**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**Department Of Computer Science Engineering**

**Course Code: 18CSC206J**

**Course Name: Software Engineering and Project Management**

**Project Title: Railway Management System**

**Name of the Student: Suvradeep Bala**

**Registration Number: RA1911028010065**

**Faculty: Mrs. P. Visalakshi**

|  |  |
| --- | --- |
| **Name of Team Members** | Nishant Jindal  Sidarth Krishna Jaitly  Suvradeep Bala |
| **Registration Number** | RA1911028010055  RA1911028010060  RA1911028010065 |
| **Day Order** | 5 |
| **Group Number** | 2 |

**Table of Contents**

**1. Problem statement**

**2. Stake Holder Analysis**

**3. Functional, Non- functional and Infrastructure Requirements**

**4. WBS and Risk Identification**

**5. System Architecture**

**6. List of Modules with Detailed Description**

**7. Testing**

**8. Conclusion**

**9. Sample code (Include main code only with partitions for each module, Max 3 pages)**

**10. Screenshots of project (For all modules with proper headings)**

**Problem Statement**

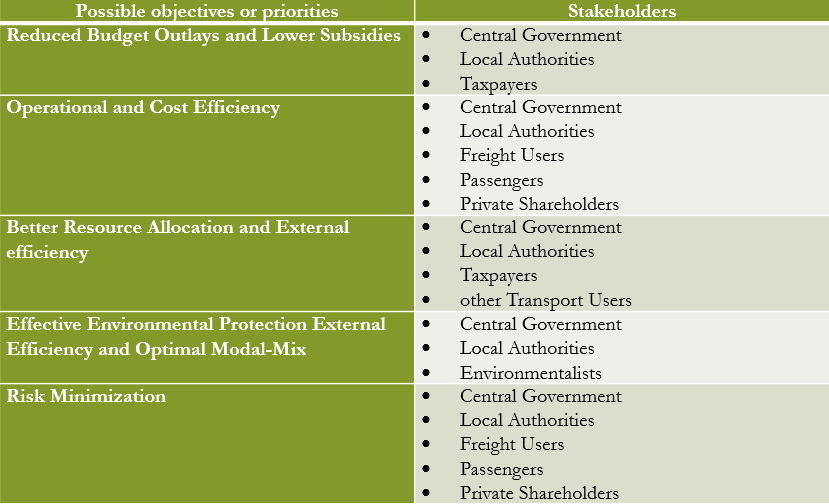
* In our day to day lives, transport is a matter of concern. Manual booking of tickets in stations is often an inconvenience. For convenient functioning of a reservation and management system, an online website has been developed
* This project aims to build a Railway Management System that helps manage the efficient arrival, departure and other related work of all the relevant trains on a train station. It has two major components –Passenger and trains. The Staff is in charge of managing trains and passengers and ensures comfortable travel for passengers.
* Passengers can buy tickets to board trains for travel (according to compartments), reserve seats, book food etc.
* This task was completed through a Web App SPA based on React framework.

**Stakeholder Analysis**

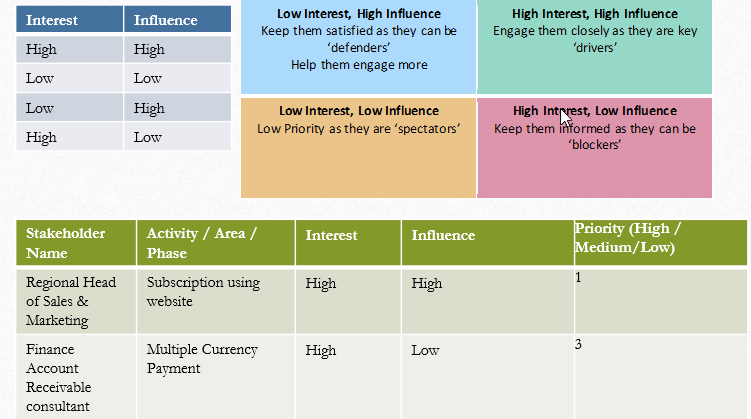
**Roles of stakeholders-**

* Educational institutions- funding
* Employees- efficient work place
* Passengers- operational cost efficiency
* Local authorities- transport of material, lower subsidies, minimizing risk, resource allocation
* Tax payers – risk minimizations
* Private shareholders- promotion and funding

**Identification of Stakeholders**

****

**Interest and Influence matrix**

****

**Communication plan for Stakeholders**

****

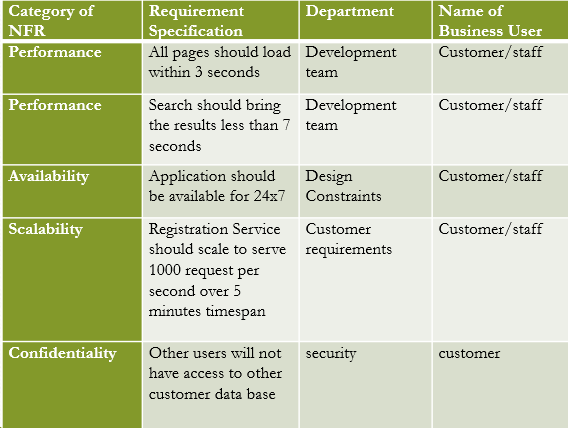
**Requirements**

* **Functional requirements**- can also be expressed in the form of “user story” which is the smallest unit of work in an agile framework. It’s an end goal, not a feature, expressed from the software user’s perspective.
* **Non function requirements**- These are requirements of minute focus
* **Infrastructure requirements**-Components forming the infrastructure

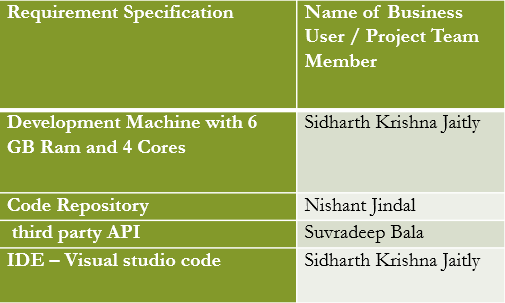
**Functional requirements**

****

**Non-functional requirements**

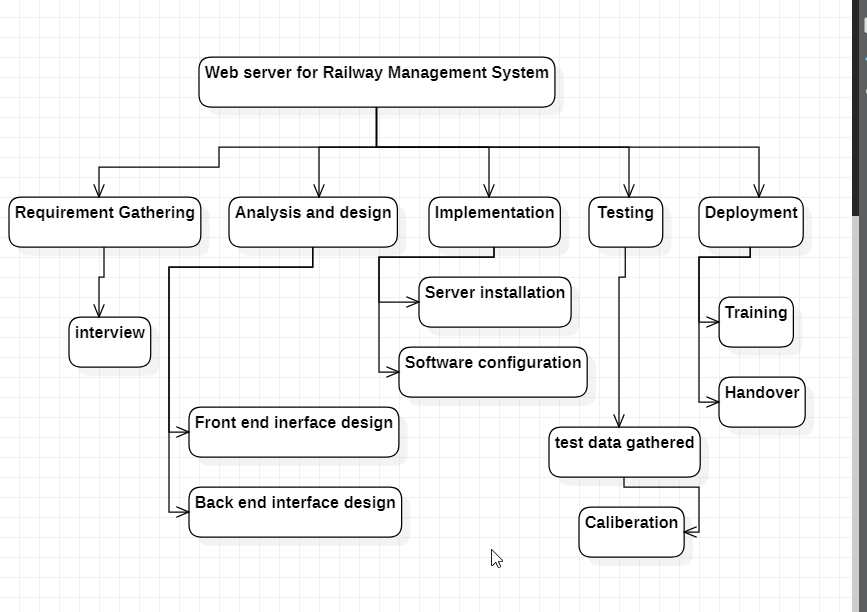
****

**Infrastructure requirements**

****

**WBS and Risk Identification**

* Work breakdown structure or WBS in project management is a method for getting a complex, multi-step project done. It’s a way to divide and conquer large projects so you can get things done faster and more efficiently.
* The goal of a WBS is to make a large project more manageable. Breaking it down into smaller chunks means work can be done simultaneously by different team members, leading to better team productivity and easier project management.
* Work breakdown structure is a hierarchical tree structure that outlines your project and breaks it down into smaller, more manageable portions.



**Distribution of work**

* Schedule structure: manages the timetable of arrival and departure of trains
* Login System: It is used to manage the login details
* Customer System: It manages the user details and information
* User System: It is used to manage user details and information
* Fare and Booking: It manages the information about fare and booking
* Integration with Payment portals: Integrates the website with most popular payment options
* Standard testing: Testing the System on compilation

**System Architecture**

**Class diagram**

Railway Management System Class Diagram describes the structure of a Railway Management System classes, their attributes, operations (or methods), and the relationships among objects. The main classes of the Railway Management System are Stations, Booking, Customers, Trains, Timetable, Employee.

