch.get("phone")
    }

    mongo.db.bus_routes.insert_one(bus_data)

    if is_api:
        return jsonify({"message": "Bus route added successfully", "data": bus_data}), 201

    flash("Bus route added successfully.", "success")
    return redirect(url_for("view_bus_routes"))
```



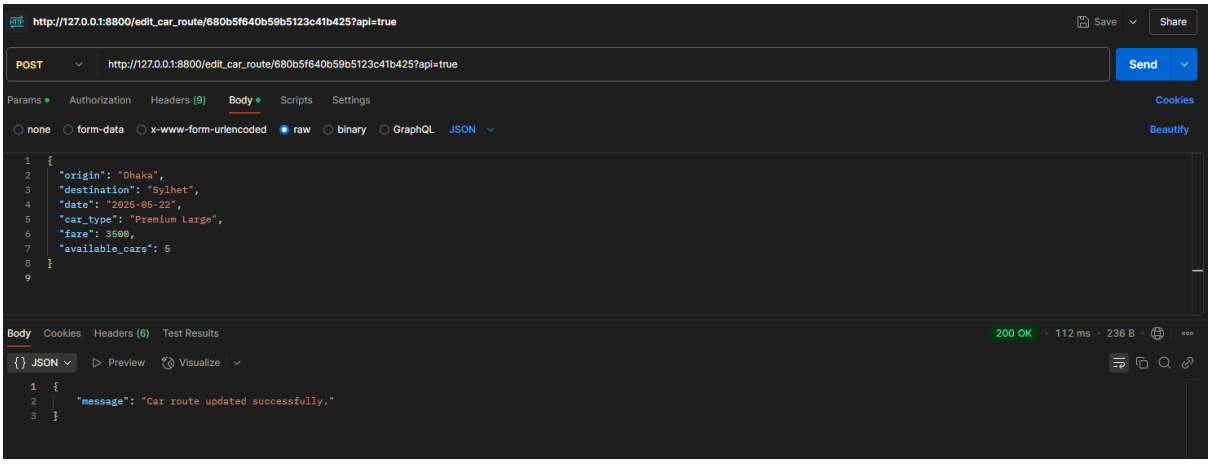
The POST method in the add_bus_route route processes both web form submissions and API requests with JSON input. When a JSON body is sent—containing fields like origin, destination, date, time, seat type, fare, and available tickets—it first checks that all required fields are present. If any are missing (except fare), it returns an error. If fare is omitted, the system automatically assigns it based on the seat type (e.g., Business = 800). It then verifies that both the origin and destination branches exist in the database. Once validated, it creates a new bus route record in the bus_routes collection with the provided details and returns a success response if it's an API call or redirects to the bus route list page if it's a web request.

Api 5: Post method, Edit Car Routes

```python
@app.route('/edit_car_route/<route_id>', methods=['GET', 'POST'])
def edit_car_route(route_id):
    route = mongo.db.car_routes.find_one({'_id': ObjectId(route_id)})

    if request.method == 'POST':
        # Support both API (JSON) and form POST
        if request.args.get("api") == "true":
            data = request.get_json()
            origin = data.get('origin')
            destination = data.get('destination')
            date = data.get('date')
            car_type = data.get('car_type')
            fare = data.get('fare')
            available_cars = data.get('available_cars')
        else:
            origin = request.form['origin']
            destination = request.form['destination']
            date = request.form['date']
            car_type = request.form['car_type']
            fare = request.form['fare']
            available_cars = request.form['available_cars']

        mongo.db.car_routes.update_one(
            {'_id': ObjectId(route_id)},
            {'$set': {
                'origin': origin,
                'destination': destination,
                'date': datetime.strptime(date, '%Y-%m-%d'),
                'car_type': car_type,
                'fare': float(fare),
                'available_cars': int(available_cars)
```

```python
                'available_cars': int(available_cars)
            }}
        )

        if request.args.get("api") == "true":
            return jsonify({"message": "Car route updated successfully."}), 200

        flash("Car route updated successfully.", "success")
        return redirect(url_for('view_car_routes'))

    branches = mongo.db.branches.find()
    return render_template('admin/transportation/car/edit_car_route.html', route=route, branches=branches)
```

You, 2 weeks ago • car …

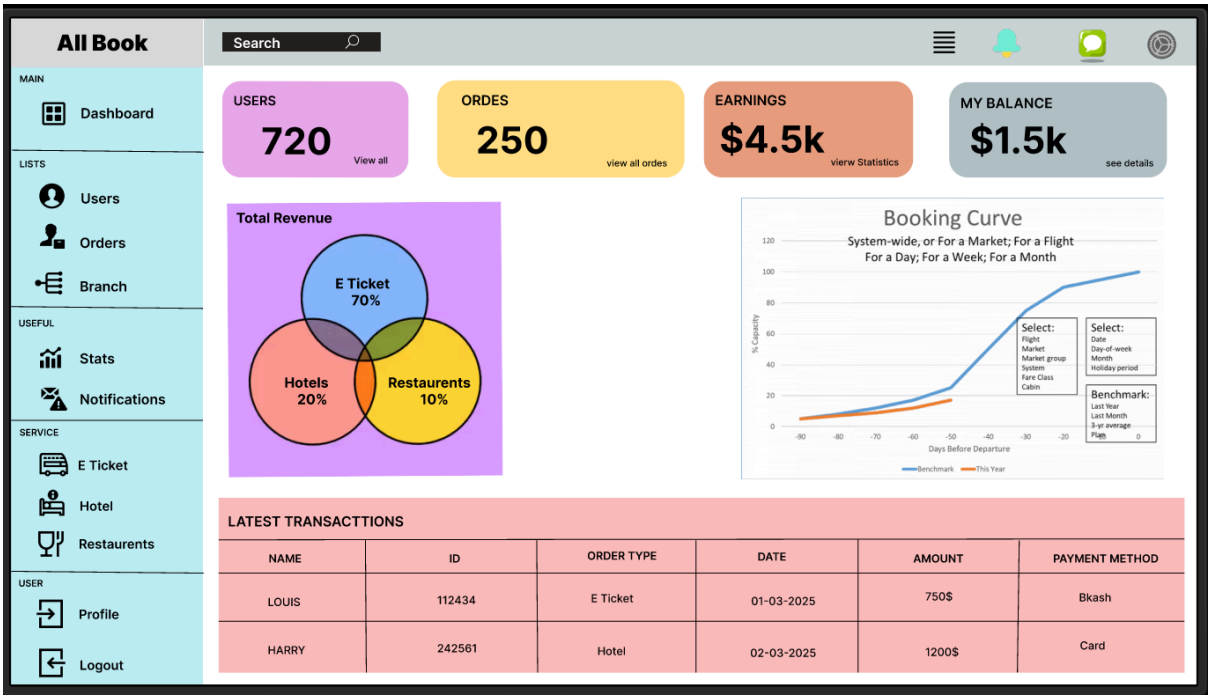


The POST method in the edit_car_route route handles both form-based submissions and API requests for updating an existing car route. When the request is from an API (indicated by the query parameter ?api=true), it reads JSON input containing updated fields like origin, destination, date, car type, fare, and available cars. For form-based submissions, it collects these values from request.form. After parsing the inputs, it updates the corresponding document in the car_routes collection using the route's unique ID. The date is converted into a datetime object, and the fare and available cars are cast to float and integer respectively. On success, the API response returns a JSON confirmation, while form-based requests display a flash message and redirect the admin to the car routes list page.

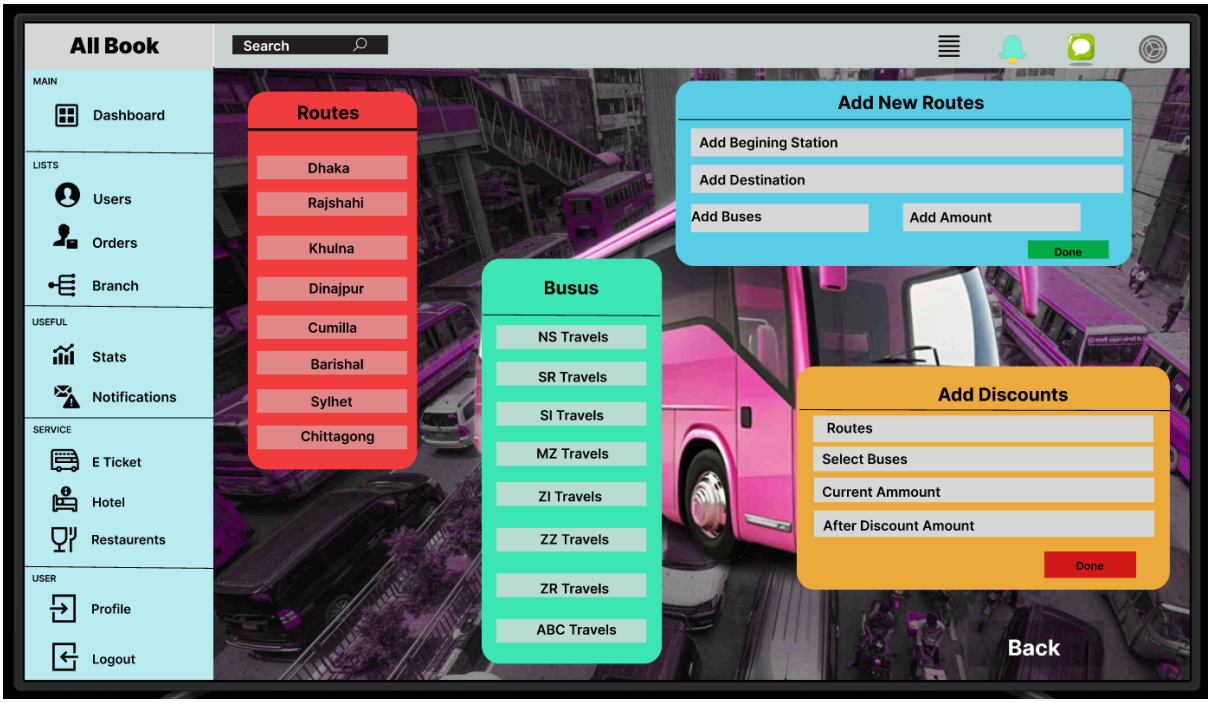
5. User Interface Design

Attach at least 5 interface designs generated in Figma. Attach the figma project link.

Admin view:

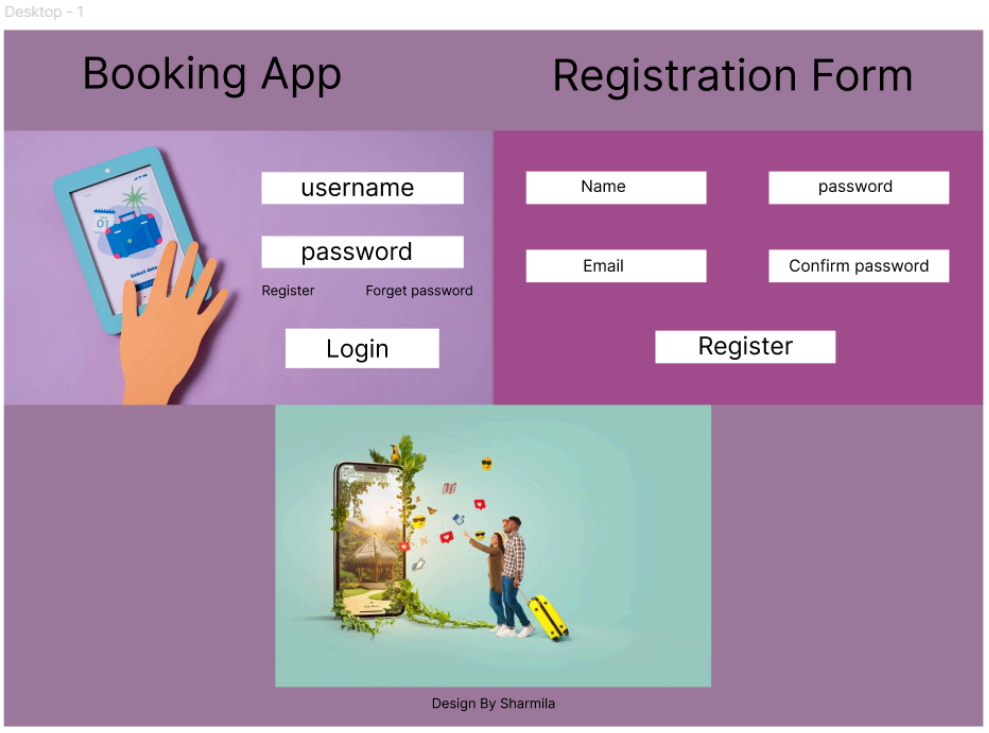


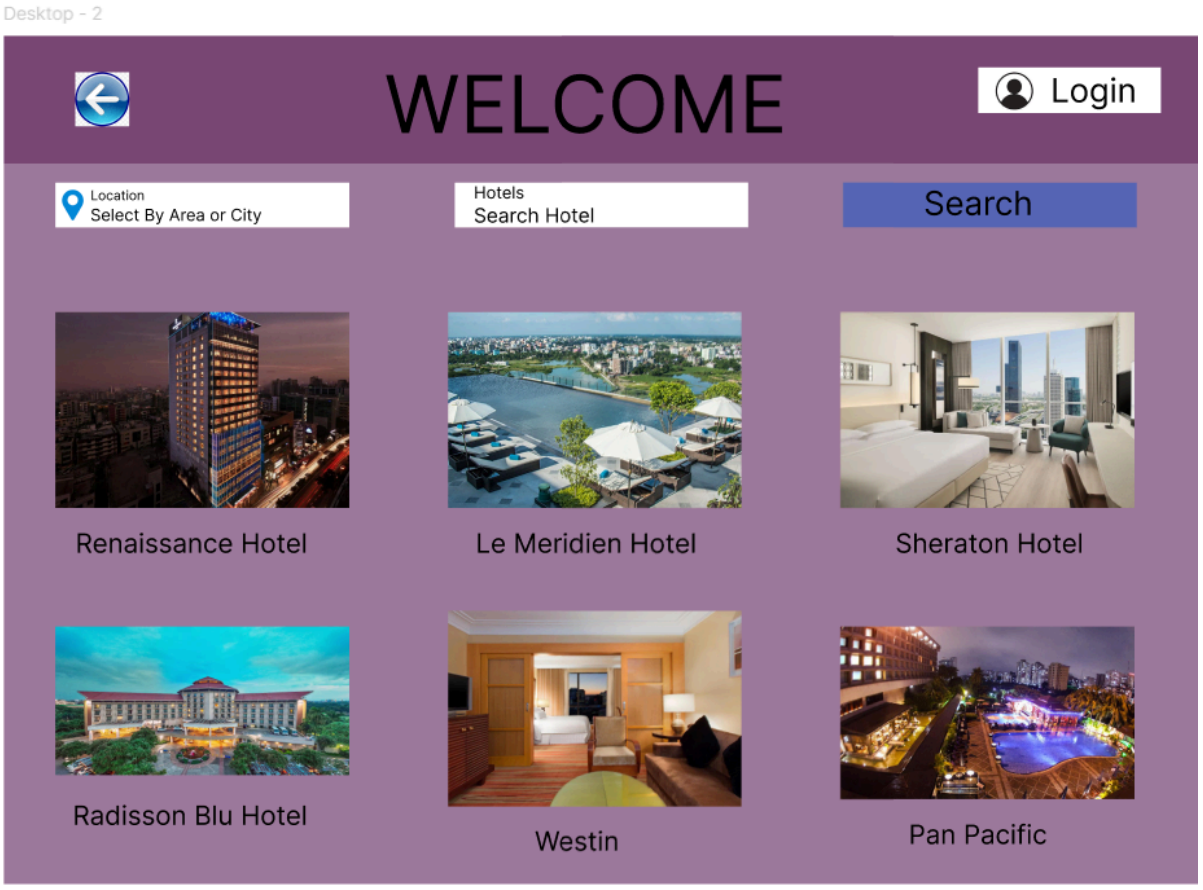
Add bus routes:



https://www.figma.com/proto/PZ36bNL49A6HRpsK4lIWSZ/Zobayer_AllBook?node-id=1-2&t=el0KxreVDJhBupxs-1

We can see the login and registration option. After clicking the login button we will navigate to next page where we can see the hotels. We can see the hotel selection option, and can also choose an area or city. Al By clicking the back logo we can go back.





Figma prototype file link:
https://www.figma.com/proto/zzL72A2DDtpNlsBSCJDayP/24341210_Sharmila-Rani-Saha?node-id=7-32&p=f&t=CaNU6yvY4JOwqh0V-0&scaling=min-zoom&content-scaling=fixed&page-id=0%3A1&starting-point-node-id=7%3A32

We can see the different options of the tickets. We can check the hotel room options, Transportation tickets options and also can find potential travel buddies. We can see the room options, transportation ticket options and travel buddy option.

Desktop - 1

Desktop - 2

Room Options

Single Room | Double Room | Junior Suits | Delux Room | Studio Room

Penthouse

Duplex Penthouses | Triplet Penthouses | Luxury Penthouses

Book Tickets

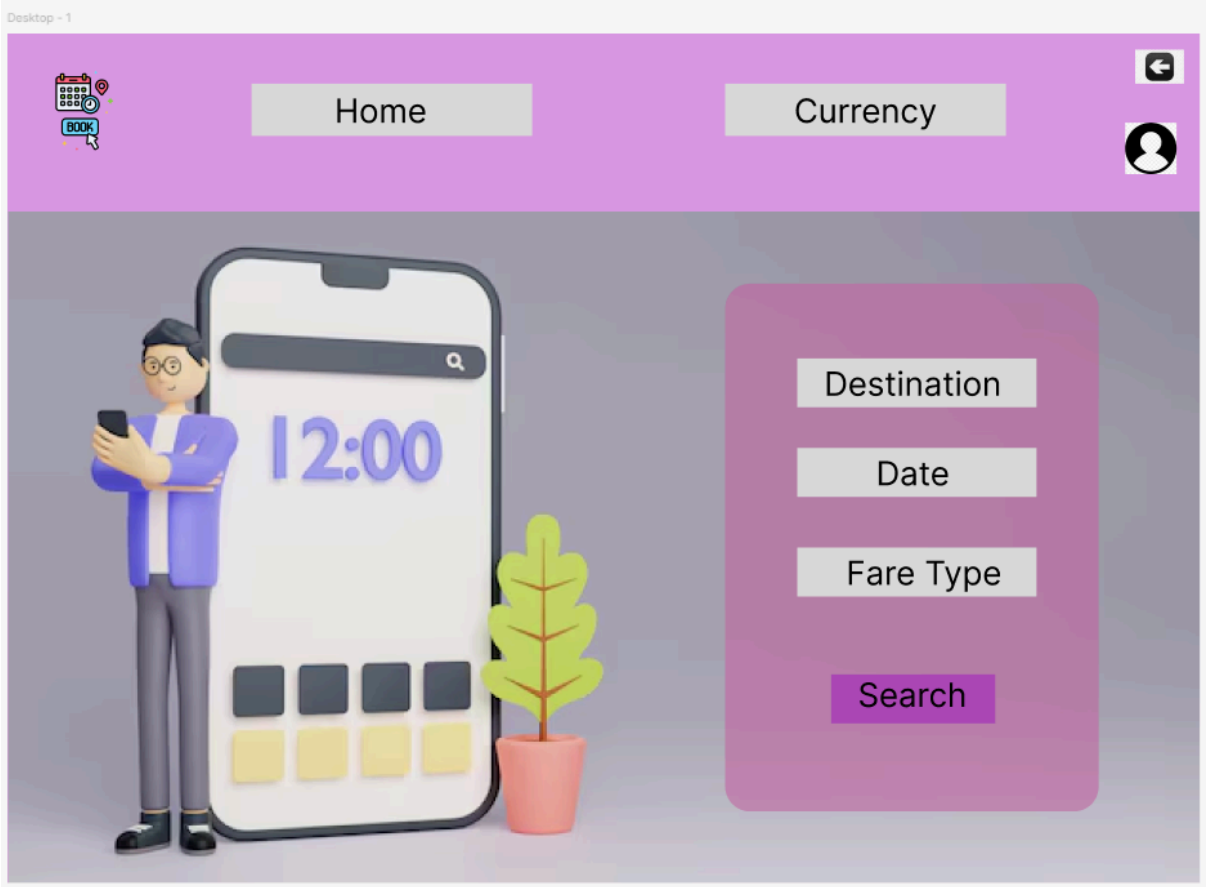
Bus Tickets | Train Tickets | Plane Tickets

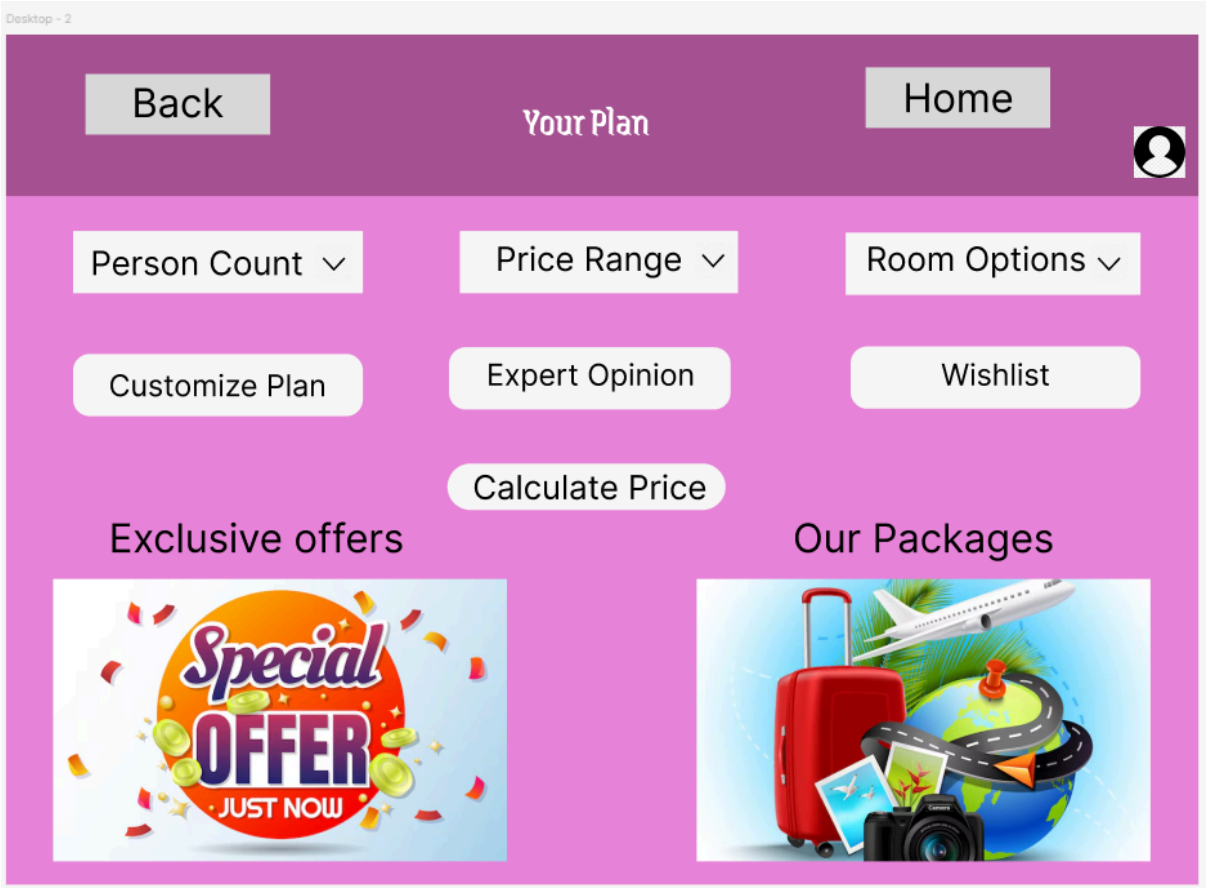
Find a Travel Buddy

Single Buddy | Group buddies

Prototype link:
https://www.figma.com/proto/iOwGOBcTmFHn6xJFSZpV3h/All-in-One-Booking-System?node-id=1-2&p=f&t=5Q5fWxRIAt2MdOou-1&scaling=min-zoom&content-scaling=fixed&page-id=0%3A1&starting-point-node-id=1%3A2

There are three options of select destination, date and fare type. Person count, price range and room options dropdown will let us select more precise options for our searching. Customize plan (book options) , expert opinion, wishlist option button here for more customization and help to find anything easily. Calculate price will give a price suggestion of our selection. Exclusive offers and our packages will provide offer suggestions.





Figma prototype link:
https://www.figma.com/proto/Z07VdtpWpK3Lgcmxp8DBIi/Mahinoor?node-id=8-16&p=f&t=7pSNhZPZEcbyNTAf-0&scaling=min-zoom&content-scaling=fixed&page-id=8%3A15&starting-point-node-id=8%3A17

6. Frontend Development

Attach at least 5 frontend code snippets with proper description.

1. User Messages:

This HTML template is designed to display a list of users who have sent messages in the system. The list shows their usernames, the timestamp of their last message, and a link to start a chat with them. This page provides an interface for admins to manage messages from users. It lists the users who have sent messages, showing a preview of the last message along with the time it was sent. Features:

- The admin can click on a user's name to start a conversation.
- If there are no users, the message "No users have messaged yet." is shown.
- The design is responsive, ensuring it looks good on all devices.

This layout allows admins to easily view and respond to user messages in a clean and efficient manner.

templates > admin > messages.html > ...

