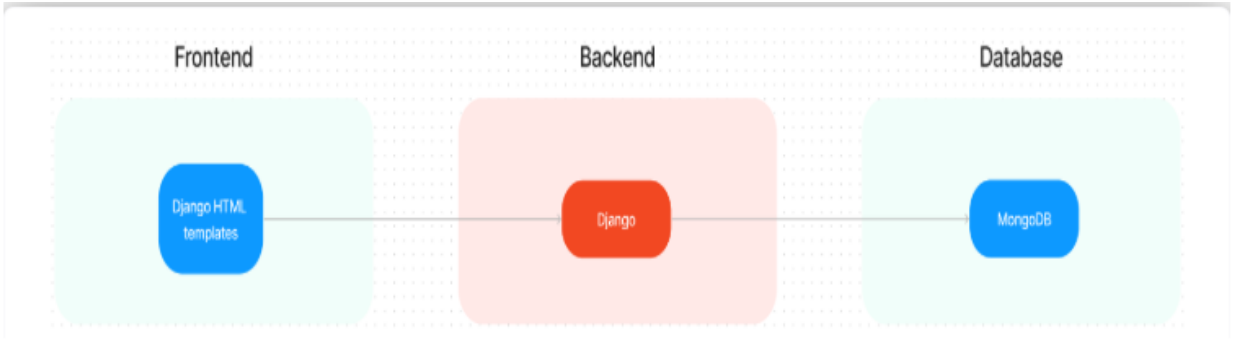
Train Booking APP USING Django

With SB TrainConnect, users can effortlessly access detailed information about train journeys, including departure and arrival times, train classes, and available amenities. The booking process is quick and easy, requiring only basic passenger information and travel preferences. Once booked, users can conveniently manage their travel plans through the booking details page. For train administrators, the platform offers an intuitive dashboard to efficiently manage ticket reservations, monitor bookings, and maintain user privacy.

Scenario based case-study

Imagine a busy professional who frequently travels for work. With SB TrainConnect, they can quickly search for available trains between their home city and business destination, compare options based on departure times and amenities, and book tickets in just a few clicks. The user can then access their booking details on the go, making it easy to share travel information with colleagues or modify plans as needed. Meanwhile, the train operator benefits from real-time booking data, allowing them to optimize train schedules and allocate resources effectively.

TECHNICAL ARCHITECTURE

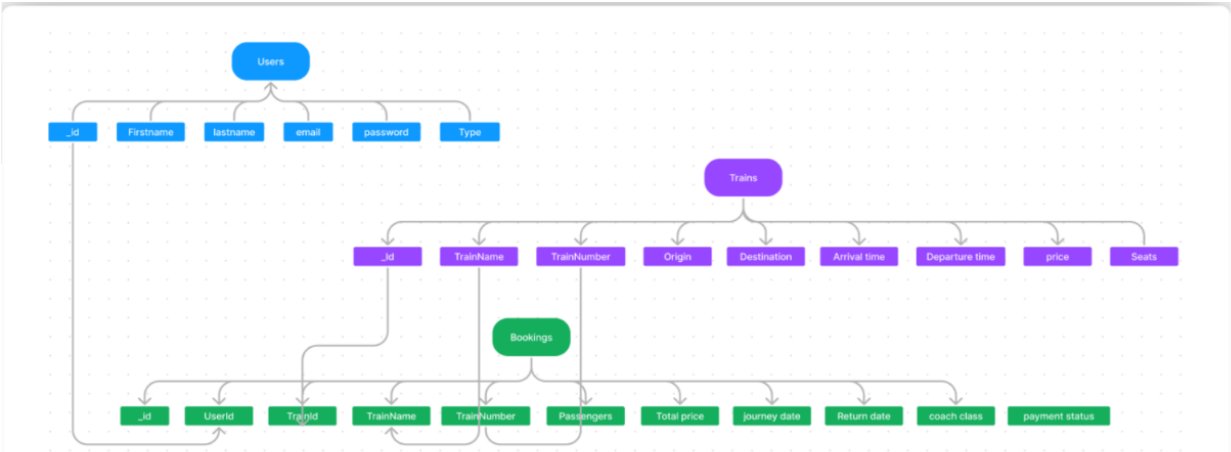


The technical architecture of our Train Booking app follows a client-server model, where the frontend serves as the client and the backend acts as the server. The frontend utilizes the bootstrap library to establish real-time and better UI experience for any user whether it is admin, client or ordinary user working on it.

On the backend side, we employ Django framework to handle the server-side logic and communication. For data storage and retrieval, our backend relies on MongoDB. MongoDB allows for efficient and scalable storage of user data, etc. It ensures reliable and quick access to the necessary information. Together, the frontend and backend components, along with Django and MongoDB, form a comprehensive technical architecture for our House rent app. This architecture enables real-time communication, efficient data exchange, and seamless integration, ensuring a smooth and immersive

booking a property and many more experience for all users.

