# **Online Railway Reservation System**

# POC

****

**Table of Contents**

[1.0Document Purpose 4](file:///C:\Users\HP\Downloads\Railway%20Reservation%20System.docx#_Toc94636300)

[2.0Intended Audience 4](file:///C:\Users\HP\Downloads\Railway%20Reservation%20System.docx#_Toc94636301)

[3.0Project Background, Objective(s) 4](file:///C:\Users\HP\Downloads\Railway%20Reservation%20System.docx#_Toc94636302)

[4.0Design Pattern 5](file:///C:\Users\HP\Downloads\Railway%20Reservation%20System.docx#_Toc94636305)

[5.0Solution Diagram 5](file:///C:\Users\HP\Downloads\Railway%20Reservation%20System.docx#_Toc94636306)

[6.0Solution Steps 5](file:///C:\Users\HP\Downloads\Railway%20Reservation%20System.docx#_Toc94636307)-6

[7.0Classes/function name 7](file:///C:\Users\HP\Downloads\Railway%20Reservation%20System.docx#_Toc94636308)

[8.0Database Diagram 8](file:///C:\Users\HP\Downloads\Railway%20Reservation%20System.docx#_Toc94636309)

[9.0Use Case Diagram 9](file:///C:\Users\HP\Downloads\Railway%20Reservation%20System.docx#_Toc94636310)

[10.0E-R Diagram 10](file:///C:\Users\HP\Downloads\Railway%20Reservation%20System.docx#_Toc94636311)

[11.0API Canvas 10-11](file:///C:\Users\HP\Downloads\Railway%20Reservation%20System.docx#_Toc94636311)

[12.0Data Fllow Diagram 11](file:///C:\Users\HP\Downloads\Railway%20Reservation%20System.docx#_Toc94636314)

[13.0Unit Testing 12-18](file:///C:\Users\HP\Downloads\Railway%20Reservation%20System.docx#_Toc94636314)

DOCUMENT APPROVAL

**Approvers of this document**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Department** | **Role** | **Signature** | **Date** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Document Change History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Document Version #** | **Author** | **Date** | **Description** |
| 1.0 | Ladly Rout | 11/07/22 | Railway Ticket Management LLD |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Document Purpose

This document describes the solution architecture for Online Railway Reservation System.

# Intended Audience

This document is intended as a reference for the following roles and stakeholders who are interested in the Online Railway Reservation System technical architecture.

|  |  |
| --- | --- |
| Role | Nature of Engagement in the Online Railway Reservation System Architecture |
| Product Owners/SME | Key stakeholder to ensure that the architecture is aligned with business goals. |
| Business Analysts | Business analysts are one of the stakeholders who are informed with the key architectural decisions. |
| Enterprise Architects | To enforce Customer management Platform Architecture is aligned to business goals and architecture, architectural guidelines. |
| Solution Architects | To ensure solution design and architecture is aligned to business requirements, architectural guidelines. |
| Developers | Use Technical Architecture Document as the guiding document for detail design and implantation approach to align with Customer management Microservice |
| End-User | An End- user can check the train timings, train fares and other trains information and book/cancel a ticket. |

# Project Background, Objective(s)

## Project Background

Online Railway Reservation System leads to perform Management of railway ticket booking details where one can register themselves and perform various operations

## Project Objective

Online Railway Reservation System will perform various operations like reservation, cancellation of ticket Details.

The user of this system should first register for any interaction with the system. Once registered, he/she will be provided with a username and password for the user to log in. After logging in the user should select the kind of activity, he would like to perform like booking a ticket, cancelling a ticket, look out for help and so on. A person can check the train timings, train fares and other trains information without login.

