TRAIN RESERVATION SYSTEM

CS23333 – Introduction to OOPS and JAVA Project Report

Submitted by

HARIHARAN MURALI -231001053

BHARATH.S -231001027

Of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

RAJALAKSHMI ENGINEERING COLLEGE, THANDALAM
(An Autonomous Institution)

RAJALAKSHMI ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

Certified that this project titled “Train Reservation System” is the bonafide work of “**HARIHARAN MURALI(231001053) and BHARATH.S(231001027)**” who carried out the project work under my supervision.

SIGNATURE	SIGNATURE
Dr.Priya Vijay	Mrs.Usha S
HEAD OF THE DEPARTMENT	COURSE INCHARGE
Information Technology	Information Technology
Rajalakshmi Engineering College,	Rajalakshmi Engineering College
Rajalakshmi Nagar, Thandalam	Rajalakshmi Nagar, Thandalam
Chennai – 602105	Chennai – 602105

This project is submitted for IT19341 – Introduction to Oops and Java held on _________

INTERNAL EXAMINAR **EXTERNAL EXAMINAR**

TABLE OF CONTENTS

1.1Abstract:

The primary objective of the JDBC-powered Train Reservation System in Java is to provide a reliable and efficient platform for users to seamlessly browse available trains, select seats, and book tickets, all while leveraging the power of Java's database connectivity capabilities. This system ensures data integrity and consistency through JDBC transactions, establishing a secure connection with a relational database for real-time updates on seat availability, train schedules, and reservation status. The user-friendly interface and scalability make it an effective solution for optimizing the train reservation process in various travel environments, ensuring a smooth and hassle-free experience for passengers.

1.2 Introduction:

The Train Reservation System is a cutting-edge software solution designed to revolutionize the traditional manual processes involved in booking and managing train tickets. This innovative system introduces automation to streamline operations, significantly reducing time and operational costs. Administered through a secure authentication protocol, the system ensures that only authorized personnel can access and manage train schedules, seat allocations, and other essential functionalities. By seamlessly integrating technology into the train reservation process, this system provides a more efficient, reliable, and user-friendly experience, marking a significant improvement over conventional methods.

1.3 Purpose:

The purpose of this project is to create an efficient and user-friendly train reservation system that benefits both travelers and train operators. The system aims to:

- Simplify the process of booking train tickets, making it more accessible and convenient for passengers.
- Enhance the travel experience by providing users with train schedules, seat availability, and easy booking options.
- Enable train operators and administrators to manage reservations, train routes, schedules, and seat allocations effectively.
- Store and analyze reservation data to gain valuable business insights and support informed decision-making.

1.4 Scope of the Project:

The envisioned Train Reservation System aims to seamlessly interact with administrators and effectively fulfill all proposed functionalities. Built on Java (JDBC) with an SQL database, the system ensures efficient management of train schedules, routes, seat availability, and reservations, reducing response time for user queries. This project addresses the complexities associated with manual train reservation processes, storing comprehensive information about trains, departure times, seat availability, and booking status. Emphasizing features such as verification, validation, security, and user-friendliness, the system strives to optimize the train reservation experience, offering a comprehensive solution for effective travel management.

1.5 Software Requirement Specification:

1. Introduction:
The **Train Reservation System** is designed to manage all aspects of train ticket reservations, including train schedules, routes, seat availability, and booking details. It serves as an automated solution to replace traditional manual reservation processes, improving the efficiency of train travel management.

2. Document Purpose:
This **SRS** document outlines the software requirements for the **Train Reservation System**, covering design decisions, architectural design, and detailed design necessary for successful implementation. It provides a clear understanding of the system's structure and outlines the necessary components to ensure smooth implementation and long-term support.

3. Product Scope:
The **Train Reservation System** is developed for widespread use in railway networks, aiming to replace outdated, paper-based reservation systems. It streamlines the train reservation process by providing a comprehensive solution for train operators and travelers alike. The system is designed to be flexible, allowing easy updates to train schedules, seat allocations, and passenger bookings, ultimately optimizing the reservation experience for both administrators and users.

4. Definitions, Acronyms, and Abbreviations:

- **TRS** - Train Reservation System
- **SRS** - Software Requirements Specification
- **DBMS** - Database Management System
- **API** - Application Programming Interface
- **UI** - User Interface

References and Acknowledgement:

[1] https://www.geeksforgeeks.org/online-railway-ticket-reservation-system/

Overall Description:

The **Train Reservation System** provides authorized users with seamless access to train ticket records, streamlining the reservation process for passengers and train operators. The system simplifies operations across various train services, optimizing the process of booking tickets, managing seat allocations, and monitoring train schedules. Designed for use in a wide range of travel environments, the system enhances the efficiency and accuracy of the entire reservation workflow, making it easier for travelers to plan their journeys and for train operators to manage seat availability and bookings in real-time.

Product Perspective:

The Train Reservation System utilizes a client/server architecture to deliver a seamless and efficient reservation experience. Designed for cross-platform compatibility, it functions smoothly on major operating systems such as Microsoft Windows and Linux. The front end employs a combination of **HTML**, **CSS**, and **Bootstrap**, ensuring a responsive and visually appealing graphical user interface (GUI) that enhances user experience.
On the back end, the system is powered by **Java (J2EE)** with **Servlets** and **JDBC** for robust application logic and database interaction. The database is managed using **Oracle SQL**, which provides reliable storage and handling of train schedules, seat allocations, and reservation data. The integration between the front end and back end ensures real-time updates and synchronization,

enabling users and administrators to access accurate and current information at all times.
This modern technology stack guarantees scalability, maintainability, and user-friendly interactions.

Product Functionality:

The Train Reservation System is built to provide an intuitive platform for passengers and administrators to manage train reservations and schedules. The system aims to enhance user experience and operational efficiency by offering the following features:

- **Train Schedule Viewing:** Check detailed train schedules for planning journeys.
- **Train Search:** Search for trains based on source, destination, and time.
- **Seat Availability:** View seat availability for selected trains.
- **Train Timings:** Display arrival and departure times for trains.
- **Fare Enquiry:** Retrieve fare details for different classes of service.
- **Trains Between Stations:** Find available trains operating between specific stations.
- **Online Seat Booking:** Book train tickets online with ease.
- **Secure Login and Logout:** Provide secure access to the system for users and administrators.
- **Password Management:** Enable password changes for users and administrators.
- **Payment Gateway Integration:** Support secure and reliable online payments for bookings.
- **Ticket Booking History:** Allow users to view their past bookings.

Admin Features:
Administrators have enhanced access to manage the system effectively:

1. **Login:** Secure login to access admin functionalities.
2. **Add Trains:** Add new trains and their schedules to the system.
3. **Update Trains:** Modify train schedules and information as needed.
4. **Remove/Cancel Trains:** Remove trains or cancel specific services.
5. **View Trains:** Access detailed information about existing trains.
6. **Profile Management:** Edit admin profile details.
7. **Logout:** Securely exit the system.

