A PROJECT ON

“Train Reservation System”

SUBMITTED IN

PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE COURSE OF

DIPLOMA IN ADVANCED COMPUTING FROM CDAC



**SUNBEAM INSTITUTE OF INFORMATION TECHNOLOGY**

Hinjawadi

**SUBMITTED BY:**

1. Amit S. Jirole (62833)
2. Akash S. Helale (62894)
3. Omkar A. Bagade (63011)
4. Pankajkumar R. Bodade (63035)

**UNDER THE GUIDENCE OF:**

Mr. Rajiv Kamune Faculty Member

Sunbeam Institute of Information Technology, PUNE.

# CERTIFICATE

This is to certify that the project work under the title „Train Reservation System‟ is done by Amit S. Jirole, Akash S. Helale, Omkar A. Bagade, Pankajkumar R. Bodade in partial fulfillment of the requirement for award of Diploma in Advanced Computing Course.

**Mr. Rajiv Kamune Mr. Yogesh Kolhe**

**Project Guide Course Co-Coordinator**

Date: 28/09/2022

# ACKNOWLEDGEMENT

A project usually falls short of its expectation unless aided and guided by the right persons at the right time. We avail this opportunity to express our deep sense of gratitude towards Mr. Nitin Kudale (Center Coordinator, SIIT, Pune) and Mr. Yogesh Kolhe (Course Coordinator, SIIT, Pune) .

We are deeply indebted and grateful to them for their guidance, encouragement and deep concern for our project. Without their critical evaluation and suggestions at every stage of the project, this project could never have reached its present form.

Last but not the least we thank the entire faculty and the staff members of Sunbeam Institute of Information Technology, Pune for their support.

Amit S. Jirole (62833) Akash S. Helale (62894) Omkar A. Bagade (63011)

Pankajkumar R. Bodade (63035)

CDAC March 2022 Batch SIIT Pune.

# ABSTRACT

The Railway Reservation System facilitates the passengers to enquire about the trains available on the basis of source and destination, Booking and Cancellation of tickets, view transaction history and upcoming journeys, etc. The aim of case study is to design and develop a database maintaining the records of different trains, train details, and passengers.

This project contains Introduction to the Railways Reservation System. It is the computerized system of reserving the seat in advanced. It is mainly used for long route. On-line reservation has made the process for the reservation of seats very much easier than even before.

In our country India, there are number of counters for the reservation of the seats and one can easily make reservation and get tickets. Then this project contains entity relationship model diagram based on railway reservation system and introduction to relation model. There is also diagram of the database of the railway reservation system based on relation model. Example of some SQL queries to retrieve data form rail management database.

# INDEX

|  |  |  |
| --- | --- | --- |
| **1.** | **Introduction** | 6 |
| **2.** | **Requirements** | 7 |
|  | 2.1 Functional Requirements |  |
|  | 2.2 Non- Functional Requirements | 11 |
|  | 2.2.1 Interface |  |
|  | 2.2.2 Performance |  |
|  | 2.2.3 Constraint |  |
|  | 2.2.4 Other Requirements |  |
| **3.** | **Design** | 12 |
|  | 3.1 Database Design |  |
| **4.** | **Coding Standard Implemented** | 16 |
| **5.** | **Test Report** | 18 |
| **6.** | **Project Management Related Statistics** | 19 |
|  | **Appendix A** | 21 |
|  | **Appendix B** | 22 |
| **7.** | **References** | 32 |

1. **INTRODUCTION**

Database is an organized collection of data. The data is typically organized to model aspect of reality in a way that supports processes requiring information. A DBMS makes it possible for end user to create, read, update and delete data in a database. The DBMS essentially serves as an interface between the database and end user or application programs, ensuring that data is consistently organized and remains accessible. The DBMS manages three important things: the data, the database engine that allows data to be accessed, locked and modified and the database schema, which defines the database‟s logical structure. This three foundational elements help provide concurrency, security, data integrity and uniform administration procedures. The DBMS can offer both logical and physical data independence. That means it can protect user and application from needing to know where data is stored or having to be concerned about changes to the physical structure of data.

The main purpose of maintaining database for Railway Reservation System is to reduce the manual error involved in the booking and cancelling of ticket and make it convenient for the customers and providers to maintain the data about their customer and also about the seat available at them. Due to automation many loopholes that exist in the manual maintenances of the record can be removed. The speed of obtaining and processing the data will be fast. For future expansion the proposed system can be web enabled so that client can make various enquiries about trains between stations. Due to this, sometimes a lot of problems occur and they are facing many disputes with customers. To solve the above problem, we design a data base which includes customer details, availability of seats in trains, no of trains and their details.

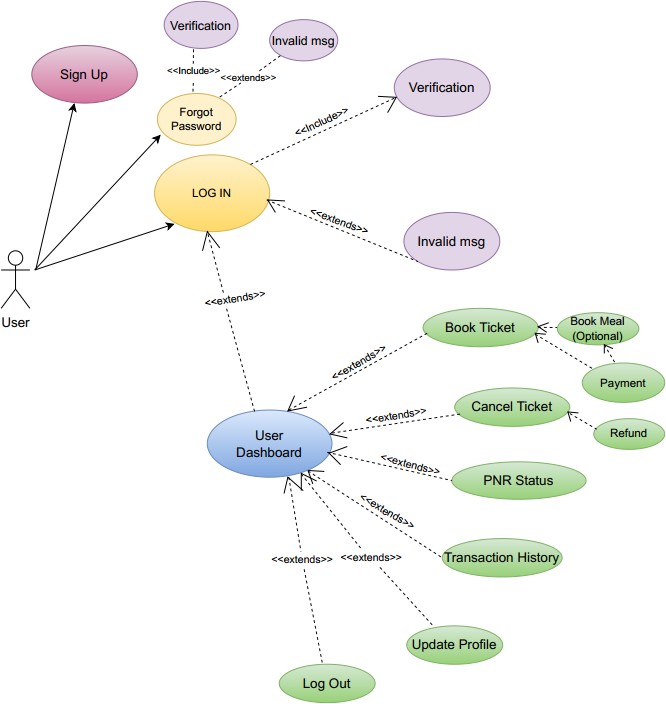
# REQUIREMENTS

* 1. **Functional Requirements**

**Customer Scope:-**

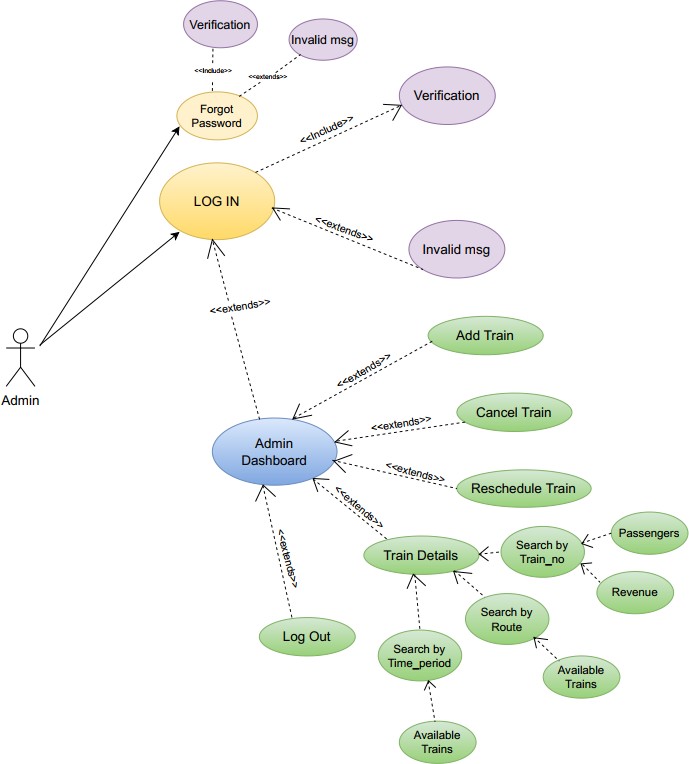
Through this system customers can get the details about trains, their timing, their types related to the same on site in just a click.

The customer can view train schedules, the booking done by them, payment details, cancel booking.



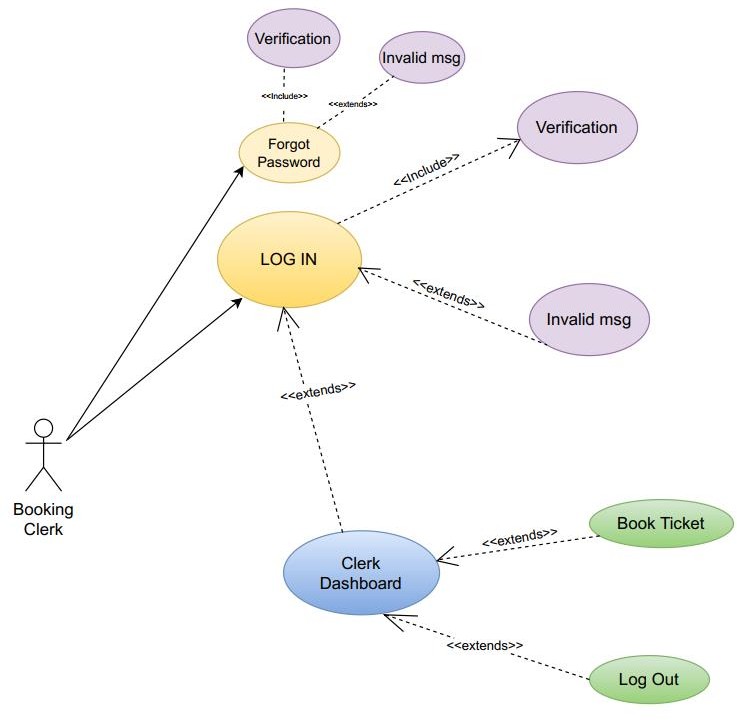
**Admin Scope:-**

Apart from functionalities like login, forgot password, check trains availability, view passengers.



**Clerk Scope:-**

Clerk can search trains and book tickets for customer, print the tickets.



* 1. **on-Functional Requirements**
     1. **Interface**

Go to Appendix B for user interfaces

* + 1. **Performance**
       - **Number of Concurrent Users:**

TRS shall be able to handle at least 1000 transactions/inquiries per second

* + - * **Booking Of Tickets:**

The system is susceptible to any temporary server failure since it uses the strong feature of Hibernate and Spring data JPA. Hence the examination will be continued even if the sever gets disconnected in between the examination.

* + 1. **Constraint**

TRS shall be able to handle at least 1000 transactions/inquiries per second

* + 1. **Other Requirements:**
* **Hardware Interfaces**

The SPMS is expected to function on Intel PIII 900 MHz Processor equivalent or above, 128 MB RAM, 20 GB HDD.

* **Software Interfaces**

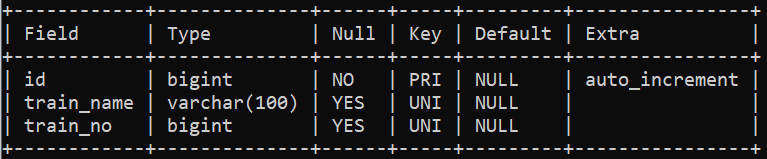
The SPMS shall work on MS Windows operating systems family (MS Windows NT Workstation, MS Windows XP, Windows 7, Windows 8, Windows 10, Windows 11). It configures to work with MySQL database. This System works on Apache Tomcat server. It uses browser IE 9.0 & above.

# DESIGN

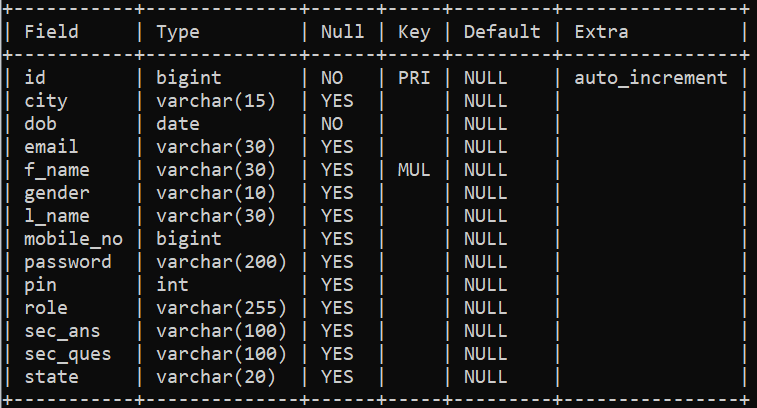
* 1. **Database Design**

The following table structures depict the database design.

