TABLE BOOKING APP

INTRODUCTION

The Table Booking App is a digital platform designed to simplify and streamline the process of making table reservations at various hotels and restaurants. It provides a convenient and user-friendly solution for both users and hotel administrators to manage bookings effectively.

With the Table Booking App, users can easily discover and explore a list of hotels and restaurants available for table bookings. They can browse through the options, view details about each establishment, and make bookings based on their preferences. The app allows users to provide essential details such as their name, contact information, date of reservation, and the number of guests.

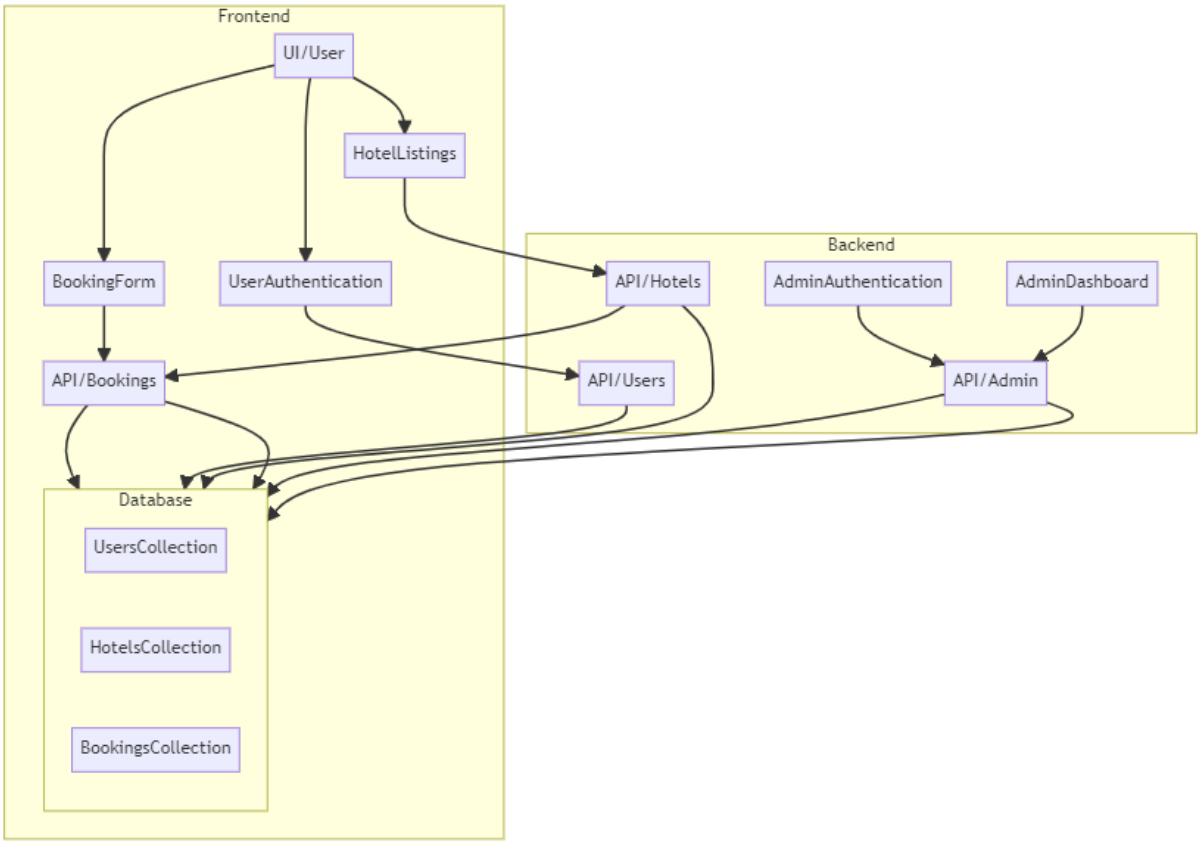
Once a booking is confirmed, users can access a dedicated booking details page to review their reservation information. This page provides them with an overview of their current and previous bookings, allowing them to keep track of their dining plans effortlessly.

On the other hand, hotel administrators have access to an intuitive admin dashboard. From there, they can manage and oversee the bookings received at their respective hotels. The dashboard enables administrators to add new hotels to the platform, view the list of added hotels, and monitor the bookings made by users. Each hotel is assigned a separate login and registration page, ensuring privacy and security by limiting access to specific hotel-related data.

Furthermore, the Table Booking App offers a comprehensive super admin functionality. Super admins have the ability to add and manage multiple hotels, view all the bookings across the platform, monitor user activity, and access an overview of all users using the app. This centralized control allows for efficient management and coordination of the table booking process.

Overall, the Table Booking App aims to enhance the dining experience for users by providing a seamless and convenient way to reserve tables at their preferred hotels and restaurants. By offering a user-friendly interface, efficient booking management, and robust administrative features, the app ensures a hassle-free and enjoyable dining experience for both users and hotel administrators alike

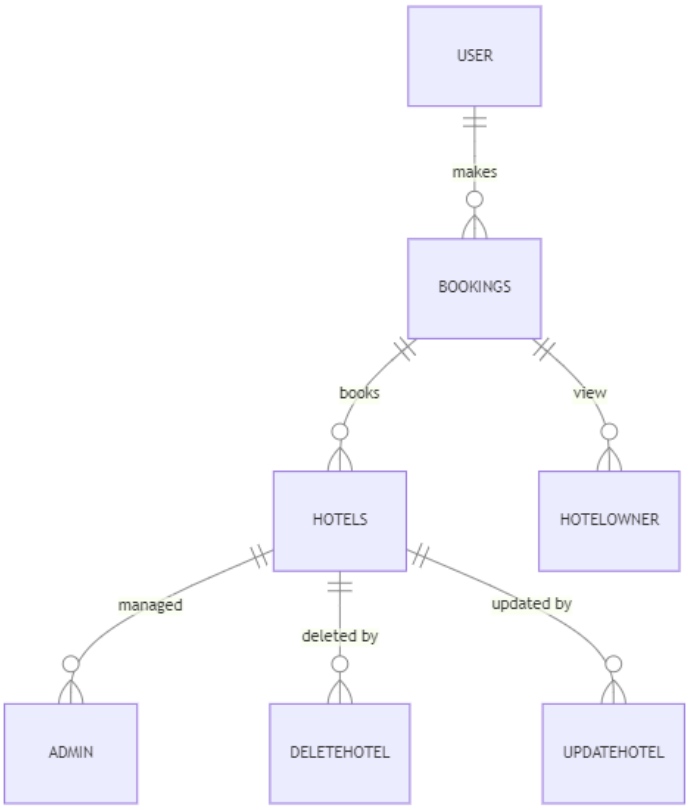
TECHINICAL ARCHITECTURE:



In this architecture diagram:

- The frontend is represented by the "Frontend" section, including user interface components such as User Authentication, Hotel Listings, and Booking Form.
- The backend is represented by the "Backend" section, consisting of API endpoints for Users, Hotels, and Bookings. It also includes Admin Authentication and an Admin Dashboard.
- The Database section represents the database that stores collections for Users, Hotels, and Bookings.

ER DIAGRAM:



- User: Represents the users of the app who can make bookings.
- Booking: Represents the bookings made by users at hotels.
- Hotel: Represents the hotels available for booking.
- DeleteHotel: Represents the deletion of a hotel.
- UpdateHotel: Represents the update of a hotel.
- HotelOwner: Represents the owner of a hotel.

Relations:

- User "Makes" a Booking: Indicates that a user can make multiple bookings.
- Hotel "Books" a Booking: Indicates that a hotel can have multiple bookings.
- Hotel "Managed" by Admin: Indicates that an admin manages multiple hotels.
- Hotel "Deleted by" DeleteHotel: Indicates that a hotel can be deleted by the DeleteHotel functionality.
- Hotel "Updated by" UpdateHotel: Indicates that a hotel can be updated by the UpdateHotel functionality.
- Booking "Viewed by" HotelOwner: Indicates that a hotel owner can view multiple bookings.

Features:

1. **Extensive Hotel Listing**: Our app offers an extensive list of hotels, ranging from casual dining to fine dining establishments. You can browse through the list and explore different cuisines, locations, and ambiances to find the perfect place for your dining experience.
2. **Booking Button:** Each hotel listing includes a convenient "Book" button. When you find a restaurant that catches your interest, simply click on the button to proceed with the reservation process.
3. **Booking Details:** Upon clicking the "Book" button, you will be directed to a booking details page. Here, you can provide relevant information such as the date, time, number of guests, and any specific preferences or requirements you may have.
4. **Secure and Efficient Booking Process:** Our app ensures a secure and efficient booking process. Your personal information will be handled with the utmost care, and we strive to make the reservation process as quick and hassle-free as possible.
5. **Confirmation and Booking Details Page:** Once you've successfully made a reservation, you will receive a confirmation message. You will then be redirected to a booking details page, where you can review all the relevant information about your booking, including the date, time, number of guests, and any special requests you made.

In addition to the user-facing features, our table booking app also includes a powerful admin dashboard, offering administrators a range of functionalities to manage the system efficiently. With the admin dashboard, admins can add hotels, view the list of added hotels, monitor user activity, and access booking details for all hotels. Furthermore, each hotel has its own separate login and register page, allowing them to view bookings specific to their establishment. Let's explore these features in more detail:

Admin Dashboard Features:

1. **Hotel Management:** The admin dashboard allows administrators to add new hotels to the app. They can input essential details such as hotel name, location, contact information, and any other relevant information. This feature ensures that the app stays updated with the latest and most accurate hotel information.

2. **Hotel Listing:** Administrators can view a comprehensive list of all the hotels that have been added to the app. This overview provides a quick reference for the available options, making it easier to manage and make changes as necessary.
3. **User Monitoring:** The admin dashboard provides insights into user activity. Admins can view and analyze data related to the number of users using the app, their preferences, and any feedback they may have provided. This information helps in making informed decisions and enhancing the overall user experience.
4. **Booking Overview:** Admins have access to the booking details for all the hotels listed in the app. They can view and manage reservations made by users, ensuring smooth operations and addressing any issues that may arise. This feature allows admins to keep track of bookings across multiple hotels from a centralized location.

Hotel-specific Features:

1. **Separate Login and Register Pages:** Each hotel listed in the app has its own dedicated login and register page. Hotel owners or authorized staff members can create accounts or log in to access their specific hotel's information and functionalities.

2. **Booking Management:** Once logged in, hotel owners/staff can view and manage bookings specific to their establishment. They can see the details of reservations made, including the date, time, number of guests, and any special requests. This feature streamlines the hotel's booking process and ensures that the staff is well-informed and prepared for incoming guests.

PRE REQUISITES:

To develop a full-stack e-commerce app using AngularJS, Node.js, and MongoDB, there are several prerequisites you should consider. Here are the key prerequisites for developing such an application:

Node.js and npm: Install Node.js, which includes npm (Node Package Manager), on your development machine. Node.js is required to run JavaScript on the server side.

- Download: https://nodejs.org/en/download/
- Installation instructions: https://nodejs.org/en/download/package-manager/

MongoDB: Set up a MongoDB database to store hotel and booking information. Install MongoDB locally or use a cloud-based MongoDB service.

- Download: https://www.mongodb.com/try/download/community
- Installation instructions: https://docs.mongodb.com/manual/installation/

Express.js: Express.js is a web application framework for Node.js. Install Express.js to handle server-side routing, middleware, and API development.

- Installation: Open your command prompt or terminal and run the following command: **npm install express**

Angular: Angular is a JavaScript framework for building client-side applications. Install Angular CLI (Command Line Interface) globally to create and manage your Angular project.

Install Angular CLI:

- Angular provides a command-line interface (CLI) tool that helps with project setup and development.
- Install the Angular CLI globally by running the following command:
  npm install -g @angular/cli

Verify the Angular CLI installation:

- Run the following command to verify that the Angular CLI is installed correctly:
  ng version

You should see the version of the Angular CLI printed in the terminal if the installation was successful.

Create a new Angular project:

- Choose or create a directory where you want to set up your Angular project.
- Open your terminal or command prompt.
- Navigate to the selected directory using the cd command.
- Create a new Angular project by running the following command:
  ng new client

Wait for the project to be created:

- The Angular CLI will generate the basic project structure and install the necessary de-pendencies.

- This process may take a few minutes, depending on your internet speed.

Navigate into the project directory:

- After the project creation is complete, navigate into the project directory by running thefollowing command:
  cd client

Start the development server:

- To launch the development server and see your Angular app in the browser, run the fol-lowing command:
  ng serve
- The Angular CLI will compile your app and start the development server.
- Open your web browser and navigate to http://localhost:4200 to see your Angular apprunning.

You have successfully set up Angular on your machine and created a new An- gular project. You can nowstart building your app by modifying the generated project files in the src directory.

Please note that these instructions provide a basic setup for Angular. You can explore more ad- vancedconfigurations and features by referring to the official Angular documentation: https://angular.io

HTML, CSS, and JavaScript: Basic knowledge of HTML for creating the structure of your app, CSS for styling,and JavaScript for client-side interactivity is essential.

Database Connectivity: Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose toconnect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations.

Front-end Framework: Utilize Angular to build the user-facing part of the application, including productslistings, booking forms, and user interfaces for the admin dashboard.

Version Control: Use Git for version control, enabling collaboration and tracking changes throughout thedevelopment process. Platforms like GitHub or Bitbucket can host your repository.

