

# Data Pirates

## Rail India

**A.**

### **What are we doing?..**

Indian Railways is the third largest Railway network in the world. With 20 million registered users, thousands of ticket bookings every second and Crores of passengers traveling every day, Railway services have difficulty in processing and analyzing big data.

We aim to build a Railway service system that can efficiently book train tickets for passengers, track live status of trains, present all trains at a given station and build useful statistics from a high volume and velocity of Railway data using databases.

### **Why a database?**

As we know there are a large number of trains and stations and passenger information, hence we'll be having a lot of data to work with, using a database in such cases will make tracking the data much easier. And while Implementing features such as listing out possible paths between two stations, databases help a lot as it can perform simultaneous processing over large sets of data.

A database will help in querying the data for various statistics, and also to manage user related information. That is, it would be required for seat availability and ticket booking.

### **What kind of db? and why?**

The database will include geospatial, relational and time-series data. Geospatial database is used for tracking the position of the train, and the time series data is used to find the actual arrival time of the train at each station. Relational database is used for storing information about trains, stations. We need Graph DB for finding a list of available trains between stations.

### **Domain**

Ticket booking system and statistics Web Application along with live location tracking facility.

**B.**

### **List the intended classes of users of your database system?**

Category of Users:-

1. Passengers
2. Admin
3. Users

Note: Passengers are the one who travel, and Users are the ones who book the tickets for the passengers ( and a user can book tickets for multiple passengers and view live status of the train ). Admin are used to represent the station managers and they have explicit control over the database ( e.g he can change/update/add the trains source and destination )

## UseCases:

### 1. Sign up for RailIndia

- A. **Description:** Users can create an account on RailIndia by providing their information. The data is validated and a new entry is added to **USER** entity if valid
- B. **Trigger:** (Human) This is performed when user fills the signup form in signup page
- C. **Primary Actor:** **USER**
- D. **Input:** Username, User email, Phone Number, Password, Age
- E. **Precondition:** User is in signup page and no user is logged in
- F. **Main success path:** If valid info is provided a new entry is added to User table
- G. **Exceptions:** If the user\_name is already present, an error message is displayed.
- H. **Post Conditions:** A new user entry is added.

### 2. Logging into RailIndia

- a. **Description:** Users can log into their account on RailIndia by providing their credentials . If the information is correct the user is taken to the user page else he stays on the login page.
- b. **Trigger:** (Human) This is performed when user fills the login form on login page
- c. **Primary Actor:** **USER**
- d. **Input:** Username/Email along with Password
- e. **Precondition:** User is in login page and no user is logged in
- f. **Main success path:** If valid info is provided user is logged in to the home page
- g. **Exceptions:** If the entered credentials are invalid, an error message is displayed.
- h. **Post Conditions:** A session based token is created and user is logged in

### 3. Book Ticket

- a. **Description:** Users can book tickets for 1 or more passengers . He can choose the seats.
- b. **Trigger:** (Human) When user fills booking form in booking page
- c. **Primary Actor:** **USER**
- d. **Input:** List of Passengers, age, sex, Start Station, End Station
- e. **Precondition:** User is in logged in
- f. **Main success path:** If valid info is provided , tickets are booked
- g. **Exceptions:** If the tickets were already booked by the time the user commits the booking, booking fails and the user is asked to retry the booking. If no of seats exceeds number of available seats then a warning message is displayed
- h. **Post Conditions:** A passenger/waiting\_passenger entry is added and a booking entry as well

### 4. View Available trains

- a. **Description:** When a user chooses start station and end station we show a list of available trains(direct trains and indirect trains with 1 halt in between them)
- b. **Trigger:** (Human) When user searches for available trains
- c. **Primary Actor:** **USER**
- d. **Input:** Source, Destination, Date
- e. **Precondition:** User is in logged in
- f. **Main Success path:** A list of available trains is shown
- g. **Exceptions :** If station is invalid then error message is displayed
- h. **Post Condition :** **None**

### 5. View Stations

- a. **Description:** A user can view stations on india map .Hovering on a station displays list of trains that stops at that station

