Topic :- Train ticket system

Introduction:

The Train Ticket Reservation System is a comprehensive web-based platform engineered to modernize and centralize the lifecycle of railway travel management. While the core of the application focuses on the efficient handling of passenger records—including adding, viewing, and updating reservations—it extends beyond a simple data repository to serve as a high-performance transactional engine. By integrating **Spring Security** for role-based authentication and **Spring JDBC** for optimized database communication, the system ensures that sensitive passenger data is handled securely while maintaining real-time accuracy of seat inventory.

The application's architecture is designed to bridge the gap between complex backend business logic and an intuitive user experience. Utilizing a responsive frontend built with **Thymeleaf** and **Bootstrap**, the system provides a seamless interface for both passengers searching for available journeys and administrators managing fleet schedules. This transition from manual paper-based tracking to an automated **PostgreSQL**-driven environment significantly reduces human error, prevents overbooking through automated constraint logic, and provides a scalable foundation for high-volume ticketing operations.

Application Flow :

User Browser

↓

PassengerController (home method)

↓

PassengerRepository.findAll() → Database

↓

Model { passengers: [...] }

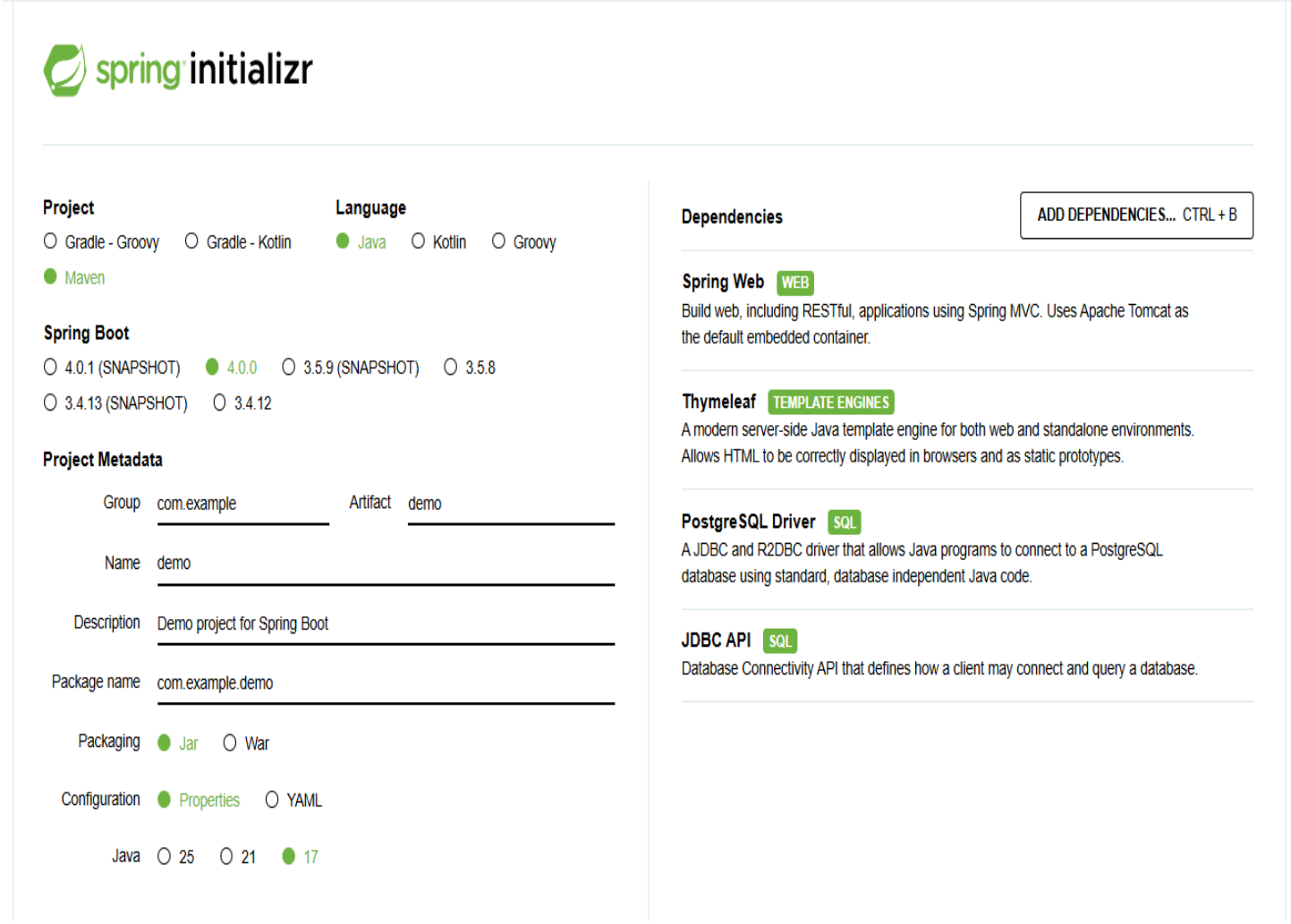
↓

Thymeleaf renders index.html

↓

HTML page displayed with passenger table

Step1:-
Go on spring intializer to create the spring project and add dependency

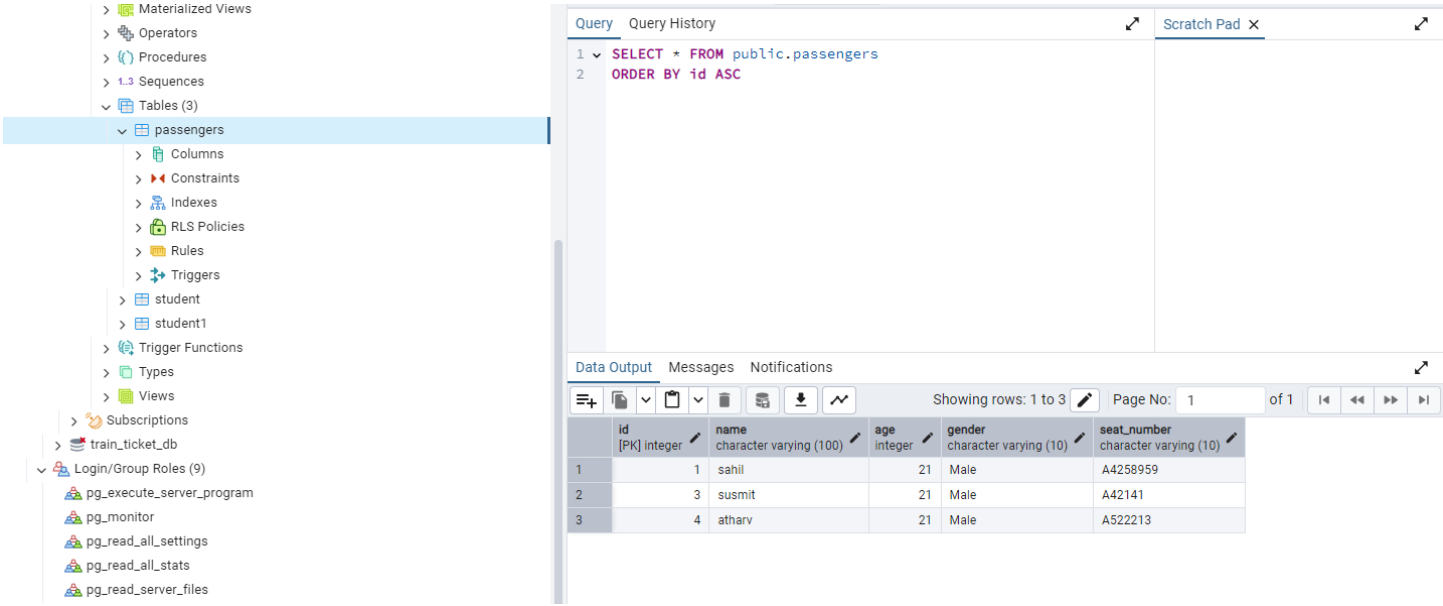


Step 2:-