- Git: Download and installation instructions can be found at: https://git-scm.com/downloads

Development Environment: Choose a code editor or Integrated Development Environment (IDE) that suits yourpreferences, such as Visual Studio Code, Sublime Text, or WebStorm.

- Visual Studio Code: Download from https://code.visualstudio.com/download
- Sublime Text: Download from https://www.sublimetext.com/download
- WebStorm: Download from https://www.jetbrains.com/webstorm/download

To Connect the Database with Node JS go through the below provided link:

- Link: https://www.section.io/engineering-education/nodejs- mongoosejs-mongodb/

To run the existing Table Booking App project downloaded from github:

Follow below steps:

Clone the Repository:

- Open your terminal or command prompt.
- Navigate to the directory where you want to store the e-commerce app.
- Execute the following command to clone the repository:
  git clone https://github.com/samhithaerukulla/bookingwebsite

Install Dependencies:

- Navigate into the cloned repository directory:
  cd bookingwebsite-main

- Install the required dependencies by running the following command:
  npm install

Start the Development Server:

- To start the development server, execute the following command:
  npm run dev or npm run start

- The e-commerce app will be accessible at http://localhost:4200 by default. You can change the portconfiguration in the .env file if needed.

Access the App:

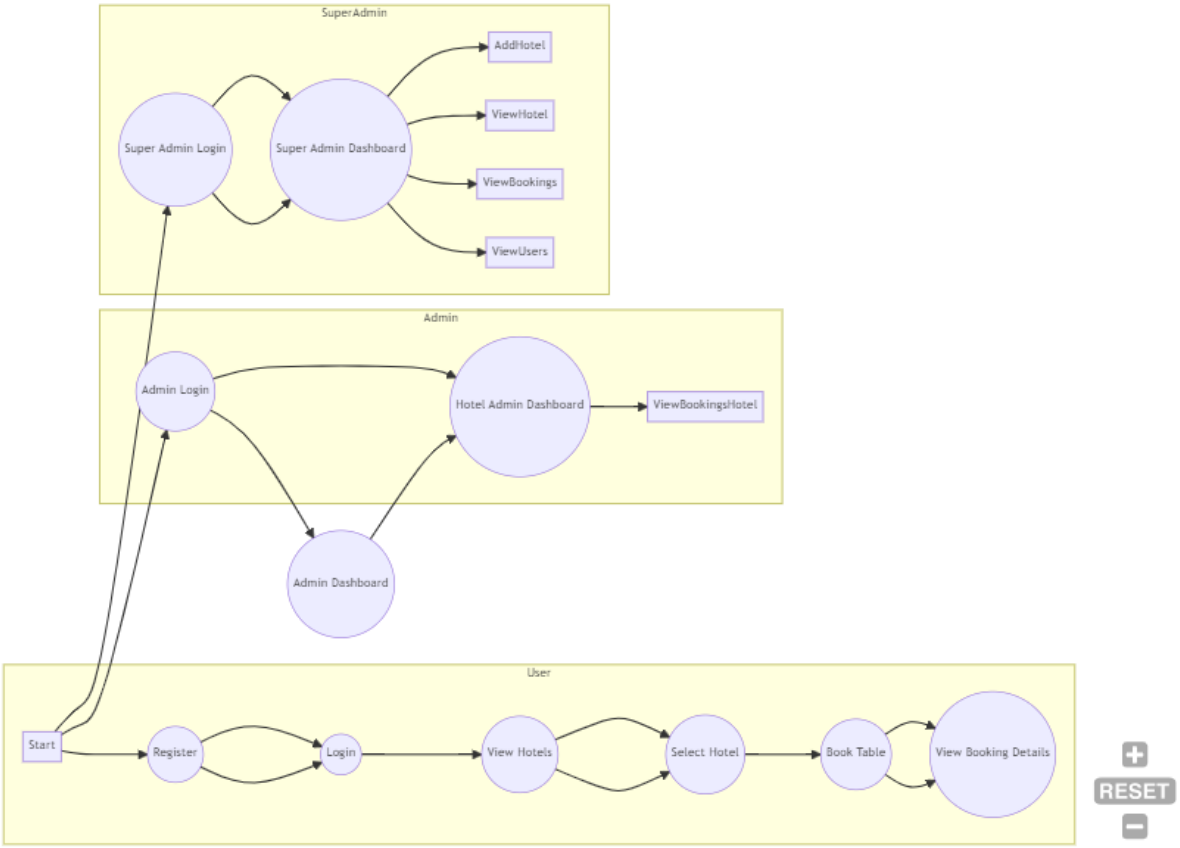
- Open your web browser and navigate to http://localhost:4200.
- You should see the e-commerce app's homepage, indicating that the installation and setup weresuccessful.

You have successfully installed and set up the table booking app on your local machine. You can now proceedwith further customization, development, and testing as needed.

git clone:

https://drive.google.com/drive/folders/16xR2BhEBVASOC63HI-ZQQuB3LHSSJlwY

USER & ADMIN FLOW:



1. **User Flow:**
    - Users start by registering for an account.
    - After registration, they can log in with their credentials.
    - Once logged in, they can view the available hotels.
    - Users can select a specific hotel from the list.
    - They can then proceed to book a table at the selected hotel.
    - After booking, they can view the details of their booking.

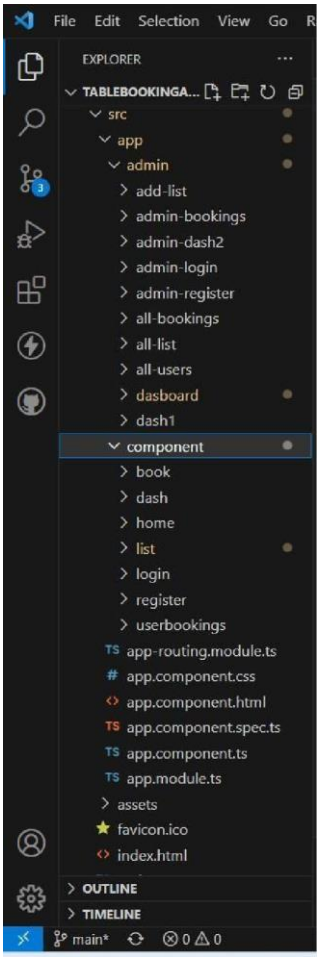
2. **Admin Flow:**
    - Admins start by logging in with their credentials.
    - Once logged in, they are directed to the Admin Dashboard.
    - Admins can access the Hotel Admin Dashboard, where they can view bookings specific to their hotel.

3. **Super Admin Flow:**

- Super Admins start by logging in with their credentials.
- Once logged in, they are directed to the Super Admin Dashboard.
- Super Admins have additional privileges, including the ability to addhotels,view hotels, view all bookings, and view all users.

4. **Database:**
    - The flowchart includes a Database subgraph representing the collections in thedatabase.
    - The UsersCollection stores user registration and login information.
    - The HotelsCollection stores hotel data for viewing and selection.
    - The BookingsCollection stores booking details for table reservations.

PROJECT STRUCTURE:



This structure assumes an Angular app and follows a modular approach. Here's a briefexplanation of the main directories and files:

- src/app/components: Contains components related to the customer app, such asregister, login, home, list,userbookings,dash,book and more.
- src/app/admin: Contains modules for different sections of the app. In this case, the admin module is included with its own set of components like addlist,admin-bookings,admindash2,admin-login,admin-register,all-bookings,alllist,all- users,and more.
- src/app/app-routing.module.ts: Defines the routing configuration for the app,specifying which components should be loaded for each route.
- src/app/app.component.ts, src/app/app.component.html, `src.

Project Flow:

Milestone 1: Project Setup and Configuration: (Record Video & Give Links)

1. **Install required tools and software:**
    - Node.js.
    - MongoDB.
    - Angular CLI.
    - Git.
2. **Create project folders and files:**
    - Client folders.
    - Server folders

Milestone 2: Backend Development:(Record Video & Give Links)

1. **Setup express server:**
    - Install express.
    - Create app.js file.
    - Define API’s

2. **Configure MongoDB:**
   - Install Mongoose.
   - Create database connection.
3. **Implement API end points:**
   - Implement CRUD operations.
   - Test API endpoints.

Milestone 3: Web Development:(Record Video & Give Links)

1. **Setup Angular Application:**
   - Create Angular application using angular CLI.
   - Configure Routing.
   - Install required libraries.
2. **Design UI components:**
   - Create Components.
   - Implement layout and styling.
   - Add navigation.
3. **Implement frontend logic:**
   - Integration with API endpoints.
   - Implement data binding.

Create database in cloud video link:-
https://drive.google.com/file/d/1CQil5KzGnPvkVOPWTLP0h-Bu2bXhq7A3/view?usp=sharing

To Setup the frontend development and to connect node.js with MongoDB Database Go through this video link: -

https://drive.google.com/file/d/1b5bMvnqmASXLnSZ74B2t3EzNjuWHj63g/view?usp=drive_link

Backend:

1. **Set Up Project Structure:**
   - Create a new directory for your project and set up a package.json file using npm init command.
   - Install necessary dependencies such as Express.js, Mongoose, and other required packages.

2. **Database Configuration:**
   - Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas.
   - Create a database and define the necessary collections for hotels, users, bookings, and other relevant data.

3. **Create Express.js Server:**
   - Set up an Express.js server to handle HTTP requests and serve API endpoints.
   - Configure middleware such as body-parser for parsing request bodies and cors for handling cross-origin requests.

4. **Define API Routes:**
   - Create separate route files for different API functionalities such as hotels, users, bookings, and authentication.
   - Define the necessary routes for listing hotels, handling user registration and login, managing bookings, etc.
   - Implement route handlers using Express.js to handle requests and interact with the database.

5. **Implement Data Models:**
   - Define Mongoose schemas for the different data entities like hotels, users, and bookings.
   - Create corresponding Mongoose models to interact with the MongoDB database.
   - Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.

6. **User Authentication:**
   - Implement user authentication using strategies like JSON Web Tokens (JWT) or session-based authentication.

- Create routes and middleware for user registration, login, and logout.
- Set up authentication middleware to protect routes that require user authentication.

7. **Handle Hotel Listings and Bookings:**
   - Create routes and controllers to handle hotel listings, including fetching hotel data from the database and sending it as a response.
   - Implement booking functionality by creating routes and controllers to handle booking requests, including validation and database updates.

8. **Admin Functionality:**
   - Implement routes and controllers specific to admin functionalities such as adding hotels, managing user bookings, etc.
   - Add necessary authentication and authorization checks to ensure only authorized admins can access these routes.

9. **Error Handling:**
   - Implement error handling middleware to catch and handle any errors that occur during the API requests.
   - Return appropriate error responses with relevant error messages and HTTP status codes.

Schema usecase:

1. **User Schema:**
   - Schema: userSchema
   - Model: ‘User’
   - The User schema represents the user data and includes fields such as firstname, lastname, email, and password.
   - It is used to store user information for registration and authentication purposes.

   The email field is marked as unique to ensure that each user has a unique email address.

2. **Hotel Schema:**
   - Schema: hotelSchema
   - Model: ‘Hotel’
   - The Hotel schema represents the hotel data and includes fields such as HotelName,

Address, City, Postalcode, image, and rating.

- It is used to store information about hotels available for table bookings.
- The image field is a string that stores the URL or path to an image representing the hotel.
- The rating field is a string that can be used to store the rating or feedback for a particular hotel.

3. Booking Schema:

- Schema: BookingsSchema
- Model: ‘Booking’
- The Booking schema represents the booking data and includes fields such as userId, HotelName, name, email, date, time, and guests.
- It is used to store information about the table bookings made by users.
- The userId field is a reference to the user who made the booking.
- The HotelName field stores the name of the hotel for which the booking is made.
- The name and email fields store the contact information of the user making the booking.
- The date, time, and guests fields store the booking details such as the date of the reservation, time slot, and the number of guests.

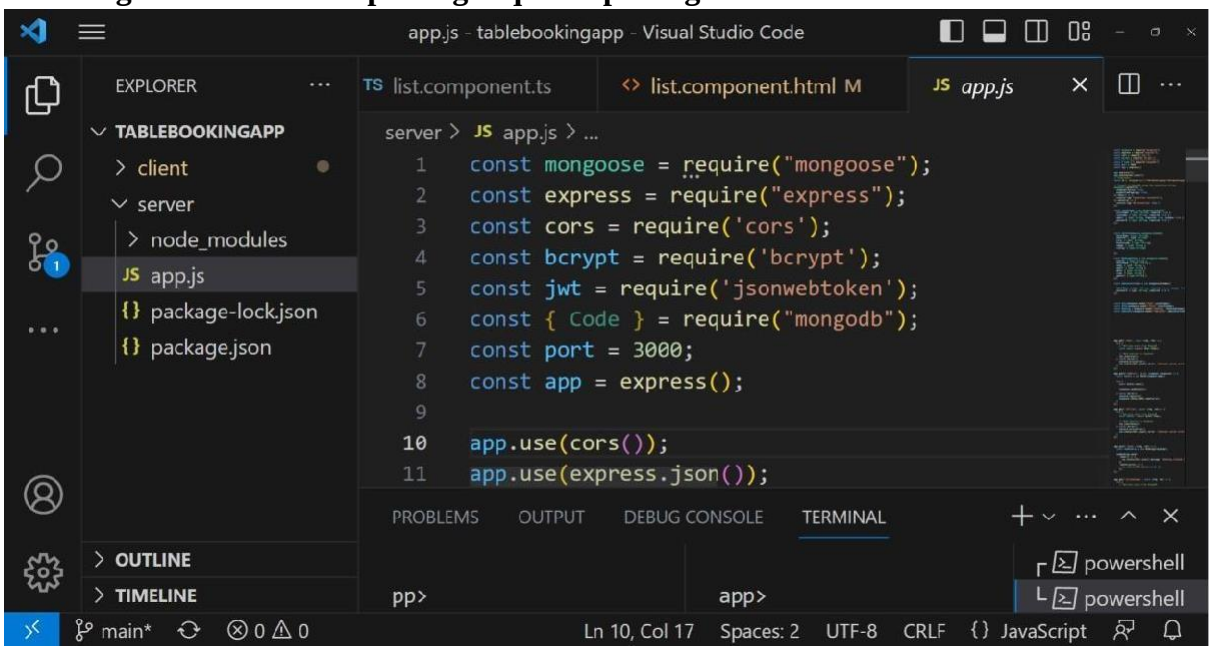
4. Admin User Schema:

- Schema: AdminuserSchema
- Model: ‘AdminUser’
- The Admin User schema represents the data for hotel administrators.
- It includes fields such as HotelName (unique) and password.
- This schema is used to store login credentials for hotel administrators and manage their access to the admin dashboard.