#### Classes of Railway Management System Class Diagram:

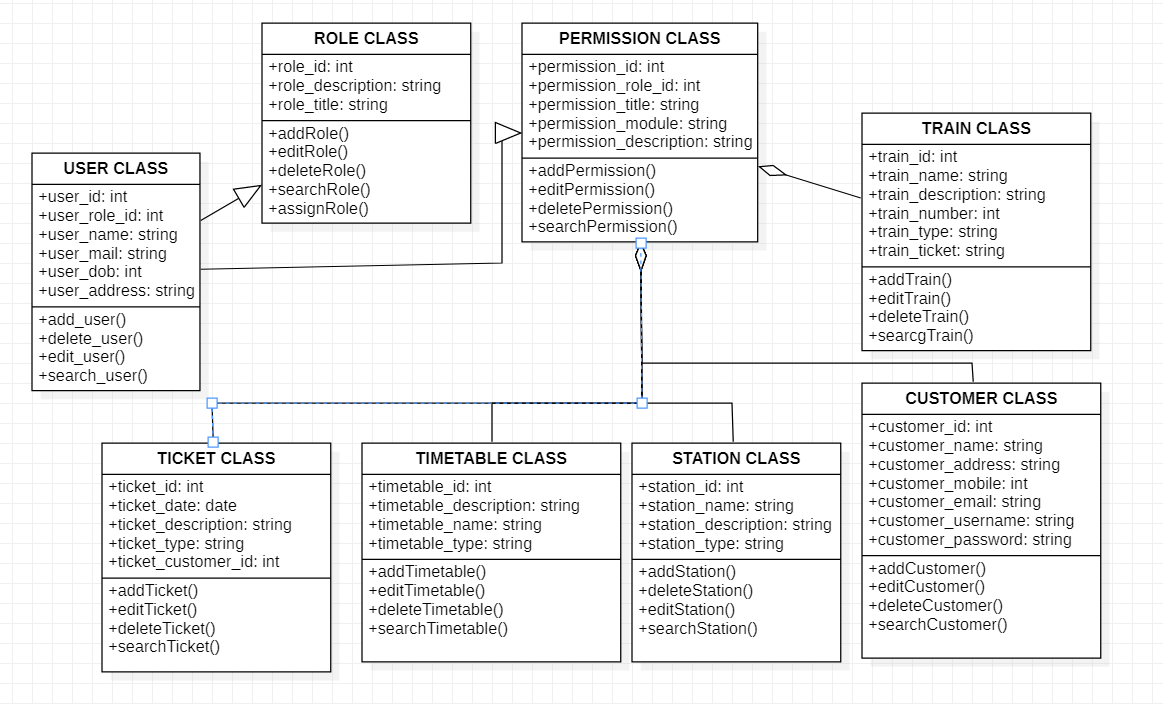
* **Stations Class** : Manage all the operations of Stations
* **Booking Class** : Manage all the operations of Booking
* **Customers Class** : Manage all the operations of Customers
* **Trains Class** : Manage all the operations of Trains
* **Timetable Class** : Manage all the operations of Timetable
* **Employee Class** : Manage all the operations of Employee

#### Classes and their attributes of Railway Management System Class Diagram:

* **Stations Attributes** : station\_id, station\_name, station\_type, station\_description
* **Booking Attributes** : booking\_id, booking\_title, booking\_type, booking\_ticket, booking\_date, booking\_description
* **Customers Attributes** : customer\_id, customer\_name, customer\_mobile, customer\_email, customer\_username, customer\_password, customer\_address
* **Trains Attributes** : train\_id, train\_name, train\_number, train\_seat\_number, train\_ticket, train\_type, train\_description
* **Timetable Attributes** : timetable\_id,rain\_id, timetable\_name, timetable\_type, timetable\_description timetable\_t
* **Employee Attributes** : employee\_id, employee\_name, employee\_mobile, employee\_email, employee\_username, employee\_password, employee\_address

#### Classes and their methods of Railway Management System Class Diagram:

* **Stations Methods** : addStations(), editStations(), deleteStations(), updateStations(), saveStations(), searchStations()
* **Booking Methods** : addBooking(), editBooking(), deleteBooking(), updateBooking(), saveBooking(), searchBooking()
* **Customers Methods** : addCustomers(), editCustomers(), deleteCustomers(), updateCustomers(), saveCustomers(), searchCustomers()
* **Trains Methods** : addTrains(), editTrains(), deleteTrains(), updateTrains(), saveTrains(), searchTrains()
* **Timetable Methods** : addTimetable(), editTimetable(), deleteTimetable(), updateTimetable(), saveTimetable(), searchTimetable()
* **Employee Methods** : addEmployee(), editEmployee(), deleteEmployee(), updateEmployee(), saveEmployee(), searchEmployee()

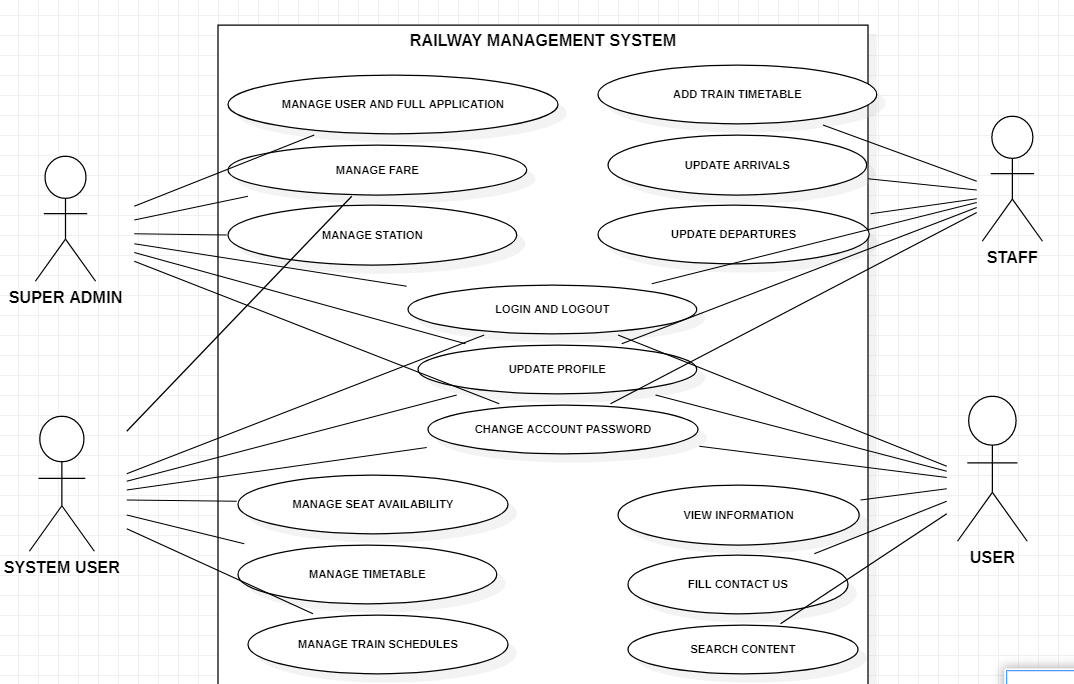
****

**Use Case diagram**

This Use Case Diagram is a graphic depiction of the interactions among the elements of Railway Management System. It represents the methodology used in system analysis to identify, clarify, and organize system requirements of Railway Management System. The main actors of Railway Management System in this Use Case Diagram are: Super Admin, System User, Station Masters, Anonymous Users, who perform the different type of use cases such as Manage Trains, Manage Timetable, Manage Fare, Manage Stations, Manage Seats Availability, Manage Routes, Manage Train Schedule, Manage Users and Full Railway Management System Operations. Major elements of the UML use case diagram of Railway Management System are shown on the picture below.

#### The relationships between and among the actors and the use cases of Railway Management System:

* **Super Admin Entity** : Use cases of Super Admin are Manage Trains, Manage Timetable, Manage Fare, Manage Stations, Manage Seats Availability, Manage Routes, Manage Train Schedule, Manage Users and Full Railway Management System Operations
* **System User Entity** : Use cases of System User are Manage Trains, Manage Timetable, Manage Fare, Manage Stations, Manage Seats Availability, Manage Routes, Manage Train Schedule
* **Station Masters Entity** : Use cases of Station Masters are Add Train Timetable, View Schedules, Update Arrivals, Update Departures
* **Anonymous Users Entity** : Use cases of Anonymous Users are View Information, Fill Contact Us, Search Content

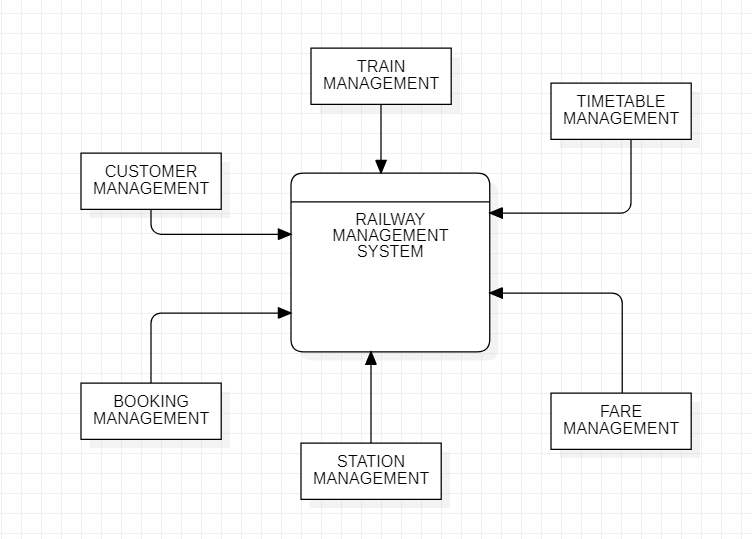
****

**DFD diagram**

is the Zero Level DFD of Railway Management System, where we have elaborated the high level process of Railway. It’s a basic overview of the whole Railway Management System or process being analyzed or modeled. It’s designed to be an at-a-glance view of Booking,Customers and Ticket showing the system as a single high-level process, with its relationship to external entities of Trains,Timetable and Fare. It should be easily understood by a wide audience, including Trains,Fare and Booking In zero level DFD of Railway Management System, we have described the high level flow of the Railway system.

