

LOCOMOTIVE

Online Railway Reservation System

**Table of Contents**

[1.0 Document Purpose 3](#_Toc94636300)

[2.0 Intended Audience 3](#_Toc94636301)

[3.0 Project Background, Objective(s) 4](#_Toc94636302)

[4.0 Design Pattern 5](#_Toc94636305)

[5.0 Solution Diagram 5](#_Toc94636306)

[6.0 Solution Steps 6](#_Toc94636307)

[7.0 Classes/function name 7](#_Toc94636308)

[8.0 Data model/Tables Diagram 8](#_Toc94636309)

[9.0 Use Case Diagram 8](#_Toc94636310)

[10.0 Sequence Flow Diagram 9](#_Toc94636314)

[11.0 API Canvas 10](#_Toc94636311)

[12.0 HTTP Status Code 11](#_Toc94636314)

**0.0 DOCUMENT APPROVAL**

Approvers of this document

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

Document Change history

|  |  |  |  |
| --- | --- | --- | --- |
| Document Version | Author | Date | Description |
| 1.00 | Y.V.S.Aadarsh | 10/07/2022 | Initial Creation of the LLD. |
|  |  |  |  |

**1.0 Document Purpose**

The documents contain a detailed description of Online Railway Ticket Reservation System.

**2.0 Intended Audience**

|  |  |
| --- | --- |
| Role | Nature of Engagement in the Online Railway Reservation System Architecture |
| Product Owner/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. |

**3.0 Project Background & Objectives**

**3.1 Project Background**

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

**3.2 Project Objectives**

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.

**3.3 System Requirements**

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: Firefox or Chrome or Opera or Microsoft Edge
5. Internet Connectivity
6. 8 GB RAM (Recommended)
7. Minimum 4 GB of free storage

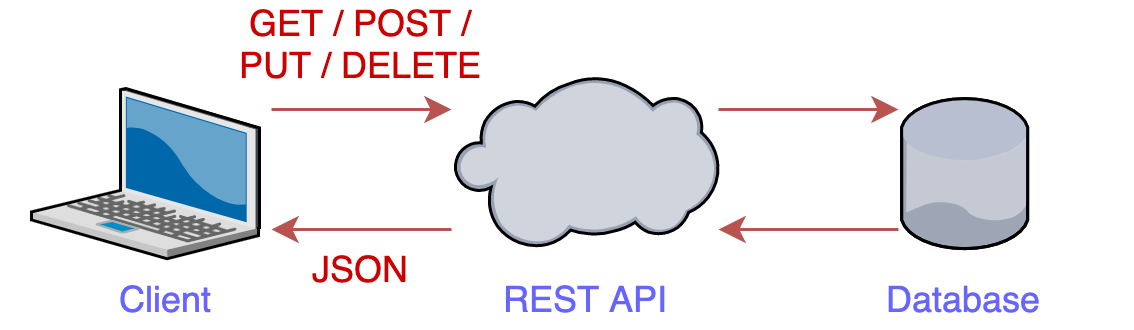
**3.4 Technologies Used**

* 1. Angular 10
  2. ASP .Net Web API (.NET Version 3.1)
  3. Microsoft SQL Server Management Studio 18

**4.0 Design Pattern**

|  |  |  |
| --- | --- | --- |
| Serial no. | Name | Description |
| 1 | API | Using HTTP requests, we will use the respective action to trigger various operations |
| 2 | Angular | Creating a user interface (Front-end) , Contains Html CSS And TypeScript files and Communicate with API |
| 3 | Database | For storing, maintaining The data Efficiently |

**5.0 Solution Diagram**



**6.0 Solution Steps**

**6.1 Admin**

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

2. Admin Can add and edit seats for the trains

3.Admin can see all trains by using GetAllTrains().

4.Admin can see the Customer Details by GetCustomerById().

**6.2 User**

**Registering User**

1. User will be able to register.
2. After Entering the user details the form is validated.
3. After the form is valid the submit button is enabled and Calls the API.
4. The call reaches the API.
5. The API Gets the Data and Saves It To Database.
6. Successful message is Displayed

**Viewing Train and Fare Details**

1. User will be able to view trains.
2. User can see trains by Giving Destination Station, Arrival Station and date of Travel. Call reaches the API gateway.
3. GetTrains method is called and all the values are passed.
4. the trains found Will Be display
5. else alert will be show with no details found message

**Booking of tickets**

1. After a user found their train, the user should click on Book Now button.
2. Only login users can book ticket
3. Login Page will open and a form is displayed with all the details to be filled
4. Then user can then Pay and Book a ticket.
5. An alert message after successful booking is shown.
6. A user can see all his previous bookings.