User Features:

Users have access to essential functionalities for booking and managing their journeys:

1. **Register:** Create a new user account.
2. **Login:** Securely log in to the system.
3. **View Trains:** Browse available trains and schedules.
4. **Check Seat Availability:** View real-time seat availability for selected trains.
5. **Search Trains:** Find trains based on stations or timings.
6. **Train Availability and Fare Enquiry:** Check train availability and fare details between stations.
7. **Book Tickets:** Reserve seats for their journey.
8. **View Booking History:** Access past ticket bookings.
9. **View and Update Profile:** Manage personal profile information.
10. **Change Password:** Update account password securely.
11. **Logout:** Log out of the system safel

<u>User and Characteristics:</u>

Qualification: Users should have at least basic educational qualifications, such as matriculation, and be comfortable with English.
Experience: Familiarity with the university registration process is advantageous.
Technical Experience: Users are expected to have elementary knowledge of computers for optimal system interaction.

<u>Operating Environment:</u>

Hardware Requirements:
- Processor: Any Processor over i5
- Operating System: Windows 8, 10, 11
- Processor Speed: 2.0 GHz
- RAM: 4GB
- Hard Disk: 500GB

Software Requirements:
- Database: SQL PLUS
- Frontend: HTML,CSS,BOOTSTRAP
- Technology: Java (JDBC)

Constraints:

- System access limited to administrators.
- Delete operation restricted to administrators without additional checks for simplicity.
- Administrators must exercise caution during deletion to maintain data consistency.

User Interface:

The Train Reservation System provides user-friendly, menu driven interfaces for:

- **Admin Register:**

Enables the registration of new administrators for efficient system management.

- **Admin Login:**

Allows existing administrators to securely log into their accounts to access administrative functionalities.

- **Add Train:**

Provides functionality to store new train details, such as train number, name, route, and schedule.

- **View Train:**

Facilitates the viewing and updating of existing train information to ensure accuracy.

- **Delete Train:**

Permits the removal of outdated or canceled train entries from the system.

- **Add Reservation:**

Allows administrators to create new reservations for passengers if needed.

- **Update Reservation:**

Grants administrators the ability to view and modify existing reservations to address user requests or errors.

Hardware Interface:

- Compatible with any version of Windows 8, 10, 11.

Software Interface:

a) MS-Windows Operating System

b) HTML,CSS,BOOTSTRAP for designing the front end

c) SQL Plus for the Database

d) JDBC for backend

e) Eclipse EE

Functional Requirements:

1. Login Module (LM):

- Users (administrators) access the Login Module.
- LM supports secure user login using a username and password.
- Passwords are masked during entry to ensure security.
- Successful login verification by the system is required for access.

---

2. Registered Users Module (RUM):

- After successful login, administrators can navigate through the application.
- Administrators can view detailed information about trains, schedules, seat availability, and reservations.
- Administrators can update and maintain train details, including modifying schedules, seat allocations, and availability.

---

3. Administrator Module (AM):

- Upon successful login, the system displays administrative functions.
- Functions include adding, updating, and deleting train details.
  - The **"Add" function** allows administrators to input new train details, such as train name, route, and timings.
  - The **"Update" function** enables administrators to modify existing train details, including schedules and seat configurations.
  - The **"Delete" function** permits the removal of canceled or outdated train entries.
- All add, update, or delete requests are processed by the Administrator Module, which communicates with the Server Module (SM) for necessary database updates.

---

4. Server Module (SM):

- SM serves as the intermediary between various system modules and the database (DB).

- It receives requests from different modules and ensures appropriate data handling.
- SM validates and processes requests, such as adding new trains, updating schedules, or checking seat availability.
- It manages all database interactions, ensuring data consistency and integrity, particularly for train schedules, seat availability, and reservation data.

Non-functional Requirements:

Performance:

- The system must handle real-time train reservation requests efficiently, ensuring a response time of less than **2 seconds** for seat selection and confirmation.
- Safety-critical failures, such as payment processing errors, must be resolved immediately to ensure a smooth user experience and prevent transaction issues.

Reliability:

- The system is mission-critical; in the event of abnormal operations or downtime, immediate measures must be taken to identify and rectify the issue.
- System recovery mechanisms should ensure restoration of normal functionality with minimal disruption to users.

Availability:

- Under normal operating conditions, user requests for train reservations, including seat selection and payment, must be processed within **2 seconds** to maintain a seamless booking experience.
- Instant feedback on reservation status and confirmation must be communicated to users to enhance their confidence in the system.

Security:

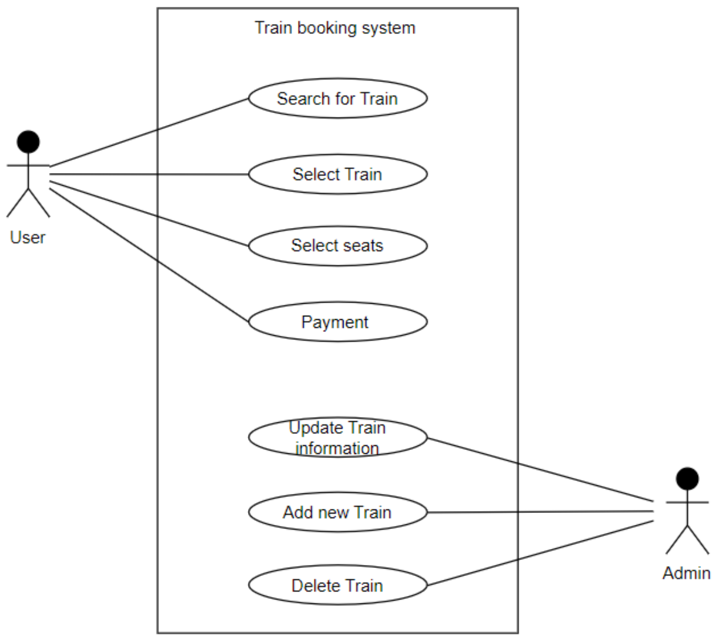
- A robust security framework must be implemented on the server side to prevent unauthorized access, protect user payment details, and ensure the integrity of the reservation system.
- User privacy must be safeguarded, with personal details securely stored and managed to maintain confidentiality and compliance with data protection standards.
- Multi-factor authentication and encrypted communication protocols should be used to enhance system security.

Maintainability:

- Comprehensive design documents outlining software architecture and database maintenance procedures must be available to enable efficient updates and modifications.
- Administrative access should be provided for maintaining both front-end and back-end systems, ensuring the platform's long-term functionality and adaptability to future requirements.
- Regular system audits and updates should be performed to keep the software aligned with evolving user and security needs.