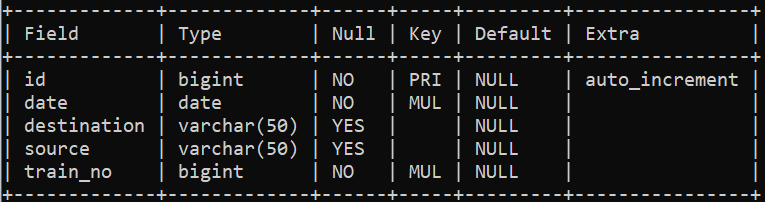
**Table 1: train;**



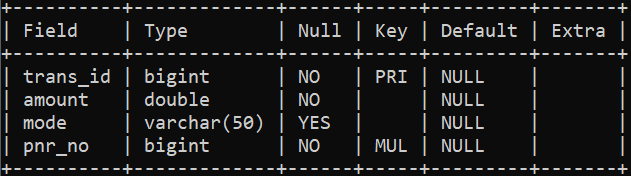
**Table 2: user;**



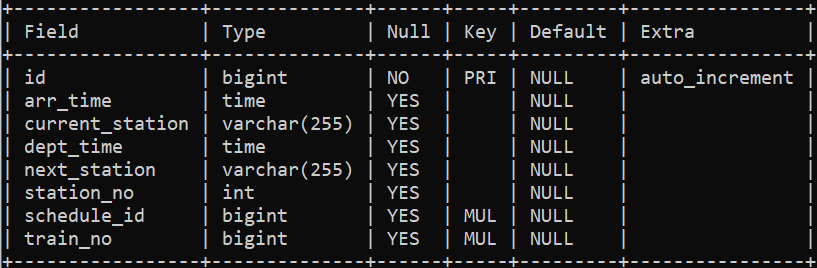
**Table 3: schedule;**



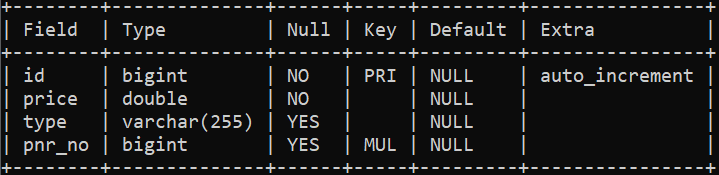
**Table 4: payment;**



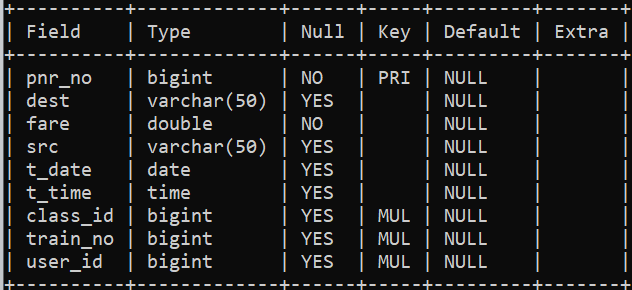
**Table 5: route;**



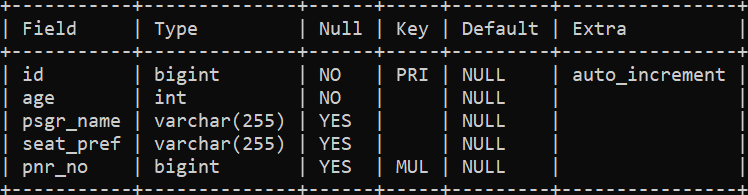
**Table 6: meal;**



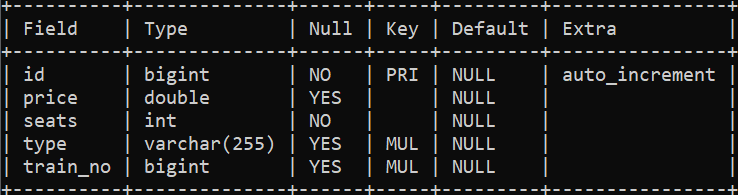
**Table 7: ticket;**



**Table 8: passenger;**



**Table 9: class;**



# CODING STANDARD IMPLEMENTED

**Naming and Capitalization:**

Below summarizes the naming recommendations foe identifiers in Pascal casing is used mainly (i.e. capitalize first letter of each word) with camel casing (capitalize each word except for the first one) being used in certain circumstances.

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | **Case** | **Example** | **Additional Notes** |
| Class | Pascal | Ticket, User, Schedule, Payment, Route, Train | Class names should be based on "objects" or "real things" and should generally be **nouns**. No „\_‟ signs allowed. Do not use type prefixes like  „C‟ for class. |
| Method | Camel | getAllTrains, addTrain, cancelTrain | Methods should use **verbs** or verb phrases. |
| Parameter | Camel | gender, trainNo,age, trainName | Use descriptive parameter names. Parameter names should be descriptive enough that the name of the parameter and its type can be used to determine its meaning in most scenarios. |
| Interface | Pascal | TrainRepo, AdminService, ClerkService | Do not use the „\_‟ sign |
| Property | Pascal | **--** | Use a noun or noun phrase to name properties. |
| Associated private member variable | \_camelCase | **--** | Use underscore camel casing for the private member variables |
| Exception Class | Pascal with “Exception” suffix | **--** | **---**  **--** |

**Comments**

* Comment each type, each non-public type member, and each region declaration.
* Use end-line comments only on variable declaration lines. End-line comments are comments that follow code on a single line.
* Separate comments from comment delimiters (apostrophe) or // with one space.
* Begin the comment text with an uppercase letter.
* End the comment with a period.
* Explain the code; do not repeat it.

# TEST REPORT

**General Testing:**

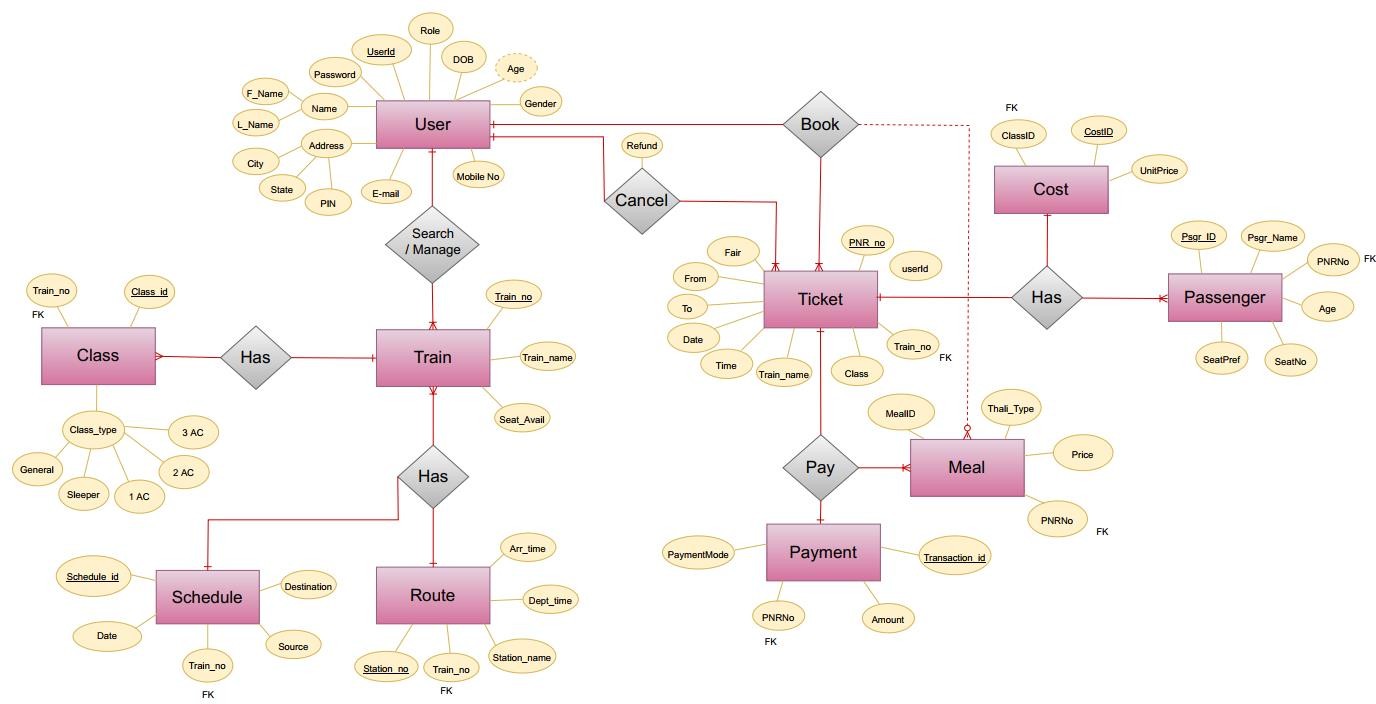
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.No** | **Test Case** | **Expected Result** | **Actual Result** | **Error Message** |
| 1 | Register Page | Redirected to Next page | OK | Nothing |
| 2 | Login Page | Pop-up will come | OK | Invalid Credentials |
| 3 | Change Password | Password will be reset | OK | Nothing |
| 4 | Search Train | Gives all train details | OK | Nothing |
| 5 | Booking Ticket | All the fields should be filled for submission | OK | Nothing |
| 6 | Checking login or not | User is logged in or not | OK | Nothing |
| 7 | Go to ticket page | Set added information about person | OK | Nothing |
| 8 | Add information in ticket table | Save this all data into ticket table | OK | Nothing |
| 9 | Transaction | On back it should be reverted to previous page | OK | Nothing |
| 10 | View transaction done | It shows you all transactions done previously | OK | Nothing |
| 11 | Logout | It will logout from user profile. | OK | Nothing |
|  |  |  |  |  |

# PROJECT MANAGEMENT RELATED

**STATISTICS**

|  |  |  |  |
| --- | --- | --- | --- |
| **DATE** | **WORK PERFORMED** | **SLC phase** | **Additional Notes** |
| 5 May 2022 | Project Allotment and User Requirements Gathering | Feasibility Study | **--** |
| 08 June 2022 | Initial SRS Document Validation And Team Structure Decided | Requirement Analysis | The initial SRS was presented to the client to understand his requirements better |
| 15 June 2022 | Designing the use-cases, Class Diagram, Collaboration Diagram, E-R  Diagram and User Interfaces | Requirement Analysis & Design Phase | Database Design completed |
| 22 June 2022 | Business Logic Component design Started | Design Phase | **--** |
| 28 June 2022 | Coding Phase Started | Coding Phase | 70% of Class Library implemented. |
| 02 July 2022 | Implementation of Web Application and Window Application Started | Coding Phase | Class Library Development going on. |
| 11 July 2022 | Implementation of Web Application and Window Application Continued | Coding Phase and Unit Testing | Class Library Modified as per the need. |
| 24 July 2022 | Implementation of Web Application and Window Application Continued | Coding Phase and Unit Testing | **--** |
| 06 August 2022 | After Ensuring Proper Functioning the Required Validations were Implemented | Coding Phase and Unit Testing | Module Integration was done by the Project Manager |

|  |  |  |  |
| --- | --- | --- | --- |
| 12 September 2022 | The Project was Tested by the respective Team Leaders and the Project Manager | Testing Phase (Module Testing) | **--** |
| 19 September 2022 | The Project was Submitted to Other Project Leader of Other Project Group For Testing | Testing Phase (Acceptance Testing) | The Project of Other Team was Taken up by the Team for Testing |
| 24 September 2022 | The Errors Found were Removed | Debugging | The Project was complete for submission |
| 29 September 2022 | Final Submission of Project | -- | -- |