- b. **Trigger:** (Human) The map is always displayed in Map page
- c. **Primary Actor:** **USER**
- d. **Input:** None
- e. **Precondition:** User is in logged in
- f. **Main Success path:** On hovering a station , its info is displayed
- g. **Exceptions :** If station is invalid then error message is displayed
- h. **Post Conditions:** None

## 6. Add a new train

- a. **Description:** Admin of railways can add a new train
- b. **Trigger:** (Human) When admin fills the new train form
- c. **Primary Actor:** **ADMIN**
- d. **Input:** Path of train, Train number, Train name, Timings
- e. **Precondition:** User is in logged in
- f. **Main Success path:** If valid data is provided then a new entry is created
- g. **Exceptions:** If the train\_id is already present, an error message is displayed.
- h. **Post Conditions:** A new entry is added in Train entity

## 7. View Train Schedule

- a. **Description:** Users can view schedule of train by selecting train number or name
- b. **Trigger:** (Human) When user fills train number or name in schedule form
- c. **Primary Actor:** **USER**
- d. **Input:** Train number or Train name
- e. **Precondition:** User is logged in
- f. **Main success path:** If correct train Id and name are provided Train's expected schedule is displayed and the train path is shown on a map
- g. **Exception :** If train number or name is invalid error is shown
- h. **Post Conditions :** User is logged in

## 8. Update Live Locations

- a. **Description:** Admin can update location of train when a train crosses the station
- b. **Trigger:** (Human) When train passes through a station the station manager updates the train live status information (latitude and longitude and recent\_station is updated)
- c. **Primary Actor:** **ADMIN**
- d. **Input:** Train number Station number , time, date
- e. **Precondition:** Train number, station number should be valid and User is logged in
- f. **Main success path:** If correct train Id and station number are provided then live location is updated
- g. **Exception :** If train number or station number is invalid error is shown
- h. **Post Conditions:** Train's live latitude and longitude is updated in Train\_instance database

## 9. View Live Locations

- a. **Description:** By selecting a train number and date of departure users can view the live running status of train
- b. **Trigger:** (Human) When a user searches for live status of train in live status page
- c. **Primary Actor:** **USER**
- d. **Input:** Train number / name along with date of departure
- e. **Precondition:** Train number, should be valid and User is logged in
- f. **Main success path:** If correct train Id is provided then live location is shown
- g. **Exception :** If train number is invalid error is shown

h. **Post Conditions:** None

#### **10. Find Nearest Railway Station**

- a. **Description:** Users can find their nearest railway station
- b. **Trigger:** (Human) When user searches for nearest railway station
- c. **Primary Actor:** **USER**
- d. **Input:** Takes your current location
- e. **Precondition:** User is logged in
- f. **Main success path:** :A nearest railway station is shown
- g. **Exception :** If unable to fetch location then error message is displayed
- h. **Post Conditions:** None

#### **11. View ticket**

- a. **Description:** Users can find their previous bookings
- b. **Trigger:** (Human) When user views his history of bookings
- c. **Primary Actor:** **USER**
- d. **Input:** Ticket no
- e. **Precondition:** User is logged in
- f. **Main success path:** :User can view list of previous tickets
- g. **Exception :** If ticket number is invalid then error message is shown
- h. **Post Conditions:** None

#### **12. Statistics**

- a. **Description:** Users can view inflow and outflow at each station, number of trains per zone, number of trains per state, state wise influx and outflow. We will use Spark for this
- b. **Trigger:** (Automatic) When a user searches for statistics
- c. **Primary Actor:** **USER**
- d. **Input:** None
- e. **Precondition:** User is in logged in
- f. **Main Success Path:** On selecting a valid station, train statistics will be displayed
- g. **Exception:** if a station is not in the list of visited stations by all trains, nothing will be displayed
- h. **Post Conditions:** All statistics are shown

#### **13. Add a Station:**

- a. **Description:** Admin of railways can add a new station
- b. **Trigger:** (Human) When admin fills the new station form
- c. **Primary Actor:** **ADMIN**
- d. **Input:** station\_id, name, latitude, longitude, city, station, zone
- e. **Precondition:** Must be logged in as Admin
- f. **Main Success Path:**
- g. **Exceptions:** If the station\_id is already present, an error message is displayed.
- h. **Post Conditions:** A new entry is added in station entity