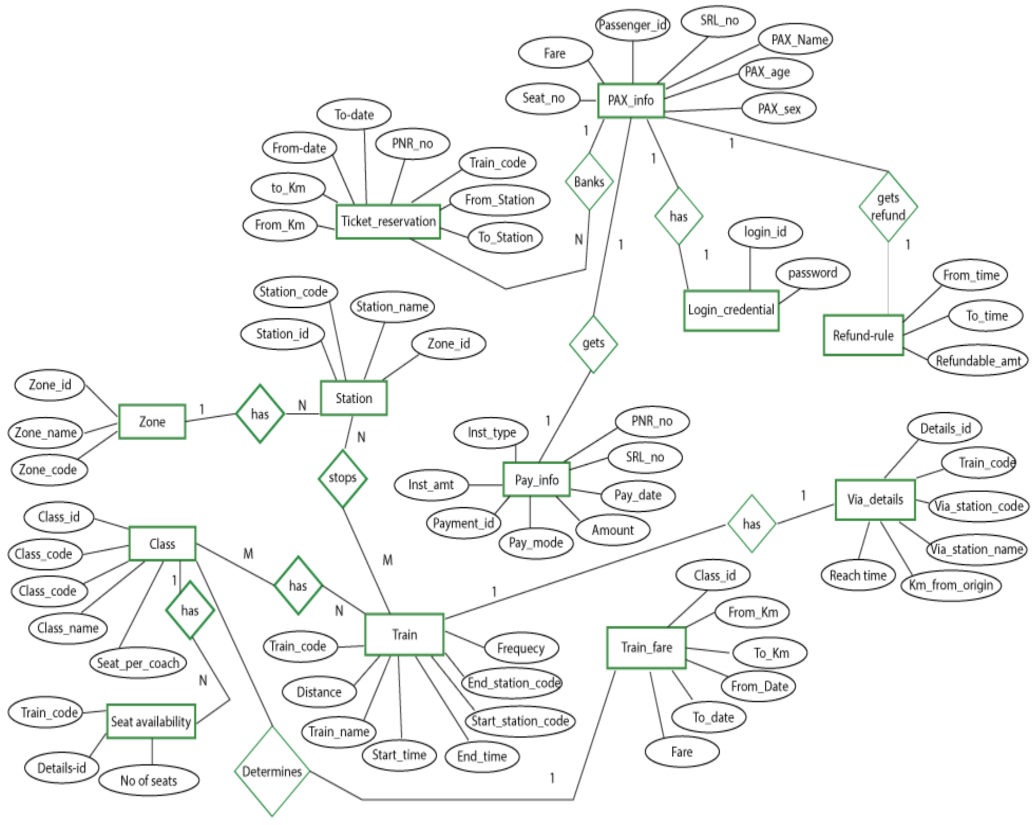
2.System Flow Diagrams:

2.1.Use Case Diagrams :

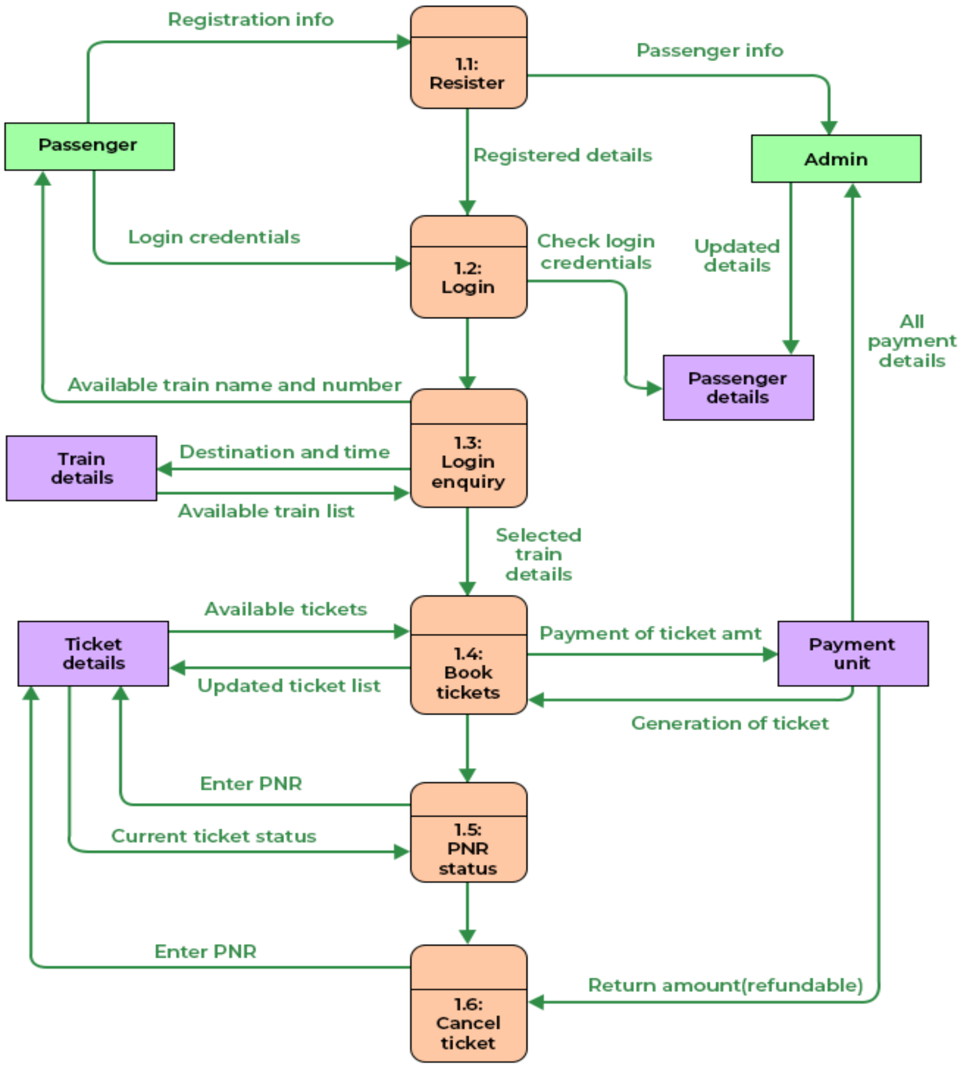


2.2 Entity-relationship diagram:

E-R (Entity-Relationship) Diagram is used to represents the relationship between entities in the table.