#### High Level Entities and process flow of Railway Management System:

* Managing all the Trains
* Managing all the Timetable
* Managing all the Fare
* Managing all the Stations
* Managing all the Booking
* Managing all the Customers
* Managing all the Ticket This

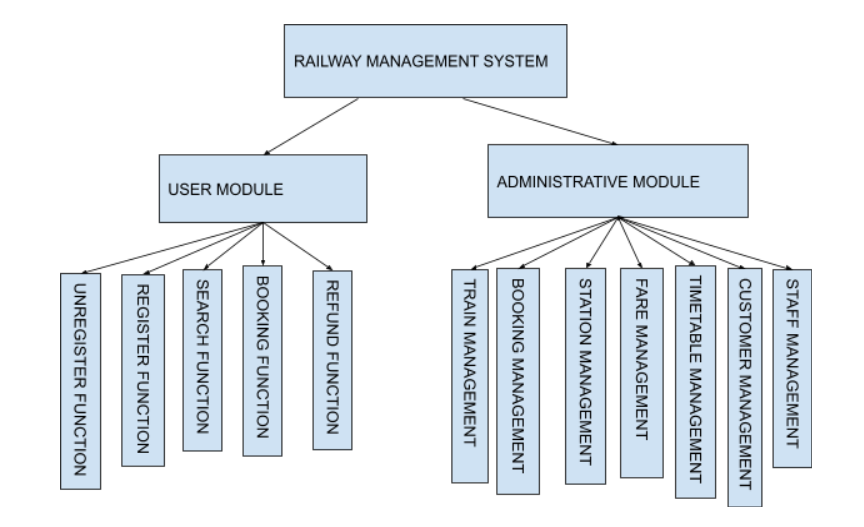
****

**First level DFD diagram**

First Level DFD (1st Level) of Railway Management System shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the Railway Management System system as a whole. It also identifies internal data stores of Ticket, Customers, Booking, Stations, Fare that must be present in order for the Railway system to do its job, and shows the flow of data between the various parts of Trains, Fare, Customers, Ticket, Booking of the system. DFD Level 1 provides a more detailed breakout of pieces of the 1st level DFD. You will highlight the main functionalities of Railway.

#### Main entities and output of First Level DFD (1st Level DFD):

* Processing Trains records and generate report of all Trains
* Processing Timetable records and generate report of all Timetable
* Processing Fare records and generate report of all Fare
* Processing Stations records and generate report of all Stations
* Processing Booking records and generate report of all Booking
* Processing Customers records and generate report of all Customers
* Processing Ticket records and generate report of all Ticket

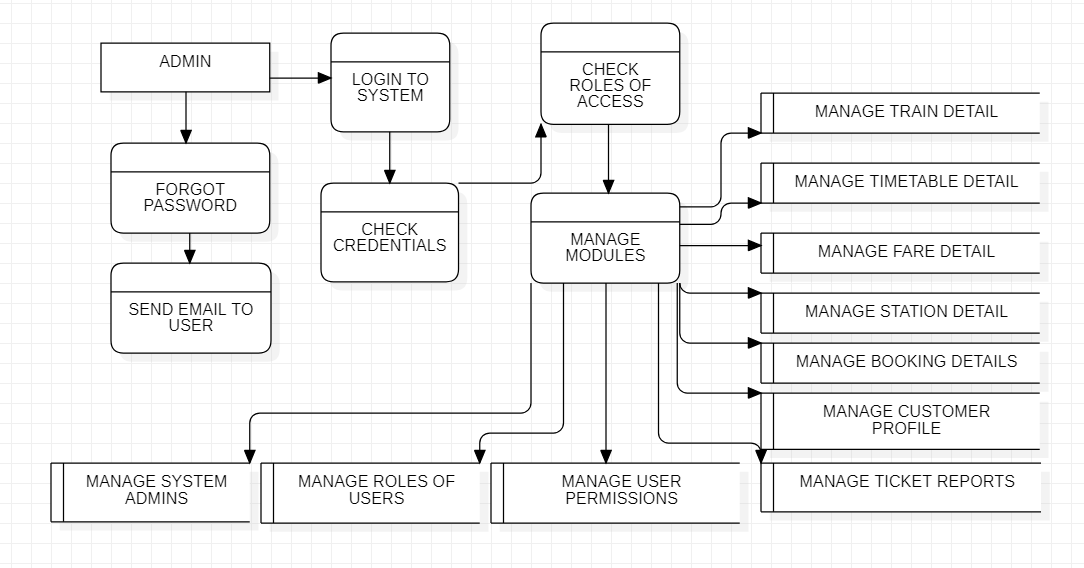
****

**Second level DFD diagram**

DFD Level 2 then goes one step deeper into parts of Level 1 of Railway. It may require more functionalities of Railway to reach the necessary level of detail about the Railway functioning. First Level DFD (1st Level) of Railway Management System shows how the system is divided into sub-systems (processes). The 2nd Level DFD contains more details of Ticket, Customers, Booking, Stations, Fare, Timetable, Trains.

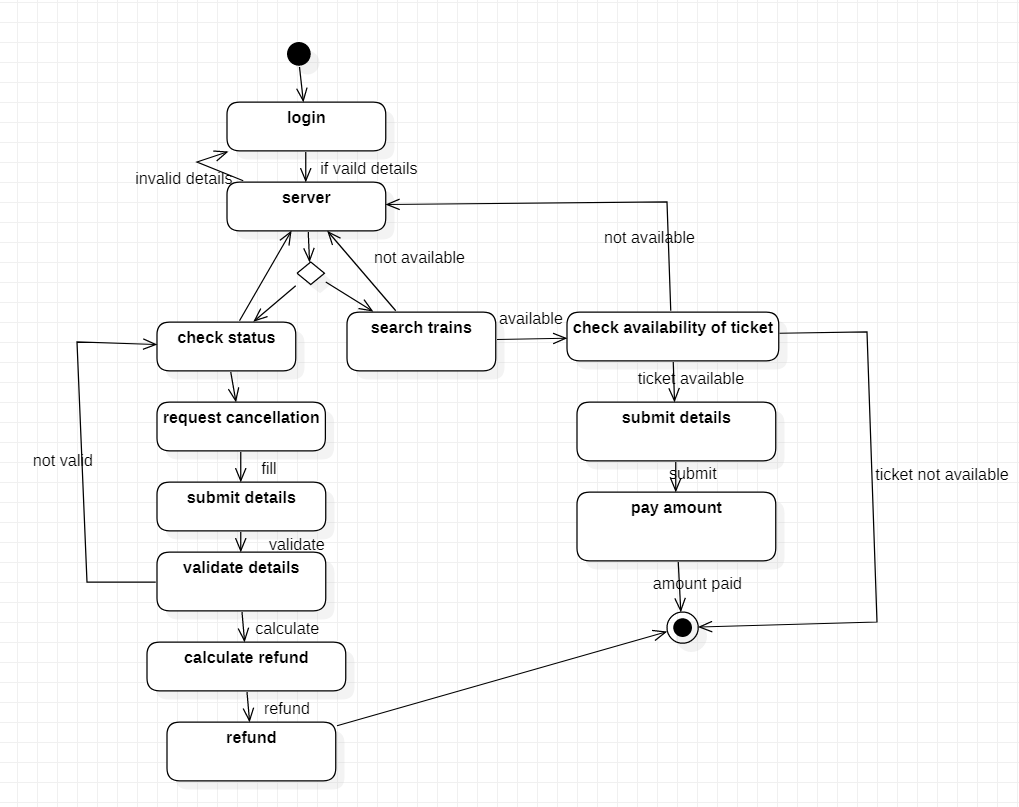
#### Low level functionalities of Railway Management System

* Admin logins to the system and manage all the functionalities of Railway Management System
* Admin can add, edit, delete and view the records of Trains, Fare, Booking, Ticket
* Admin can manage all the details of Timetable, Stations, Customers
* Admin can also generate reports of Trains, Timetable, Fare, Stations, Booking, Customers
* Admin can search the details of Timetable, Booking, Customers
* Admin can apply different level of filters on report of Trains, Stations, Booking
* Admin can tracks the detailed information of Timetable, Fare, Stations, , Booking

****

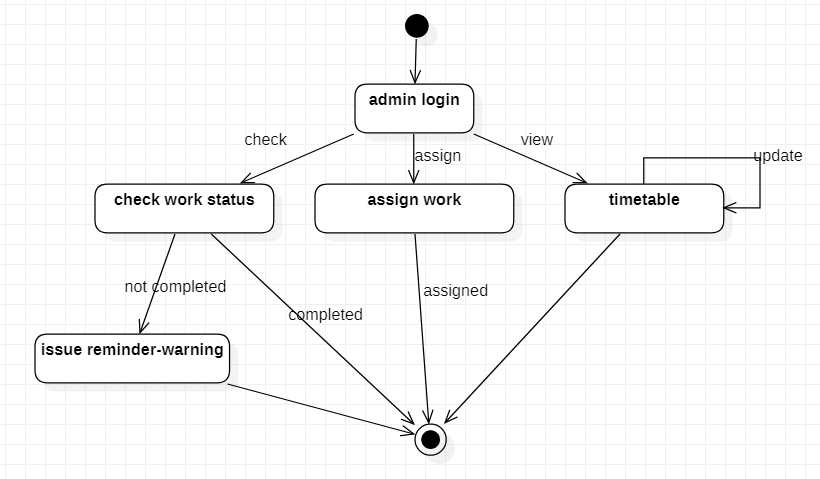
**State diagram for reservation**

A **state diagram** is a type of **diagram** used in computer science and related fields to describe the behavior of systems. This state diagram for reservation shows how the user can login to the server and then can perform activities like checking their train status, searching for trains, checking ticket availability, booking tickets and cancelling tickets.  All the pages such as Booking, Payment and cancellation are secure and user can access these page after login.