```html
<!DOCTYPE html>
<html>
<head>
    <title>User Messages</title>
    <style>
        body { font-family: Arial, sans-serif; background: #f4f4f4; padding: 20px; }
        .container { max-width: 700px; margin: auto; background: white; padding: 20px; border-radius: 10px; box-shadow: 0 0 10px #ccc; }
        .user-entry { border-bottom: 1px solid #ccc; padding: 10px 0; }
        .user-entry:last-child { border: none; }
        .user-entry a { text-decoration: none; color: #007bff; font-weight: bold; }
        .timestamp { color: gray; font-size: 0.85em; }
    </style>
</head>
<body>
    <div class="container">
        <h2>Users Who Messaged</h2>
        {% for user in users %}
            <div class="user-entry">
                <a href="{{ url_for('admin_chat', user_id=user.user_id) }}">{{ user.username }}</a><br>
                <span class="timestamp">{{ user.timestamp.strftime('%Y-%m-%d %H:%M') }} - "{{ user.last_message }}"</span>
            </div>
        {% else %}
            <p>No users have messaged yet.</p>
        {% endfor %}
    </div>
</body>
</html>

```

2. Edit Car Route:

This code interface is for managing car routes, where the admin can view and update route details in a well-structured form. The form is submitted using the POST method, allowing the updated route details to be sent to the server for processing. The form contains several input fields that allow admins to edit different attributes of a car route:

- Origin and Destination: Admin can select from a list of available locations. The options are dynamically populated from a server-side list (branches).
- Date: The date input field allows the user to select or modify the route's date.
- Car Type: A dropdown menu to select the type of car (small, large, or extra-large).
- Fare: A dropdown menu for selecting the fare of the route, with predefined options (2000, 3000, or 4000 BDT).
- Available Cars: A numeric input to define the number of cars available for the route.

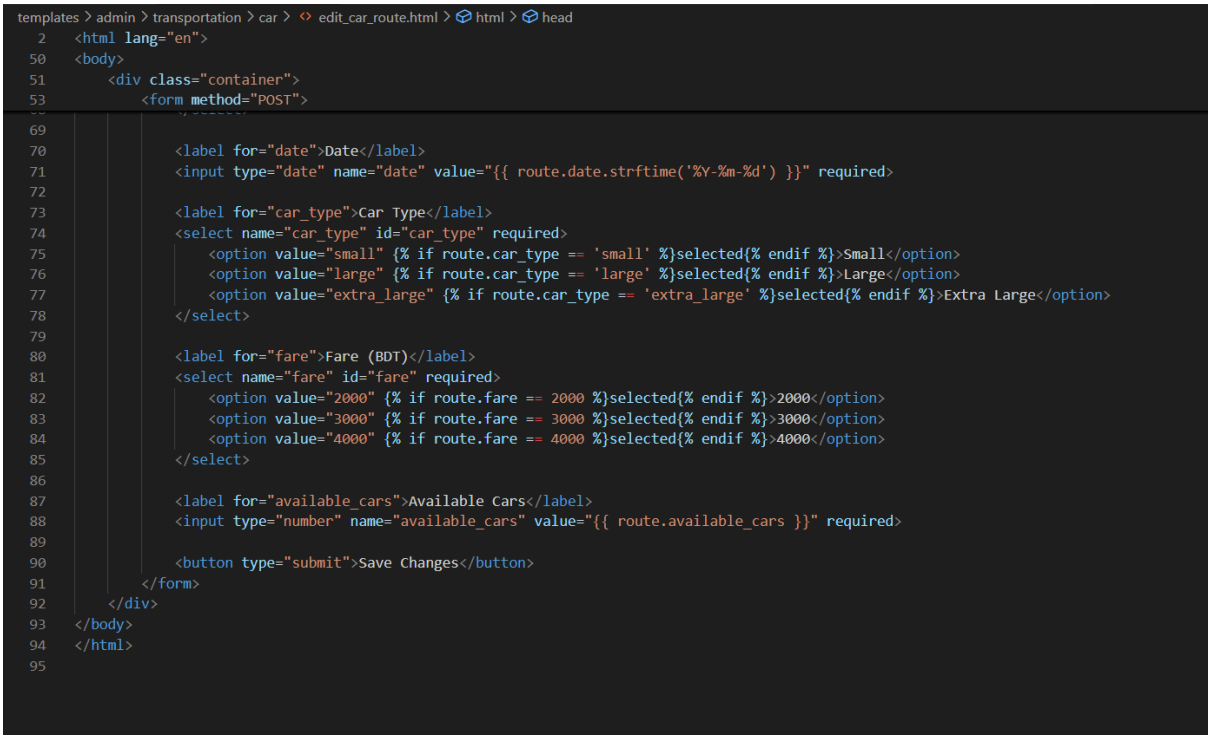
We can save changes to update the data in the database.

templates > admin > transportation > car > edit_car_route.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Edit Car Route</title>
    <style>
        body {
            font-family: sans-serif;
            background: #f0f2f5;
        }
        .container {
            max-width: 600px;
            background: #fff;
            margin: 40px auto;
            padding: 30px;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0,0,0,0.1);
        }
        h2 {
            color: #ff6600;
            text-align: center;
        }
        label {
            display: block;
            margin-top: 15px;
            font-weight: bold;
        }
        select, input[type="number"], input[type="date"] {
            width: 100%;
            padding: 10px;
            margin-top: 5px;
            border-radius: 5px;
            border: 1px solid #ccc;
        }
        button {
            margin-top: 20px;
            background: #ff6600;
```

templates > admin > transportation > car > edit_car_route.html > html > head > style

```
<html lang="en">
<head>
    <style>
        button {
            background: #ff6600;
            color: white;
            border: none;
            padding: 12px;
            width: 100%;
            font-size: 16px;
            border-radius: 5px;
        }
        button:hover {
            background: #e55d00;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Edit Car Route</h2>
        <form method="POST">
            <label for="origin">Origin</label>
            <select name="origin" id="origin" required>
                <option value="" disabled>Select origin</option>
                {% for branch in branches %}
                    <option value="{{ branch.name }}" {% if branch.name == route.origin %}selected{% endif %}>{{ branch.name }}</option>
                {% endfor %}
            </select>

            <label for="destination">Destination</label>
            <select name="destination" id="destination" required>
                <option value="" disabled>Select destination</option>
                {% for branch in branches %}
                    <option value="{{ branch.name }}" {% if branch.name == route.destination %}selected{% endif %}>{{ branch.name }}</option>
                {% endfor %}
            </select>
```