## System Requirements

**1.Development Environment:**

a. Windows 10 or higher

b. RAM: 8GB Recommended

c. Processor: 2GHz or more  
d. Storage: 10Gb

e. SDK and IDE: .NET 5, Visual Studio 2019 or Higher, Visual Studio Code

**2.Testing Environment**

1. Database: Microsoft SQL Server Management Studio 18
2. Operating System: Windows 10
3. SDK and IDE: .NET 5, Visual Studio 2019 or Higher, Visual Studio Code
4. Web Browser: Latest Browser
5. Internet Connectivity
6. 8 GB RAM (Recommended)
7. Minimum 4 GB of free storage
8. Processor – 2 GHz or higher

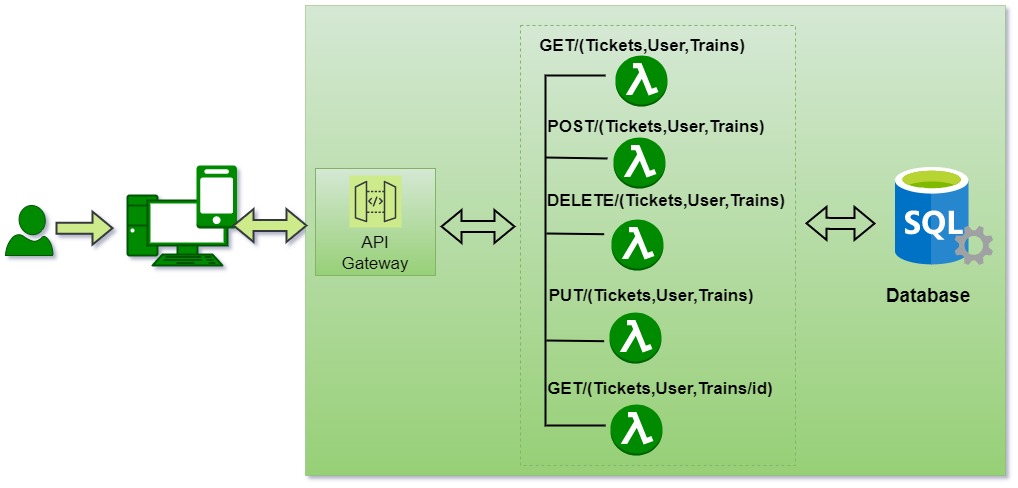
## Technologies Used

* 1. Angular 10
  2. ASP .Net Web API
  3. Microsoft SQL Server Management Studio 18

# Design Pattern

|  |  |  |
| --- | --- | --- |
| # | Name | Description |
| 1 | Angular | Creating a user interface (Front-end), and consuming API services. |
| 2 | API | Using HTTP requests, we will use the respective action to trigger various operations |
| 3 | Database | For storing, maintaining and accessing user, train and booking details. |

# Solution Diagram



# 6.0 Solution Steps

**6.1 User**

**Registering User**

1. User will be able to register himself by entering the details like Name, age, gender, address, Phone, email and Password.
2. After filling the user credentials the form is validated.
3. If the validation is successful, by clicking the submit button browser directs the request to customer registration API.
4. The call reaches the API gateway.
5. API gateway does the routing and saves the data in the database.
6. Once a user is successfully registered a successful message is displayed and the user is redirected to the login page.

**Viewing Train and Fare Details**

1. User will be able to view trains without logging in.
2. User can view trains by providing Source Station, Destination Station and date of journey.
3. Call reaches the API gateway.
4. If the trains are found with the given parameters it is displayed along with their fare, to the user otherwise an information is (‘No trains found’) displayed.

**Booking of tickets**

1. After a user found their suitable train the user will click on Book Now button.
2. In order to book the train the user will have to log in first.
3. The user will be redirected to the Log in page where they can fill their details and if they do not have an account they can click on Sign-up.
4. Once the user is logged in, they can add passenger (max. 6 at a time).
5. Once the passengers are added fare details are updated and displayed to the user.
6. The user can then Pay and Book a ticket.
7. An information is shown after successful booking.
8. A user can track their booking status by clicking on bookings.

