Data Pirates
Rail India

A.
What are we doing?..

We aim to create a railways tracking system similar to the IRCTC website. We plan to develop the features such as

1. Booking train tickets
2. Listing out possible trains between two stations (indirect routes upto 1 intermediate station as well)
3. Live Tracking for each train
4. Displaying all trains that are currently at a station
5. Predicting hotspots based on train's Influx and Outflux at each station
6. Use Indian geographic Map to represent all the railway stations

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 expected arrival time of the train at each station . Relational database is used for storing information about trains, 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:** 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. **Exceptions:** If the user_name is already present, an error message is displayed.
    F. **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**: This is performed when user fills the login form on login page
    c. **Primary Actor: USER**
    d. **Input:** Username/Email along with Password
    e. **Post Conditions**: A session based token is created
    f. **Exceptions:** If the entered credentials are invalid, an error message is displayed.

3. **Book Ticket**
    a. **Description:** Users can book tickets for 1 or more passengers . He can choose the seats.
    b. **Trigger:** When user fills booking form in booking page
    c. **Primary Actor: USER**
    d. **Input:** List of Passengers, age, sex, Start Station, End Station
    e. **Post Conditions:** A passenger entry is added and a booking entry as well
    f. **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.

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:** When user searches for available trains
    c. **Primary Actor: USER**
    d. **Input:** Source, Destination, Date
    e. **Main Success path:** A list of available trains is shown

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:** The map is always displayed in Map page
    c. **Primary Actor: USER**
    d. **Input:** None
    e. **Post Conditions:** None

6. **Add a new train**
    a. **Description:** Admin of railways can add a new train

b. **Trigger:** When admin fills the new train form
c. **Primary Actor: ADMIN**
d. **Input:** Path of train, Train number, Train name, Timings
e. **Post Conditions:** A new entry is added in Train entity
f. **Exceptions:** If the train_id is already present, an error message is displayed.

7. **<u>View Train Schedule</u>**
    a. **Description:** Users can view schedule of train by selecting train number or name
    b. **Trigger:** When user fills train number or name in schedule form
    c. **Primary Actor: USER**
    d. **Precondition:** Train name/number should already be present.
    e. **Input:** Train number or Train name
    f. **Main success path:** Train's expected schedule is displayed and the train path is shown on a map

8. **<u>Update Live Locations</u>**
    a. **Description:** Admin can update location of train when a train crosses the station
    b. **Trigger:** 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. **Precondition:** Train number, station number should be valid
    e. **Input:** Train number Station number time
    f. **Post Conditions:** Train's live latitude and longitude is updated

9. **<u>View Live Locations</u>**
    a. **Description:** By selecting a train number and date of departure users can view the live running status of train
    b. **Trigger:** 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

10. **<u>Find Nearest Railway Station</u>**
    a. **Description:** Users can find their nearest railway station
    b. **Trigger:** When user searches for nearest railway station
    c. **Primary Actor: USER**
    d. **Input:** None
    e. **Main success:**A nearest railway station is shown

11. **<u>View ticket</u>**
    a. **Description:** Users can find their previous bookings
    b. **Trigger:** When user views his history of bookings
    c. **Primary Actor: USER**
    d. **Input:** None
    e. **Main success:** User can view list of previous tickets

12. **<u>Statistics</u>**

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:** When a user searches for statistics
c. **Primary Actor: USER**
d. **Input:** None
e. **Post Conditions:** All statistics are shown

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
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

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

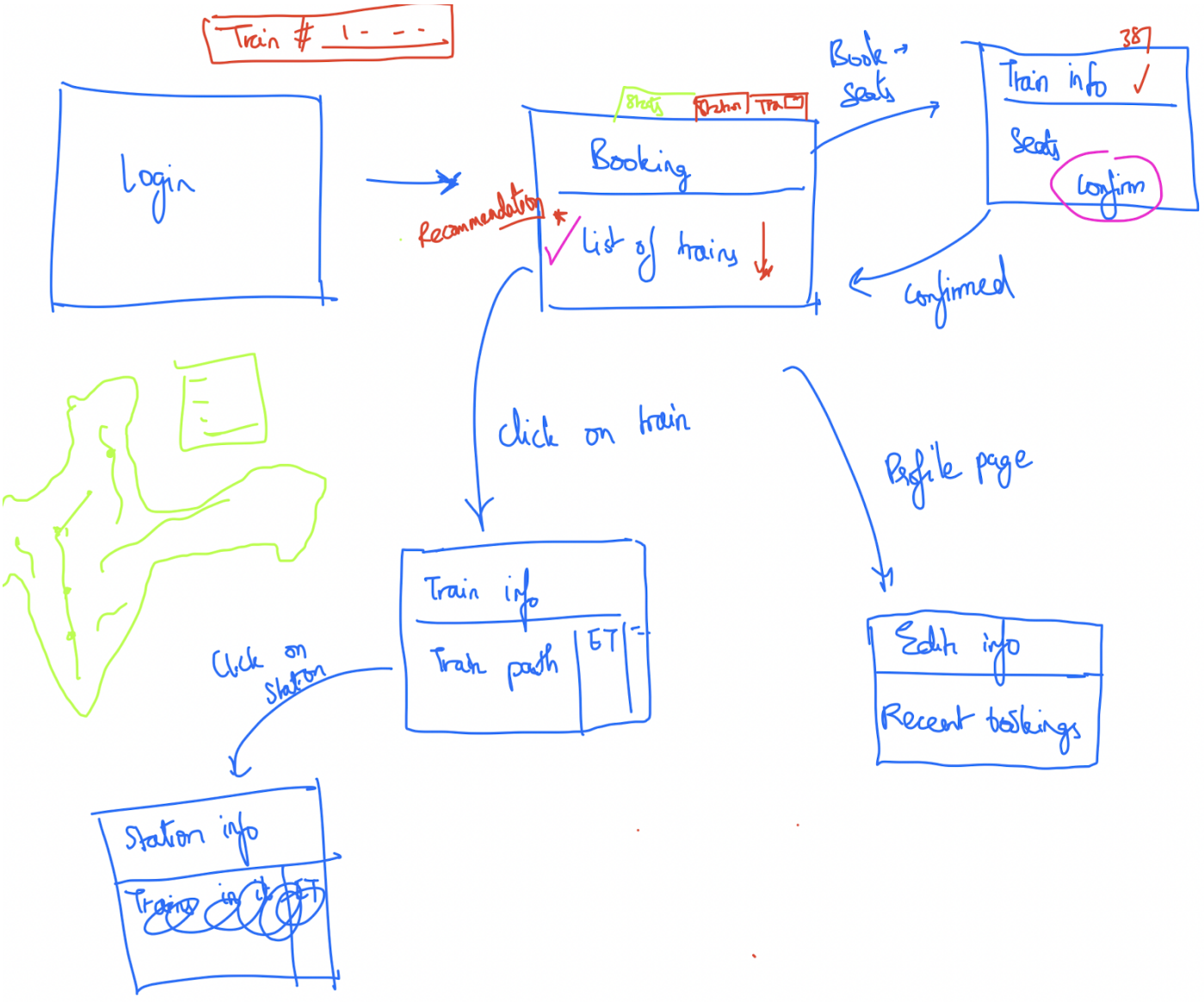
Book Ticket
Input: Passenger Details, Train No, Date, Time
Output: Update bookings Table