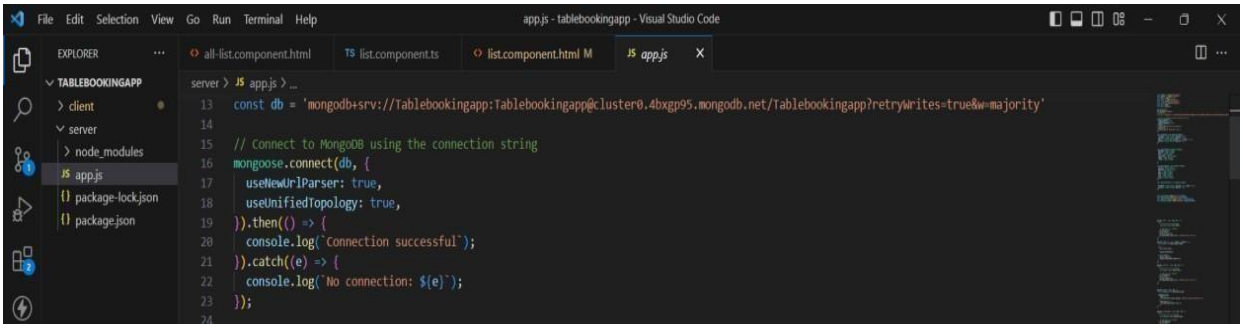
Backend Explanation with code snippets:

Code Snippets for API's:

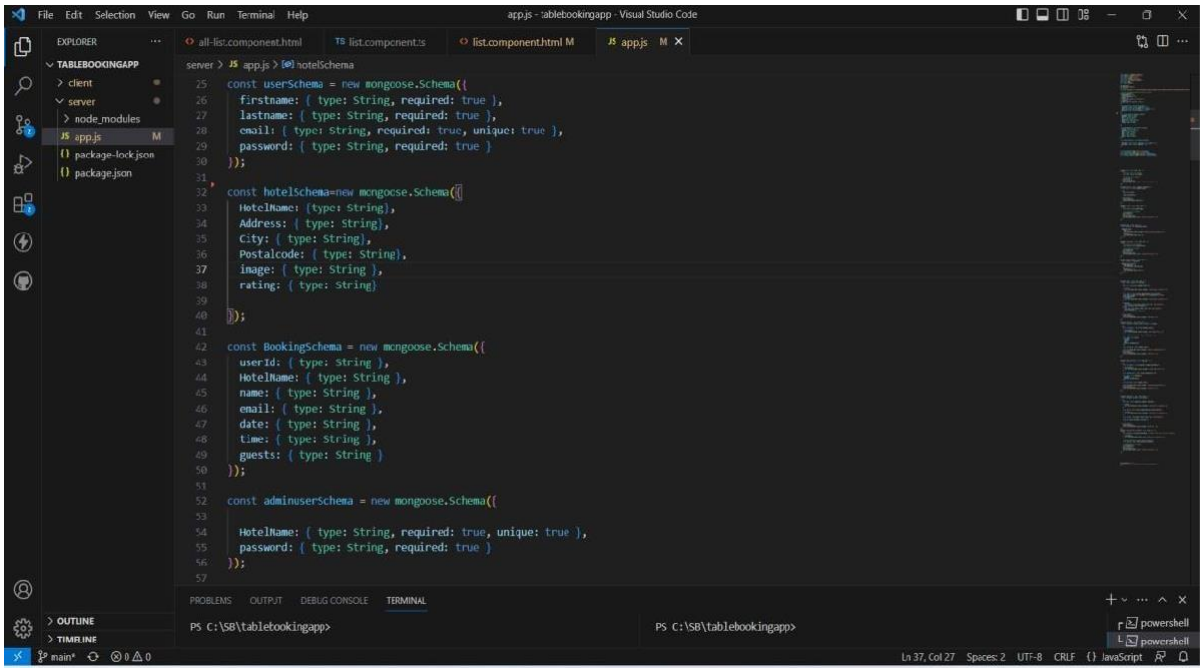
Handling with cors and importing required packages:

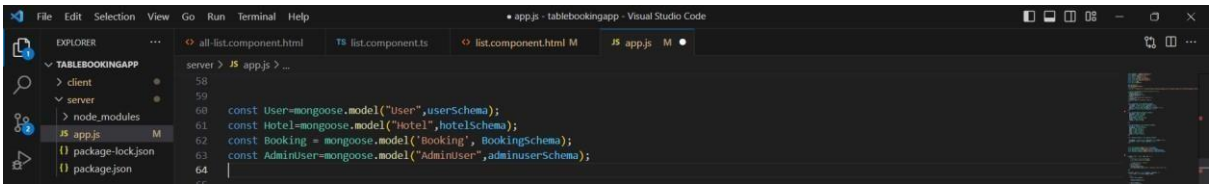


Database connection:



Schema:





API's for Register and get Users:

API for Register: Allows users to create an account by sending their registration information to the server.

API for Get Users: Retrieves a list of registered users from the server for further processing or display.

```
app.post('/register', async (req, res) => {
  const { firstname, lastname, email, password } = req.body;
  try {
    const existingUser = await User.findOne({ email });
    if (existingUser) {
      return res.status(400).json({ message: 'User already exists' });
    }
    const hashedPassword = await bcrypt.hash(password, 10);
    const newUser = new User({
      firstname,
      lastname,
      email,
      password: hashedPassword
    });
    const userCreated = await newUser.save();
    return res.status(201).json({ message: 'Successfully Registered' });
  } catch (error) {
    console.log(error);
    return res.status(500).json({ message: 'Server Error' });
  }
});

app.get('/user', async (req, res) => {
  try {
    // Retrieve users from MongoDB
    const users = await User.find();
    // Send users as a response
    res.json(users);
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal server error' });
  }
});
```

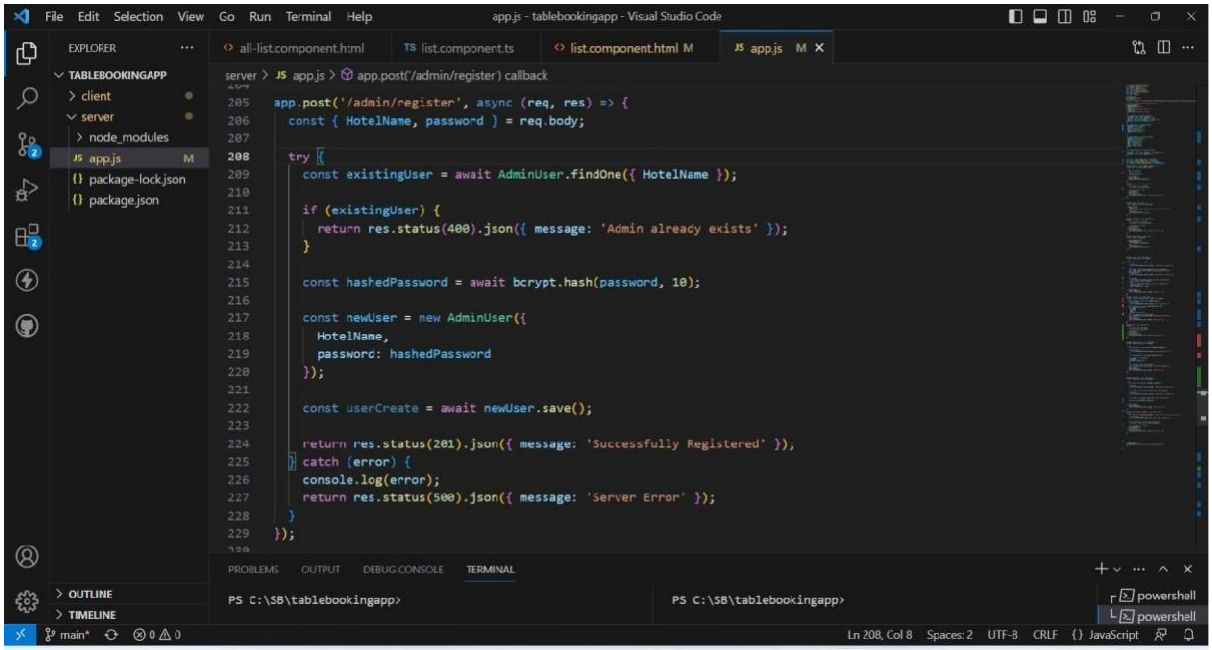
API for User Login:

Authenticates user credentials (username and password) against the server's database, granting access to the system if valid.

```
app.post('/login', async (req, res) => {
  const { email, password } = req.body;

  try {
    const user = await User.findOne({ email });

    if (!user) {
      return res.status(401).json({ message: 'Invalid email or password' });
    }

    const isMatch = await bcrypt.compare(password, user.password);
    const isAdmin = email === "admin@gmail.com" && password === "admin"
    console.log(isAdmin)
    if (!isMatch) {
      return res.status(401).json({ message: 'Invalid email or password' });
    }

    if (isAdmin) {
      const jwtToken = jwt.sign({ userId: user._id }, 'mysecretkey2');
      return res.json({ user, jwtToken });
    } else {
      const token = jwt.sign({ userId: user._id }, 'mysecretkey1');
      console.log(token)
      return res.json({ user, token });
    }

  } catch (error) {
    console.log(error);
    return res.status(500).json({ message: 'Server Error' });
  }
});
```

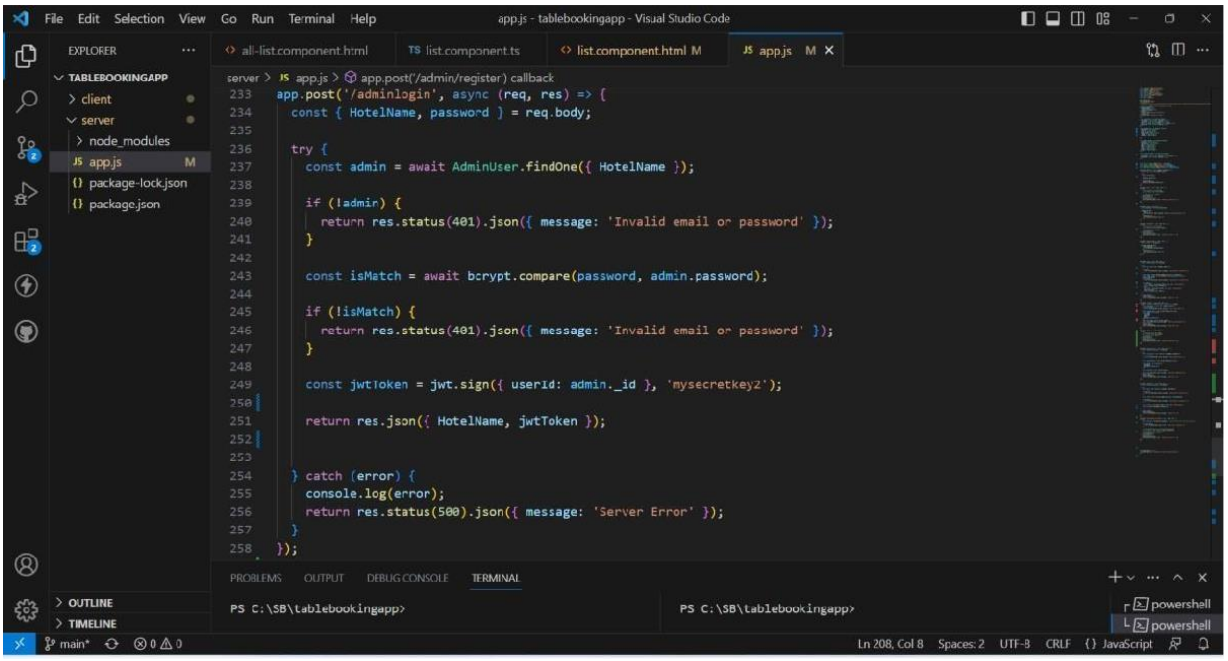
API for Admin Register:

Enables administrators to create a new admin account by providing necessary registration details, such as username, password, and admin privileges, which are stored in the server's database.