3. Manage Travel Buddies:

This is a travel buddy management interface where the admin can approve or reject travel buddy posts. It displays both single buddy and group requests, showing their destination, details, and status.

templates > admin > manage_buddies.html > ...

```
<!DOCTYPE html>
<html>
<head>
    <title>Manage Travel Buddy Posts</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        body {
            background-color: #f8f9fa;
            font-family: Arial, sans-serif;
        }

        .card {
            border-radius: 15px;
            box-shadow: 0px 4px 12px rgba(0, 0, 0, 0.1);
        }

        .card-title {
            font-weight: bold;
            color: #333;
        }

        .card-text {
            color: #555;
        }

        .btn-custom {
            padding: 8px 16px;
            border-radius: 25px;
            font-weight: bold;
            transition: all 0.3s ease;
        }

        .btn-approve {
            background-color: #28a745;
            border: none;
        }
```

templates > admin > manage_buddies.html > html > head > style > .card-text

```
<html>
<head>
    <style>
        }

        .btn-approve:hover {
            background-color: #218838;
        }

        .btn-reject {
            background-color: #dc3545;
            border: none;
        }

        .btn-reject:hover {
            background-color: #c82333;
        }

        .btn-send {
            background-color: #007bff;
            border: none;
            color: white;
        }

        .btn-send:hover {
            background-color: #0056b3;
        }

        .btn-custom:focus {
            box-shadow: none;
        }

        .header {
            color: #343a40;
            font-size: 2rem;
            font-weight: bold;
```

templates > admin > manage_buddies.html > html > head > style > .card-text

```
<html>
<head>
    <style>
        .header {
            font-weight: bold;
        }

        .message-box {
            margin-top: 20px;
            width: 100%;
            max-width: 600px;
        }

        .message-box textarea {
            width: 100%;
            height: 150px;
            padding: 10px;
            border-radius: 10px;
            border: 1px solid #ccc;
            font-size: 16px;
            font-family: Arial, sans-serif;
        }
    </style>
</head>
<body class="p-5">
    <h2 class="header mb-4">Manage Travel Buddy Posts</h2>

    {% for buddy in buddies %}
        <div class="card mb-3">
            <div class="card-body">
                {% if buddy.type == 'single' %}
                    <h5 class="card-title">Single Buddy: {{ buddy.name }}</h5>
                    <p class="card-text">Destination: {{ buddy.destination }}</p>
                    <p class="card-text">{{ buddy.details }}</p>
                {% else %}
                    <h5 class="card-title">Group: {{ buddy.group_name }}</h5>
                    <p class="card-text">Destination: {{ buddy.destination }}</p>
```

templates > admin > manage_buddies.html > html

```
<html>
<body class="p-5">
        <div class="card mb-3">
            <div class="card-body">
                    <p class="card-text">Destination: {{ buddy.destination }}</p>
                    <p class="card-text">{{ buddy.group_details }}</p>
                {% endif %}
                <p>Status: <strong>{{ buddy.status }}</strong></p>

                {% if buddy.status == 'pending' %}
                    <a href="{{ url_for('approve_buddy', buddy_id=buddy._id|string) }}" class="btn btn-custom btn-approve">
                        <i class="bi bi-check-circle"></i> Approve
                    </a>
                    <a href="{{ url_for('reject_buddy', buddy_id=buddy._id|string) }}" class="btn btn-custom btn-reject">
                        <i class="bi bi-x-circle"></i> Reject
                    </a>

                    <!-- Message Box Section (appears for pending requests) -->
                    <div class="message-box">
                        <h5>Send a Message</h5>
                        <textarea placeholder="Type your message here..."></textarea>
                        <button class="btn btn-send mt-3">Send</button>
                    </div>
                {% elif buddy.status == 'approved' %}
                    <!-- For approved requests, display the message box below the status -->
                    <div class="message-box">
                        <h5>Send a Message</h5>
                        <textarea placeholder="Type your message here..."></textarea>
                        <button class="btn btn-send mt-3">Send</button>
                    </div>
                {% endif %}
            </div>
        </div>
    {% else %}
        <p>No travel buddy requests found.</p>
    {% endfor %}
</body>
</html>
```

4. Book Penthouse:

Booking a penthouse displays penthouse details such as title, location, descriptions, and price, with an option to view an image. The form collects necessary booking details such as dates, guest information, additional services, and payment options. The form data is submitted to the server when the user clicks the "Confirm Booking" button. The POST method sends the booking details, including check-in and check-out dates, room type, services, user information, and payment method, to the server-side function that handles the booking.

templates > book_penthouse.html > html > head > style > .form-section

```
<!-- templates/book_penthouse.html -->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Book {{ penthouse.title }}</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
    <style>
        body {
            background-color: #f8f9fa;
            padding-top: 30px;
        }

        .container {
            background: #fff;
            padding: 30px;
            border-radius: 16px;
            box-shadow: 0px 4px 25px rgba(0, 0, 0, 0.1);
        }

        h2 {
            color: #2c3e50;
        }

        .form-section {
            margin-top: 30px;
        }

        .img-fluid {
            max-height: 300px;
            object-fit: cover;
            border-radius: 12px;
        }

        .btn-success {
            background-color: #27ae60;
            border: none;
        }
```

templates > book_penthouse.html > html > head > style > .form-section

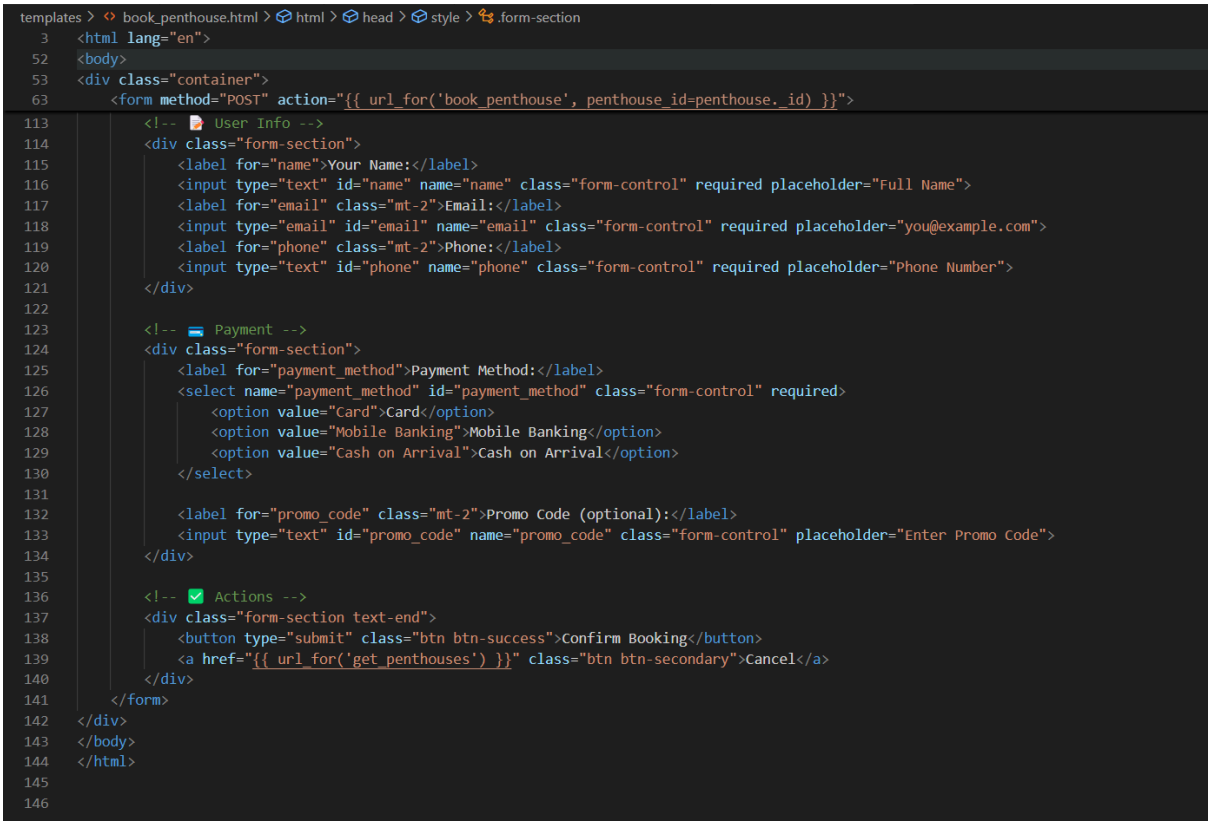
```
<html lang="en">
<head>
    <style>
        .btn-success {
        }

        .btn-secondary {
            margin-left: 10px;
            background-color: #bdc3c7;
            border: none;
        }

        .btn-secondary:hover {
            background-color: #95a5a6;
        }
    </style>
</head>
<body>
<div class="container">
    <h2>🚙 Book {{ penthouse.title }}</h2>
    <p><strong>Location:</strong> {{ penthouse.location }}</p>
    <p><strong>Description:</strong> {{ penthouse.features }}</p>
    <p><strong>Price:</strong> ${{ penthouse.price }} per night</p>

    {% if penthouse.image_url %}
        <img src="{{ penthouse.image_url }}" alt="Penthouse Image" class="img-fluid mb-4">
    {% endif %}

    <form method="POST" action="{{ url_for('book_penthouse', penthouse_id=penthouse._id) }}">
        <!-- 🗓️ Date Selection -->
        <div class="row form-section">
            <div class="col-md-6">
                <label for="check_in">Check-in Date:</label>
                <input type="date" id="check_in" name="check_in" class="form-control" required>
            </div>
            <div class="col-md-6">
                <label for="check_out">Check-out Date:</label>
                <input type="date" id="check_out" name="check_out" class="form-control" required>
            </div>
```

templates > book_penthouse.html > html > head > style > .form-section

```
<html lang="en">
<body>
<div class="container">
    <form method="POST" action="{{ url_for('book_penthouse', penthouse_id=penthouse._id) }}">
        <div class="row form-section">
        </div>

        <!-- 👥 Guest Info -->
        <div class="row form-section">
            <div class="col-md-4">
                <label for="adults">Adults:</label>
                <input type="number" id="adults" name="adults" class="form-control" required min="1" placeholder="1+">
            </div>
            <div class="col-md-4">
                <label for="children">Children:</label>
                <input type="number" id="children" name="children" class="form-control" min="0" placeholder="0+">
            </div>
            <div class="col-md-4">
                <label for="room_type">Room Type:</label>
                <select name="room_type" id="room_type" class="form-control" required>
                    <option value="King">King</option>
                    <option value="Queen">Queen</option>
                    <option value="Double">Double</option>
                </select>
            </div>
        </div>

        <!-- 🚗 Extra Services -->
        <div class="form-section">
            <label>Additional Services:</label>
            <div class="form-check">
                <input class="form-check-input" type="checkbox" name="services" value="Breakfast" id="breakfast">
                <label class="form-check-label" for="breakfast">Breakfast</label>
            </div>
            <div class="form-check">
                <input class="form-check-input" type="checkbox" name="services" value="Parking" id="parking">
                <label class="form-check-label" for="parking">Parking</label>
            </div>
            <div class="form-check">
                <input class="form-check-input" type="checkbox" name="services" value="Airport Pickup" id="airport pickup">
```

templates > book_penthouse.html > html > head > style > .form-section

```
  3  <html lang="en">
 52  <body>
 53  <div class="container">
 63      <form method="POST" action="{{ url_for('book_penthouse', penthouse_id=penthouse. id) }}">
 97          <div class="form-section">
106              </div>
107              <div class="form-check">
108                  <input class="form-check-input" type="checkbox" name="services" value="Airport Pickup" id="airport_pickup">
109                  <label class="form-check-label" for="airport_pickup">Airport Pickup</label>
110              </div>
111          </div>
112
113          <!-- 📝 User Info -->
114          <div class="form-section">
115              <label for="name">Your Name:</label>
116              <input type="text" id="name" name="name" class="form-control" required placeholder="Full Name">
117              <label for="email" class="mt-2">Email:</label>
118              <input type="email" id="email" name="email" class="form-control" required placeholder="you@example.com">
119              <label for="phone" class="mt-2">Phone:</label>
120              <input type="text" id="phone" name="phone" class="form-control" required placeholder="Phone Number">
121          </div>
122
123          <!-- 💳 Payment -->
124          <div class="form-section">
125              <label for="payment_method">Payment Method:</label>
126              <select name="payment_method" id="payment_method" class="form-control" required>
127                  <option value="Card">Card</option>
128                  <option value="Mobile Banking">Mobile Banking</option>
129                  <option value="Cash on Arrival">Cash on Arrival</option>
130              </select>
131
132              <label for="promo_code" class="mt-2">Promo Code (optional):</label>
133              <input type="text" id="promo_code" name="promo_code" class="form-control" placeholder="Enter Promo Code">
134          </div>
135
136          <!-- ✅ Actions -->
137          <div class="form-section text-end">
138              <button type="submit" class="btn btn-success">Confirm Booking</button>
139              <a href="{{ url_for('get_penthouses') }}" class="btn btn-secondary">Cancel</a>
```



5. Transportation Tickets Page:

The Transportation Tickets page allows users to choose between booking a bus ticket or a car service for their travels. This page:

- Displays a background image to create a visually appealing experience.
- Utilizes Bootstrap for a responsive layout and cards to present the bus and car options.
- Provides dynamic links that redirect users to different pages for more detailed information about bus routes or cars availability.
- Includes a back-to-home button for easy navigation.