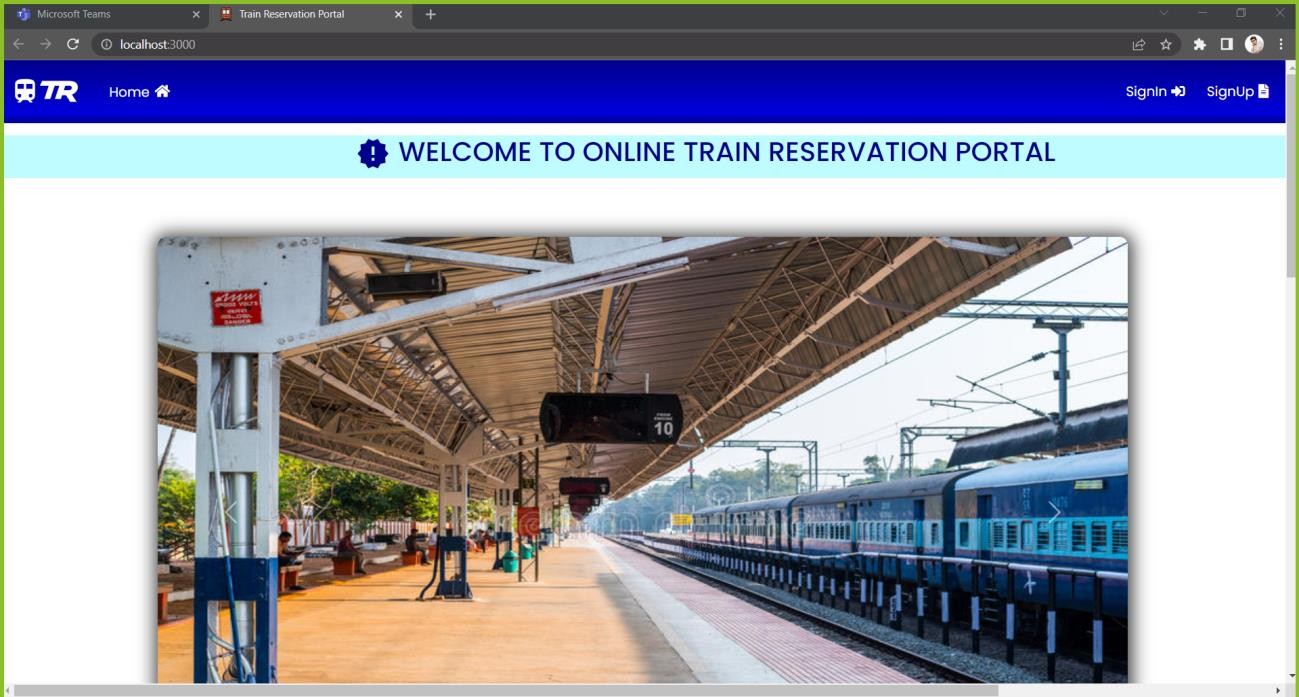


**Appendix A**

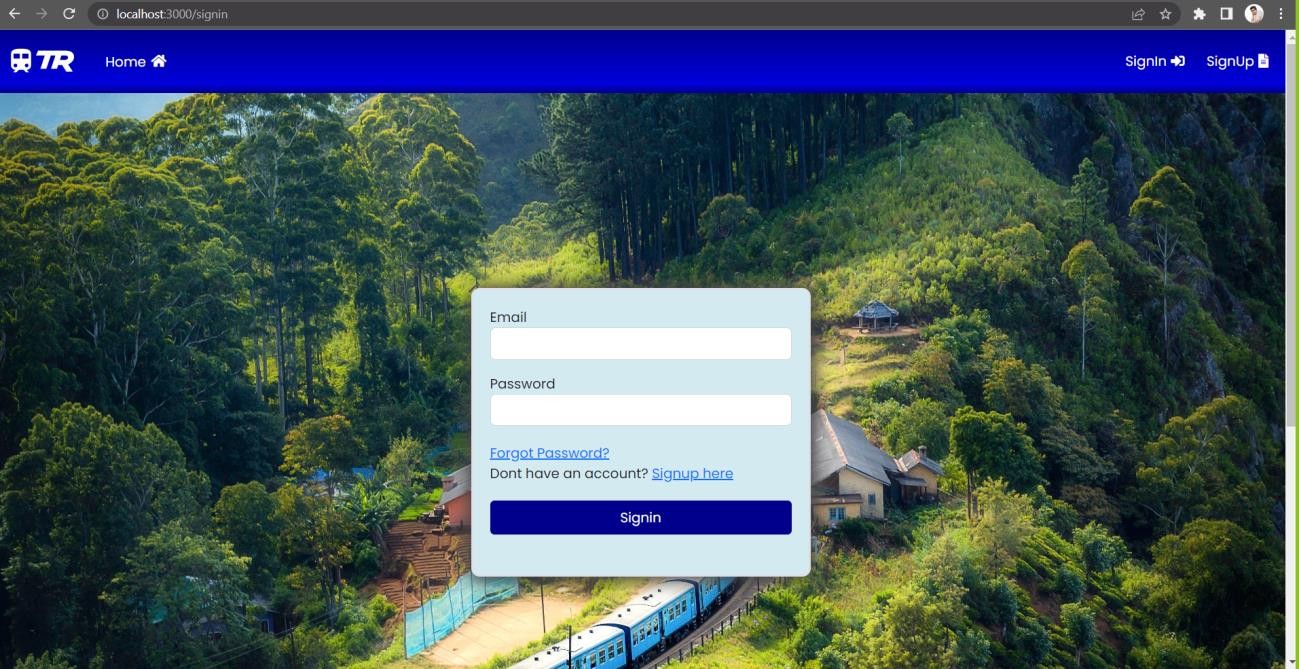
**Entity Relationship Diagram:**

**Appendix B**

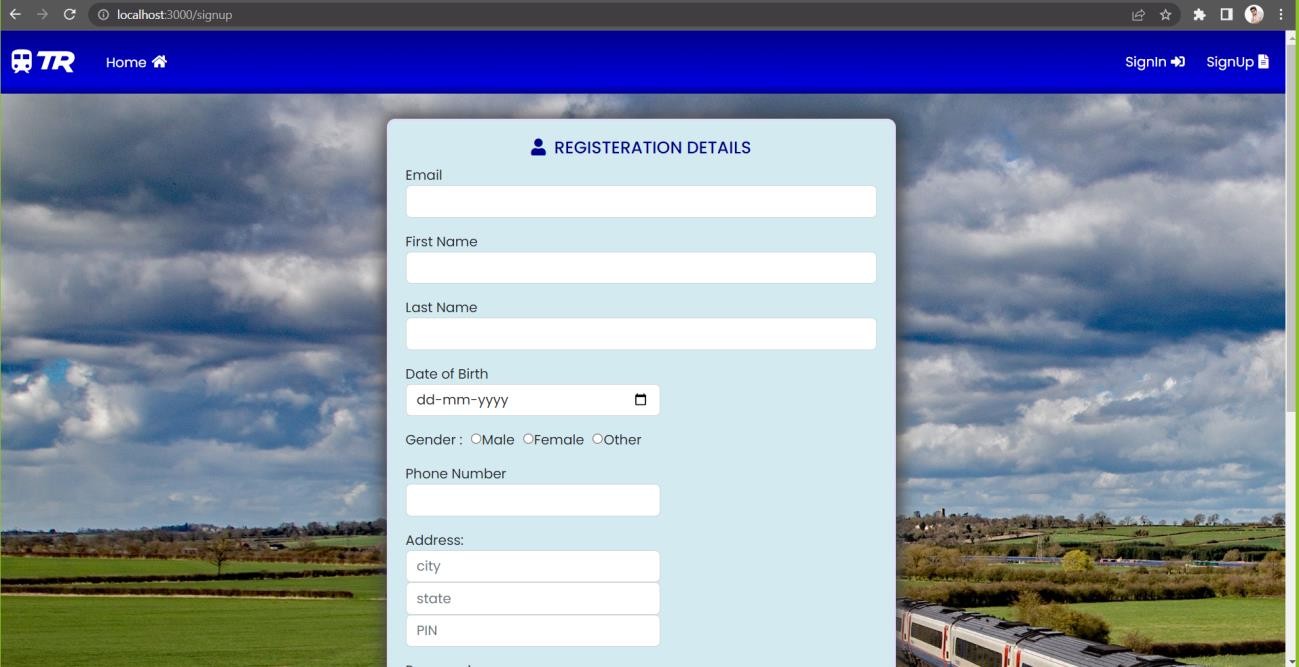