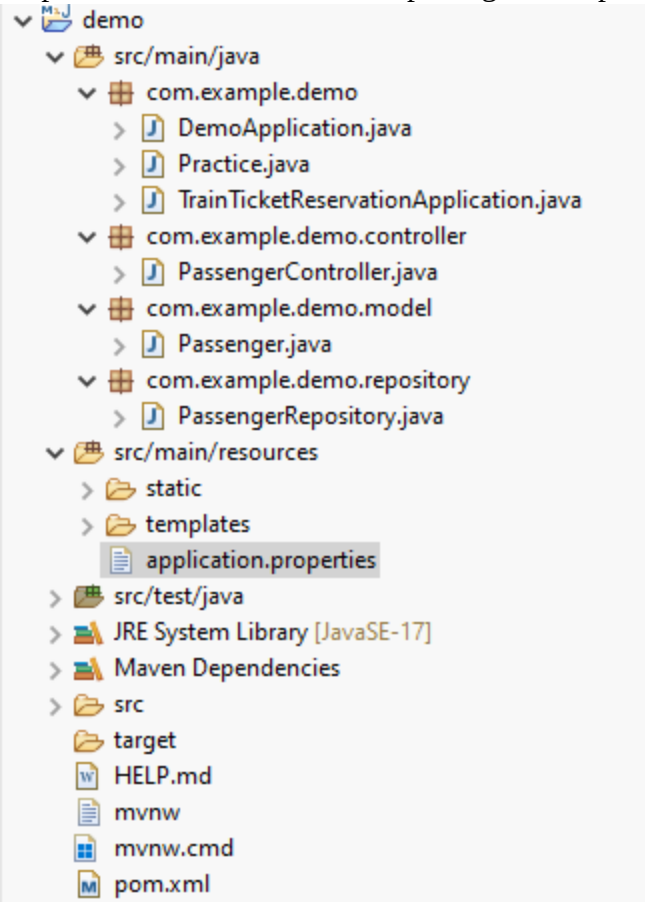
Create database table

```
CREATE DATABASE train_ticket_db;

CREATE TABLE passengers (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    age INT NOT NULL,
    gender VARCHAR(10),
    seat_number VARCHAR(10)
);
```



Step 3 - create controller,model package in the prebuild package



- **PassengerRepository.java -** Data Access Layer
- **RowMapper -** Database Row to Object Conversion
- **@Repository** -Data Access Object (DAO)

Application.properties

```
server.port=8086

spring.datasource.url=jdbc:postgresql://localhost:5432/postgres
spring.datasource.username=postgres
spring.datasource.password=password
spring.datasource.driver-class-name=org.postgresql.Driver

```

Application

workflow:- Home

Page
User lands on the main dashboard displaying an overview of recent entries and navigation options