**Cancellation of tickets**

1. User can also cancel a booked ticket if they require.
2. When the user clicks on booking and sees their status there will be a cancel button.
3. After clicking on cancel button an alert is displayed (‘Are you sure you want to cancel?’)
4. Once the user confirms their ticket gets cancelled and get a conformation page that their ticket is successfully cancelled.
5. The booking status of the ticket will be changed to cancelled.

**6.2 Admin**

1.Admin will be able to edit and add new train and fare details

2.GetAllTrains() will let the admin view all the train details.

3.GetCustomerById() will allow Admin to view customer details by ID.

# 7.0 Classes/function

|  |  |  |
| --- | --- | --- |
| Serial no. | Class | Description |
| 1 | Model Folder | Model for holding the booking schema details for user, train, booking, ticket and transaction respectively. |
| 2 | Repository Folder | The Interface in Data Access Layer for the user, train, booking, ticket and transaction model respectively. |
| 3 | Services Folder | It’s the Business Access Layer holding the Business Logic and meditates the communication between the controller and repository (Data Access) Layer. |
| 4 | Controller Folder | Controller handles the incoming HTTP requests and send the response back to the caller. |

# 8.0 Database Diagram

**User Table**

|  |  |  |
| --- | --- | --- |
| Sl no. | Name | Type |
| 1 (PK) | **UserID** | **Numeric** |
| 2 | **Name** | **Varchar(50)** |
| 3 | **Email** | **Varchar(50)** |
| 4 | **Address** | **Varchar(100)** |
| 5 | **Mobile** | **Varchar(10)** |
| 6 | **Password** | **Varchar(20)** |
| 7 | **isActive** | **Boolean** |

**Booking Table**

|  |  |  |
| --- | --- | --- |
| Sl no. | Name | Type |
| 1 (PK) | **BookingID** | **Numeric** |
| 2 (FK) | **TrainID** | **Numeric** |
| 3 (FK) | **UserID** | **Numeric** |
| 4 | **Classes** | **Varchar(20)** |
| 5 | **Date** | **Date Time** |
| 6 | **Status** | **Varchar(20)** |
| 7 | **SeatNum** | **Numeric** |
| 8 (FK) | **PassengerID** | **Numeric** |
| 9 | **Fare** | **Decimal(10,2)** |

**Train Table**

|  |  |  |
| --- | --- | --- |
| Sl no. | Name | Type |
| 1 (PK) | **TrainID** | **Numeric** |
| 2 | **Name** | **Varchar(50)** |
| 3 | **Arrival Time** | **Varchar(20)** |
| 4 | **DepartureTime** | **Varchar(20)** |
| 5 | **Arrival Date** | **Date Time** |
| 6 | **Departure Date** | **Date Time** |
| 7 | **Arrival Station** | **Varchar(20)** |
| 8 | **Departure Station** | **Varchar(20)** |
| 9 | **Distance** | **Decimal(10,2)** |
| 10 | **isActive** | **Boolean** |

**Transaction Table**

|  |  |  |
| --- | --- | --- |
| Sl no. | Name | Type |
| 1 (PK) | **TransactionID** | **Numeric** |
| 2 (FK) | **BookingID** | **Numeric** |
| 3 | **Fare** | **Decimal(10,2)** |
| 4 | **Status** | **Varchar(20)** |