****

**State diagram for admin**

This diagram shows how the admin can login and the check the work status, assign work, view and edit the timetable. All pages are secure and can be accessed after logging in.

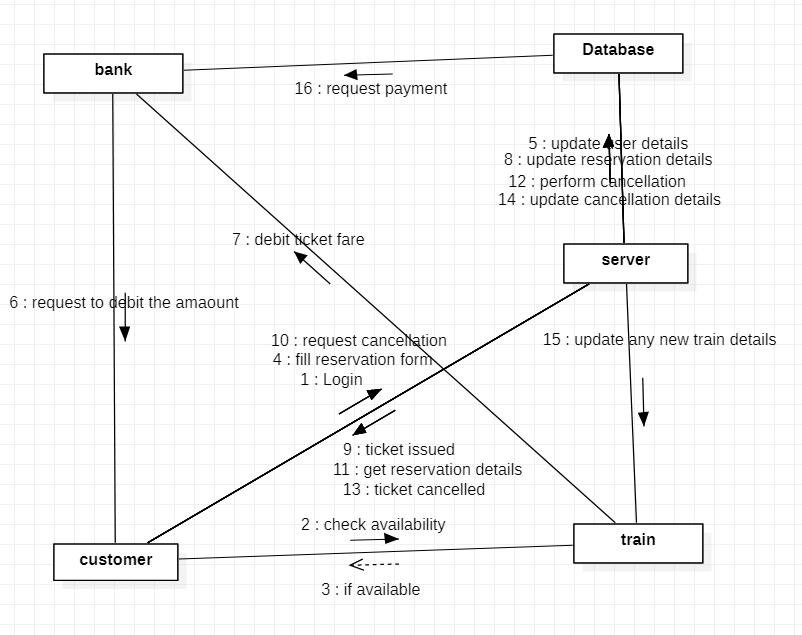
****

**Collaboration diagram for reservation**

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software [objects](https://searchapparchitecture.techtarget.com/definition/object). These diagrams can be used to portray the dynamic behavior of a particular [use case](https://searchsoftwarequality.techtarget.com/definition/use-case) and define the role of each object. Collaboration diagrams are created by first identifying the structural elements required to carry out the functionality of an interaction. A model is then built using the relationships between those elements.

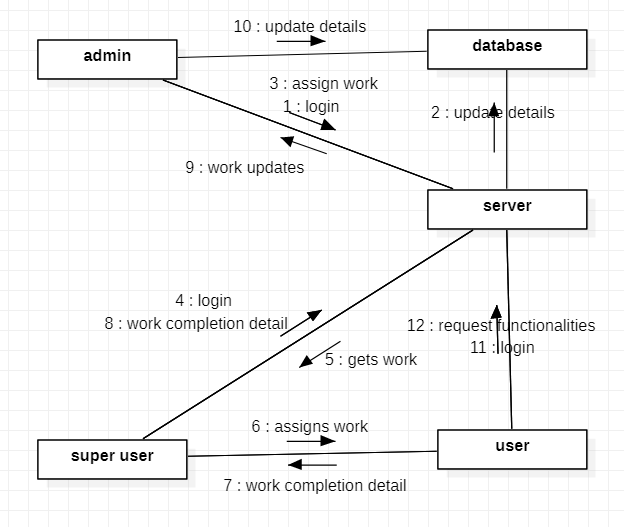
This collaboration diagram shows the relation between different objects of a railway reservation system.

We have 5 major objects i.e. customer, train, server, database and bank. The customer can login onto the server where he/she can check for trains, book tickets, cancel tickets, make the payment and get refund.

****

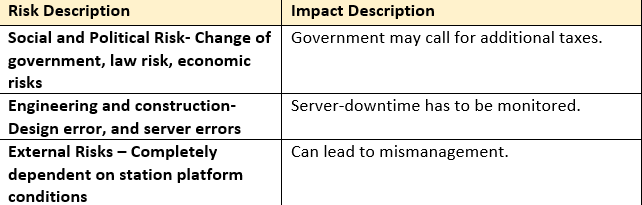
**Collaboration diagram for admin**

The administrative part of the project include the Admin, the super user and the user. The Admin has the major role of assigning work to the super user and maintaining the system. He can also update details on the database. The super user gets the work which he again divides and assign amongst the users. The users then complete the work and send the details to the super user who updates the details on the server. The user can also request required functionalities from the server. The work completion details provided by the super user also gets updates on the database.

****

**Risk Identification**

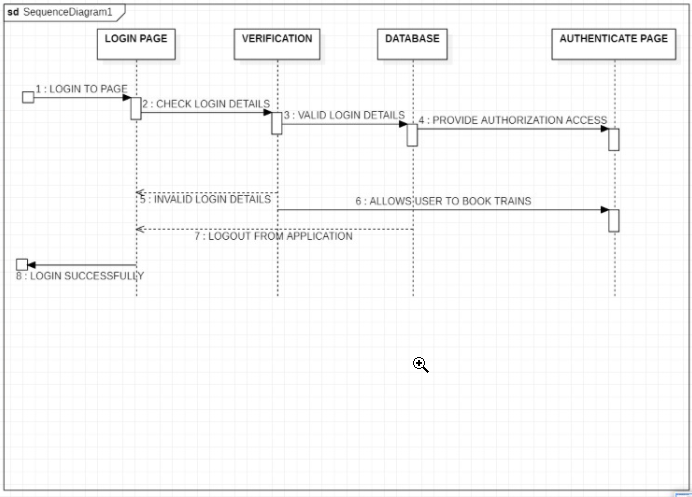
* There is High level of external risks as the system is completely dependent on how the trains work on physical Station Platform.
* No financial risk as the system is independent and cost free.
* Mid-level Engineering risk as Server-downtime has to be monitored.



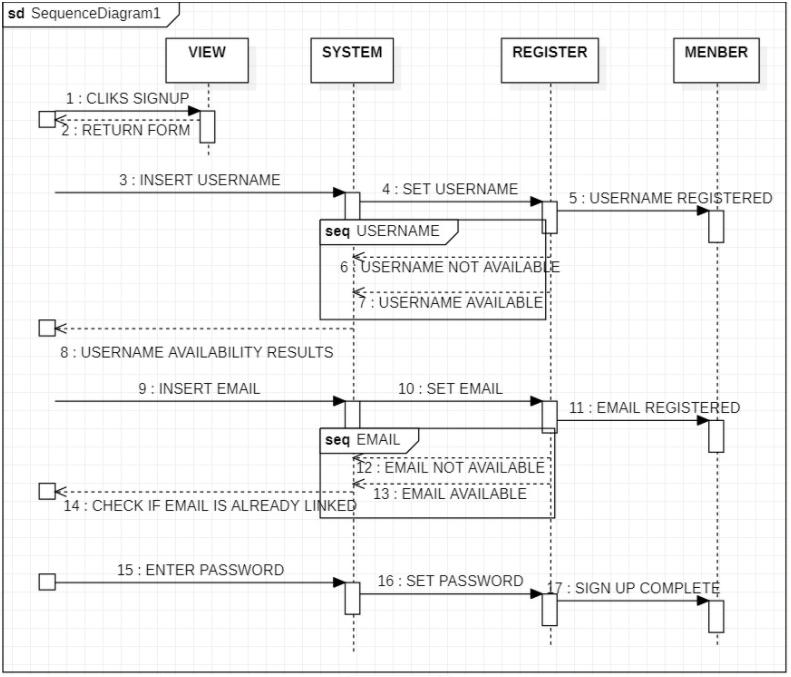
**Module-1**

* The module covered here is the combined code snippet for Sign-in and Sign-up system for user registration. This is done so that user will be recorded on the database and will be able to use further services once signed in.