The design is clean, simple, and user-friendly, providing a seamless experience for users to make transportation bookings.

templates > transport_tickets.html > html > head > style > .card

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Transportation Tickets</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
    <style>
        body {
            background-image: url('https://cdnsm5-hosted.civiclive.com/UserFiles/Servers/Server_7964838/Image/Government/Departments/Public%2
            background-size: cover;
            background-position: center;
            background-attachment: fixed;
        }
        .container {
            background: rgba(255, 255, 255, 0.85);
            padding: 30px;
            border-radius: 12px;
            box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.1);
            margin-top: 50px;
        }
        .card {
            border-radius: 10px;
            box-shadow: 0px 3px 8px rgba(0, 0, 0, 0.1);
            transition: transform 0.3s;
        }
        .card:hover {
            transform: scale(1.05);
        }
    </style>
</head>
<body>
    <div class="container mt-5">
        <h2 class="text-center">Choose Your Transportation</h2>

        <div class="row mt-4">
            <div class="col-md-4">
                <div class="card text-center p-3">
```

templates > transport_tickets.html > html > head > style > .card

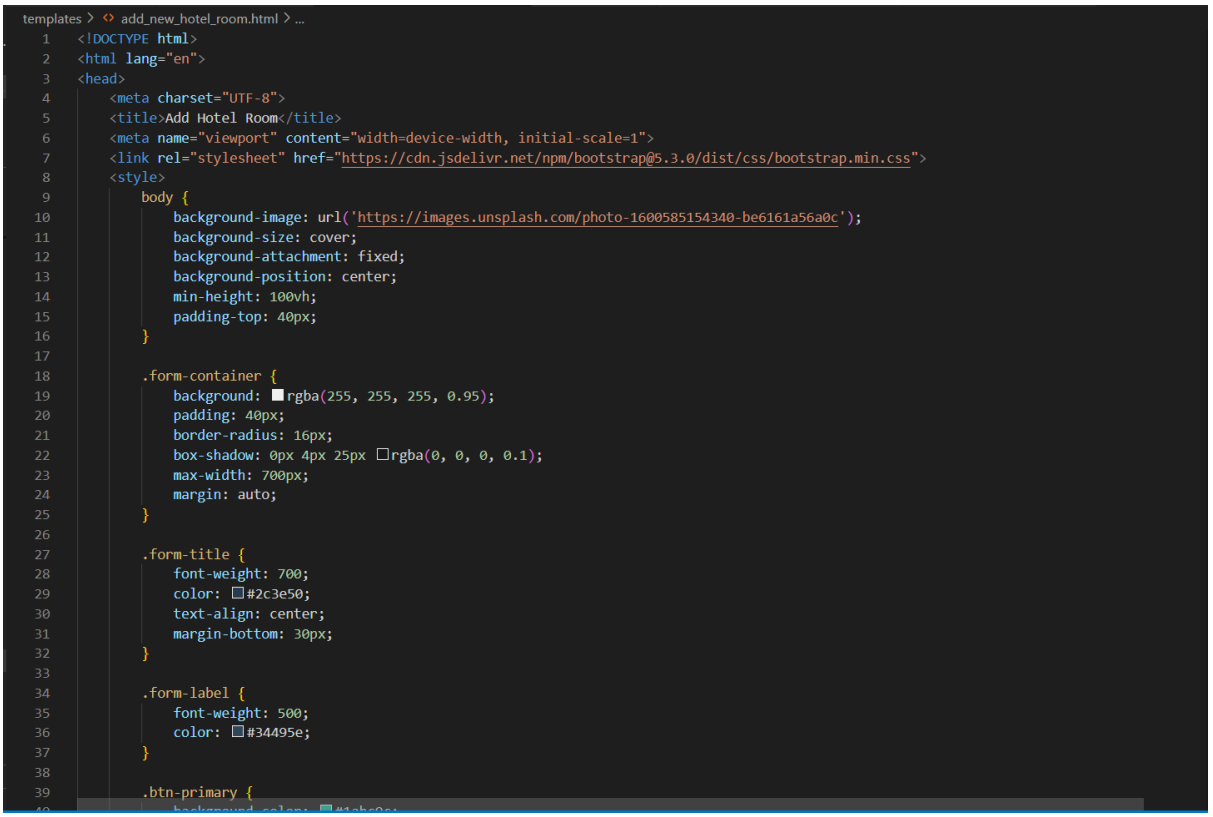
```
<html lang="en">
<head>
    <style>
        .card:hover {
            transform: scale(1.05);
        }
    </style>
</head>
<body>
    <div class="container mt-5">
        <h2 class="text-center">Choose Your Transportation</h2>

        <div class="row mt-4">
            <div class="col-md-4">
                <div class="card text-center p-3">
                    <h4>Bus Ticket</h4>
                    <p>Book your comfortable bus tickets for intercity travel.</p>
                    <button class="btn btn-primary" onclick="window.location.href='{{ url_for('show_bus_tickets') }}'">See Bus Routes</button>
                </div>
            </div>
            <div class="col-md-4">
                <div class="card text-center p-3">
                    <h4>Car Service</h4>
                    <p>Choose your preferred Car for smooth and scenic travel.</p>
                    <button class="btn btn-primary" onclick="window.location.href='{{ url_for('show_available_cars') }}'">See Available Cars</
                </div>
            </div>
        </div>

        <div class="text-center mt-4">
            <a href="/" class="btn btn-secondary">Back to Home</a>
        </div>
    </div>
</body>
</html>
```

6. Add New Hotel Room:

This page is a clean, user-friendly form for adding new hotel room details to the system. It allows the admin to:

- Add various details about the room, including room type, amenities, price, and description.
- Dynamically add multiple room numbers.

Submit the room data to the backend for storage.



templates > add_new_hotel_room.html > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Add Hotel Room</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
    <style>
        body {
            background-image: url('https://images.unsplash.com/photo-1600585154340-be6161a56a0c');
            background-size: cover;
            background-attachment: fixed;
            background-position: center;
            min-height: 100vh;
            padding-top: 40px;
        }

        .form-container {
            background: rgba(255, 255, 255, 0.95);
            padding: 40px;
            border-radius: 16px;
            box-shadow: 0px 4px 25px rgba(0, 0, 0, 0.1);
            max-width: 700px;
            margin: auto;
        }

        .form-title {
            font-weight: 700;
            color: #2c3e50;
            text-align: center;
            margin-bottom: 30px;
        }

        .form-label {
            font-weight: 500;
            color: #34495e;
        }

        .btn-primary {
```

templates > add_new_hotel_room.html > ...

```
<html lang="en">
<head>
    <style>

        .btn-primary {
            background-color: #1abc9c;
            border: none;
            font-weight: 600;
        }

        .btn-primary:hover {
            background-color: #16a085;
        }
    </style>
</head>
<body>
    <div class="form-container">
        <h2 class="form-title">🚗 Add New Hotel Room</h2>
        {% with messages = get_flashed_messages(with_categories=true) %}
            {% if messages %}
                {% for category, message in messages %}
                    <div class="alert alert-{{ category }} alert-dismissible fade show" role="alert">
                        {{ message }}
                        <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
                    </div>
                {% endfor %}
            {% endif %}
        {% endwith %}

        <form method="POST" action="{{ url_for('add_new_hotel_room') }}">
            <div class="mb-3">
                <label for="room_type" class="form-label">Room Type</label>
                <input type="text" class="form-control" id="room_type" name="room_type" placeholder="e.g. Deluxe Suite" required>
            </div>

            <div class="mb-3">
                <label for="beds" class="form-label">Number of Beds</label>
                <input type="number" class="form-control" id="beds" name="beds" min="1" required>
            </div>
```

templates > add_new_hotel_room.html > html > body > div.form-container

```
<html lang="en">
<body>
    <div class="form-container">
        <form method="POST" action="{{ url_for('add_new_hotel_room') }}">

            <div class="mb-3">
                <label for="amenities" class="form-label">Amenities <small class="text-muted">(comma-separated)</small></label>
                <input type="text" class="form-control" id="amenities" name="amenities" placeholder="WiFi, TV, Mini Bar">
            </div>

            <div class="mb-3">
                <label for="price" class="form-label">Price per Night (USD)</label>
                <input type="number" class="form-control" id="price" name="price" step="0.01" min="0" required>
            </div>

            <div class="mb-3">
                <label for="image_url" class="form-label">Image URL</label>
                <input type="url" class="form-control" id="image_url" name="image_url" placeholder="https://example.com/image.jpg" required>
            </div>

            <div class="mb-3">
                <label for="description" class="form-label">Room Description</label>
                <textarea class="form-control" id="description" name="description" rows="4" placeholder="Brief description of the room"></text
            </div>

            <div class="mb-3">
                <label class="form-label">Room Numbers</label>
                <div id="roomNumbersContainer">
                    <div class="input-group mb-2">
                        <input type="text" name="room_numbers[]" class="form-control" placeholder="Enter room number" required>
                        <button type="button" class="btn btn-outline-secondary remove-room" onclick="removeRoomField(this)">✕</button>
                    </div>
                </div>
                <button type="button" class="btn btn-sm btn-success" onclick="addRoomField()">➕ Add Room Number</button>
            </div>

            <button type="submit" class="btn btn-primary w-100">Add Room</button>
        </form>
```

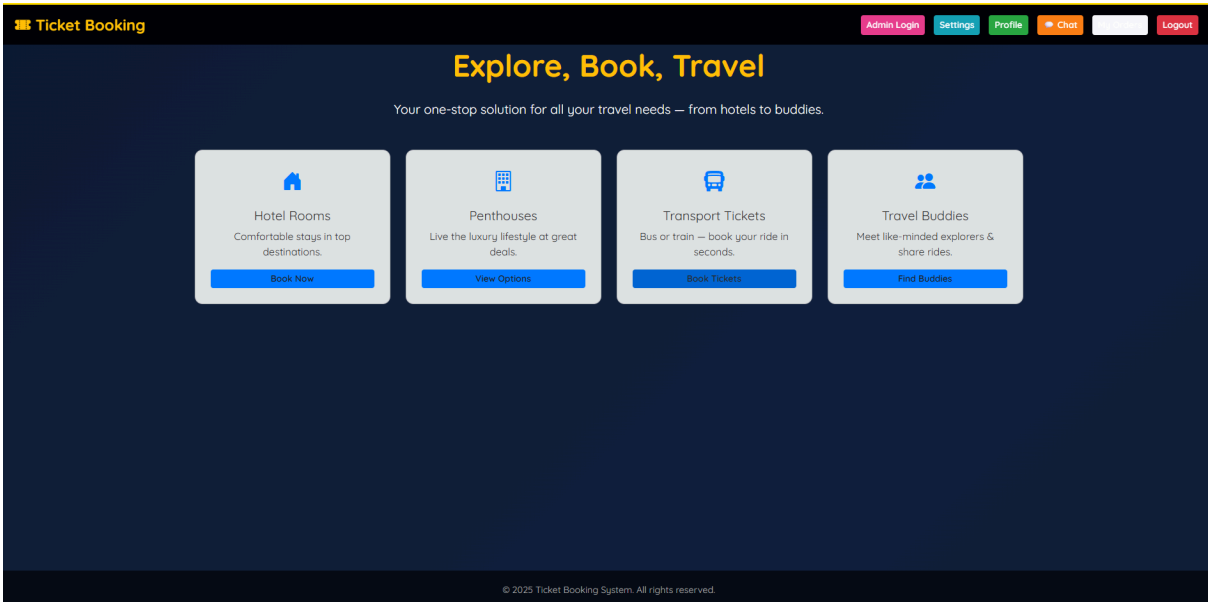
templates > add_new_hotel_room.html > html > body > div.form-container > form

```
<html lang="en">
<body>
    <div class="form-container">
        </form>
    </div>

    {% if rooms %}
    <div class="container mt-5">
        <hr>
        <h4 class="text-center text-white">🏞️ Recently Added Hotel Rooms</h4>
        <div class="row mt-4">
            {% for room in rooms %}
            <div class="col-md-4 mb-4">
                <div class="card h-100 shadow-sm">
                    <img src="{{ room.image_url }}" class="card-img-top" alt="Room Image" style="height: 200px; object-fit: cover;">
                    <div class="card-body">
                        <h5 class="card-title">{{ room.room_type }}</h5>
                        <p class="card-text"><strong>Beds:</strong> {{ room.beds }}</p>
                        <p class="card-text"><strong>Price:</strong> ${{ room.price }} per night</p>
                        <p class="card-text"><strong>Amenities:</strong> {{ room.amenities }}</p>
                        <p class="card-text text-muted small">{{ room.description }}</p>
                    </div>
                </div>
            </div>
            {% endfor %}
        </div>
    </div>
    {% endif %}

    <script>
        function addRoomField() {
            const container = document.getElementById('roomNumbersContainer');
            const field = document.createElement('div');
            field.classList.add('input-group', 'mb-2');
            field.innerHTML = `
                <input type="text" name="room_numbers[]" class="form-control" placeholder="Enter room number" required>
                <button type="button" class="btn btn-outline-secondary remove-room" onclick="removeRoomField(this)">✖</button>
            `;
            container.appendChild(field);
```

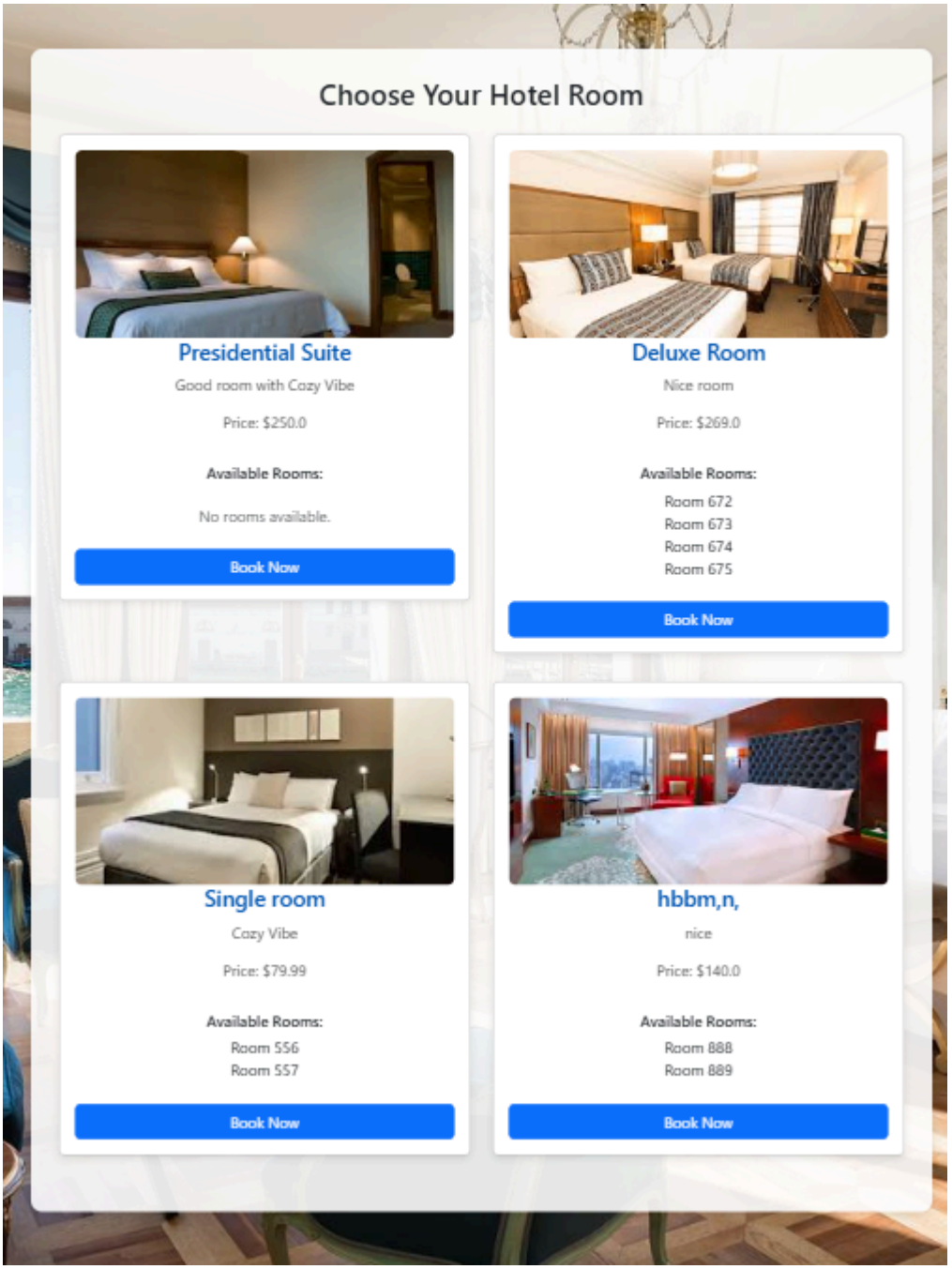
templates > add_new_hotel_room.html > html > body

```
<html lang="en">
<body>
    <div class="container mt-5">
        <div class="row mt-4">
            <div class="col-md-4 mb-4">
                </div>
                {% endfor %}
            </div>
        </div>
        {% endif %}

        <script>
            function addRoomField() {
                const container = document.getElementById('roomNumbersContainer');
                const field = document.createElement('div');
                field.classList.add('input-group', 'mb-2');
                field.innerHTML = `
                    <input type="text" name="room_numbers[]" class="form-control" placeholder="Enter room number" required>
                    <button type="button" class="btn btn-outline-secondary remove-room" onclick="removeRoomField(this)">✖</button>
                `;
                container.appendChild(field);
            }

            function removeRoomField(button) {
                button.parentElement.remove();
            }
        </script>

        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
    </body>
    </html>
```