ER DIAGRAM



The train booking ER-diagram represents the entities and relationships involved in a train booking system. It illustrates how users, bookings, trains, passengers, and payments are interconnected. Here is a breakdown of the entities and their relationships:

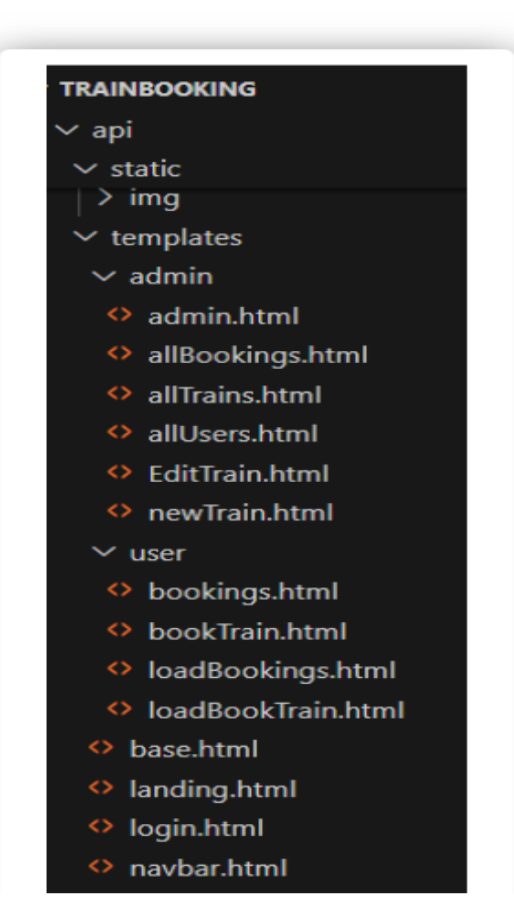
USER: Represents the individuals or entities who book trains. A customer can place multiple bookings and make multiple payments.

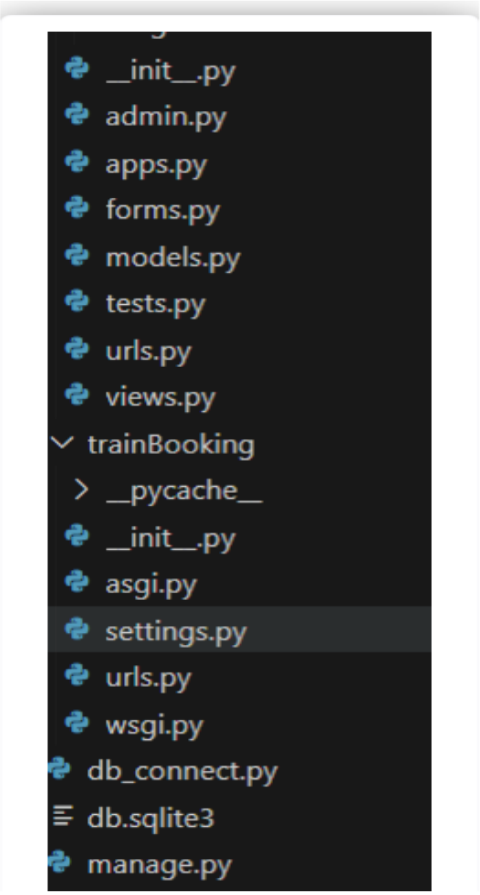
BOOKING: Represents a specific train booking made by a customer. A booking includes a particular train details and passenger information. A customer can have multiple bookings.

TRAIN: Represents a train that is available for booking. Here, the details of train will be provided and the users can book them as much as the available seats.

ADMIN: Admin is responsible for all the backend activities. Admin manages all the bookings, adds new trains, etc.,

PROJECT STRUCTURE





The first image shows the frontend content. It has a well-organized structure with dedicated folders for the API, static files (CSS and images), and templates (admin, and user). The API folder likely contains the code for the application's functionalities. The static folder stores the application's static files, such as Cascading Style Sheets (CSS) used for styling the user interface and images that may be displayed throughout the application. Finally, the templates folder contains the HTML templates that define the application's layout and user interface. These templates are used to dynamically generate the web pages that users will see.

The second image shows details about the server login. It follows a well-defined structure separating the core application logic from the overall project configuration. The application code resides in the Train Booking directory, containing models, views, and templates specific to the Train Booking functionality. The main project directory Train Booking project-wide settings, URL patterns, and management tools. This separation promotes maintainability and scalability as the project grows.

Features:

- **Extensive Train Listing:** SB TrainConnect offers an extensive list of train services, providing a wide range of routes and options for travelers. You can easily browse through the list and explore different train journeys, including departure and arrival times, train classes, and available

amenities, to find the perfect travel option for your journey.

- **Book Now Button:** Each train listing includes a convenient "Book Now" button. When you find a train journey that suits your preferences, simply click on the button to proceed with the reservation process.
- **Booking Details**: Upon clicking the "Book Now" button, you will be directed to a booking details page. Here, you can provide relevant information such as your preferred travel dates, departure and arrival stations, the number of passengers, and any special requirements you may have.
- **Secure and Efficient Booking Process:** SB TrainConnect 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.
- **Confirmation and Booking Details Page:** Once you have 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 your travel dates, departure and arrival stations, the number of passengers, and any special requirements you specified.

PRE-REQUISITES

Here are the key prerequisites for developing a full-stack application using Django, MongoDB:

Django: Setting Up Your Project

Django is a high-level Python framework for building web applications. It streamlines the development process by providing a robust structure and handling common web development tasks.