**Homepage:**



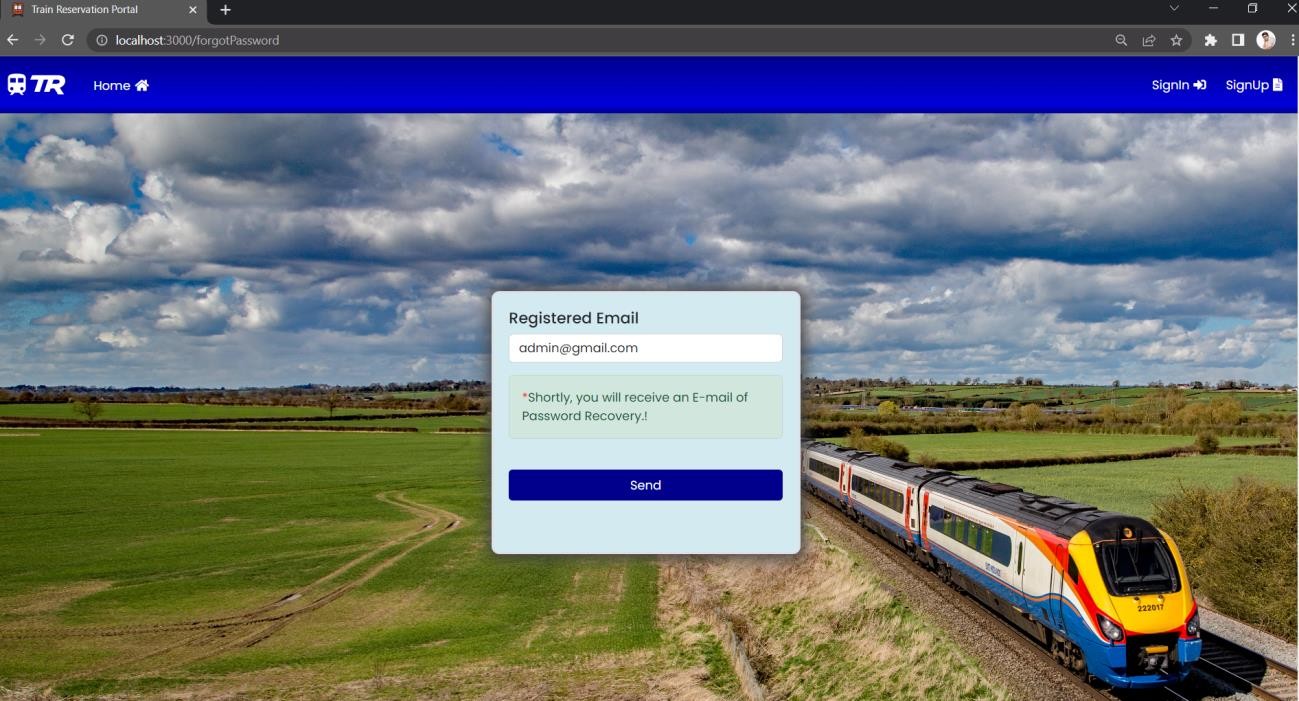
**Sign In:**



**Sign Up:**

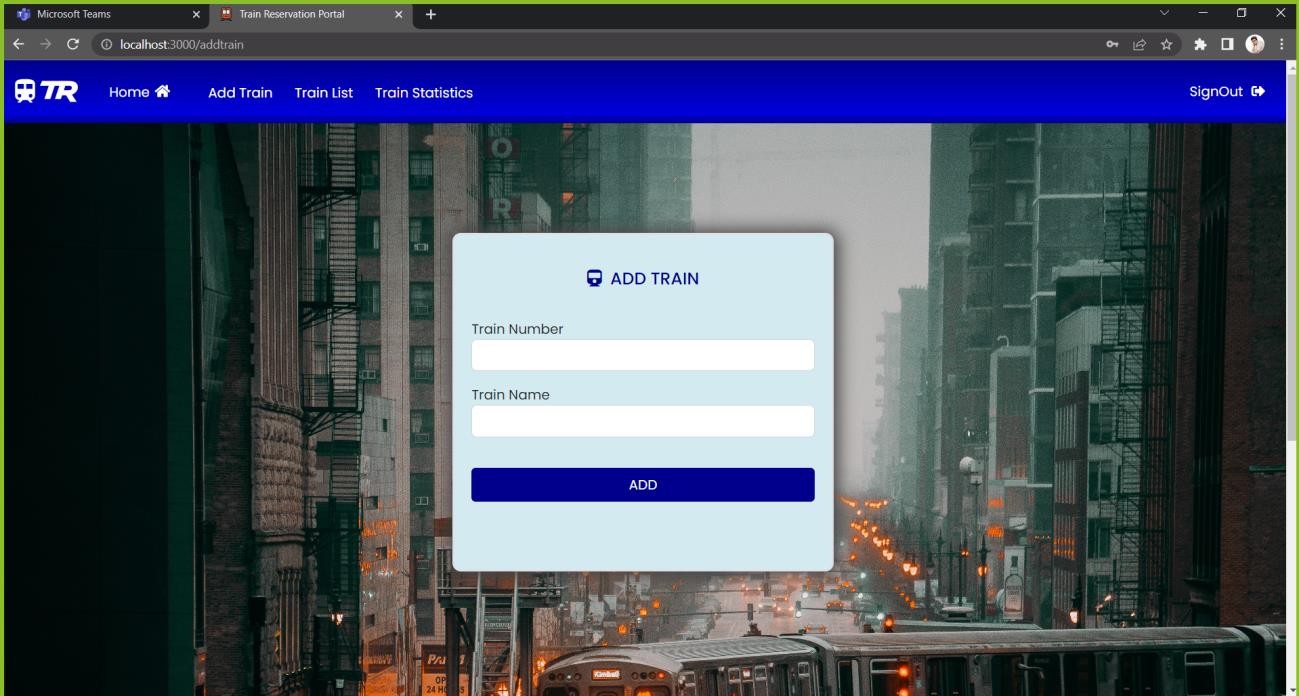


**Forgot Password:**

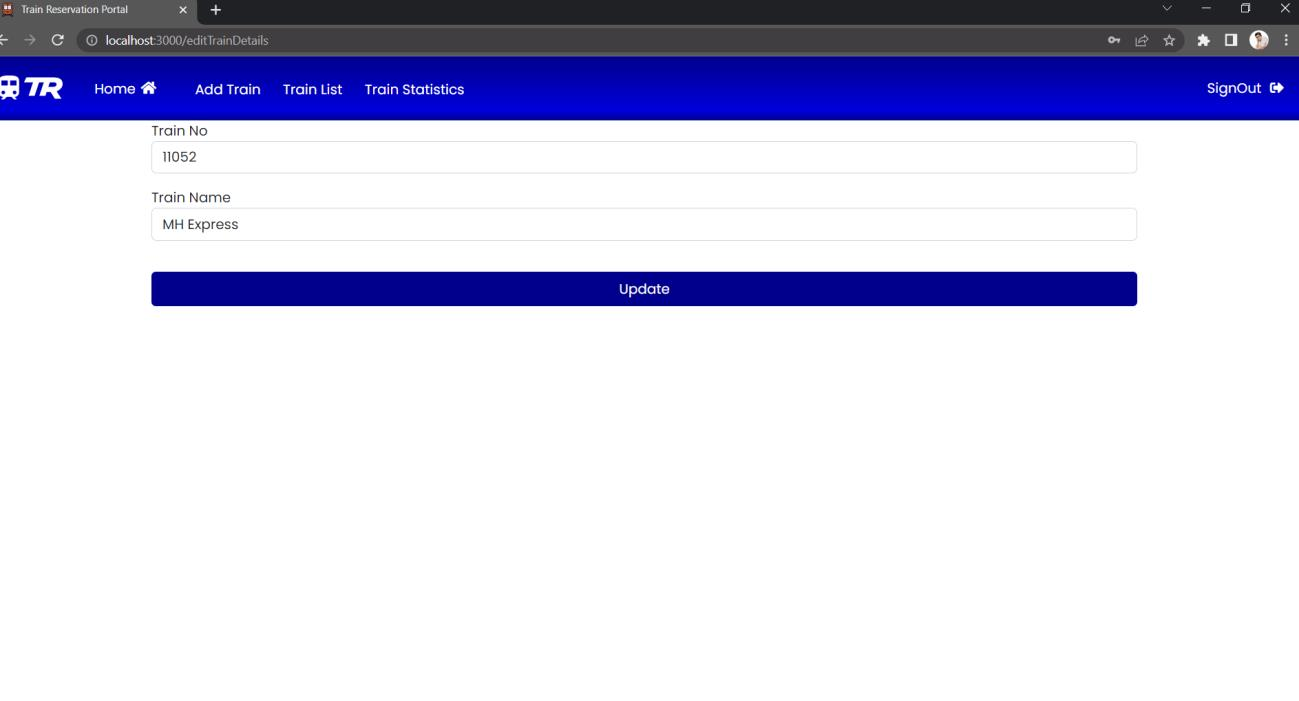
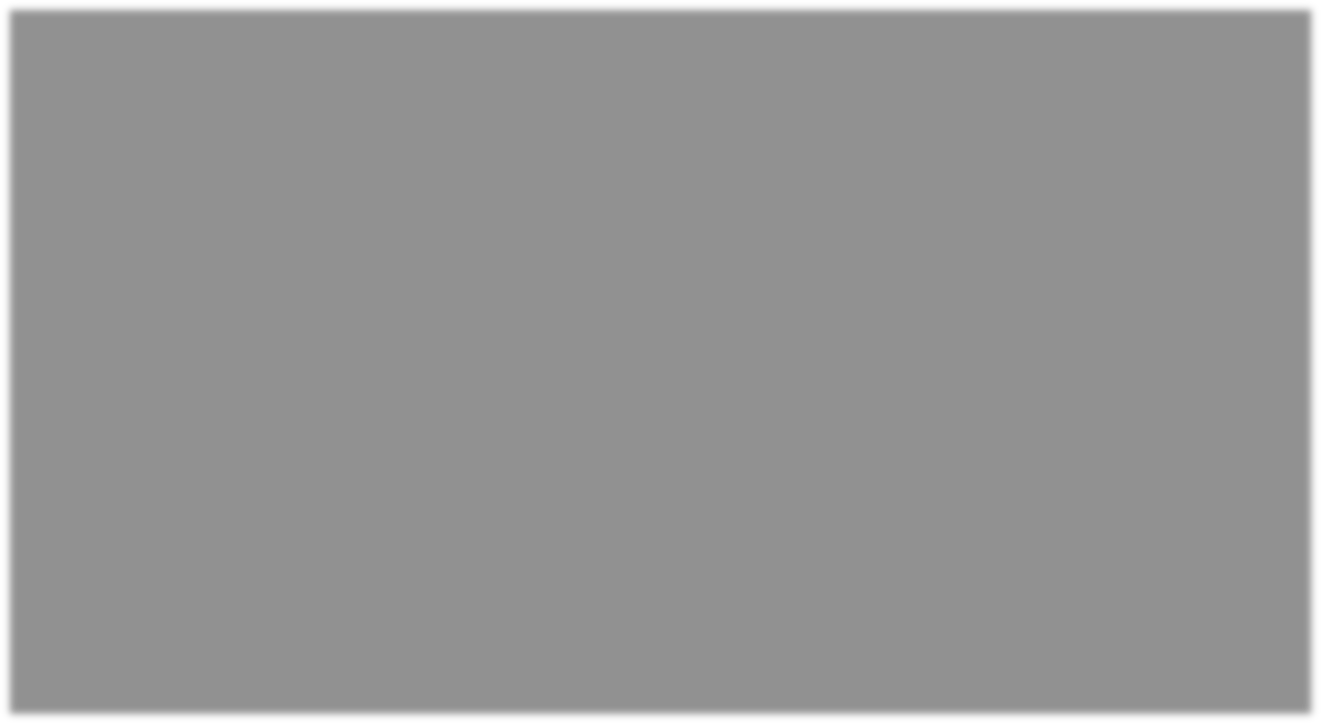