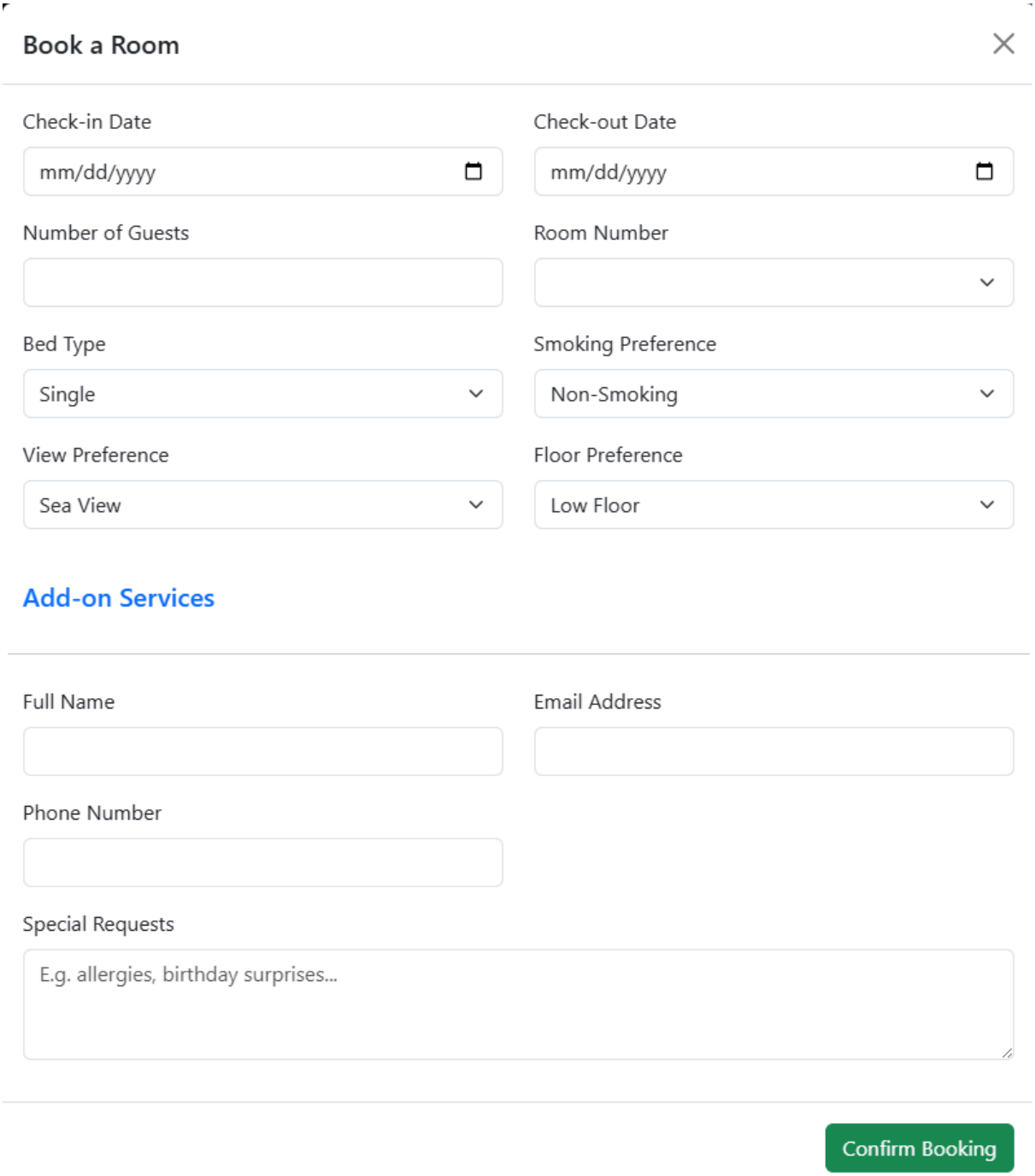
7. User Manual



On the user home page of the Ticket Booking System, users are welcomed with a vibrant and modern interface featuring a navbar for easy navigation. Logged-in users can access their profile, settings, orders, and even chat with support. The main section highlights four key services: booking hotel rooms, exploring penthouses, reserving transport tickets (bus or train), and connecting with travel buddies. Each service is represented with a card containing an icon, brief description, and a call-to-action button. The overall design ensures a smooth user experience with scroll animations and a responsive layout.

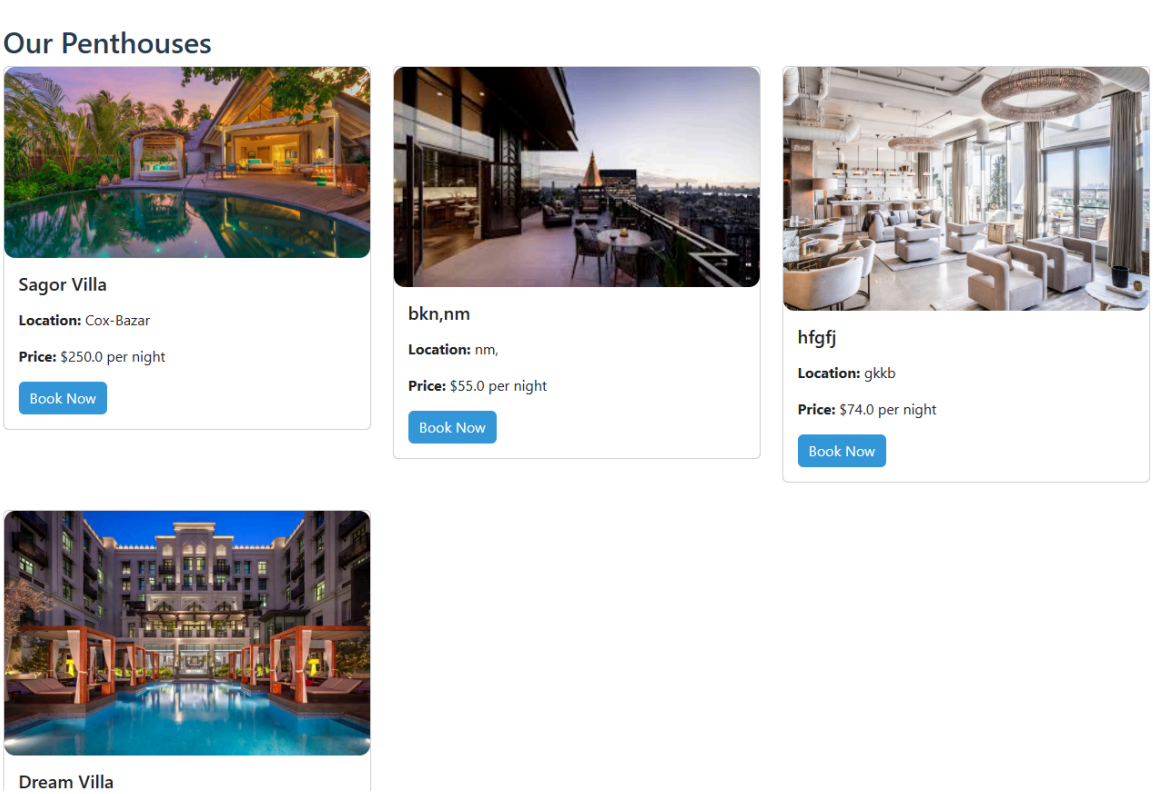


Here users can see all the available hotels and rooms. Based on availability of rooms, users can book that particular room in that hotel.

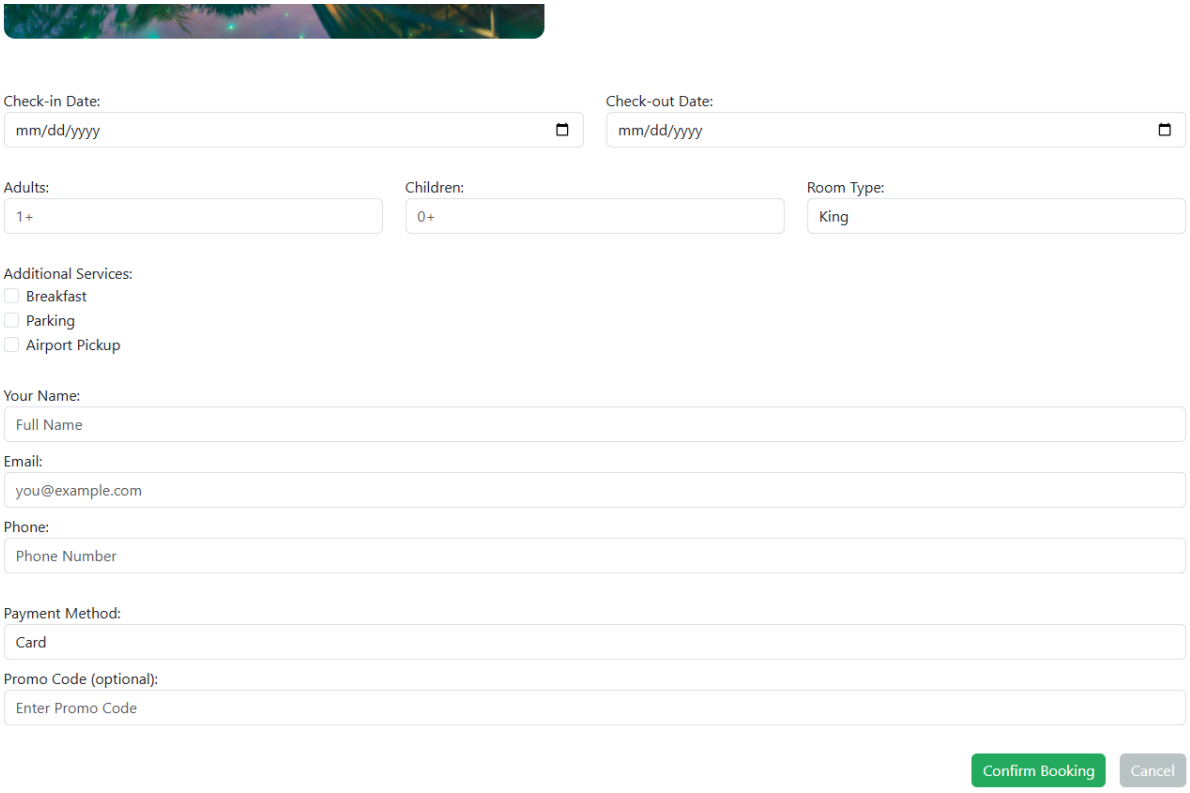


This is the information page that needs to be filled by the users to book hotel room.

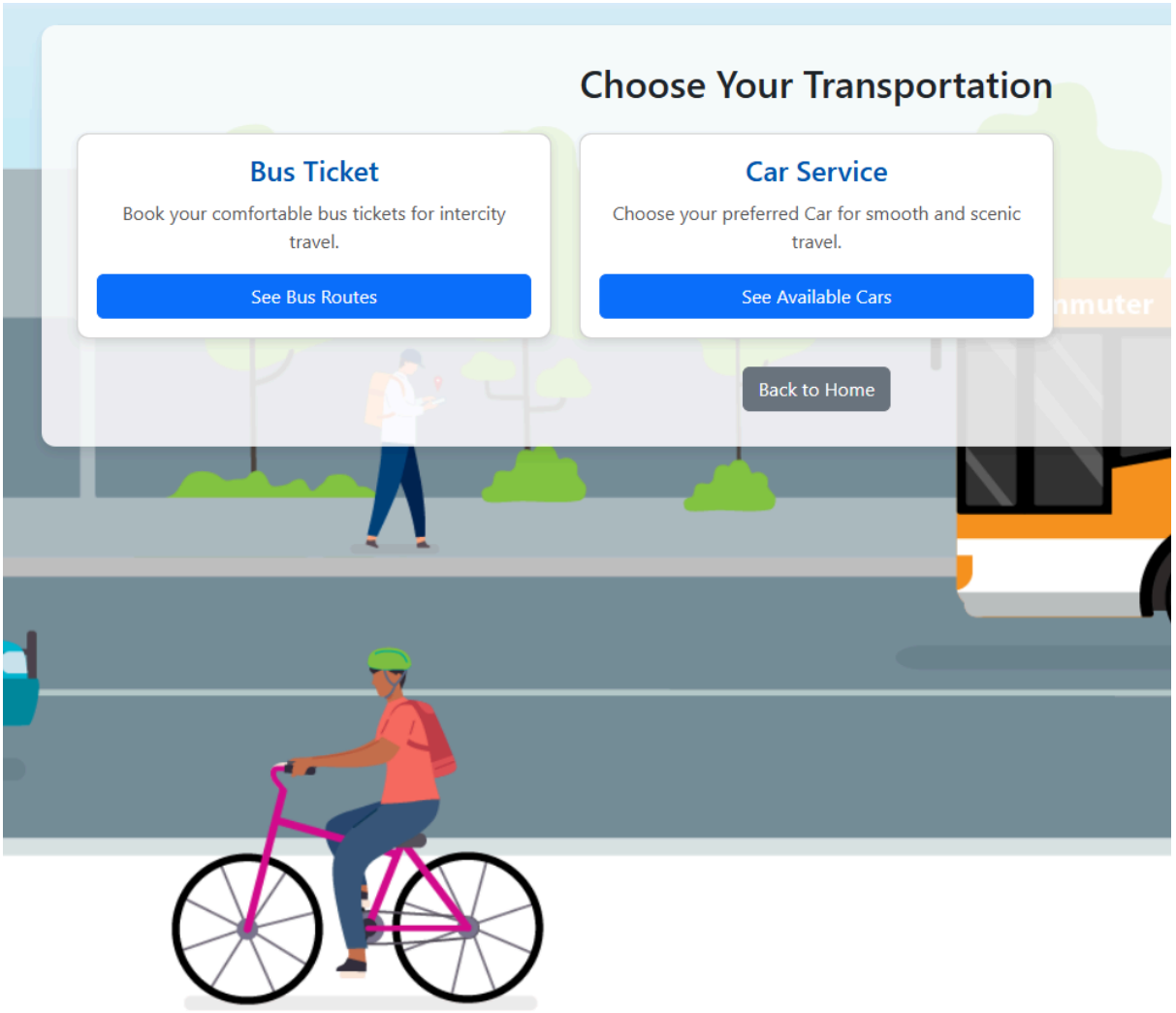
Penthouse:



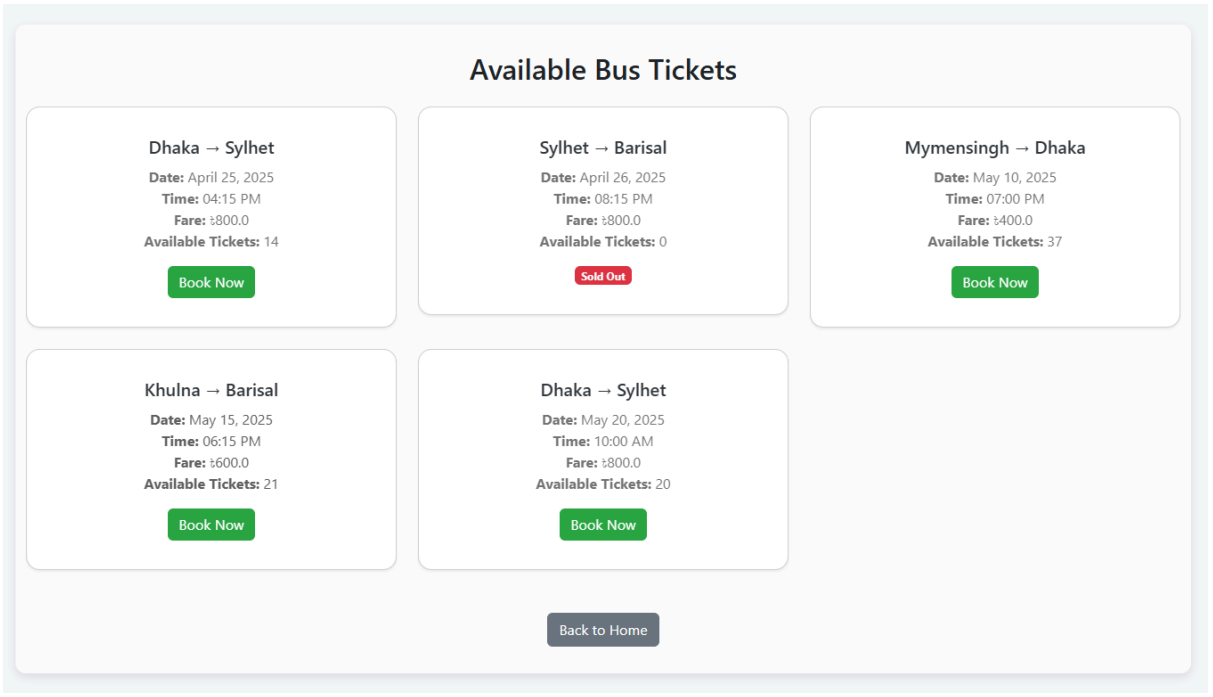
Users can See all the available penthouses and can book them upon availability. Below the form is the information form to book penthouses.