Prerequisites:

- Python (version 3.6 or later recommended)
- pip (Python package installer)

Installation:

- Verify Python Installation: Open a terminal or command prompt and type python --version. If Python is installed, you'll see the version number.
- Install pip: If you don't have pip, install it using get-pip.py from https://bootstrap.pypa.io/get-pip.py

Creating a Django Project:

- Open a terminal: Navigate to your desired project directory.
- Create a project: Use the following command, replacing mysite with your project name:

```
django-admin startproject mysite
```

Running the Development Server:

- Navigate to the project directory: Use cd mysite.
- Start the server: Run python manage.py runserver.

Access Your Application:

Open http://127.0.0.1:8000/ in your web browser. You should see the Django welcome page, indicating a successful setup.

?**MongoDB:**

MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-like format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data.

Set up a MongoDB database to store your application's data.
Download: https://www.mongodb.com/try/download/community

Installation instructions: https://docs.mongodb.com/manual/installation/

?**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 to connect your Django server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations. To Connect the Database add the dependency in the pom.xml file

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-data-mongodb</artifactId>
</dependency>
```

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

• Visual Studio Code: Download from https://code.visualstudio.com/download

To run the existing Train Booking App project downloaded from Google Drive, Follow below steps:
Download the code from the drive link provided below.

https://drive.google.com/drive/folders/1qUB_et9kr782_RE4JLzH-3PWyAR4nSHc?usp=sharing

Then, open the project in a suitable ide or code editor and run the application using the run button provided.

• The Train Booking app will be accessible at http://127.0.0.1:8000/

You have successfully installed and set up the Train Booking application on your local machine. You can now proceed with further customization, development, and testing as needed.

Application flow

1. **User flow**

- After registration, they can log in with their credentials.
- Once logged in, they can check for the availability of trains in their desired route and dates.
- Users can select a specific train from the list.