**Add Train:**

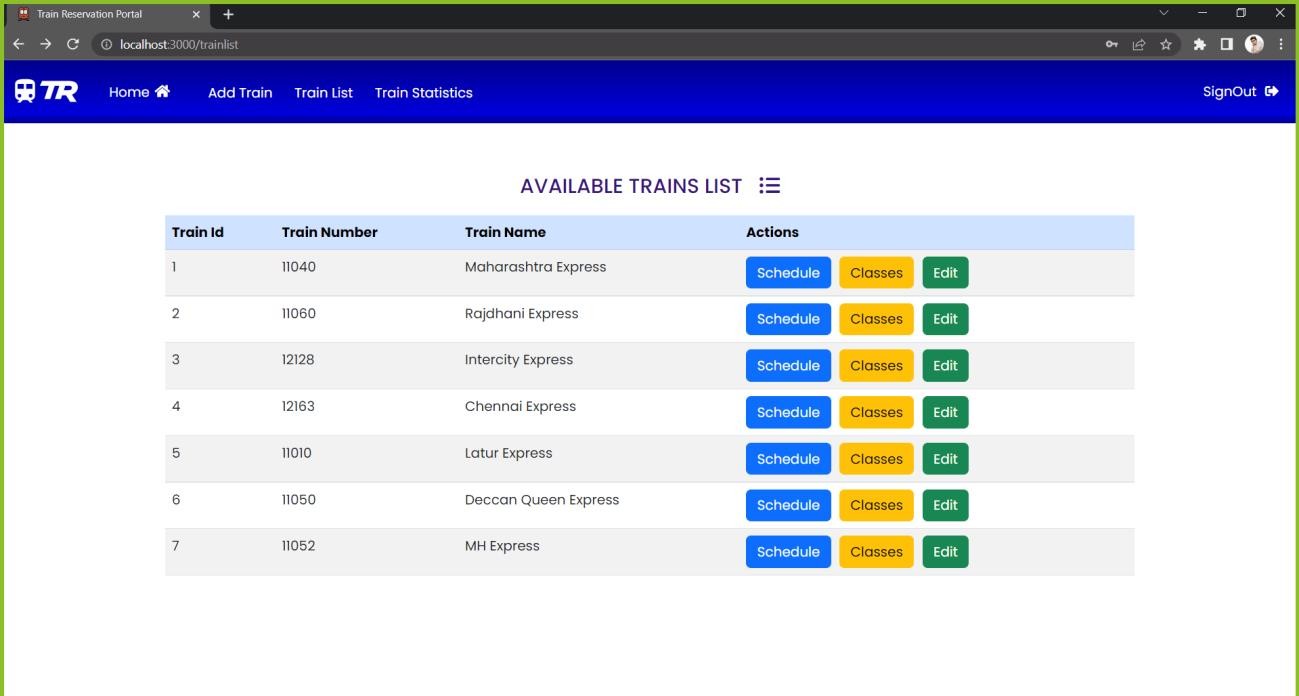
**ADMIN PAGES**



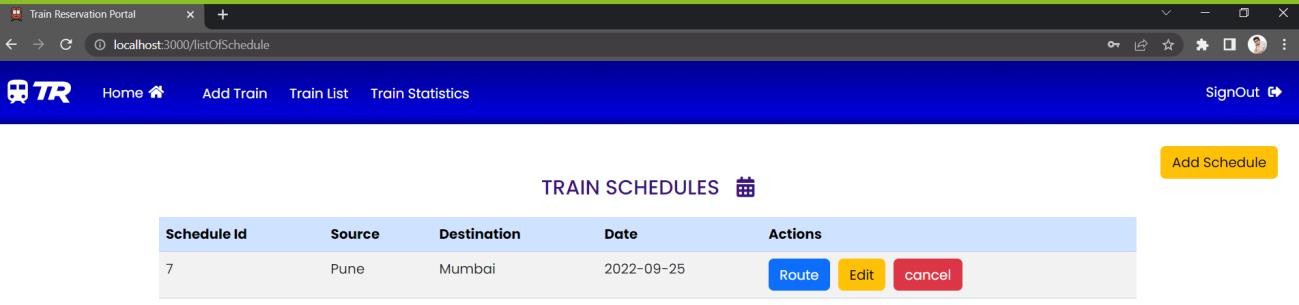
**Edit Train Details:**



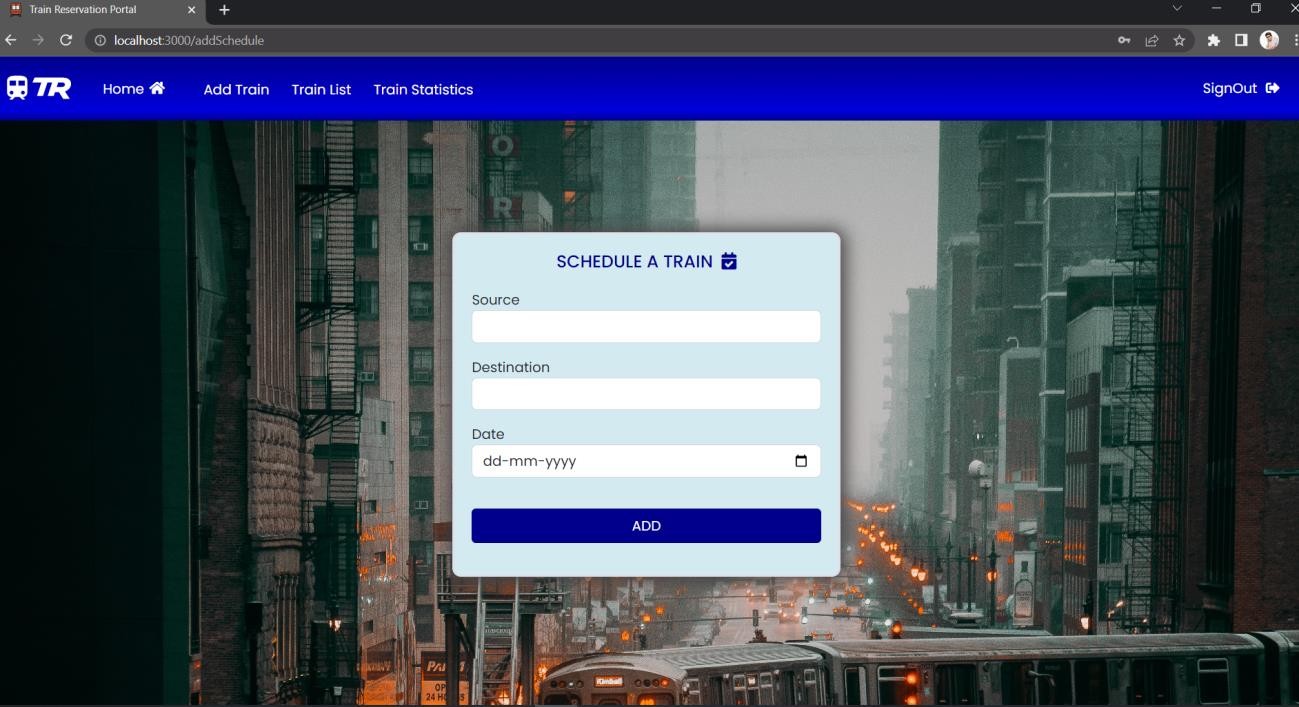
**Train List:**



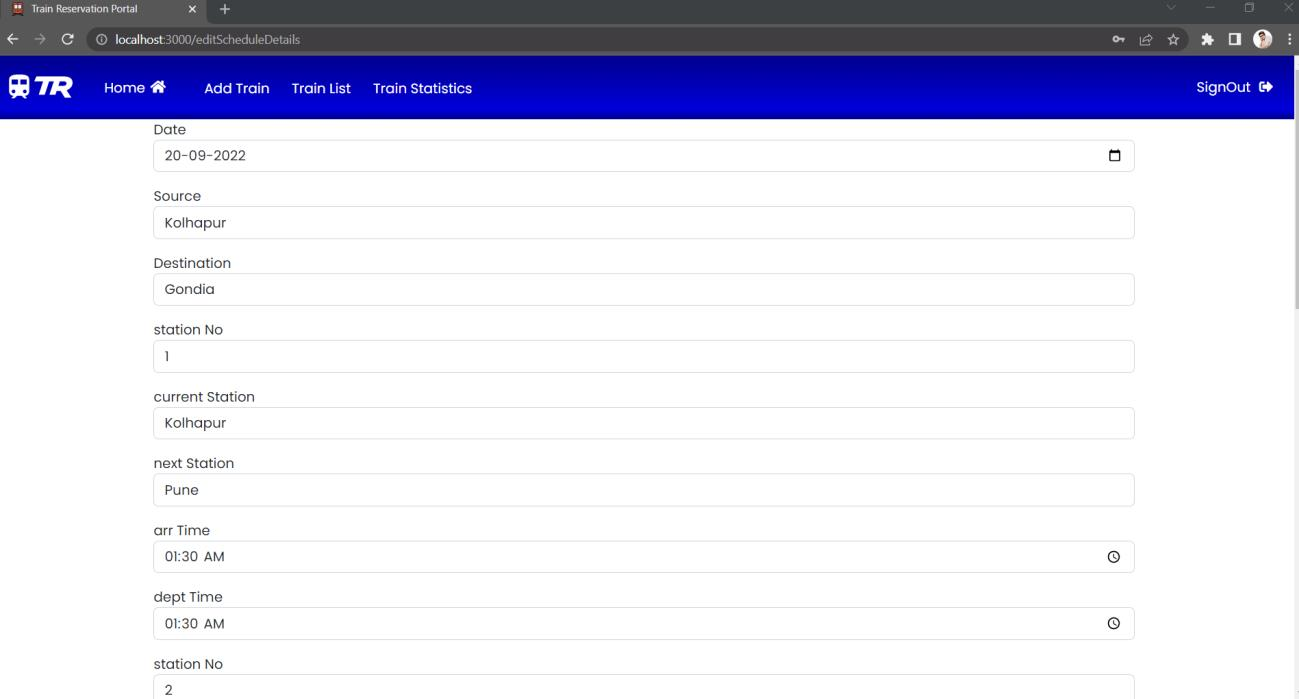
**Schedules List:**



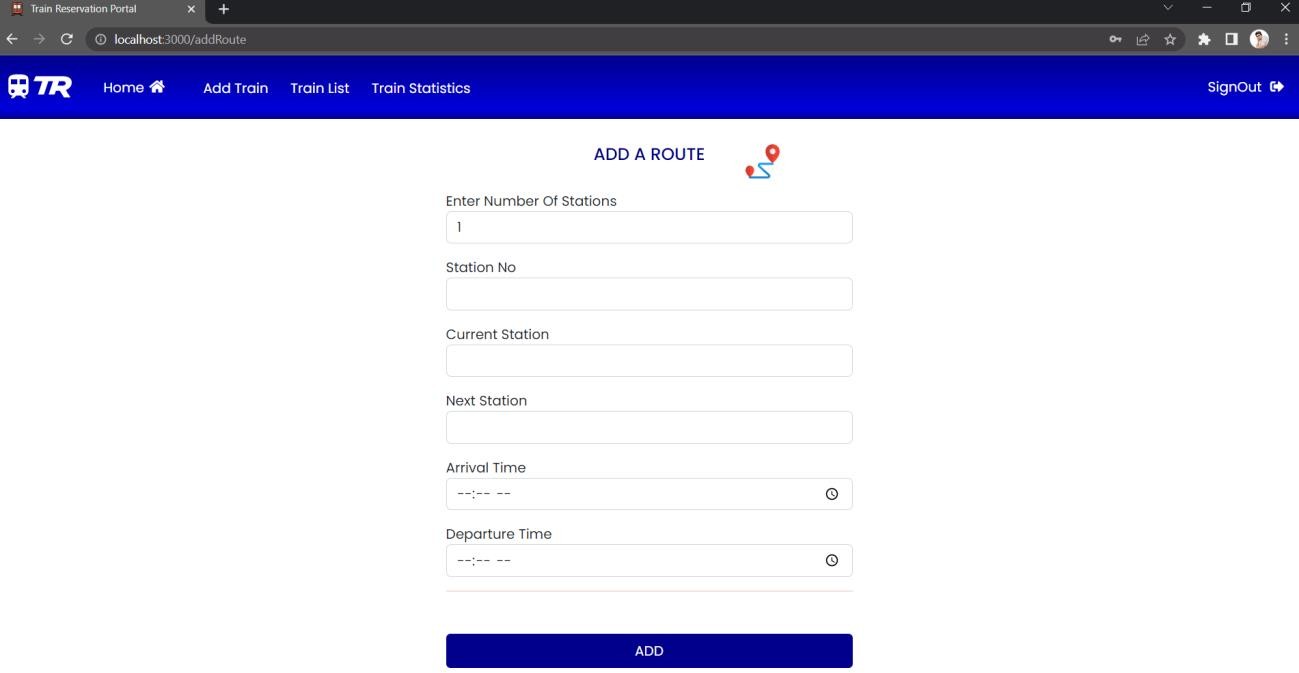
**Add Schedule:**



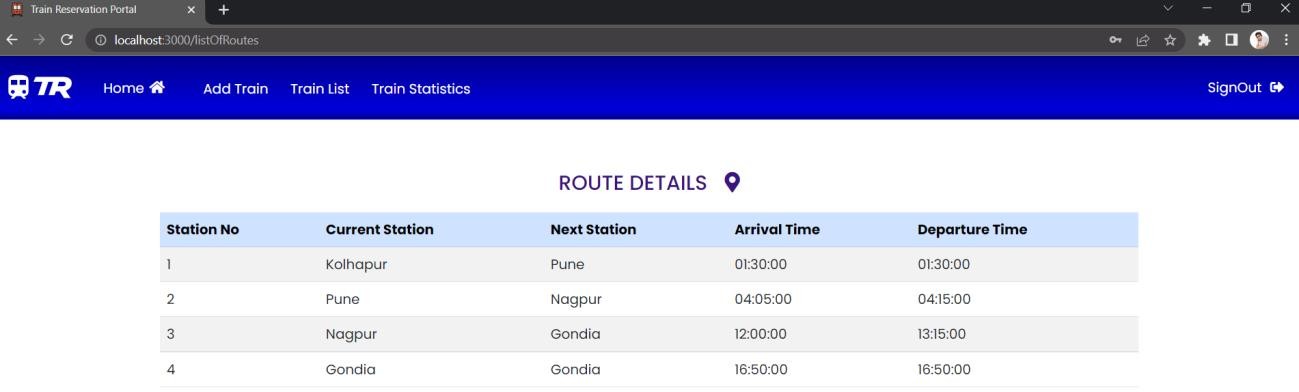
**Reschedule Train:**



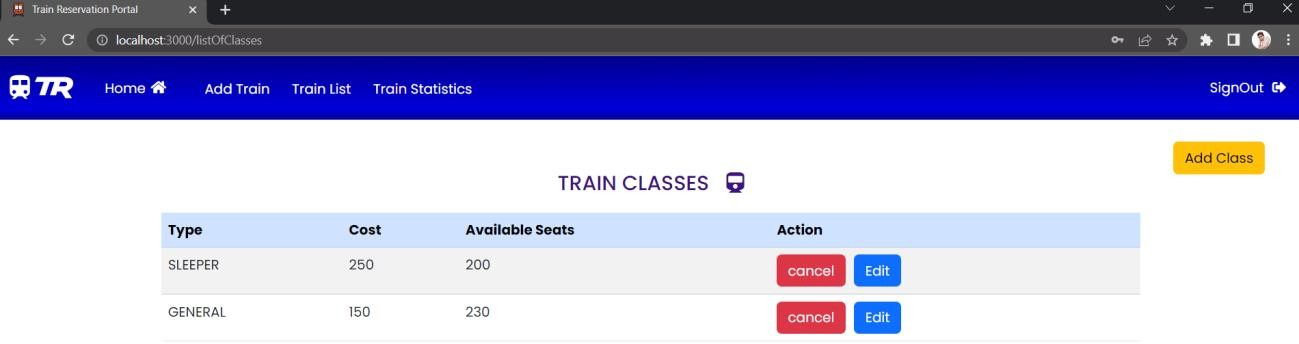
**Add Route:**



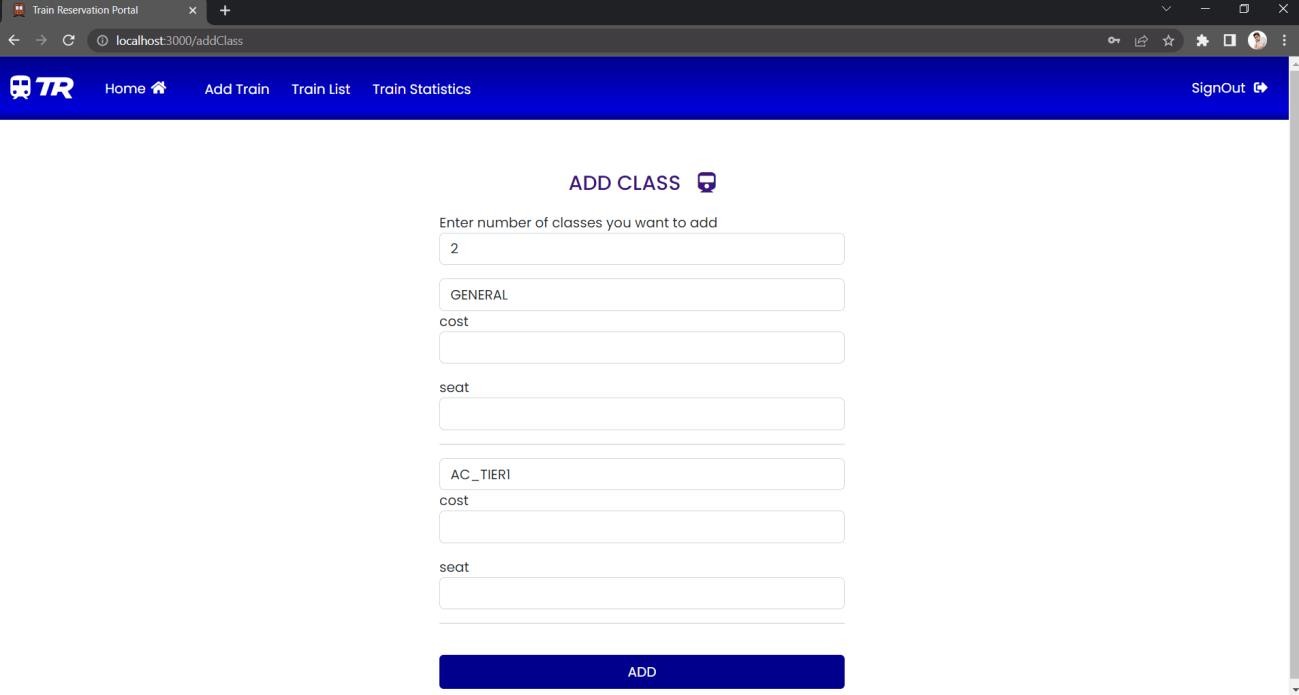