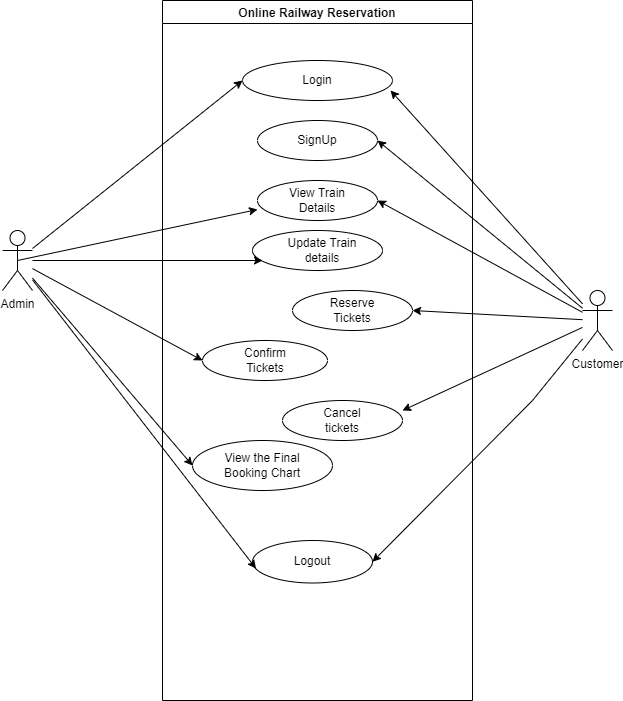
**Passenger Table**

|  |  |  |
| --- | --- | --- |
| Sl no. | Name | Type |
| 1 (PK) | **PassengerID** | **Numeric** |
| 2 (FK) | **UserID** | **Numeric** |
| 3 | **Name** | **Varchar(20)** |
| 4 | **Age** | **Varchar(20)** |
| 5 | **Gender** | **Varchar(20)** |
| 6 | **Class** | **Varchar(20)** |

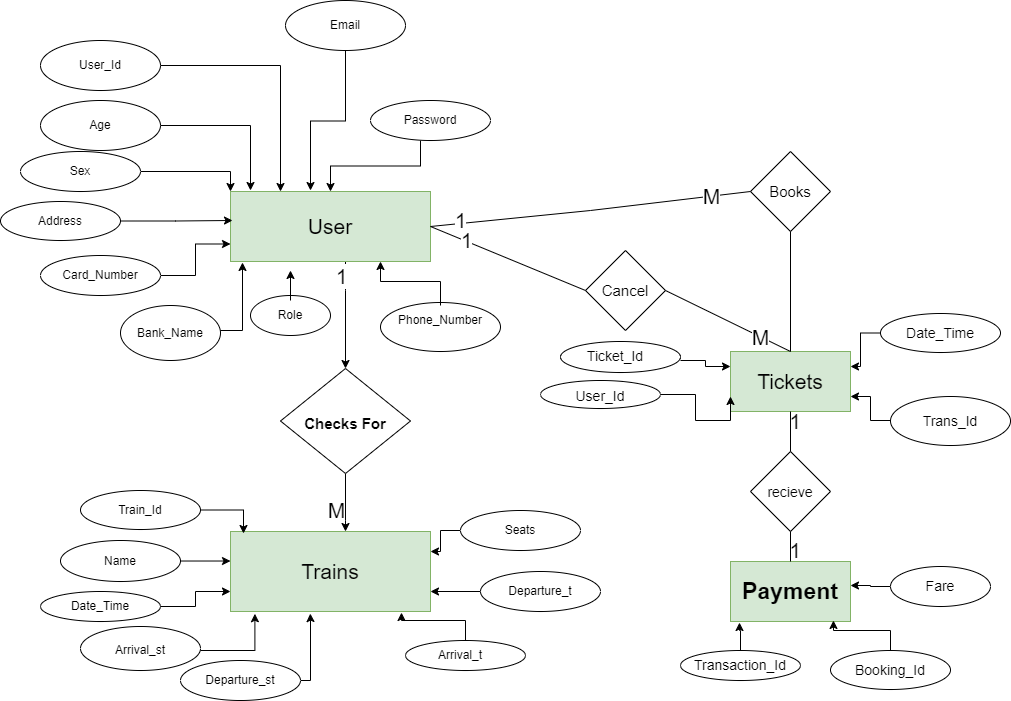
**Seat Table**

|  |  |  |
| --- | --- | --- |
| Sl no. | Name | Type |
| 1 (PK) | **SeatID** | **Numeric** |
| 2 (FK) | **TrainID** | **Numeric** |
| 3 | **FirstAC** | **Varchar(20)** |
| 4 | **SecondAC** | **Varchar(20)** |
| 5 | **Sleeper** | **Varchar(20)** |
| 6 | **Total** | **Varchar(20)** |