2.3 Data-flow diagram:



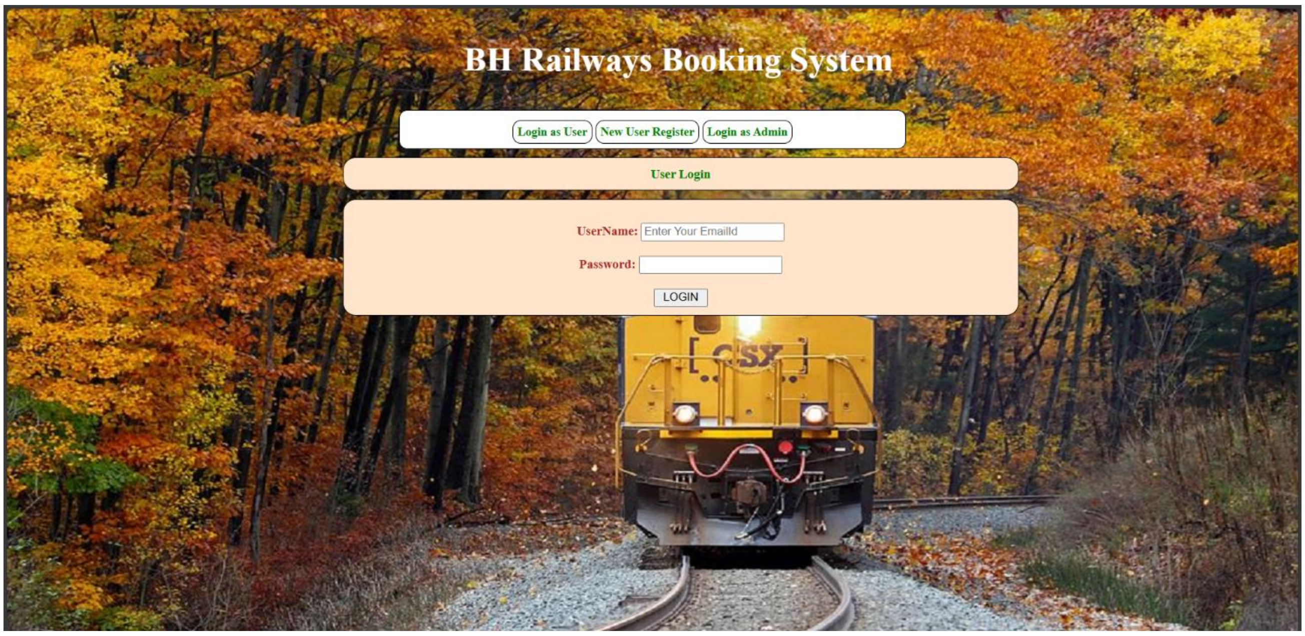
3. Module description:

Admin Module:

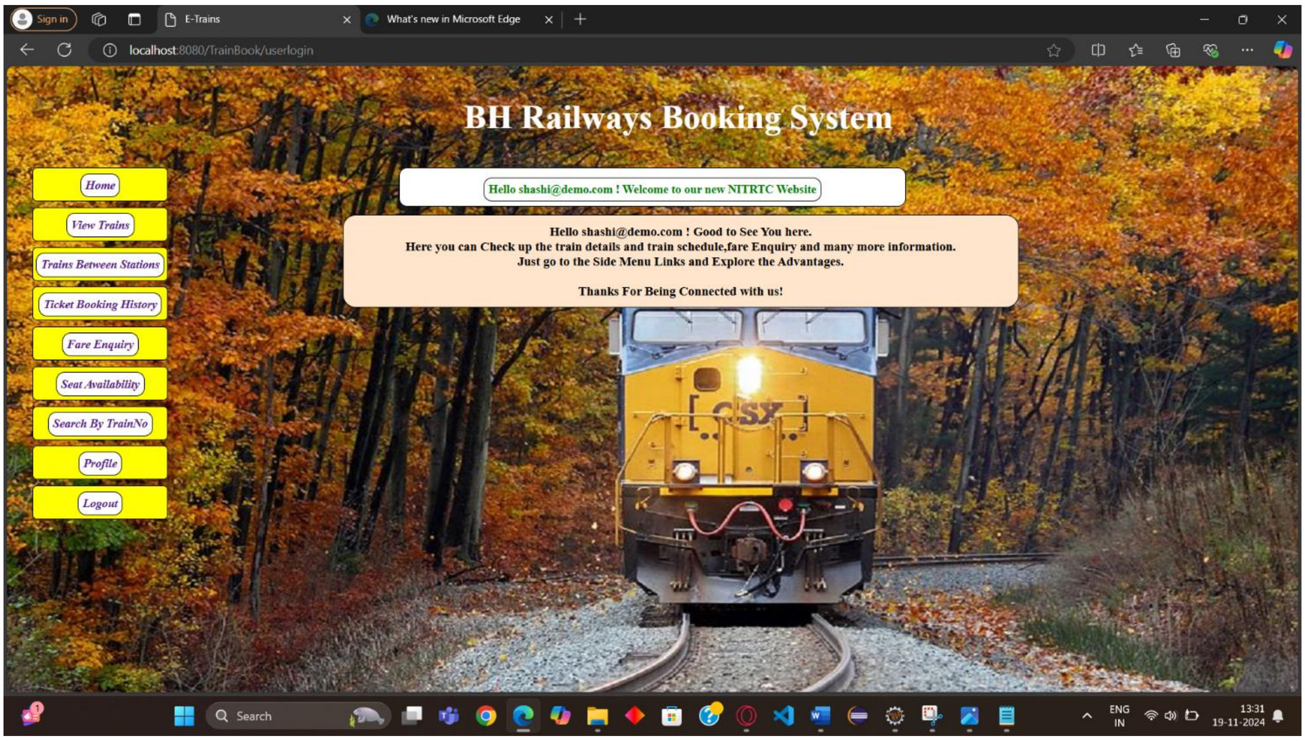
1. **Register:**
   - Admin can register by providing a username and password for authentication.
2. **Login:**
   - Admin can securely log in using their registered username and password.
3. **After Login Functionalities:**
   - **Add Train:**
     - Admin can add details about new trains, including train name, train number, route, departure and arrival times, and available seat classes.
   - **View Train:**
     - Admin can view train details such as train name, number, schedule, and routes.
     - Admin can also update train information, including timings, routes, and seat availability.
   - **Delete Train:**
     - Admin can remove trains from the system, deleting their details if the train is no longer operational.
   - **Add Reservation:**
     - Admin can create reservations for users, specifying train details, date, time, seat class, and number of seats.
   - **Update Reservation:**
     - Admin can modify reservation details, such as the date, time, seat class, or number of seats reserved.
   - **Remove Admin:**
     - Admin can delete other admin accounts or their own if necessary, ensuring proper control and management of the system.