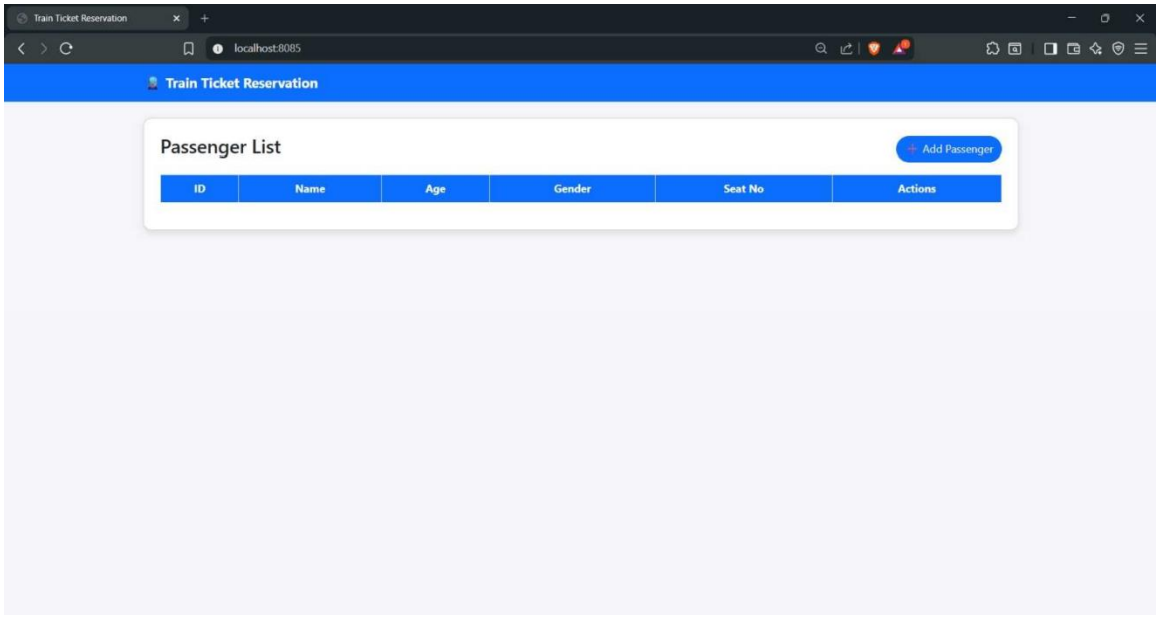
Add Entry
User navigates to the entry form to create a new journal entry with title, content, and mood selection

Save to Database
System validates and persists the entry to MySQL database with automatic timestamp generation

View Entries
User browses through all saved journal entries in chronological order with mood indicators

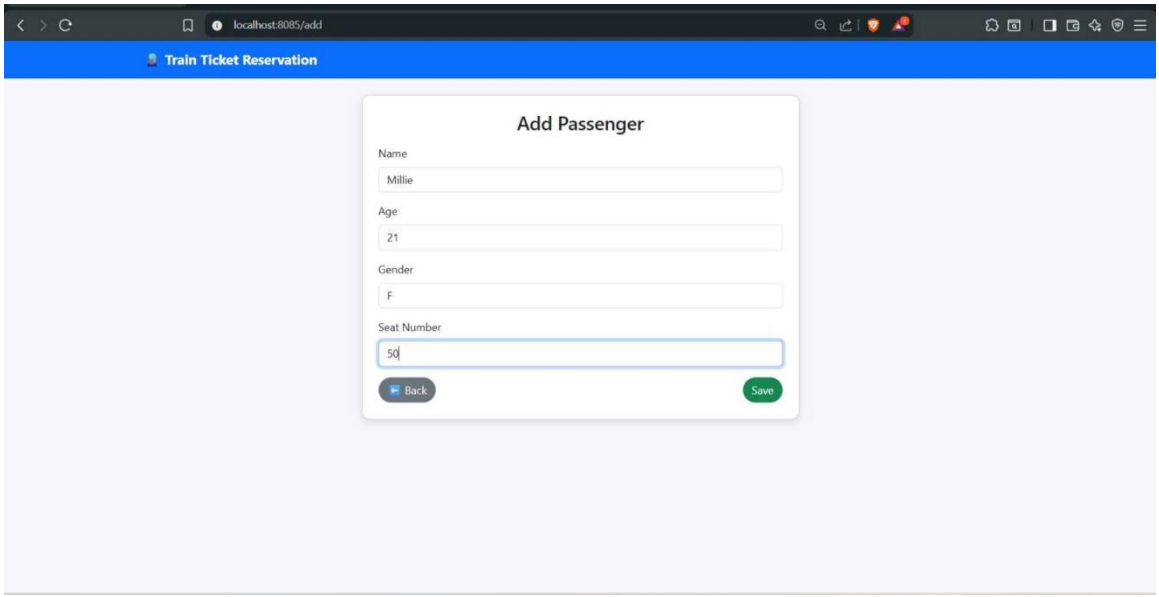
Delete Entry
User can remove unwanted entries with confirmation, triggering database deletion operation