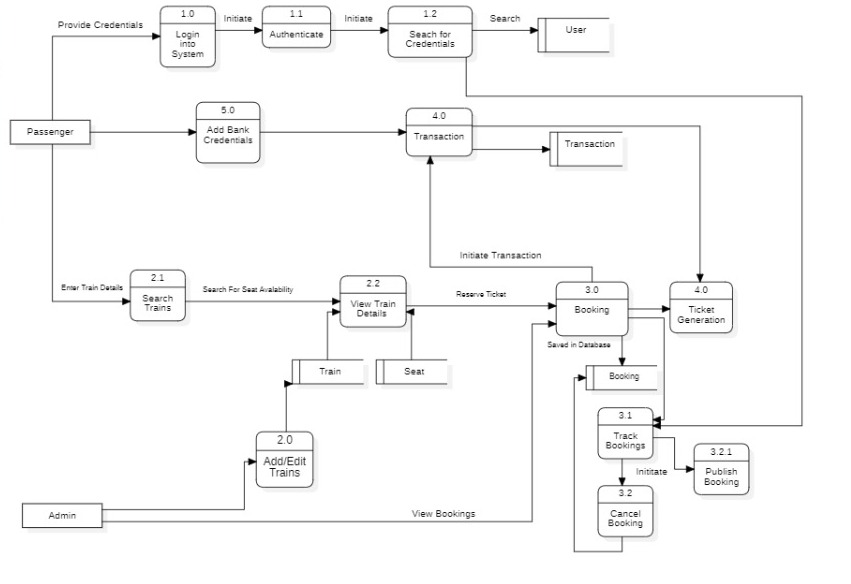
9.0 Use Case Diagram



10.0 Entity Relation Diagram



11.0 Data-Flow-Diagram



# 12.0 API Canvas

12.1 User

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Service | Path | Verb | API Description | Role | Auth |
| Customer-management | /Customer | POST | To register a customer | Customer | True |
| Customer-management | /Customer/Id | GET | To get a customer by Id | Admin | True |
| Customer- management | /Customer | GET | To get the list of customers | Admin | True |
| Customer- management | /Customer | PUT | To edit the details of a customer | Customer | True |

12.2 Booking

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Service | Path | Verb | API Description | Role | Auth |
| Booking-management | /Booking | POST | To reserve a Booking | Customer | True |
| Booking-management | /Booking/Id | PUT | To cancel a Booking | Customer | True |
| Booking-management | /Booking | PUT | To update Booking details | Customer | True |
| Booking-management | /Booking/Id | GET | To get the Booking  Details by Id | Customer | True |

12.3 Train

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Service | Path | Verb | API Description | Role | Auth |
| Train-management | /train | POST | To add a train | Admin | True |
| Train-management | /train | PUT | To update the details of a train | Admin | True |
| Train-management | /train | GET | To get the list of the trains | Customer | True |
| Train-management | /train/ID | PUT | To deactivate the details of a train | Admin | True |
| Train-management | /train/Id | GET | To get the train details by id | Customer | True |

