Train Ticket Booking API Documentation

This document provides information on how to use the Train Ticket Booking API for purchasing tickets, managing user information, and modifying bookings.
an overview of the train ticketing system implemented using Spring Boot and JPA with MySQL as the underlying database.

Project Functionality

This system allows users to:

- Purchase train tickets specifying origin, destination, and user details.
- Retrieve details of a purchased ticket using a unique identifier.
- View a list of users traveling in a specific section.
- Modify an existing ticket
- Cancel a ticket purchase (by deleting the ticket record).

Currently, the system supports a maximum of 50 reservations per section (A & B).

Code Structure

The project is organized into the following core packages:

- **model:** Contains entity classes representing data structures (User, Destination).
- **repository:** Provides JPA repository interfaces for interacting with the database (TicketRepository, UserRepository).
- **service:** Encapsulates business logic for train ticket management (TrainService).
- **controller:** Defines REST API endpoints for user interaction (TrainController).

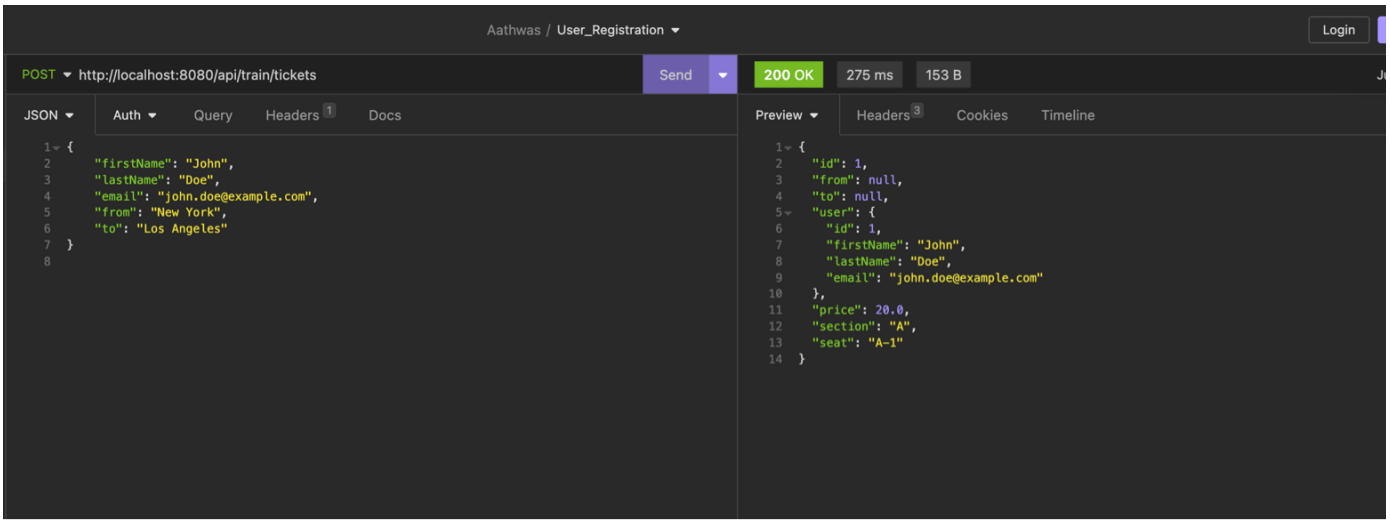
Technical Details

- **Java Version:** 17
- **Spring Boot**
- **Database:** MySQL with JPA configured.
- **Maven**
- **configure your database connection details in the application properties file**

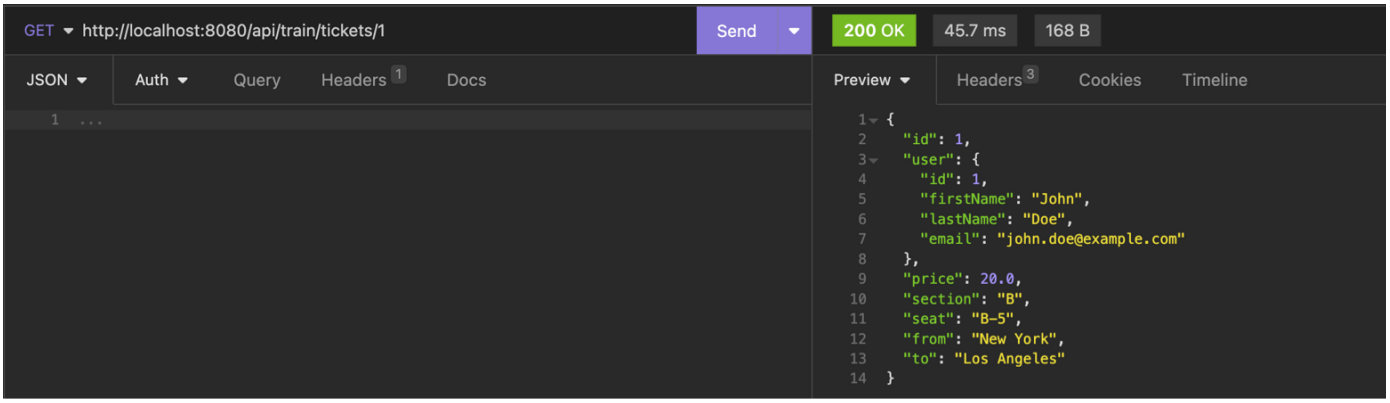
API Endpoints

The application exposes the following REST API endpoints:

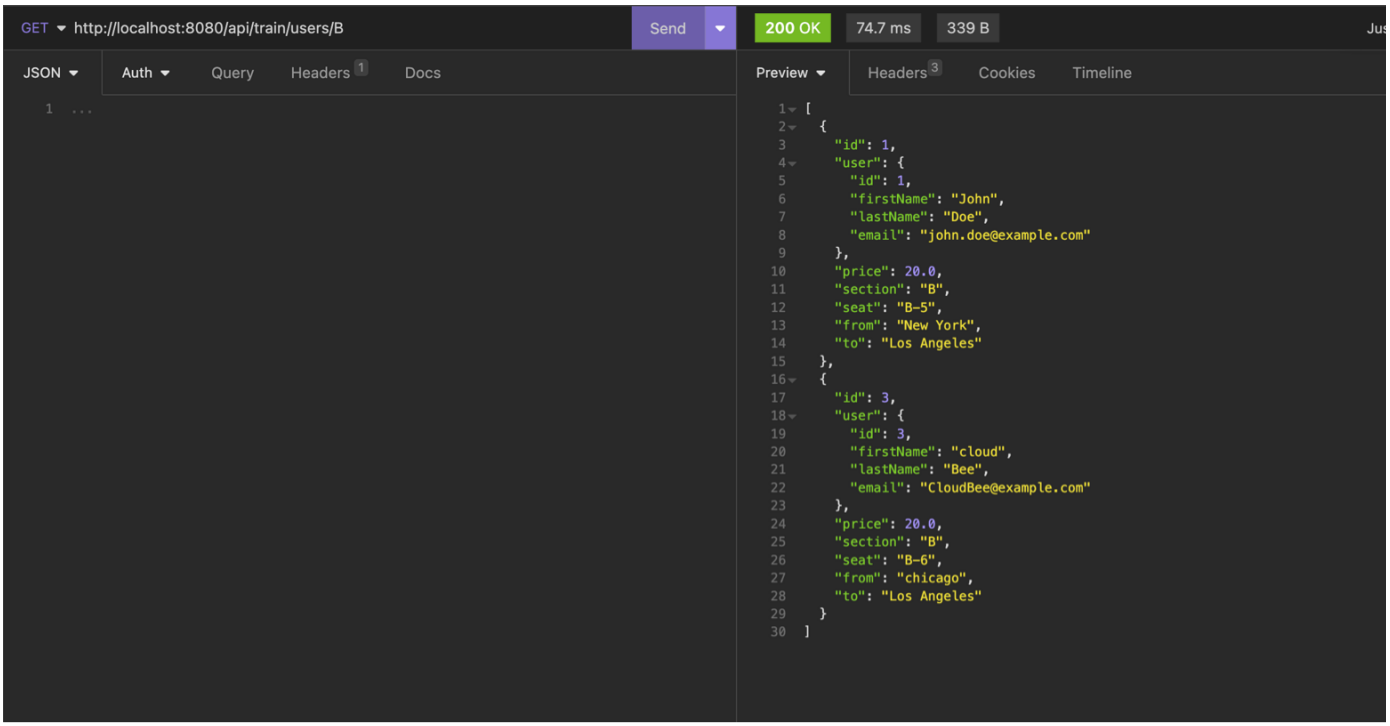
- **POST /api/train/tickets** - Purchase a new train ticket.
  - Request Body: User object (User) and origin & destination strings (from, to).
  - Response: Purchased ticket details (Destination).



