SRI SIVASUBRAMANIYA NADAR COLLEGE OF ENGINEERING

(AN AUTONOMOUS INSTITUTION,
AFFILIATED TO ANNA UNIVERSITY)

Rajiv Gandhi Salai (OMR), Kalavakkam - 603 110.

LABORATORY RECORD

NAME : NITHISH BHARATHWAJ.K.R

Reg. No. : 195001072

Dept. : CSE Sem. : 6 Sec. : B

SSN

SRI SIVASUBRAMANIYA NADAR COLLEGE OF ENGINEERING, CHENNAI

(AN AUTONOMOUS INSTITUTION, AFFILIATED TO ANNA UNIVERSITY)

BONAFIDE CERTIFICATE

Certified that this is the bonafide record of the practical work done in the

UCS1617 - MINI PROJECT ..... **Laboratory by**

Name .....NITHISH BHARATHWAJ.K.R.....

Register Number .....195001072.....

Semester .....SIX.....

Branch .....CSE.....

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam.

During the Academic year .....2021-22.....

9/6/22

Faculty **Head of the Department**

Submitted for the.....Practical Examination held at SSNCE on.....

Internal Examiner **External Examiner**

INDEX

Name : Nithish Bharathwaj. K.R. Reg. No. 19500107 2

Sem : VI Sec : B

Ex. No.	Date of Expt.	Title of the Experiment	Page No.	Signature of the Faculty	Remarks
1	10/3/22	Problem Statement	2		
2	17/3/22	Software Requirements Specifications	5		
3	24/3/22	Usecase Diagram	17		
4	31/3/22	Domain Model & Class Diagram	28		
5	7/4/22	Interaction Diagram	37		
6	21/4/22	State Machine & Activity Diagram	42		
7	23/4/22	Implementation Demo	47		
8	26/5/22	Test Cases & Test Plans	106		
9	2/6/22	Applying Design Patterns	109		
10	9/6/22	Testing after Refinement - Refactoring	111		

Completed 9/6/22

UCS1617 - MINI PROJECT

ONLINE RAILWAYS RESERVATION SYSTEM

RANJANI A - 195001089
POOJA M - 195001076
SANJANA A - 195001097
NITHISH BHARATHWAJ K R - 195001072

Exp. No: 1
Date: 10/3/22

PROBLEM STATEMENT

Our project is carried out to develop a website for online railway reservation system. This system has various options like reservation, cancellation, refunding and to view details about available trains and seats. The reservation option enables a person to reserve for a ticket from their current environment.

Initially, the user has to create a new account with username and password. If the username already exists it shows an error that the username already exists. Then the user has to use a new username and password with the required details. The existing user can directly login to their account with their username and password. After this, the details have been stored in the database for the future use. After the login process the user should enter the travel data with source and destination place. Based on the travel date, the types of trains and availability of different types of seats will be shown with advanced or tatkal booking. The user can view the seats based upon their wish with price. For this above process the administrator side should contain all types of trains, train routes, types of seats and their respective price. Based upon the user's requirements and preferences seats will be allocated. Suppose if the user's requirement is not present, it will show the available seats.

After the confirmation of the seats, it goes to the passenger details page. The passenger's details are filled in that page. After this the payment procedure will be done and the ticket will be confirmed. The users can view the status of the ticket in booked ticket in booked details tab. The users can also cancel the booked tickets and the refunded amount will be sent to the users account in which they have booked the tickets. The user can also view the history of refund details. The user will get the notification once the ticket is confirmed with the seat number and the compartment number.

On the Administrator side, each admin has a username and password to log in to the system. After logging in, all information about the current day's train schedule will be displayed, which would be maintained in a database. The provisions to update this schedule will also be available. Then, the customers' reservations along with their preferences will be listed. For advance booking scheme, admin would check the available seats in the suitable trains and fill them accordingly by putting maximum effort to satisfy the user's preferences as much as possible. This involves updating the respective databases. For tatkal booking scheme, a certain amount of seats, which will not be opened before one or two days before the train's departure, will be available for customers with higher prices compared to actual cost. These seats will be get filled fast. The reservation procedure is same as before. Admin has to monitor the boarding point and destination point of the customer through the databases, so that if the destination is an intermediate station, the seat has to be made open for reservation from that point.

Admin has to contact the bank agent to process the payment. The bank agent then confirms the payment success to admin. Admin, now completes the reservation process and finally the ticket is confirmed and the customer is notified at once. If it is

not possible to book the tickets with the stated needs, the customer gets the rejection message as well.

If the customer opts for refund, admin has to track down the customer's booked ticket details and make it open for others. Admin contacts the bank agent and debits a part of the paid amount into the customers' account. This refund process is also recorded in databases so that the users can refer them in future.

BENEFITS

- Convenient – You can book or cancel your tickets sitting in the comfort of your home or office.
- Saves time and effort – You can save the time needed to travel to the railway reservation office and waiting in the queue for your turn.
- Towards a greener planet – Instead of printing your ticket you can also choose to travel with the SMS or soft copy of your booked ticket in your laptop or even on your mobile.
- Easy payment - While booking counter ticket you had to carry cash and while booking E-ticket you are paying through online directly from bank which makes work easier for you.
- Keeping track of reservations – Sometimes user reserve tickets months ahead, especially when they have a holiday planned out during peak seasons. Manually managing such reservations can lead to errors. Fortunately, online reservation system can solve your booking problems.

Exp. No.: 2
Date: 17/3/22

SOFTWARE REQUIREMENTS SPECIFICATION

Table of Contents

1 Introduction

Our project is carried out to develop a website for online railway reservation system. This system has various options like reservation, cancellation, refunding and to view details about available trains and seats. The reservation options enable a person to reserve for a ticket from their current environment.

1.1 Purpose

The purpose of this source is to describe the railway reservation system which provides the train timing details, reservation and cancellation on various types of reservation namely, confirm reservation for confirm seats, reservation against cancellation, waiting list reservation and tatkal reservation. The purpose to SRS document is to describe the external behaviour of the online railway reservation system. Requirement specification defines and describes the operations, interfaces, performance and quality assurance requirements of the online railway reservation system the document also describes the non-functional requirements such as the user interfaces. It also describes the design constraints that are to be considered when the system is to be designed and other factors necessary to provide a complete and comprehensive description of the requirements for the system or a portion of the system.

1.2 Scope

“Railway Reservation System” is an attempt to simulate the basic concepts of an online Reservation system. The system enables to perform the following functions:

- Login credential
- Search for Train
- Creating reservation
- Payment
- Cancel reservation
- View reservation status
- Generating reports
- Update train schedule
- Update reservation details

1.3 Definition, Acronyms and Abbreviations

- SRS – Software Requirements Specification
- ORRS – Online Railway Reservation System
- MTBF – Mean time between failures
- MTTR – Mean time to repair
- SSL – Secure Socket Layer
- LAN – Local Area Network

1.4 References

The following websites were used as references:

- www.irctc.co.in
- www.makemytrip.com

1.5 Overview

The SRS will provide a detailed description of the Railway Reservation System. This document will provide the outline of the requirements, overview of the characteristics and constraints of the system.

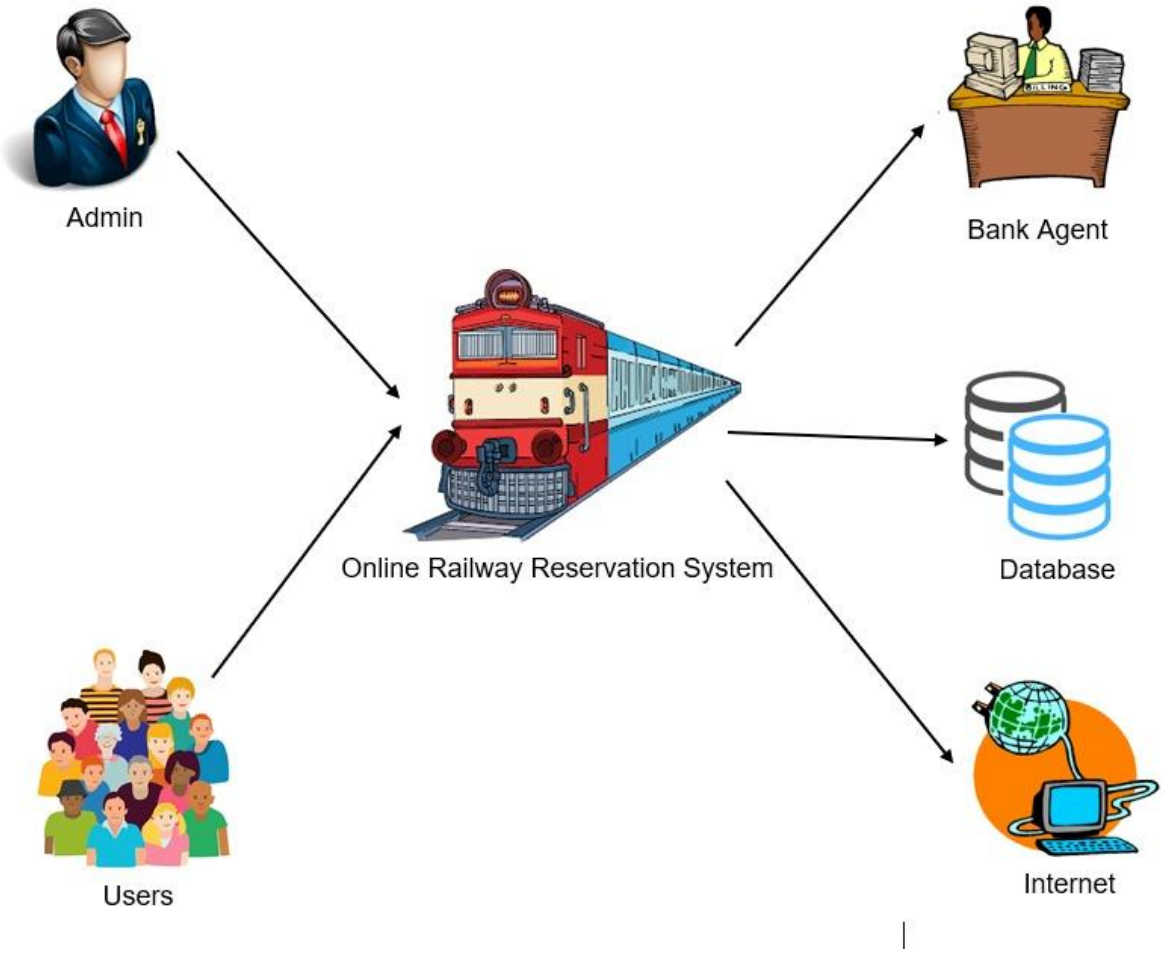
1.5.1 Section 2: This section of the SRS will provide the general factors that affect the product and its requirements. It provides the background for those requirements. The items such as product perspective, product function, user characteristics, constraints, assumptions and dependencies and requirements subsets are described in this section.

1.5.2 Section 3: This section of SRS contains all the software requirements mentioned in section 2 in detail sufficient to enable designers to design the system to satisfy the requirements and testers to test if the system satisfies those requirements.

2 Overall Description

- **Product Perspective**

The Online Railway Reservation System is a package to be used by all people in the world to improve the efficiency of Administrator, Cashier and Users in Railways. The system enables us to maintain the railway train details like their timings, number of seat available and cancelling the tickets. The existing system is highly manual involving a lot of paper work and calculation and therefore may be erroneous. This has led to inconsistency and inaccuracy in the maintenance of data. The data, which is stored on the paper only, may be lost, stolen or destroyed due to natural forces like fire and water. The existing system is sluggish and consumes a lot of time causing inconvenience to customers and the staff. Due to manual nature, it is difficult to update, delete, add or view the data. The number of passengers has drastically increased and so maintaining and retrieving detailed record of passenger is extremely difficult. A railway has many offices around the world, an absence of a link between these offices leads to the lack of coordination and communication. Hence the railways reservation system is proposed with the new developments like computerization of the reservation system which will reduce a lot of paperwork. The machine performs all calculations. Hence chances of error are nil. The passenger, reservation, cancellation list can easily be retrieved and any required addition, deletion or updating can be performed. The system provides for user-ID validation, hence unauthorized access is prevented.



Overview of the proposed system

- ## Product Functions

The Online Railway Reservation System provides real time information about the number of seats available in the trains and the user information. The Product functions are more or less the same as described in the product perspective. The functions of the system include the system providing different type of services based on the type of users.

The scope of this project encompasses:

- Search - This function allows the user to search for train that are available between the two travel cities, namely the "Departure city" and "Arrival city" as desired by the traveller. The system initially prompts

the agent for the departure and arrival city, the date of departure, preferred time slot and the number of passengers. It then displays a list of trains in that route.

- Selection - This function allows a particular train to be selected from the displayed list. All the details of the train are shown, namely:
    1. Train Number
    2. Date, time and place of departure
    3. Date, time and place of arrival
    4. Travel Duration
    5. Fare per head
    6. Number of stoppages
- Review - If the seats are available, then the software prompts for the booking of train. The train information is shown. The total fare including taxes is shown and the details are reviewed.
- Traveller Information - It asks for the details of all the passengers supposed to travel including name, address, phone number and e-mail id.
- Payment - It asks the user to enter the various credit card details for making the reservation.
- Cancellation - The system also allows the passenger to cancel an existing reservation. This function registers the information regarding a passenger who has requested for a cancellation of his/her ticket.

User Characteristics

- Educational Level - At least user of the system should be comfortable with English language.
- Technical Expertise - User should be comfortable using general purpose applications on the computer system.

The administrators of the system to have more knowledge of the internals of the system and should be able to rectify the small problems that may arise due to disk crashes, power failures and other catastrophes to maintain the system. The proper user interface, user's manual, online help and the guide to install and maintain the system must be sufficient to educate the users on how to use the system without any problems.

Constraints

The information of all the users must be stored in a database that is accessible by the Online Railway Reservation System. The Online Railway Reservation System will be available for all 24 hours a day. The users access the system from any computer that has Internet browsing capabilities and an Internet connection. The

billing system is connected to the system and the database used by the billing system must be compatible with the interface of the Online Railway Reservation System. The users must have their correct usernames and passwords to enter into the Online Railway Reservation System.

- **Assumptions and dependencies**

Customers will be having a valid user name and password to access the software. The software needs passengers to have complete knowledge of railways reservation system. Software is dependent on access to internet. The users must have sufficient knowledge of computers. The users must know English, as the user interface will be provided in English.

3 Specific Requirements

3.1 Functionality

3.1.1 Logon Capabilities

The system allows the customers and administrators to login into it.

3.1.2 Getting Input

The system enables the customers to enter their requirements and preferences through a form, which updates the back end database.

3.1.3 Notifications

After confirming the ticket reservation, the user gets a notification fro the system about the confirmation of their ticket

3.2 Usability

3.2.1 User Accessibility

The system allows the users to access the system from the Internet using HTML or its derivative technologies.

3.2.2 User Friendliness

The system is such that it stands up to the user's expectations. The system is easy to learn and understand. A native user can also use the system effectively, without any difficulties.

3.3 Performance

3.3.1 Response Time

The response of all the operations is good. This has been made possible by careful programming.

3.3.2 Error Handling

The system is able to respond to user errors and undesired situations, without halting or crashing.

3.3.3 Safety and Robustness

The system is able to avoid or tackle disastrous actions. In other words, it should be fool proof. The system safeguards against undesired events, without human intervention.

3.3.4 Portability

The software is not architecture specific. It should be easily transferable to other platforms if needed.

3.4 Reliability

3.4.1 Separate Components' Reliability

The reliability of the whole project depends on the reliability of separate components. The main pillar of reliability of the system is the backup of the database which is continuously maintained and updated to reflect the most recent changes.

3.4.2 Computer Stability

The system will be functioning inside a computer. Thus, the overall stability of the system depends on the stability of the computer and its underlying operating system.

3.4.3 Availability

The system is available 100% for the user and is used 24 hrs a day and 365 days a year. The system shall be operational 24 hours a day and 7 days a week.

3.4.4 Mean Time Between Failures (MTBF)

The system will be developed in such a way that it may fail once in a year.

3.4.5 Mean Time to Repair (MTTR)

Even if the system fails, the system will be recovered back up within an hour or less.

3.4.6 Accuracy

The accuracy of the system is limited by the accuracy of the speed at which the admins and customers use the system.

3.4.7 Access Reliability

The system shall provide 100% access reliability.

3.5 Supportability

3.5.1 Maintenance

The maintenance of the system shall be done as per the maintenance contract.

3.5.2 Internet Protocols

The system shall comply with the TCP/IP protocol standards and shall be designed accordingly.

3.5.3 Standards

The coding standards and naming conventions will be as per the American standards.

3.6 Security

The system uses SSL(Secure Socket Layer) in all transactions that include any confidential customer information. The system will automatically log out all customers after a period of inactivity. The system should not leave any cookies on the customer's computer containing the user's passwords. The system's back-end servers shall only be accessible to authenticated management.

3.7 Online User Documentation and Help System Requirements

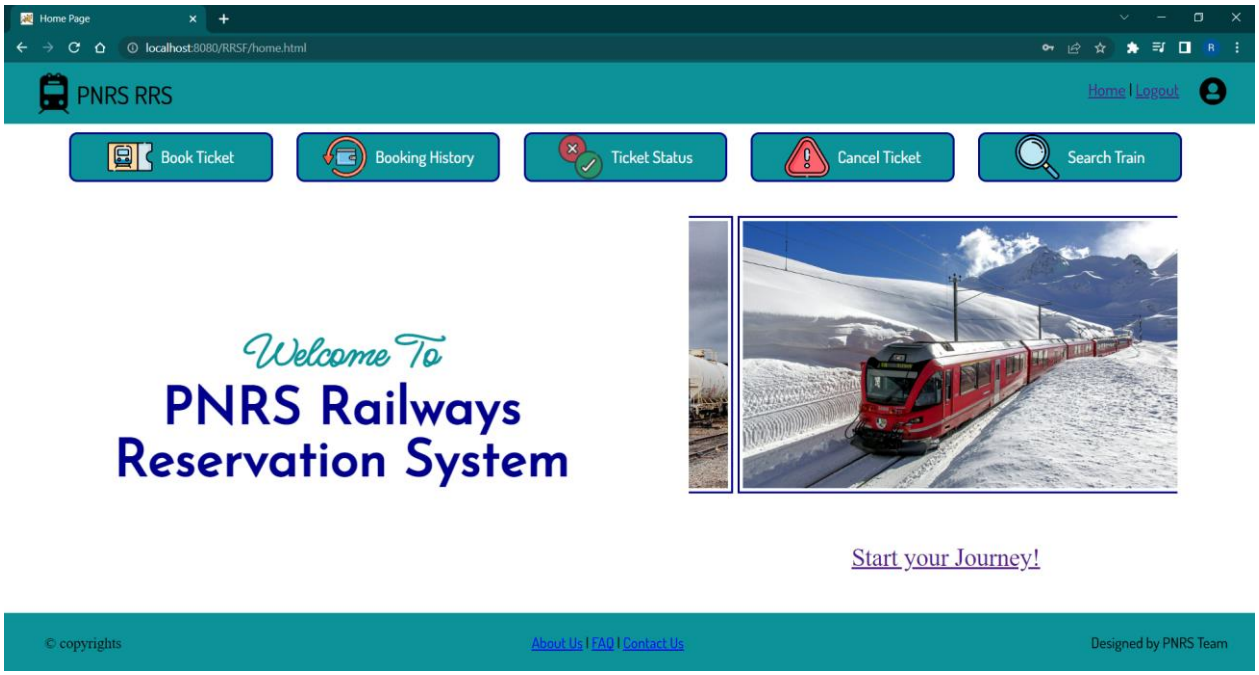
Online help is provided for each of the feature available with the Online Railways Reservation System. All the applications provide an on-line help system to assist the user. The nature of these systems is unique to application development as they combine aspects of programming (hyperlinks, etc) with aspects of technical writing (organization, presentation). Online help is provided for each and every feature provided by the system. The code and supporting modules of the system will be well documented and easy to understand. The User Manual describes the use of the system to Customers. It describes the use of the system on mobile systems. The user manual should be available as a hard copy and also as online help. An installation document will be provided that includes the installation instructions and configuration guidelines, which is important to a full solution offering. Also, a Read Me file is typically included as a standard component. The Read Me includes a "What's New with This Release" section, and a discussion of compatibility issues with earlier releases. Most users also appreciate documentation defining any known bugs and workarounds in the Read Me file.

3.8 Interfaces

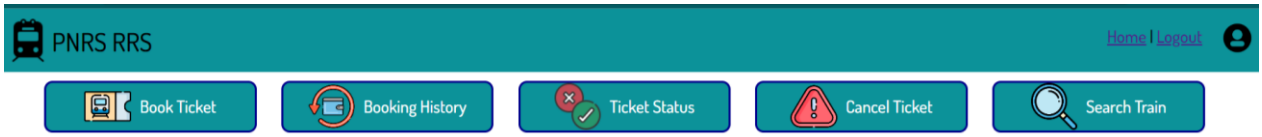
3.8.1 User Interface

The system will make use of the existing Web Browsers such as Google Chrome or Mozilla Firefox. The user interface of the system shall be designed as shown in the user-interface prototypes.

Homepage Prototype



Dashboard Prototype



3.8.2 Hardware Interfaces

The existing Local Area Network (LAN) will be used for collecting data from the users.

3.8.3 Software Interfaces

A firewall will be used with the server to prevent unauthorized access to the system.

3.8.4 Communications Interfaces

The Online Railways Reservation System will be connected to the World Wide Web.

4 Supporting Information

The use-case storyboards or the user-interface prototypes are not available. The appendices are not to be considered as part of the requirements.

Exp. No: 3
Date: 24/3/22

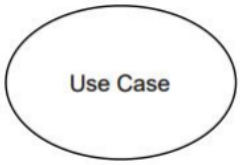
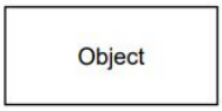
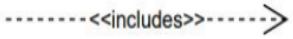
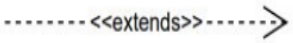
USE CASE MODEL

AIM

To create a UML use case model for the problem domain that includes:

- Researching the business domain.
- Documenting UML notations.
- Identifying use cases.
- Identifying different types of actors.
- Identifying scenarios.
- Identifying sub-functions.
- Determining relationship between use cases.
- Determining relationship between use cases and actors.
- Creating use case diagrams adhering to UML standards for the identified scenarios.
- Expanding identified sub-functions by creating use case diagrams adhering to UML standards.
- Providing a fully dressed use case description for each scenario.
- Documenting observations on the use case model of the system.

NOTATION

Figure	Notation	Explanation
	Use case	Represents various functionality of the system.
	Actor	Represents the various types of users in the system.
	Object	Represents an object associated with the system.
	Relationship	Represents “includes” relationship between use cases.
	Relationship	Represents “extends” relationship between use cases.
	Relationship	Represents generalization relationship between use cases.
	Relationship	Represents association between actors and use cases.

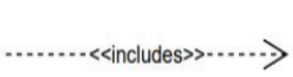
IDENTIFICATION OF ACTORS

- Administrator
- Passenger
- Bank Agent

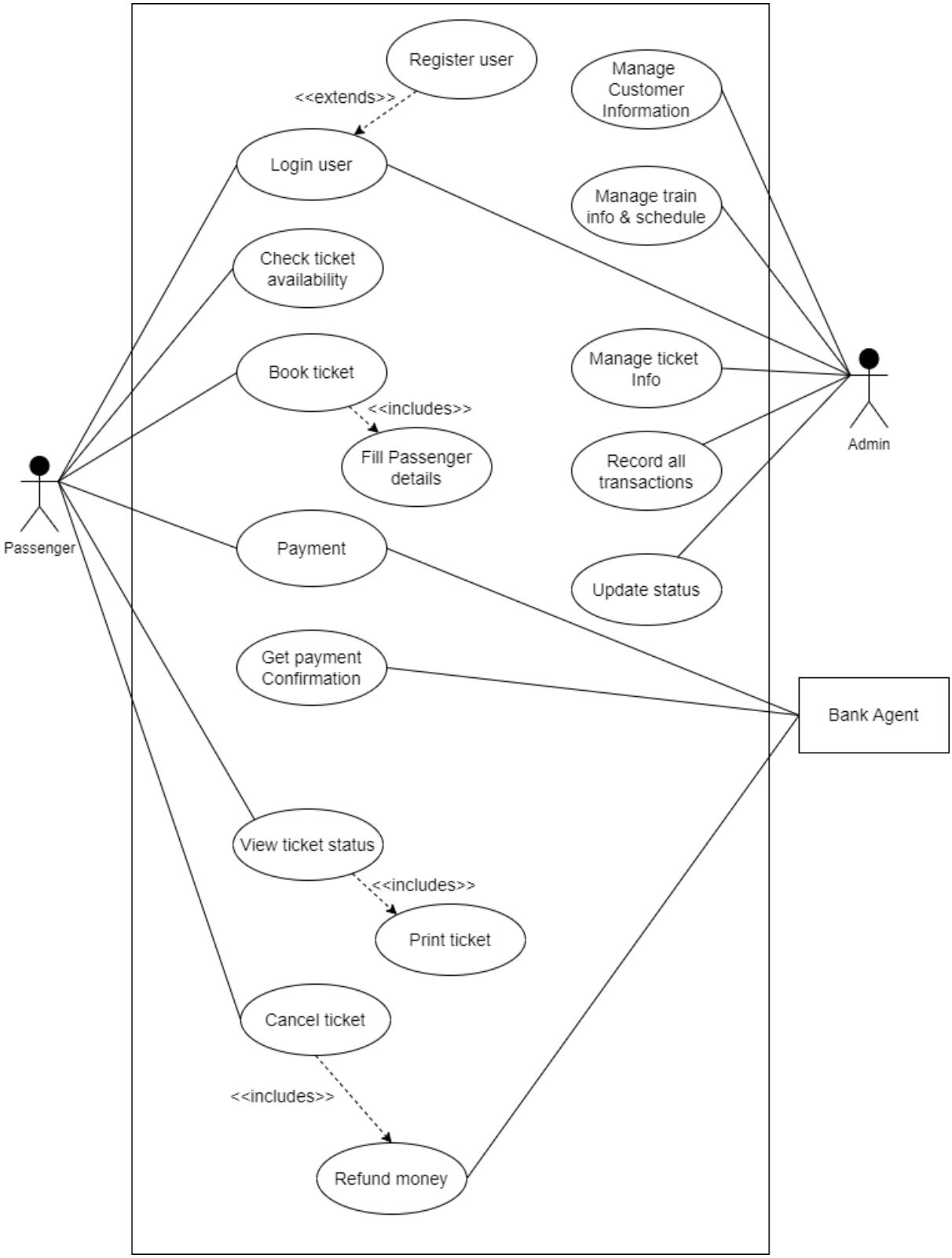
IDENTIFICATION OF SCENARIOS

Scenario	Type	Description
Main Success	User Goal Scenario	Complete model inclusive of all the system functionalities and behavior.
Unavailability	Alternate Scenario	Describes what would happen when the searched train is not available.
Login Failure	Alternate scenario	Describes what happens when user login fails.
Payment	Sub-function	Flow of actions when payment is made.
Cancellation	Sun-function	Cancellation of booked ticket.

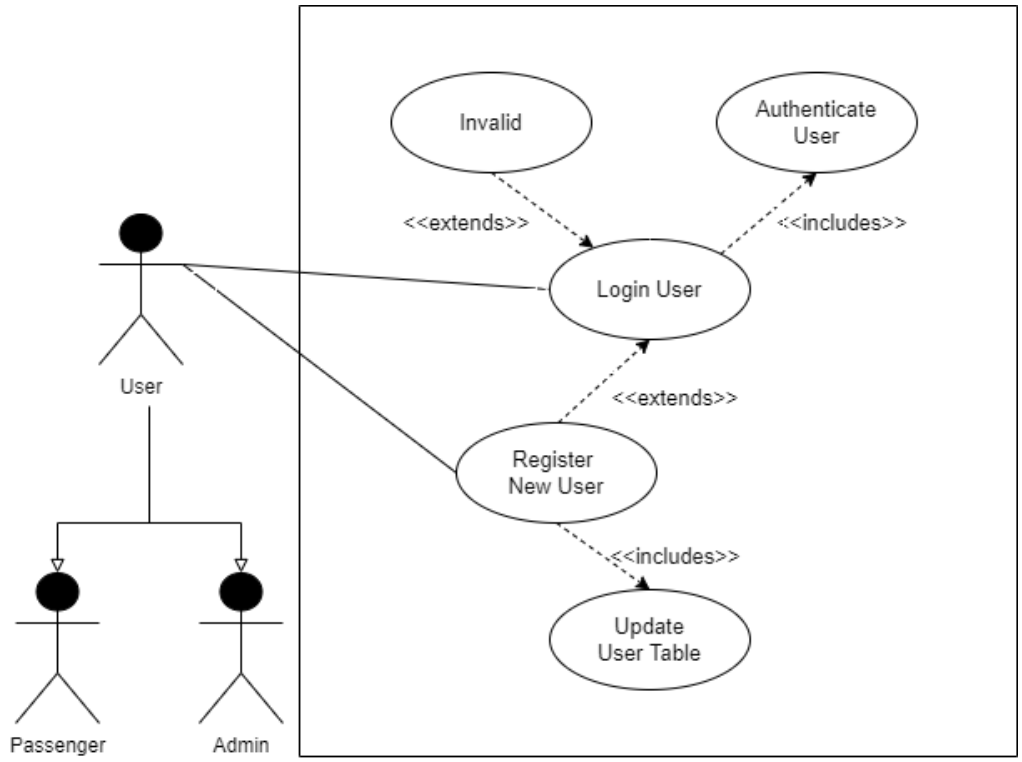
RELATING USE CASES

Figure	Explanation
	Represents “includes” relationship between use cases.
	Represents “extends” relationship between use cases.
	Represents generalization relationship between use cases.
	Represents association between actors and use cases.