- They can then proceed by entering passenger details and other required data.
- 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 Train Booking Admin Dashboard, where they can view bookings, add new train routes, etc.,

Project setup and configuration

a. Create project folders and files:

- Now, firstly create the Django project with the below command:
    - `django-admin startproject project_name`

-
- Also, create an app to manage the rest operations with the command given below:
    - `django-admin startapp app_name`
-

a. **Install required tools and software:**

- For the Django application to function well, we use the libraries mentioned in the prerequisites. Those libraries include
    - Django
    - MongoDB
    - Corsheader
    - Rest_framework

Backend Development

1. **Set Up Project Structure:**

- Create a project directory using your preferred IDE.
- Open a terminal in the project directory.
- Create an app with the instructions provided above.

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 users, properties and applications.

3. Define API Routes:

  - Create separate route files for different API functionalities such as authentication, users, properties and applications.

- Implement urls and views to handle the app route paths and user requests.

4. Implement Data Models:

  - Define MongoDB schemas for the different data entities like users, properties and applications.

- Create corresponding Mongoose models to interact with the MongoDB database.
- Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.

5. Admin Functionality:

- Implement routes and controllers specific to admin functionalities such as fetching all the data regarding users, properties and applications.

6. 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.

Database development

- 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 users, properties and applications.
- Also let’s see the detailed description for the schemas used in the database.

1. Users:

The User model captures essential information for identification, including names, email addresses

(acting as a unique identifier). Additionally, the unique identifier is very helpful to map the Local Tourists to the particular user profile.

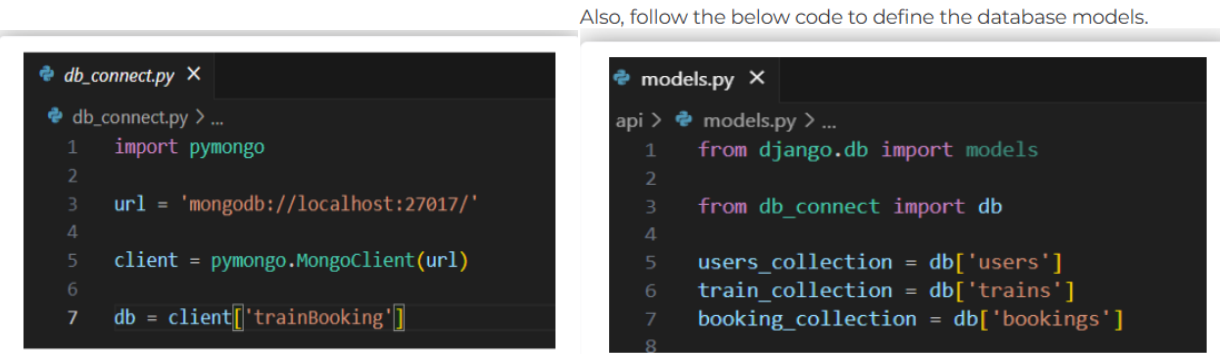
2. Trains:

The Trains model encompasses comprehensive details about each train service offered. Key attributes include train number, name, source station, destination station, departure time, arrival time, total journey time, available classes, amenities, and pricing structures. This information serves as a foundational dataset for users to explore and select suitable train options.