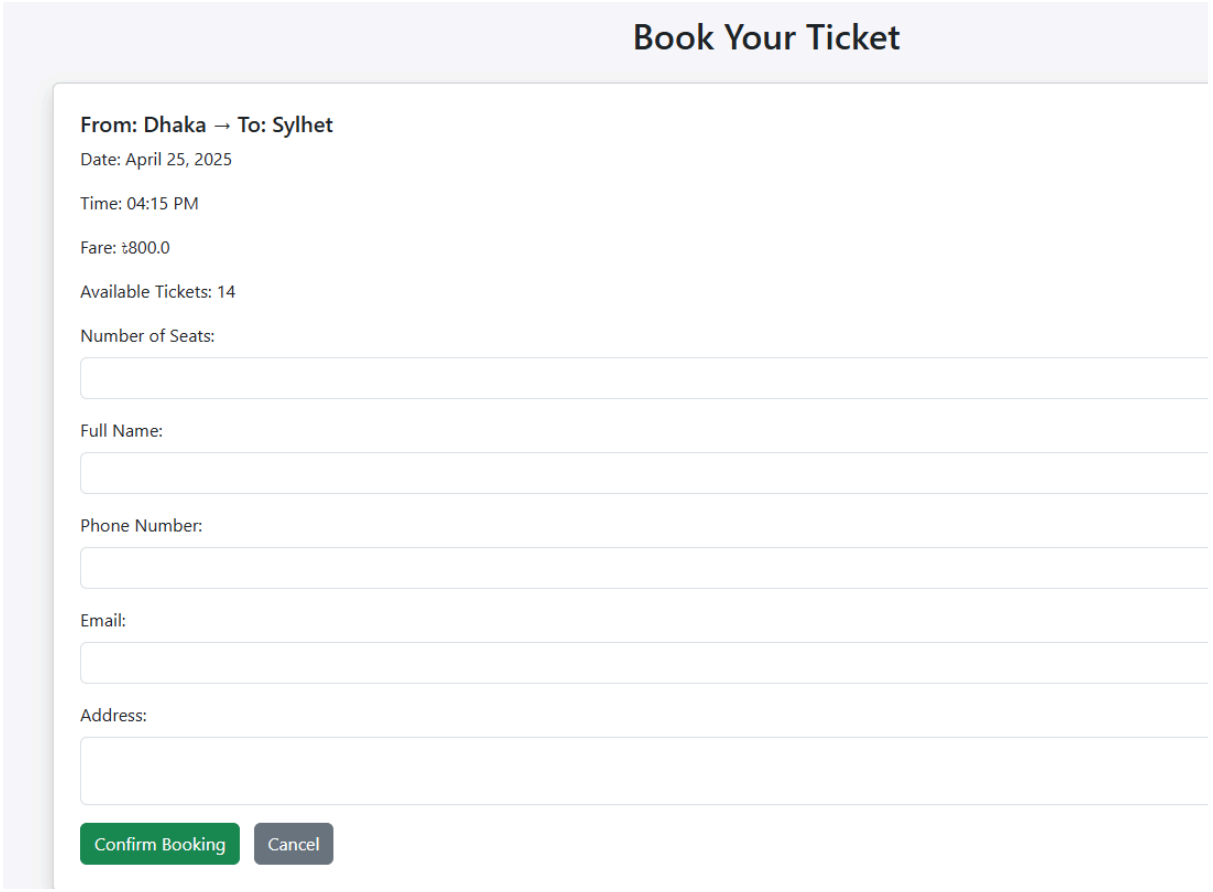
Transportation:



There are two types of transportation systems.



This is available bus tickets. Users can book bus tickets on any available bus routes upon availability. Below the form is the information form.

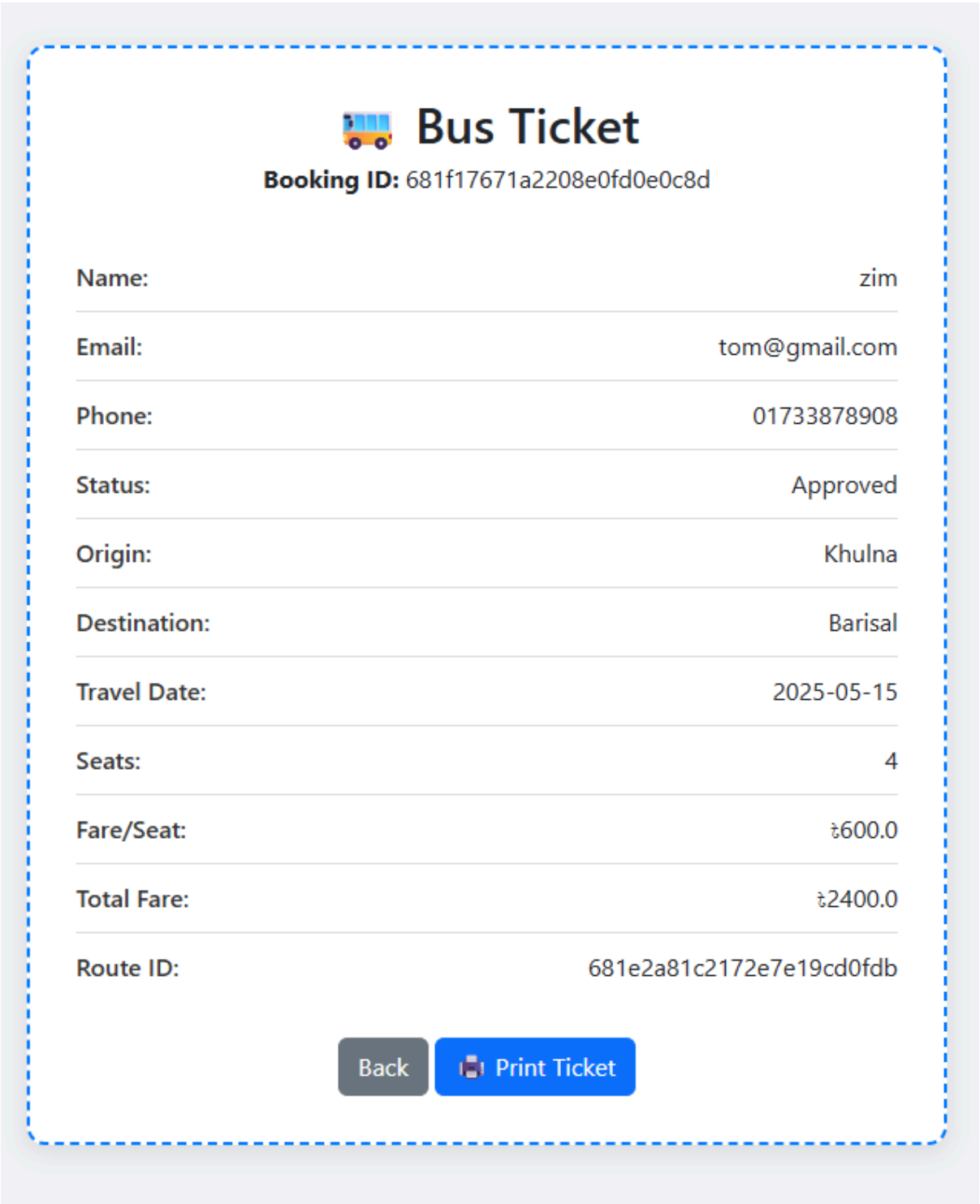


And after confirming the order by admin users can print their tickets from my orders/bill.

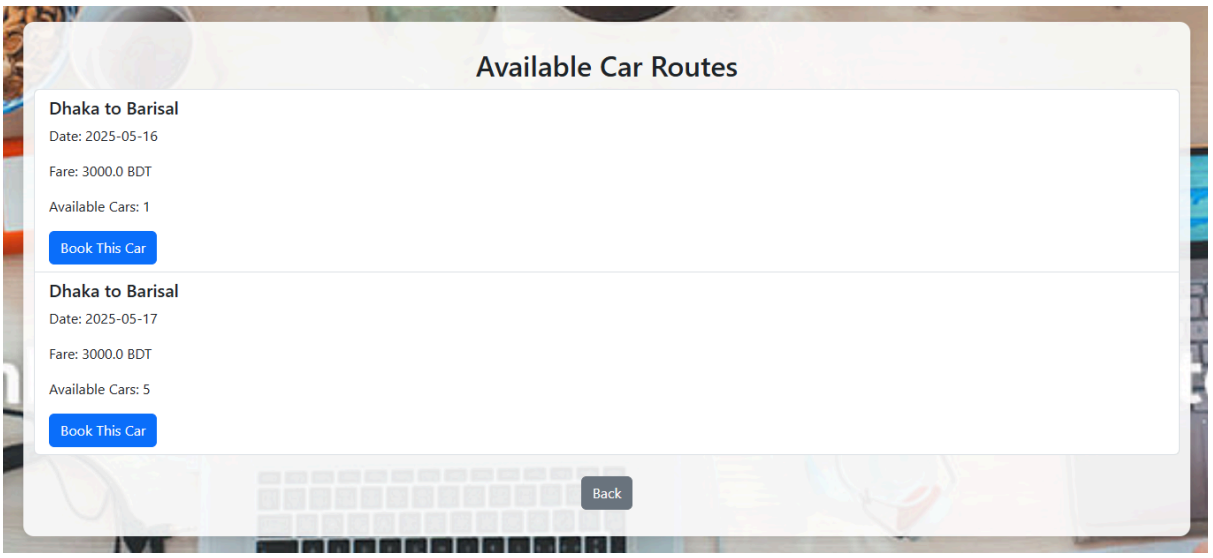
My Bus Reservations

Booking ID	Origin	Destination	Date	Seats	Total Fare	Status	View Bill
6814dfd20f64334d7ab7ff8f	Dhaka	Sylhet	April 25, 2025	3	$2400.0	Approved	View Bill
6814e244c0179d344997343b	Dhaka	Sylhet	April 25, 2025	3	$2400.0	Approved	View Bill
681dde69c2172e7e19cd0fd3	Dhaka	Sylhet	April 25, 2025	5	$4000.0	Approved	View Bill
681f0dc61a2208e0fd0e0c8c	Khulna	Barisal	May 15, 2025	5	$3000.0	Approved	View Bill
681f17671a2208e0fd0e0c8d	Khulna	Barisal	May 15, 2025	4	$2400.0	Approved	View Bill

Below the picture is a ticket.



Car:
Same for the cars.



Book Car: Dhaka → Barisal

Your Name *

Preferred Start Time *

--:-- --

Number of People *

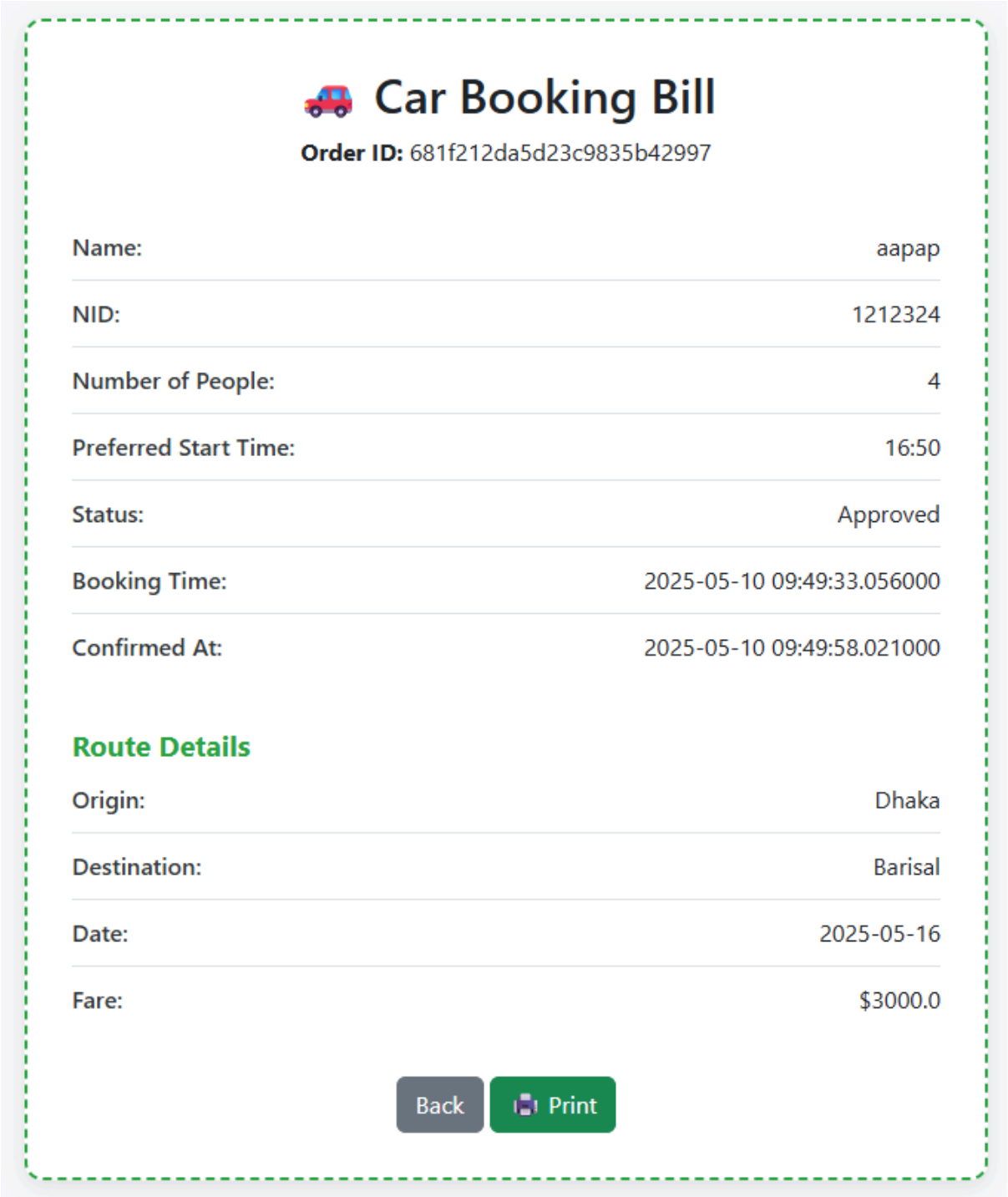
NID Card Number *

Additional Notes (optional)

