System Requirements Specification

Index

For

Train Information Management

Version 1.0

Table of Contents

TRAIN INFORMATION MANAGEMENT
System Requirements Specification

You need to consume APIs exposed by Backend application in Angular to make application work as FULLSTACK

BACKEND-SPRING BOOT RESTFUL APPLICATION

1 PROJECT ABSTRACT

The **Train Information Management System** is a FullStack Application with a backend implemented using Spring Boot with a MySQL database and a frontend developed using Angular. It serves as a comprehensive platform for managing and organizing train-related information and services.

Following is the requirement specifications:

	Train Information Management System
Modules	
1	Train
Event Module Functionalities	
1	Create a Train
2	Update the existing Train details
3	Get the Train by Id
4	Get all Trains
5	Delete a Train
6	Get the Train by Train number
7	Search for Train by Train name
8	Search for Train by Seats availability

2 ASSUMPTIONS, DEPENDENCIES, RISKS / CONSTRAINTS

2.1 TRAIN CONSTRAINTS

- When fetching a Train by ID, if the train ID does not exist, the operation should throw a custom exception.
- When updating a Train, if the train ID does not exist, the operation should throw a custom exception.
- When removing a Train, if the train ID does not exist, the operation should throw a custom exception.
- When fetching a Train by number, if the train number does not exist, the operation should throw a custom exception.
- When searching for a Train by train name, if the train name does not exist, the operation should throw a custom exception.

Common Constraints

- For all rest endpoints receiving @RequestBody, validation check must be done and must throw custom exception if data is invalid
- All the business validations must be implemented in dto classes only.
- All the database operations must be implemented on entity object only
- Do not change, add, remove any existing methods in service layer
- In Repository interfaces, custom methods can be added as per requirements.
- All RestEndpoint methods and Exception Handlers must return data wrapped in **ResponseEntity**

3 BUSINESS VALIDATIONS

- Train number is not null.
- Train name is not null, min 3 and max 100 characters.
- Departure station is not null.
- Arrival station is not null.
- Duration is not null.
- Distance is not null.
- Fare is optional.
- Seats available is optional.

4 REST ENDPOINTS

Rest End-points to be exposed in the controller along with method details for the same to be created

4.1 TRAINCONTROLLER

URL Exposed		Purpose
1. /trains		Fetches all the trains
Http Method	GET	
Parameter	-	
Return	List<Trains>	
2. /trains		Add a new train details
Http Method	POST	
Parameter 1	Train	
Return	Train	
3. /trains/{id}		Delete train with given train id
Http Method	DELETE	
Parameter 1	Long (id)	
Return	-	
4. /trains/{id}		Fetches the train with the given id
Http Method	GET	
Parameter 1	Long (id)	
Return	Train	
5. /trains/{id}		Updates existing Train info
Http Method	PUT	
Parameter 1	Long (id)	
Parameter 2	Train	
Return	Train	
6. /trains/number/{number}		Get the train with the given train number
Http Method	GET	
Parameter 1	Integer (number)	
Return	List<Trains>	
7. /trains/name/{name}		Search the train with the given name
Http Method	GET	
Parameter 1	String (name)	
Return	List<Trains>	

8. /trains/seats-available/{seatsAvailable}		Search the train by the seats availability
Http Method	GET	
Parameter 1	Integer (seatsAvailable)	
Return	List<Trains>	

5 Template Code Structure

5.1 Package: com.traininfo

Resources

TrainInfoApplication (Class)	This is the Spring Boot starter class of the application.	Already Implemented

5.2 PACKAGE: COM.TRAININFO.REPOSITORY

Resources

Class/Interface	Description	Status
TrainDAO (interface)	• Repository interface exposing CRUD functionality for Train Entity. • You can go ahead and add any custom methods as per requirements.	Partially implemented.

5.3 PACKAGE: COM.TRAININFO.SERVICE

Resources

Class/Interface	Description	Status
TrainService (interface)	• Interface to expose method signatures for train related functionality. • Do not modify, add or delete any method.	Already implemented.

TrainServiceImpl (class)	● Implements TrainService. ● Contains template method implementation. ● Need to provide implementation for train related functionalities. ● Do not modify, add or delete any method signature	To be implemented.