3. Bookings:

The Bookings model records all ticket reservations made by users. Essential fields include booking ID, user ID (foreign key referencing the Users model), train ID (foreign key referencing the Trains model), departure date, number of passengers, booked class, total fare, payment status, and booking status (confirmed, cancelled, etc.). This model acts as a transaction log, maintaining a record of all bookings for both users and administrators to reference.

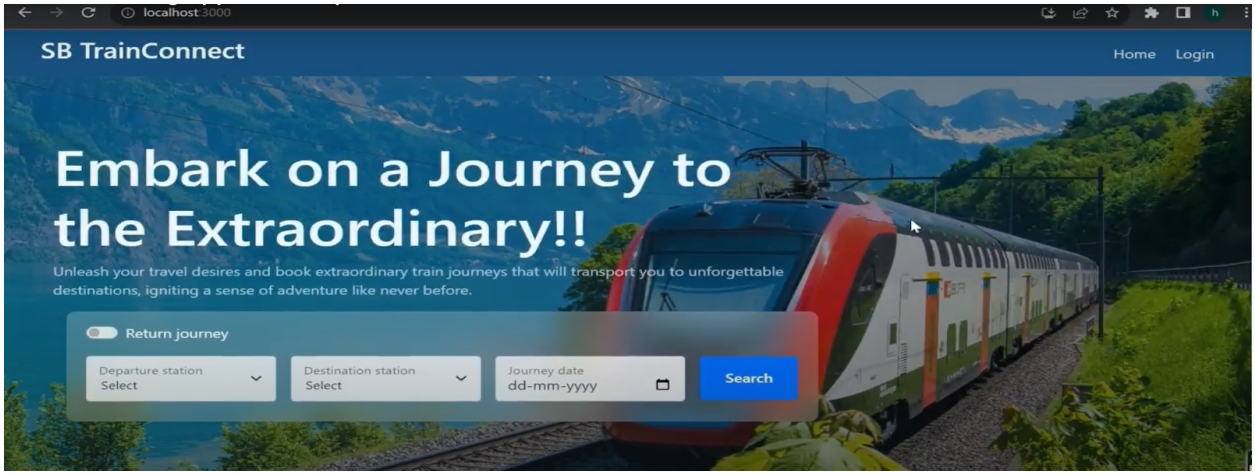
The code for the database connection looks like,

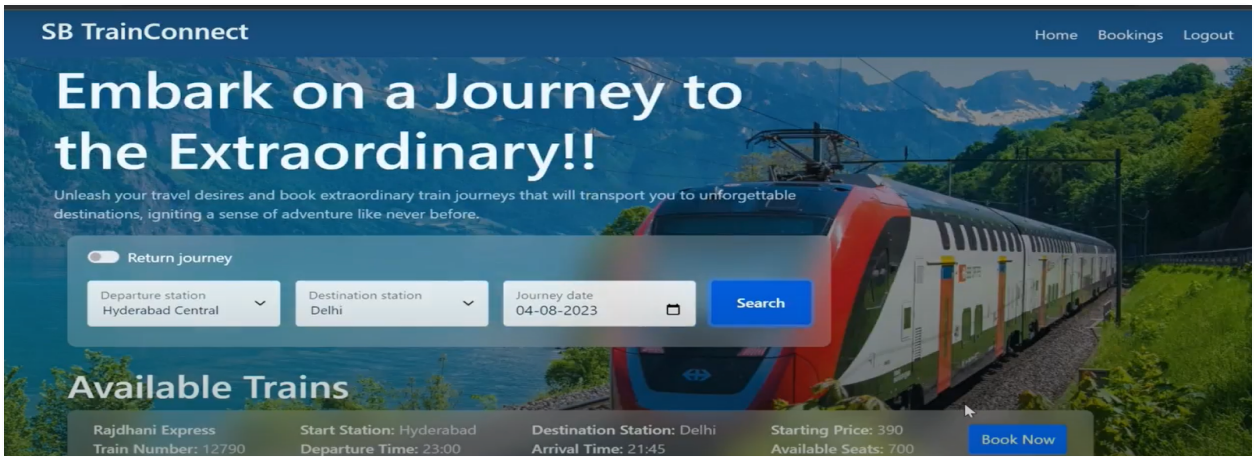


Also, follow the below code to define the database models.

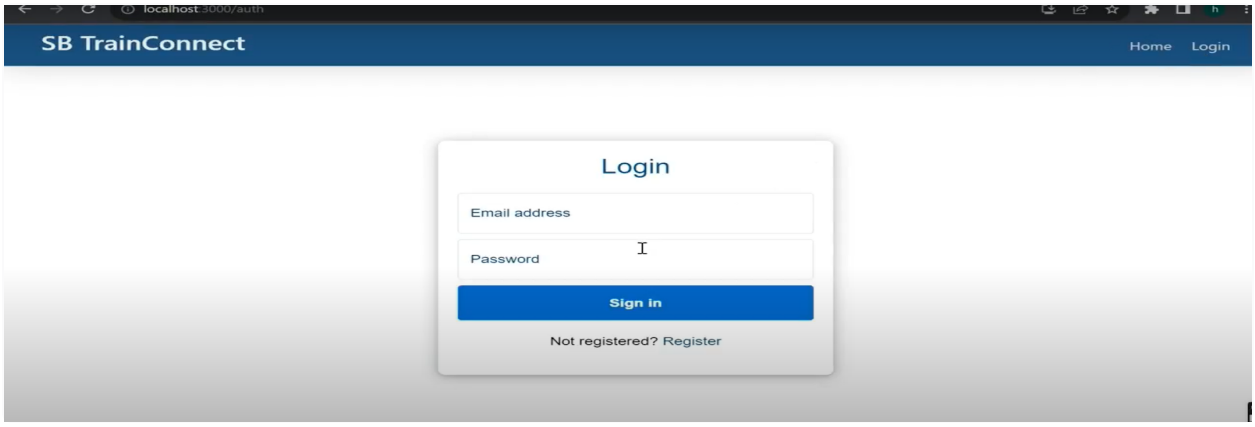
Project Implementation

Landing page:

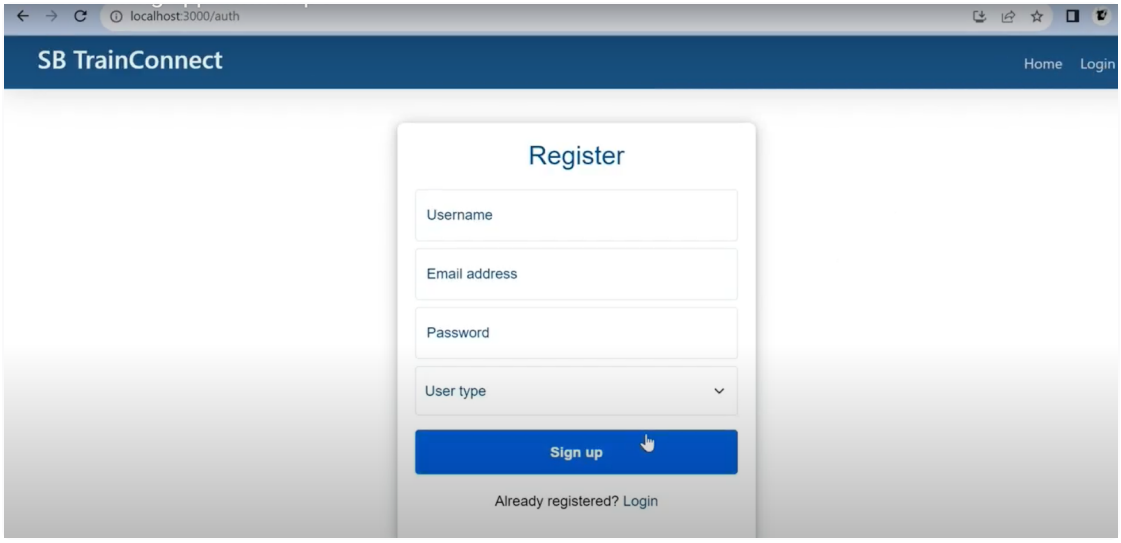




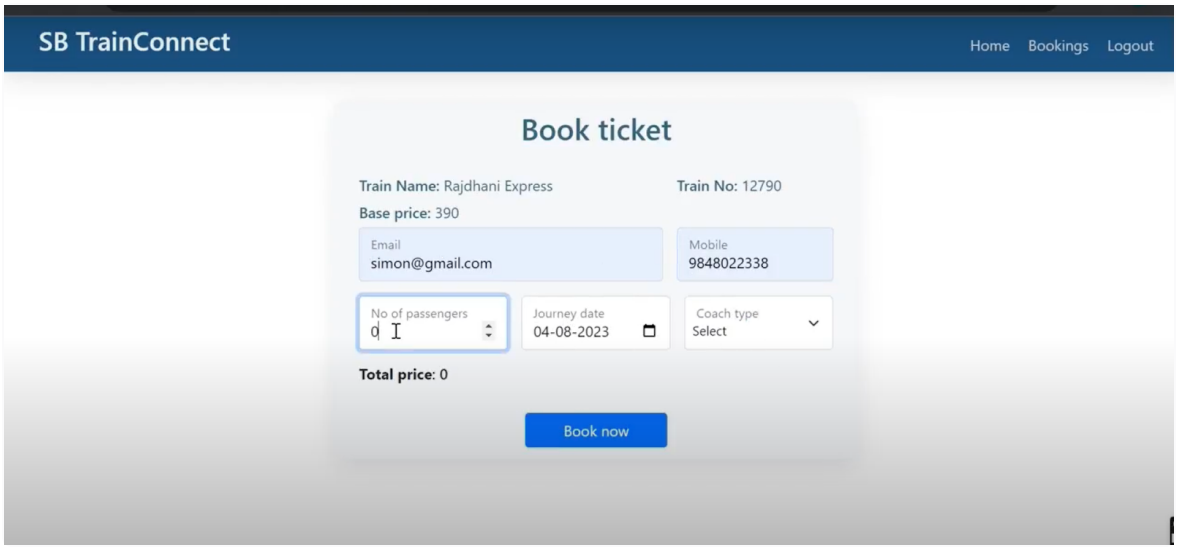
Authentication:



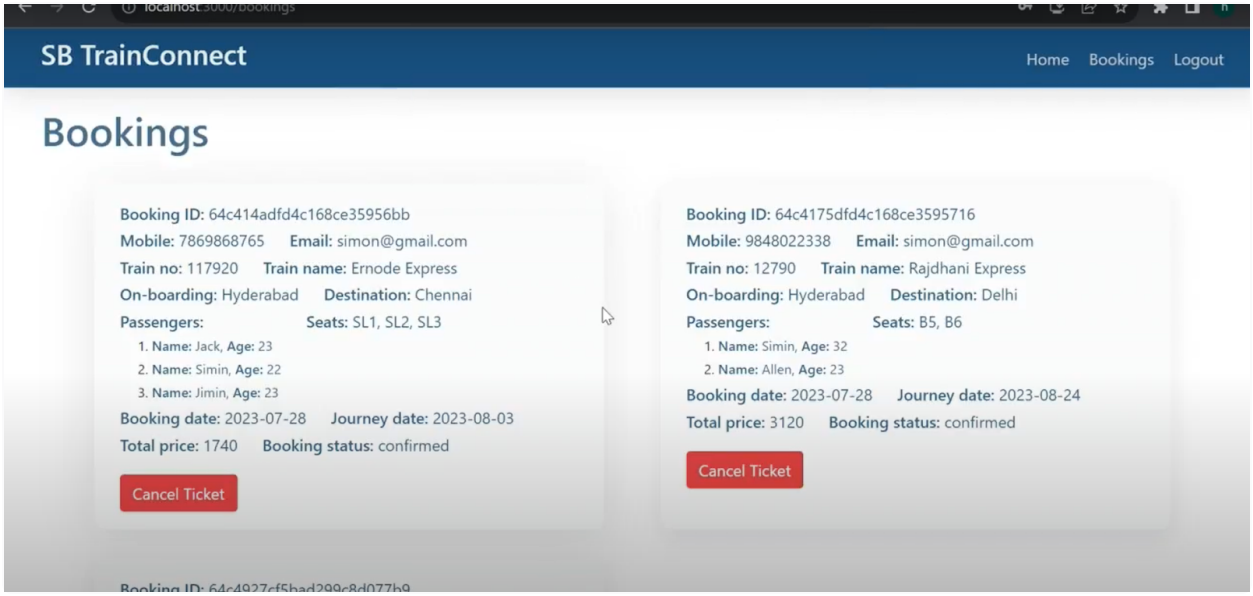
Register:



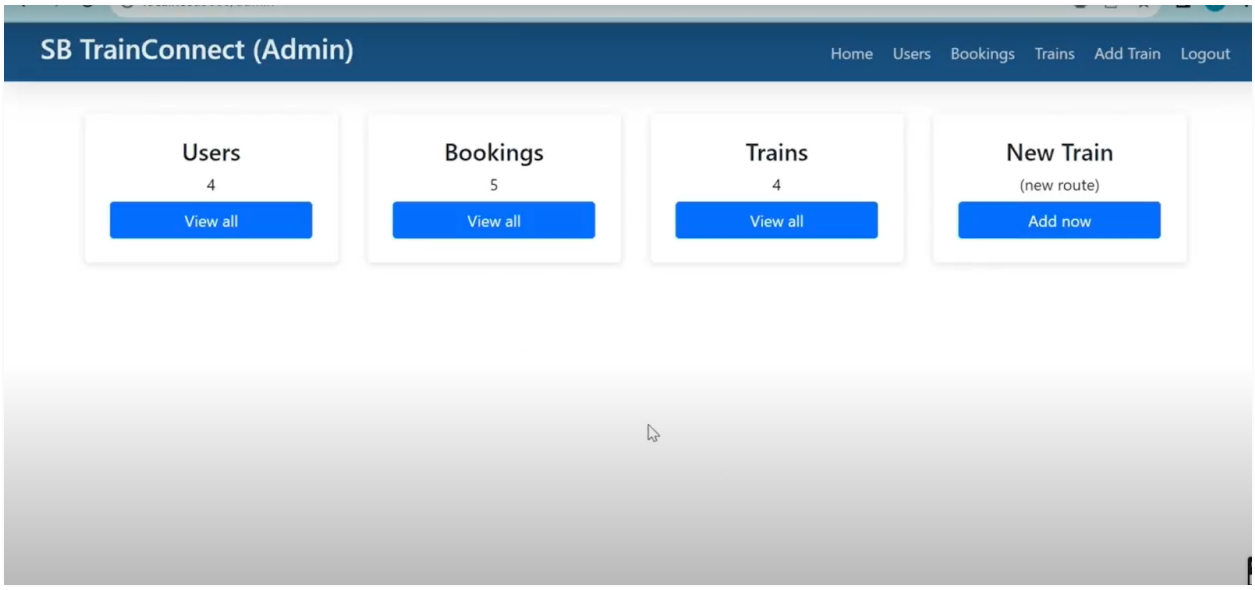
Book Train:



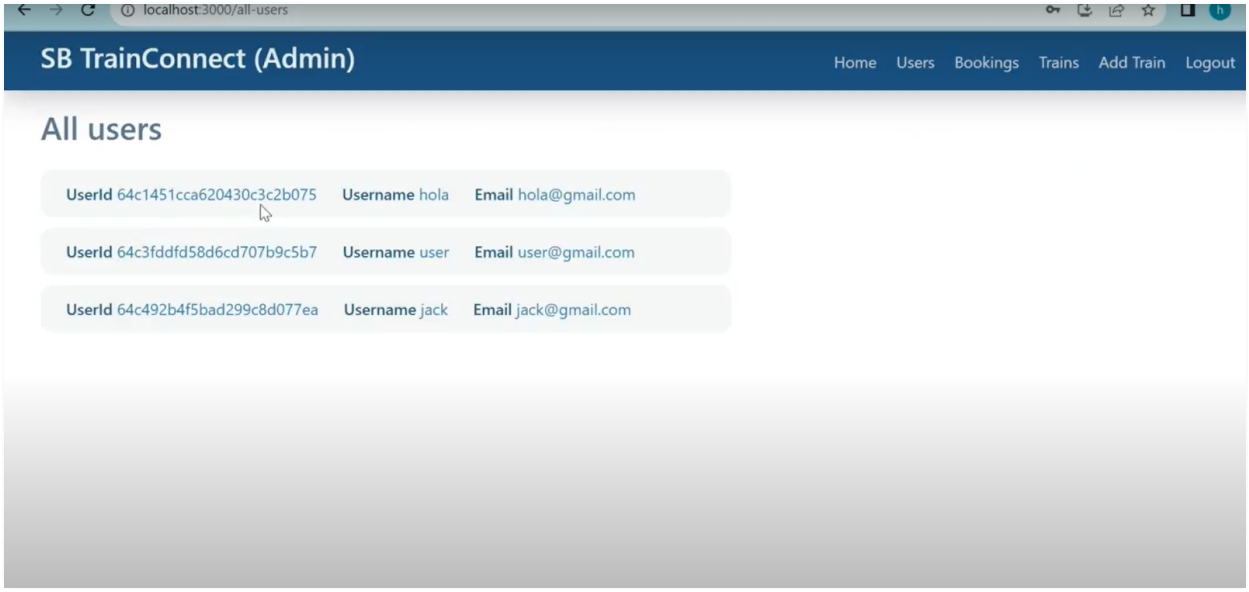
Bookings:



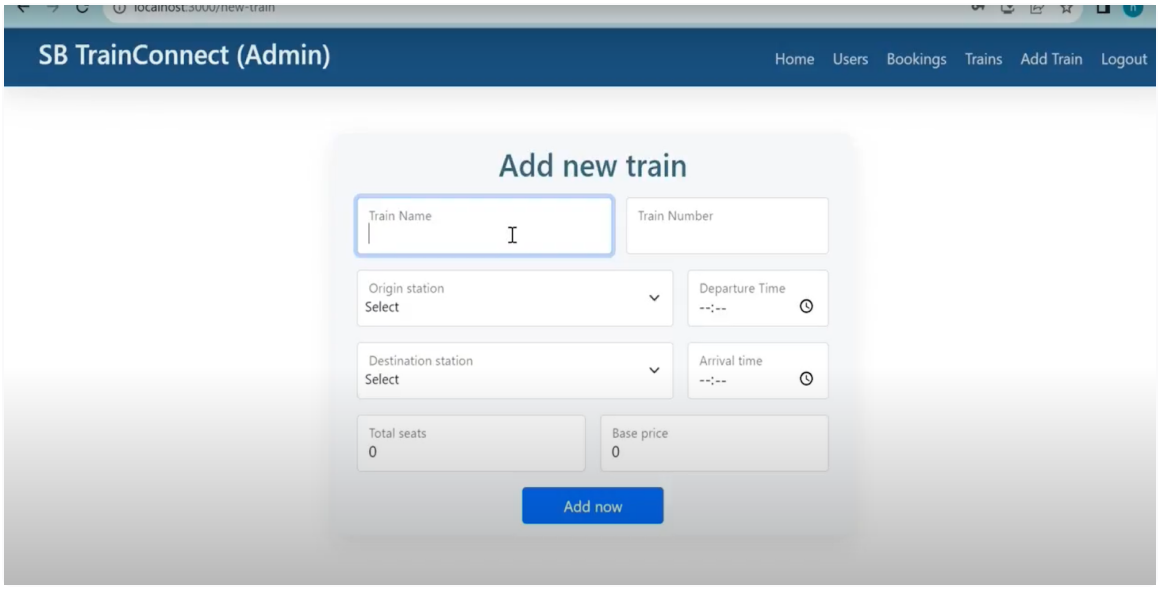
Admin:



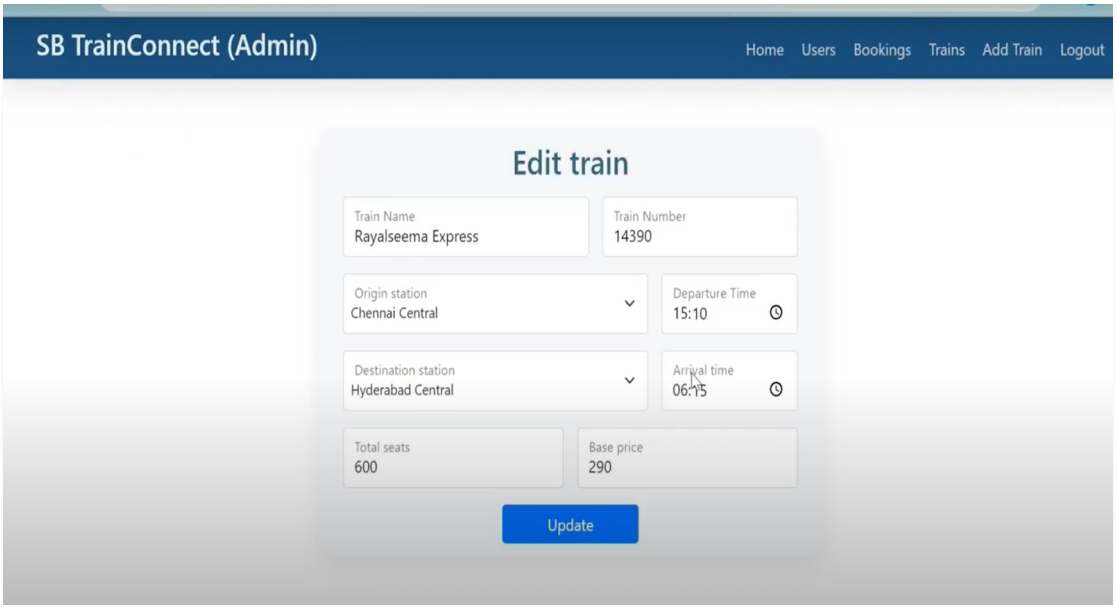
All Users:



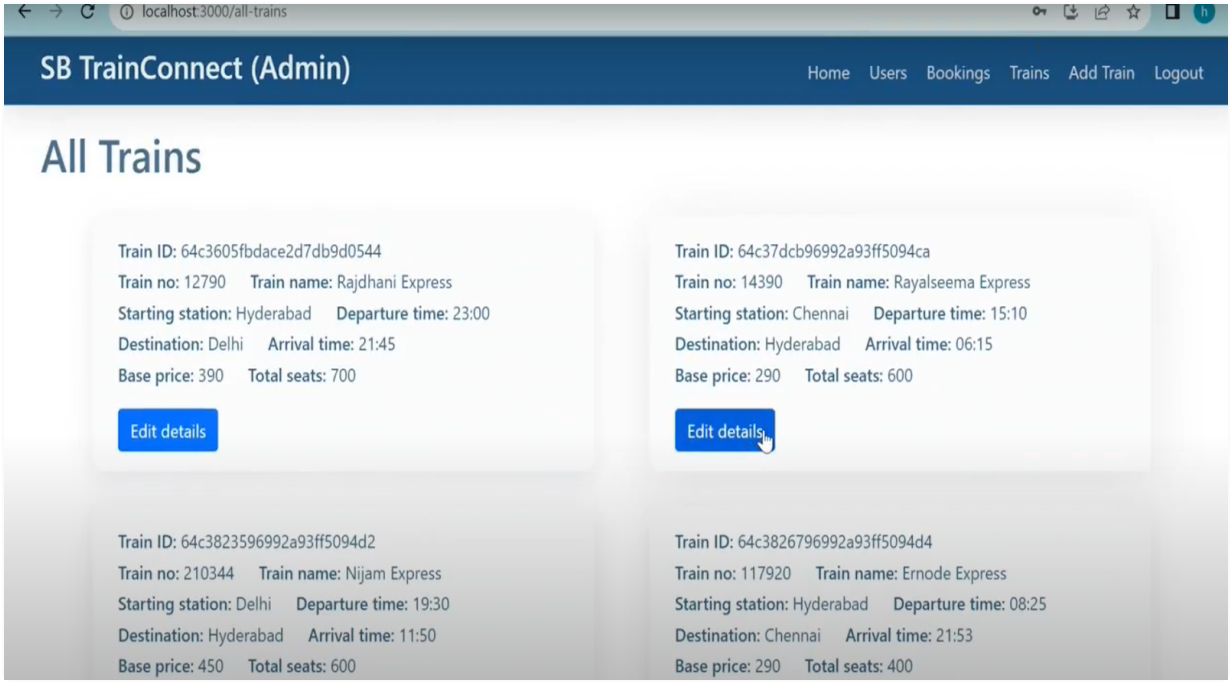