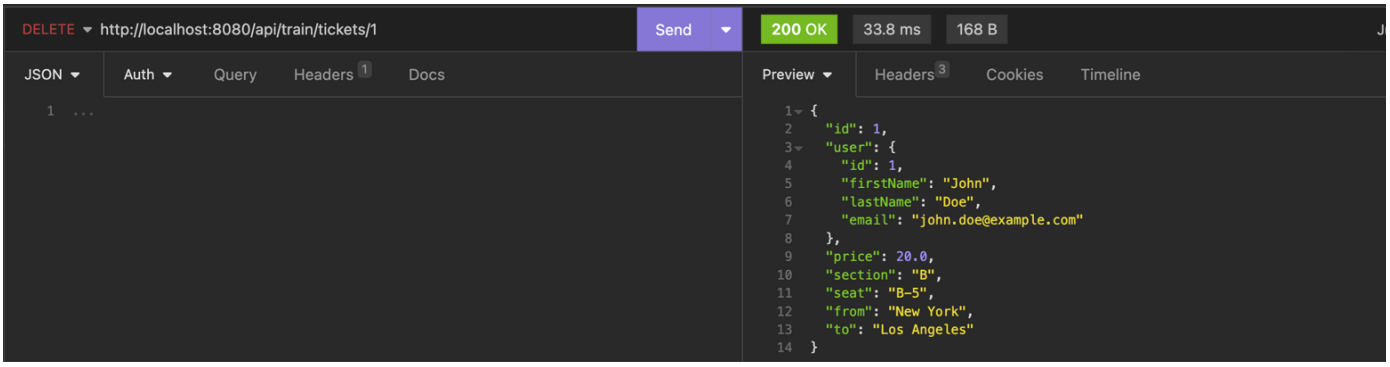
- **GET /api/train/tickets/{id}** - Retrieve details of a specific ticket.
  - Path Variable: Ticket ID (Long).
  - Response: Ticket details (Destination).



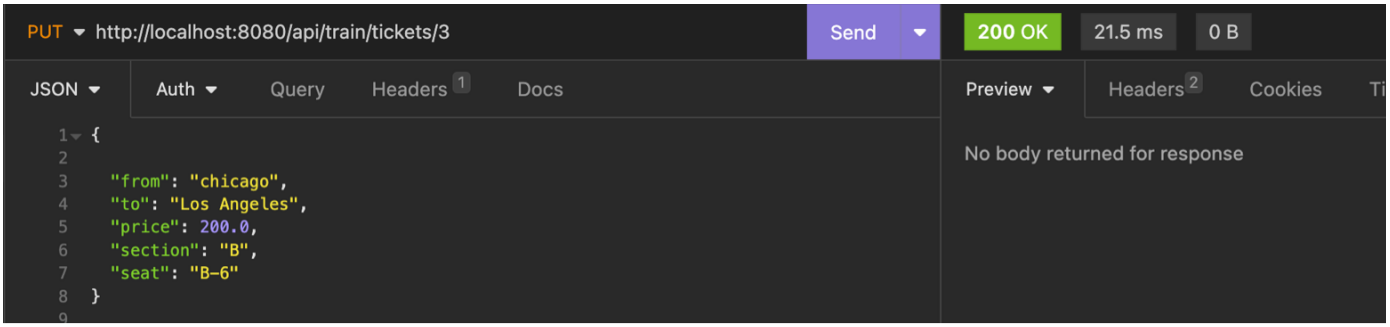
- **GET /api/train/users/{section}** - List users traveling in a particular section.
  - Path Variable: Section name (String).
  - Response: List of user tickets (List<Destination>).



- **DELETE /api/train/tickets/{id}** - Cancel a ticket purchase.
  - Path Variable: Ticket ID (Long).
  - Response: No response body (status code 200 indicates success).



- **PUT /api/train/tickets/{id}** - Modify an existing ticket (excluding section change).
  - Path Variable: Ticket ID (Long).
  - Request Body: Modified ticket details (Destination).
  - Response: Updated ticket details (Destination).



Tables in purchase ticket

- The two tables listed are destination and user

```
[mysql> create database purchaseTicket;
 Query OK, 1 row affected (0.02 sec)

[mysql> use purchaseTicket;
 Reading table information for completion of table and column names
 You can turn off this feature to get a quicker startup with -A

 Database changed
[mysql> show tables;
 +-------------------------+
 | Tables_in_purchaseticket |
 +-------------------------+
 | destination             |
 | user                    |
 +-------------------------+
 2 rows in set (0.01 sec)
```

```
[mysql> select * from destination;
 +-------+----+---------+----------------+---------------+------+---------+
 | price | id | user_id | destination_to | from_location | seat | section |
 +-------+----+---------+----------------+---------------+------+---------+
 |    20 |  1 |       1 | Los Angeles    | New York      | B-5  | B       |
 |    20 |  2 |       2 | chicago        | New York      | A-1  | A       |
 |    20 |  3 |       3 | Los Angeles    | chicago       | B-6  | B       |
 +-------+----+---------+----------------+---------------+------+---------+
 3 rows in set (0.00 sec)

[mysql> select * from user;
 +----+---------------------+------------+-----------+
 | id | email               | first_name | last_name |
 +----+---------------------+------------+-----------+
 |  1 | john.doe@example.com | John       | Doe       |
 |  2 | sahil@example.com    | sahil      | Amin      |
 |  3 | CloudBee@example.com | cloud      | Bee       |
 +----+---------------------+------------+-----------+
 3 rows in set (0.00 sec)
```

Additional

- Error handling is implemented using custom exceptions and appropriate status codes in responses.
- This is a basic implementation and can be further enhanced with features like seat selection, different ticket classes, and integration with payment gateways.