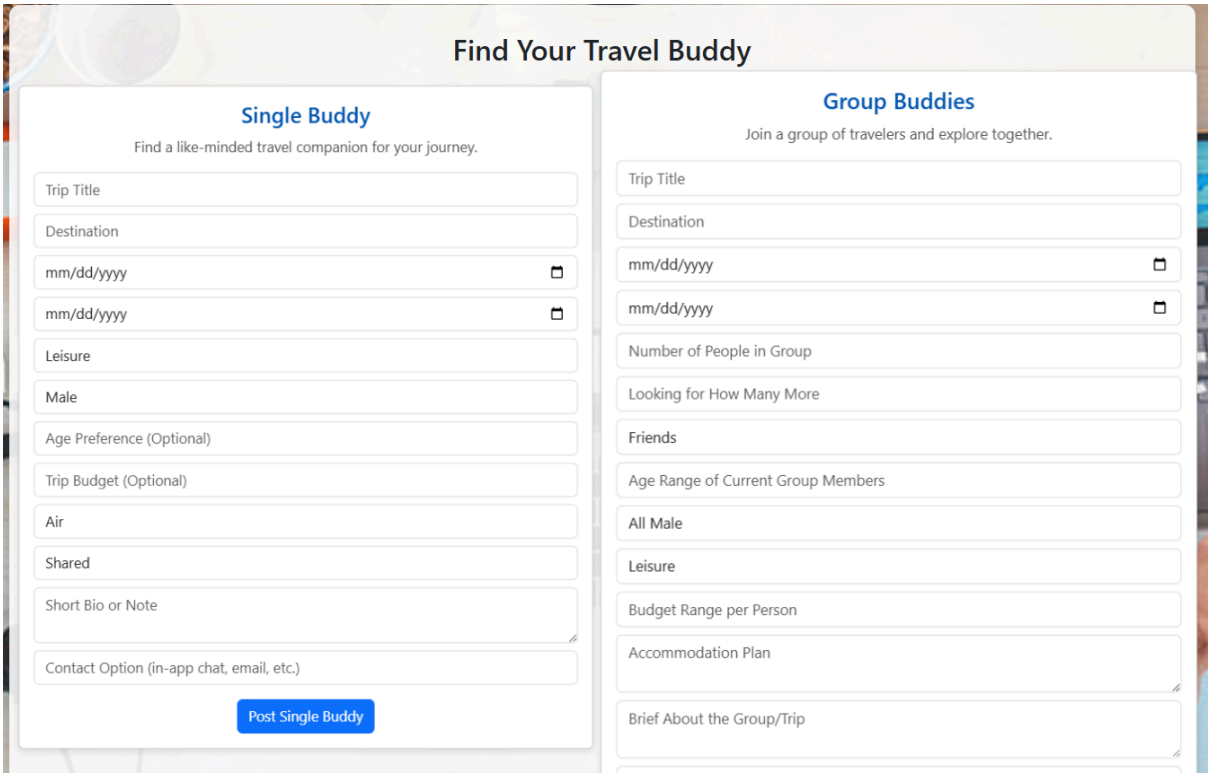
Confirm Booking Cancel

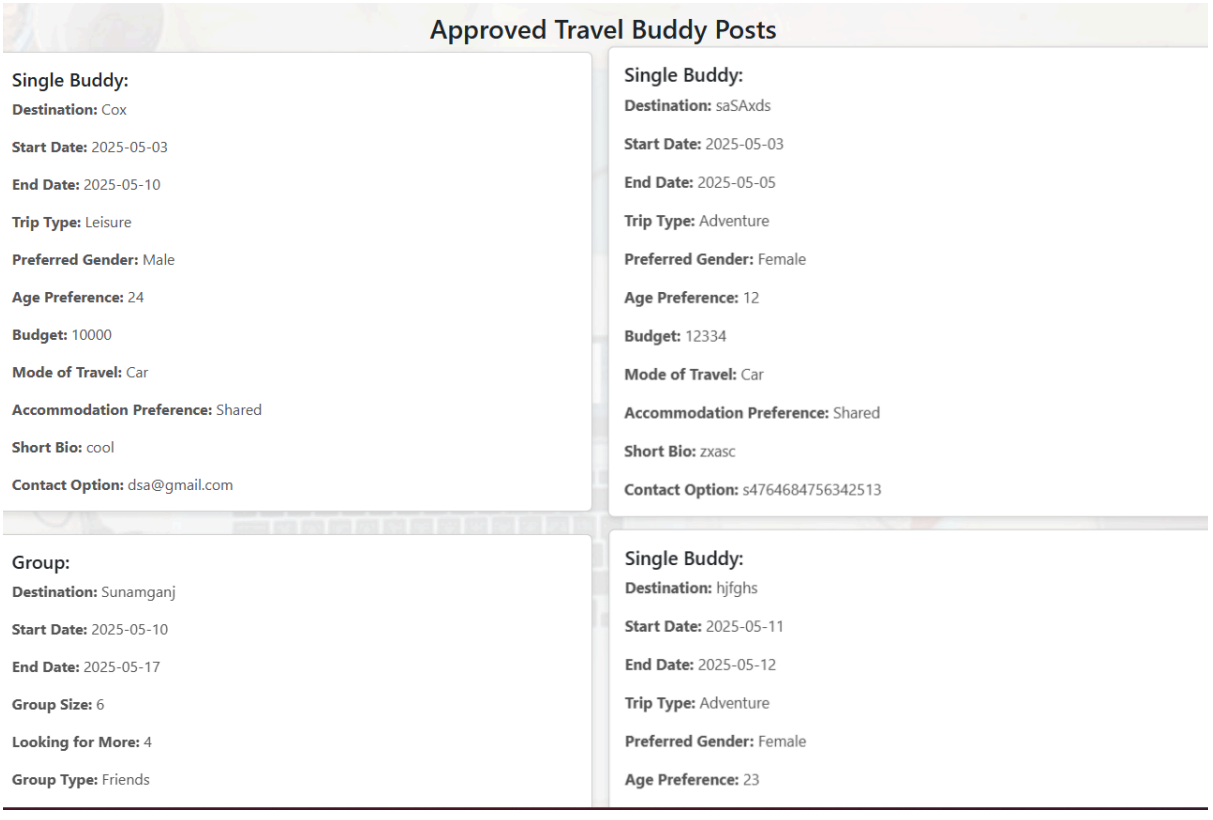
My Car Reservations

Reservation ID	Origin	Destination	Date	Status	View Bill
6813dbe5e19632149a70eaee	Dhaka	Barisal	2025-05-16	approved	View Bill
681f212da5d23c9835b42997	Dhaka	Barisal	2025-05-16	approved	View Bill



Buddy: user can post to find travels buddy for both single buddy and group buddies. And can see the posts of others to communicate with.





Users profile display and settings: users can see their profile by pressing profile button and can edit their information by pressing settings button.

Your Profile

Name: tommy

Username: tom

Email: tom@gmail.com

Phone: 12345678

Address: N/A

Description: N/A

Edit Your Profile

Enter Image URL

Username

tom

Email

tom@gmail.com

Phone Number

12345678

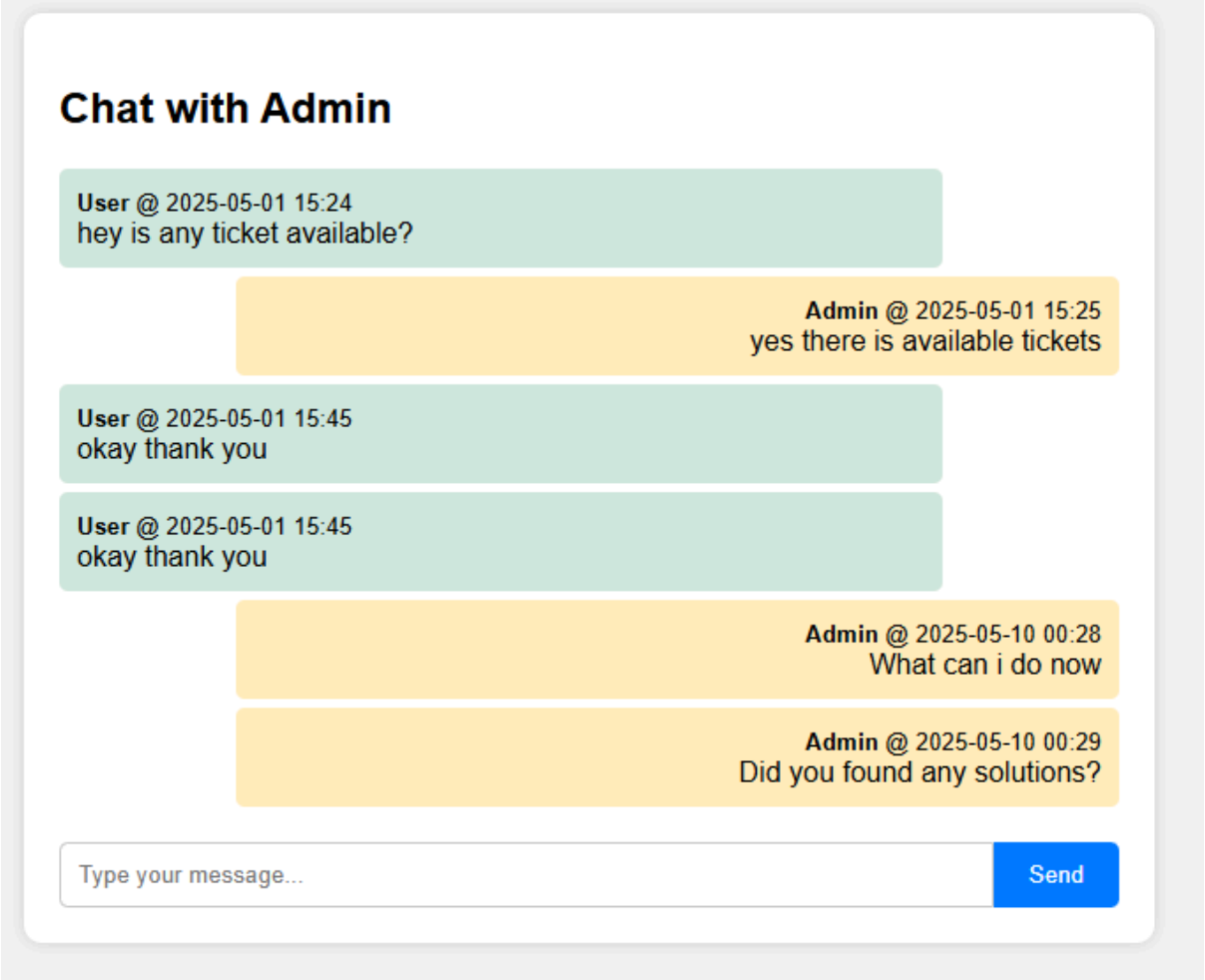
Address

Short Description

New Password (optional)

Update Profile

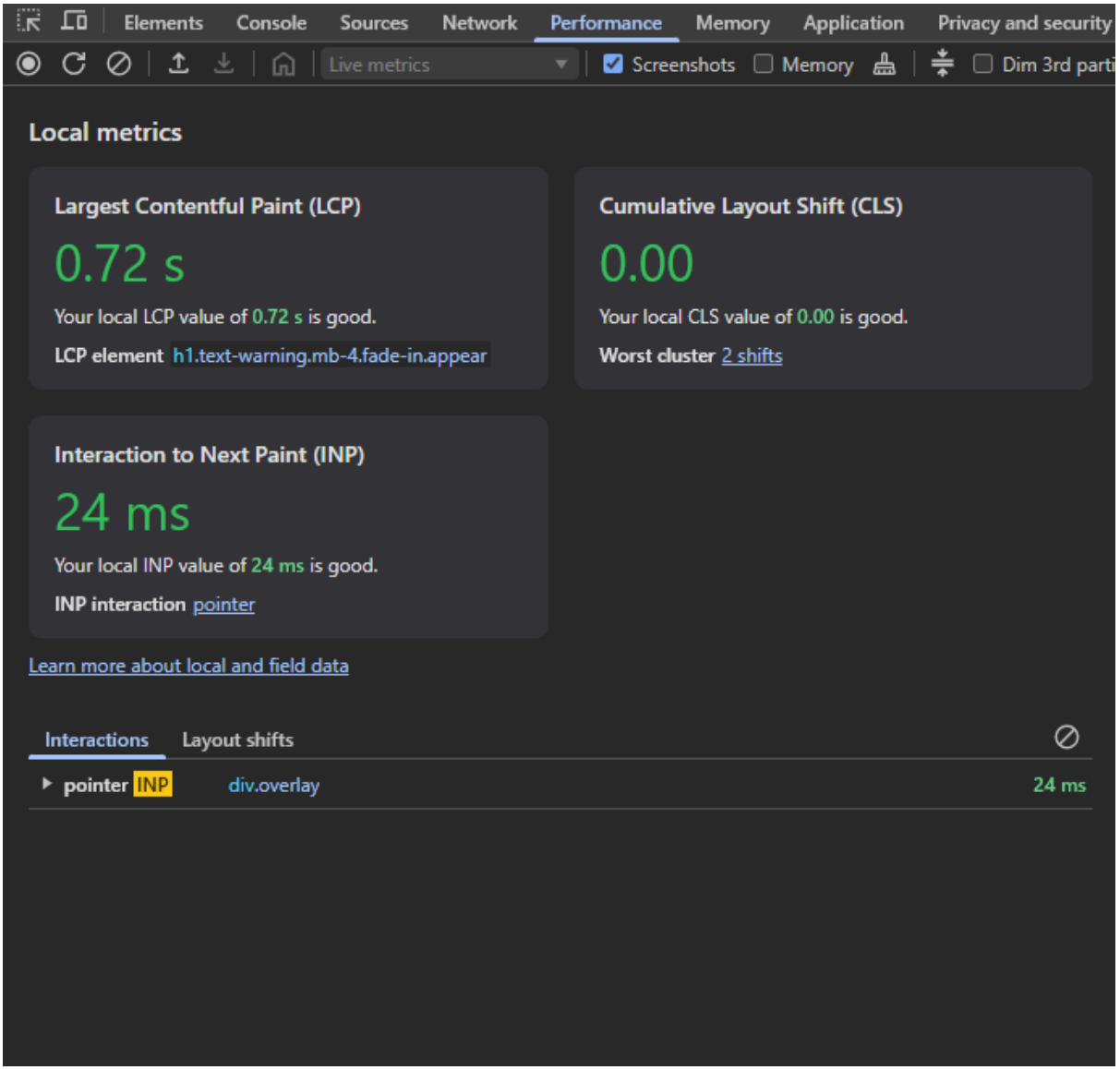
Chat: Users can directly chat with admins through a realtime chat system by pressing the chat button.