New Train:



Edit Train:


SB TrainConnect (Admin)
Home
Users
Bookings
Trains
Add Train
Logout
Edit train
Train Name
Rayalseema Express
Train Number
14390
Origin station
Chennai Central
Departure Time
15:10
Destination station
Hyderabad Central
Arrival time
06:15
Total seats
600
Base price
290
Update

All Trains:


localhost:3000/all-trains
SB TrainConnect (Admin)
Home
Users
Bookings
Trains
Add Train
Logout
All Trains
Train ID: 64c3605fbdace2d7db9d0544
Train no: 12790 Train name: Rajdhani Express
Starting station: Hyderabad Departure time: 23:00
Destination: Delhi Arrival time: 21:45
Base price: 390 Total seats: 700
Edit details
Train ID: 64c37dcb96992a93ff5094ca
Train no: 14390 Train name: Rayalseema Express
Starting station: Chennai Departure time: 15:10
Destination: Hyderabad Arrival time: 06:15
Base price: 290 Total seats: 600
Edit details
Train ID: 64c3823596992a93ff5094d2
Train no: 210344 Train name: Nijam Express
Starting station: Delhi Departure time: 19:30
Destination: Hyderabad Arrival time: 11:50
Base price: 450 Total seats: 600
Train ID: 64c3826796992a93ff5094d4
Train no: 117920 Train name: Ernode Express
Starting station: Hyderabad Departure time: 08:25
Destination: Chennai Arrival time: 21:53
Base price: 290 Total seats: 400