**Railway Reservation System Backend**

**Introduction**

This document provides instructions for setting up and running the backend server for the Railway Reservation System. The server is built using Flask and interfaces with an SQLite database to manage train reservations and passenger data.

**Requirements**

* Python 3.x
* Flask
* SQLite3

**Installation**

* Clone or download the repository to your local machine.
* Navigate to the directory containing the **app.py** file
* **Setup**
* Ensure Python 3.x is installed on your system. Then, install the necessary Python packages:
* h
* Copy code
* pip install Flask sqlite3
* **Database Configuration**
* Run the provided script to initialize and configure the SQLite database. Make sure that the SQLite database file (**trains.db**) is located in the same directory as the Flask application.
* **Running the Server**
* To start the Flask server, run the following command in your terminal:
* h
* Copy code
* python app.py
* The server should start and be accessible via **http://localhost:5000** or a similar local address.
* **API Endpoints**
* The application provides several API endpoints for interacting with the railway reservation system. These may include functionalities like retrieving train information based on passenger details, checking ticket status, etc.
* [Provide specific details about each API endpoint here].

Queries:

Task1:

SELECT

t."Train Number",

t."Train Name",

t."Source Station",

t."Destination Station"

FROM

Passenger p

JOIN

booked b ON p.SSN = b.Passanger\_ssn

JOIN

Train t ON b.Train\_Number = t."Train Number"

WHERE

p.first\_name = ? AND

p.last\_name = ?

In this query we finding the booking of passenger by his first and last name.

Task2:

SELECT Passanger\_ssn, first\_name, last\_name, address, city, county, phone, SSN, bdate

FROM booked b

JOIN Passenger p ON b.Passanger\_ssn = p.SSN

JOIN Train\_status ts ON b.Train\_Number = b.Train\_Number

WHERE b.Staus = 'Booked' AND ts.TrainDate = ?

In this query we are finding the train on specific date.

Task3:

SELECT t."Train Number", t."Train Name", t."Source Station", t."Destination Station",

p.first\_name || ' ' || p.last\_name AS passengerName,

p.address, b.Ticket\_Type, b.Staus

FROM train t

JOIN booked b ON t."Train Number" = b.Train\_Number

JOIN passenger p ON b.Passanger\_ssn = p.SSN

WHERE ({current\_year} - CAST(strftime('%Y', '19' || substr(p.bdate, 7, 2) || '-' ||

substr(p.bdate, 1, 2) || '-' ||

substr(p.bdate, 4, 2)) AS INTEGER)) BETWEEN {min\_age} AND {max\_age}

In this query we are finding the passenger who’s age are in between the range 50-60

Task4:

SELECT

t."Train Name",

COUNT(b.Passanger\_ssn) AS PassengerCount

FROM

Train t

JOIN

booked b ON t."Train Number" = b.Train\_Number

GROUP BY

t."Train Name"

In this query we are finding the train no.and the desired platform on which train is going to arrive.

Task5:

SELECT p."first\_name", p."last\_name", p."SSN"

FROM Passenger p

JOIN booked b ON p."SSN" = b."Passanger\_ssn"

JOIN Train t ON b."Train\_Number" = t."Train Number"

WHERE t."Train Name" = ? AND b.Staus = 'Booked'

In this query we are finding the status of passenger either booked or waiting.

Task6:

UPDATE booked

SET Staus = 'Booked'

WHERE Passanger\_ssn = (

SELECT Passanger\_ssn

FROM booked

WHERE Staus = 'Waiting'

LIMIT 1

)

"""

new\_booked\_query = """

SELECT first\_name, last\_name, SSN

FROM Passenger

WHERE SSN = (

SELECT Passanger\_ssn

FROM booked

WHERE Staus = 'Booked'

LIMIT 1

In this query we are updating the passenger who are on waitlist and canceling there journey.