5.4 PACKAGE: COM.TRAININFO.CONTROLLER

Resources

Class/Interface	**Description**	**Status**
TrainController (Class)	● Controller class to expose all rest-endpoints for train related activities. ● May also contain local exception handler methods	To be implemented

5.5 PACKAGE: COM.TRAININFO.DTO

Resources

Class/Interface	**Description**	**Status**
TrainDTO (Class)	Use appropriate annotations from the Java Bean Validation API for validating attributes of this class.	Partially implemented.

5.6 PACKAGE: COM.TRAININFO.ENTITY

Resources

Class/Interface	**Description**	**Status**
Train (Class)	● This class is partially implemented. ● Annotate this class with proper annotation to declare it as an entity class with **trainId** as primary key. ● Map this class with a **train table**. ● Generate the **trainId** using the IDENTITY strategy	Partially implemented.

5.7 PACKAGE: COM.TRAININFO.EXCEPTION

Resources

Class/Interface	**Description**	**Status**
ResourceNotFoundException (Class)	● Custom Exception to be thrown when trying to fetch or delete the train info which does not exist. ● Need to create Exception Handler for same wherever needed (local or global)	Already implemented.
GlobalExceptionHandler (Class)	● RestControllerAdvice Class for defining global exception handlers. ● Contains Exception Handler for **InvalidDataException** class. ● Use this as a reference for creating exception handler for other custom exception classes.	Already implemented.

6 CONSIDERATIONS

A. There is no roles in this application
B. You can perform the following possible action

Train

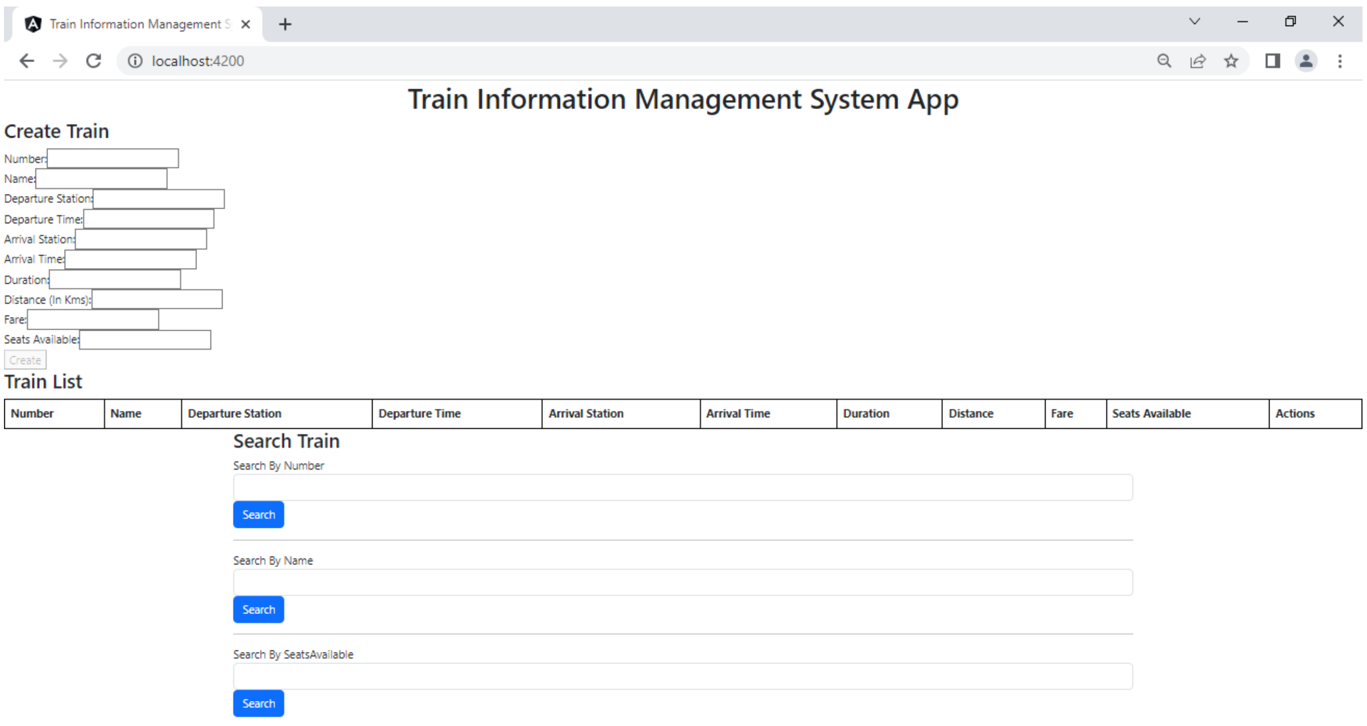
FRONTEND-ANGULAR SPA

1 PROBLEM STATEMENT

The Train Information Management System is SPA (Single Page Application), it allows to manage the trains with functionalities to create a new train, update an existing train, get detailed information, and search any particular train.