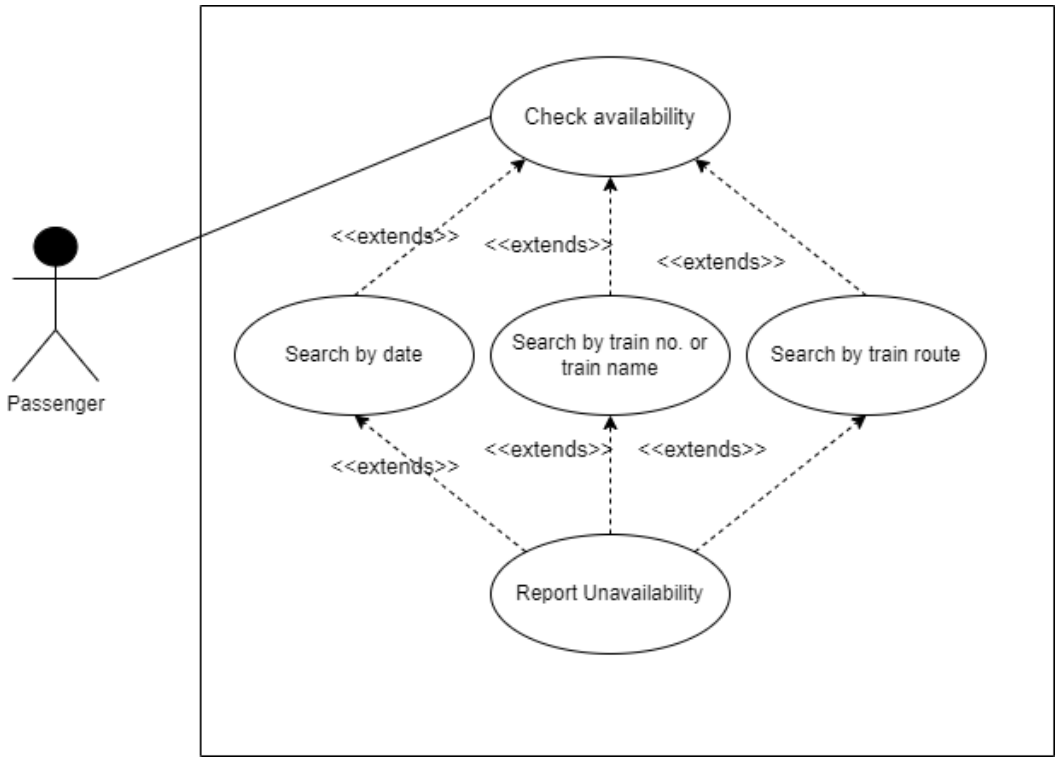
User Goal: Main Success Scenario



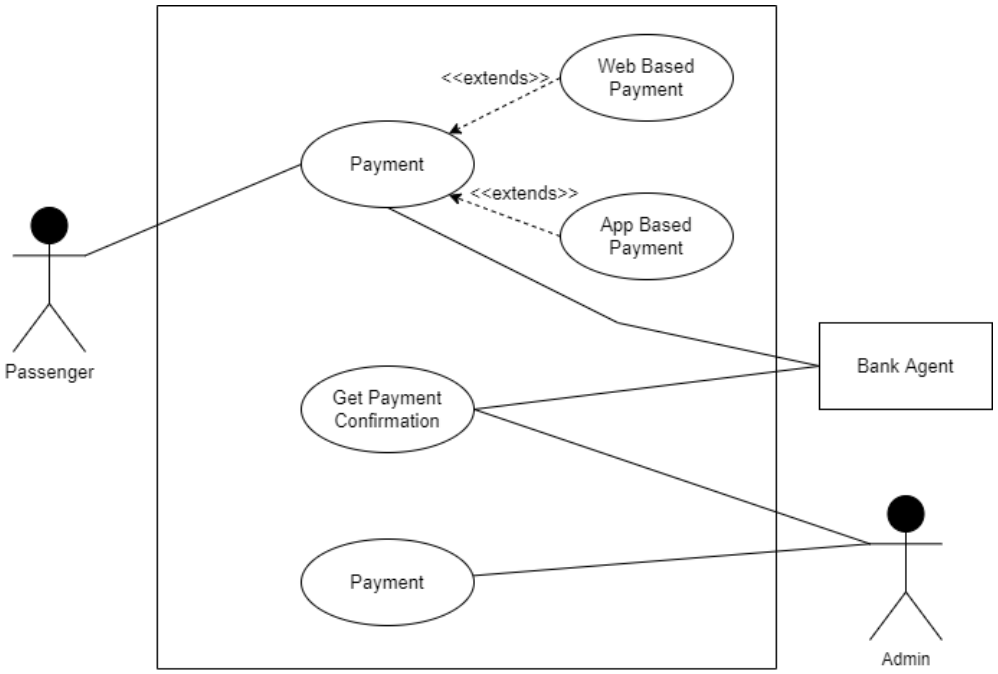
Alternate Scenario: Login Failure



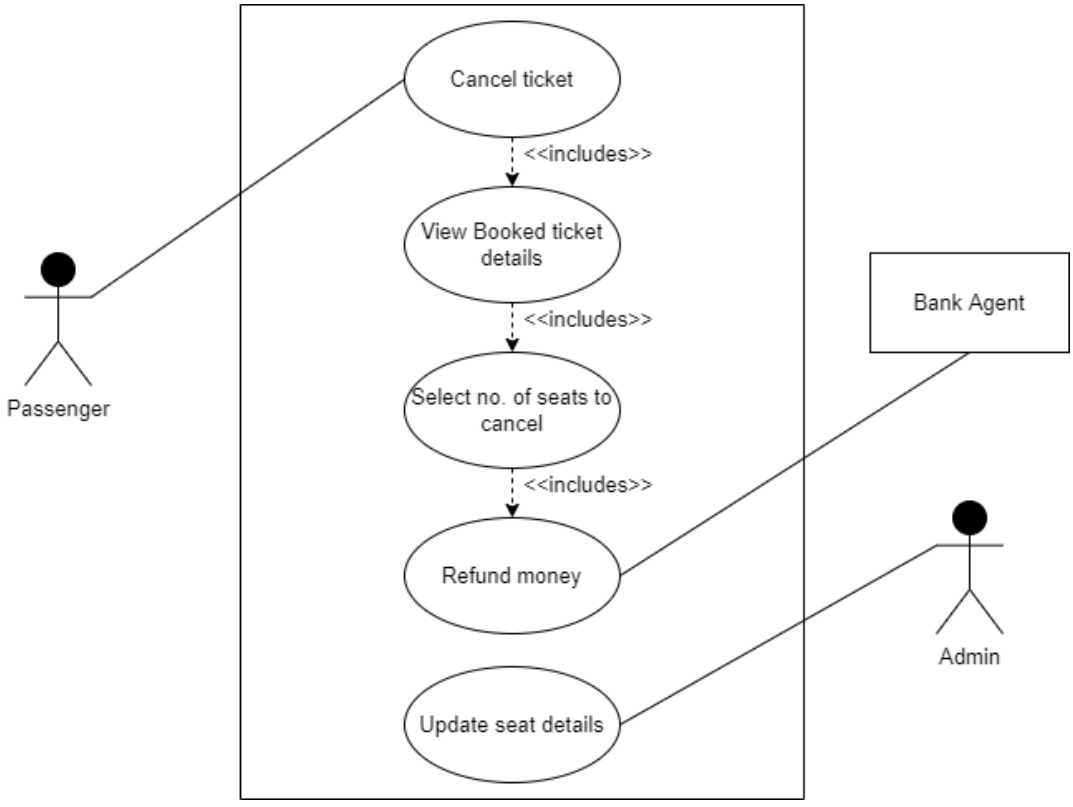
Alternate Scenario: Unavailability



Sub-Function: Payment



Sub-Function: Cancellation



FULLY DRESSED USE CASE DESCRIPTION

Main Success Scenario

Scope: Online Railway Ticket Booking
Level: User Goal
Primary Actors: Passengers, Admin
Stakeholders and Interests:

- **Passenger:** Wants to book train ticket online.
- **Administrator:** Has complete control over the system and wants to manage all activities.
- **Bank Agent:** Wants to take over payment process.

Preconditions: Administrators are authenticated and databases are connected. User has internet connection.
Success Guarantee: The ticket is booked successfully. The tickets have been confirmed. The payment is validated and recorded. The receipt is generated.

Main Success scenario:

1. Passenger has to login to the their account.
2. Passenger is redirected to their homepage/ dashboard after validating login.
3. Passenger checks for the ticket availability.
4. Passenger books the ticket with the required details being asked.
5. Passenger provides the payment information.
6. System validates the payment and reservation.
7. System provides PNR number with the traveller's details.
8. User checks the current status of the ticket in the ticket status menu.
9. System prints the receipt.

Extensions (Alternate Scenarios):

1a. Login does not exist.

1. User creates a new account with username and password.
2. If the username already exists, it shows an error that the username already exists.
3. User has to use a new username and password with the required details.

1b. Login information is incorrect.

1. If either the username or password is incorrect, it will show an error.
2. The user has to retry with correct username and password.
3. If the user forgets their password, then they can reset the password using their email id or phone number.
4. The details have been stored in the database for the future use.
5. After the login process the user has to move to the homepage.

3a. Seats being searched is not available.

1. Passenger checks availability of train tickets based upon the travel date / source and destination / train number or train names.
2. If the train with given constraints is not available, then it will show that the train is unavailable.
3. Passenger leaves the system.

Special Requirements:

1. Availability of mobile or laptop with internet connection.
2. Knowledge on booking tickets online.
3. Should be comfortable with english language.
4. The passenger should have proper bank account.
5. The passenger should have sufficient balance in their account.

Technology and Data Variation List:

*a. We can book ticket through desktops, mobile phones or laptops.
5a. Payment mode can be of either Web based or App based.

Frequency of Occurrence: Should be nearly continuous.

Payment Module

Scope: Online Railway Ticket Booking
Level: Sub-Function
Primary Actors: Passengers, Bank Agent
Stakeholders and Interests:

- **Passenger:** Wants to book train ticket online.
- **Bank Agent:** Wants to take over payment process.

Preconditions: Bank agent are logged in and databases are connected. Users have internet connection.
Success Guarantee: The tickets have been confirmed. The payment is validated and recorded. The receipt is generated.
Payment Module Sub-function:

1. After filling the passenger's details, payment process starts.
2. Passenger selects one from payment modes: browser based payment and app based payment.
3. The browser based payment is done with the use of debit card or credit card.
4. Details like card number , cvv number, card's expiry date are given while doing the payment.
5. The app based payment method is done through apps like Gpay, Paytm, etc.
6. Details like UPI id are given.
7. After this, the passenger recieves the payment confirmation notification.
8. The payment status is updated by the Admin and ticket gets booked.

Special Requirements:

1. Availability of mobile or laptop with internet connection.
2. Passenger should have proper bank account.
3. Knowledge on direct payment through bank account or through payment apps.
4. The passenger should have sufficient balance in their account.

Frequency of Occurrence: Should be nearly continuous.

Ticket Cancellation Function

Scope: Online Railway Ticket Booking
Level: Sub-Function
Primary Actors: Passengers, Admin
Stakeholders and Interests:

- **Passenger:** Wants to book train ticket online.
- **Administrator:** Has complete control over the system and wants to manage all activities.
- **Bank agent:** Wants to take over refund process.

Preconditions: Bank agent are logged in and databases are connected. Users have internet connection. User must have booked ticket recently.
Success Guarantee: The tickets have been cancelled successfully. Refund amount is credited into bank account.
Payment Module Sub-function:

1. Passenger cancels the booked tickets at any case.
2. Passenger has to go to the cancel ticket option, which is present in the homepage and in there, the booked ticket details are shown.
3. Passenger selects the number of seats to be cancelled.
4. After the cancellation, the refunded amount is tranferred to the passenger's account by the bank agent.
5. Seat details are updated by the Admin in the database.

Special Requirements:

1. Availability of mobile or laptop with internet connection.
2. Passenger should have proper bank account.

Frequency of Occurrence: not nearly continuous.

Exp. No: 4
Date: 31/3/22

DOMAIN DIAGRAM
&
CLASS DIAGRAM

AIM

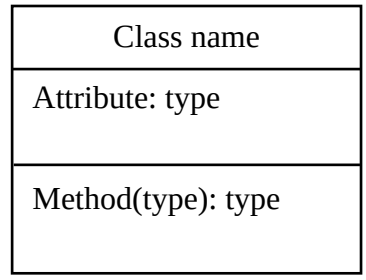
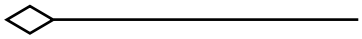
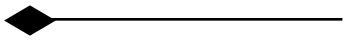
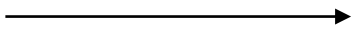
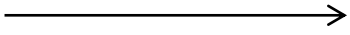
To identify the classes and their associations and draw domain model and class diagram for Online Railways Reservation System.

UML NOTATIONS

a) For Domain model Diagram

Figure	Notation	Explanation
Class name Attribute: type	Conceptual Class	Represents various objects of the system.
	Relationship	Represents the association objects or classes.
1..* 1	Multiplicity of the role	Represents how many instances of class A can be associated with one instances of a class B.
	Generalization	Child class inherits the characteristics of base class.

b) For Class Diagram

Figure	Notation	Explanation
	Conceptual Class	Represents various objects of the system along with its attributes and methods.
	Aggregation	Represents the relationship between two classes where one class is a part of another where one class is meaningful even without the aggregate.
	Composition	Represents the relationship between two classes where one class is a part of another where one class is not meaningful without the other.
	Generalization	Child class inherits the characteristics of base class.
	Association	Relationship between objects or the classes

IDENTIFICATION OF CLASSES

a) Conceptual Class Category List

Conceptual class category	Conceptual Class
Physical or tangible object	Train
Specifications, designs, or descriptions of things	Train description
Places	-
Transactions	Payment, Reservation
Transaction line items	-
Roles of people	Passenger, Admin, Bank Agent
Containers of other things	Train
Things in a container	Passenger
Other computer or electro-mechanical systems external to the system	Credit payment, Authorization system
Abstract noun concepts	-
Organizations	-
Events	Payment, Booking
Processes	Booking a seat
Rules and policies	Refund policy, Cancellation policy

Records of finance, work, contracts, legal matters	-
Financial instruments and services	-
Manuals, documents, reference papers, books	Daily price change list, Repair manual

b) Identification of Noun Phrases

- User
- Admin
- Passenger
- Bank agent
- Train
- Train description
- Railways
- Ticket
- Booking Details
- Payment Gate
- Account
- Database
- Card
- Payment app
- Reservation system

IDENTIFICATION OF ASSOCIATIONS

a) Association Category List

Category	Between
A is a physical part of *B*	Train – Railways
A is a logical part of *B*	User – Account, Card – Payment_module, Payment_App – Payment_Module
A is physically contained in *B*	-
A is logically contained in *B*	Railway_Reservation_System – Payment_module, Railway_reservation_system – Database
A is recorded by *B*	Railway_Reservation_System – Ticket

A manages *B*	Admin – Railway_Reservation_System
A uses *B*	Passenger – Railway_Reservation_System
A is related to a transaction *B*	Passenger – Payment_module, Passenger – Ticket
A is a transaction related to another transaction *B*	Ticket – Payment_Module
A is next to *B*	Train – Train
A is owned by B	Admin – Railways_Reservation_System

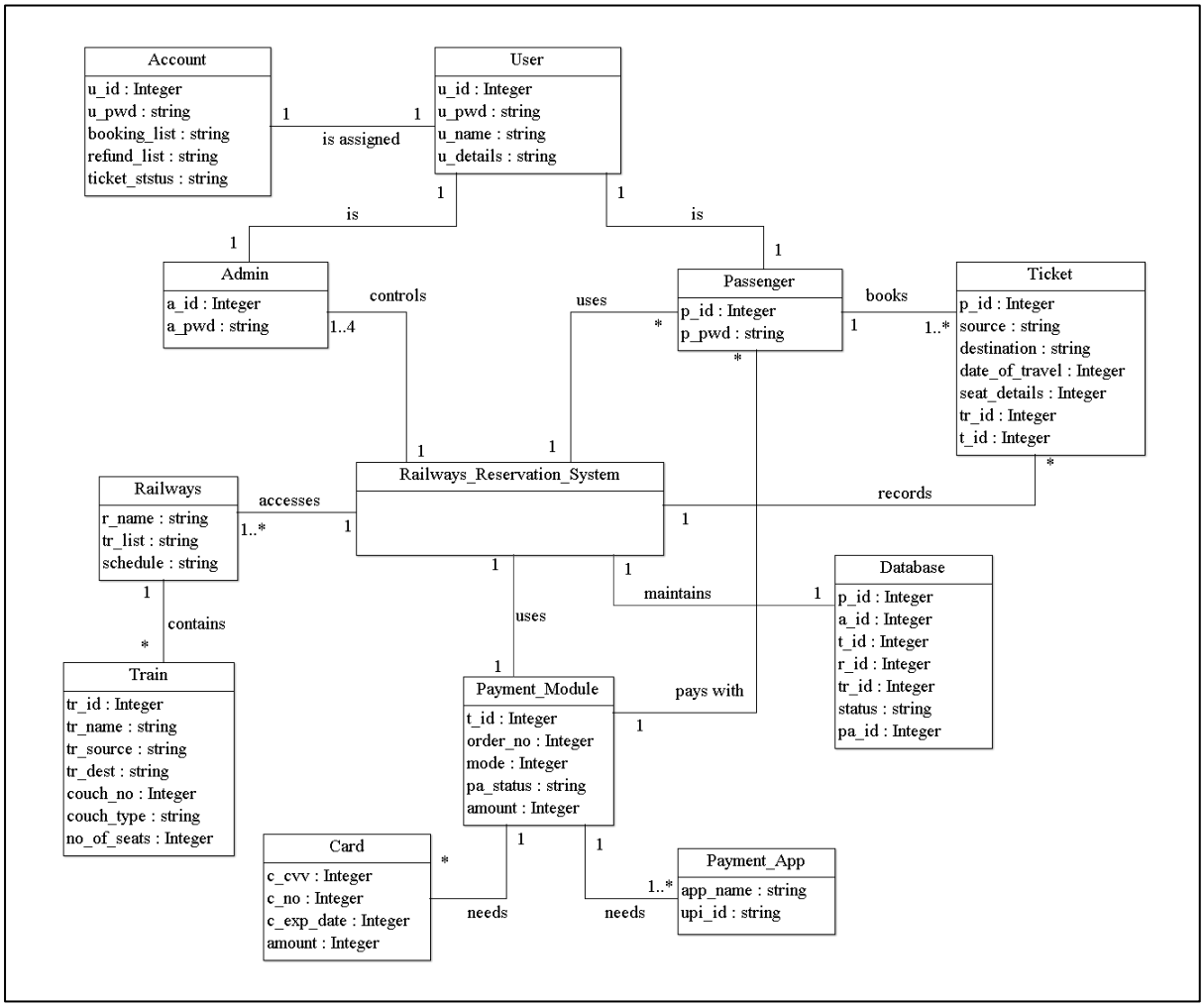
b) Definition of Associations

Association	**Explanation**
Generalization	The Generalization association ("is a") is the relationship between the base class that is named as “superclass” or “parent” and the specific class that is named as “subclass” or “child”
Aggregation	An aggregation is a relationship between classes which says one class ‘is a part of’ another class. In aggregation, the part (constituent) is meaningful without the whole (aggregate)
Composition	An aggregation is a relationship between classes which says one class ‘is a part of’ another class. In composition, the part (component) is not meaningful without the whole (container)

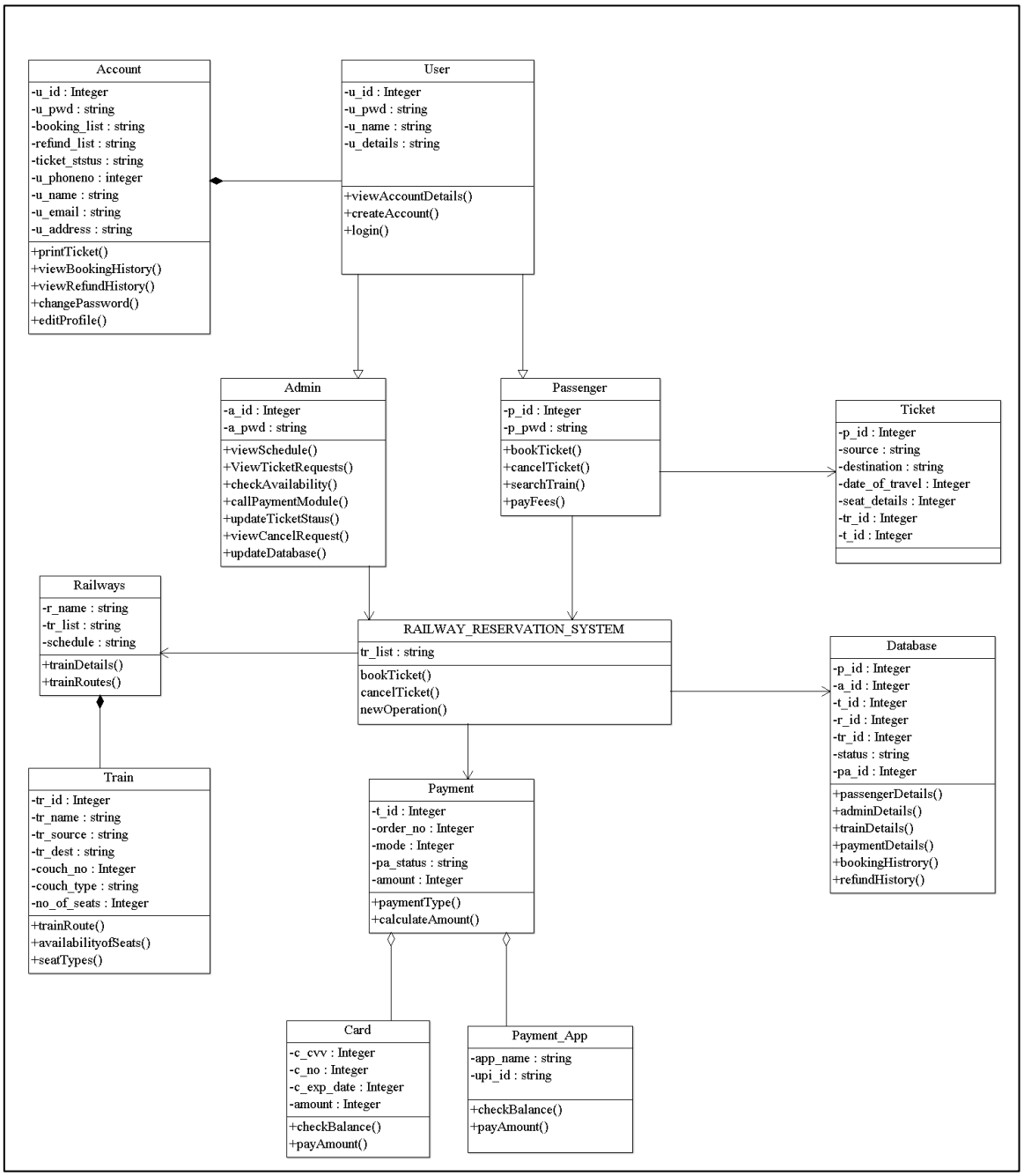
c) Multiplicity

Relationship	**Multiplicity**
Railway_Reservation_System - Railways	One System (1) can access one or more Railways (1..*)
User - Admin	User (1) may be an Administrator (1)
User - Passenger	User (1) may be an Passenger (1)
Admin - Railway Reservation System	One to Four Admins (1..4) can control the System (1)
Passenger – Railway_Reservation_System	Any number of Passengers (*) can use the system (1)
Passenger - Ticket	One Passenger (1) can book one or more tickets (1..*)
Railway Reservation_System - Database	The System (1) can maintain only one Database(1)
Railway Reservation System - Payment Module	The System (1) uses one Payment Module (1)
Payment_Module - Card	Payment (1) can be done through any type of Card (*)
Payment_Module - Payment_App	Payment (1) can be done in one or more Payment apps (1..*)
Railway_Reservation_System - Ticket	The System (1) can record any number of ticket bookings (*)
User - Account	One User (1) can have only one Account (1)
Railways - Train	One Railways (1) can contain any number of trains (*)
Passenger – Payment_Module	All Passengers (*) can access only one payment module (1)

DOMAIN MODEL DIAGRAM



CLASS DIAGRAM

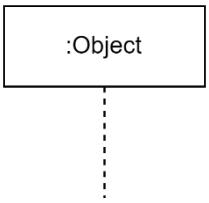
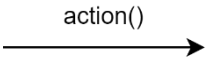
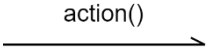
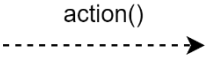


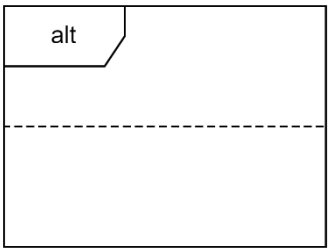
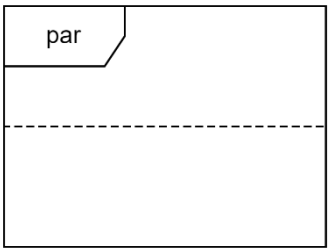
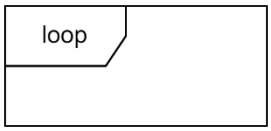
Exp. No: 5
Date: 7/4/22