API for Admin Login:

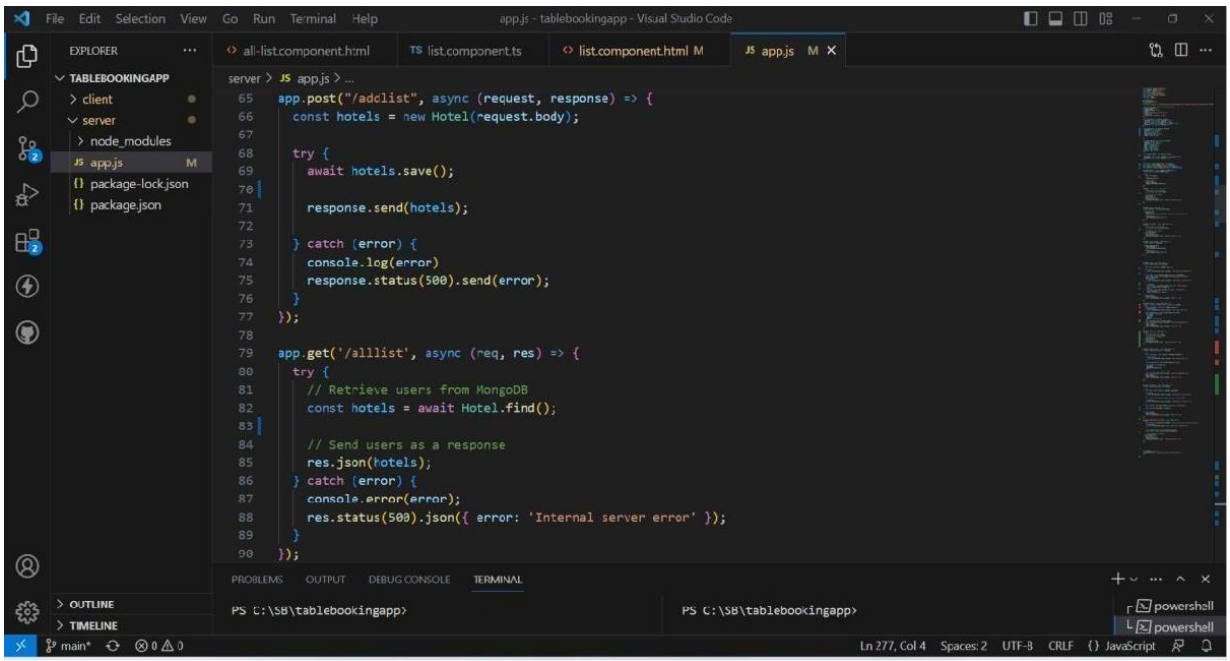
Verifies the credentials of an administrator (username and password) against the server's database, granting access to the administrative panel or privileged functionalities if the authentication is successful.



API'S for Post and Get Hotels:

API for Post Hotels: Allows the submission of hotel information, such as name, address, amenities, and photos, to the server for creating a new hotel listing.

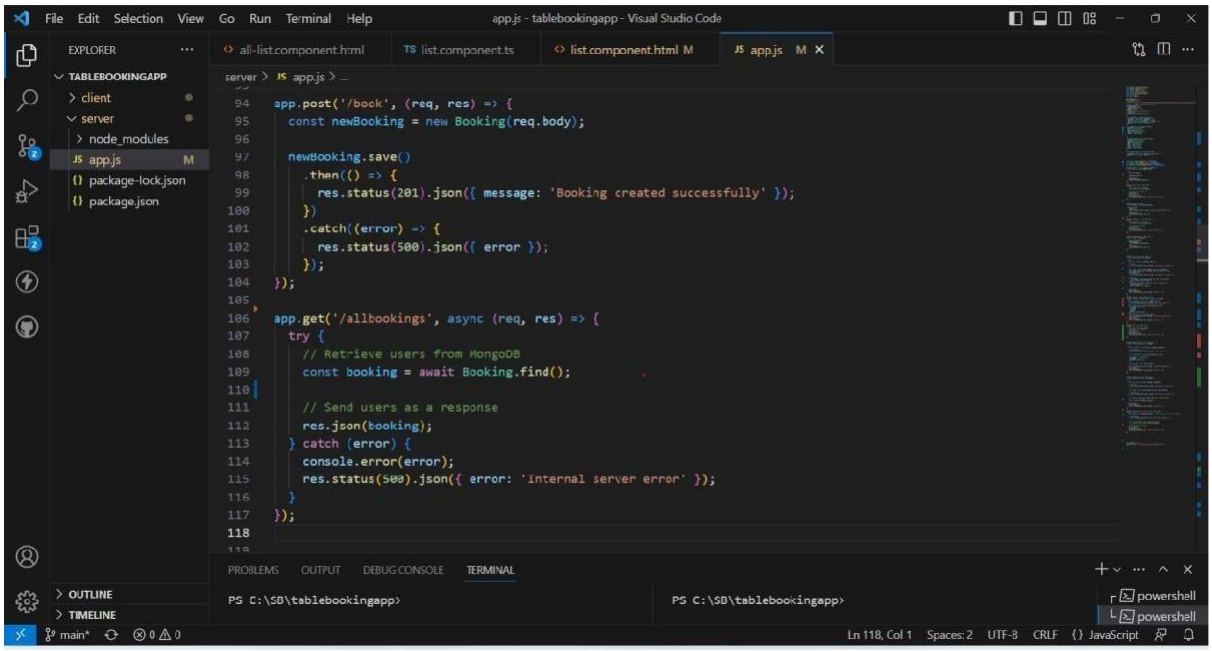
API for Get Hotels: Retrieves a list of hotels from the server, providing information like name, address, amenities, and photos, for displaying or further processing, such as searching or filtering based on specific criteria.



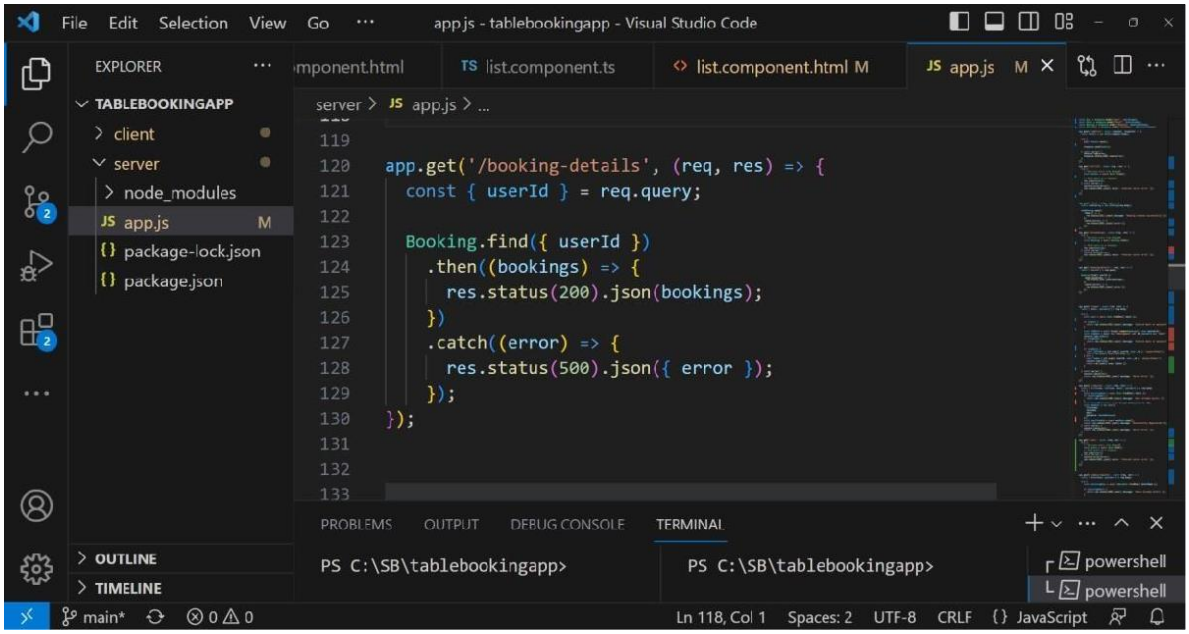
API'S for Post and Get Bookings:

API for Post Bookings: Enables users to make a booking by submitting relevant information such as check-in/out dates, guest details, and hotel ID to the server, creating a new booking record.

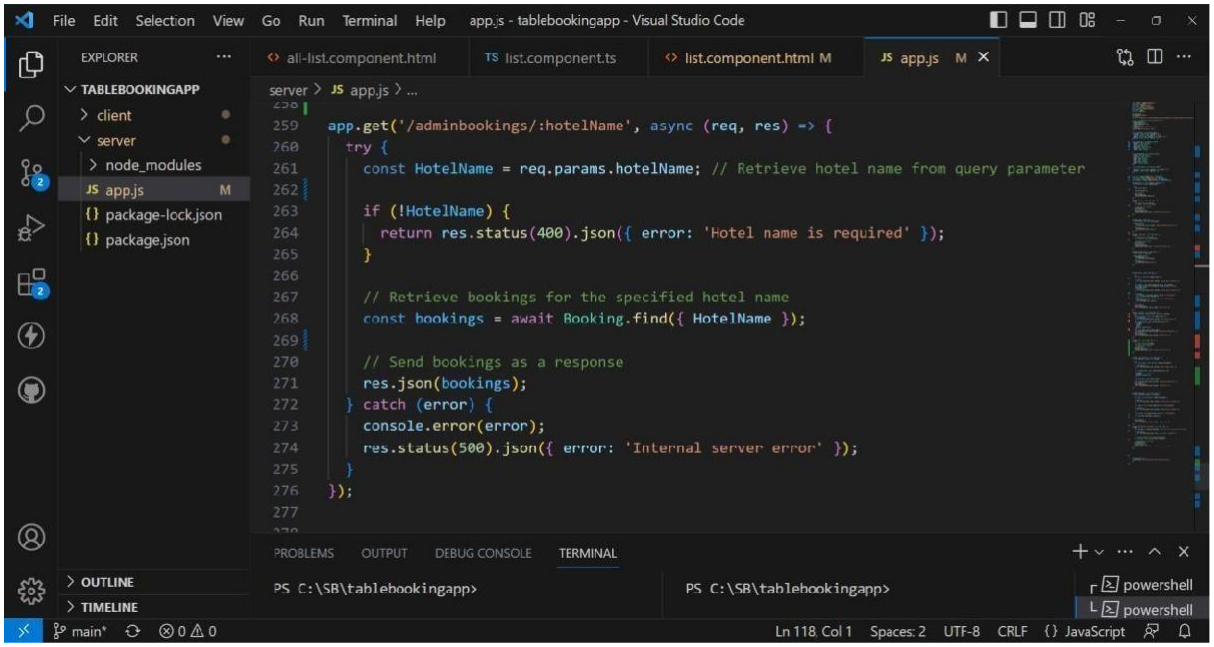
API for Get Bookings: Retrieves a list of bookings from the server, providing details such as booking ID, hotel information, guest details, and check-in/out dates, for displaying or further processing, such as viewing, modifying, or canceling existing bookings.



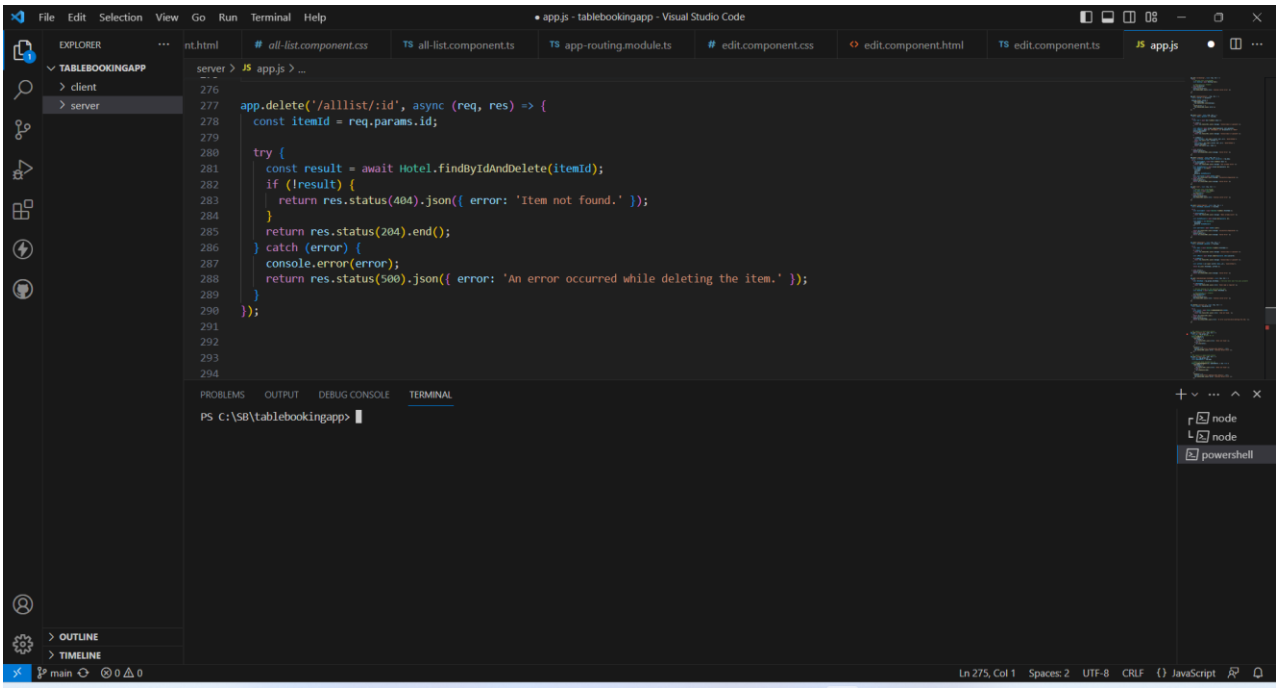
API for Get Bookings based on User ID: Retrieves a list of bookings associated with a specific user ID from the server, providing details such as booking ID, hotel information, guest details, and check-in/out dates, for displaying or further processing specific to that user's bookings.