**Login sequence diagram**

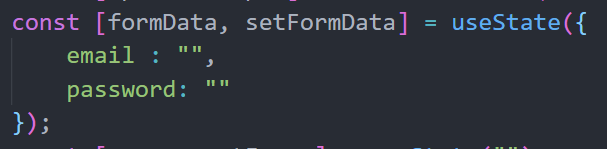
****

**Sign-up Sequence diagram**

****

**Code Snippet**

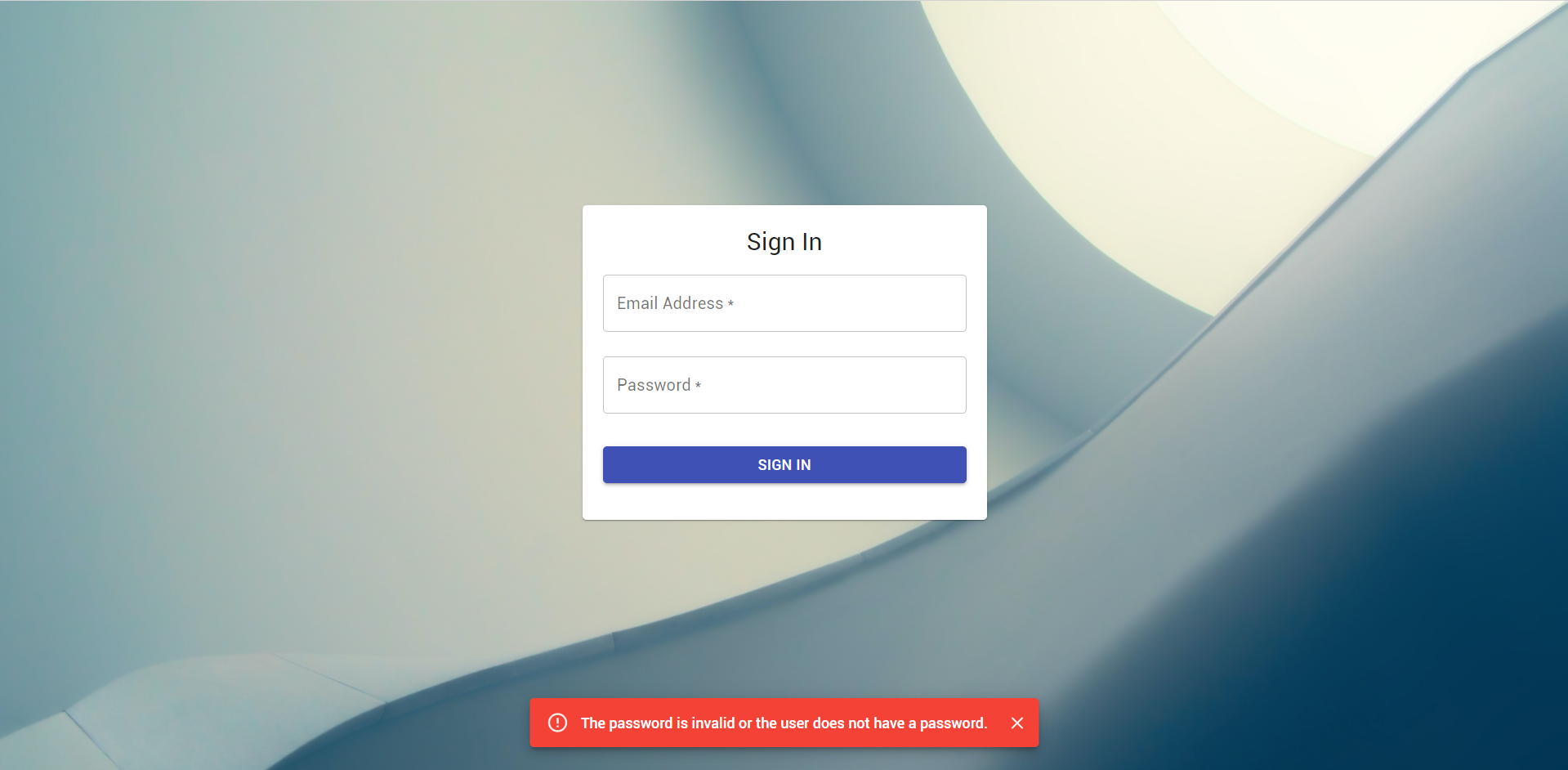
****

****

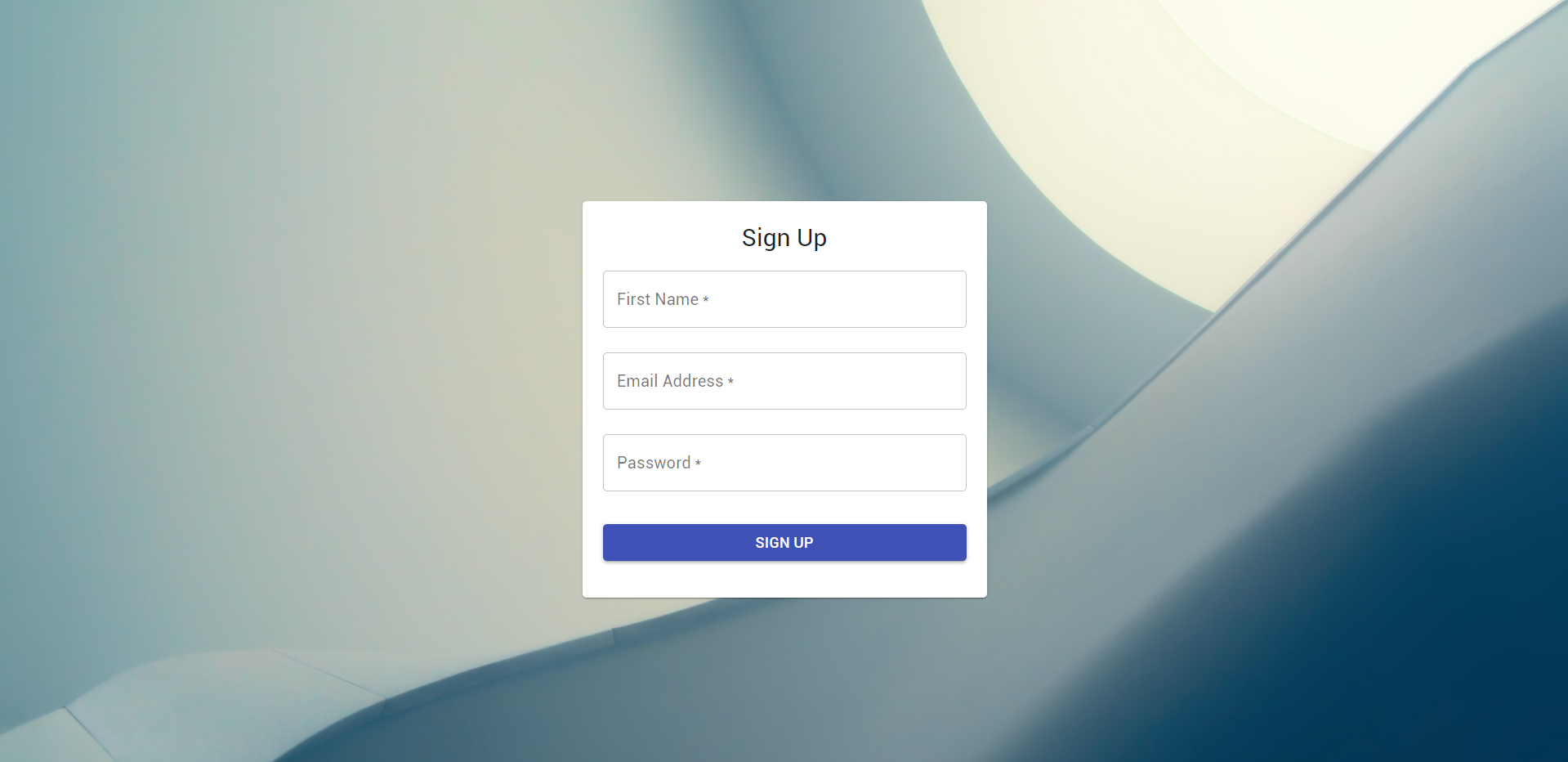
****

**Result**

**Sign-in page with error**

****

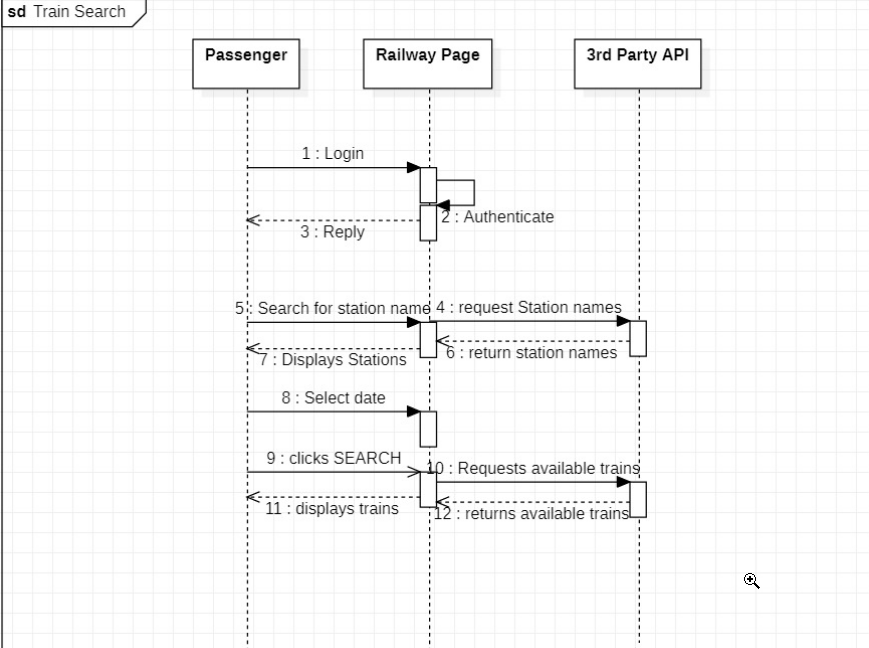
**Sign-up page**

****

**Module-2**

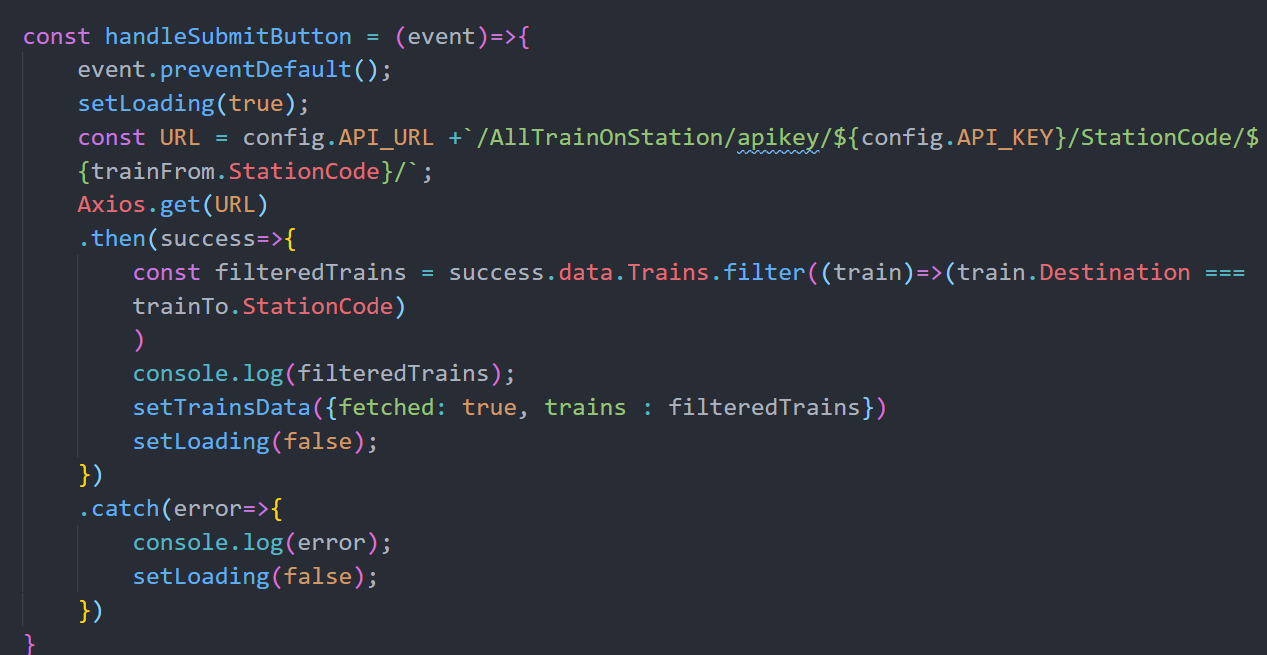
* The module covered here explains the layout and code of the homepage that will be available to the user when he/she visits the website. The homepage contains a navigation bar, a hero component and the option to search for available tickets (railway-search component)

**Sequence diagram**

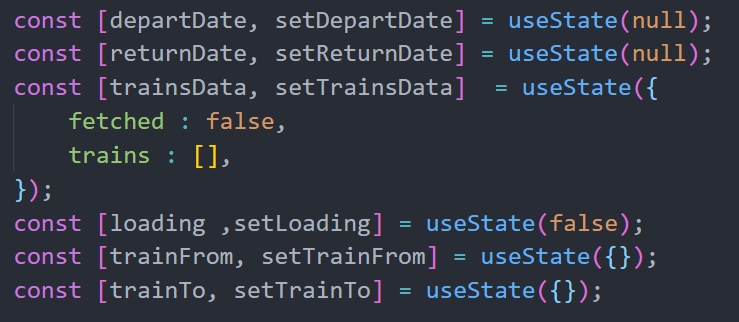