INTERACTION DIAGRAM

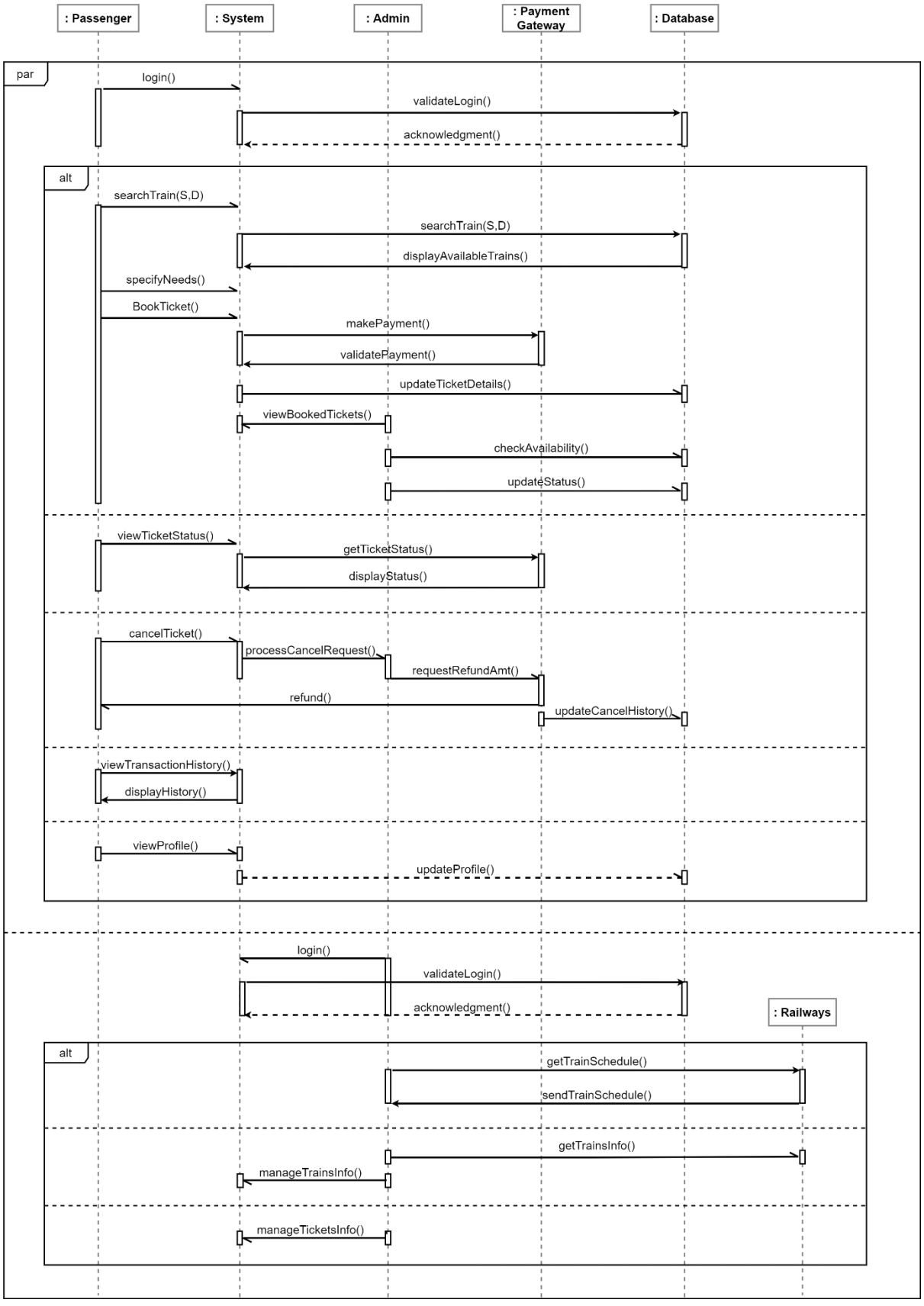
UML NOTATIONS

a) For Sequence Diagram

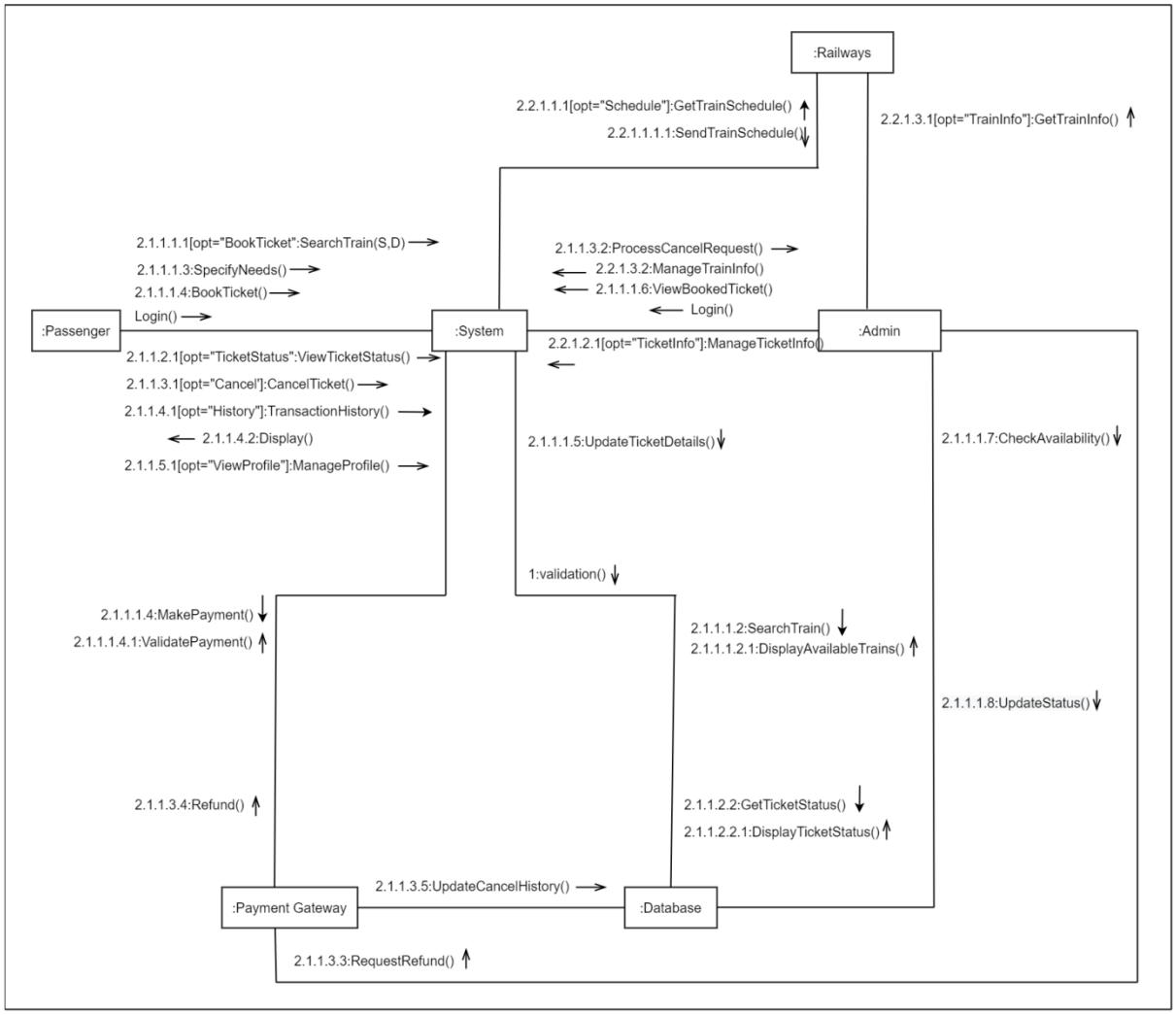
Figure	Notation	Explanation
	Object	Represents an actor or object. The dashed vertical line shows the sequential events that occur to an object during the charted process
	Activation Frame	Represents the time needed for an object to complete a task. The longer the task will take, the longer the activation box becomes.
	Synchronous Action	Represents a synchronous event happening from one object to another. Must contain a response back with the same notation.
	Asynchronous Action	Represents an asynchronous event happening from one object to another. Does not need a response back.
	Optional Action	Represents an event that may or may not happen.

	Alternate Events Frame	Represents a set of actions that may based on a condition for each section in the frame.
	Parallel Events Frame	Represents a set of actions that happen parallelly.
	Loop Frame	Represents a set of actions that happen in a loop.
	Actor	Represents an actor in the system.

SEQUENCE DIAGRAM



COLLABORATION DIAGRAM

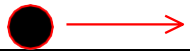


Exp. No: 6
Date: 21/4/22

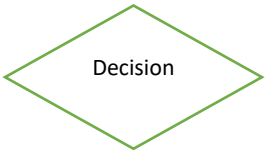
STATE MACHINE DIAGRAM
&
ACTIVITY DIAGRAM

UML NOTATIONS

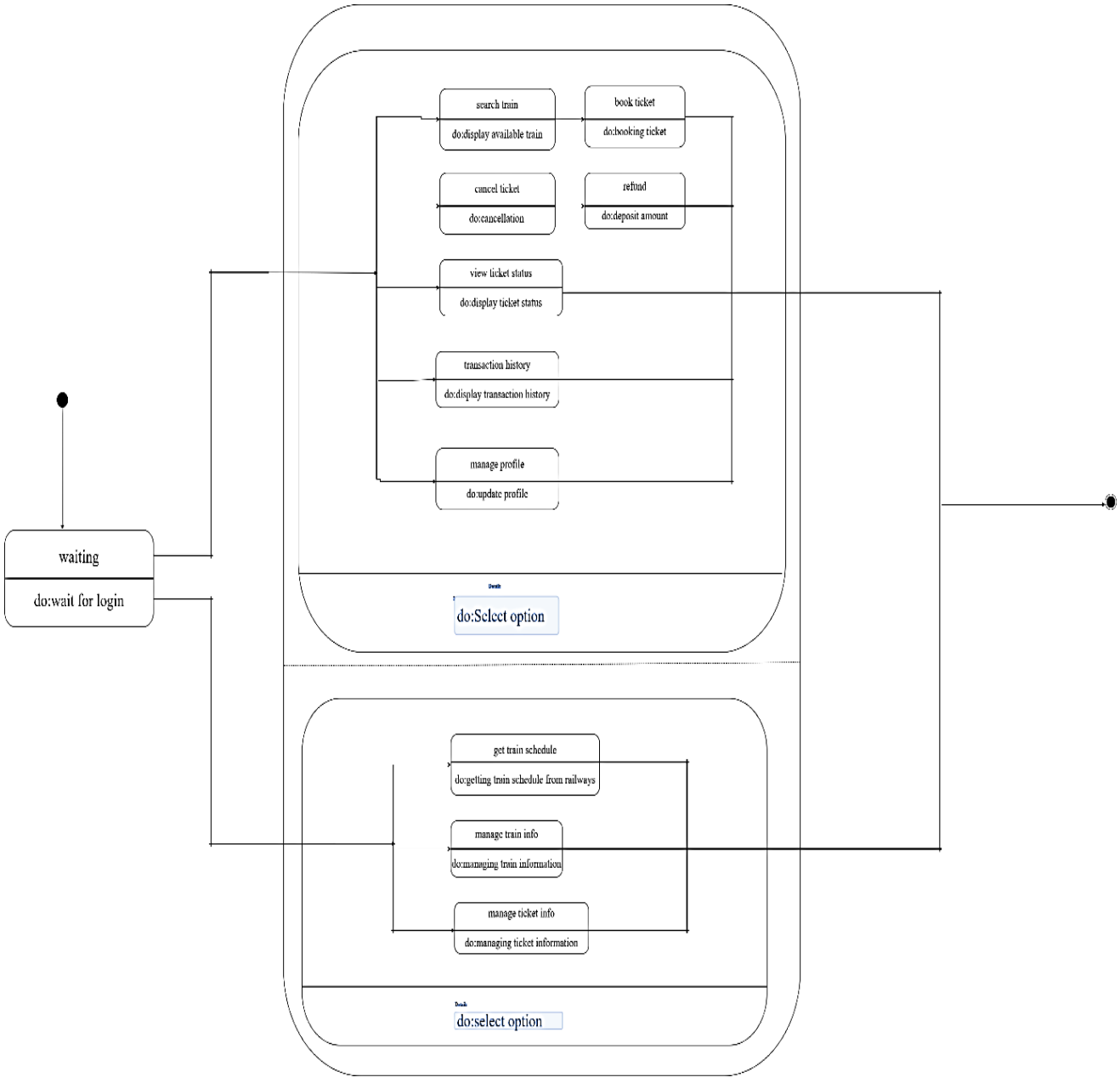
a) For State Machine Diagram

Notation	Type	Explanation
	Start	It initiates the transition
	End	It marks the end of the transitions
	State	Each state in the state machine is represented using thisrounded rectangle
[condition]	Transition based on condition	It represents transition based on condition
\Action	Transition based on action	It represents the transition based on the user action
	Transition	It represents transition without any action or condition

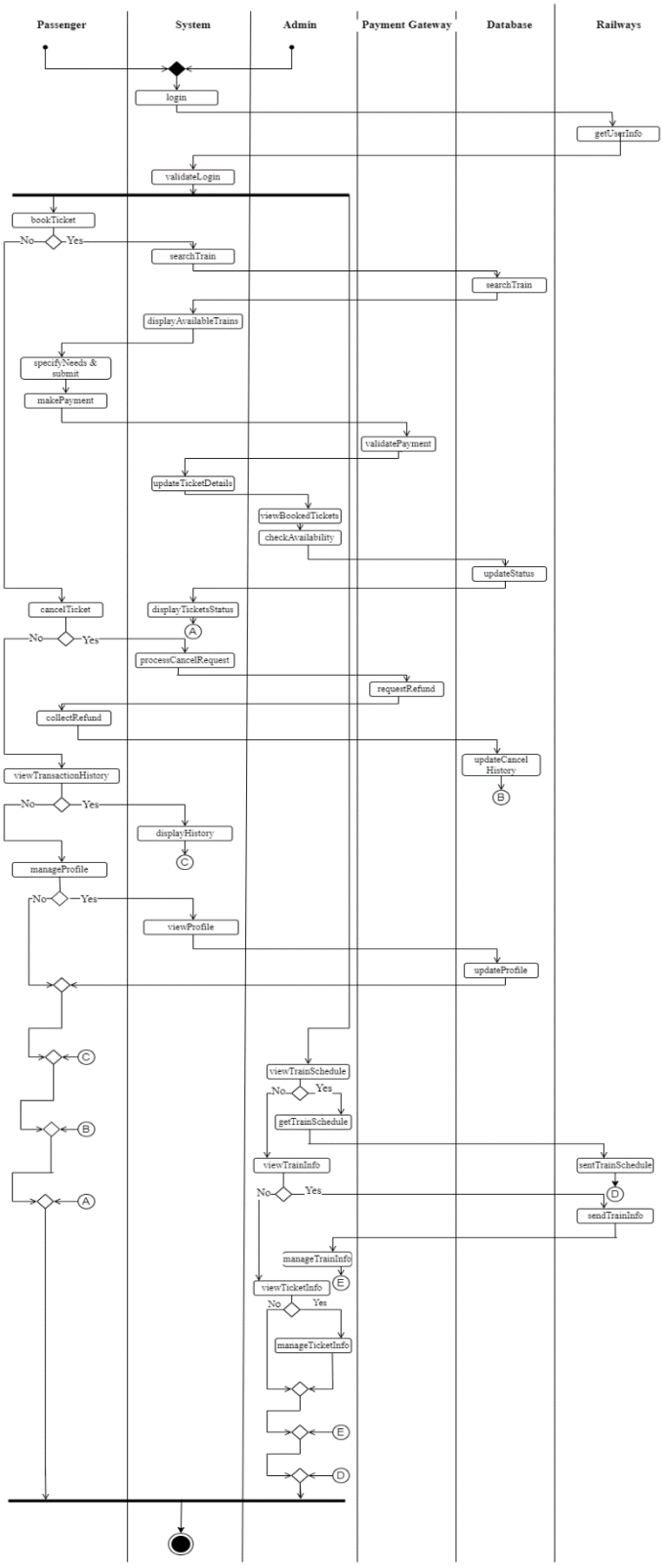
b) For Activity Diagram

Notation	Type	Explanation
	Start	It initiates the transition
	End	It marks the end of the transitions
	State	Each state in the state machine is represented using thisrounded rectangle
	Decision	It represents the transitions based on the conditions
	Fork	It is used for concurrent executions

STATE MACHINE DIAGRAM

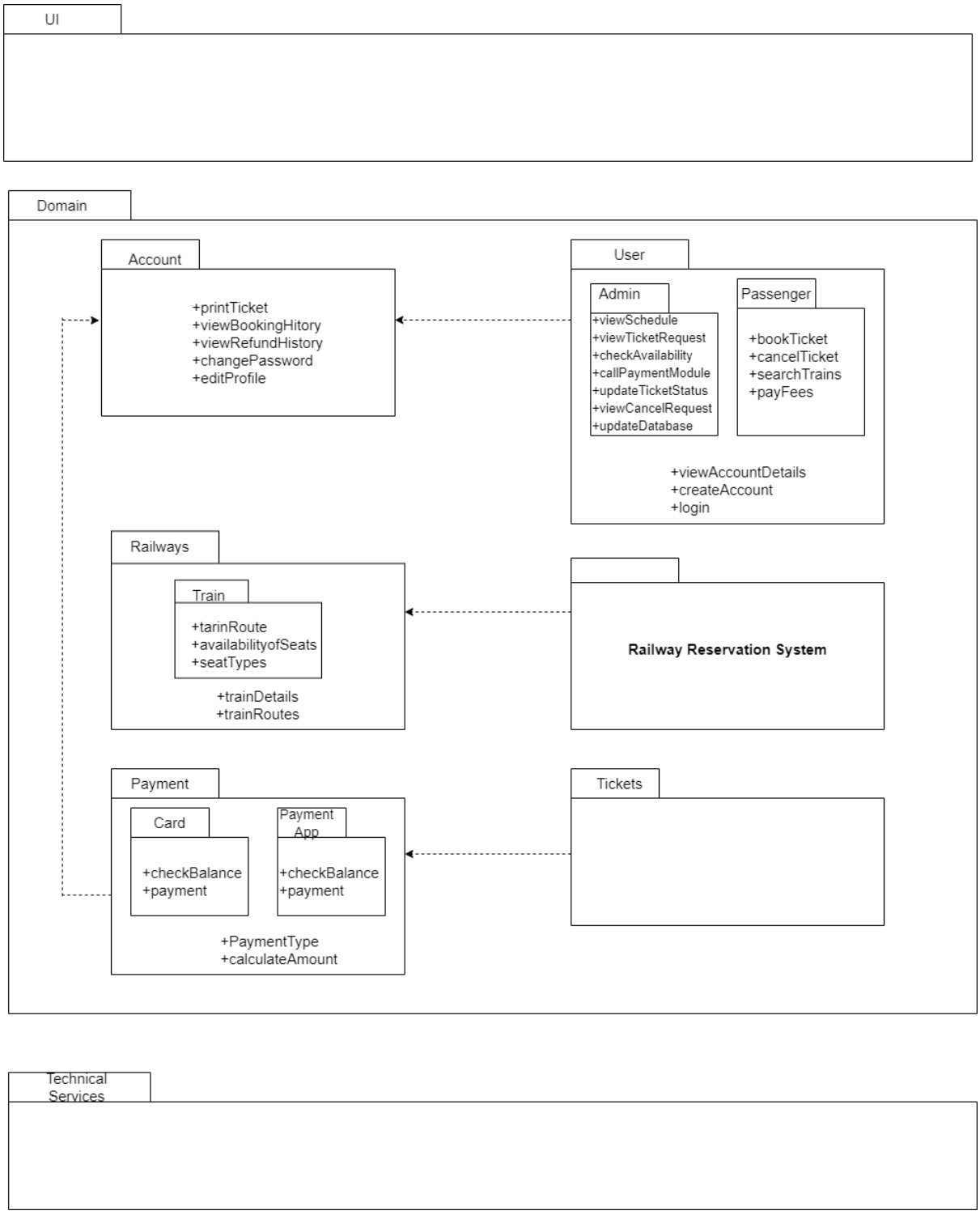


ACTIVITY DIAGRAM

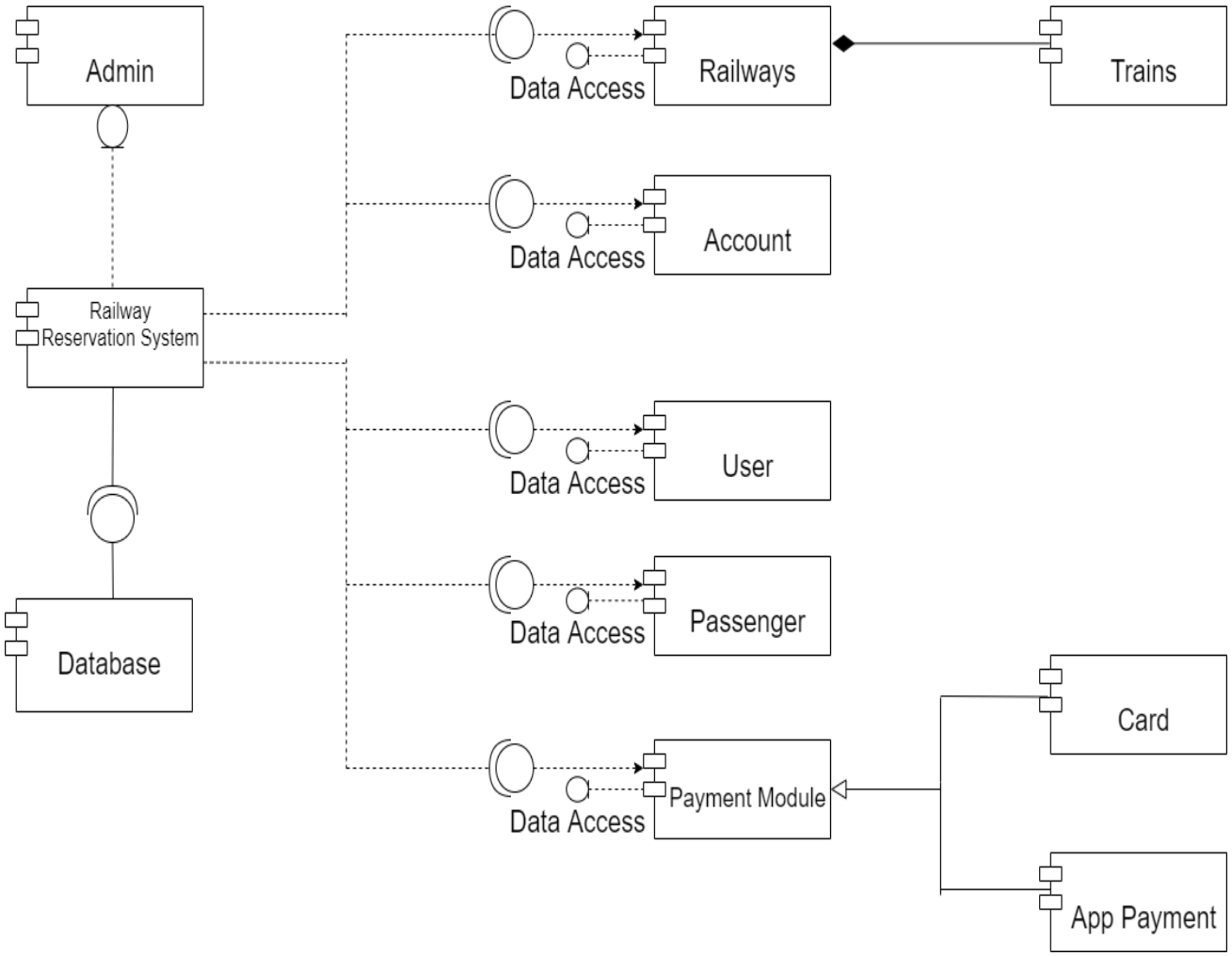


Exp. No: 7
Date: 23/4/22

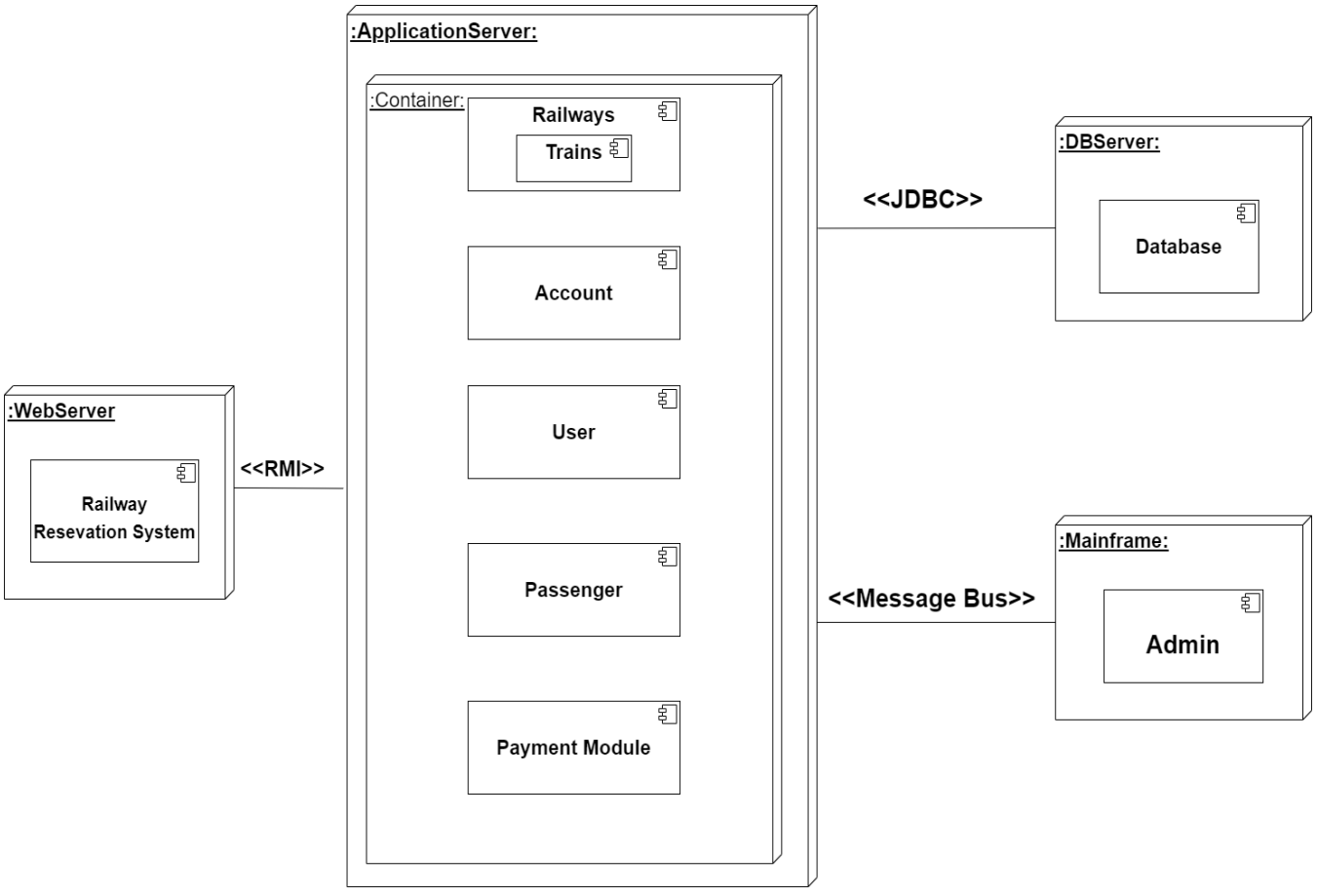
PACKAGE DIAGRAM


UI
Domain
Account
+printTicket
+viewBookingHitory
+viewRefundHistory
+changePassword
+editProfile
User
Admin
+viewSchedule
+viewTicketRequest
+checkAvailability
+callPaymentModule
+updateTicketStatus
+viewCancelRequest
+updateDatabase
Passenger
+bookTicket
+cancelTicket
+searchTrains
+payFees
+viewAccountDetails
+createAccount
+login
Railways
Train
+tarinRoute
+availabilityofSeats
+seatTypes
+trainDetails
+trainRoutes
Railway Reservation System
Payment
Card
+checkBalance
+payment
Payment App
+checkBalance
+payment
+PaymentType
+calculateAmount
Tickets
Technical Services

COMPONENT DIAGRAM

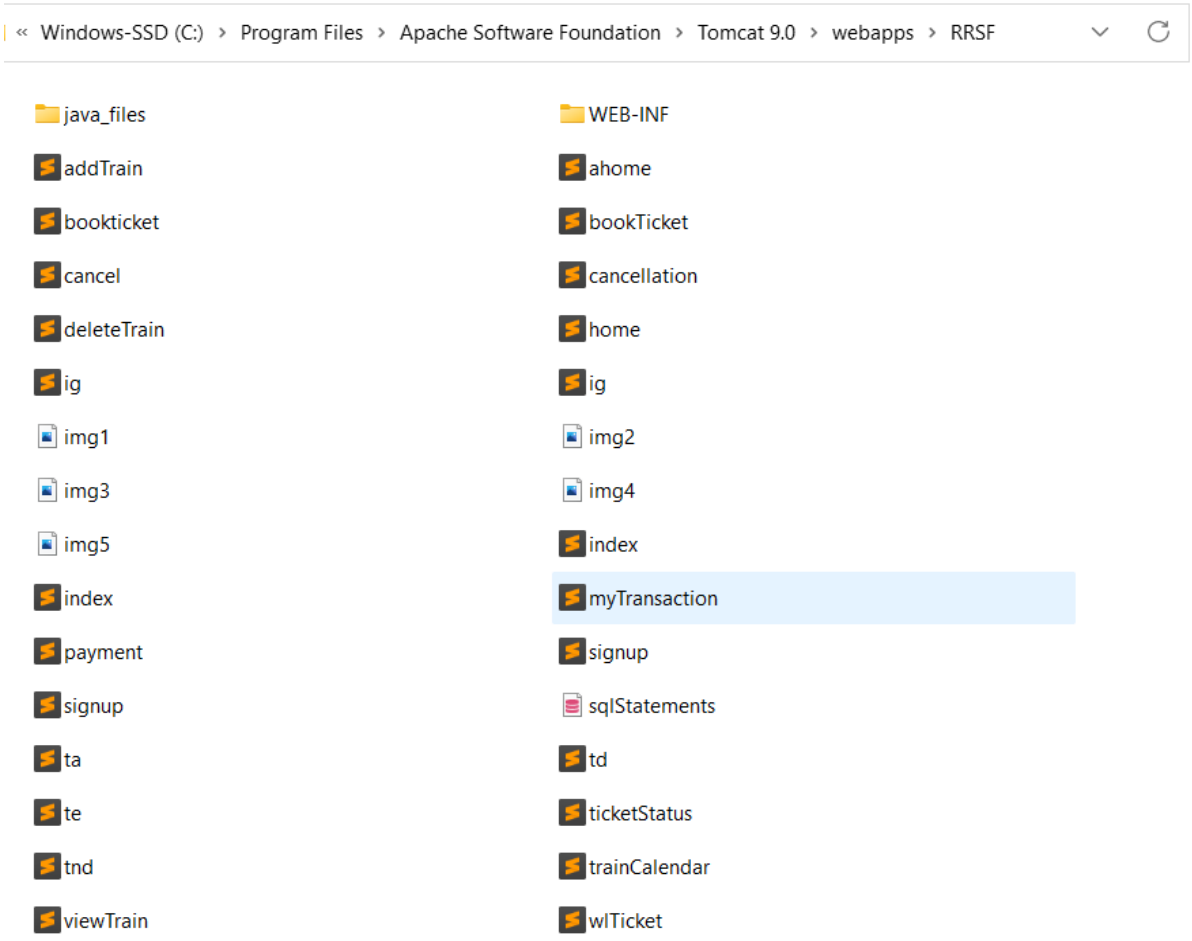

Admin
Railway
Reservation System
Database
Data Access
Railways
Trains
Data Access
Account
Data Access
User
Data Access
Passenger
Data Access
Payment Module
Card
App Payment