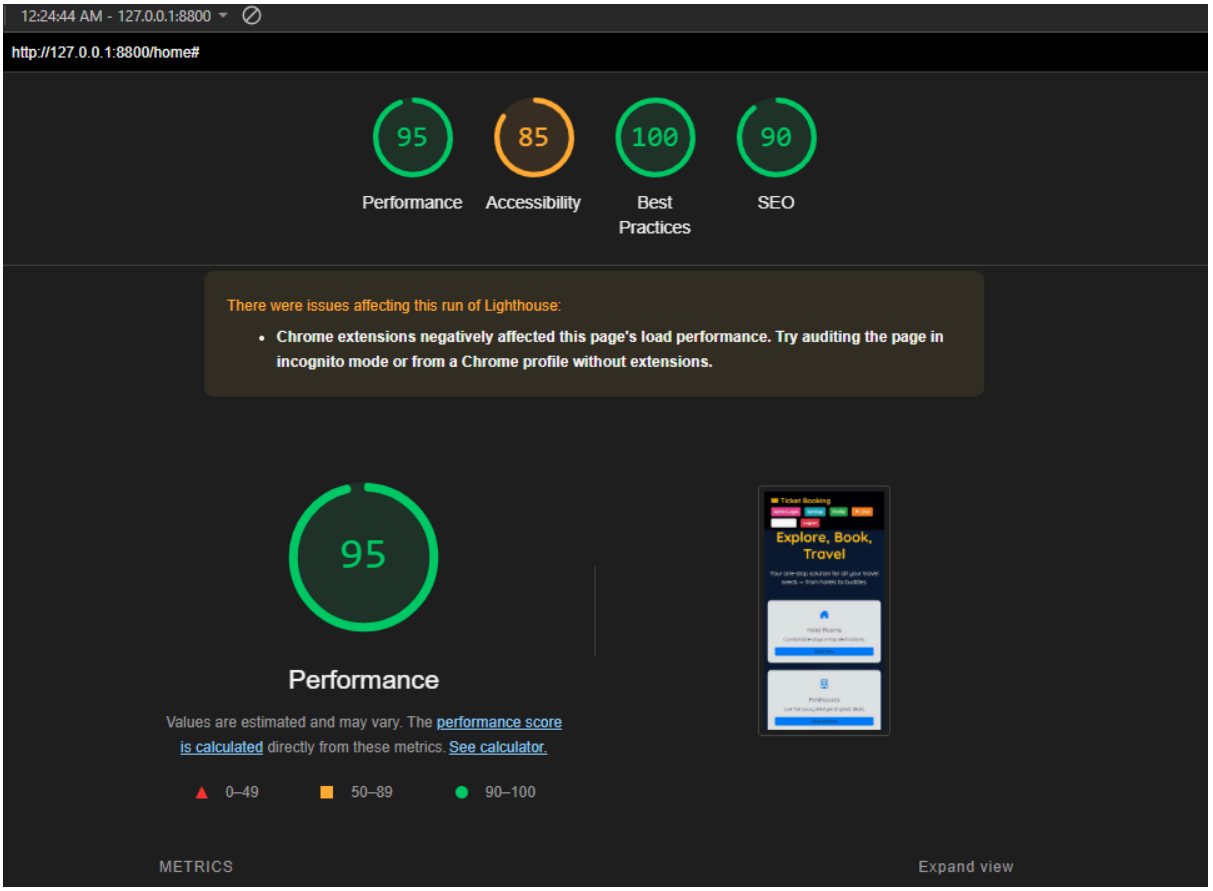
8. Performance and Network Analysis

Generate Performance and Network analysis report using lighthouse DevTool. Add screenshots of the report. Also, attach a screenshot of your system in any mobile viewport. **The system UI must be responsive.**

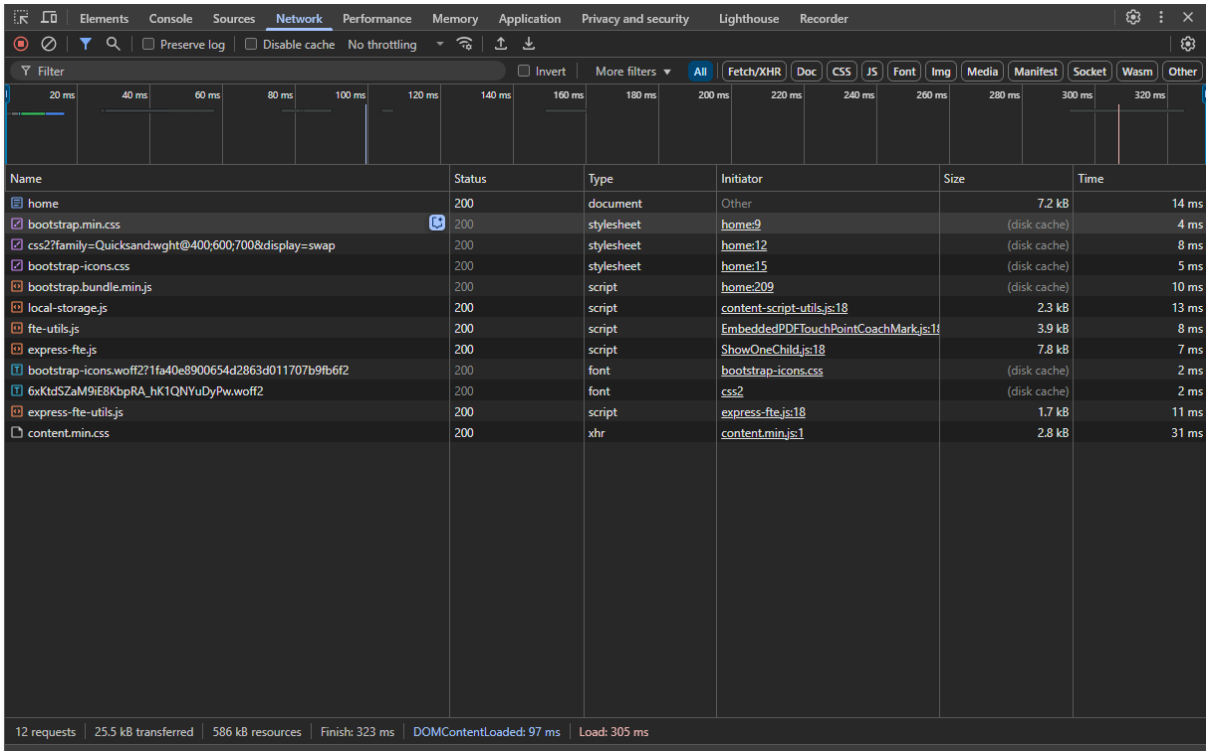
Performance:


Elements
Console
Sources
Network
Performance
Memory
Application
Privacy and security
Live metrics
Screenshots
Memory
Dim 3rd parti
Local metrics
Largest Contentful Paint (LCP)
0.72 s
Your local LCP value of 0.72 s is good.
LCP element h1.text-warning.mb-4.fade-in.appear
Cumulative Layout Shift (CLS)
0.00
Your local CLS value of 0.00 is good.
Worst cluster 2 shifts
Interaction to Next Paint (INP)
24 ms
Your local INP value of 24 ms is good.
INP interaction pointer
Learn more about local and field data
Interactions
Layout shifts
pointer INP
div.overlay
24 ms

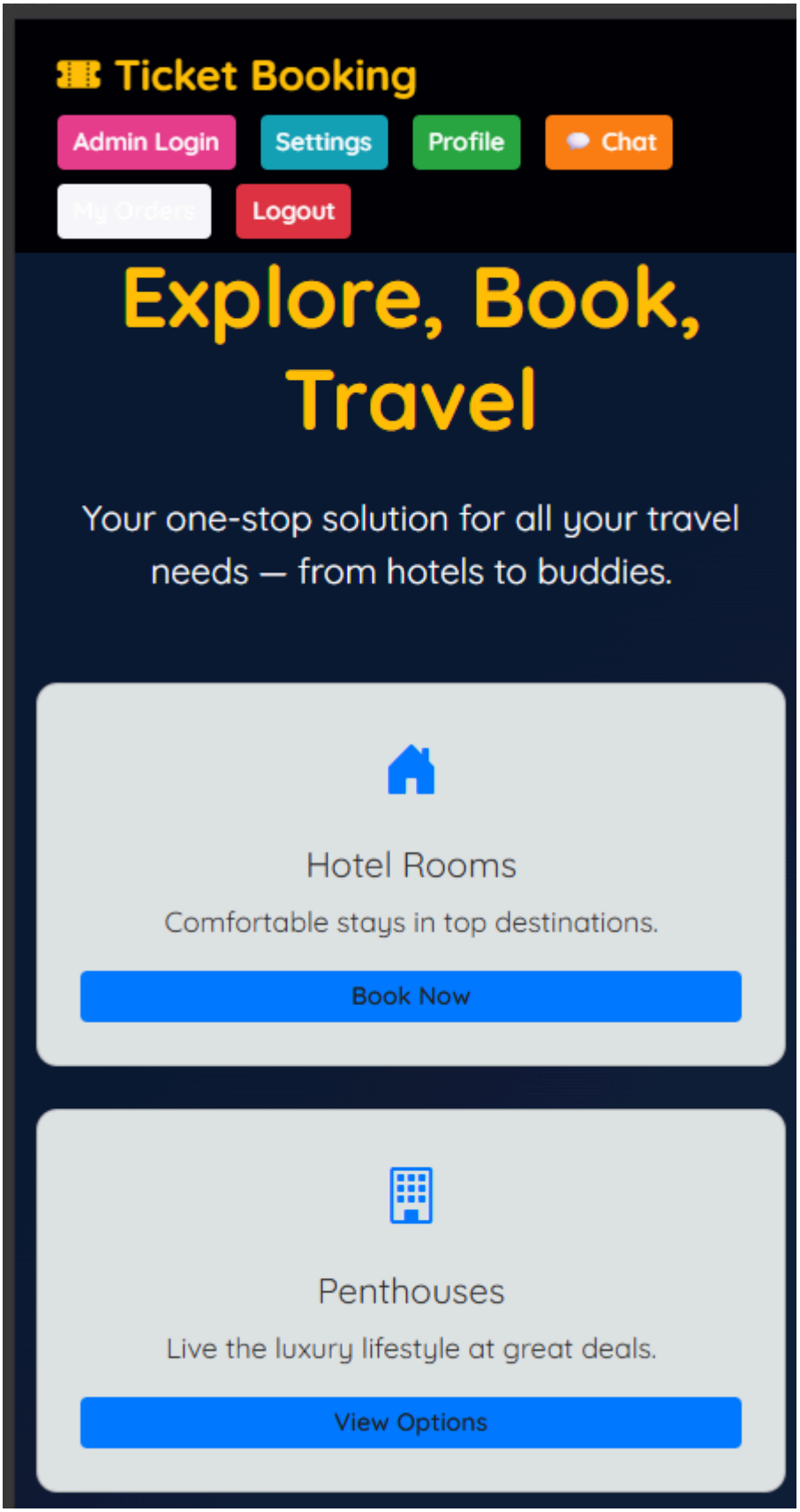
Lighthouse:

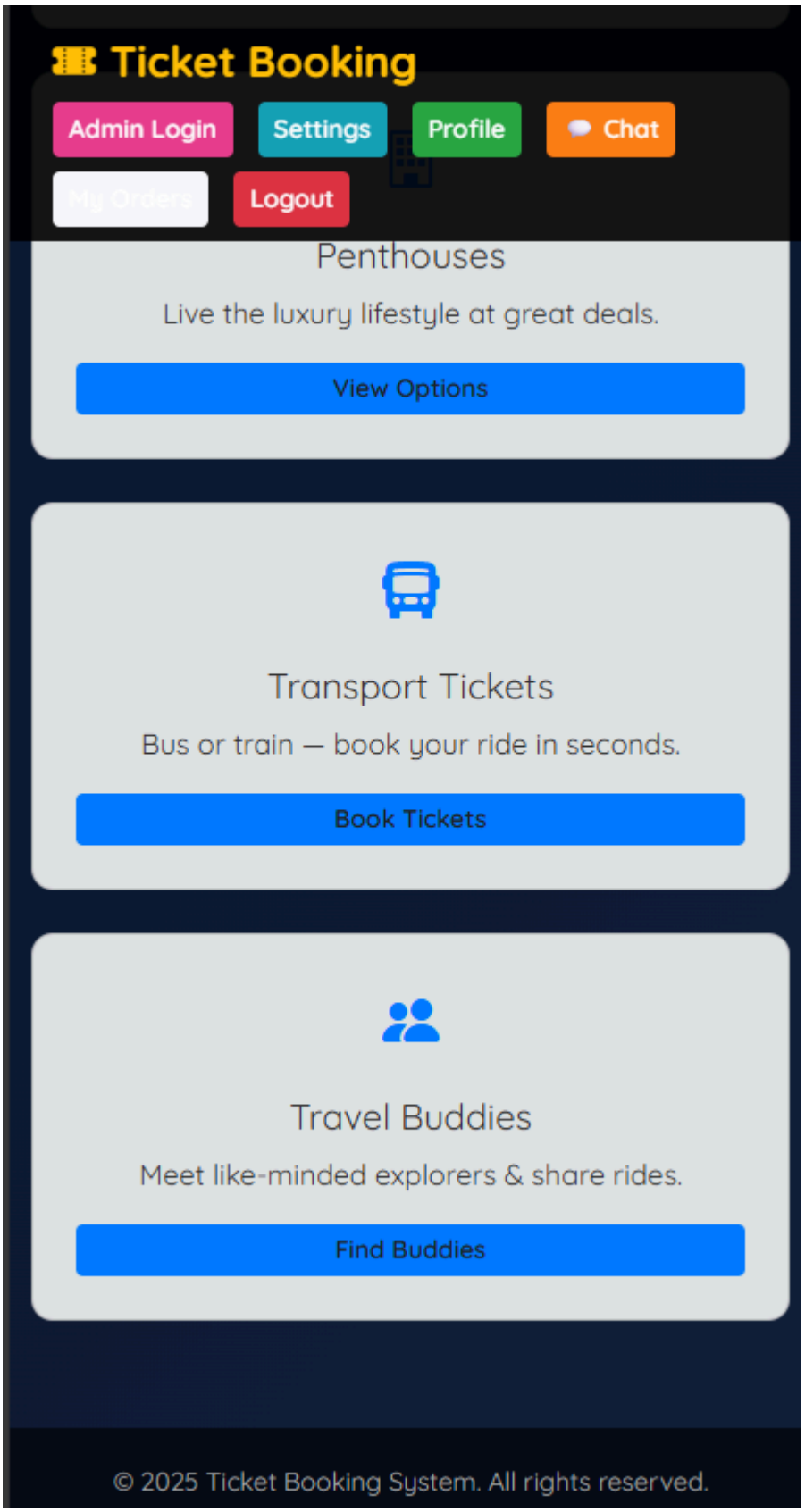

12:24:44 AM - 127.0.0.1:8800
http://127.0.0.1:8800/home#
95
Performance
85
Accessibility
100
Best Practices
90
SEO
There were issues affecting this run of Lighthouse:
Chrome extensions negatively affected this page's load performance. Try auditing the page in incognito mode or from a Chrome profile without extensions.
95
Performance
Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.
0–49
50–89
90–100
Explore, Book, Travel
METRICS
Expand view

Network Analysis:



Mobile View:


Ticket Booking
Admin Login
Settings
Profile
Chat
My Orders
Logout
Explore, Book, Travel
Your one-stop solution for all your travel needs — from hotels to buddies.
Hotel Rooms
Comfortable stays in top destinations.
Book Now
Penthouses
Live the luxury lifestyle at great deals.
View Options



9. Github Repo [Public] Link

https://github.com/irfanhossain57/Booking-System_Flask.git

10. Link of Deployed Project

https://allinonebookingsystem-16judl2z5-irfanhossain57s-projects.vercel.app

11. Individual Contribution

Group member - 01	
Name: Md. Zobayer	Student ID: 22101518
Functional Requirements which are developed by this member:	
1.Users can access a flexible transportation system with both bus and car options, view schedules, and easily print their tickets online.	
2.Admins can manage the entire branch network by adding new branches, updating existing ones, or removing outdated entries.	
3.Admins have the ability to create bus routes and assign multiple buses with varied departure times to the same route for better service.	
4.Admins can post rental cars of different types, sizes, and seating capacities, allowing users to select vehicles as per their needs.	

Group member - 02	
Name: Mahinoor Rahman	Student ID:24341209
Functional Requirements which are developed by this member:	
1.Users can reach out through a “Contact Us” page and use a real-time chatbox feature for support.	
2.The admin panel includes access to dashboard controls, profile management, and system display settings.	
3.Admins can approve or reject user transportation bookings, including both car and bus orders.	
4.Admins can view all registered users, delete accounts if needed, and assign or update admin privileges.	

Group member - 03	
Name: Sharmila Rani Saha	Student ID:24341210
Functional Requirements which are developed by this member:	

1.Users can sign up, log in, log out, and gain access to a personalized dashboard.
2.Users can manage and update profile details and see their information displayed on the profile page.
3.Users have the ability to book any available penthouse accommodation.
4.Admins can oversee and manage all travel buddy posts and penthouse listings from the backend.

Group member - 04	
Name: Md. Irfan Hossain	Student ID:24341069
Functional Requirements which are developed by this member:	
1.Users can book a wide range of hotel rooms as per their preference.	
2.Any user can create and publish travel buddy posts—either solo or as a group.	
3.Admins can add or remove hotel rooms of various types, which are then visible on the user dashboard.	
4.Admins also have the ability to add penthouses, and these listings appear directly on the user dashboard.	

12. References:

1. Flask: The web application for this project was constructed using Flask, a lightweight Python web framework. It manages routing, requests, and server-side logic. Because of Flask's ease of use, adaptive web applications can easily be developed quickly.
   URL: https://flask.palletsprojects.com/en/stable/

2. Python: The backend of this application was constructed using Python as the programming language. It is an excellent option for web development due to its wide environment and ease of use.
   URL: https://docs.python.org/3/

3. HTML: The foundation of the frontend is HTML, which gives the web pages structure. All of the fundamental HTML components and how to use them in web development are covered in this project.
   URL: https://developer.mozilla.org/en-US/docs/Web/HTML