****

**Code Snippet**

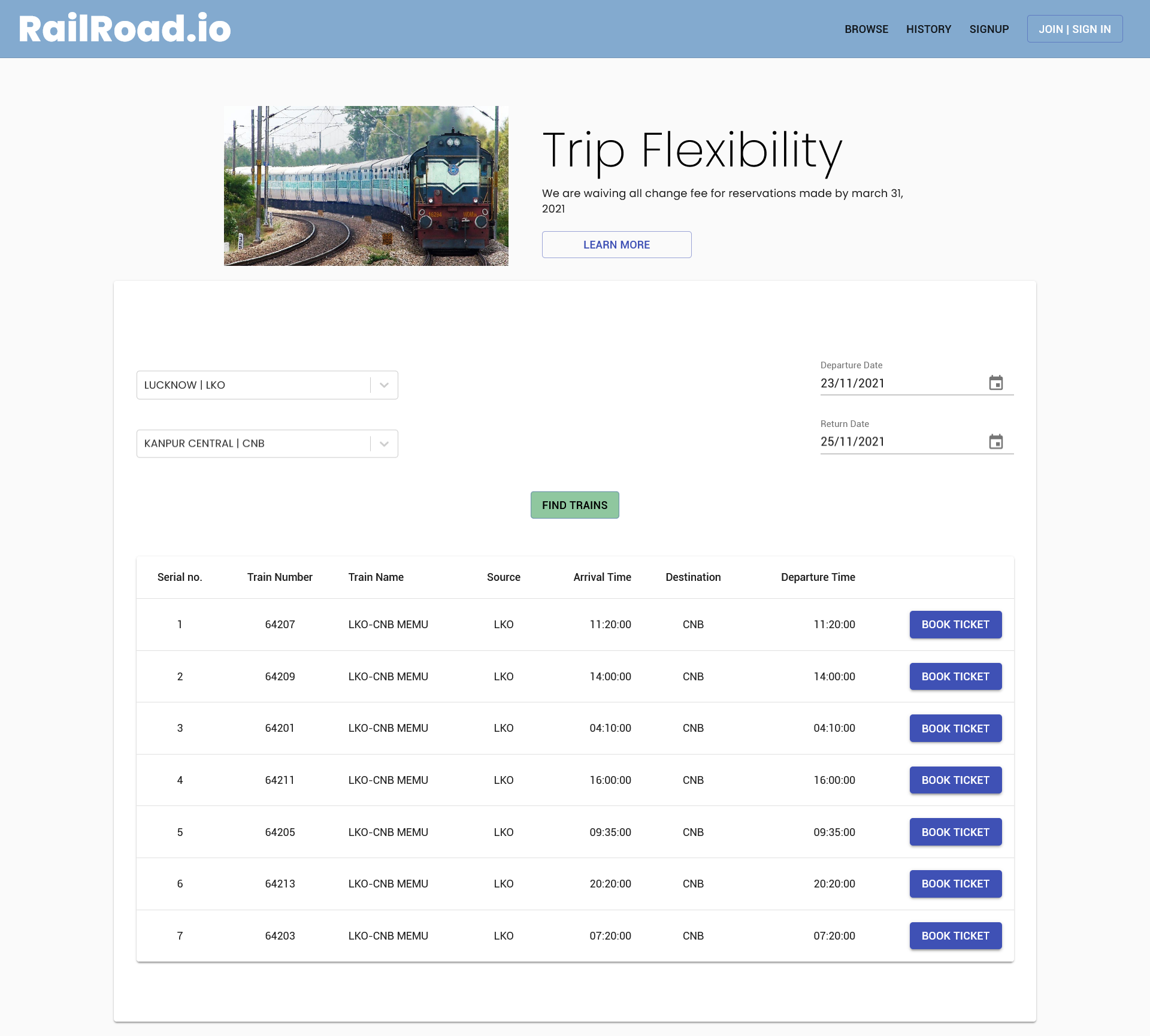
* This is the code for the function that performs the train search and api integration when the user will click the search button.



* Train details are stored in these variables.



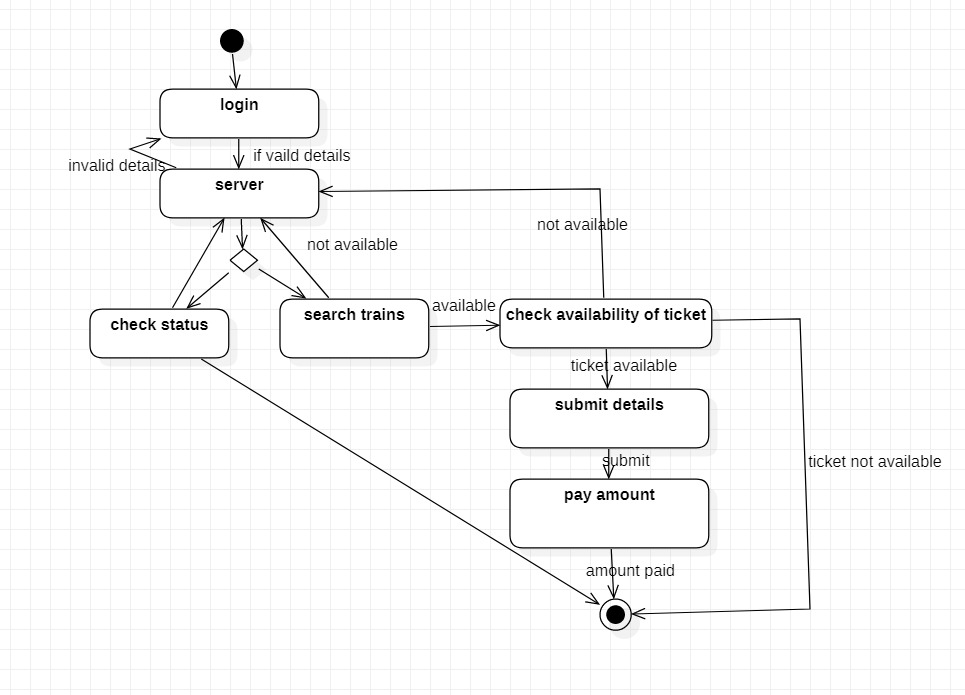
**Result**

****

**Module-3**

* The module covered here explains the layout and code of the Browse page whenever he/she performs login or signs up for the first time. The browse page contains a navigation bar, a dashboard component and the option to search for available tickets (railway-search component). The Checkout page shows the details of the selected train as well as its timetable on that day.