API for Get Bookings based on Hotel Name: Retrieves a list of bookings associated with a specific hotel name from the server, providing details such as booking ID, user information, guest details, and check-in/out dates, for displaying or further processing specific to that particular hotel's bookings.

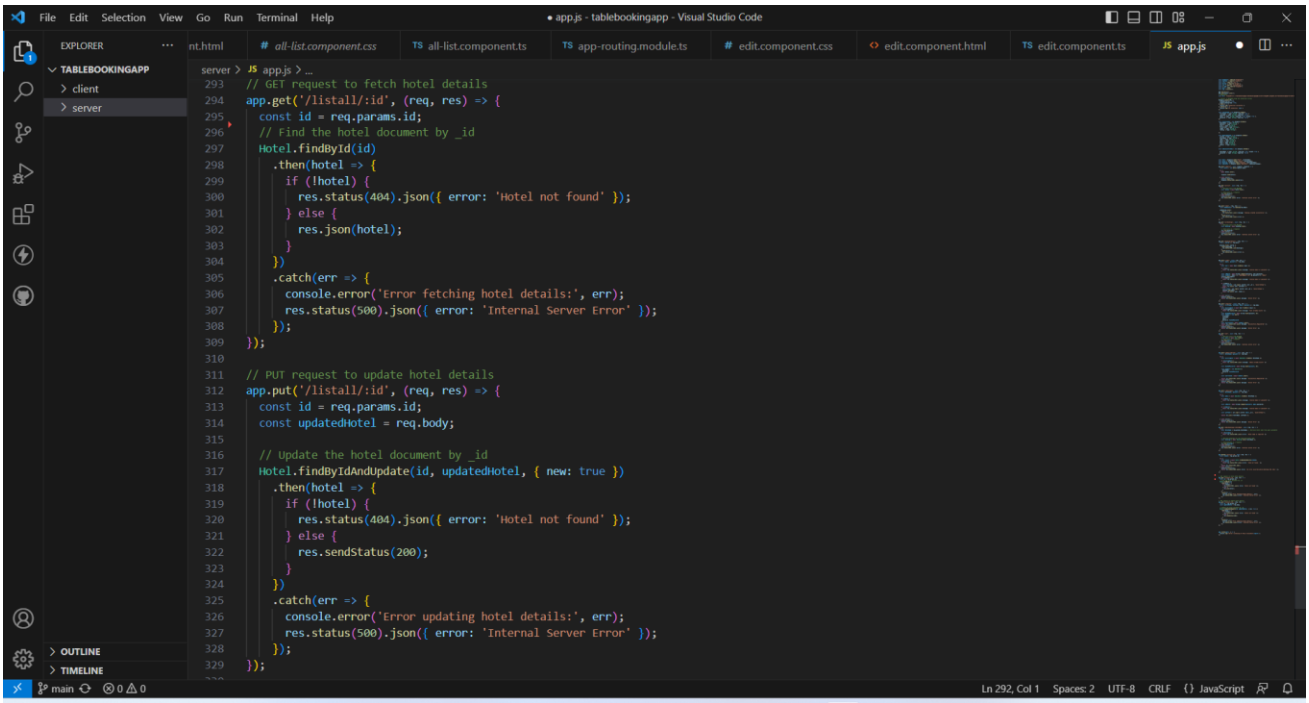


API to delete the hotel: To delete a hotel via an API, you can typically make a DELETE request to the appropriate endpoint or URL associated with the hotel resource. The API will then process the request and remove the hotel from the system, returning a success or confirmation response upon successful deletion.



API to get the hotel by id and update it:

To get a hotel by its ID and update it via an API, you can typically make a GET request to the specific endpoint or URL that includes the hotel's ID, retrieving the current information. Then, you can make a subsequent PUT or PATCH request to the same endpoint with the updated data, allowing the API to process the request and modify the hotel's details accordingly, returning a success response upon successful update.



FRONTEND:

User Login:

- Create a user login page where users can enter their credentials (username/email and password) to log in.
- Implement user authentication logic to verify the user's credentials against the database.
- Upon successful login, store the user's authentication token in the browser's session or local storage for future requests.

Hotel Listings:

- Design a page to display the list of hotels available for booking.
- Fetch the hotel data from the backend API based on the user's authentication status.
- Render the hotel listings dynamically on the page, including relevant information such as hotel name, location, and available dates.

Hotel Selection and Booking:

- Allow the user to select a specific hotel from the displayed listings.
- Provide a booking form where the user can enter their details, including name, email, preferred date, and number of guests.
- Validate the form input and send the booking request to the backend API to create a new booking entry in the database.

Booking Confirmation and Redirect:

- Upon successful booking, display a confirmation message to the user.
- Redirect the user to a booking details page where they can see the details of their current and previous bookings.
- Retrieve the booking information from the backend API and display it on the page.

Super Admin Page:

- Design a separate admin page accessible only to super admins.
- Implement authentication and authorization checks to ensure only super admins can access this page.
- Create sections to add and display hotels added, list all users using the app, and view bookings of all hotels.
- Fetch the required data from the backend API and display it on the respective sections of the super admin page.

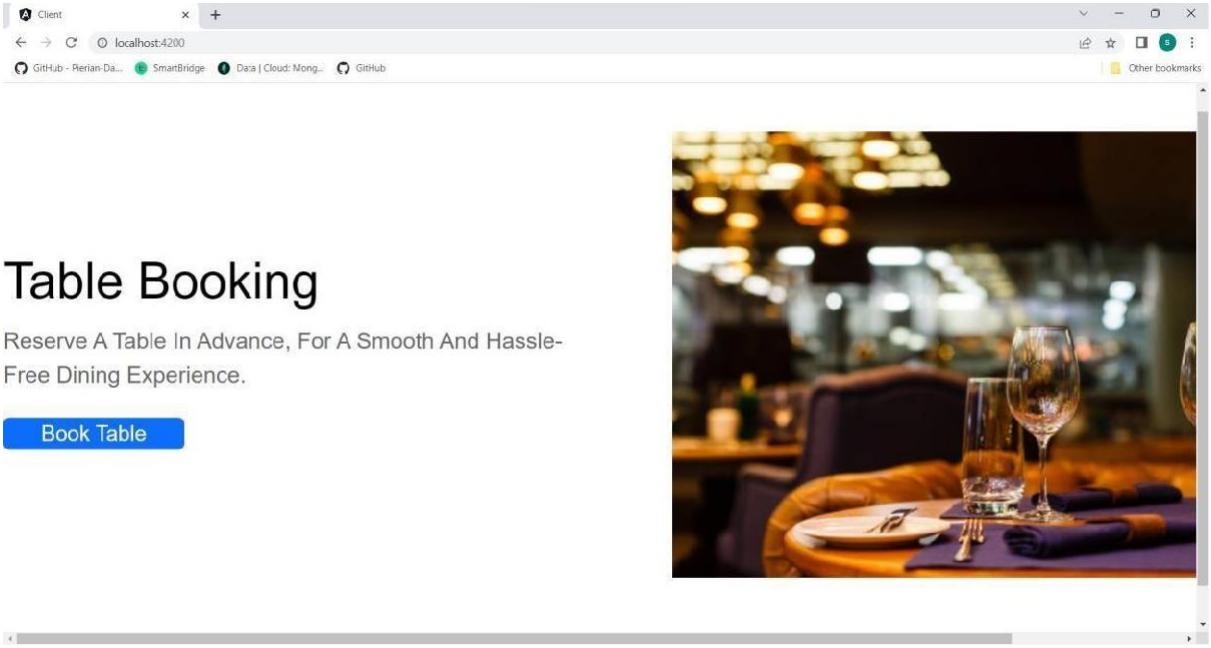
Separate Admin Register and Login Pages:

- Design dedicated register and login pages for individual hotel admins.
- Implement authentication logic specific to hotel admins, verifying their credentials against the database.
- After successful login, store the admin's authentication token in the browser's session or local storage for future requests.

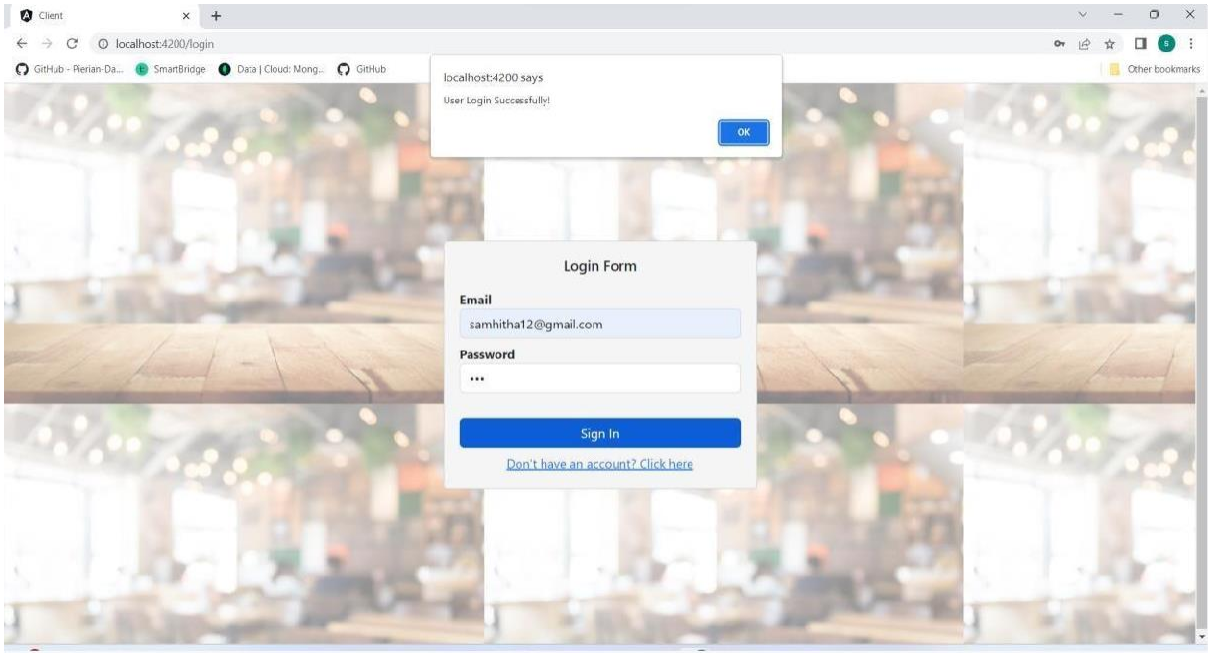
Admin Bookings Display:

- Create a separate admin dashboard or bookings page for hotel admins.
- Fetch thebookings specific to the admin's hotel(s) from the backend API.
- Display the bookings on the page, including relevant details such as guest name, booking dates, and guest count

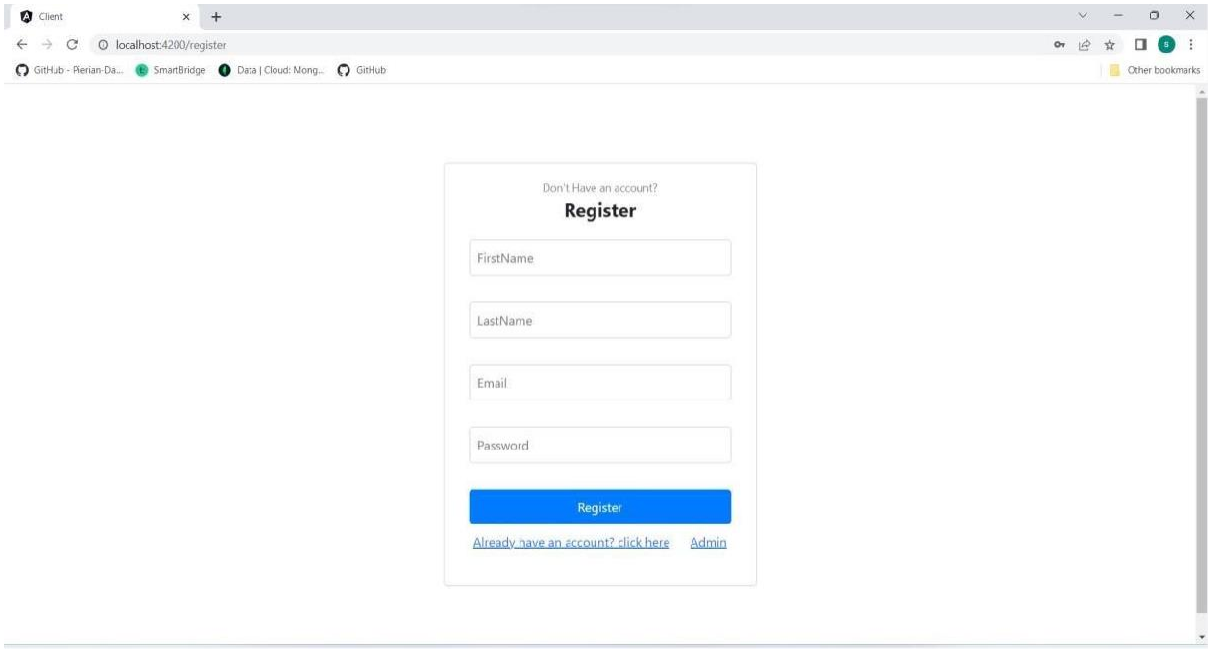
Landing Page:



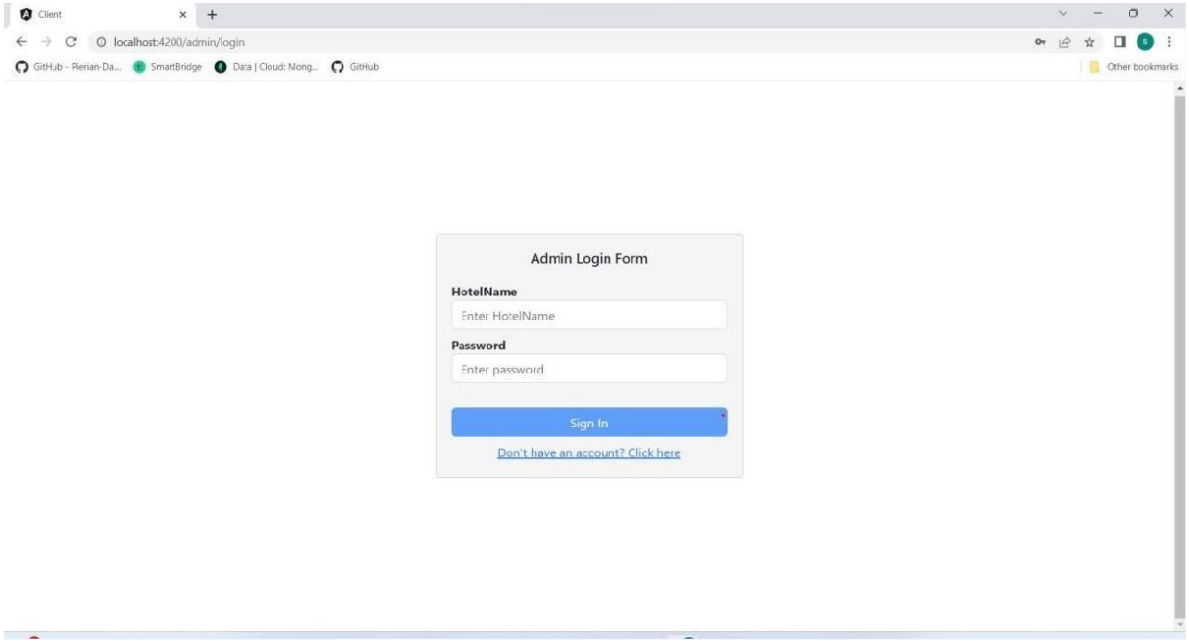
User Login: login page for user to login into application.



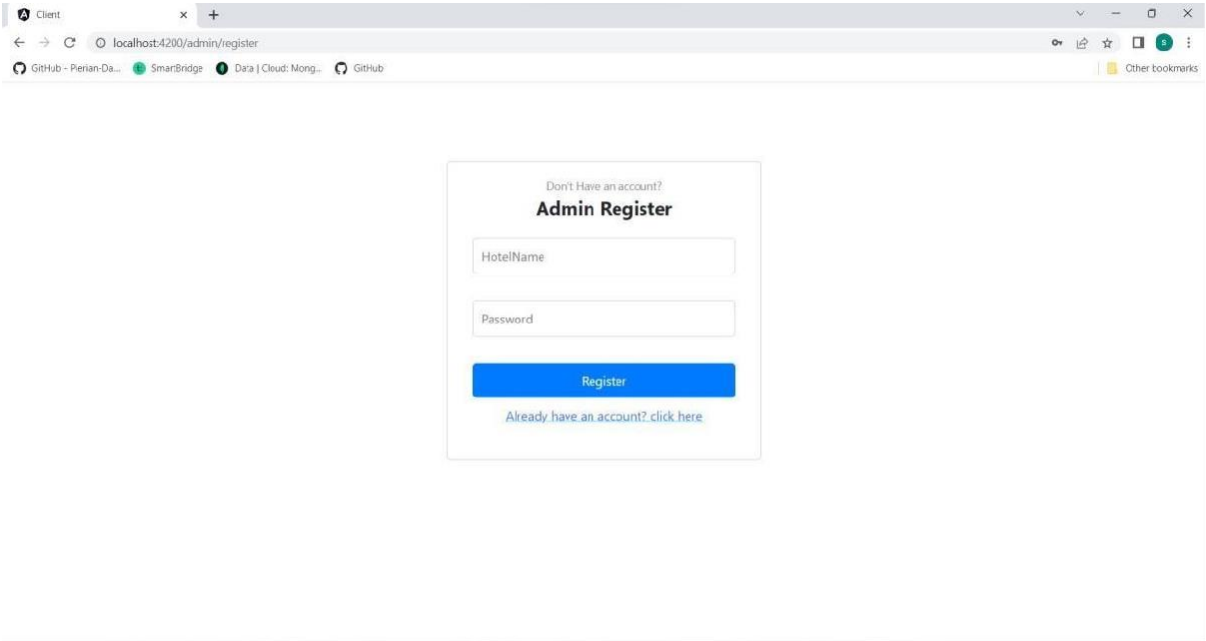
User Registration: registration page for new user to register.



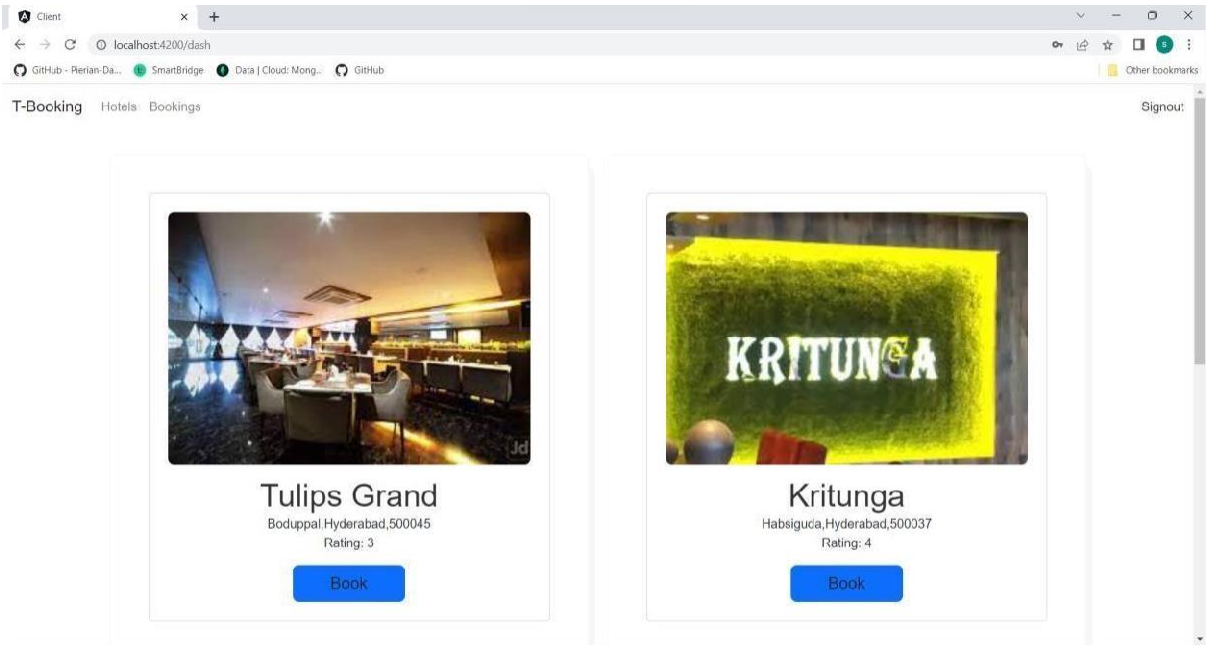
Admin login: admin login page for admin to login.



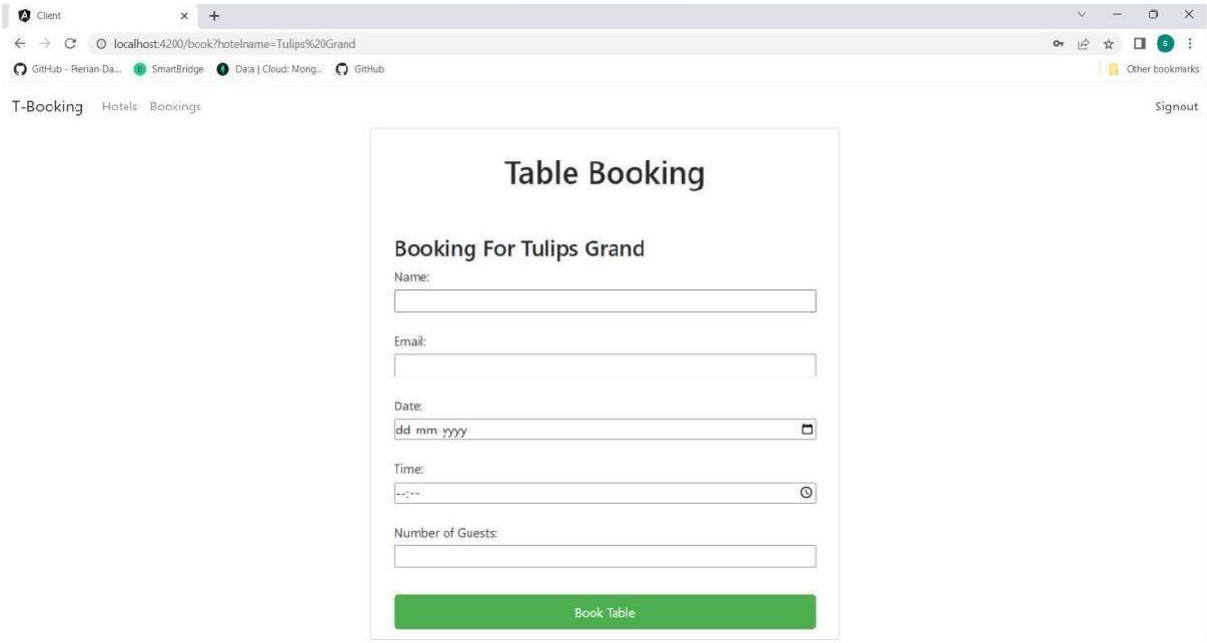
Admin Register: Registration Page for admin to register.



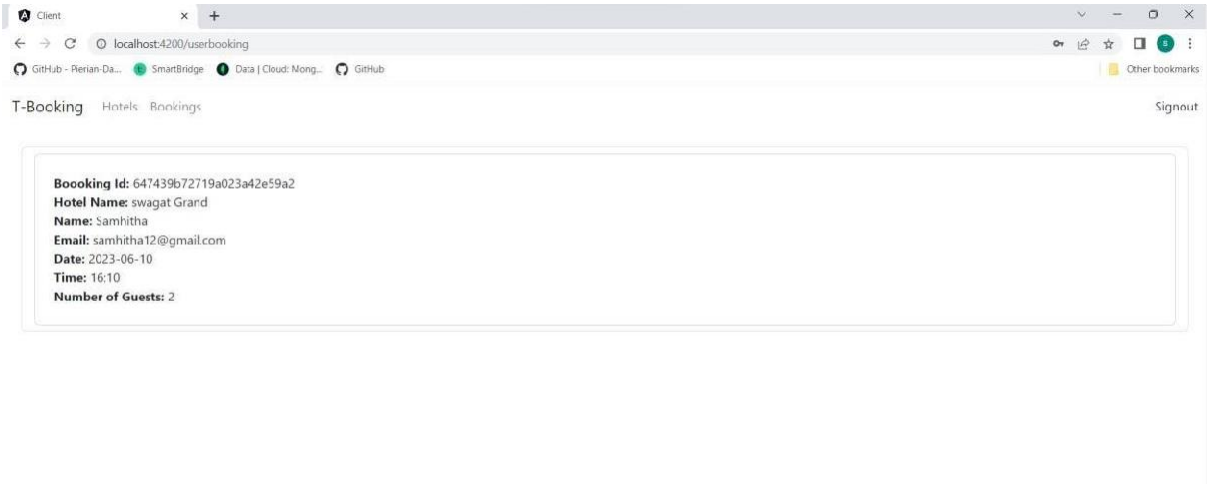
Hotels List: List of hotels to book the table in your favorite restaurant.