Output-



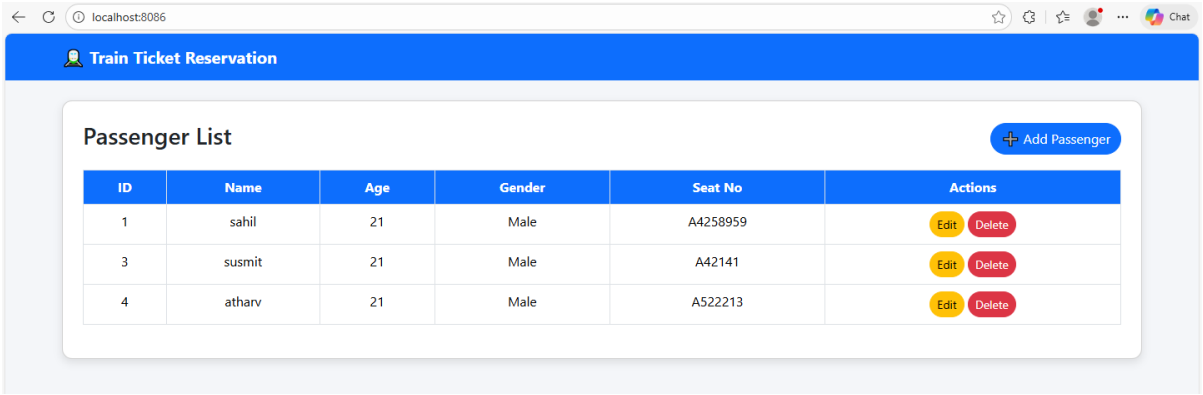
Pic 1.0

This output screen shows the Passenger List page of the Train Ticket Reservation System. It displays passenger details such as ID, Name, Age, Gender, and Seat Number in a tabular format. The “Add Passenger” button allows users to add new passenger details for ticket reservation.



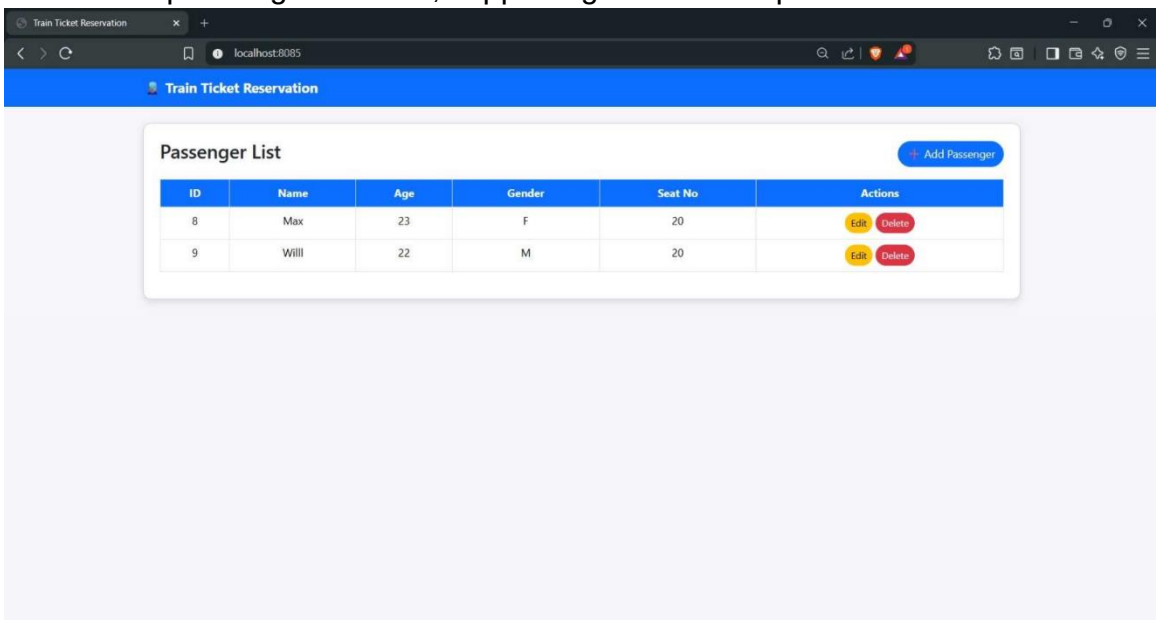
Pic 1.1

This screen shows the **Add Passenger** form of the Train Ticket Reservation System. It allows users to enter passenger details such as **Name, Age, Gender, and Seat Number**. After filling the form, the **Save** button stores the passenger information in the system, while **Back** returns to the passenger list page.