#### **14. Update Waiting List:**

- a. **Description:**Waiting List of a train is updated
- b. **Trigger:** Automatically triggered
- c. **Primary Actor:** **SYSTEM**
- d. **Input:** train\_number, waiting passengers id
- e. **Precondition:** Ticket must be booked for the required train number
- f. **Main Success Path:** If valid ticket is used, then waiting list is updated when seats get filled
- g. **Exceptions:** if ticket is not in waiting list, then nothing should be displayed
- h. **Post Conditions:** The waiting list is updated, which will further reflect for each ticket

### 15. Cancel Ticket:

- a. **Description:** Cancel a ticket that was already booked
- b. **Trigger:** (Human) On clicking "Cancel Ticket" Button available on frontend, ticket is canceled
- c. **Primary Actor:** USER
- d. **Input:** booking\_id
- e. **Precondition:** Ticket must be booked for the required train number
- f. **Main Success Path:** If valid ticket is used, then current ticket will get canceled
- g. **Exceptions:** If the booking\_id is not present, an error message is displayed.
- h. **Post Conditions:** All statistics are shown

### 16. Waiting list Position:-

- a. **Description:** Display the number of passengers ahead of the current user in the waiting list
- b. **Trigger:** (Automatic) When a user searches for his ticket booking status
- c. **Primary Actor:** USER
- d. **Input:-** Train number, booking Id
- e. **Precondition:-** Ticket must be booked for the required train number
- f. **Main Success Path:-** If valid ticket is used, then current waiting list position is displayed
- g. **Exceptions:** if ticket is not in waiting list, then nothing should be displayed
- h. **Post Conditions:** Position in the waiting list is displayed

Note: We are also planning to add GPS based live location system which gives current location .What we are currently doing the location is updated manually by a station manager at a railway station

C.

### Identify 10-15 main "things"

#### List of Entities:

#### a. **Passenger:**

Weak entity identified by the booking entity.

Passenger\_ID int,

Name varchar,

Seat No. int,

Age int,

Sex varchar,

Waiting\_pref\_no int

**Primary Key - Passenger\_ID**

#### b. **User:**

ID int,

Name varchar,

age int,

is\_admin int,

phone int,

email varchar,

sex varchar

**Primary Key - User ID**

#### c. **Booking:**

Primary Key - Booking Id

booking\_Date date\_time

**d. Train:**

Train No int,  
Name varchar,  
Capacity int,  
num\_stations int  
Primary Key - train\_no

**e. Train Schedule:**

Weak entity identified by both Train and Station.  
path\_index, Expected Arrival time, Expected Departure Time  
Primary Key - { train\_no, path\_index, station\_no}  
Note: Index is used to refer to the position of the station in the path followed by the train

**f. Path:**

Weak entity identified by both Train\_ID and Station.  
path\_index, Expected Arrival time, Expected Departure Time, actual arrival time, actual departure time  
Primary Key - { train\_no, path\_index, station\_no}  
Note: Index is used to refer to the position of the station in the path followed by the train

**g. Train\_instance**

Weak entity defined by train  
Primary key - date  
available seats, price, latitude, longitude, last\_passed\_index

**h. Station:**

Station ID, City, State, Longitude, Latitude, zone, name  
Primary Key - station\_id

**List of Relationships**

**1. Train\_path**

Identifying relationship of path by TRAIN\_instance entity.

**2. Station\_path**

Identifying relationship of path by Station entity.

**3. booked\_train**

Relationship from booking to instance

**4. user\_booked**

Acts as a foreign key for the user id.

**5. passengers\_booking**

Identifying relationship of Passengers by Booking entity.