**Route List:**



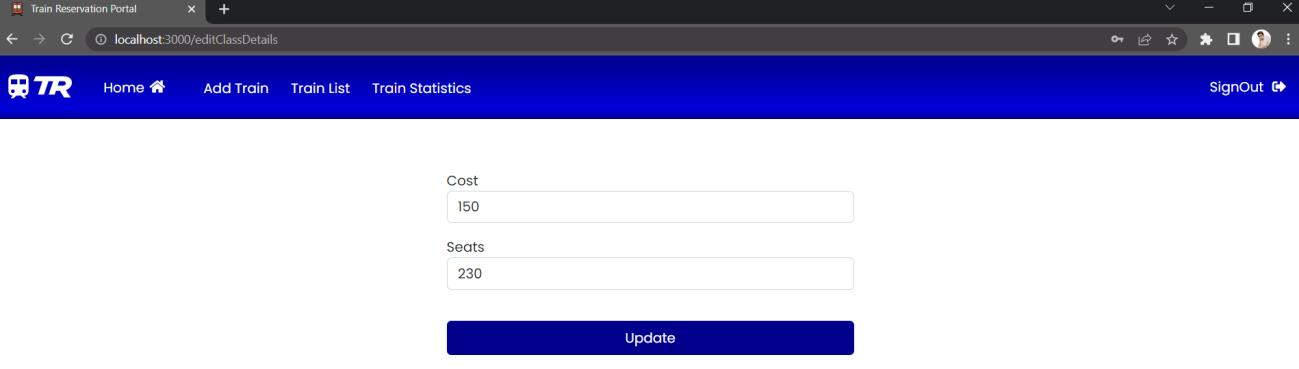
**Train Classes List:**



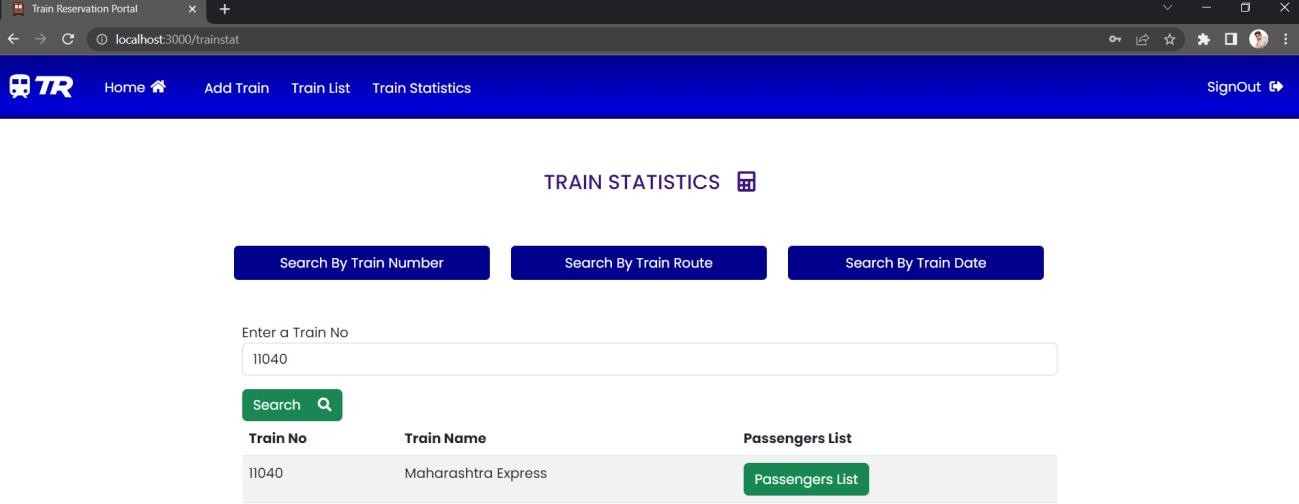
**Add Class:**



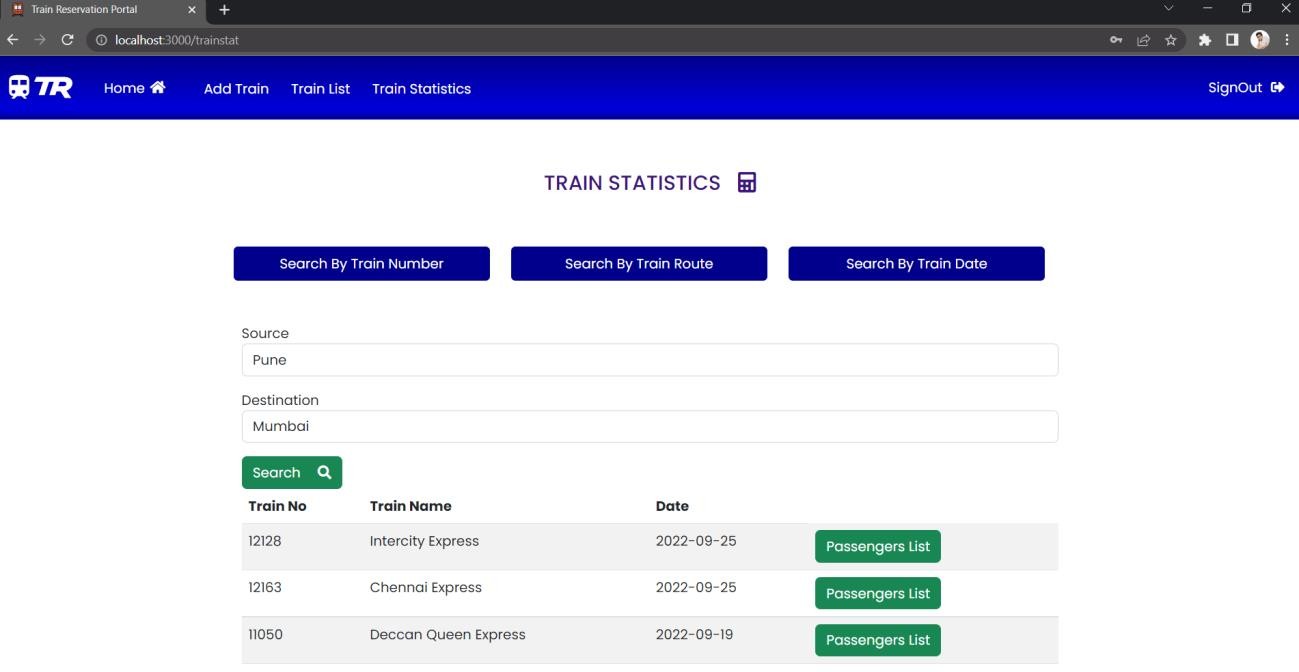
**Edit Class Details:**



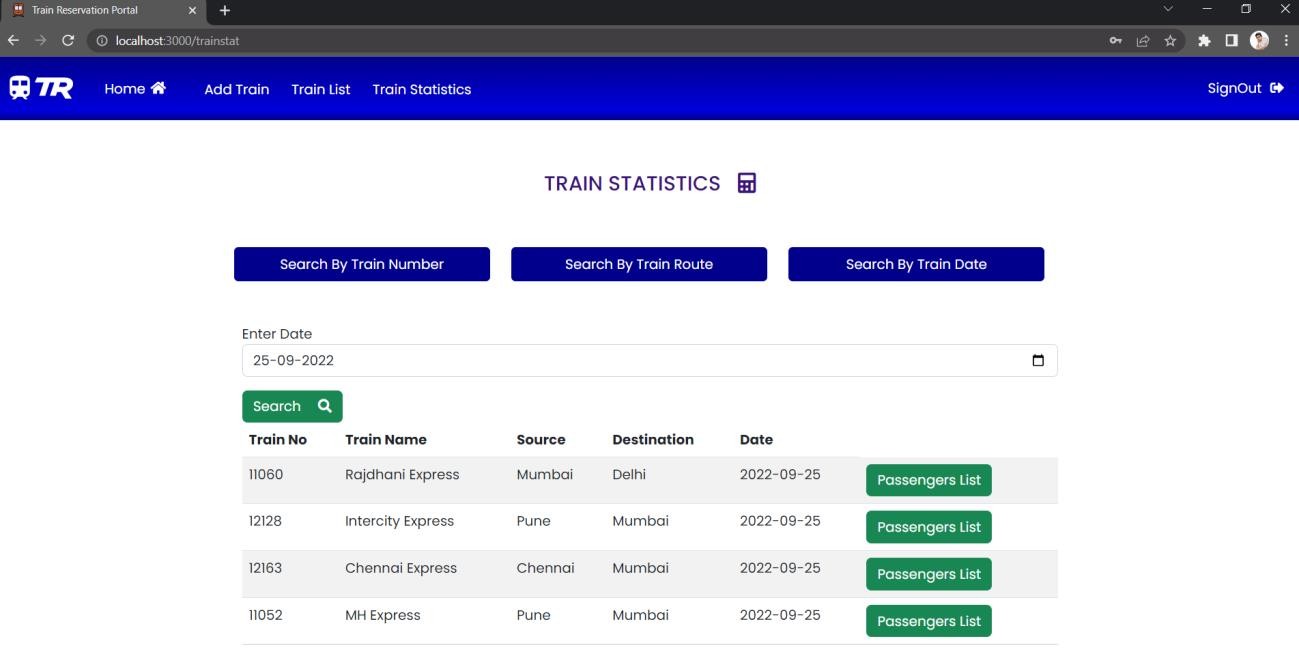
**Search Train By Number:**



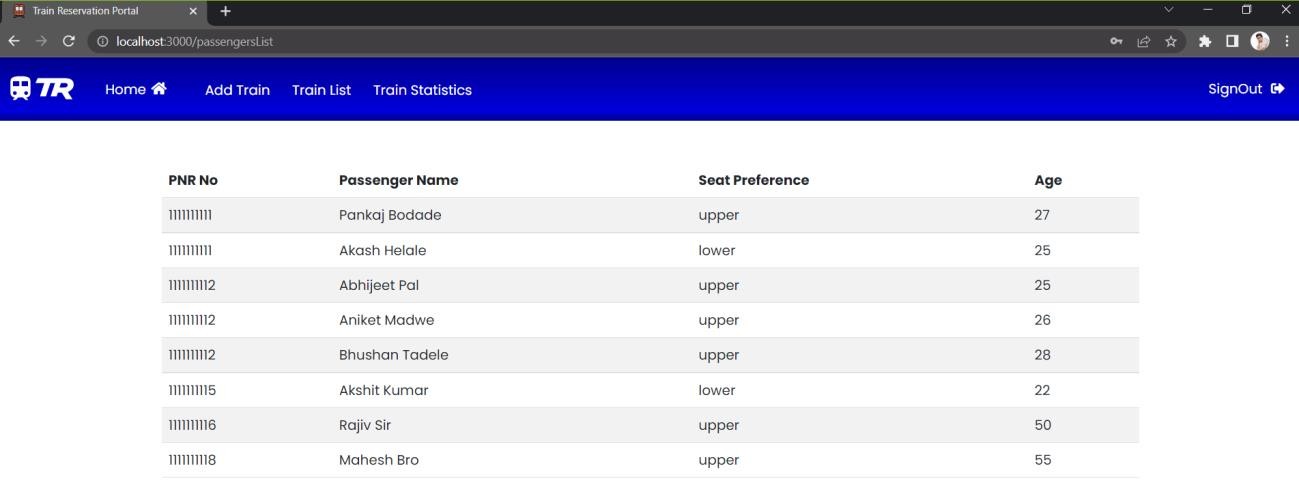
**Search Train By Route:**



**Search Train By Date:**

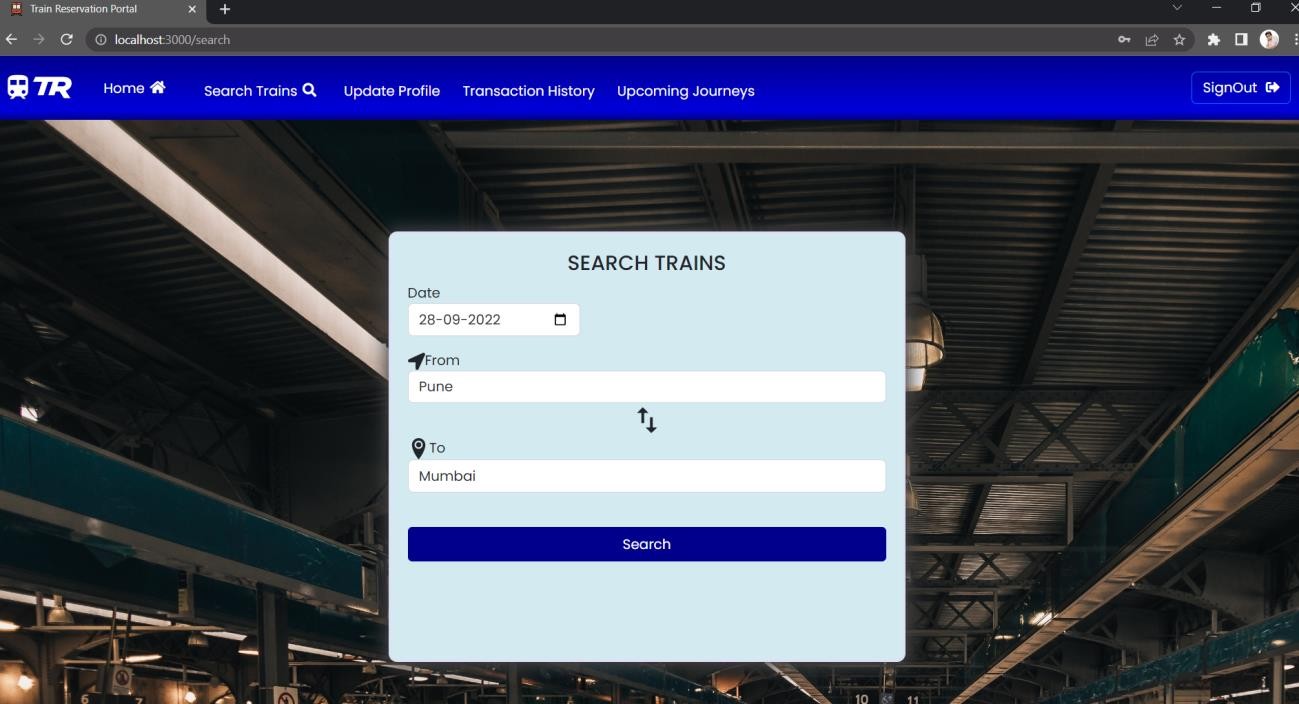


**Passengers List:**

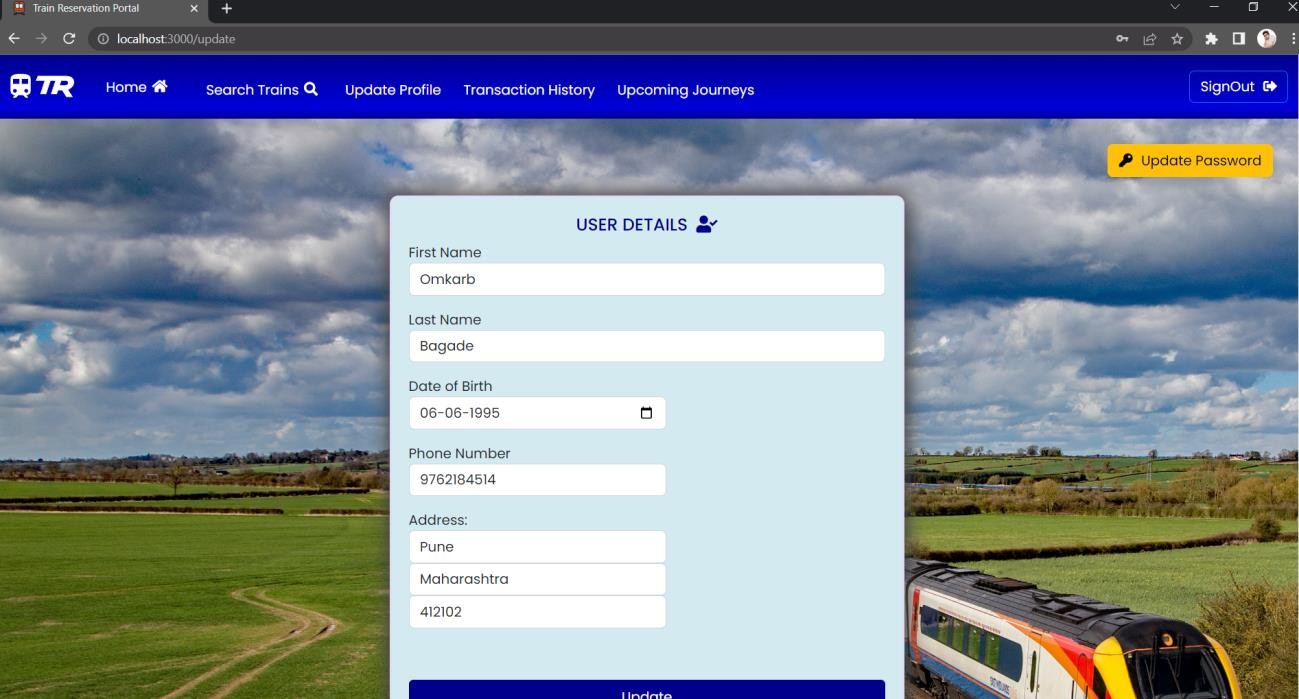