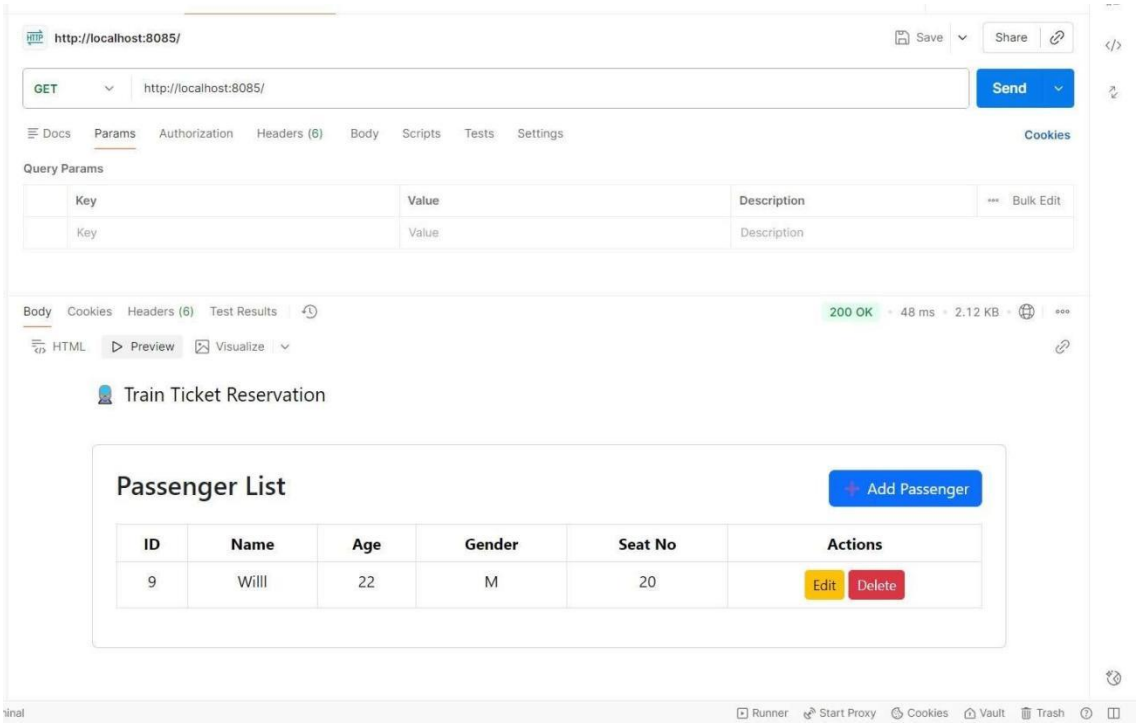
Pic 1.2

This screen displays the **updated Passenger List** after adding a passenger. It shows stored details such as **ID, Name, Age, Gender, and Seat Number** retrieved from the database. The **Edit** and **Delete** buttons allow users to update or remove passenger records, supporting full CRUD operations.



Pic 1.3

This screen shows the Passenger List page with multiple records in the Train Ticket Reservation System. It displays details of all registered passengers, including Name, Age, Gender, and SeatNumber ,fetched from the database. The Add Passenger, Edit, and Delete options allow users to manage passenger information efficiently.



Pic 1.4

This output shows the **GET API response** of the Train Ticket Reservation System tested using an API tool. The request returns a **200 OK** status and successfully loads the **Passenger List page**, displaying passenger data retrieved from the backend. It confirms proper integration between the **Spring Boot backend** and the frontend for fetching and displaying records.