DEPLOYMENT DIAGRAM

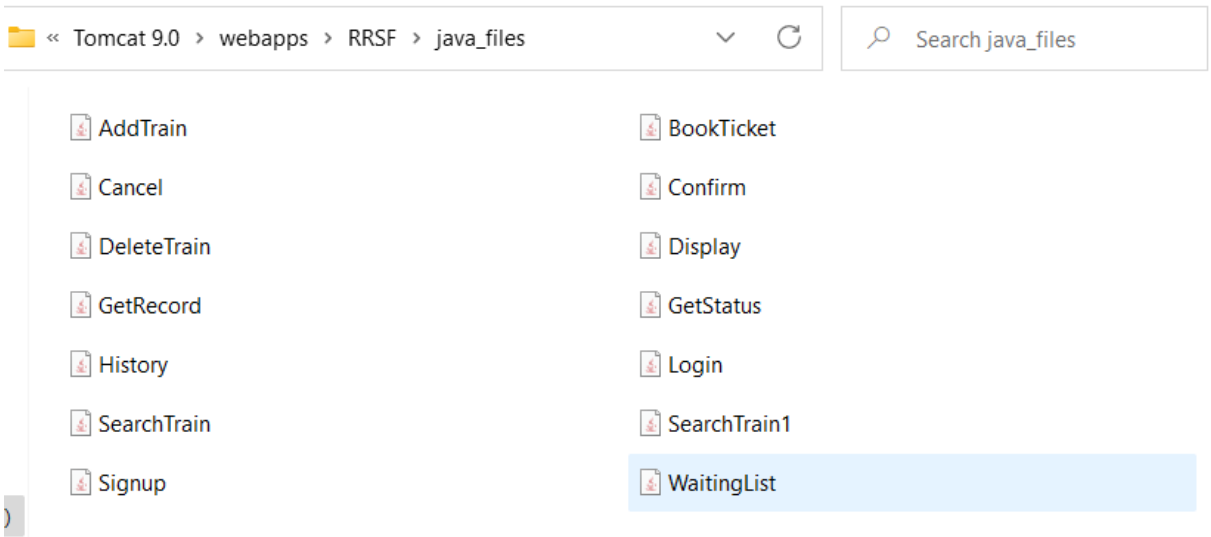

:ApplicationServer:
:Container:
Railways
Trains
Account
User
Passenger
Payment Module
:WebServer
Railway
Resevation System
<<RMI>>
<<JDBC>>
:DBServer:
Database
<<Message Bus>>
:Mainframe:
Admin

IMPLEMENTATION DEMO

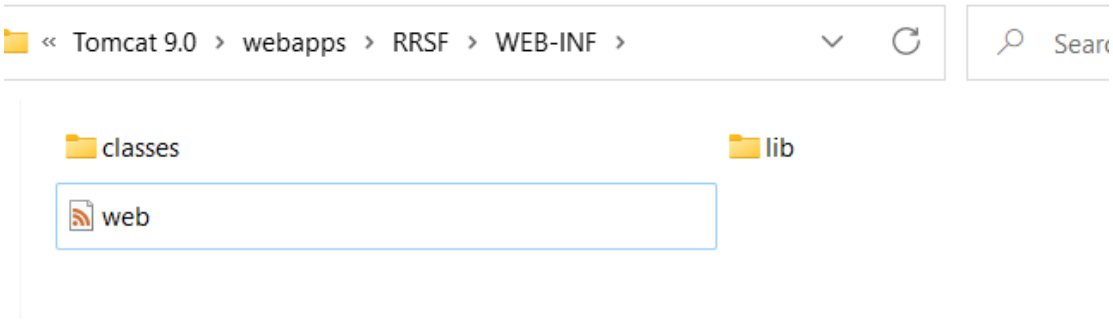
Directory Structure



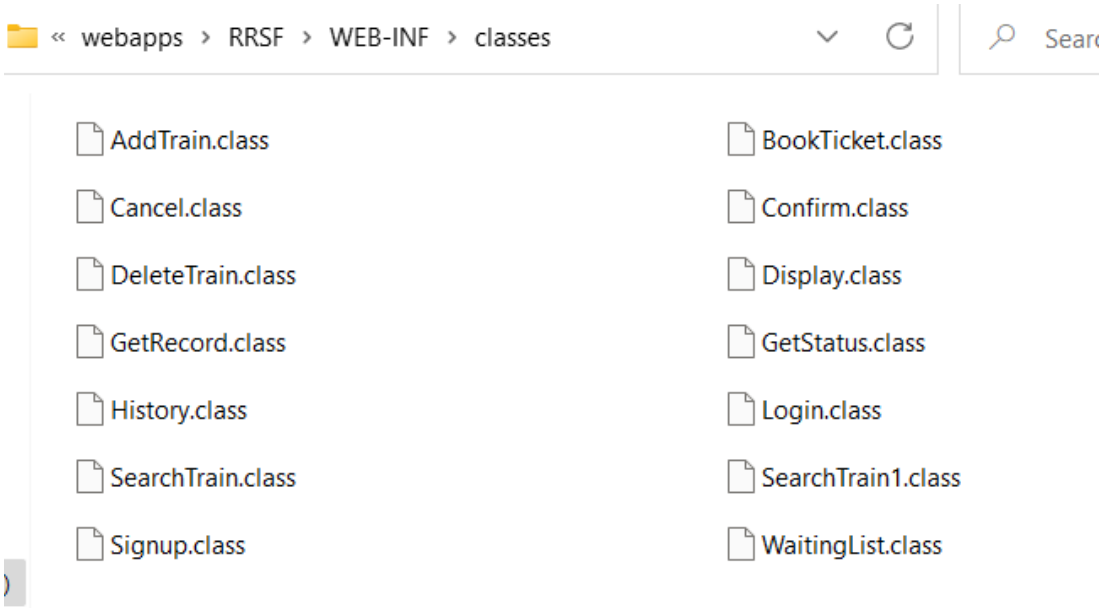
Inside java_files:



Inside WEB-INF:



Inside classes:



Source code

AddTrain.java:

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
import java.sql.*;
import java.util.*;

public class AddTrain extends HttpServlet {
    public int addTrain(String tid, String name, String source, String
destination, String max_seats, String duration) throws Exception{
        String url = "jdbc:mysql://localhost:3306/rrs";
        String sqluser = "root";
        String sqlpwd = "Mysql@02";

        Class.forName("com.mysql.cj.jdbc.Driver");

        System.out.println("Hey I am after forname method in addTrain() -
AddTrain class");

        Connection con = DriverManager.getConnection(url, sqluser, sqlpwd);

        Statement st = con.createStatement();

        System.out.println("Hey I am after create method in addTrain() -
AddTrain class");
        int ms=Integer.parseInt(max_seats);
        int ti=Integer.parseInt(tid);
        int du=Integer.parseInt(duration);
        try {
            String s = String.format("select count(*) from train where tid =
'%d'", ti);
            ResultSet rs = st.executeQuery(s);
            rs.next();
            int count = rs.getInt(1);
            if(count > 0) {
                return 0;
            }

            s = String.format("insert into train(tid, name, source,
destination, max_seats, duration) values('%d','%s','%s','%s','%d','%d')", ti,
name, source, destination, ms, du);
            st.executeUpdate(s);
            System.out.println("Train record inserted...");
        }
```

```
        catch(Exception e) {
            System.out.println(e);
            return -1;
        }
        return 1;
    }
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter pw = res.getWriter();
        String id = req.getParameter("tid");
        String name = req.getParameter("tname");
        String source = req.getParameter("source");
        String destination = req.getParameter("destination");
        String max_seats = req.getParameter("maxseats");
        String duration = req.getParameter("duration");
        System.out.println(id + ", " + name + ", " + source + ", " +
destination + ", " + max_seats + ", " + duration);

        try {
            int result = addTrain(id, name, source, destination, max_seats,
duration);
            if(result == 0) {
                System.out.println("Train already exists...");
                res.sendRedirect("te.html");
            }
            else if(result == 1) {
                System.out.println("New Train added");
                res.sendRedirect("ta.html");
            }
        }
        catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

BookTicket.java:

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.sql.*;
import java.io.*;
import java.util.*;
```

```
public class BookTicket extends HttpServlet {
    public int addTicket(String pname, String trname, int trno, String src,
String dest, String dot, int asno, int csno) {
        String url = "jdbc:mysql://localhost:3306/rrs";
        String sqluser = "root";
        String sqlpwd = "Mysql@02";
        try {
            Class.forName("com.mysql.jdbc.Driver");

            System.out.println("Hey I am after forname method in addTicket() -
Bookticket class");
            Connection con = DriverManager.getConnection(url, sqluser,
sqlpwd);
            Statement st = con.createStatement();
            System.out.println("Hey I am after create method in addTicket() -
Bookticket class");
            String s = String.format("select pid from passenger where
name='%s'", pname);
            ResultSet rs = st.executeQuery(s);
            if(rs.next()) {
                int pid = rs.getInt(1);
                s = String.format("insert into ticket(pid, tid, dot, ano, cno,
ticstatus) values('%d', '%d', '%s', '%d', '%d', 'Waiting')", pid, trno, dot,
asno, csno);
                st.executeUpdate(s);
                System.out.println("Ticket details inserted...");
            }
            else {
                System.out.println("Passenger doesn't exist");
                return 0;
            }
            return 1;
        }
        catch(Exception e) {
            System.out.println(e);
            return -1;
        }
    }
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
        String pname = req.getParameter("spname");
        String trname = req.getParameter("stname");
        int trno = Integer.parseInt(req.getParameter("stno"));
        String src = req.getParameter("sfrom");
        String dest = req.getParameter("sto");
        String dot = req.getParameter("sdot");
        int asno = Integer.parseInt(req.getParameter("sasno"));
        int csno = Integer.parseInt(req.getParameter("scsno"));
```

```
        System.out.println("Ticket to be added...");
        System.out.println(pname + ", " + trname + ", " + trno + ", " + src +
", " + dest + ", " + dot + ", " + asno + ", " + csno);
        try {
            int result = addTicket(pname, trname, trno, src, dest, dot, asno,
csno);
            if(result == 1) {
                System.out.println("Ticket Booked...");
                res.sendRedirect("payment.html");
            }
            else {
                System.out.println("Ticket not booked...");
                res.sendRedirect("bookTicket.html");
            }
        }
        catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

Cancel.java:

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.sql.*;
import java.io.*;
import java.lang.reflect.Executable;
import java.util.*;
public class Cancel extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
        String url = "jdbc:mysql://localhost:3306/rrs";
        String sqluser = "root";
        String sqlpwd = "Mysql@02";
        int ticno = Integer.parseInt(req.getParameter("ticno"));
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            System.out.println("Hey I am after forname method in get() -
Cancel class");
            Connection con = DriverManager.getConnection(url, sqluser,
sqlpwd);
            Statement st = con.createStatement();
            System.out.println("Hey I am after create method in get() - Cancel
class");
            String s = String.format("update ticket set ticstatus='Cancelled'
where ticno='%d'", ticno);
            st.executeUpdate(s);
```

```
        }
        catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

Confirm.java:

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.sql.*;
import java.io.*;
import java.lang.reflect.Executable;
import java.util.*;

public class Confirm extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
        String url = "jdbc:mysql://localhost:3306/rrs";
        String sqluser = "root";
        String sqlpwd = "Mysql@02";
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            System.out.println("Hey I am after forname method in get() -
Confirm class");
            Connection con = DriverManager.getConnection(url, sqluser,
sqlpwd);
            Statement st = con.createStatement();
            System.out.println("Hey I am after create method in get() -
Confirm class");
            String s = String.format("update ticket set ticstatus='Confirmed'
where ticstatus='Waiting'");
            st.executeUpdate(s);
            PrintWriter pw = res.getWriter();
            pw.println("OK");
        }
        catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

DeleteTrain.java:

```
import javax.servlet.http.*;
import javax.servlet.*;
```

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

public class DeleteTrain extends HttpServlet {
    public int deleteTrain(String tid) throws Exception{
        String url = "jdbc:mysql://localhost:3306/rrs";
        String sqluser = "root";
        String sqlpwd = "Mysql@02";
        Class.forName("com.mysql.cj.jdbc.Driver");
        System.out.println("Hey I am after forname method in deleteTrain() -
DeleteTrain class");
        Connection con = DriverManager.getConnection(url, sqluser, sqlpwd);
        Statement st = con.createStatement();
        System.out.println("Hey I am after create method in deleteTrain() -
DeleteTrain class");
        int ti=Integer.parseInt(tid);
        try {
            String s = String.format("select count(*) from train where tid =
'%d'", ti);
            ResultSet rs = st.executeQuery(s);
            rs.next();
            int count = rs.getInt(1);
            if(count == 0) {
                return 0;
            }
            s = String.format("delete from train where tid = '%d'", ti);
            st.executeUpdate(s);
            System.out.println("Train Record deleted...");
        }
        catch(Exception e) {
            System.out.println(e);
            return -1;
        }
        return 1;
    }
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter pw = res.getWriter();
        String id = req.getParameter("tid");
        System.out.println(id);
        try {
            int result = deleteTrain(id);
            if(result == 0) {
                System.out.println("Train Record does not exist...");
                res.sendRedirect("tnd.html");
            }
```

```
            else if(result == 1) {
                System.out.println("Train Record deleted...");
                res.sendRedirect("td.html");
            }
        }
        catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

Display.java:

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.sql.*;
import java.io.*;
import java.util.*;
public class Display extends HttpServlet {
    public String[][] getTrains(String src, String dest) throws Exception {
        String[][] trainlist = new String[10][6];
        String url = "jdbc:mysql://localhost:3306/rrs";
        String sqluser = "root";
        String sqlpwd = "Mysql@02";
        try {
            Class.forName("com.mysql.jdbc.Driver");
            System.out.println("Hey I am after forname method in getTrains() -
SearchTrain class");
            Connection con = DriverManager.getConnection(url, sqluser,
sqlpwd);
            Statement st = con.createStatement();
            System.out.println("Hey I am after create method in getTrains() -
SearchTrain class");
            String s = String.format("select * from train where source = '%s'
and destination = '%s'", src, dest);
            ResultSet rs = st.executeQuery(s);
            int i = 0;
            while(rs.next()) {
                trainlist[i][0] = Integer.toString(rs.getInt(1));
                trainlist[i][1] = rs.getString(2);
                trainlist[i][2] = rs.getString(3);
                trainlist[i][3] = rs.getString(4);
                trainlist[i][4] = Integer.toString(rs.getInt(5));
                trainlist[i][5] = Integer.toString(rs.getInt(6));
                System.out.println(trainlist[i][0] + ", " + trainlist[i][1] +
", " + trainlist[i][2] + ", " + trainlist[i][3] + ", " + trainlist[i][4] + ",
" + trainlist[i][5]);
                i++;
```

```
            }
            return trainlist;
        }
        catch(Exception e) {
            System.out.println(e);
            return null;
        }
    }
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
        String source = req.getParameter("source");
        String destination = req.getParameter("destination");
        // String dot = req.getParameter("dot");
        System.out.println(source + ", " + destination);
        try {
            res.setContentType("text/html");
            PrintWriter pw = res.getWriter();

            // String[][] trlist = getTrains(source, destination, dot);
            String[][] trlist = getTrains(source, destination);
            for(int i = 0; i < trlist.length; i++) {
                for(int j = 0; j < 6; j++) {
                    System.out.println(trlist[i][j]);
                }
            }
            String trlistcontent = new String();
            // pw.println("testing");
            trlistcontent = "<table id='trainlist' border='2' cellspacing='3'
cellpadding='8' bordercolor='Blue' align='center'><tr><th>Train
Name</th><th>Train
Number</th><th>Price</th><th>Source</th><th>Destination</th><th>Available
Seats</th><th>Duration</th></tr>";
            for(int i = 0; i < trlist.length; i++) {
                if(trlist[i][0] != null) {
                    trlistcontent += "<tr>";
                    trlistcontent += "<td id='tnoin'>" + trlist[i][0] +
"</td>";
                    trlistcontent += "<td id='tnamein'>"+ trlist[i][1]
+"</td>";
                    trlistcontent += "<td id='pricein'>540</td>";
                    trlistcontent += "<td id='srcin'>"+ trlist[i][2] +"</td>";
                    trlistcontent += "<td id='desin'>"+ trlist[i][3]
+"</td>";
                    trlistcontent += "<td id='seatsin'>"+ trlist[i][4]
+"</td>";
                    trlistcontent += "<td id='durin'>"+ trlist[i][5] +"</td>";
                    trlistcontent += "</tr>";
                }
```

```
            }
            trlistcontent += "</table>";
            System.out.println(trlistcontent);
            pw.println(trlistcontent);
            pw.close();
        }
        catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

GetRecord.java:

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.sql.*;
import java.io.*;
import java.lang.reflect.Executable;
import java.util.*;

public class GetRecord extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
        String url = "jdbc:mysql://localhost:3306/rrs";
        String sqluser = "root";
        String sqlpwd = "Mysql@02";
        int ticno = Integer.parseInt(req.getParameter("ticno"));
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");

            System.out.println("Hey I am after forname method in get() -
GetRecord class");

            Connection con = DriverManager.getConnection(url, sqluser,
sqlpwd);

            Statement st = con.createStatement();

            System.out.println("Hey I am after create method in get() -
GetRecord class");
            res.setContentType("text/html");
            PrintWriter pw = res.getWriter();

            String s = String.format("select * from ticket where ticno='%d'",
ticno);
            ResultSet rs = st.executeQuery(s);
            rs.next();
```

```
            int pid = rs.getInt(2);
            int tid = rs.getInt(3);
            String dot = rs.getString(4);
            int asno = rs.getInt(5);
            int csno = rs.getInt(6);
            String status = rs.getString(7);
            s = String.format("select name from passenger where pid='%d'",
pid);
            ResultSet rs1 = st.executeQuery(s);
            rs1.next();
            String pname = rs1.getString(1);
            s = String.format("select name,source,destination from train where
tid='%d'", tid);
            ResultSet rs2 = st.executeQuery(s);
            rs2.next();
            System.out.println("executed sql");
            String record = new String();
            record = "<table border='2' cellspacing='3' cellpadding='30'
bordercolor='Blue' style='margin-left:420px;margin-top: 30px;'><tr><th>Ticket
No.</th><th>Passenger Name</th><th>Train Number</th><th>Train
Name</th><th>Source</th><th>Destination</th><th>Date of Travel</th><th>No of
Seats</th><th>Cancel Now</th></tr>";
            record += "<tr>";
            record += "<td>" + Integer.toString(ticno) + "</td>"; //ticket no
            record += "<td>" + pname + "</td>"; // passenger name
            record += "<td>" + Integer.toString(tid) + "</td>"; // train id
            record += "<td>"+ rs2.getString(1) +"</td>"; // train name
            record += "<td>"+ rs2.getString(2) +"</td>"; // src
            record += "<td>"+ rs2.getString(3) +"</td>"; // dest
            record += "<td>"+ dot +"</td>"; // date of travel
            record += "<td>"+ Integer.toString(asno + csno) +"</td>"; // no of
seats
            record += "<td><button class='open-button'
onclick='cancel()'>Cancel Ticket</button></td>";
            record += "</tr>";
            System.out.println(record);
            pw.println(record);
            pw.close();
        }
        catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

GetStatus.java:

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.sql.*;
import java.io.*;
import java.lang.reflect.Executable;
import java.util.*;

public class GetStatus extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
        String url = "jdbc:mysql://localhost:3306/rrs";
        String sqluser = "root";
        String sqlpwd = "Mysql@02";

        int ticno = Integer.parseInt(req.getParameter("ticno"));

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");

            System.out.println("Hey I am after forname method in get() -
GetStatus class");

            Connection con = DriverManager.getConnection(url, sqluser,
sqlpwd);

            Statement st = con.createStatement();

            System.out.println("Hey I am after create method in get() -
GetStatus class");

            res.setContentType("text/html");
            PrintWriter pw = res.getWriter();

            String s = String.format("select * from ticket where ticno='%d'",
ticno);
            ResultSet rs = st.executeQuery(s);
            rs.next();
            int pid = rs.getInt(2);
            // System.out.println("pid: "+ pid);
            int tid = rs.getInt(3);
            // System.out.println("tid: "+ tid);
            String dot = rs.getString(4);
            int asno = rs.getInt(5);
            int csno = rs.getInt(6);
```

```
            String status = rs.getString(7);
            s = String.format("select name from passenger where pid='%d'",
pid);
            ResultSet rs1 = st.executeQuery(s);
            rs1.next();
            String pname = rs1.getString(1);
            System.out.println("pname: "+ pname);

            s = String.format("select name,source,destination from train where
tid='%d'", tid);
            ResultSet rs2 = st.executeQuery(s);
            rs2.next();
            // System.out.println("pid: "+ pid);
            System.out.println("executed sql");
            String statusList = new String();
            statusList = pname + "*";
            statusList += dot + "*";
            statusList += rs2.getString(2) + "*";
            statusList += rs2.getString(3) + "*";
            statusList += Integer.toString(tid) + "*";
            statusList += rs2.getString(1) + "*";
            statusList += Integer.toString(asno) + "*";
            statusList += Integer.toString(csno) + "*";
            statusList += status;
            System.out.println(statusList);
            pw.println(statusList);
            pw.close();
        }
        catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

History.java:

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.sql.*;
import java.io.*;
import java.lang.reflect.Executable;
import java.util.*;

public class History extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
        String url = "jdbc:mysql://localhost:3306/rrs";
```

```
        String sqluser = "root";
        String sqlpwd = "Mysql@02";

        String email = req.getParameter("email");

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");

            System.out.println("Hey I am after forname method in getTrains() -
SearchTrain class");

            Connection con = DriverManager.getConnection(url, sqluser,
sqlpwd);

            Statement st = con.createStatement();

            System.out.println("Hey I am after create method in getTrains() -
SearchTrain class");

            res.setContentType("text/html");
            PrintWriter pw = res.getWriter();

            String s = String.format("select pid from passenger where
email='%s'", email);
            ResultSet rs = st.executeQuery(s);
            rs.next();
            int pid = rs.getInt(1);
            s = String.format("select * from ticket where pid='%d'", pid);
            ResultSet rs1 = st.executeQuery(s);

            String histList = new String();
            histList = "<table border='2' cellspacing='3' cellpadding='30'
bordercolor='Blue' align='center'><tr><th>Ticket Number</th><th>Train
id</th><th>Date of Travel</th><th>No.of.Seats</th><th>Status</th></tr>";
            while(rs1.next()) {
                int ticno = rs1.getInt(1);
                int tid = rs1.getInt(3);
                String dot = rs1.getString(4);
                int asno = rs1.getInt(5);
                int csno = rs1.getInt(6);
                String status = rs1.getString(7);

                histList += "<tr>";
                histList += "<td>" + Integer.toString(ticno) + "</td>";
//ticket no
                histList += "<td>"+ Integer.toString(tid)+"</td>"; // train id
                histList += "<td>"+ dot +"</td>"; // date of travel
```

```
                histList += "<td>"+ Integer.toString(asno+csno) +"</td>"; //
no of seats
                histList += "<td>"+ status +"</td>"; // status
                histList += "</tr>";
            }
            pw.println("</table>");
            System.out.println(histList);
            pw.println(histList);
            pw.close();
        }
        catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

Login.java:

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
import java.sql.*;
import java.util.*;

public class Login extends HttpServlet {
    public int loginUser(String email, String password) throws Exception{
        String url = "jdbc:mysql://localhost:3306/rrs";
        String sqluser = "root";
        String sqlpwd = "Mysql@02";

        Class.forName("com.mysql.jdbc.Driver");

        System.out.println("Hey I am after forname method in loginUser() -
Login class");

        Connection con = DriverManager.getConnection(url, sqluser, sqlpwd);

        Statement st = con.createStatement();

        System.out.println("Hey I am after create method in loginUser() -
Login class");

        try {
            String s = String.format("select pwd from passenger where email =
'%s'", email);
            ResultSet rs = st.executeQuery(s);
            if(rs.next()) {
                String pwd = rs.getString(1);
```

```
                if(pwd.equals(password)) {
                    System.out.println("Password matches...");
                    if((email.equalsIgnoreCase("admin@gmail.com") &&
pwd.equalsIgnoreCase("admin"))) {
                        return 2;
                    }
                    return 1;
                }
                else {
                    return 0; // invalid password
                }
            }
            else {
                return -1; // invalid username
            }
        }
        catch(Exception e) {
            System.out.println(e);
            return -1;
        }
    }

    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
        String email = req.getParameter("email");
        String pwd = req.getParameter("password");
        System.out.println(email + ", " + pwd);

        try {
            int result = loginUser(email, pwd);
            if(result == -1) {
                System.out.println("Invalid Username");
                Cookie c = new Cookie("InvalidOrNot", "invalid");
                c.setMaxAge(2);
                res.addCookie(c);
                res.sendRedirect("index.html");
            }
            else if(result == 0) {
                System.out.println("Invalid Password");
                Cookie c = new Cookie("InvalidOrNot", "invalid");
                c.setMaxAge(2);
                res.addCookie(c);
                res.sendRedirect("index.html");
            }
            else if(result == 1) {
                System.out.println("Logging in...");
                Cookie c = new Cookie("InvalidOrNot", "valid");
```

```
                c.setMaxAge(2);
                res.addCookie(c);
                Cookie c1 = new Cookie("email", email);
                res.addCookie(c1);
                res.sendRedirect("home.html");
            }
            else if(result == 2) {
                System.out.println("Admin Logging in...");
                Cookie c = new Cookie("InvalidOrNot", "valid");
                c.setMaxAge(2);
                res.addCookie(c);
                Cookie c1 = new Cookie("email", email);
                res.addCookie(c1);
                res.sendRedirect("ahome.html");
            }
        }
        catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

SearchTrain.java:

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.sql.*;
import java.io.*;
import java.util.*;

public class SearchTrain extends HttpServlet {
    public String[][] getTrains(String src, String dest) throws Exception {
        String[][] trainlist = new String[10][6];

        String url = "jdbc:mysql://localhost:3306/rrs";
        String sqluser = "root";
        String sqlpwd = "Mysql@02";

        try {
            Class.forName("com.mysql.jdbc.Driver");

            System.out.println("Hey I am after forname method in getTrains() -
SearchTrain class");

            Connection con = DriverManager.getConnection(url, sqluser,
sqlpwd);

            Statement st = con.createStatement();
```

```
            System.out.println("Hey I am after create method in getTrains() -
SearchTrain class");

            String s = String.format("select * from train where source = '%s'
and destination = '%s'", src, dest);
            ResultSet rs = st.executeQuery(s);
            int i = 0;
            while(rs.next()) {
                trainlist[i][0] = Integer.toString(rs.getInt(1));
                trainlist[i][1] = rs.getString(2);
                trainlist[i][2] = rs.getString(3);
                trainlist[i][3] = rs.getString(4);
                trainlist[i][4] = Integer.toString(rs.getInt(5));
                trainlist[i][5] = Integer.toString(rs.getInt(6));
                System.out.println(trainlist[i][0] + ", " + trainlist[i][1] +
", " + trainlist[i][2] + ", " + trainlist[i][3] + ", " + trainlist[i][4] + ",
" + trainlist[i][5]);
                i++;
            }
            return trainlist;
        }
        catch(Exception e) {
            System.out.println(e);
            return null;
        }
    }
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
        String source = req.getParameter("source");
        String destination = req.getParameter("destination");
        String dot = req.getParameter("dot");
        System.out.println(source + ", " + destination + ", " + dot);
        try {
            res.setContentType("text/html");
            PrintWriter pw = res.getWriter();

            // String[][] trlist = getTrains(source, destination, dot);
            String[][] trlist = getTrains(source, destination);
            for(int i = 0; i < trlist.length; i++) {
                for(int j = 0; j < 6; j++) {
                    System.out.println(trlist[i][j]);
                }
            }
            String trlistcontent = new String();
            // pw.println("testing");
            trlistcontent = "<table id='trainlist' border='2' cellspacing='3'
cellpadding='8' bordercolor='Blue' align='center' style='align-items:
```