**Search Train:**

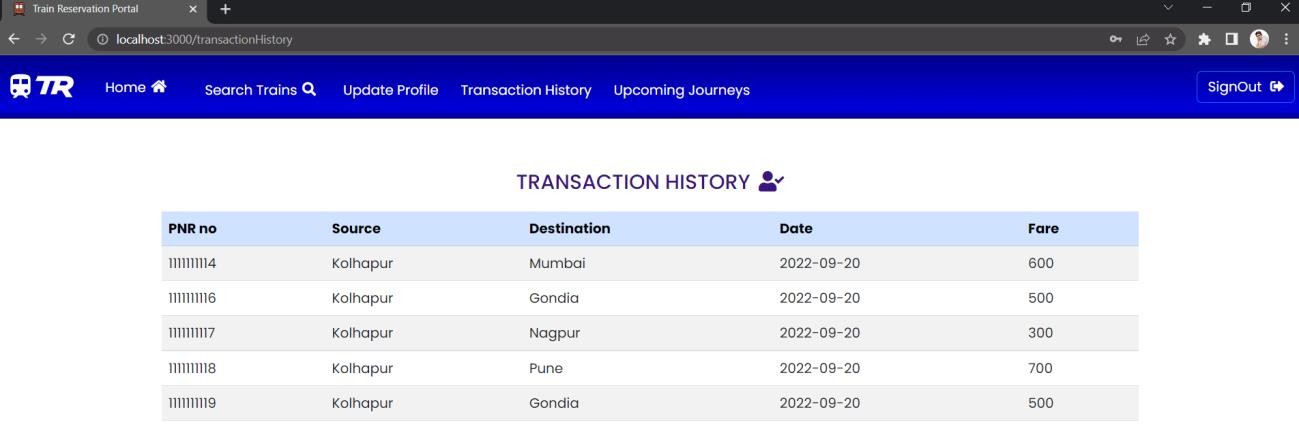
**USER PAGES**



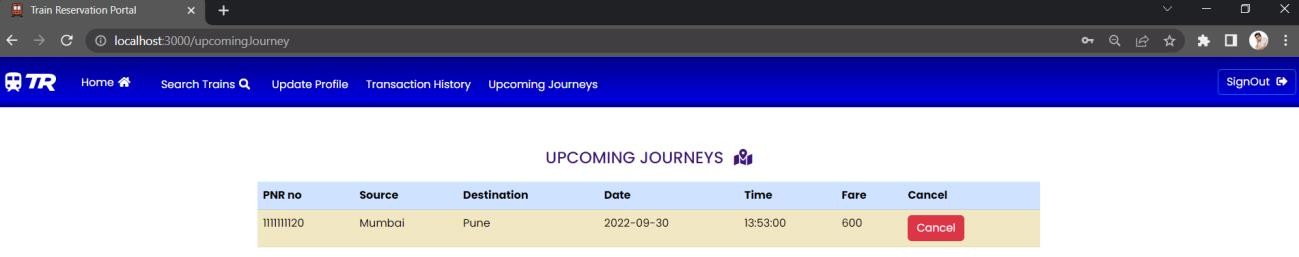
**Update Profile:**



**Transaction History:**



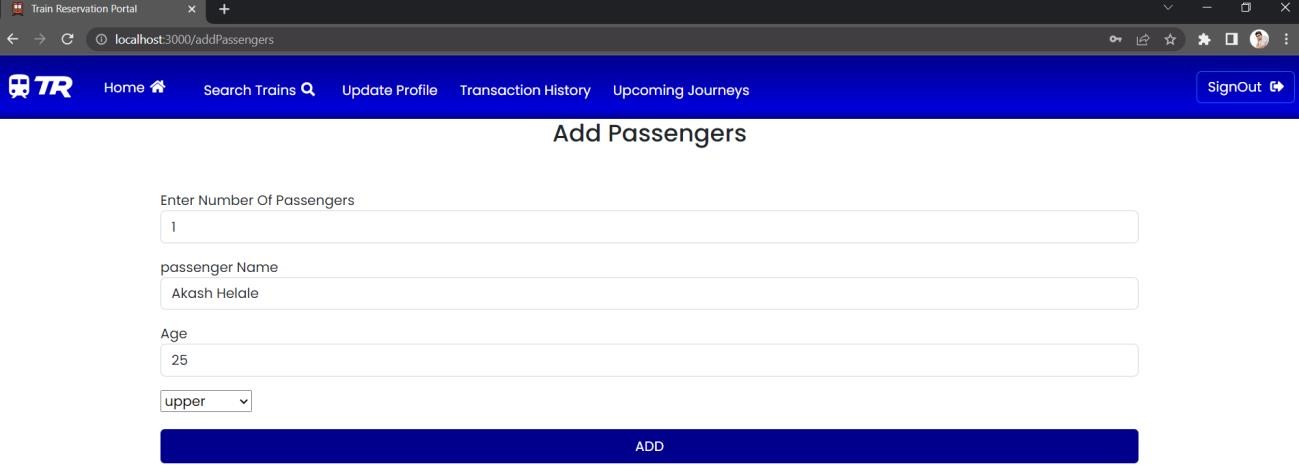
**Upcoming Journeys:**



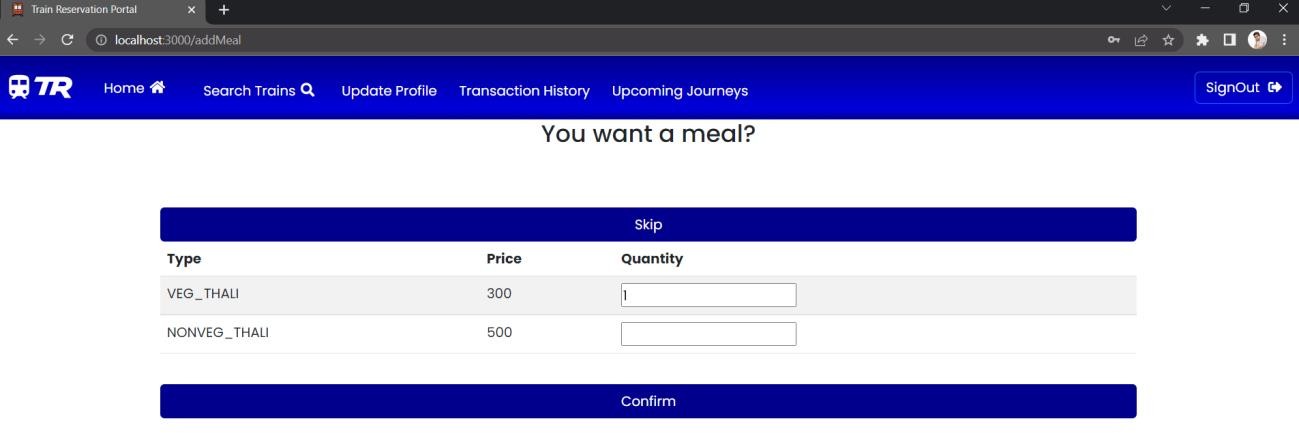
**Available Trains:**



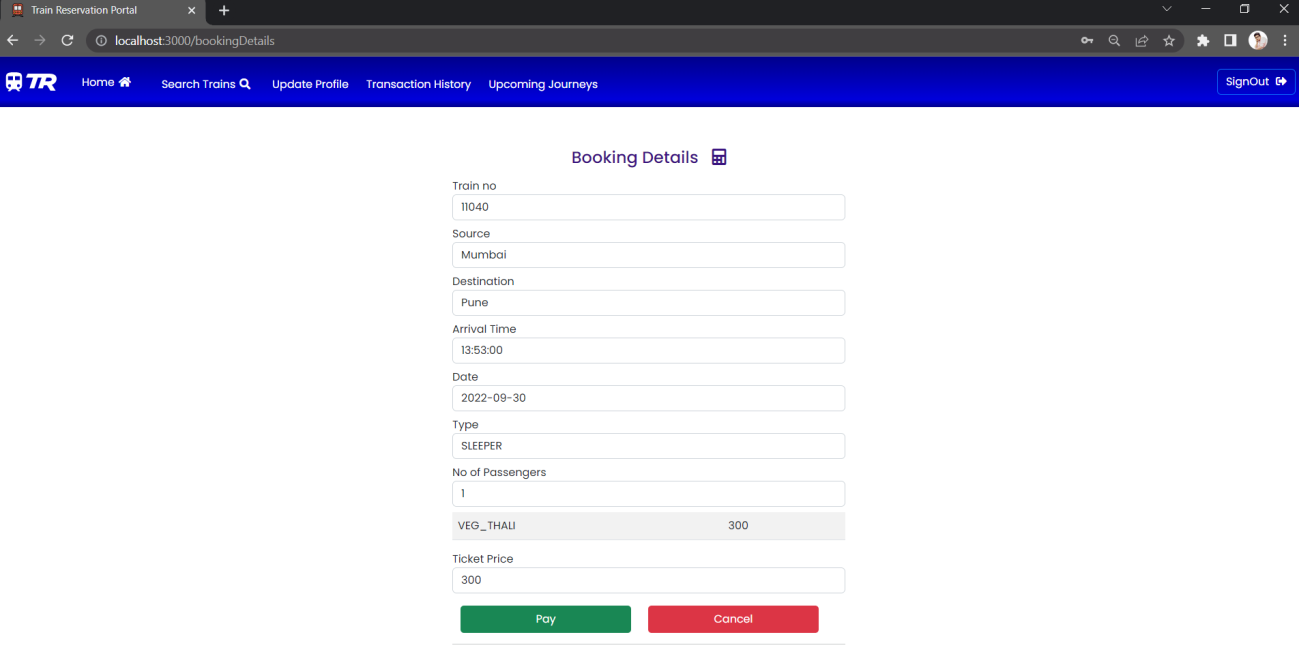
**Add Passenger Details:**



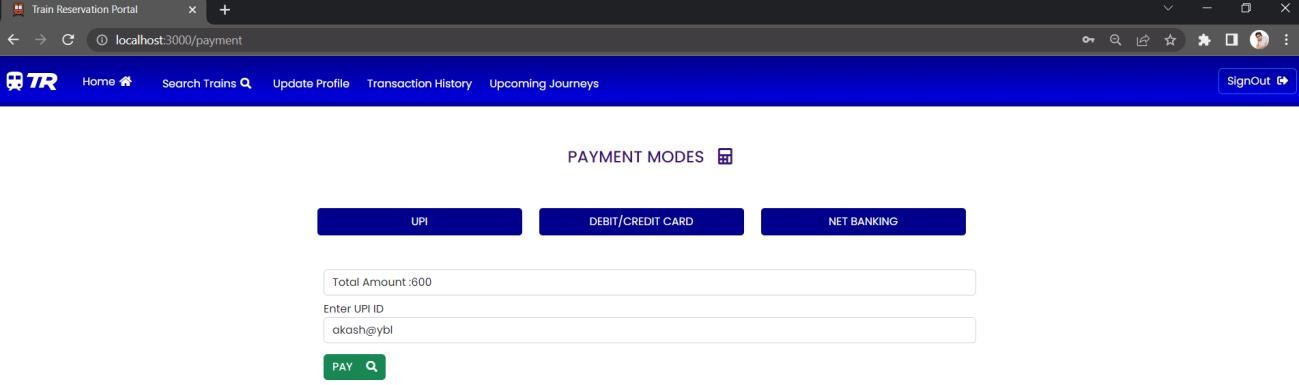
**Meal (Optional):**



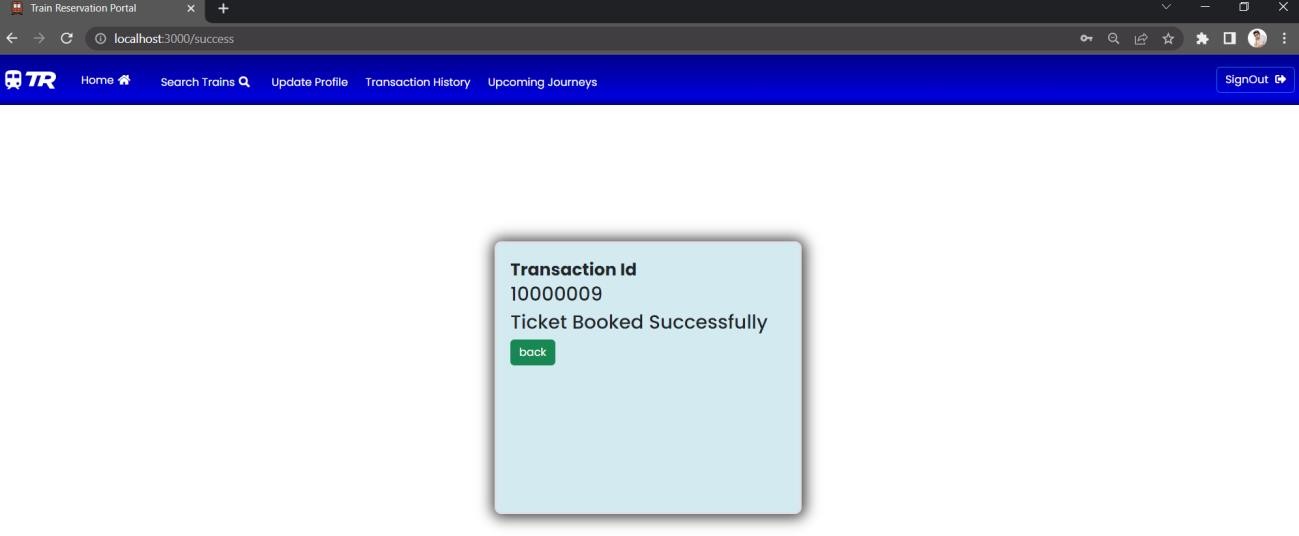