4.Implementation: 4.1 Design:

User Page:



Home page:

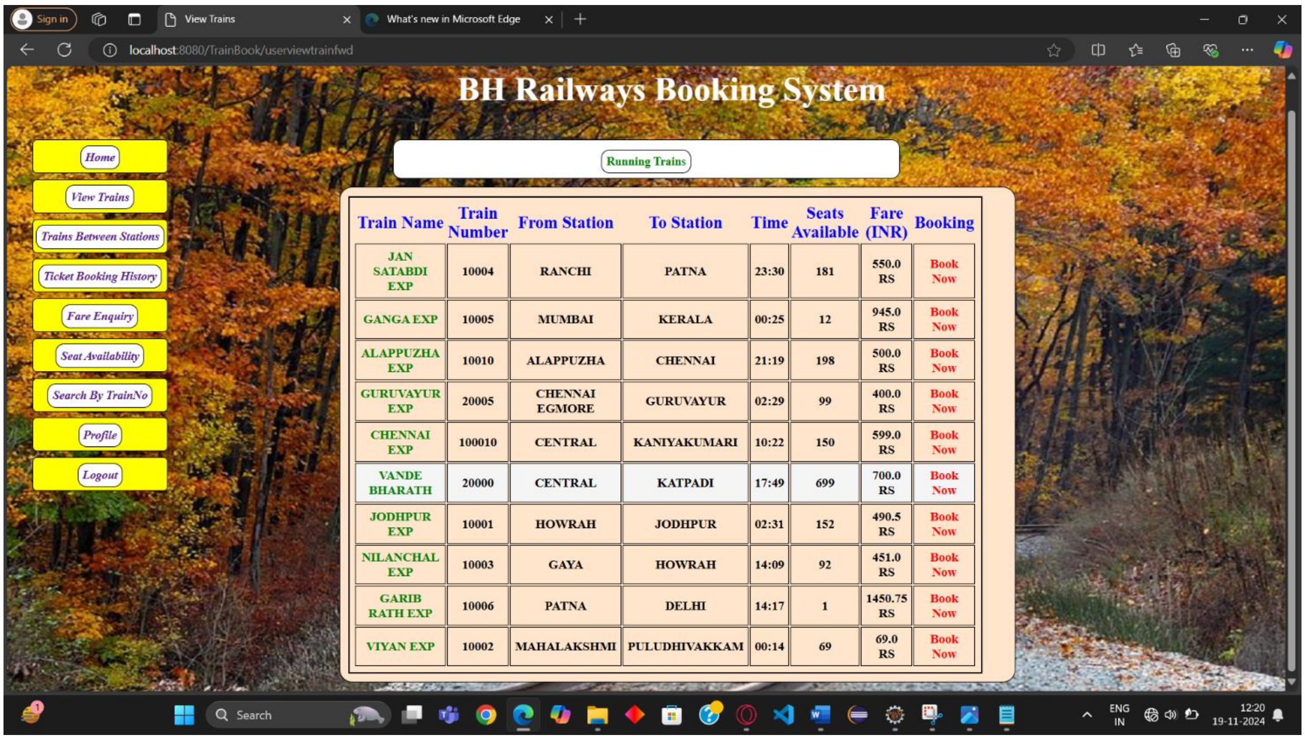


Hello shashi@demo.com ! Welcome to our new NITRTC Website

Hello shashi@demo.com ! Good to See You here.
Here you can Check up the train details and train schedule,fare Enquiry and many more information.
Just go to the Side Menu Links and Explore the Advantages.

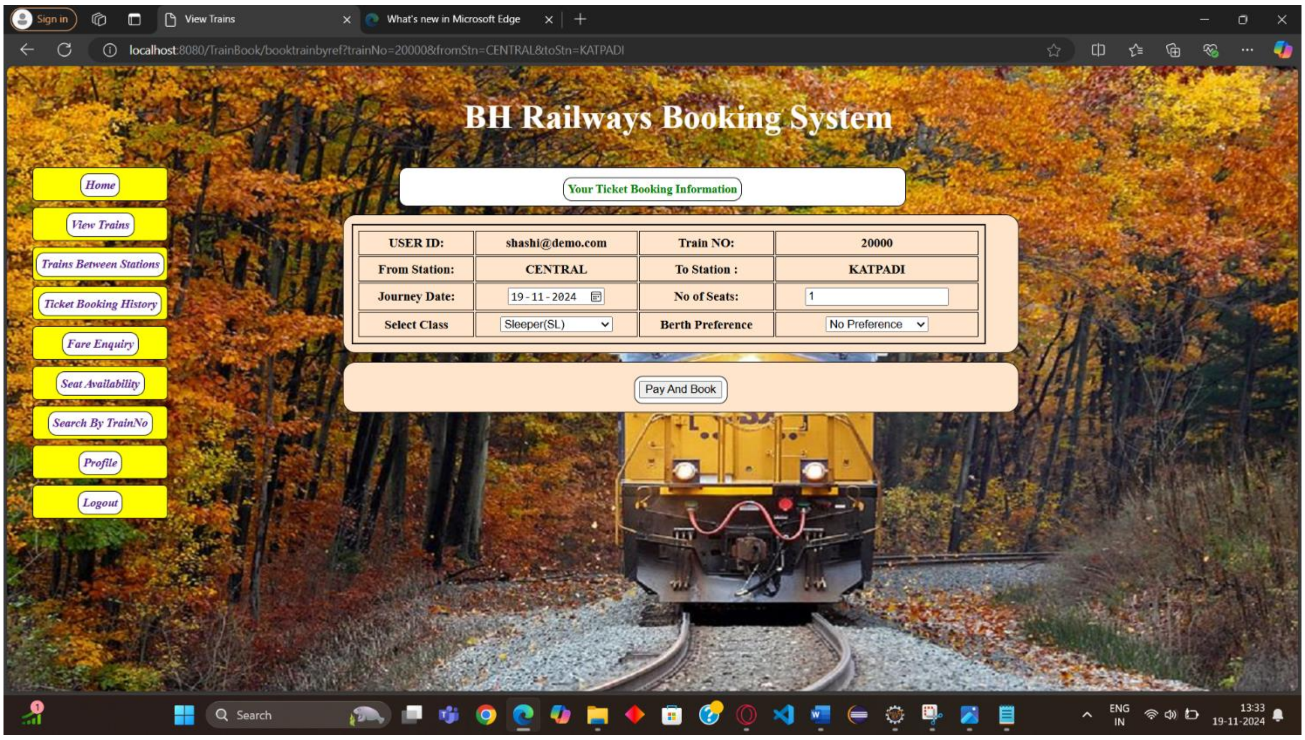
Thanks For Being Connected with us!

Train Selection:



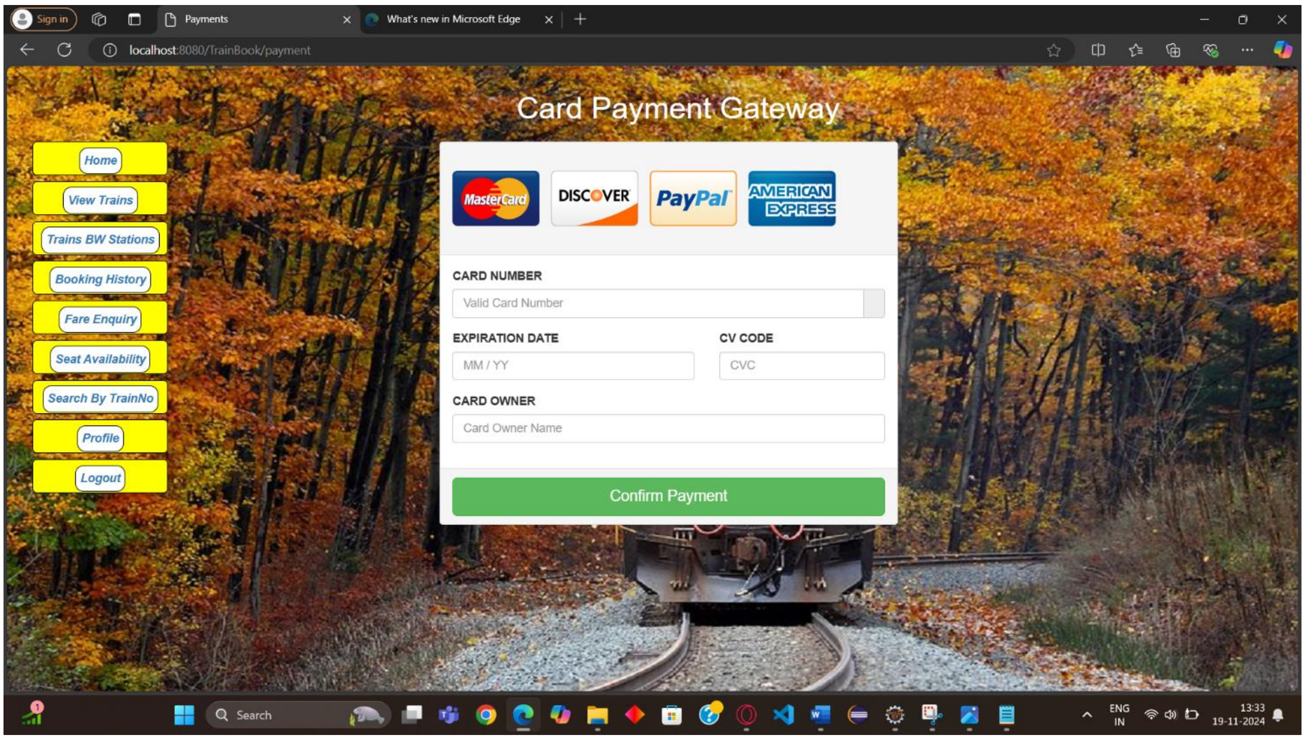
Train Name	Train Number	From Station	To Station	Time	Seats Available	Fare (INR)	Booking
JAN SATABDI EXP	10004	RANCHI	PATNA	23:30	181	550.0 RS	Book Now
GANGA EXP	10005	MUMBAI	KERALA	00:25	12	945.0 RS	Book Now
ALAPPUZHA EXP	10010	ALAPPUZHA	CHENNAI	21:19	198	500.0 RS	Book Now
GURUVAYUR EXP	20005	CHENNAI EGMORE	GURUVAYUR	02:29	99	400.0 RS	Book Now
CHENNAI EXP	100010	CENTRAL	KANIYAKUMARI	10:22	150	599.0 RS	Book Now
VANDE BHARATH	20000	CENTRAL	KATPADI	17:49	699	700.0 RS	Book Now
JODHPUR EXP	10001	HOWRAH	JODHPUR	02:31	152	490.5 RS	Book Now
NILANCHAL EXP	10003	GAYA	HOWRAH	14:09	92	451.0 RS	Book Now
GARIB RATH EXP	10006	PATNA	DELHI	14:17	1	1450.75 RS	Book Now
VIYAN EXP	10002	MAHALAKSHMI	PULUDHIVAKKAM	00:14	69	69.0 RS	Book Now

Booking Selection:


View Trains
What's new in Microsoft Edge
localhost:8080/TrainBook/booktrainbyref?trainNo=20000&fromStn=CENTRAL&toStn=KATPADI
BH Railways Booking System
Home
View Trains
Trains Between Stations
Ticket Booking History
Fare Enquiry
Seat Availability
Search By TrainNo
Profile
Logout
Your Ticket Booking Information
Pay And Book

USER ID:	shashi@demo.com	Train NO:	20000
From Station:	CENTRAL	To Station :	KATPADI
Journey Date:	19-11-2024	No of Seats:	1
Select Class	Sleeper(SL)	Berth Preference	No Preference

Payment :



Card Payment Gateway

CARD NUMBER
Valid Card Number

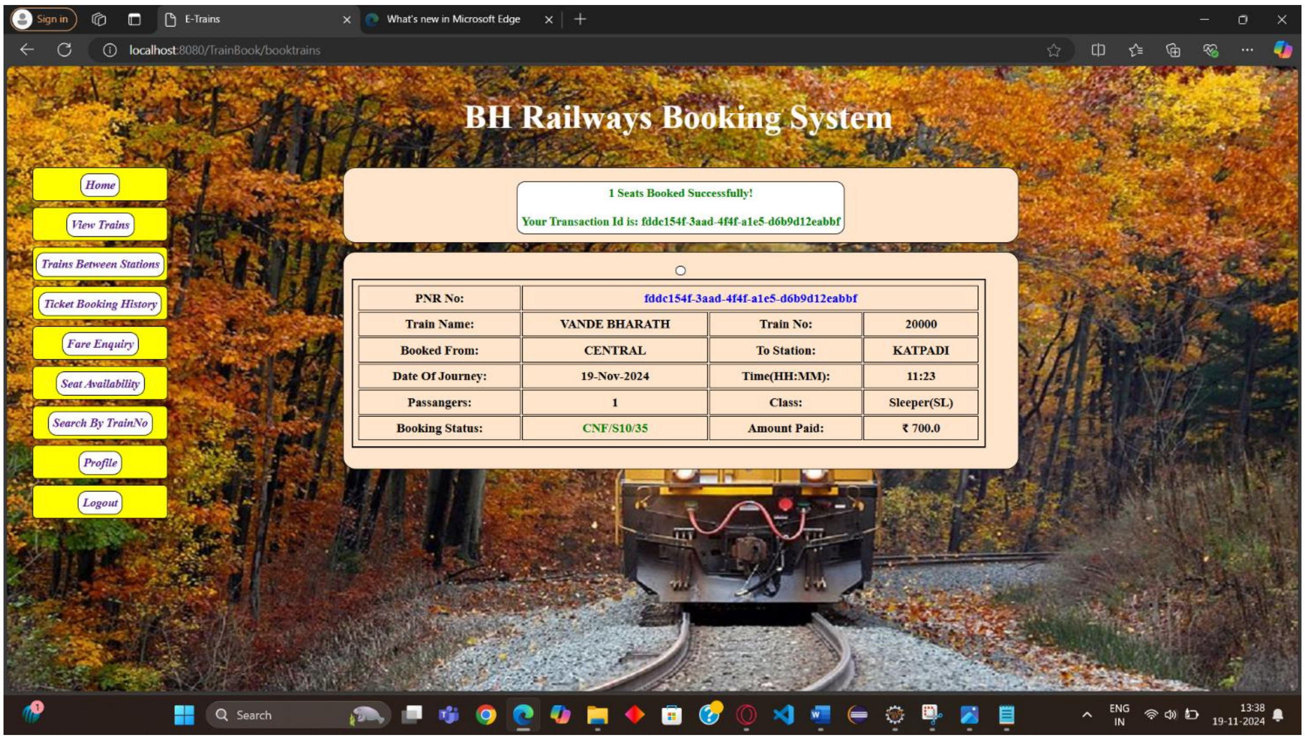
EXPIRATION DATE
MM / YY

CV CODE
CVC

CARD OWNER
Card Owner Name

Confirm Payment

Confirm status:



1 Seats Booked Successfully!

Your Transaction Id is: fddc154f-3aad-4f4f-a1e5-d6b9d12eabbf

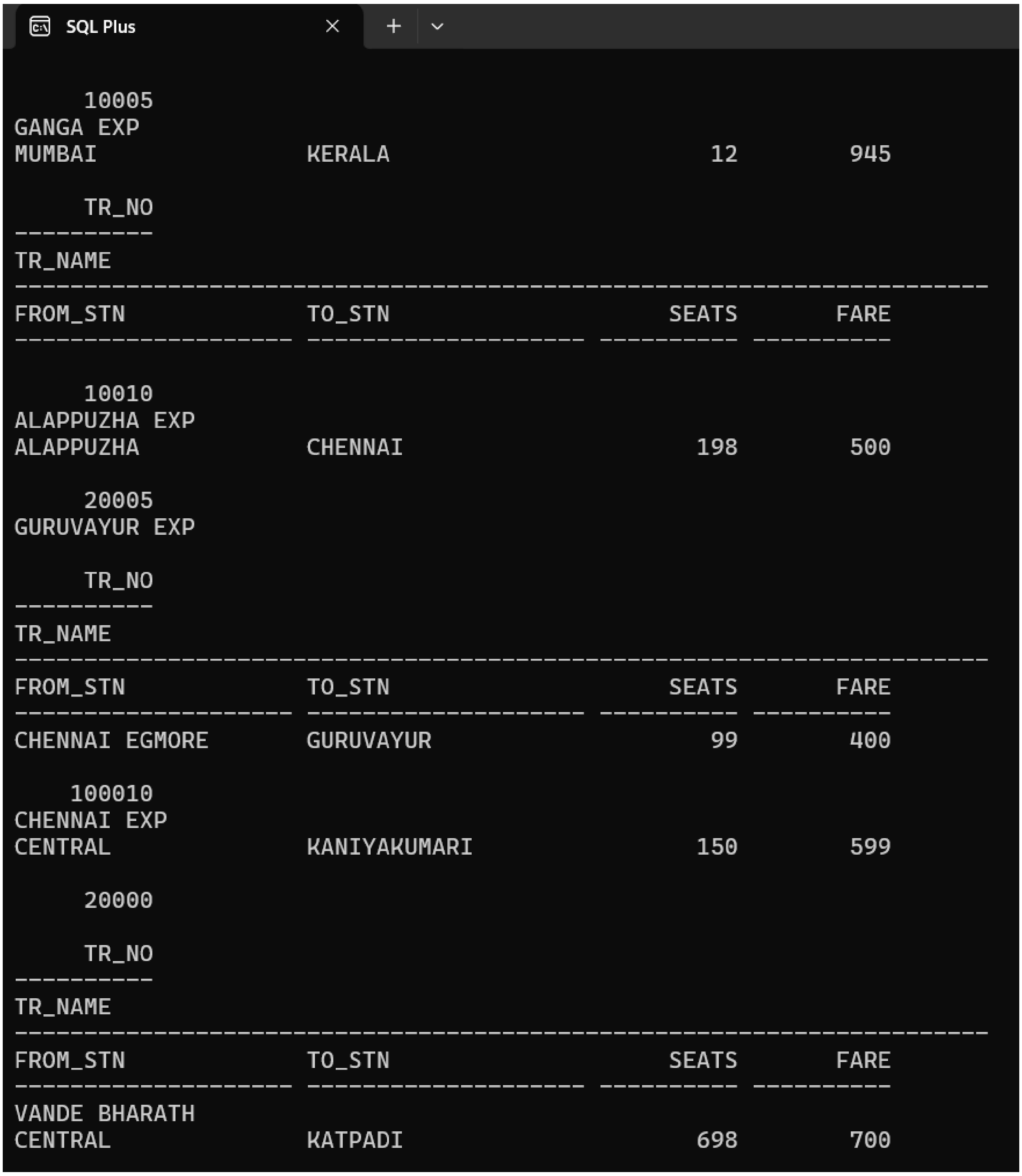
PNR No:	fddc154f-3aad-4f4f-a1e5-d6b9d12eabbf		
Train Name:	VANDE BHARATH	Train No:	20000
Booked From:	CENTRAL	To Station:	KATPADI
Date Of Journey:	19-Nov-2024	Time(HH:MM):	11:23
Passangers:	1	Class:	Sleeper(SL)
Booking Status:	CNF/S10/35	Amount Paid:	₹ 700.0

4.2 Database Design:

The data in the system has to be stored and retrieved from database. Designing the database is part of system design. Data elements and data structures to be stored have been identified at analysis stage. They are structured and put together to design the data storage and retrieval system.

A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently. The general objective is to make database access easy, quick, inexpensive and flexible for the user. Relationships are established between the data items and unnecessary data items are removed. Normalization is done to get an internal consistency of data and to have minimum redundancy and maximum stability. This ensures minimizing data storage required, minimizing chances of data inconsistencies and optimizing for updates. The SQL Plus database has been chosen for developing the relevant databases.

Train reservation database (Sql Plus) :



```
     10005
GANGA EXP
MUMBAI               KERALA                       12        945

     TR_NO
----------
TR_NAME
----------------------------------------------------------------------
FROM_STN             TO_STN                    SEATS       FARE
-------------------- -------------------- ---------- ----------

     10010
ALAPPUZHA EXP
ALAPPUZHA            CHENNAI                     198        500

     20005
GURUVAYUR EXP

     TR_NO
----------
TR_NAME
----------------------------------------------------------------------
FROM_STN             TO_STN                    SEATS       FARE
-------------------- -------------------- ---------- ----------
CHENNAI EGMORE       GURUVAYUR                    99        400

    100010
CHENNAI EXP
CENTRAL              KANIYAKUMARI                150        599

     20000

     TR_NO
----------
TR_NAME
----------------------------------------------------------------------
FROM_STN             TO_STN                    SEATS       FARE
-------------------- -------------------- ---------- ----------
VANDE BHARATH
CENTRAL              KATPADI                     698        700
```