**6. path\_schedule**

identifying relationship for train schedule

**7. train\_date**

Identifying relationship of TRAIN\_ID by train

**8. inst\_schedule**

identifying relationship for train schedule

**9. distance**

Relationship of distance from source to destination

**10. station\_source\_ref, station\_dest\_ref, start\_station, end\_station**

Act as a foreign key to Station ID

## **D. Forms**

### **Login**

Input : email, Password

Output: Valid the info and take to home page

### **Signup**

Input: email, name, age, sex, phone, password

Output: Create entry in User database

### **Book Ticket**

Input: Passenger Details, Train No, Date, Time

Output: Update bookings Table

### **Logout**

Input: None

Output: Take back to login page

### **ViewStations**

Input: Station Name

Output: Station Geospatial info

### **Add a train**

Input: Train number Train name Train route

Output: New train is added

### **View Ticket**

Input: Ticket Id

Output: Ticket Details

## **DATASETS :**

[Train database](#)

[Indian Railway Database](#)

## **E.**

### **Website Design:**

The layout of the webpage in detail is given in the next few pages

# User Pages

Login Page

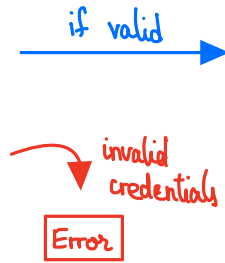
Email

Password

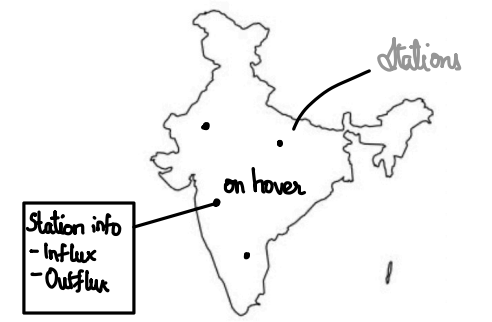
Login

Register

Forgot Password



Homepage



The homepage will display all the stations along with a heat map based on the density of trains in each locality.

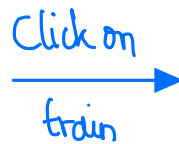
Accessible through Navigation Bar -

Booking Page

Train No.  
From To  
Date

\_\_\_\_\_

List of trains  
.  
.  
.



Confirmation Page

Train No.  
Available Seats  
Waiting List

Add Passenger

Final Amount

## Booking page

The user is shown a list of trains based on the input source and final destination. The user can then pick a train convenient for them along with the date of travel. Then, the user is navigated to the confirmation page after the data is validated. We shall display the number of available seats along with the waiting list.

In the confirmation page, the user can add one or more passengers using their name, age and sex. The number of passengers should be less than the number of available seats, and the total amount will be displayed to the users. If the seat capacity is full, then the passengers will be added to a waiting list.

View Ticket Page

PNR

List of Passengers ☐

☐

.

Cancel Booking

View Trains

Train	Train Name	Expected Time

## View Ticket Page

The User can view the details of their ticket on this page. They can also cancel the tickets for each passenger individually for the trains that have not yet started. The waiting list is updated accordingly

## List of Trains Page

Each train will have a link to its information and schedule.

Click on train



Train Info Page			
Train No.			
Train Name.			
Start Station.			
...			
Path			
Index	Station	Expected Time	Actual Time

Click  
Station

Click  
Train

Station Info Page		
Station Code		
Station Name		
Other details		
...		
List of close by/recent trains		
Train	Coming From /Going To	ETA / Departure time

### Train Info Page

It displays the train info along with its path. This page will include live tracking of the train where it shows the expected time of arrival at each station. Every station will lead to the station info page on click.

### Station Info Page

It displays information of a station along with the list of trains that have recently left station or are yet to arrive. Clicking on a train link would redirect the user to the train info page.

Profile Page			
Name			
Other Details			
...			
List of Bookings			
PNR	Date	Source	Destination

To view  
ticket  
page

Registration/Forgot Password Page

### Profile Info Page

It displays the user info along with the associated booking. Clicking on the booking will lead the user to the view ticket page.

These pages are accessible to the user when they are logged out. The layout is similar to that of the login page.

# Admin Pages

Common pages such as login , forgot password and registration would be very similar to that of the user. The admin can also view the list of trains, train info, station info and the homepage. However, the admin will consist of the following pages - Add Train, Add Station, Update Live Tracking

Add Train Page	
Train Number	_____
Train Name	_____
Expected Arrival	_____
Expected Departure	_____
<div>Add Stations in Path</div>	
<div>Add</div>	

## Add Train Page

The admin will be able to add new trains along with the stations in its path. The admin can choose a new station from the list of available stations.