```
center;'><tr><th>Train Name</th><th>Train Number</th><th>Price</th><th>Date of
Travel</th><th>Source</th><th>Destination</th><th>Available
Seats</th><th>Duration</th><th>Book Now</th></tr> ";
            for(int i = 0; i < trlist.length; i++) {
                if(trlist[i][0] != null) {
                    trlistcontent += "<tr>";
                    trlistcontent += "<td id='tnoin'>" + trlist[i][0] +
"</td>";
                    trlistcontent += "<td id='tnamein'>"+ trlist[i][1]
+"</td>";
                    trlistcontent += "<td id='pricein'>540</td>";
                    trlistcontent += "<td id='dotin'>" + dot + "</td>";
                    trlistcontent += "<td id='srcin'>"+ trlist[i][2] +"</td>";
                    trlistcontent += "<td id='desin'>"+ trlist[i][3]
+"</td>";
                    trlistcontent += "<td id='seatsin'>"+ trlist[i][4]
+"</td>";
                    trlistcontent += "<td id='durin'>"+ trlist[i][5] +"</td>";
                    trlistcontent += "<td><button class='open-button'
onclick='booknow(document.getElementById(\"tnamein\").innerHTML,
document.getElementById(\"tnoin\").innerHTML,
document.getElementById(\"pricein\").innerHTML,
document.getElementById(\"dotin\").innerHTML,
document.getElementById(\"srcin\").innerHTML,
document.getElementById(\"desin\").innerHTML,
document.getElementById(\"seatsin\").innerHTML,
document.getElementById(\"durin\").innerHTML);'>Book Ticket</button></td>";
                    trlistcontent += "</tr>";
                }
            }
            trlistcontent += "</table>";
            System.out.println(trlistcontent);
            pw.println(trlistcontent);
            pw.close();
        }
        catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

SearchTrain1.java:

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.sql.*;
import java.io.*;
import java.util.*;
```

```
public class SearchTrain1 extends HttpServlet {
    public String[][] getTrains(String src, String dest) throws Exception {
        String[][] trainlist = new String[10][6];

        String url = "jdbc:mysql://localhost:3306/rrs";
        String sqluser = "root";
        String sqlpwd = "Mysql@02";

        try {
            Class.forName("com.mysql.jdbc.Driver");

            System.out.println("Hey I am after forname method in getTrains() -
SearchTrain1 class");

            Connection con = DriverManager.getConnection(url, sqluser,
sqlpwd);

            Statement st = con.createStatement();

            System.out.println("Hey I am after create method in getTrains() -
SearchTrain1 class");

            String s = String.format("select * from train where source = '%s'
and destination = '%s'", src, dest);
            ResultSet rs = st.executeQuery(s);
            int i = 0;
            while(rs.next()) {
                trainlist[i][0] = Integer.toString(rs.getInt(1));
                trainlist[i][1] = rs.getString(2);
                trainlist[i][2] = rs.getString(3);
                trainlist[i][3] = rs.getString(4);
                trainlist[i][4] = Integer.toString(rs.getInt(5));
                trainlist[i][5] = Integer.toString(rs.getInt(6));
                System.out.println(trainlist[i][0] + ", " + trainlist[i][1] +
", " + trainlist[i][2] + ", " + trainlist[i][3] + ", " + trainlist[i][4] + ",
" + trainlist[i][5]);
                i++;
            }
            return trainlist;
        }
        catch(Exception e) {
            System.out.println(e);
            return null;
        }
    }
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
```

```
        String source = req.getParameter("source");
        String destination = req.getParameter("destination");
        // String dot = req.getParameter("dot");
        System.out.println(source + ", " + destination);
        try {
            res.setContentType("text/html");
            PrintWriter pw = res.getWriter();

            // String[][] trlist = getTrains(source, destination, dot);
            String[][] trlist = getTrains(source, destination);
            for(int i = 0; i < trlist.length; i++) {
                for(int j = 0; j < 6; j++) {
                    System.out.println(trlist[i][j]);
                }
            }
            String trlistcontent = new String();
            // pw.println("testing");
            trlistcontent = "<table id='trainlist' border='2' cellspacing='3'
cellpadding='8' bordercolor='Blue' align='center'><tr><th>Train
Name</th><th>Train
Number</th><th>Price</th><th>Source</th><th>Destination</th><th>Available
Seats</th><th>Duration</th></tr>";
            for(int i = 0; i < trlist.length; i++) {
                if(trlist[i][0] != null) {
                    trlistcontent += "<tr>";
                    trlistcontent += "<td id='tnoin'>" + trlist[i][0] +
"</td>";
                    trlistcontent += "<td id='tnamein'>"+ trlist[i][1]
+"</td>";
                    trlistcontent += "<td id='pricein'>540</td>";
                    trlistcontent += "<td id='srcin'>"+ trlist[i][2] +"</td>";
                    trlistcontent += "<td id='desin'>"+ trlist[i][3]
+"</td>";
                    trlistcontent += "<td id='seatsin'>"+ trlist[i][4]
+"</td>";
                    trlistcontent += "<td id='durin'>"+ trlist[i][5] +"</td>";
                    trlistcontent += "</tr>";
                }
            }
            trlistcontent += "</table>";
            System.out.println(trlistcontent);
            pw.println(trlistcontent);
            pw.close();
        }
        catch(Exception e) {
            System.out.println(e);
        }
    } }
```

Signup.java:

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.io.*;
import java.sql.*;
import java.util.*;



public class Signup extends HttpServlet {
    public int addPassenger(String name, String email, String pwd, char
gender, String city, String contact) throws Exception{
        String url = "jdbc:mysql://localhost:3306/rrs";
        String sqluser = "root";
        String sqlpwd = "Mysql@02";

        Class.forName("com.mysql.jdbc.Driver");

        System.out.println("Hey I am after forname method in addUser() -
Signup class");

        Connection con = DriverManager.getConnection(url, sqluser, sqlpwd);

        Statement st = con.createStatement();

        System.out.println("Hey I am after create method in addUser() - Signup
class");

        try {
            String s = String.format("select count(*) from passenger where
email = '%s'", email);
            ResultSet rs = st.executeQuery(s);
            rs.next();
            int count = rs.getInt(1);
            if(count > 0) {
                return 0;
            }
            s = String.format("insert into passenger(email, pwd, name, gender,
city, contact) values('%s','%s','%s','%s','%s','%s')", email, pwd, name,
gender, city, contact);
            st.executeUpdate(s);
            System.out.println("Passenger record inserted...");
        }
        catch(Exception e) {
            System.out.println(e);
            return -1;
        }
```

```
        return 1;
    }

    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
        String name = req.getParameter("fullname");
        String email = req.getParameter("email");
        String pwd = req.getParameter("password");
        String gender = req.getParameter("gender");
        String city = req.getParameter("city");
        String contact = req.getParameter("contact");
        System.out.println(name + ", " + email + ", " + pwd + ", " + gender +
", " + city + ", " + contact);

        try {
            int result = addPassenger(name, email, pwd, gender.charAt(0),
city, contact);
            if(result == 0) {
                System.out.println("Username already exists...");
                Cookie c = new Cookie("InvalidSignup", "invalid");
                c.setMaxAge(2);
                res.addCookie(c);
                res.sendRedirect("signup.html");
            }
            else if(result == 1) {
                System.out.println("Passenger Signed Up");
                Cookie c = new Cookie("InvalidSignup", "valid");
                c.setMaxAge(2);
                res.addCookie(c);
                Cookie c1 = new Cookie("email", email);
                res.addCookie(c1);
                res.sendRedirect("home.html");
            }
        }
        catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

WaitingList.java:

```
import javax.servlet.http.*;
import javax.servlet.*;
import java.sql.*;
import java.io.*;
import java.lang.reflect.Executable;
import java.util.*;
```

```
public class WaitingList extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException {
        String url = "jdbc:mysql://localhost:3306/rrs";
        String sqluser = "root";
        String sqlpwd = "Mysql@02";

        try {
            Class.forName("com.mysql.cj.jdbc.Driver");

            System.out.println("Hey I am after forname method in get() -
WaitingList class");

            Connection con = DriverManager.getConnection(url, sqluser,
sqlpwd);

            Statement st = con.createStatement();

            System.out.println("Hey I am after create method in get() -
WaitingList class");

            res.setContentType("text/html");
            PrintWriter pw = res.getWriter();

            String wlist = new String();

            String s = String.format("select * from ticket where
ticstatus='Waiting'");
            ResultSet rs = st.executeQuery(s);
            wlist = "<table border='2' cellspacing='3' cellpadding='30'
bordercolor='Blue' style='margin-left: 320px; margin-top:
30px;'><tr><th>Ticket No.</th><th>Passenger Name</th><th>Train
Number</th><th>Train Name</th><th>Source</th><th>Destination</th><th>Date of
Travel</th><th>No of Seats</th></tr>";
            while(rs.next()) {
                int ticno = rs.getInt(1);
                int pid = rs.getInt(2);
                int tid = rs.getInt(3);
                String dot = rs.getString(4);
                int asno = rs.getInt(5);
                int csno = rs.getInt(6);
                String status = rs.getString(7);
                System.out.println(">> WaitingList.java: Read columns");

                Statement st1 = con.createStatement();
```

```
                s = String.format("select name from passenger where pid='%d'",
pid);
                ResultSet rs1 = st1.executeQuery(s);
                rs1.next();
                String pname = rs1.getString(1);
                // System.out.println("pname: "+ pname);
                rs1.close();
                System.out.println(">> WaitingList.java: Passenger Query
Executed");

                Statement st2 = con.createStatement();
                s = String.format("select name,source,destination from train
where tid='%d'", tid);
                ResultSet rs2 = st2.executeQuery(s);
                rs2.next();
                System.out.println(">> WaitingList.java: Train Query
Executed");


                String record = new String();
                record += "<tr>";
                record += "<td>" + Integer.toString(ticno) + "</td>"; //ticket
no
                record += "<td>" + pname + "</td>"; // passenger name
                record += "<td>" + Integer.toString(tid) + "</td>"; // train
id
                record += "<td>" + rs2.getString(1) + "</td>"; // train name
                record += "<td>" + rs2.getString(2) + "</td>"; // src
                record += "<td>" + rs2.getString(3) + "</td>"; // dest
                record += "<td>" + dot + "</td>"; // date of travel
                record += "<td>" + Integer.toString(asno + csno) + "</td>"; //
no of seats
                record += "</tr>";
                rs2.close();
                wlist += record;
                System.out.println(">> WaitingList.java: Table Row
Generated");
            }
            wlist += "</table>";
            System.out.println(">> WaitingList.java: Table Generated");

            System.out.println(wlist);
            pw.println(wlist);
            pw.close();
        }
        catch(Exception e) {
            System.out.println(e);
        }
```

```
    }
}
```

web.xml:

```
<web-app>
    <servlet>
        <servlet-name>Login</servlet-name>
        <servlet-class>Login</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Login</servlet-name>
        <url-pattern>/Login</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>Signup</servlet-name>
        <servlet-class>Signup</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Signup</servlet-name>
        <url-pattern>/Signup</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>SearchTrain</servlet-name>
        <servlet-class>SearchTrain</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>SearchTrain</servlet-name>
        <url-pattern>/SearchTrain</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>BookTicket</servlet-name>
        <servlet-class>BookTicket</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>BookTicket</servlet-name>
        <url-pattern>/BookTicket</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>History</servlet-name>
        <servlet-class>History</servlet-class>
    </servlet>
```

```
<servlet-mapping>
    <servlet-name>History</servlet-name>
    <url-pattern>/History</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>GetStatus</servlet-name>
    <servlet-class>GetStatus</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>GetStatus</servlet-name>
    <url-pattern>/GetStatus</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>GetRecord</servlet-name>
    <servlet-class>GetRecord</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>GetRecord</servlet-name>
    <url-pattern>/GetRecord</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>Cancel</servlet-name>
    <servlet-class>Cancel</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>Cancel</servlet-name>
    <url-pattern>/Cancel</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>Display</servlet-name>
    <servlet-class>Display</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>Display</servlet-name>
    <url-pattern>/Display</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>AddTrain</servlet-name>
    <servlet-class>AddTrain</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>AddTrain</servlet-name>
    <url-pattern>/AddTrain</url-pattern>
```

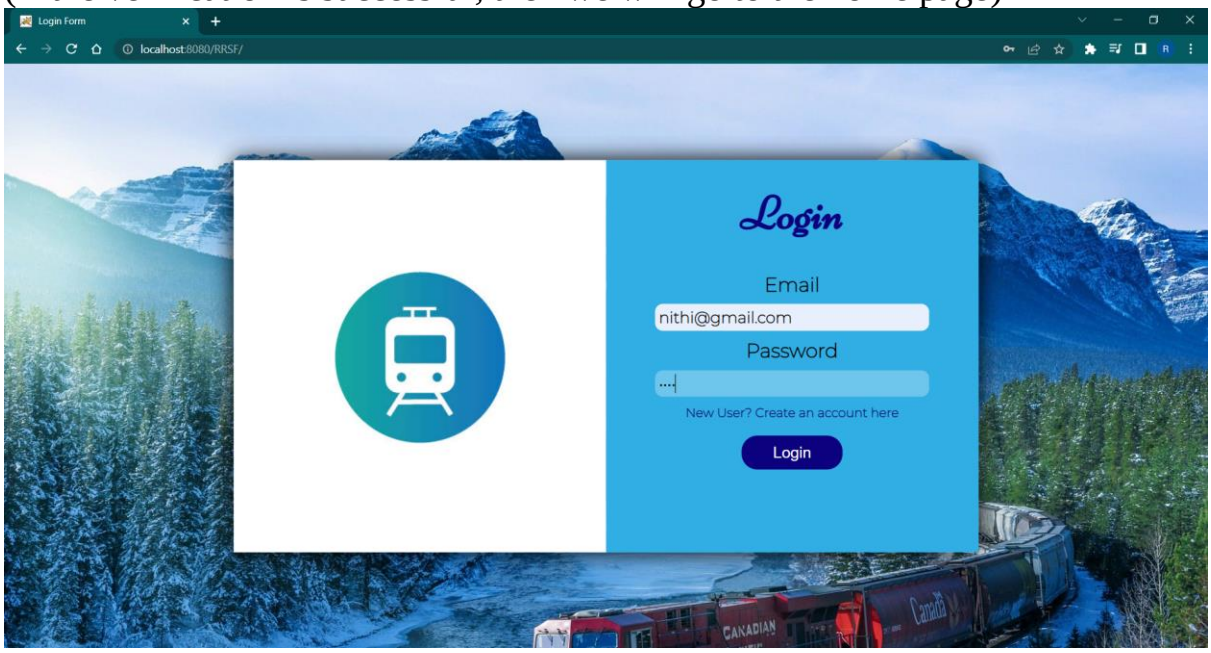
```
    </servlet-mapping>

    <servlet>
        <servlet-name>SearchTrain1</servlet-name>
        <servlet-class>SearchTrain1</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>SearchTrain1</servlet-name>
        <url-pattern>/SearchTrain1</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>SearchTrain2</servlet-name>
        <servlet-class>SearchTrain2</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>SearchTrain2</servlet-name>
        <url-pattern>/SearchTrain2</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>DeleteTrain</servlet-name>
        <servlet-class>DeleteTrain</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>DeleteTrain</servlet-name>
        <url-pattern>/DeleteTrain</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>WaitingList</servlet-name>
        <servlet-class>WaitingList</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>WaitingList</servlet-name>
        <url-pattern>/WaitingList</url-pattern>
    </servlet-mapping>