4.3 CODE:

```
import java.sql.*;
import java.util.Scanner;

public class TrainReservationSystem {

   // JDBC URL, username, and password
   private static final String URL = "jdbc:mysql://localhost:3306/train_reservation";
   private static final String USER = "root";
   private static final String PASSWORD = "password"; //

   private static Connection connection;

   // Initialize database connection
   static {
      try {
         connection = DriverManager.getConnection(URL, USER, PASSWORD);
      } catch (SQLException e) {
         System.err.println("Database connection failed: " + e.getMessage());
         System.exit(1);
      }
   }

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.println("Welcome to Train Reservation System");

      while (true) {
         System.out.println("\nMenu:");
         System.out.println("1. Admin - Add Train");
         System.out.println("2. Admin - View Trains");
         System.out.println("3. Admin - Delete Train");
         System.out.println("4. User - Book Reservation");
         System.out.println("5. User - View Reservations");
         System.out.println("6. Exit");
         System.out.print("Enter your choice: ");

         int choice = scanner.nextInt();
         scanner.nextLine(); // Consume newline

         switch (choice) {
             case 1:
                 addTrain(scanner);
                 break;
             case 2:
                 viewTrains();
                 break;
             case 3:
                 deleteTrain(scanner);
                 break;
             case 4:
                 bookReservation(scanner);
                 break;
             case 5:
                 viewReservations();
                 break;
             case 6:
                 System.out.println("Exiting the system. Goodbye!");
                 System.exit(0);
                 break;
```

```
                default:
                    System.out.println("Invalid choice! Try again.");
            }
        }
    }

    // Method to add a new train
    private static void addTrain(Scanner scanner) {
        try {
            System.out.print("Enter train name: ");
            String trainName = scanner.nextLine();
            System.out.print("Enter source: ");
            String source = scanner.nextLine();
            System.out.print("Enter destination: ");
            String destination = scanner.nextLine();
            System.out.print("Enter departure time: ");
            String departureTime = scanner.nextLine();
            System.out.print("Enter arrival time: ");
            String arrivalTime = scanner.nextLine();
            System.out.print("Enter available seats: ");
            int seatsAvailable = scanner.nextInt();

            String query = "INSERT INTO trains (train_name, source, destination,
departure_time, arrival_time, seats_available) VALUES (?, ?, ?, ?, ?, ?)";
            PreparedStatement preparedStatement = connection.prepareStatement(query);
            preparedStatement.setString(1, trainName);
            preparedStatement.setString(2, source);
            preparedStatement.setString(3, destination);
            preparedStatement.setString(4, departureTime);
            preparedStatement.setString(5, arrivalTime);
            preparedStatement.setInt(6, seatsAvailable);
            int rows = preparedStatement.executeUpdate();
            if (rows > 0) {
                System.out.println("Train added successfully!");
            }
        } catch (SQLException e) {
            System.err.println("Error adding train: " + e.getMessage());
        }
    }

    // Method to view all trains
    private static void viewTrains() {
        try {
            String query = "SELECT * FROM trains";
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery(query);

            System.out.println("\nTrain Details:");
            while (resultSet.next()) {
                System.out.printf("ID: %d | Name: %s | Source: %s | Destination: %s |
Departure: %s | Arrival: %s | Seats: %d\n",
                        resultSet.getInt("train_id"),
                        resultSet.getString("train_name"),
                        resultSet.getString("source"),
                        resultSet.getString("destination"),
                        resultSet.getString("departure_time"),
                        resultSet.getString("arrival_time"),
                        resultSet.getInt("seats_available"));
            }
        } catch (SQLException e) {
            System.err.println("Error viewing trains: " + e.getMessage());
        }
    }

    // Method to delete a train
    private static void deleteTrain(Scanner scanner) {
        try {
            System.out.print("Enter train ID to delete: ");
            int trainId = scanner.nextInt();

            String query = "DELETE FROM trains WHERE train_id = ?";
            PreparedStatement preparedStatement = connection.prepareStatement(query);
            preparedStatement.setInt(1, trainId);

            int rows = preparedStatement.executeUpdate();
            if (rows > 0) {
                System.out.println("Train deleted successfully!");
            } else {
                System.out.println("Train not found!");
            }
        } catch (SQLException e) {
            System.err.println("Error deleting train: " + e.getMessage());
        }
    }

    // Method to book a reservation
    private static void bookReservation(Scanner scanner) {
        try {
            System.out.print("Enter your name: ");
            String userName = scanner.nextLine();
            System.out.print("Enter train ID: ");
```

```
            int trainId = scanner.nextInt();
            System.out.print("Enter number of seats to book: ");
            int seats = scanner.nextInt();

            String checkSeatsQuery = "SELECT seats_available FROM trains WHERE
train_id = ?";
            PreparedStatement checkSeatsStmt =
connection.prepareStatement(checkSeatsQuery);
            checkSeatsStmt.setInt(1, trainId);
            ResultSet resultSet = checkSeatsStmt.executeQuery();

            if (resultSet.next()) {
                int availableSeats = resultSet.getInt("seats_available");
                if (seats <= availableSeats) {
                    String bookQuery = "INSERT INTO reservations (user_name, train_id,
seats_booked) VALUES (?, ?, ?)";
                    PreparedStatement bookStmt =
connection.prepareStatement(bookQuery);
                    bookStmt.setString(1, userName);
                    bookStmt.setInt(2, trainId);
                    bookStmt.setInt(3, seats);
                    bookStmt.executeUpdate();

                    String updateSeatsQuery = "UPDATE trains SET seats_available =
seats_available - ? WHERE train_id = ?";
                    PreparedStatement updateStmt =
connection.prepareStatement(updateSeatsQuery);
                    updateStmt.setInt(1, seats);
                    updateStmt.setInt(2, trainId);
                    updateStmt.executeUpdate();

                    System.out.println("Reservation successful!");
                } else {
                    System.out.println("Not enough seats available!");
                }
            } else {
                System.out.println("Train not found!");
            }
        } catch (SQLException e) {
            System.err.println("Error booking reservation: " + e.getMessage());
        }
    }

    // Method to view all reservations
    private static void viewReservations() {
        try {
            String query = "SELECT r.reservation_id, r.user_name, t.train_name,
r.seats_booked " +
                           "FROM reservations r JOIN trains t ON r.train_id = t.train_id";
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery(query);

            System.out.println("\nReservations:");
            while (resultSet.next()) {
                System.out.printf("Reservation ID: %d | User: %s | Train: %s | Seats
Booked: %d\n",
                        resultSet.getInt("reservation_id"),
                        resultSet.getString("user_name"),
                        resultSet.getString("train_name"),
                        resultSet.getInt("seats_booked"));
            }
        } catch (SQLException e) {
            System.err.println("Error viewing reservations: " + e.getMessage());
        }
    }
}
```

Conclusion:

The Train Reservation System project, developed under expert guidance, exemplifies a comprehensive approach to system design and implementation. With user-friendly functionalities such as adding/viewing trains, managing reservations, and checking seat availability, the system ensures a seamless booking experience for users. Robust security measures, particularly in administrative functions like "Remove Admin," emphasize a strong commitment to data integrity and system security. This project stands as a well-structured solution, effectively addressing current requirements while offering scalability and adaptability for future enhancements.
4o

Reference links:

1. https://www.geeksforgeeks.org/online-railway-ticket-reservation-system/