Add Station Page	
Station Name	_____
State	_____
City	_____
Zone	_____
<div>Add</div>	

## Add Station Page

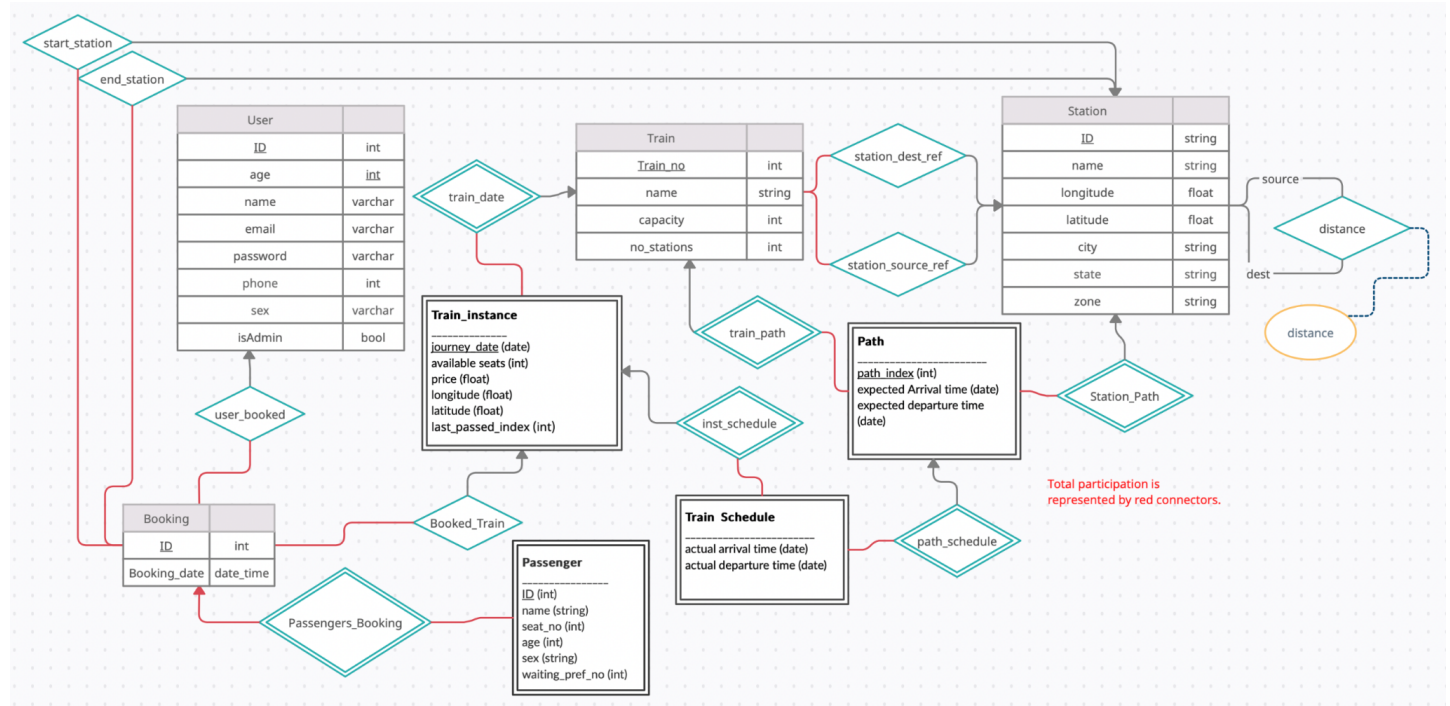
The admin will be able to add new stations on this page.

Update Live Location	
Station Code	_____
Train No.	_____
Arrival Time	_____
Dept Time	_____
Date	_____
<div>Submit</div>	

## Update Live Location

The administrator can manually set the arrival and departure times of each train in each station through this page. We also plan to implement a GPS based live tracking which would automate some of this process.

## ER DIAGRAM -



The link to the creately project is [here](#).

## TEAM

Sudhansh Peddabomma	- 190050118
Pokala Mohith	- 190050084
Poluparthi Preetham	- 190050085
Hitesh Kumar Punna	- 190050093