**Sequence diagram**

****

**Code Snippet**

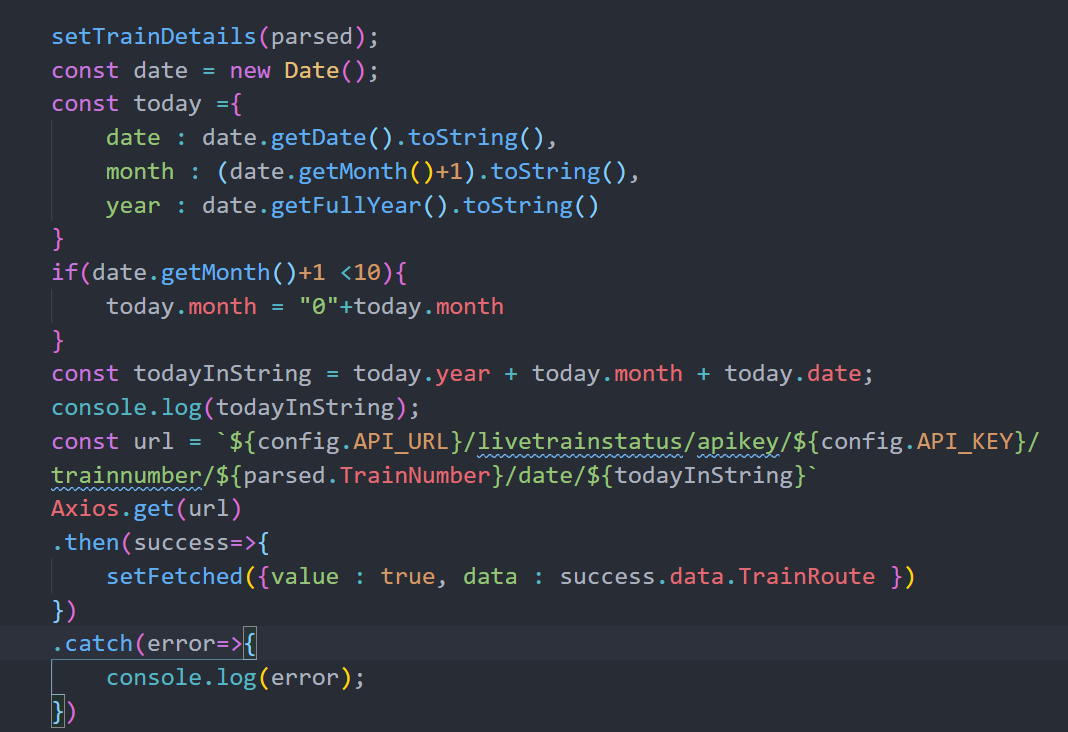
* Code to get all the booked trains from the database and to cancel any train

****

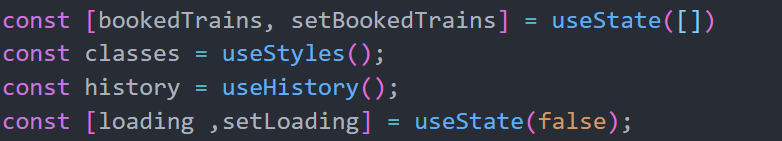
* Code for checkout function

****

* Code for date calculation and api call

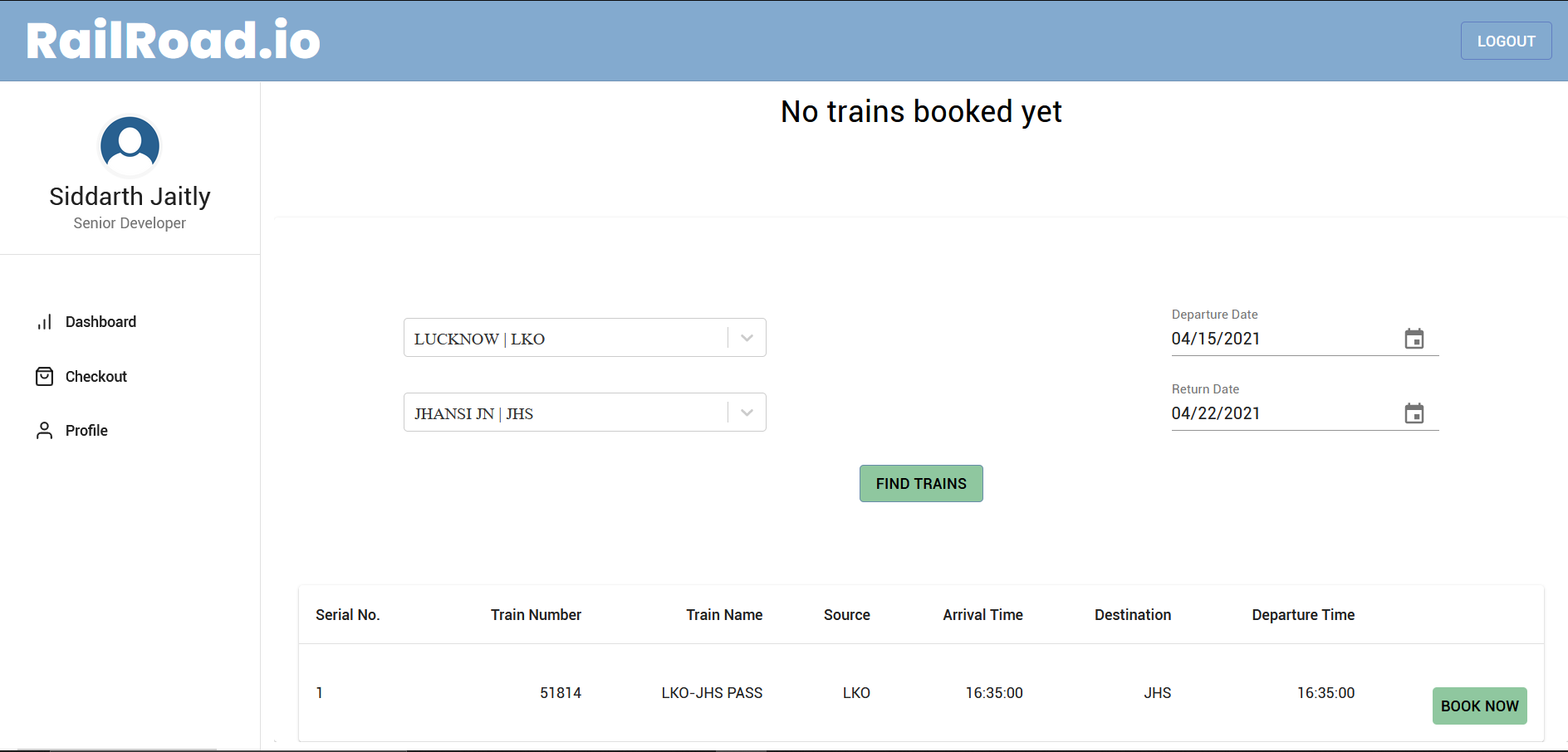
****

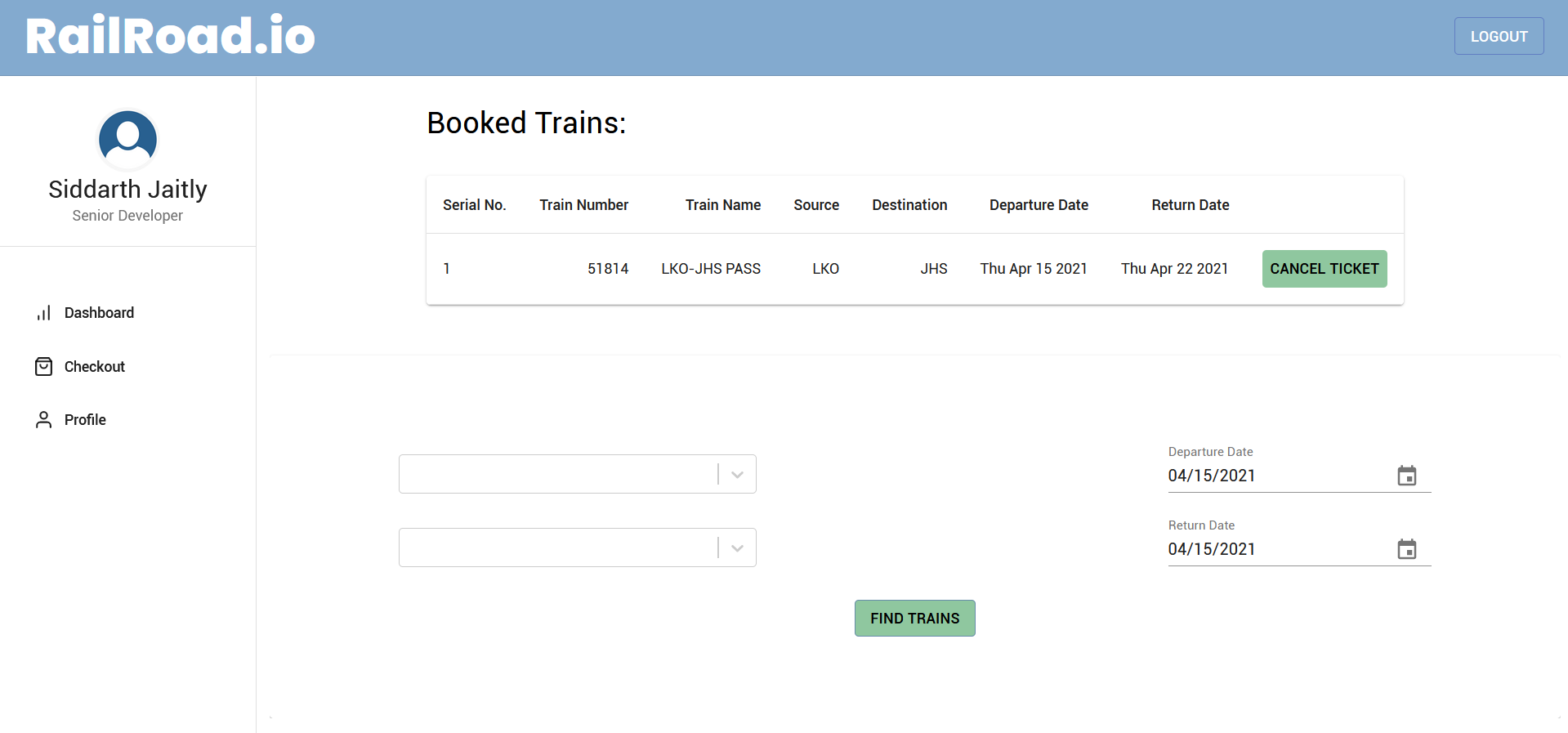
* Train details are stored in these variables