    <servlet>
        <servlet-name>Confirm</servlet-name>
        <servlet-class>Confirm</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Confirm</servlet-name>
        <url-pattern>/Confirm</url-pattern>
    </servlet-mapping>
</web-app
```

Output Snapshots

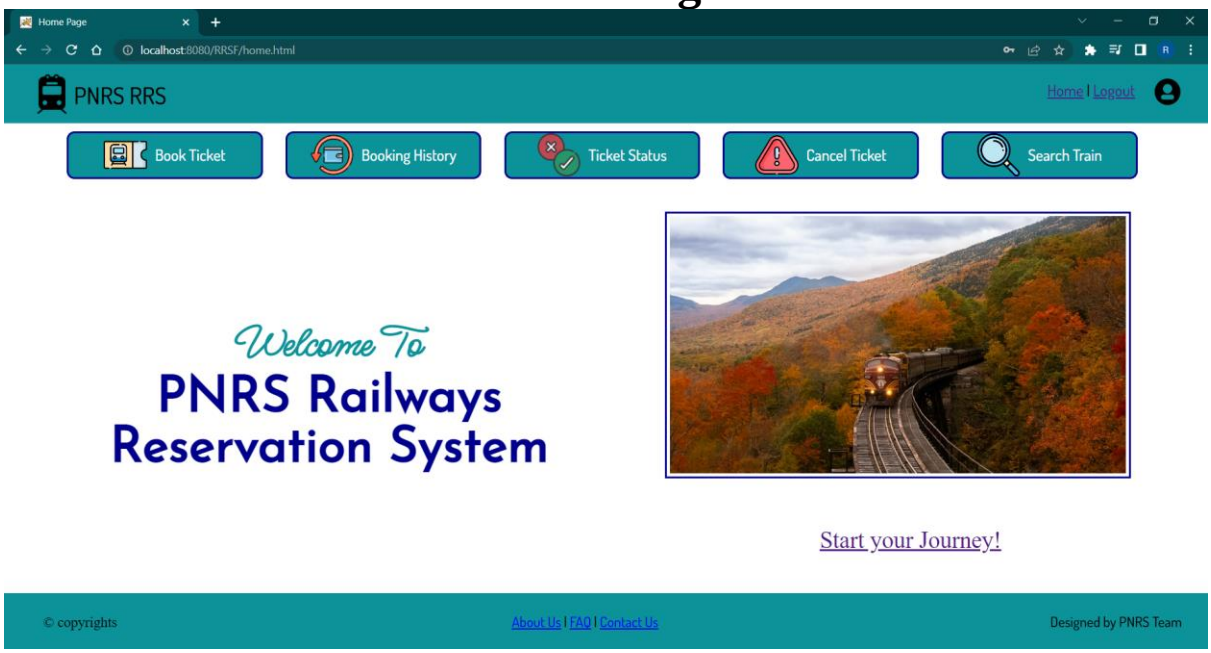
Customer Side:

Login Page

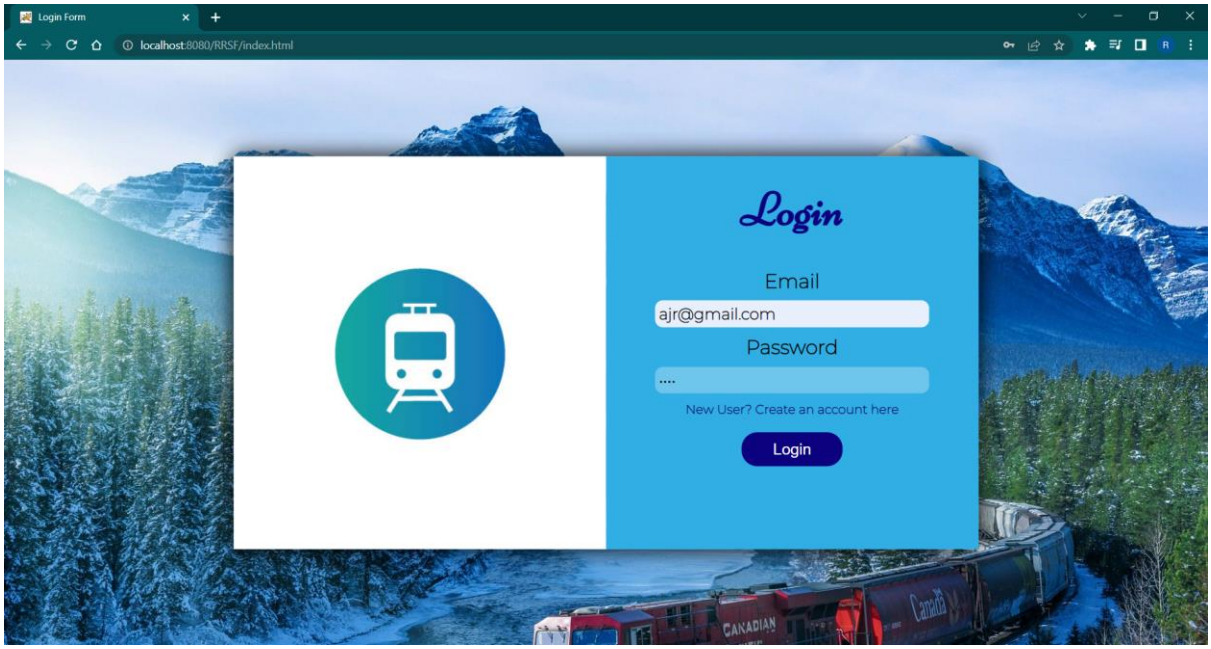
(If the verification is successful, then we will go to the home page)



Home Page

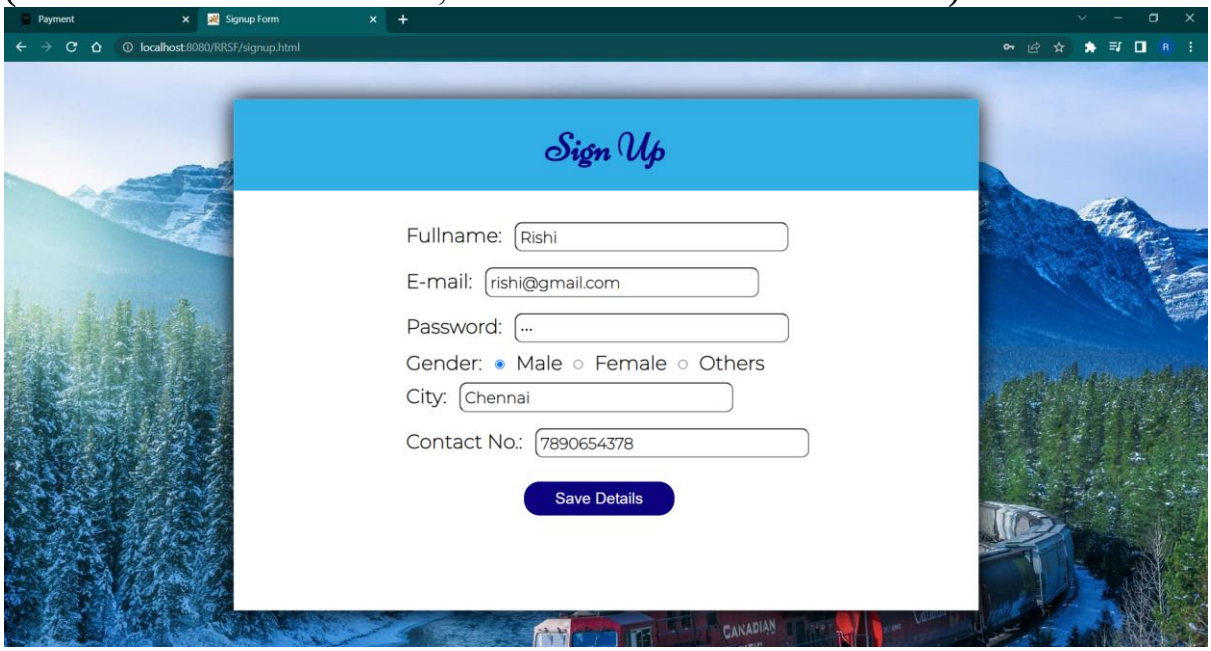


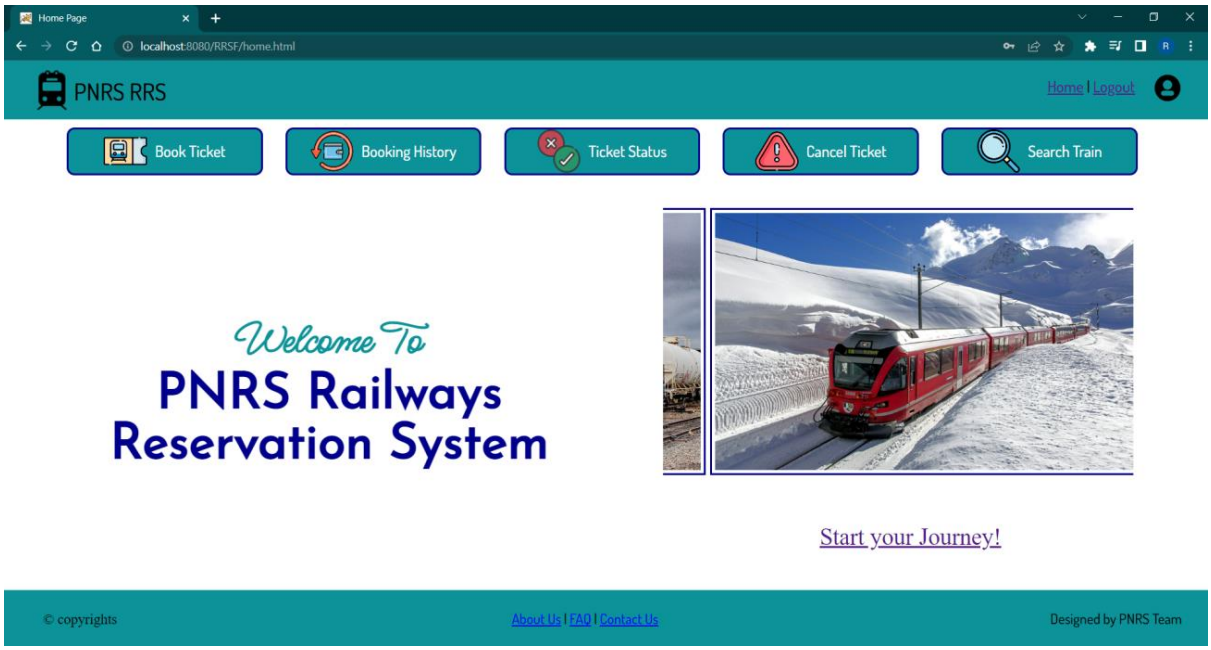
(If the verification is unsuccessful, then we will remain on the same page)



Signup Page

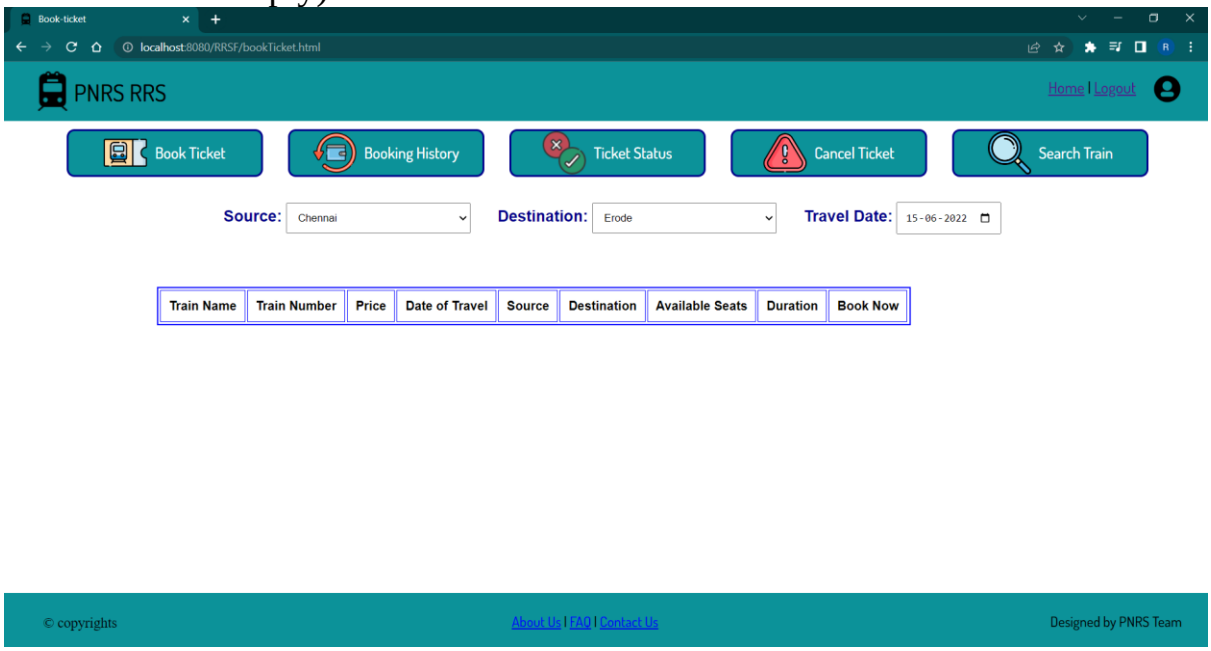
(If user doesn't have account, then he can create a new account)



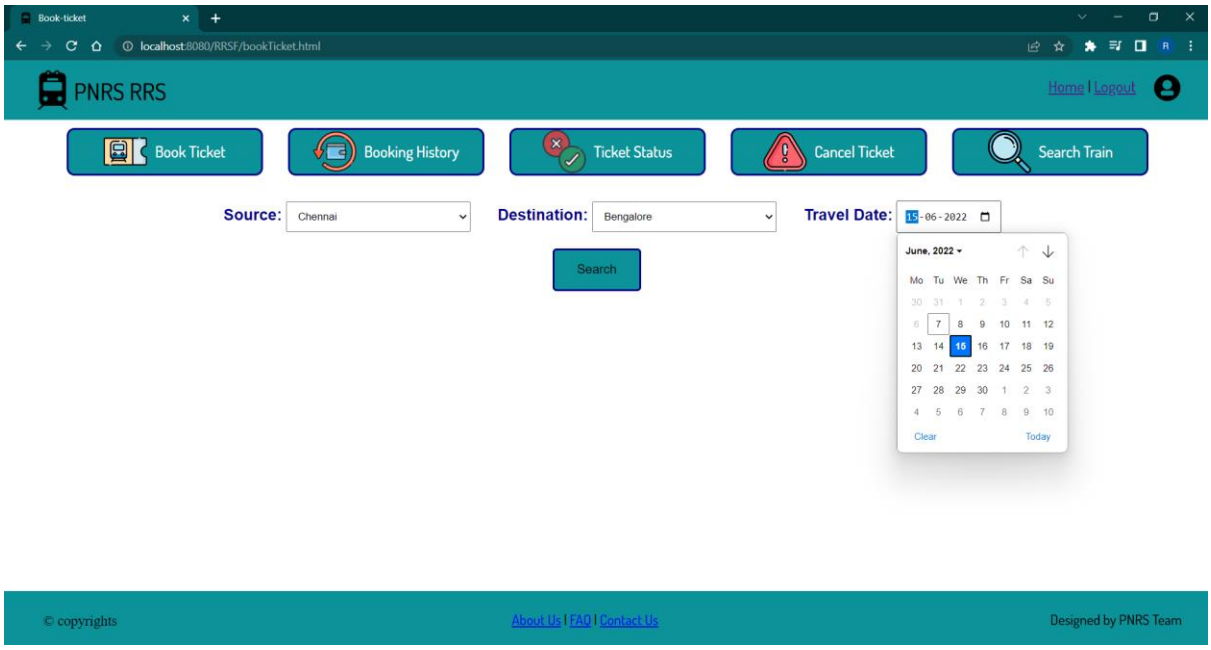


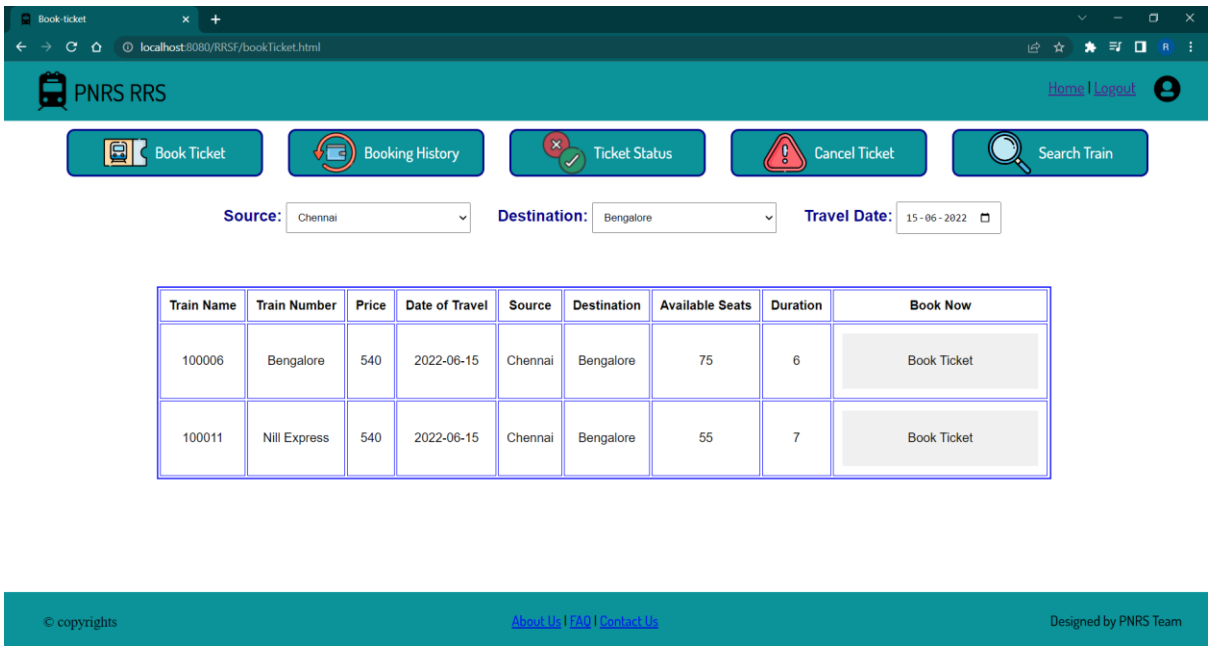
Book Ticket Page

(If the trains are not available between the source & destination, then the table will be shown empty)

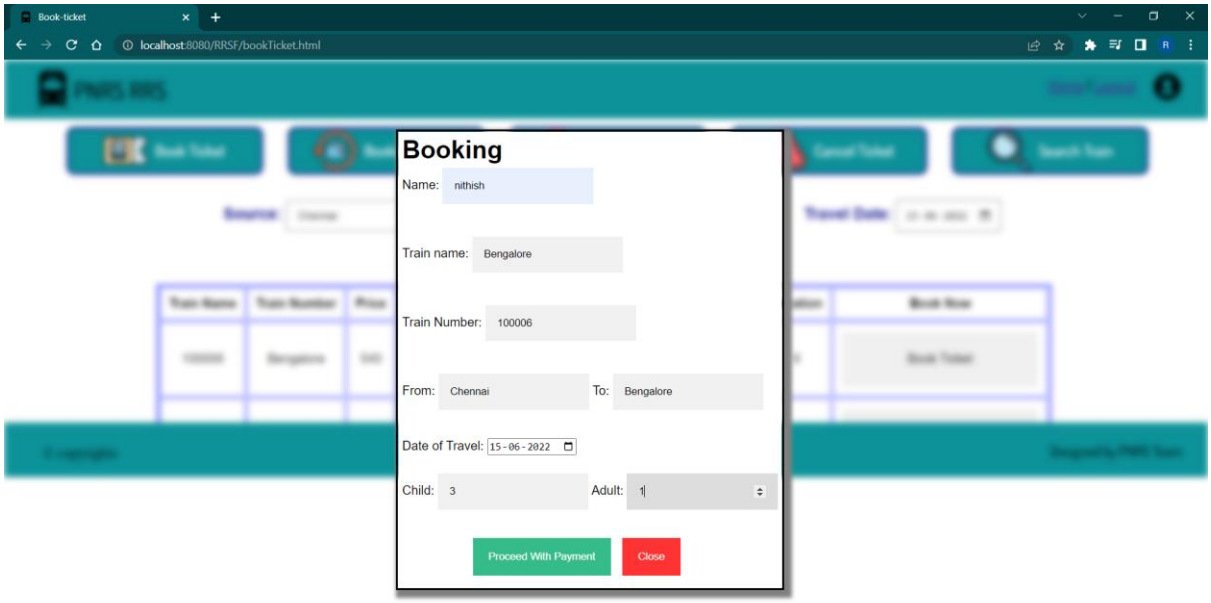


(If there are any trains available between the source & destination, then the trains will be displayed on the table. According to our preference we can choose one the trains from the listed trains, by clicking on Book Ticket button then we will move to next page)



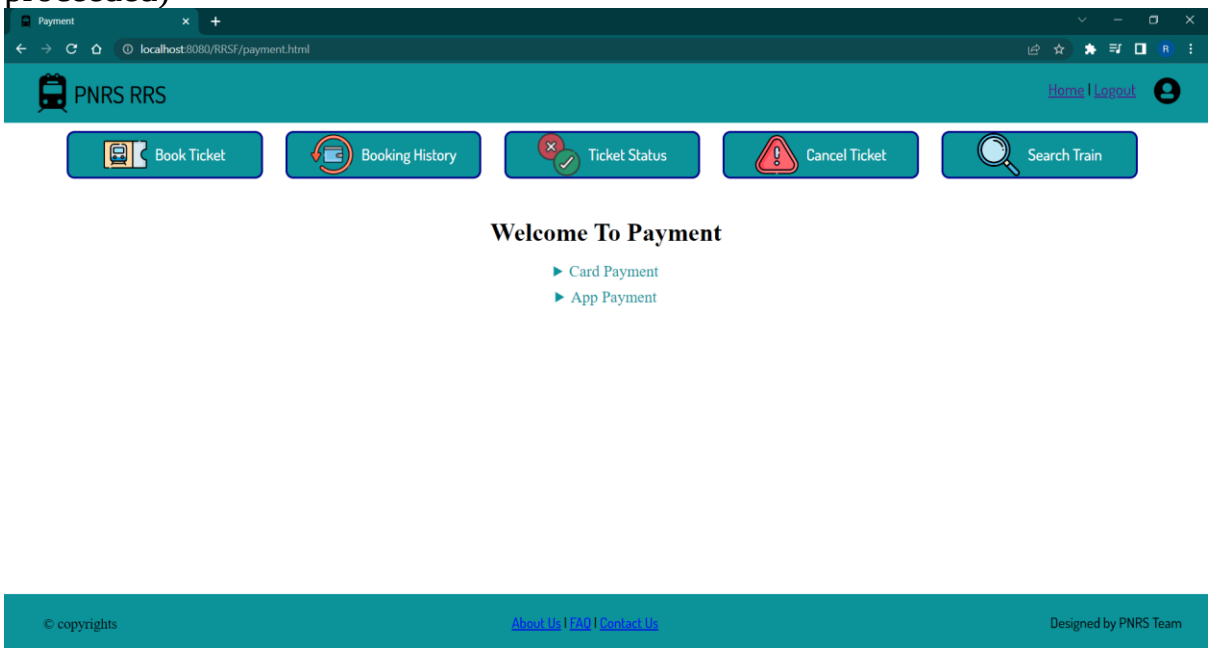


(For booking the tickets, passenger details will be entered here)

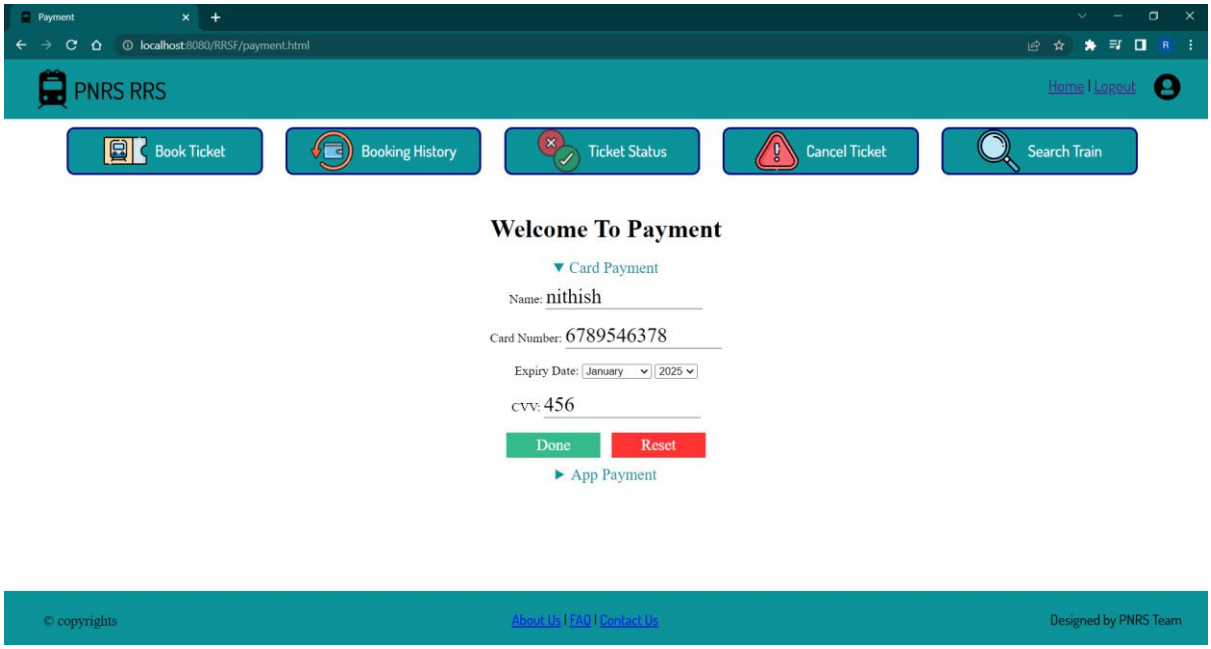


Payment Page

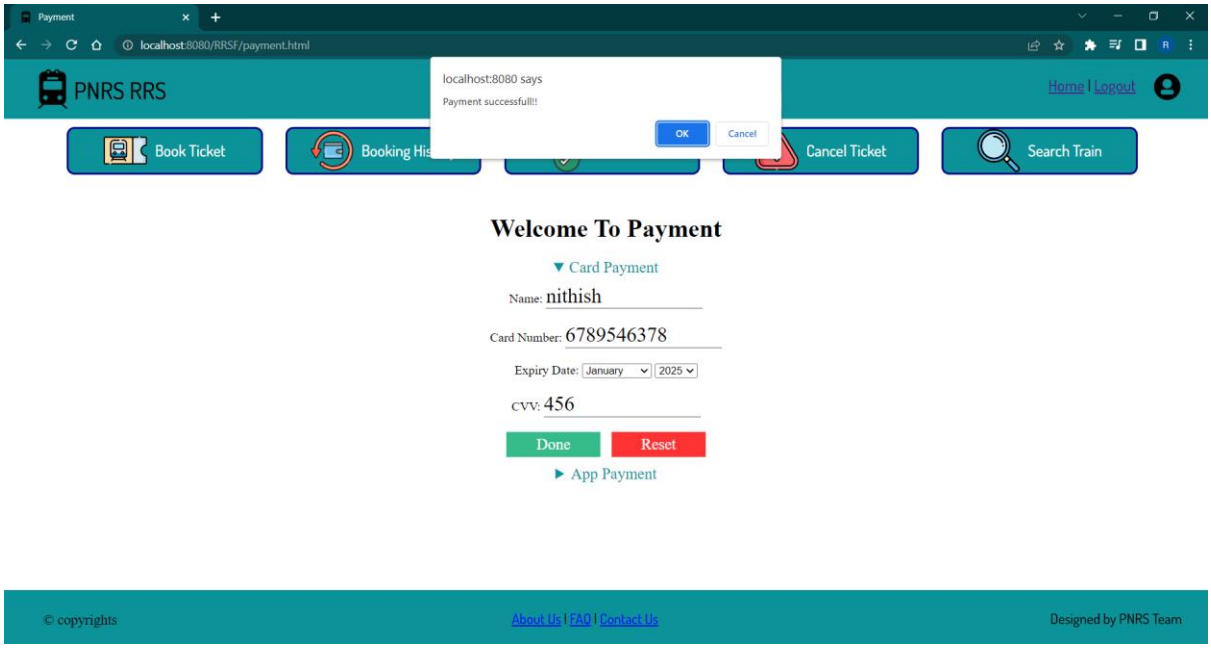
(Two options are available for the payment, anyone can be chosen, and it can be proceeded)



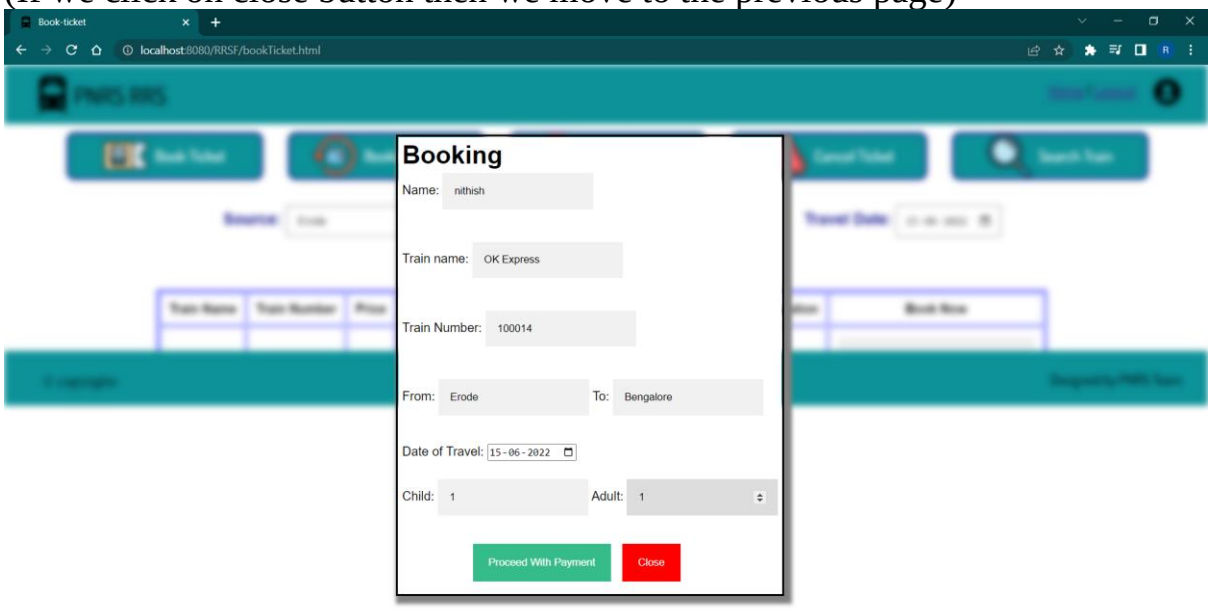
(Enter the card details of passenger, for the card payment method)

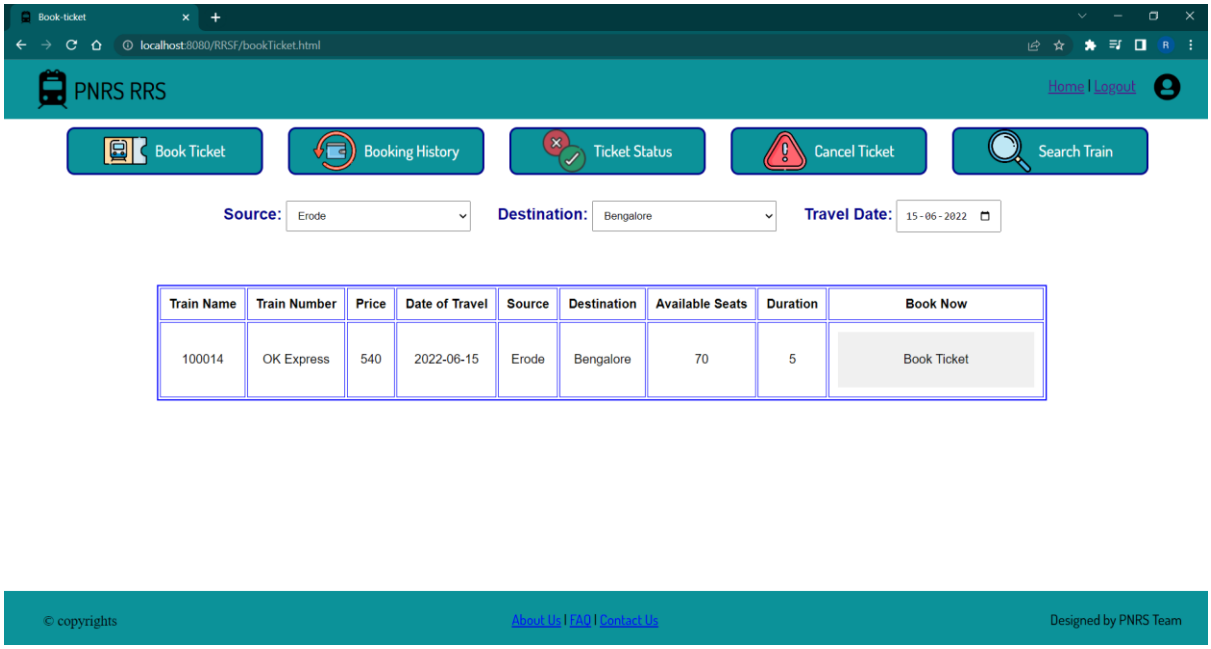


(Payment is successful)

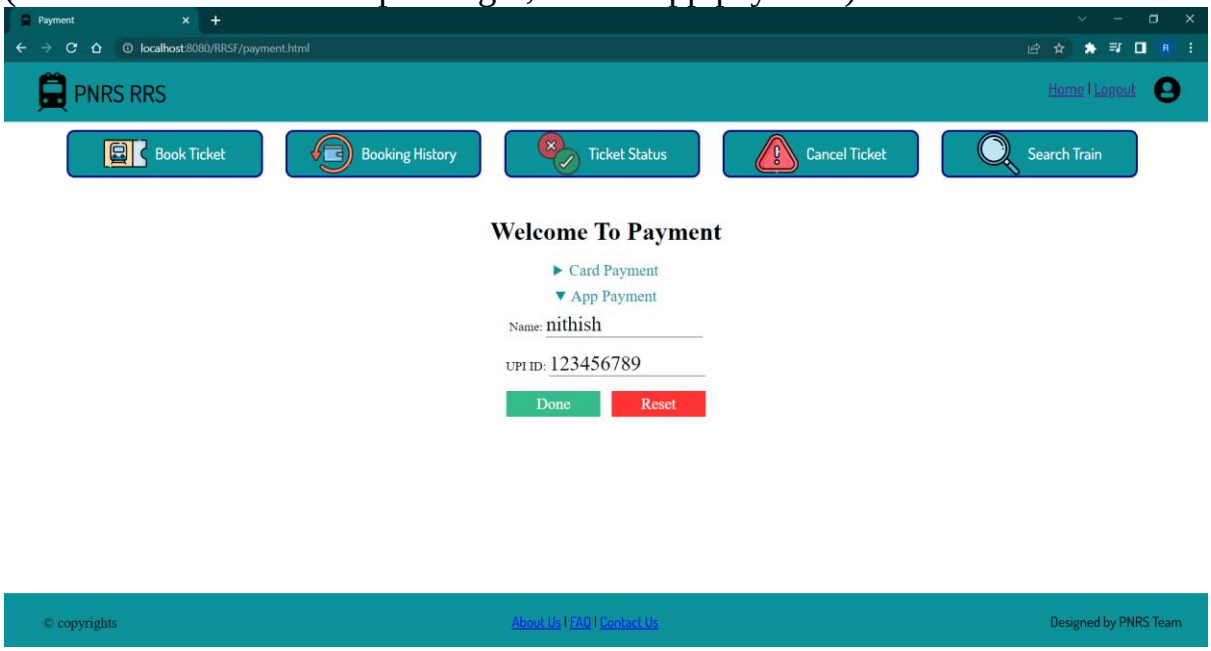


(If we click on close button then we move to the previous page)

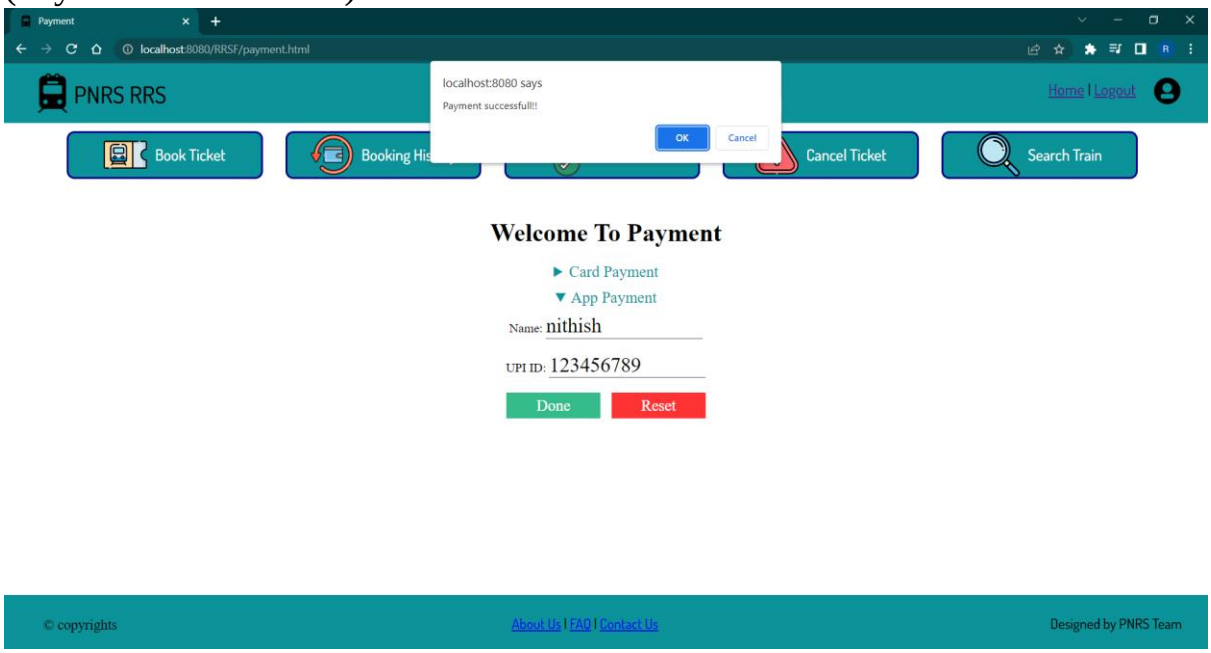




(Enter the UPI details of passenger, for the app payment)

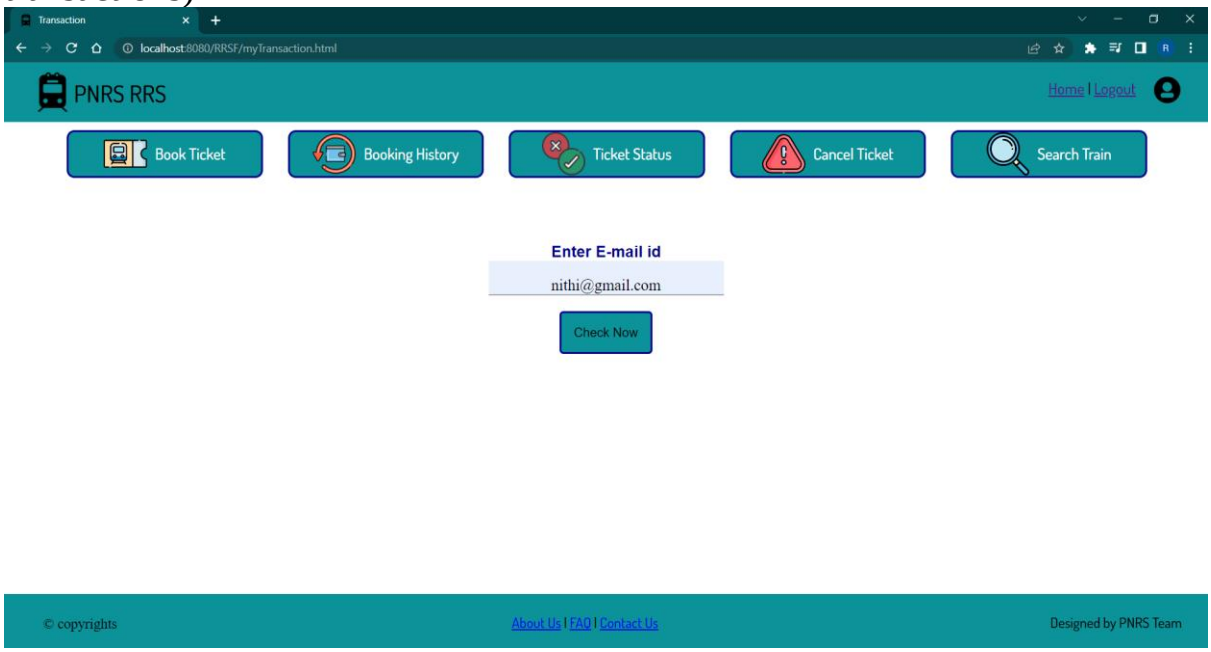


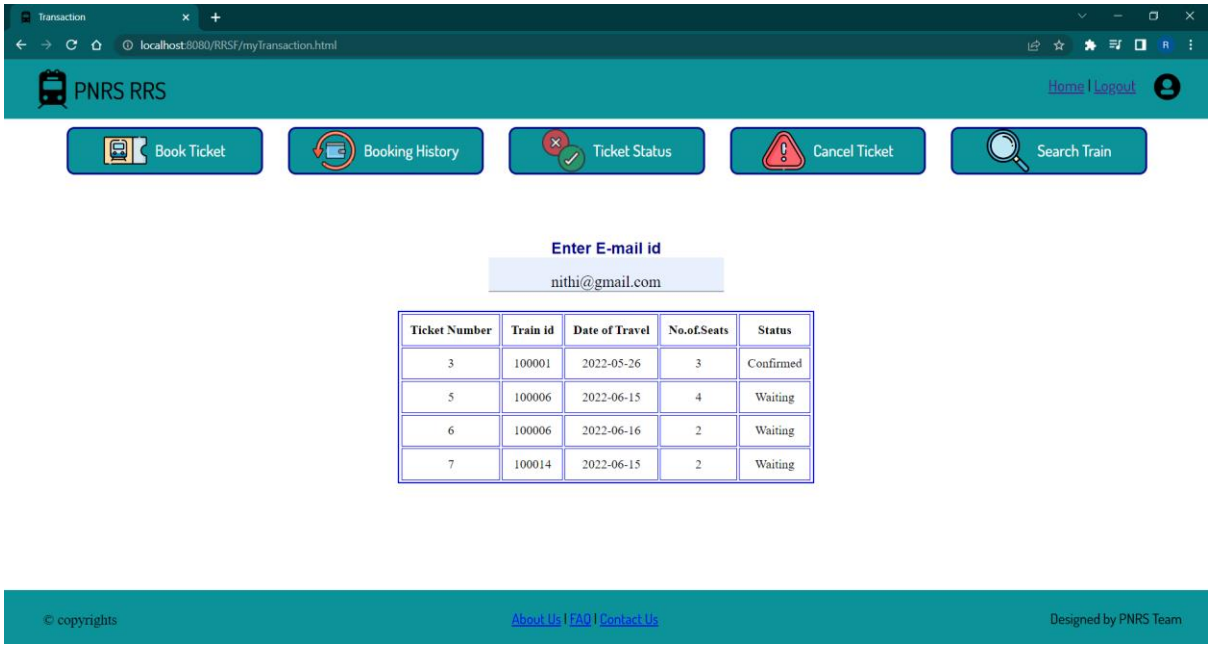
(Payment is successful)



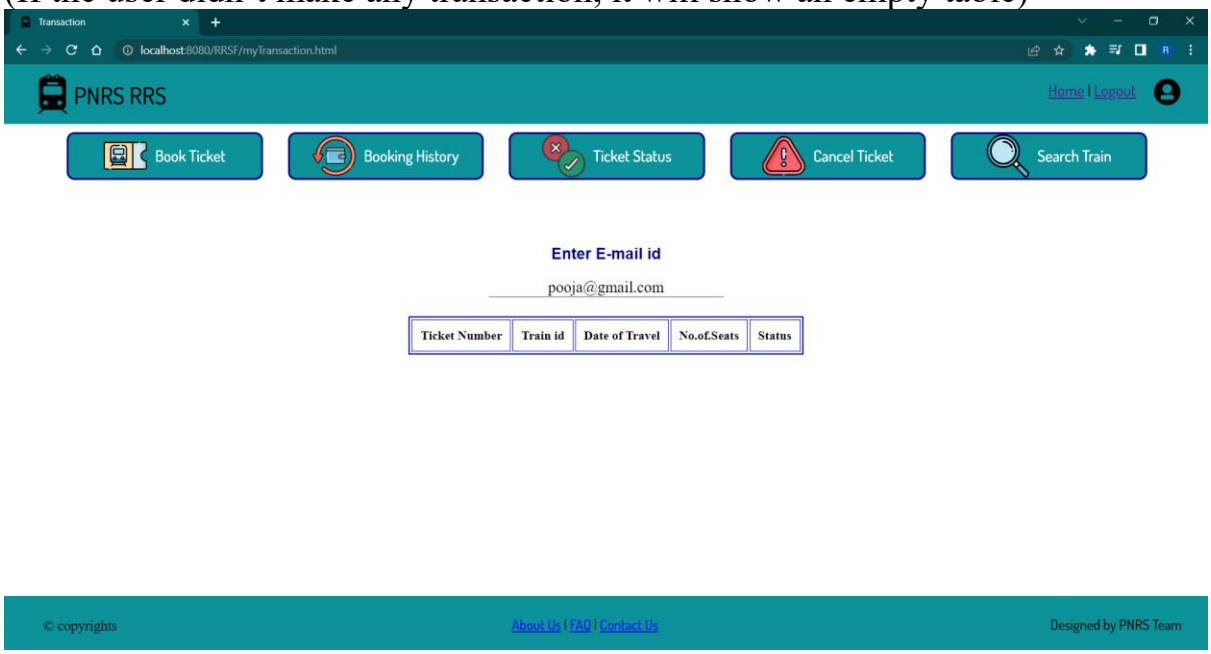
Booking History Page

(This page will display booking history of the passenger if he/she has made any transactions)



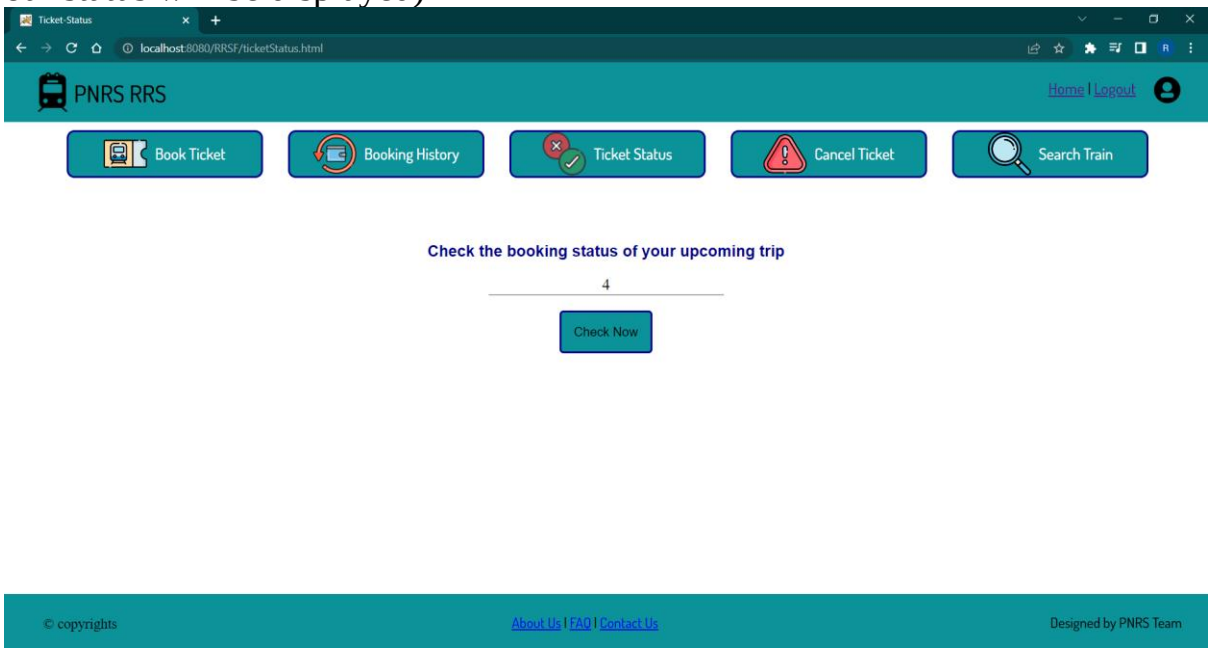


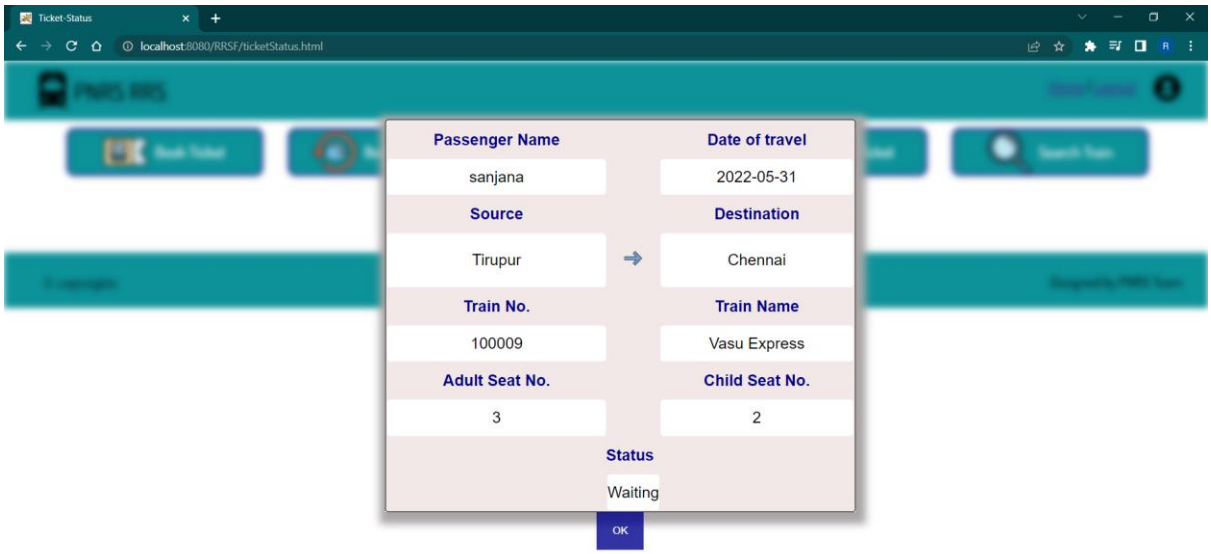
(If the user didn’t make any transaction, it will show an empty table)



Ticket Status Page

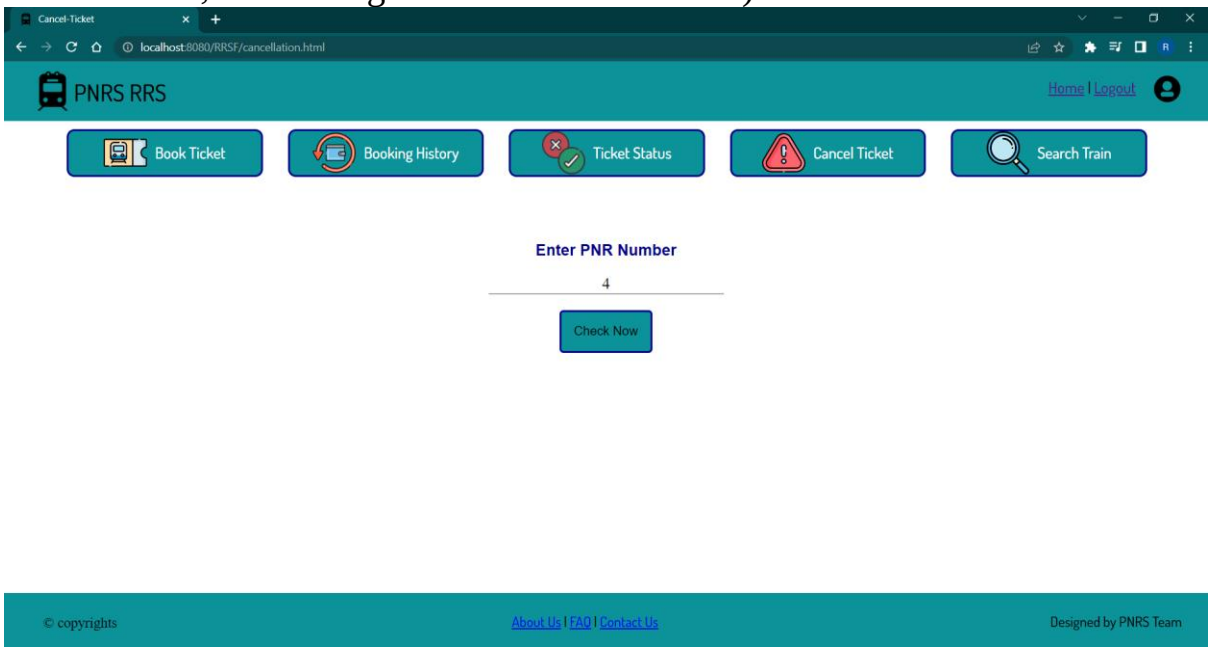
(This page will display status of our tickets, by clicking on check now button our status will be displayed)



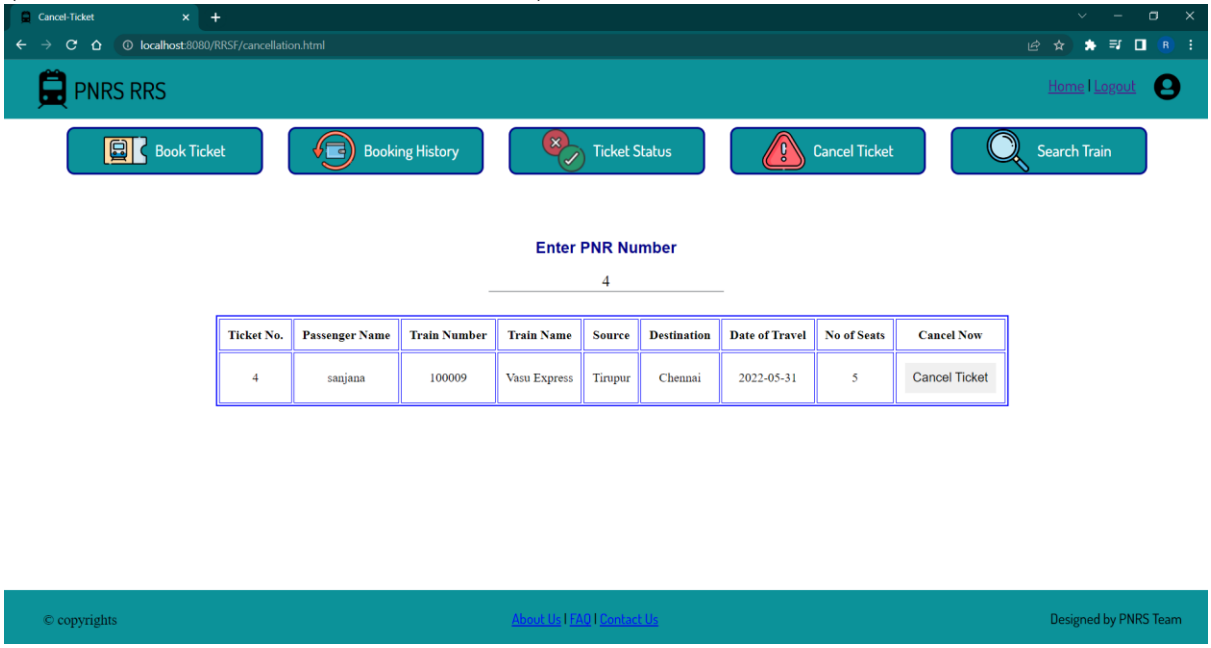


Cancel Ticket Page

(If the reserved ticket is wanted to be cancelled, it can be done by entering the PNR number, on clicking the cancel ticket button)



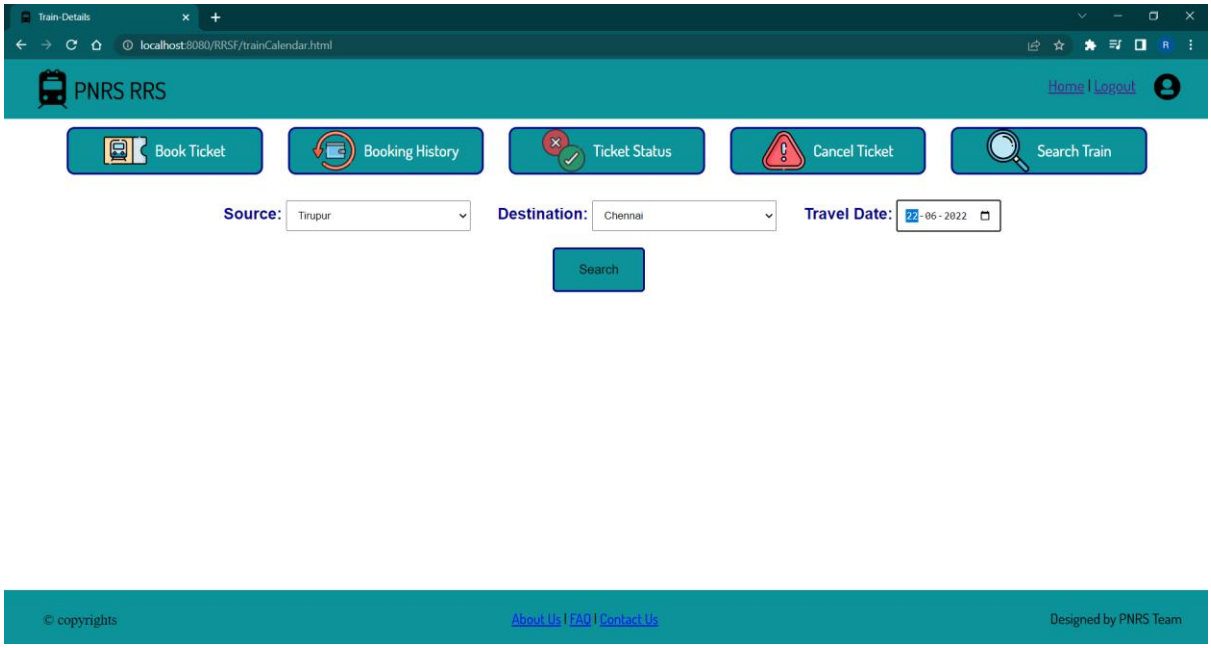
(Click the cancel button in the table)



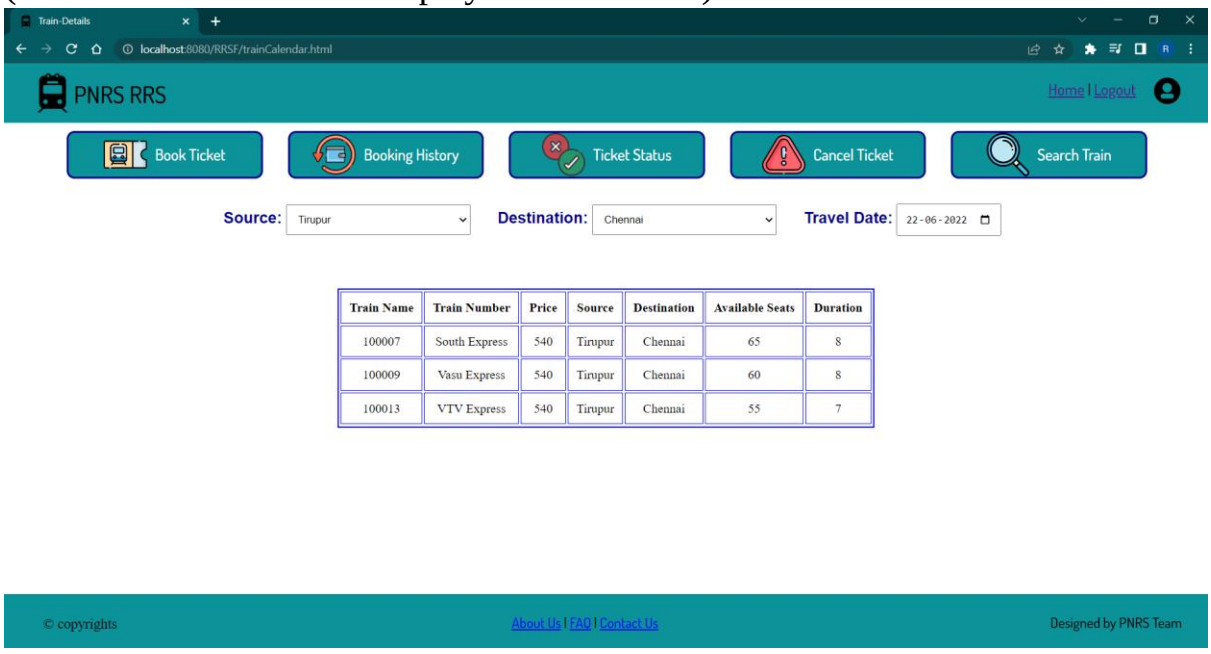
(Ticket has been cancelled successfully)

Search Train Page

(A particular train can be searched by clicking on the search train button)



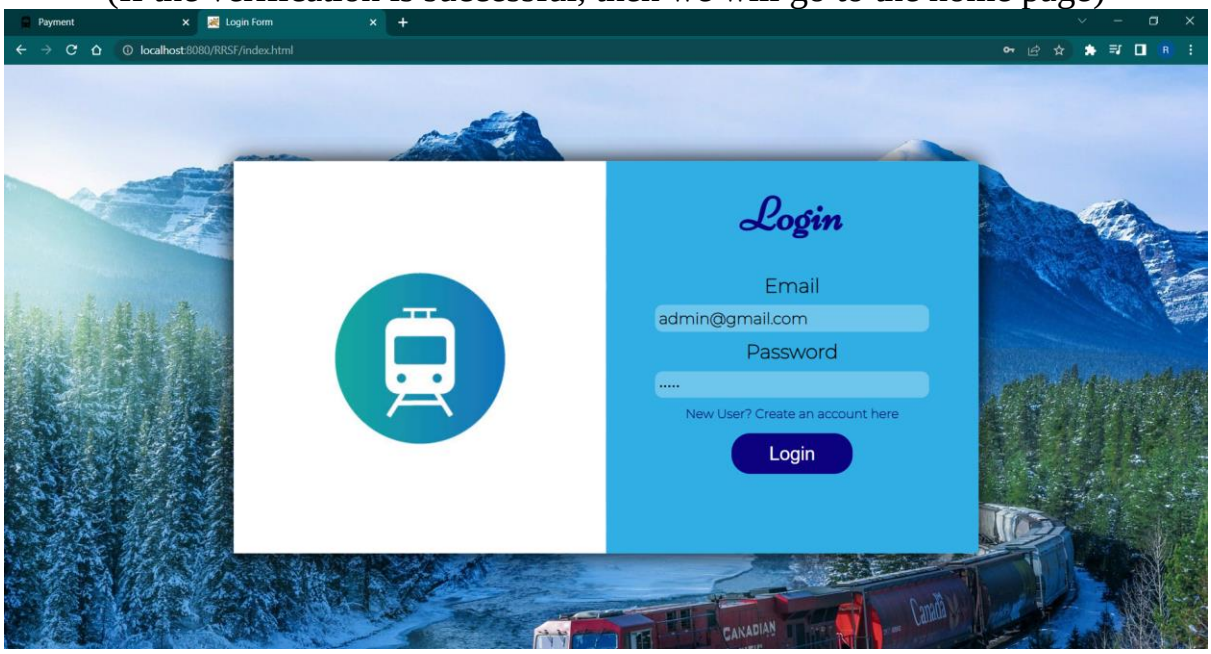
(The available trains are displayed on the table)



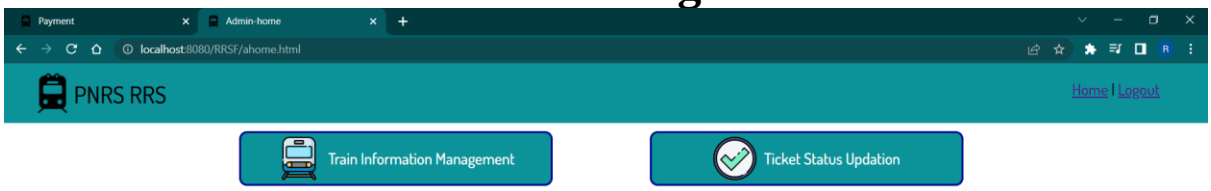
Admin Side:

Login Page

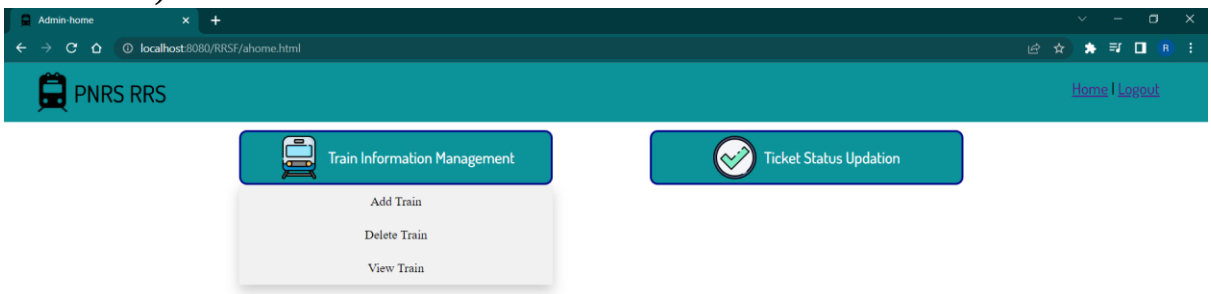
(If the verification is successful, then we will go to the home page)



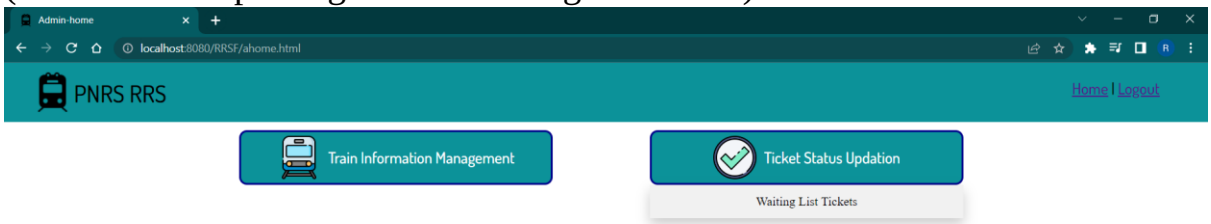
Home Page



(Train information management has options like add train, delete train, and view train)

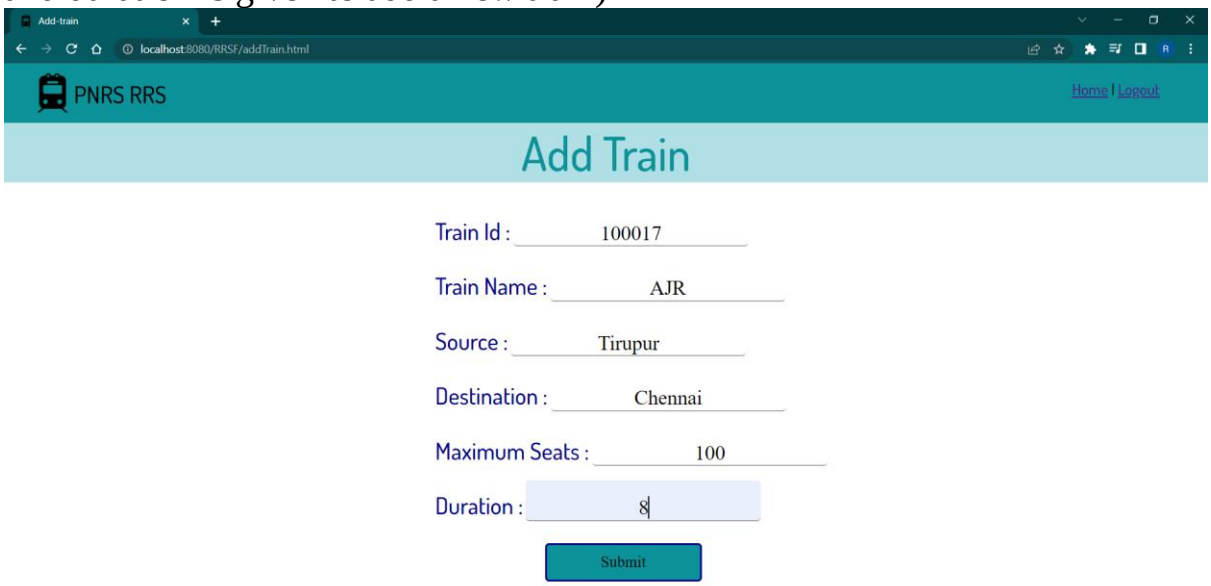


(Ticket status updating has the waiting list tickets)

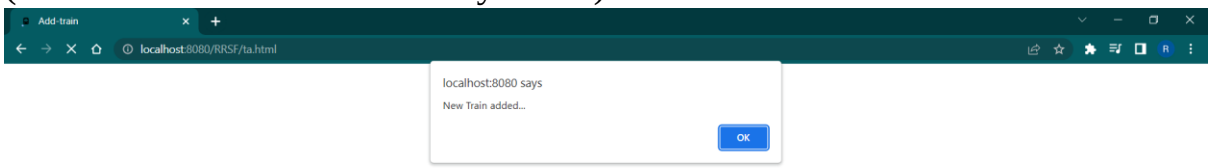


Add Train Page

(The train details like train id, train name, source, destination, maximum seats and duration is given to add a new train)



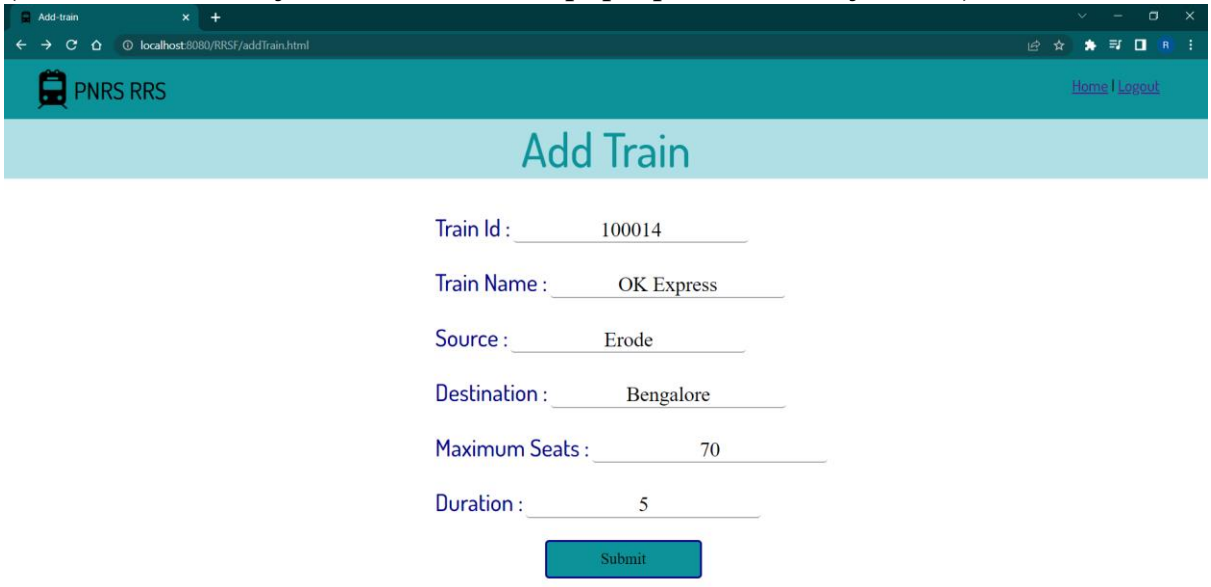
(New train has been successfully added)

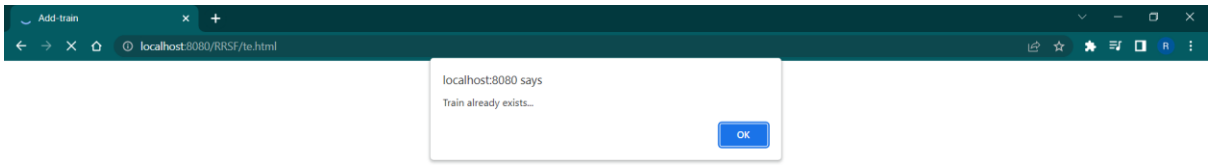


(New train has been inserted in the database)