# 13.0 Unit Testing

1. **For Registration of customers**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE-CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | Actual Result |
| 1 | Customer Registration | Enter the valid data to get registered | Customer needs to enter all the valid details | 1) Enter  Customer:  Name: Jane Eyre  Age:22  Sex: Female  Address: 101, Baker’s Street  Email: jane123@gmail.com  Phone: 6572450126  Password: Jane123  2) Enter Submit | <Valid data | Successful registration | Successful registration |
| 2 | Customer registration | Enters missing data to get registered | Customer needs to enter the valid details with no phone number and no age | 1) Enter  Customer:  Name: Jane Eyre  Age:  Sex: Female  Address: 101, Baker’s Street  Email: jane123@gmail.com  Phone:  Password: Jane123  2) Enter Submit | <Invalid Data,  Customer needs to provide details of all the required field | Unsuccessful  Registration | Unsuccessful Registration |
| 3 | Customer registration | Enters invalid data to get registered | Customer needs to enter the valid details with wrong phone number type | 1) Enter  Customer:  Name: Jane Eyre  Age:22  Sex: Female  Address: 101, Baker’s Street  Email: jane123@gmail.com  Phone: 987654332  Password: Jane123  2) Enter Submit | <Invalid Data,  Customer needs to enter 10 digit phone number | Unsuccessful  Registration | Unsuccessful  Registration |

**2. To get customer details by Id**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE-CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | ACTUAL RESULTS |
| 1 | Customer List by Id | enter the valid Customer Id of customer in parameter to get details | That id needs to be present in database | 1) Enters Valid Id  2) Enter Submit | Valid Id | Customer Details | Customer Details |
| 2 | Customer List by Id | enters the wrong Customer Id which is not there in database to get customer details | That id needs to be present in database | 1) Enters invalid Id  2) Enter Submit | Invalid Id | Customer Details with this id is not present | Customer Details with this id is not present |

**3.For customer’s details Updating**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE-CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | ACTUAL RESULT | STATUS  (PASS/  FAIL) |
| 1 | Customer Updation | enter the valid id of customer in parameter to get details | That id needs to be present in database | 1) Enters Valid Id  2) Enter Submit | Valid Id | Customer Update  Successfully | Customer Update  Successfully | pass |
| 2 | Customer Updation | enters the wrong id which is not there in database to get customer details | That id needs to be present in database | 1) Enters invalid Id  2) Enter Submit | Invalid Id | Customer Details with this  Customer Id you want to update is not present | Customer Details with this id you want to update is not present | fail |

**4**.**To add a Train**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE-CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | Actual Result |
| 1 | Train Registration | Enter the valid data to register. | Admin needs to add train details. | 1) Enter  Train details:  Name: Shatabdi  Seats:250  Arrival Time: 8:30  Departure Time: 14:30  Arrival Station: Karachi  Departure Station: Lahore  2) Enter Submit | <Valid data | Successful registration | Successful registration |
| 2 | Train Registration | Enters missing data to register. | Admin needs to enter the valid details with no seat number | 1) Enter  Train details:  Name: Shatabdi  Seats:  Arrival Time: 8:30  Departure Time: 14:30  Arrival Station: Karachi  Departure Station: Lahore  2) Enter Submit | <Invalid Data,  Admin needs to provide details of all the required field | Unsuccessful  Registration | Unsuccessful Registration |
| 3. | Train Registration | Enters Invalid data to register | Admin needs to enter invalid details. | 1) Enter  Train details:  Name: Shatabdi  Seats:250  Arrival Time: 8:30  Departure Time: 25:30  Arrival Station: Karachi  Departure Station: Lahore  2) Enter Submit | <invalid details>,  Incorrect time. | Unsuccessful  Registration | Unsuccessful  Registration |

**5. To get train details by Id**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE-CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | ACTUAL RESULTS |
| 1 | Train details by Id | enter the valid Train Id of Trains in parameter to get details | That id needs to be present in database | 1) Enters Valid Id  2) Enter Submit | Valid Id | Train Details | Train Details |
| 2 | Train details by Id | enters the wrong Train Id which is not there in database to get Trains details | That id needs to be present in database | 1) Enters invalid Id  2) Enter Submit | Invalid Id | Train Details with this id is not present | Train Details  with this id is not present |