**Booking Details:**



**Payment Modes:**

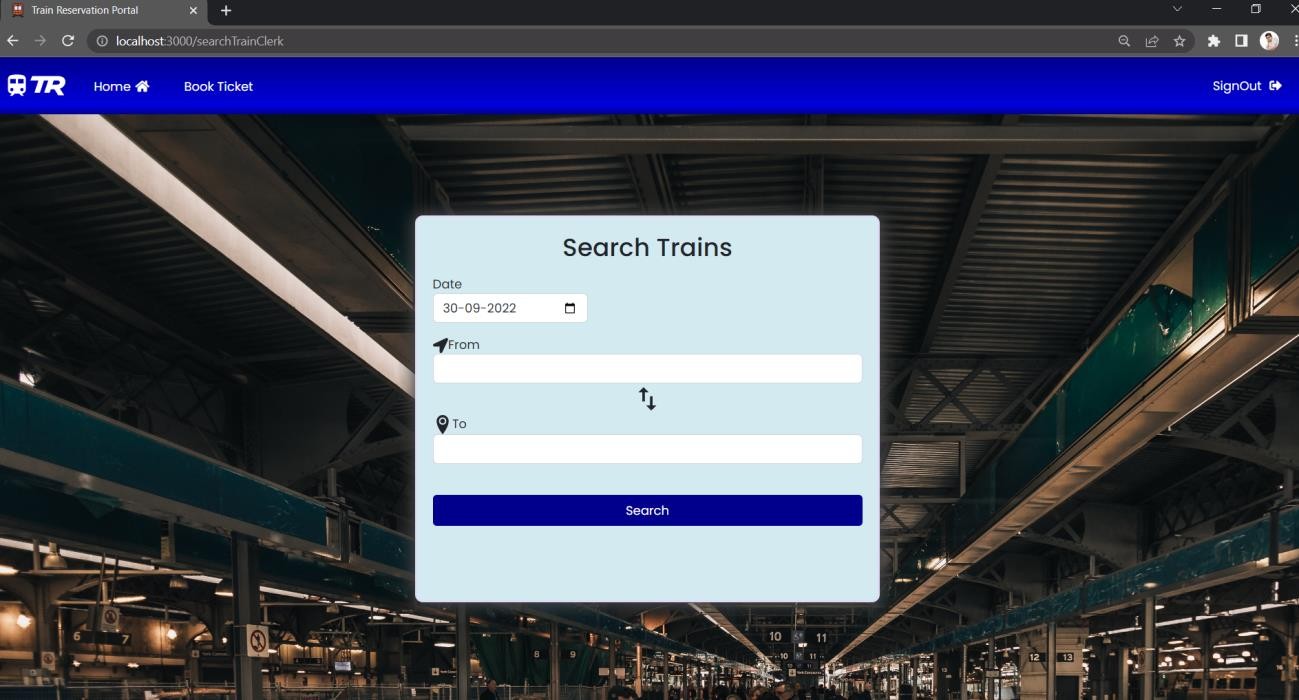


**Transaction Successful :**

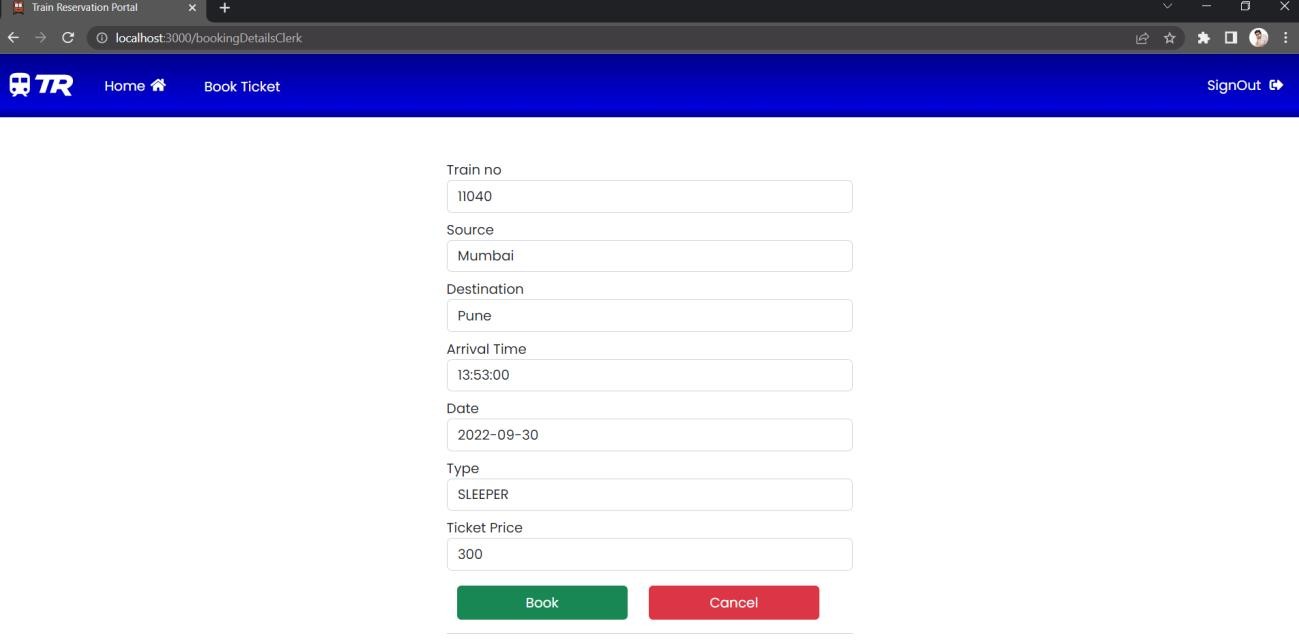


**CLERK PAGES**

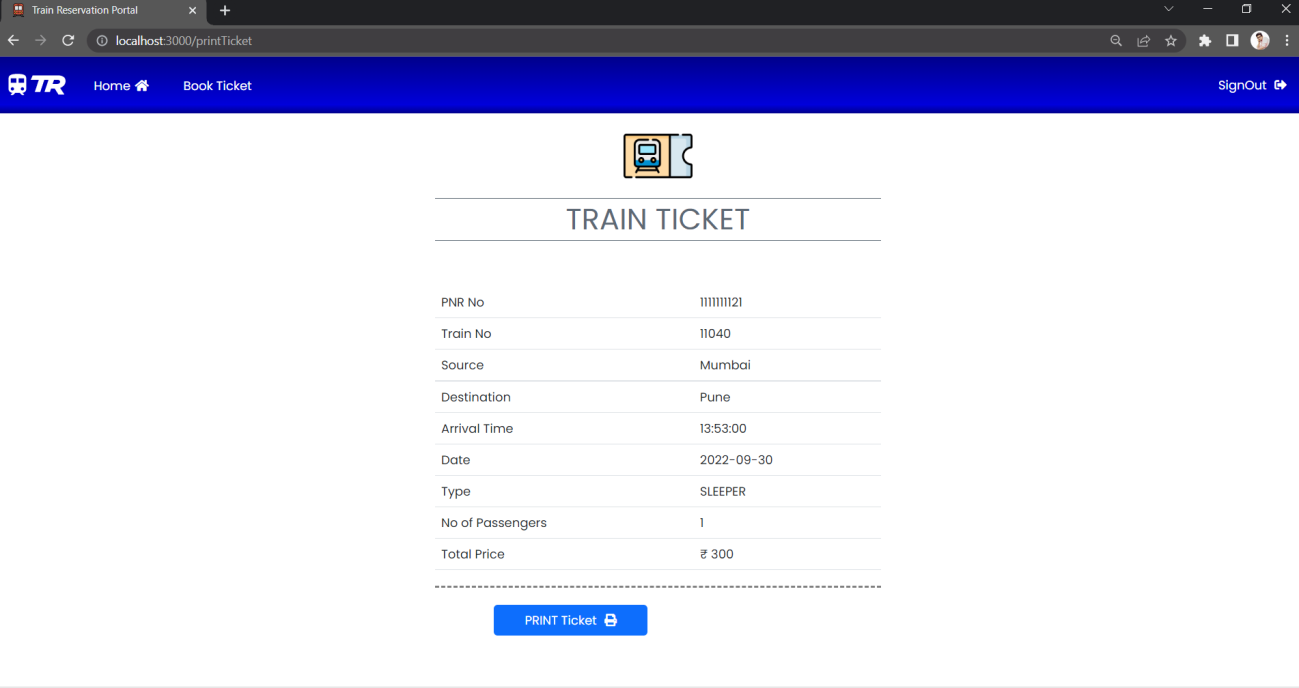
**Search Train:**



**Booking Details:**



**Train Ticket:**



**REFERANCES**

* [http://www.google.com](http://www.google.com/)
* <https://www.w3schools.com/>
* <https://docs.spring.io/spring-framework>
* <https://react-bootstrap.netlify.app/components/navbar>
* [https://stackoverflow.com](https://stackoverflow.com/)