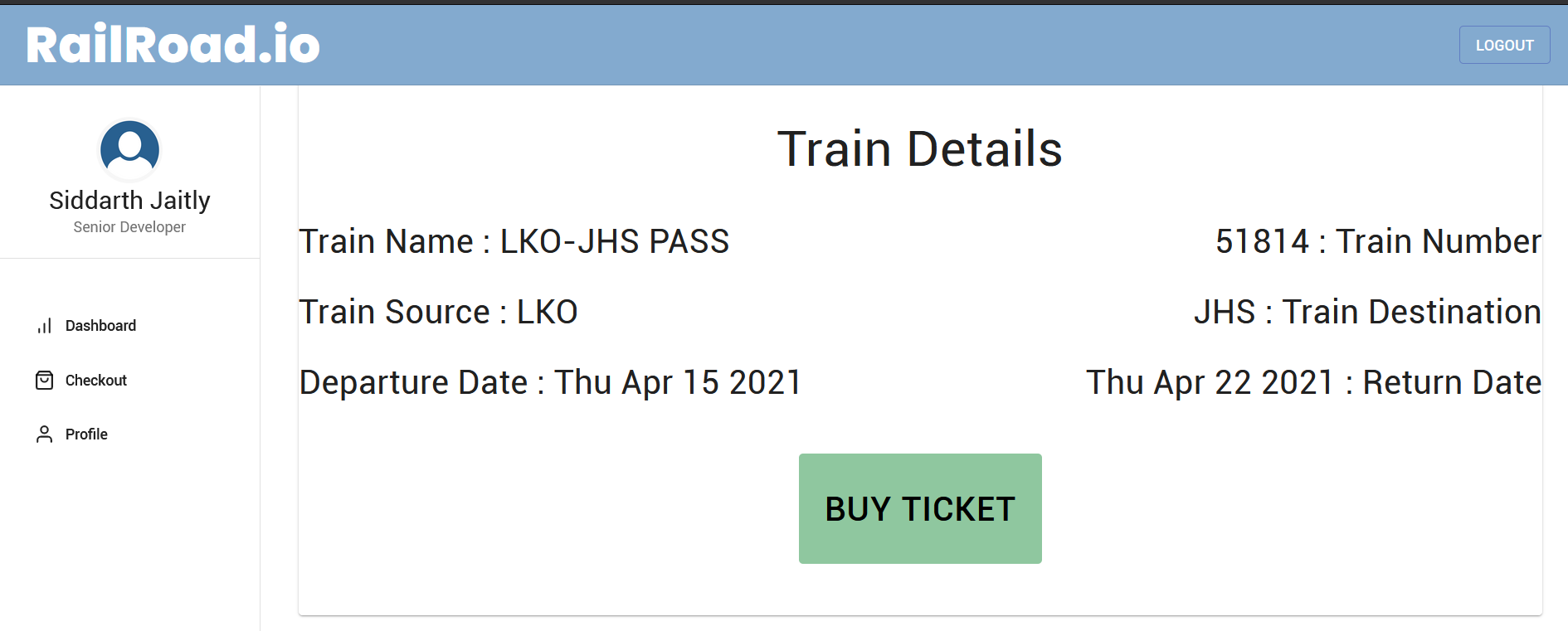
**Result**

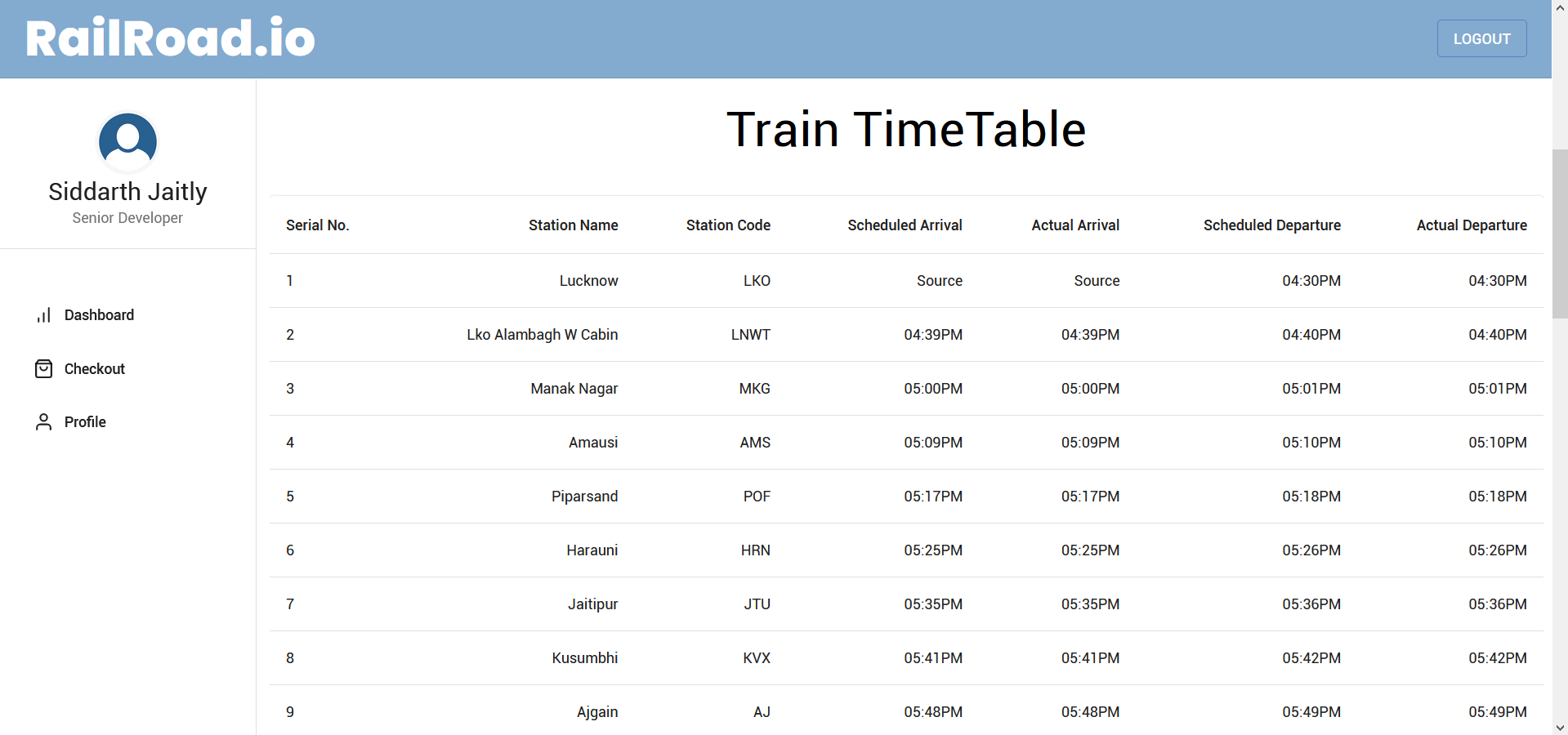
* **Browse Page**

****

****

* **Check out page**

****

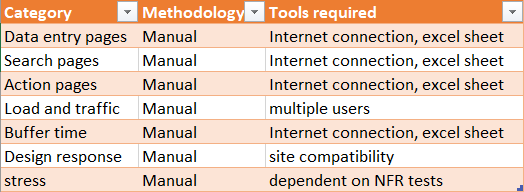


**Test Plan**

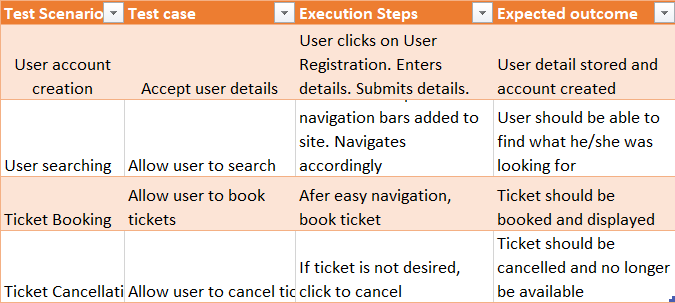
* The plan involves testing all modules working built together checking the efficiency of the entire product as whole.
* Scope of testing-

1. Functional modules to be tested: Sign in page, registration page, home page, browse page, checkout page.
2. Non-functional modules include load test(amount of users that can sign in), time test(checking buffer time), responsive test(checking screen sizes which do not work), internet test and stress test.

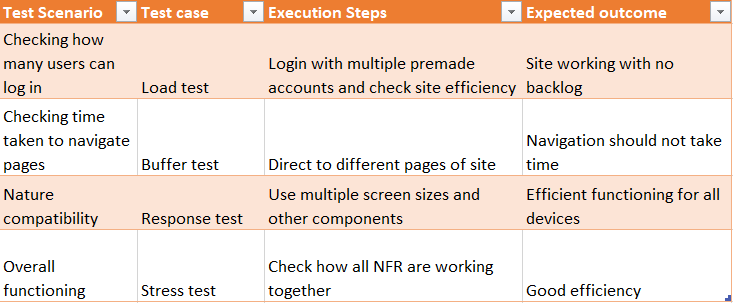
**Types of testing**

****

**Functional Test cases**

****

**Non-functional Test cases**

****

**Conclusion**

Successful website was made which allowed customers to navigate and buy tickets according to their transport wishes with the option to cancel any plan made.