**6.For Updating Train Details**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE-CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | ACTUAL RESULT | STATUS  (PASS/  FAIL) |
| 1 | Train Updation | enter the valid id of train to get details | That id needs to be present in database | 1) Enters Valid Id  2) Enter Submit | Valid Id | Update  Successfully | Update  Successfully | pass |
| 2 | Train Updation | enters the wrong id which is not there in database to get Train details | That id needs to be present in database | 1) Enters invalid Id  2) Enter Submit | Invalid Id | Train Details with this  Train Id you want to update is not present | Train Details with this id you want to update is not present | fail |

**7.Deleting Train Record**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE-CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | ACTUAL RESULT |
| 1 | Train Deletion | enter the valid id of Train in parameter to get details | That id needs to be present in database | 1) Enters Valid Id  2) Enter Submit | Valid Id | Deleted successfully | Deleted successfully |
| 2 | Train Deletion | enters the wrong id which is not there in database to get Train details | Id needs to be present in database | 1) Enters invalid Id  2) Enter Submit | Invalid Id | Train with this id you want to delete is not present | Train with this id you want to delete is not present |

**8**.**To add a Ticket**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE-CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | Actual Result |
| 1 | Ticket Registration | Enter the valid data to register. | User needs to add details. | 1) Enter  details:  User\_Id: 101  Transaction\_Id:501  Date & Time: 20:30  2) Enter Submit | <Valid data | Successful registration | Successful registration |
| 2 | Train Registration | Enters missing data to register. | Enter the valid details with no Date & Time | 1) Enter  details:  User\_Id: 101  Transaction\_Id:501  Date & Time:  2) Enter Submit | <Invalid Data,  Needs to provide details of all the required field | Unsuccessful  Registration | Unsuccessful Registration |
| 3. | Train Registration | Enters Invalid data to register | Needs to enter invalid details. | 1) Enter  details:  User\_Id: 0  Transaction\_Id:501  Date & Time: 25:30  2) Enter Submit | <invalid details>,  Incorrect time and Incorrect User\_Id. | Unsuccessful  Registration | Unsuccessful  Registration |

**9. To get Ticket details by Id**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE-CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | ACTUAL RESULTS |
| 1 | Ticket Details by Id | enter the valid Ticket\_Id to get details | That id needs to be present in database | 1) Enters Valid Id  2) Enter Submit | Valid Id | Ticket Details | Ticket Details |
| 2 | Ticket Details by Id | enters the enter the valid Ticket\_Id which is not there in database to get customer details | That id needs to be present in database | 1) Enters invalid Id  2) Enter Submit | Invalid Id | Ticket Details with this id is not present | Ticket Details with this id is not present |

**10.For updating ticket details**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE-CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | ACTUAL RESULT | STATUS  (PASS/  FAIL) |
| 1 | Ticket Updation | enter the valid id of Ticket to get details | That id needs to be present in database | 1) Enters Valid Id  2) Enter Submit | Valid Id | Update  Successfully | Update  Successfully | pass |
| 2 | Ticket Updation | enters the wrong id which is not there in database to get Ticket details | That id needs to be present in database | 1) Enters invalid Id  2) Enter Submit | Invalid Id | Ticket Details with this Id you want to update is not present | Ticket Details with this id you want to update is not present | fail |

**11.Deleting Ticket Record**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test CASE ID | TEST CASE SCENARIO | TEST CASE | PRE-CONDITION | TEST STEPS | TEST DATA | EXPECTED RESULT | ACTUAL RESULT |
| 1 | Ticket Deletion | enter the valid id of Ticket to get details | That id needs to be present in database | 1) Enters Valid Id  2) Enter Submit | Valid Id | Deleted successfully | Deleted successfully |
| 2 | Ticket Deletion | enters the wrong id which is not there in database to get ticket details | Id needs to be present in database | 1) Enters invalid Id  2) Enter Submit | Invalid Id | Ticket with this id you want to delete is not present | Ticket with this id you want to delete is not present |