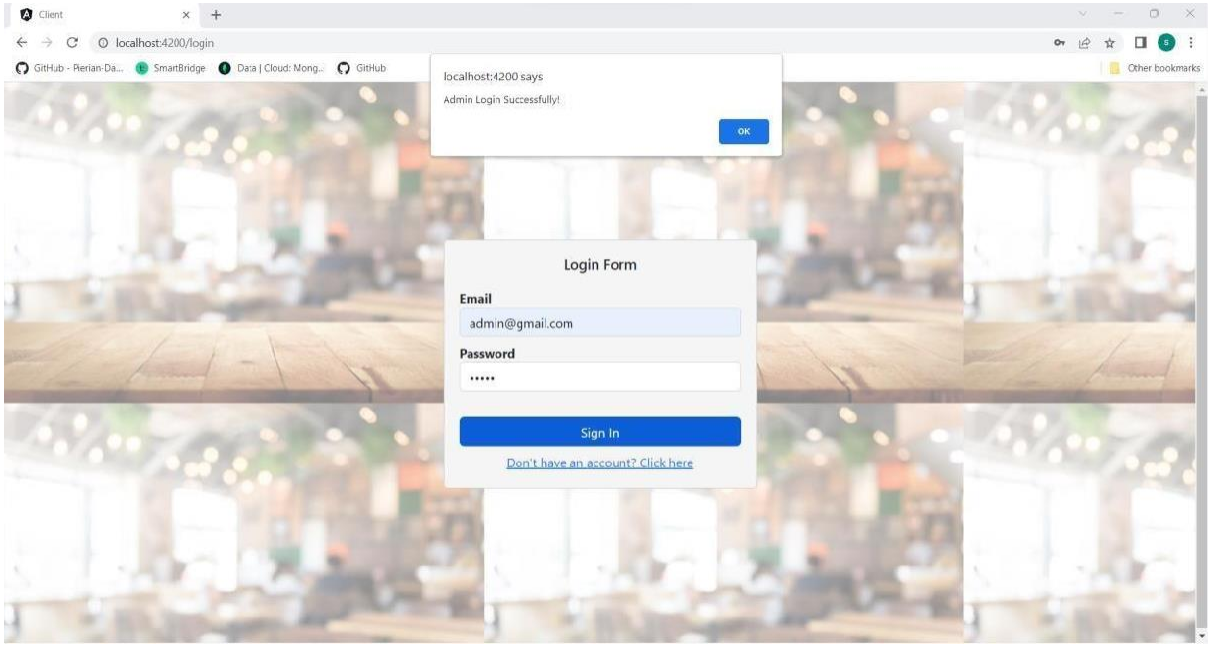
Booking Page: It collects the necessary details for booking a table.



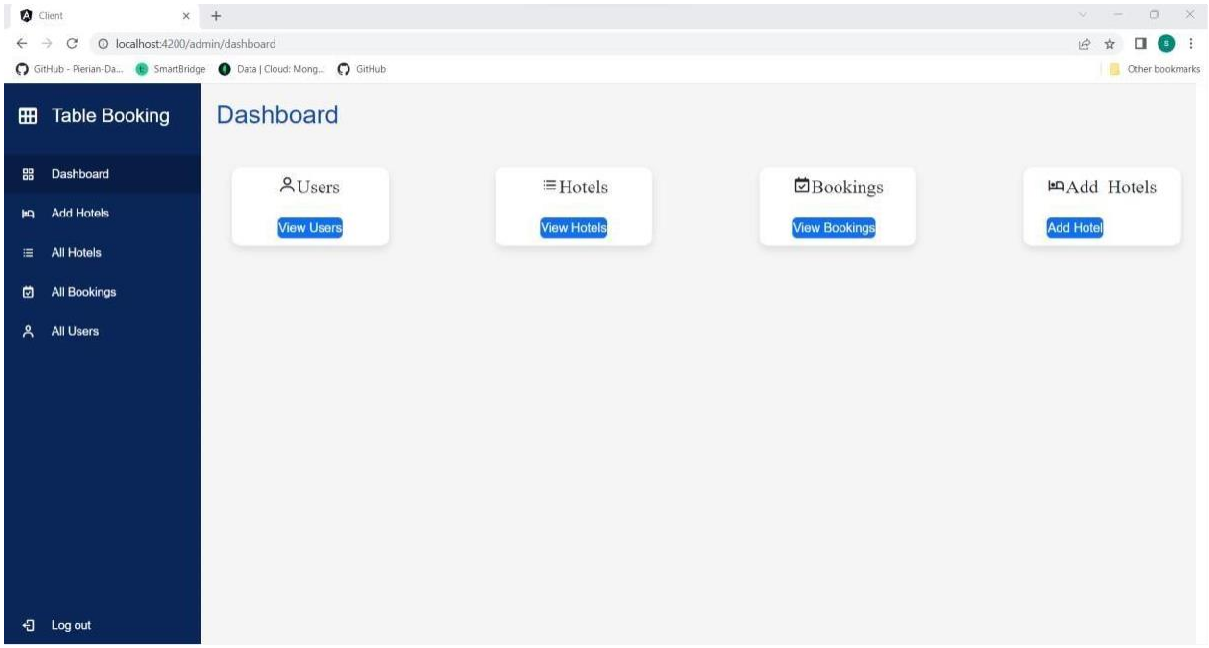
Bookings: This page consists of present and previous bookings of table.



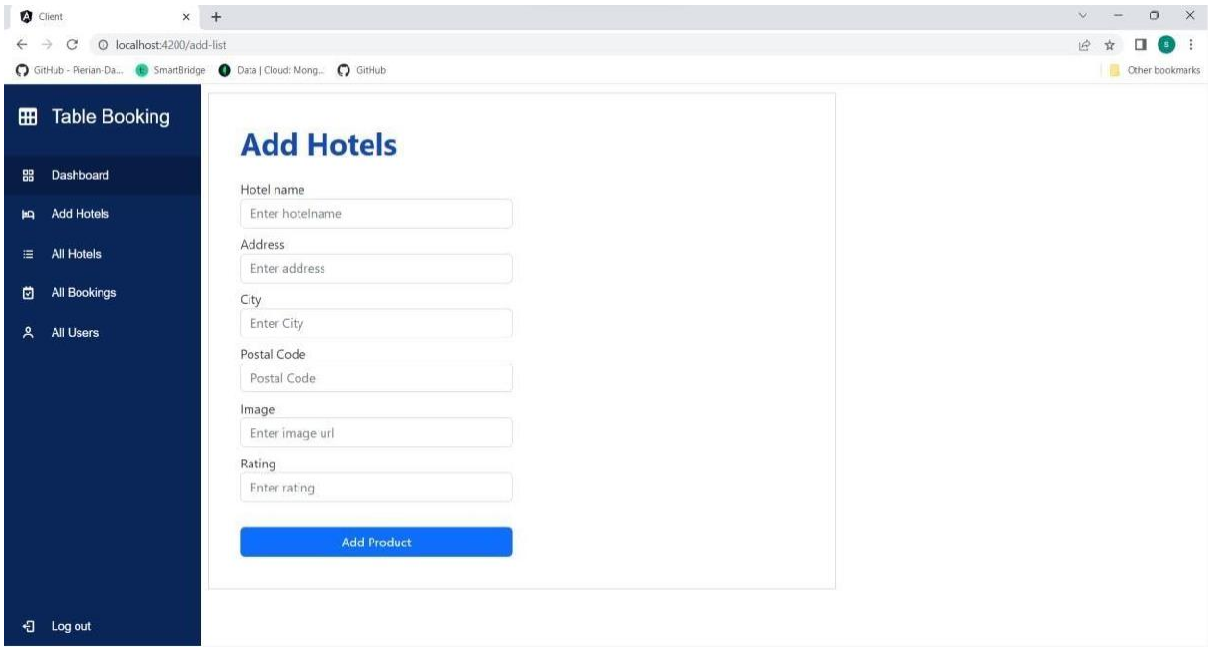
Super Admin Login:



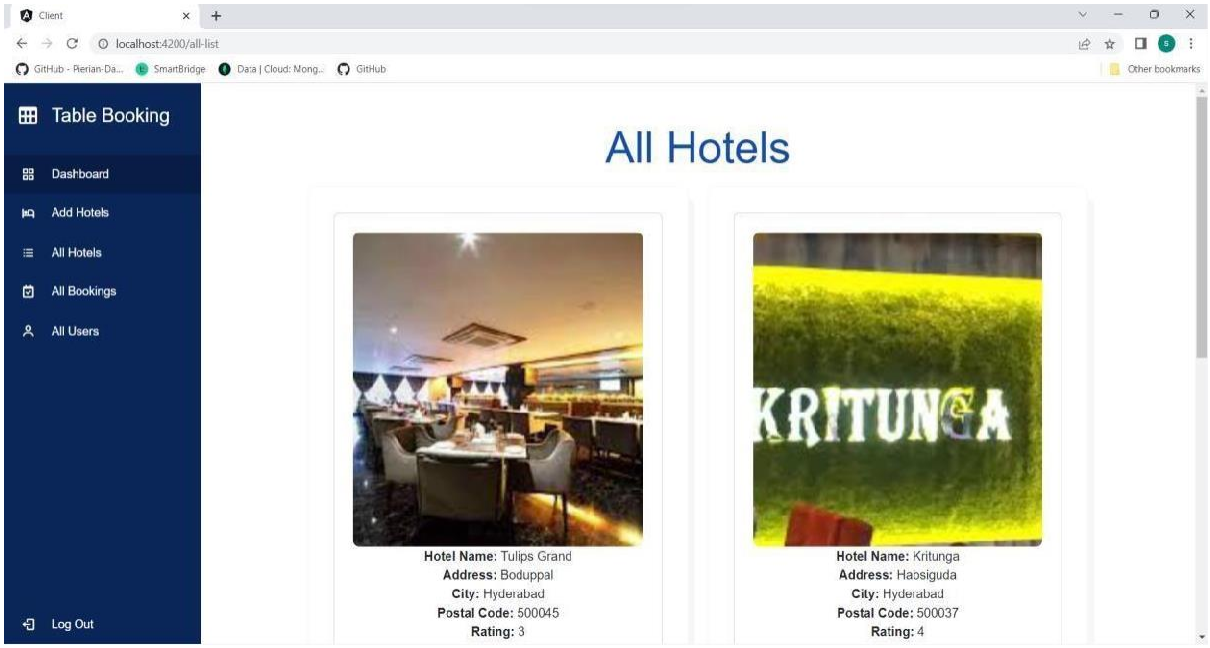
Admin dashboard:



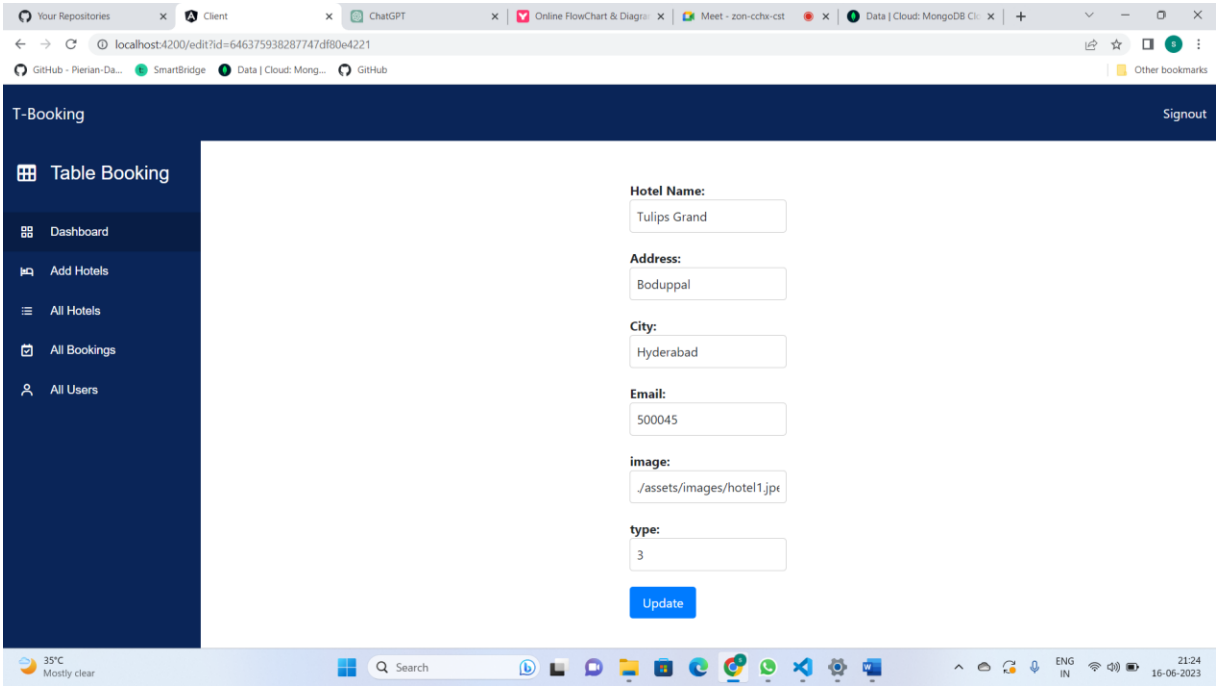
Add Hotels: Admin can add hotels here.