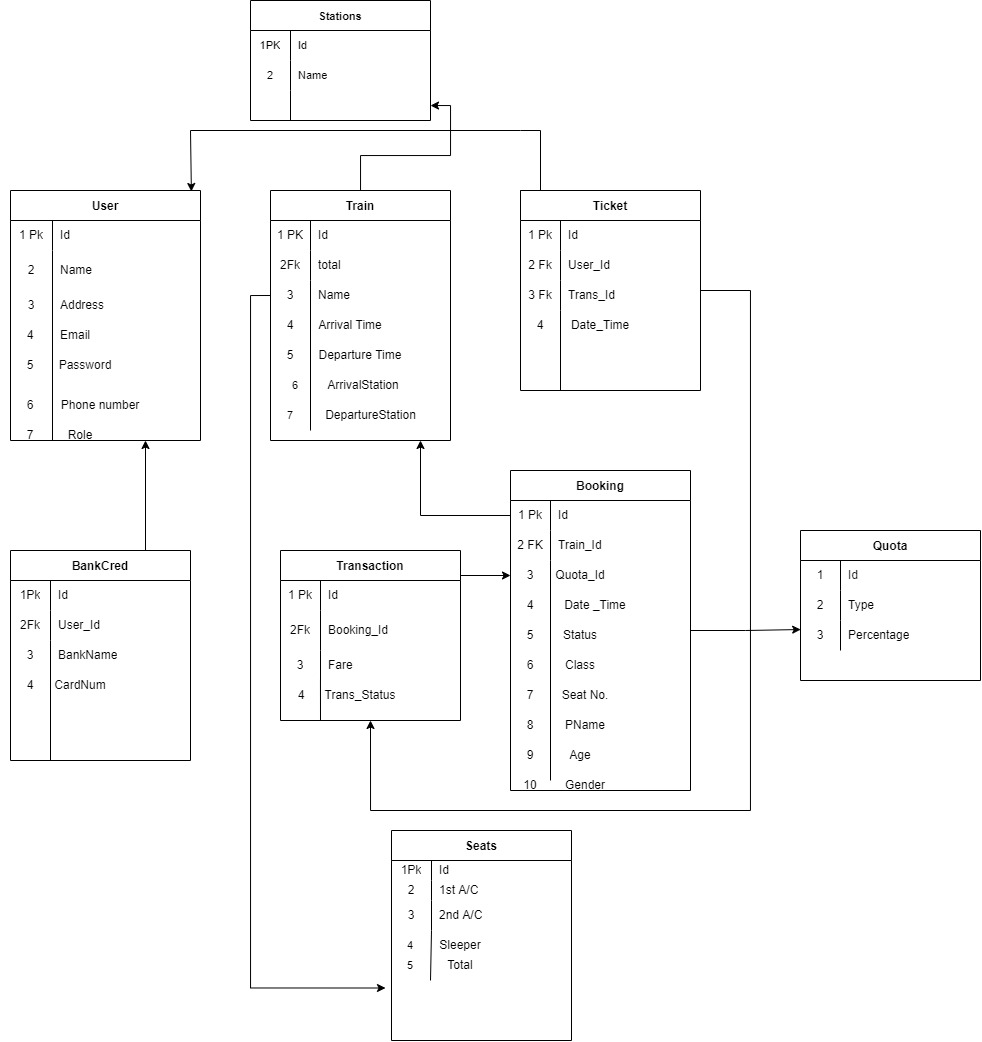
**Cancellation of tickets**

1. User can also cancel a ticket if they require.
2. User need to navigate to booking history and then he can cancel the ticket.

**7.0 Classes/Functions**

|  |  |  |
| --- | --- | --- |
| Serial no. | Class | Description |
| 1 | User.CS , Train.CS , Booking.CS , Ticket.CS , Transaction.CS | Model For Storing Schema for all the tables |
| 2 | IUser.CS, ITrains.CS, ITickets.CS, IBooking.CS, ITransaction.CS, User.CS, Train.CS, Booking.CS, Ticket.CS, Transaction.CS | Created Interfaces for maintaining the structure of the classes |
| 3 | UserController.CS , TrainController.CS , BookingController.CS , TicketController.CS , TransactionController.CS | Controllers are used to handle the HTTP Requests |
| 4 | UserServices.CS , TrainServices.CS , BookingServices.CS , TicketServices.CS , TransactionServices.CS | Services holding the Logic and communications between the controller and repository Layer. |
| 5 | app.component.css | Contains the styling if the component |
| 6 | app.component.html | Contains the html for a component |
| 7 | app.component.ts | File for writing type script methords component |
| 8 | Appsetting.JSON | It generally used to store the application configuration settings |
| 9 | Startup.CS | Start-up class is triggered at first when application started HTTP request is trigged |

**8.0 Database Diagram**

****

**9.0 Use Case Diagram**

**Diagram

Description automatically generated**

**10.0 Data Flow Diagram**



**11.0 API Canvas**

**11.1 User**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Service | Path | Verb | API Description | Role | Auth |
| Customer-management | /Customer | POST | To register a customer | No | 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 |

**11.2 Booking**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Service | Path | Verb | API Description | Role | Auth |
| Ticket-management | /ticket | POST | To reserve a ticket | No | True |
| Ticket-management | /ticket/Id | DELETE | To delete a ticket | No | True |
| Ticket-management | /ticket | PUT | To update ticket details | No | True |
| Ticket-management | /ticket/Id | GET | To get the ticket  Details by Id | No | True |

**11.3 Train**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Service | Path | Verb | API Description | Role | Auth |
| Train-management | /train | POST | To add ticket | Admin | True |
| Train-management | /ticket/ID | PUT | To update the details of a train | Admin | True |
| Train-management | /train | GET | To get the list of the trains | No | True |

**12.0 HTTP Status Codes**

400 Error: Bad Request

403 Error: Access Denied

404 Error: Not Found

502 Error: Bad Gateway

503 Error: Service Unavailable