```
mysql>
mysql> select * from train;
+--------+------------------+-----------+-------------+-----------+----------+
| tid    | name             | source    | destination | max_seats | duration |
+--------+------------------+-----------+-------------+-----------+----------+
| 100001 | Shari Express    | Chennai   | Madurai     |        78 |        6 |
| 100002 | MM Express       | Chennai   | Madurai     |        89 |        6 |
| 100003 | Window Express   | Coimbatore| Madurai     |        80 |        5 |
| 100004 | Chennai Express  | Chennai   | Coimbatore  |        95 |        7 |
| 100005 | Superfast Express| Erode     | Madurai     |        70 |        4 |
| 100006 | Bengalore        | Chennai   | Bengalore   |        75 |        6 |
| 100007 | South Express    | Tirupur   | Chennai     |        65 |        8 |
| 100008 | Natural Express  | Madurai   | Chennai     |        40 |        8 |
| 100009 | Vasu Express     | Tirupur   | Chennai     |        60 |        8 |
| 100010 | SPR Express      | Bengalore | Coimbatore  |        45 |        5 |
| 100011 | Nill Express     | Chennai   | Bengalore   |        55 |        7 |
| 100012 | Magical Express  | Erode     | Madurai     |        50 |        4 |
| 100013 | VTV Express      | Tirupur   | Chennai     |        55 |        7 |
| 100014 | OK Express       | Erode     | Bengalore   |        70 |        5 |
| 100017 | AJR              | Tirupur   | Chennai     |       100 |        8 |
+--------+------------------+-----------+-------------+-----------+----------+
15 rows in set (0.00 sec)