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

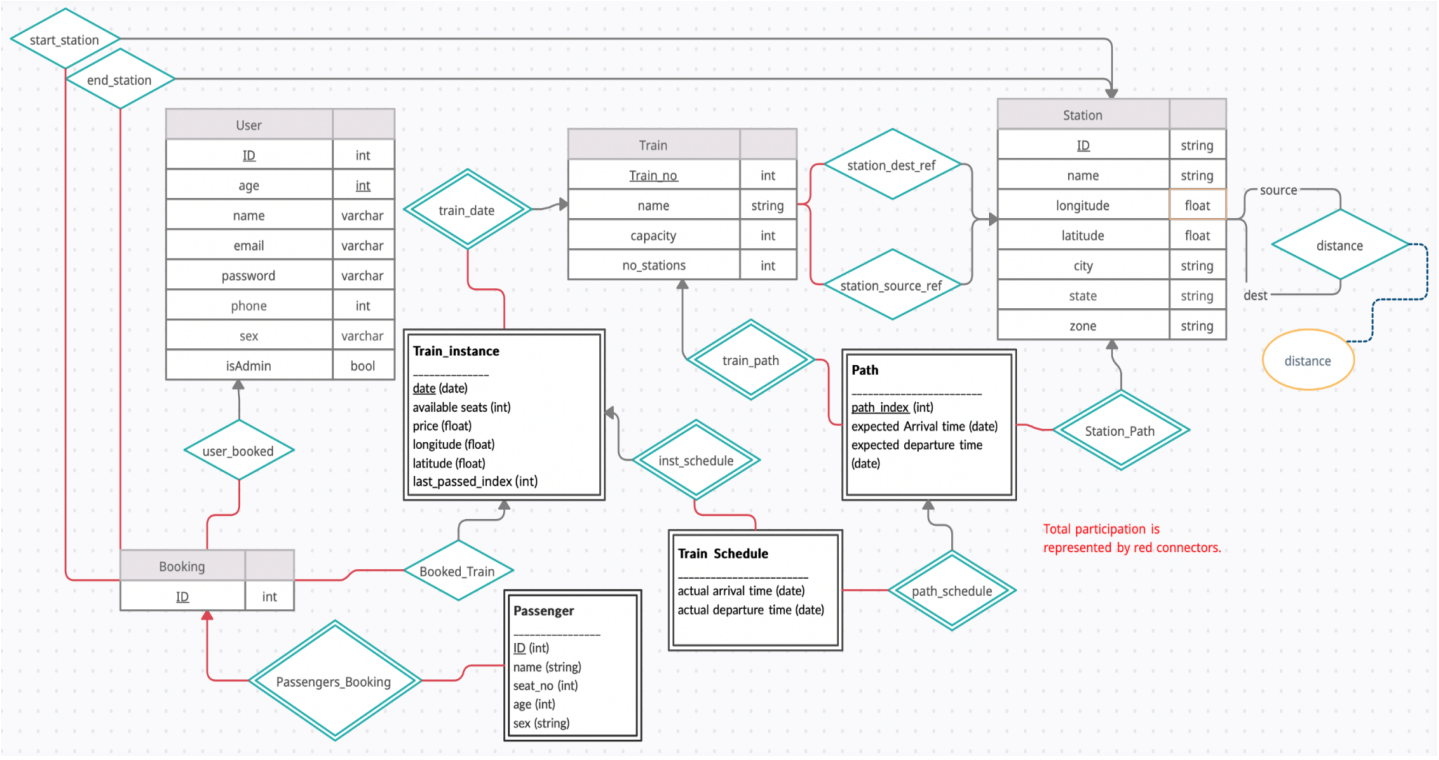
View Ticket
Input: Ticket Id
Output: Ticket Details



E.

DATASETS :

[Train database]



TEAM

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