2. PROPOSED Train Information Management System WIREFRAME

2. 1 HOME PAGE



Train Information Management System App

Create Train

Number:
Name:
Departure Station:
Departure Time:
Arrival Station:
Arrival Time:
Duration:
Distance (In Kms):
Fare:
Seats Available:
Create

Train List

Number	Name	Departure Station	Departure Time	Arrival Station	Arrival Time	Duration	Distance	Fare	Seats Available	Actions

Search Train

Search By Number

Search

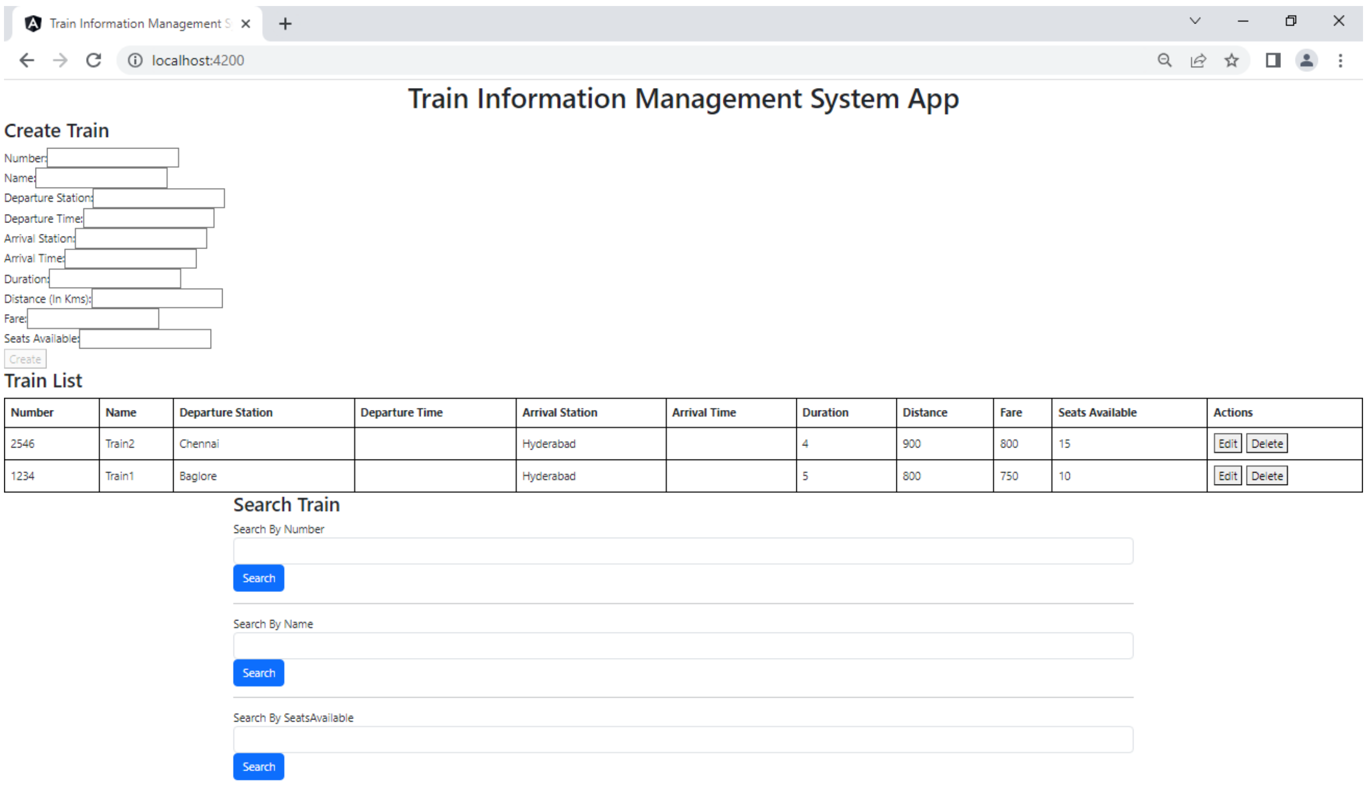
Search By Name

Search

Search By SeatsAvailable

Search

AFTER CREATING or ADDING THE TRAINS:


Train Information Management S

localhost:4200

Train Information Management System App

Create Train

Number:	
Name:	
Departure Station:	
Departure Time:	
Arrival Station:	
Arrival Time:	
Duration:	
Distance (In Kms):	
Fare:	
Seats Available:	

Create

Train List

Number	Name	Departure Station	Departure Time	Arrival Station	Arrival Time	Duration	Distance	Fare	Seats Available	Actions
2546	Train2	Chennai		Hyderabad		4	900	800	15	Edit Delete
1234	Train1	Baglore		Hyderabad		5	800	750	10	Edit Delete

Search Train

Search By Number

Search

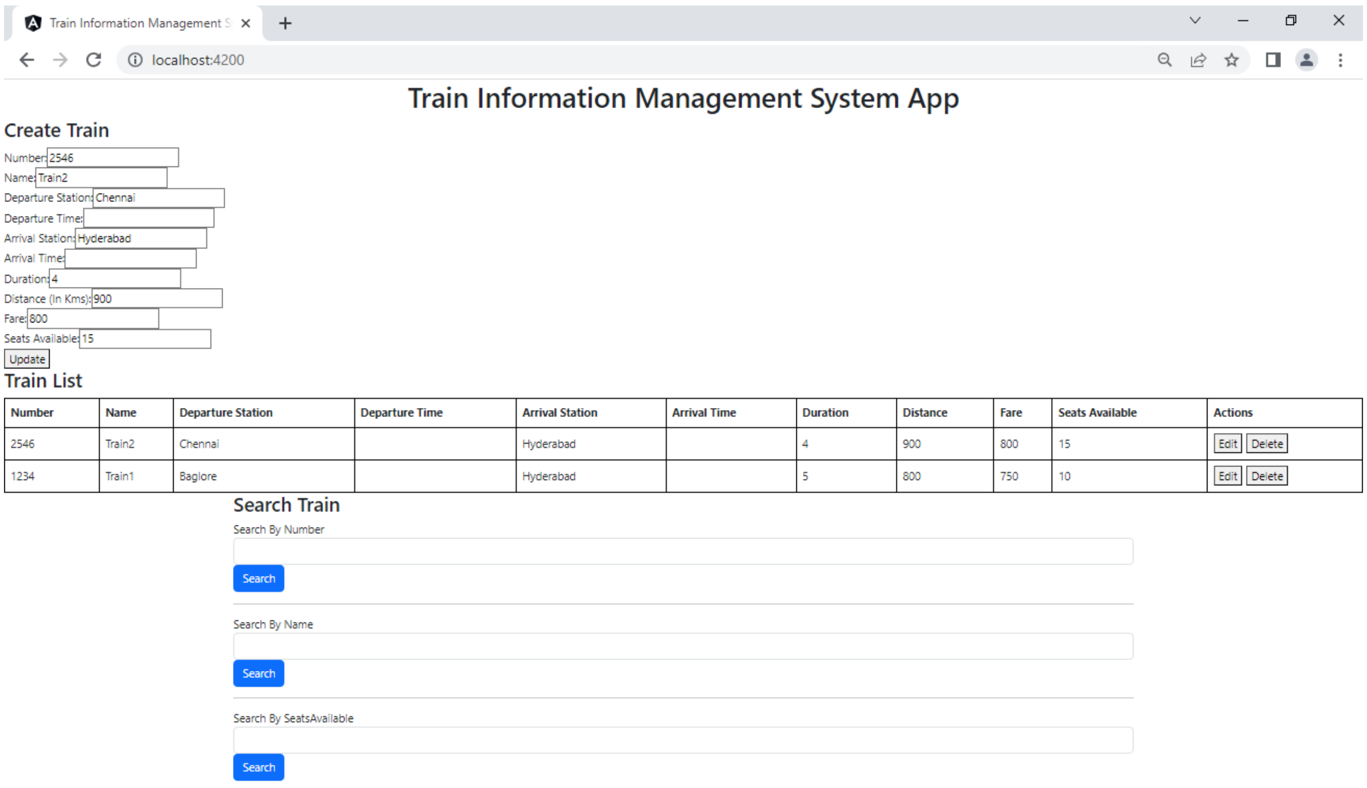
Search By Name

Search

Search By SeatsAvailable

Search

ON CLICK UPDATE BUTTON:


Train Information Management S

localhost:4200

Train Information Management System App

Create Train

Number:	2546
Name:	Train2
Departure Station:	Chennai
Departure Time:	
Arrival Station:	Hyderabad
Arrival Time:	
Duration:	4
Distance (In Kms):	900
Fare:	800
Seats Available:	15

Update

Train List

Number	Name	Departure Station	Departure Time	Arrival Station	Arrival Time	Duration	Distance	Fare	Seats Available	Actions
2546	Train2	Chennai		Hyderabad		4	900	800	15	Edit Delete
1234	Train1	Baglore		Hyderabad		5	800	750	10	Edit Delete

Search Train

Search By Number

Search

Search By Name

Search

Search By SeatsAvailable

Search

3. BUSINESS-REQUIREMENT:

As an application developer, develop the Train Information Management System App (Single Page App) with below guidelines:

User Story #	User Story Name	User Story
US_01	Home Page	As a user I should be able to visit the home page as the default page. Where I can see a form to create or update the train, list of all trains with options to edit and delete any train and at last there should be a form to search any train on its number, name, and seat available criteria.
US_01	Home Page	As a user I should be able to see the homepage and perform all operations: Acceptance criteria: 1. As a user I should be able to furnish the following details at the time of add or creating the train. 1.1 Number 1.2 Name 1.3 Departure Station 1.4 Departure Time 1.5 Arrival Station 1.6 Arrival Time 1.7 Duration 1.8 Distance 1.9 Fare 1.10 Seats Available 2. Create button should be disabled until all fields are validated. 3. Update button should be displayed when you click on the Edit button. 4. Name field min length is 3 and max length 100. 5. The fields, number, name, departureStation, departureTime, arrivalStation, arrivalTime, duration, distance are mandatory. If any field is missing or if any constraint is not satisfied then must show a message. 6. Form control names should be case sensitive and they should be like as follows:

		id number name departureStation departureTime arrivalStation arrivalTime duration distance fare seatsAvailable 7. Form control names for 3 search operations should be case sensitive and they should be like as follows: number name seats 8. While searching by seats available, it should return all records whose seats are greater than or equal to given number.

1 EXECUTION STEPS TO FOLLOW FOR BACKEND

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers need to go to the Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal.**
3. **cd into your backend project folder**
4. **To build your project use command:**

   ```
   mvn clean package -Dmaven.test.skip
   ```
5. **To launch your application, move into the target folder (cd target). Run the following command to run the application:**

   ```
   java -jar <your application jar file name>
   ```
6. **This editor Auto Saves the code.**
7. **If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use CTRL+Shift+B-command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.**
8. **These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.**
9. **To test any Restful application, the last option on the left panel of IDE, you can find ThunderClient, which is the lightweight equivalent of POSTMAN.**
10. **To test any UI based application the second last option on the left panel of IDE, you can find Browser Preview, where you can launch the application.**
11. **Default credentials for MySQL:**
    a. **Username: root**
    b. **Password: pass@word1**

11. **To login to mysql instance: Open new terminal and use following command:**
    a. **sudo systemctl enable mysql**
    b. **sudo systemctl start mysql**
    c. **mysql -u root -p**
    The last command will ask for password which is 'pass@word1'

12. **Mandatory: Before final submission run the following command:**

    ```
    mvn test
    ```

13. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**

1 Execution Steps to Follow for Frontend

1. **All actions like build, compile, running application, running test cases will be through Command Terminal.**
2. **To open the command terminal the test takers, need to go to**

   Application menu (Three horizontal lines at left top) -> Terminal ->New Terminal.
3. **This is a web-based application, to run the application on a browser, use the internal browser in the environment.**
4. **You can follow series of command to setup Angular environment once you are in your project-name folder:**
   a. **npm install -> Will install all dependencies -> takes 10 to 15 min**
   b. **npm run start -> To compile and deploy the project in browser. You can press <Ctrl> key while clicking on localhost:4200 to open project in browser -> takes 2 to 3 min**
   c. **npm run test -> to run all test cases. It is mandatory to run this command before submission of workspace -> takes 5 to 6 min**
5. **You need to use CTRL+Shift+B - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.**