mysql>
```

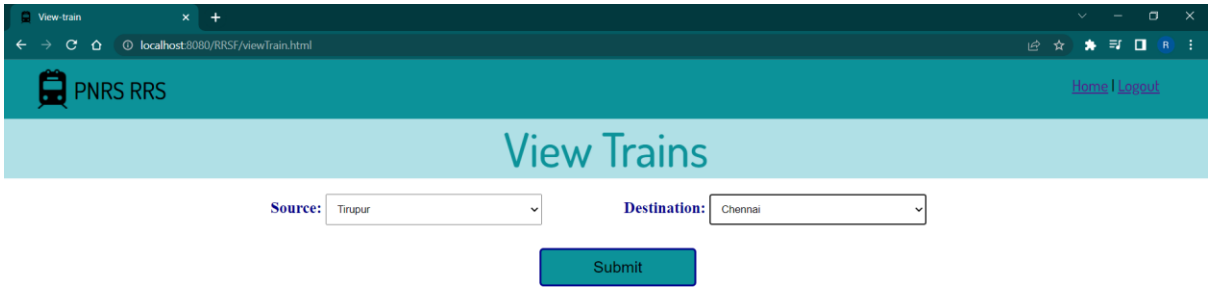
(If the train already exists, it shows a pop-up train already exists)

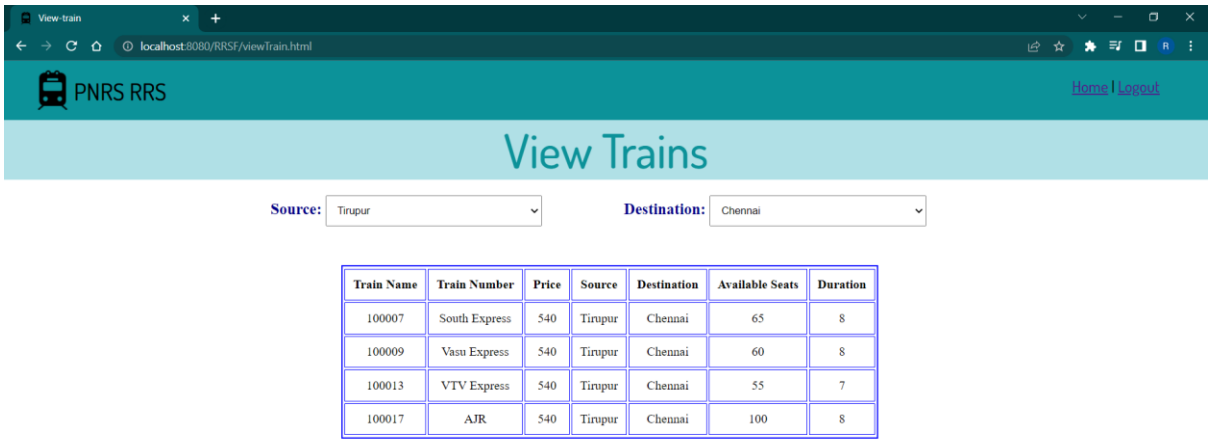




View Train Page

(To view the available trains the source and destination details are given on clicking the submit button it displays the available trains)

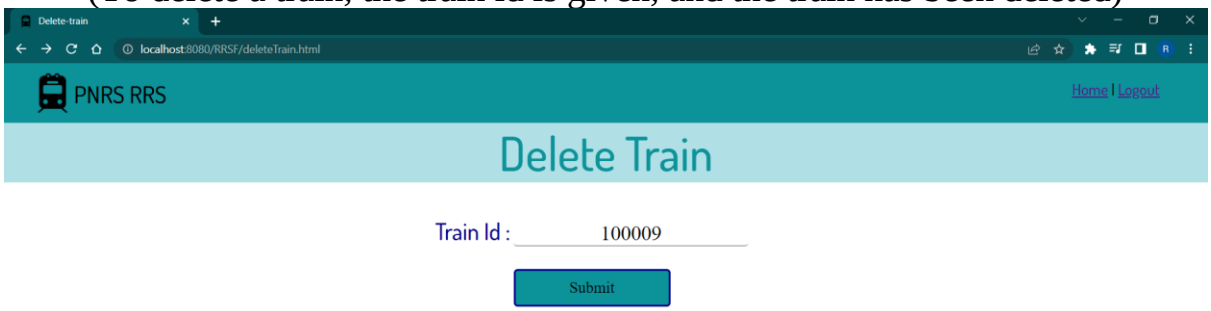


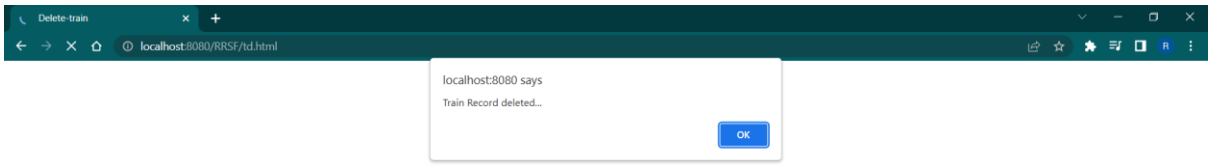


Train Name	Train Number	Price	Source	Destination	Available Seats	Duration
100007	South Express	540	Tirupur	Chennai	65	8
100009	Vasu Express	540	Tirupur	Chennai	60	8
100013	VTV Express	540	Tirupur	Chennai	55	7
100017	AJR	540	Tirupur	Chennai	100	8

Delete Train Page

(To delete a train, the train id is given, and the train has been deleted)

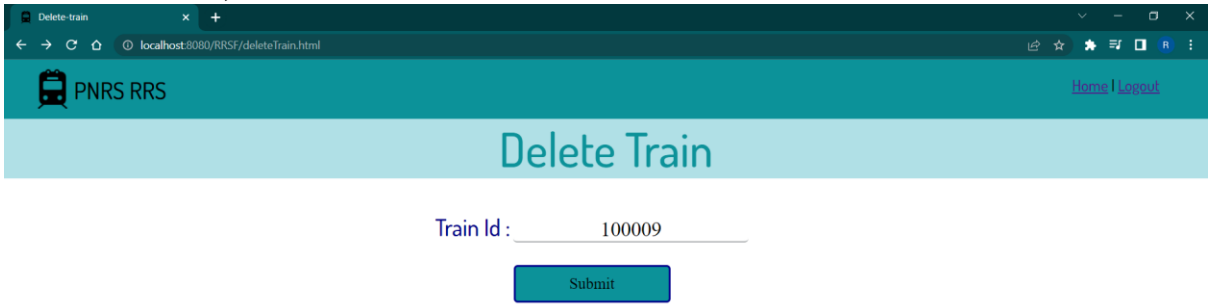




(The train has been removed from the database)

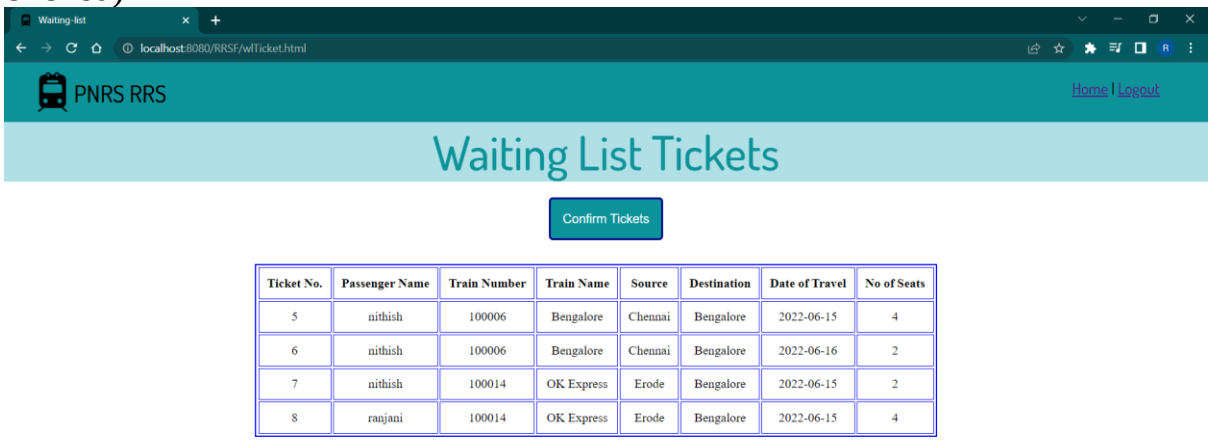
```
mysql> select * from train;
+--------+------------------+------------+-------------+-----------+----------+
| tid    | name             | source     | destination | max_seats | duration |
+--------+------------------+------------+-------------+-----------+----------+
| 100001 | Shari Express    | Chennai    | Madurai     |        78 |        6 |
| 100002 | MM Express       | Chennai    | Madurai     |        89 |        6 |
| 100003 | Window Express   | Coimbatore | Madurai     |        80 |        5 |
| 100004 | Chennai Express  | Chennai    | Coimbatore  |        95 |        7 |
| 100005 | Superfast Express| Erode      | Madurai     |        70 |        4 |
| 100006 | Bengalore        | Chennai    | Bengalore   |        75 |        6 |
| 100007 | South Express    | Tirupur    | Chennai     |        65 |        8 |
| 100008 | Natural Express  | Madurai    | Chennai     |        40 |        8 |
| 100010 | SPR Express      | Bengalore  | Coimbatore  |        45 |        5 |
| 100011 | Nill Express     | Chennai    | Bengalore   |        55 |        7 |
| 100012 | Magical Express  | Erode      | Madurai     |        50 |        4 |
| 100013 | VTV Express      | Tirupur    | Chennai     |        55 |        7 |
| 100014 | OK Express       | Erode      | Bengalore   |        70 |        5 |
| 100017 | AJR              | Tirupur    | Chennai     |       100 |        8 |
+--------+------------------+------------+-------------+-----------+----------+
14 rows in set (0.01 sec)

mysql>
```

(If a particular train id does not exist, it shows a pop-up that the train record does not exists)



Waiting List Tickets Updating Page

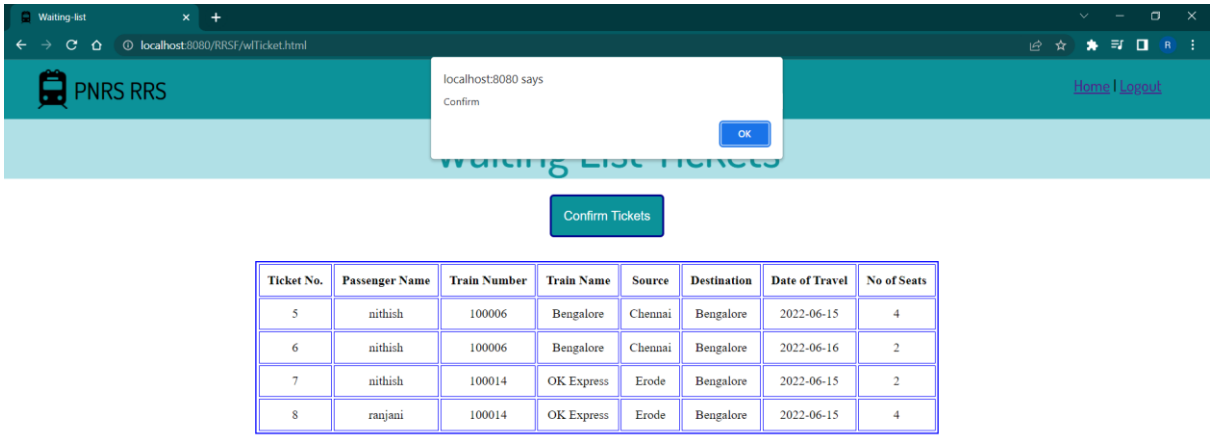
(The waiting list tickets will get confirmed when the confirm tickets button is clicked)



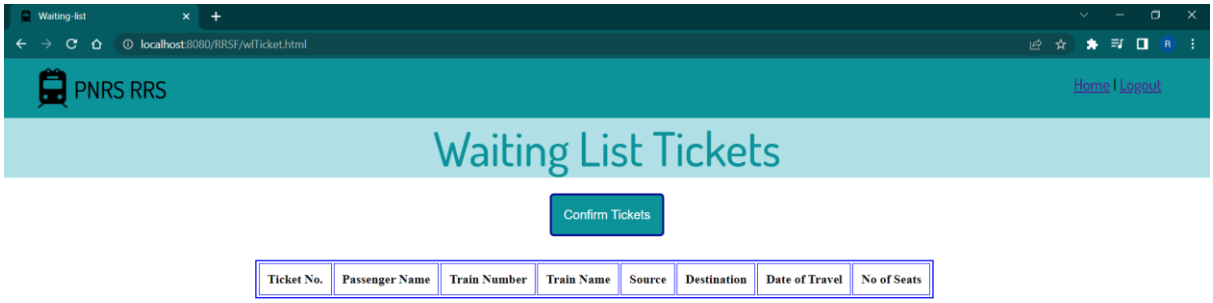
Ticket No.	Passenger Name	Train Number	Train Name	Source	Destination	Date of Travel	No of Seats
5	nithish	100006	Bengalore	Chennai	Bengalore	2022-06-15	4
6	nithish	100006	Bengalore	Chennai	Bengalore	2022-06-16	2
7	nithish	100014	OK Express	Erode	Bengalore	2022-06-15	2
8	ranjani	100014	OK Express	Erode	Bengalore	2022-06-15	4

```
mysql> select * from ticket;
+-------+-----+--------+------------+-----+-----+-----------+
| ticno | pid | tid    | dot        | ano | cno | ticstatus |
+-------+-----+--------+------------+-----+-----+-----------+
|     3 |   1 | 100001 | 2022-05-26 |   2 |   1 | Confirmed |
|     4 |   2 | 100009 | 2022-05-31 |   3 |   2 | Waiting   |
|     5 |   1 | 100006 | 2022-06-15 |   1 |   3 | Waiting   |
|     6 |   1 | 100006 | 2022-06-16 |   1 |   1 | Waiting   |
|     7 |   1 | 100014 | 2022-06-15 |   1 |   1 | Waiting   |
+-------+-----+--------+------------+-----+-----+-----------+
5 rows in set (0.02 sec)
```

(After clicking the OK button, the waiting list tickets are removed from the table, and it has been confirmed and updated in the database)



Ticket No.	Passenger Name	Train Number	Train Name	Source	Destination	Date of Travel	No of Seats
5	nithish	100006	Bengalore	Chennai	Bengalore	2022-06-15	4
6	nithish	100006	Bengalore	Chennai	Bengalore	2022-06-16	2
7	nithish	100014	OK Express	Erode	Bengalore	2022-06-15	2
8	ranjani	100014	OK Express	Erode	Bengalore	2022-06-15	4



Ticket No.	Passenger Name	Train Number	Train Name	Source	Destination	Date of Travel	No of Seats

```
mysql> select * from ticket;
+-------+-----+--------+------------+-----+-----+-----------+
| ticno | pid | tid    | dot        | ano | cno | ticstatus |
+-------+-----+--------+------------+-----+-----+-----------+
|     3 |   1 | 100001 | 2022-05-26 |   2 |   1 | Confirmed |
|     5 |   1 | 100006 | 2022-06-15 |   1 |   3 | Confirmed |
|     6 |   1 | 100006 | 2022-06-16 |   1 |   1 | Confirmed |
|     7 |   1 | 100014 | 2022-06-15 |   1 |   1 | Confirmed |
|     8 |   4 | 100014 | 2022-06-15 |   2 |   2 | Confirmed |
+-------+-----+--------+------------+-----+-----+-----------+
5 rows in set (0.00 sec)

mysql>
```

Exp. No.: 8
Date: 26/5/22

TEST CASES AND TEST PLANS

TEST CASES

Aim: To develop test cases for the Railway Reservation System and improve the design based on the results.

Identification of Testing Scenarios:

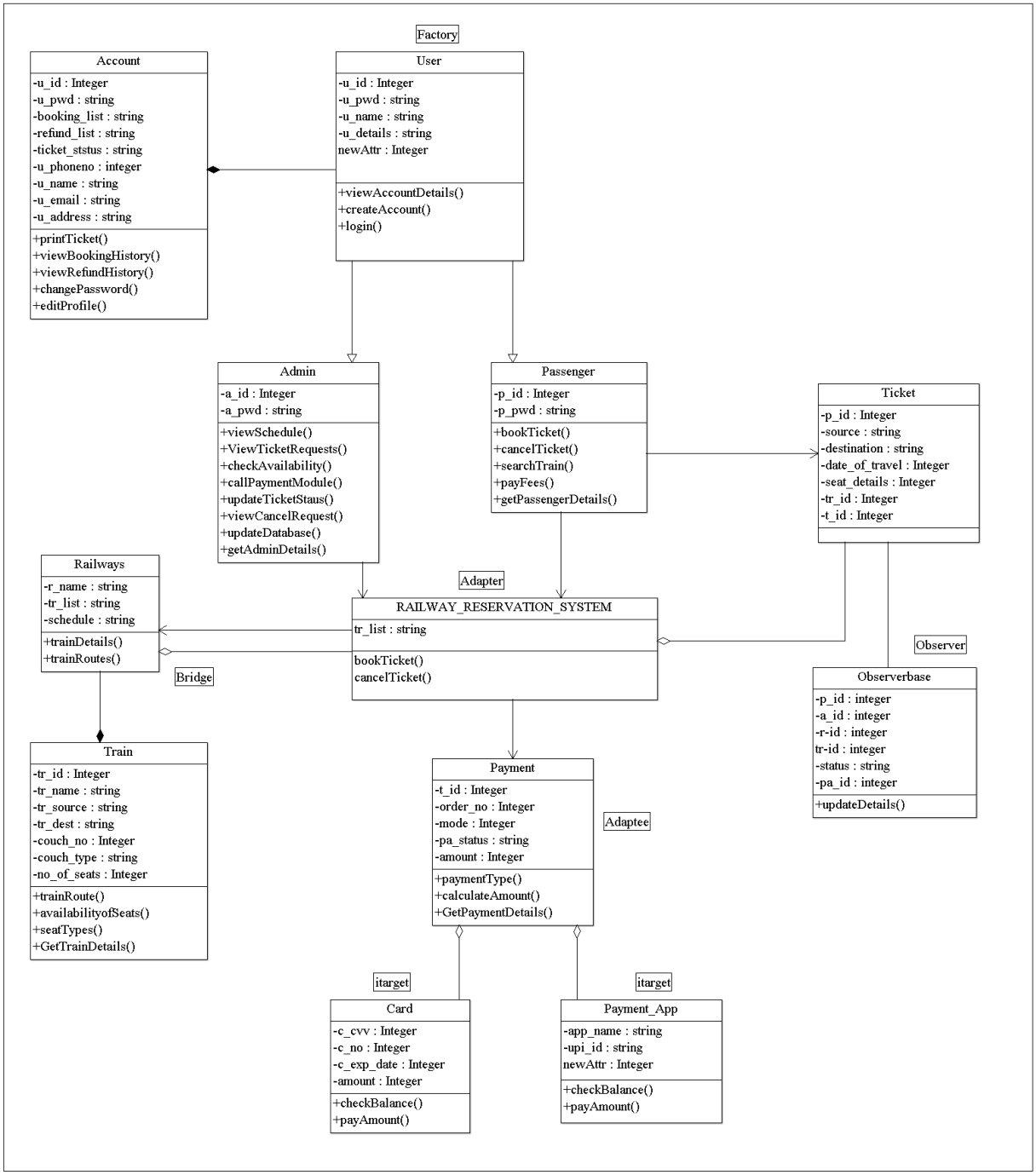
1. Check if user can be authenticated.
2. Check if new user can login successfully.
3. Check if the Tickets are booked successfully.
4. Check if the payment is successful.
5. Check if new trains are added.
6. Check if a train is deleted.
7. Check if the status of the train is displayed correctly.

TC No.	Scenario	Testing Step	Input	Output	Pass or Fail
1.	Login (user)	Enter login credentials like Email, password	nithish@gamil.com 123	Logged in successfully	Pass
2.	Login (user)	Enter login credentials like Email, password	nithish@gmail.com 321	Logged in failure due to invalid credentials	Fail
3.	Login (Admin)	Enter login credentials like Email, password	admin@gmail.com admin	Logged in successfully	Pass
4.	Sign up	Enter login credentials like Fullname,Email, Password,Gender, City,Contact No.	Ranjani ranjani@gmail.com 456 Female Chennai 9876344567	Signed up successfully	Pass
5.	Book Ticket	Click the Book ticket button and select the details like source,destination and date of travel	Chennai Madurai 25/05/2022	Available trains are displayed	Pass
6.	Payment (card)	Click the Book Now button to book a ticket using card and enter the details like name,card number,expiry date,cvv	John Doe January 2018 xxx	Ticket booked successfully	Pass

7	Payment (Payment App)	Click the Book Now button to book a ticket using Payment App and enter the details like name,UPI ID	Ram 54321	Ticket booked successfully	Pass
8.	Booking History	Click the Booking history button and enter the Email id	ranjani@gmail.com	Booking history details are displayed	Pass
9.	Ticket Status	Click the Ticket Status button and enter the PNR number	8765432109	Current ticket status is displayed	Pass
10.	Cancel Ticket	Click the Cancel Ticket button and enter the PNR number	2345678901	Ticket cancelled successfully	Pass
11.	Search Train	Click the Search Train button and enter the details like source and destination	Erode Madurai	The trains available in the Database are displayed	Pass
12.	Train Information Management -Add Train	Click the Train Information Management button and choose the Add Train option and enter the details like train id,train name,source,destination,maximum seats,duration	1 Chennai Express Chennai Madurai 70 7 hrs	The new train added is displayed in the database table	Pass
13.	Train Information Management -Add Train	Click the Train Information Management button and choose the Add Train option and enter the details like train id,train name,source,destination,maximum seats,duration	1 Indian Express Erode Chennai 70 7 hrs	The new train is not added to the database if the train id is already present	Fail
14.	Train Information Management -Delete Train	Click the Train Information Management button and choose the Delete Train option and enter the train id	2	The particular train is deleted	Pass

Exp. No.: 9
Date: 2/6/22

APPLYING DESIGN PATTERNS


Factory
Account
-u_id : Integer
-u_pwd : string
-booking_list : string
-refund_list : string
-ticket_ststus : string
-u_phoneno : integer
-u_name : string
-u_email : string
-u_address : string
+printTicket()
+viewBookingHistory()
+viewRefundHistory()
+changePassword()
+editProfile()
User
-u_id : Integer
-u_pwd : string
-u_name : string
-u_details : string
newAttr : Integer
+viewAccountDetails()
+createAccount()
+login()
Admin
-a_id : Integer
-a_pwd : string
+viewSchedule()
+ViewTicketRequests()
+checkAvailability()
+callPaymentModule()
+updateTicketStaus()
+viewCancelRequest()
+updateDatabase()
+getAdminDetails()
Passenger
-p_id : Integer
-p_pwd : string
+bookTicket()
+cancelTicket()
+searchTrain()
+payFees()
+getPassengerDetails()
Ticket
-p_id : Integer
-source : string
-destination : string
-date_of_travel : Integer
-seat_details : Integer
-tr_id : Integer
-t_id : Integer
Railways
-r_name : string
-tr_list : string
-schedule : string
+trainDetails()
+trainRoutes()
Adapter
RAILWAY_RESERVATION_SYSTEM
tr_list : string
bookTicket()
cancelTicket()
Observer
Bridge
Observerbase
-p_id : integer
-a_id : integer
-r-id : integer
tr-id : integer
-status : string
-pa_id : integer
+updateDetails()
Train
-tr_id : Integer
-tr_name : string
-tr_source : string
-tr_dest : string
-couch_no : Integer
-couch_type : string
-no_of_seats : Integer
+trainRoute()
+availabilityofSeats()
+seatTypes()
+GetTrainDetails()
Payment
-t_id : Integer
-order_no : Integer
-mode : Integer
-pa_status : string
-amount : Integer
+paymentType()
+calculateAmount()
+GetPaymentDetails()
Adaptee
itarget
Card
-c_cvv : Integer
-c_no : Integer
-c_exp_date : Integer
-amount : Integer
+checkBalance()
+payAmount()
itarget
Payment_App
-app_name : string
-upi_id : string
newAttr : Integer
+checkBalance()
+payAmount()

Exp. No.: 10
Date: 9/6/22

TESTING AFTER REFINEMENT REFACTORING

In our PNRS Railway Reservation System, the header section containing links to pages implementing different functionalities has been reused in all the pages to ease the process of navigation to each page available in the system without accessing the home page every time, such as book ticket, booking history, ticket status, cancel ticket and search train.The home link has been reused in all the pages to get back to the home page at any point of time. The logout link has been reused in all the pages for logging out of the system at any given time.

The components like book ticket for booking the ticket, booking history for showing the transaction history, ticket status for showing whether the ticket is confirmed or not, cancel ticket for cancelling the ticket and search train for searching the available trains for the source and destination have also been reused in all the pages for easy navigation purpose.

In case if the user gets any problem in the process of booking the ticket the contact us link has been reused in all the pages for contacting us. About us link has been reused in all the pages for enabling the user to know about the system.

The train details have been retrieved from the database for both booking the ticket and for searching the train. For viewing the ticket status and cancelling the tickets, the ticket details are retrieved from the database.

In Admin side also the home link has been reused in all the pages to get back to the home page at any point of time. We have also reused the logout link in all the pages for logging out purpose at any point of time. The train details have been retrieved from the database for adding , deleting and viewing the trains.