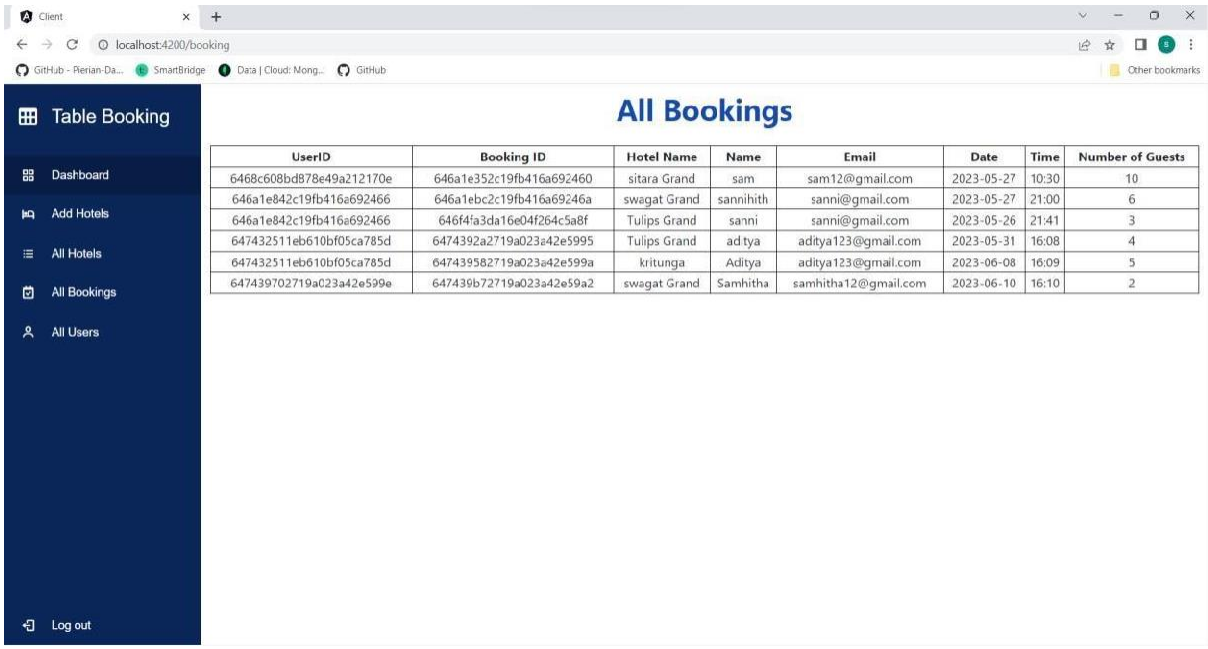
All Hotels: Admin can view hotels he added.



Update Hotel: If any update in address or any fields we can update here

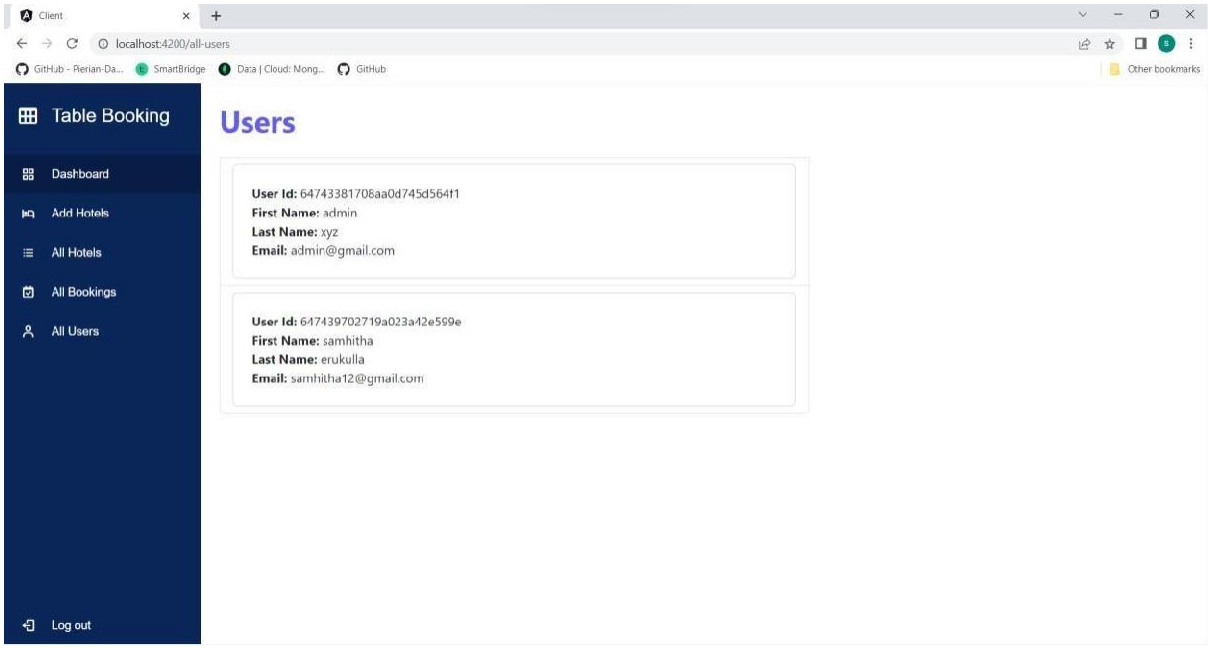


All Bookings: Admin can see bookings of all hotels.

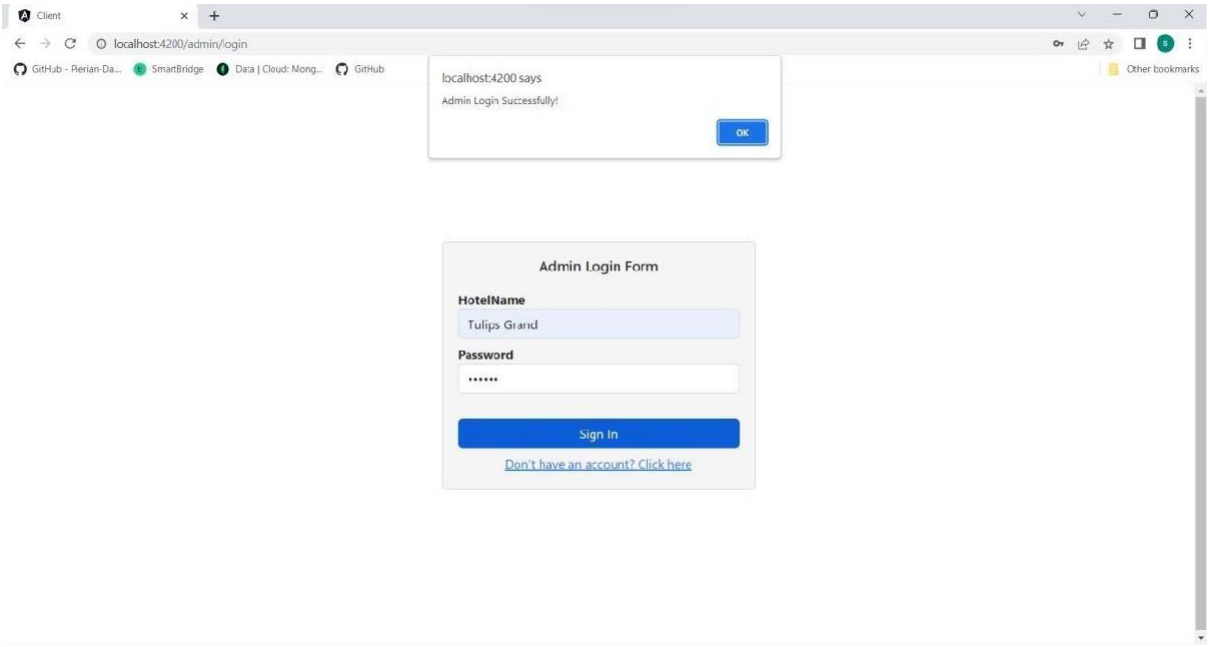


UserID	Booking ID	Hotel Name	Name	Email	Date	Time	Number of Guests
6468c608bd878e49a212170e	646a1e352c19fb416a692460	sitara Grand	sam	sam12@gmail.com	2023-05-27	10:30	10
646a1e842c19fb416a692466	646a1ebc2c19fb416a69246a	swagat Grand	sannihith	sanni@gmail.com	2023-05-27	21:00	6
646a1e842c19fb416a692466	646f4fa3da16e04f264c5a8f	Tulips Grand	sanni	sanni@gmail.com	2023-05-26	21:41	3
647432511eb610bf05ca785d	6474392a2719a023a42e5995	Tulips Grand	aditya	aditya123@gmail.com	2023-05-31	16:08	4
647432511eb610bf05ca785d	647439582719a023a42e599a	kritunga	Aditya	aditya123@gmail.com	2023-06-08	16:09	5
647439702719a023a42e599e	647439b72719a023a42e59a2	swagat Grand	Samhitha	samhitha12@gmail.com	2023-06-10	16:10	2

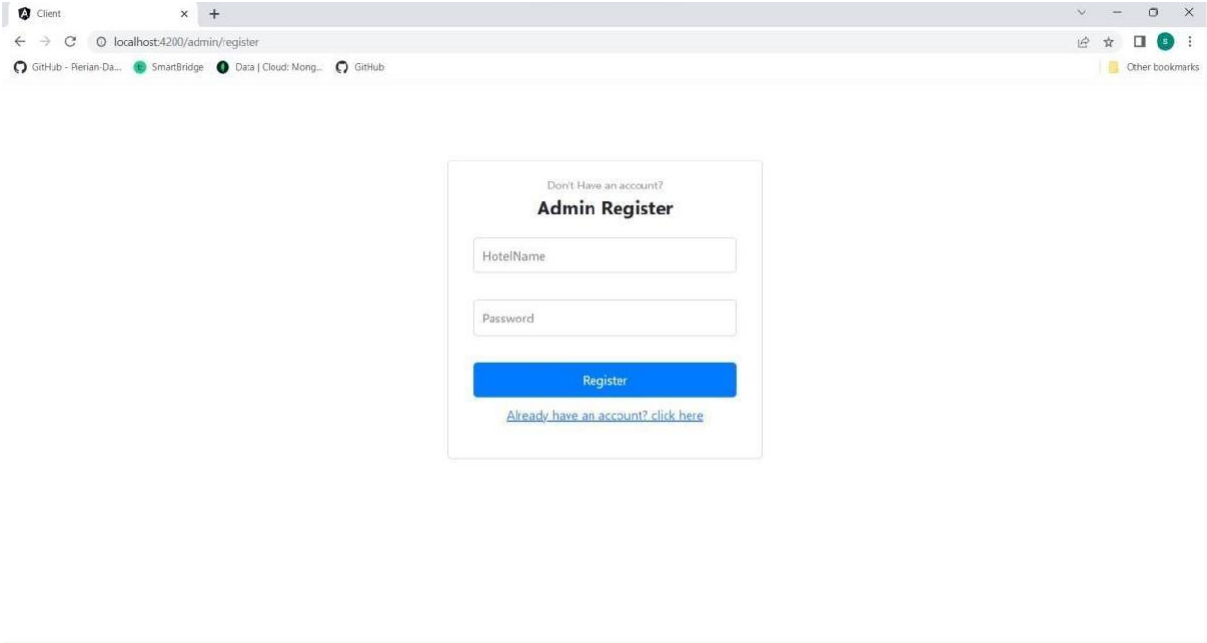
All Users: Admin can see the users who is using this app.



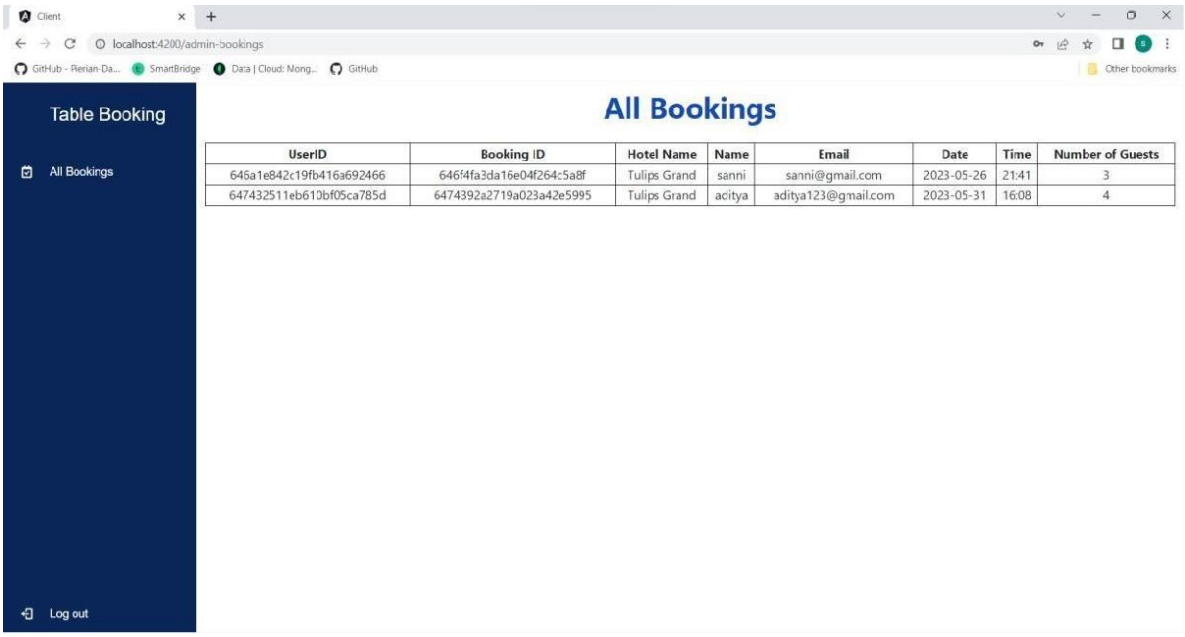
Admin Login : Different hotel owners can login using hotel name and password.



Admin Register: Different hotel owners can register if they don't have account.



Bookings: Bookings of hotels is seen by specific hotel owners.



UserID	Booking ID	Hotel Name	Name	Email	Date	Time	Number of Guests
646a1e842c19fb416a692466	646f4fa3da16e04f264c5a8f	Tulips Grand	sanni	sanni@gmail.com	2023-05-26	21:41	3
647432511eb610bf05ca785d	6474392a2719a023a42e5995	Tulips Grand	acitya	aditya123@gmail.com	2023-05-31	16:08	4