Trains Info Case Study

Develop 2 Projects for this case study. Please Read the below instructions carefully

Project-1: Train Information Microservices

Overview

The goal of this project is to design and develop a set of microservices that provide information about trains, specifically their Train Number, Name, and Stations from the source to the destination. This information will be accessible through HTTP calls.

Microservices

1. **Station Details Microservice**: This microservice will provide station details based on a Train ID.
2. **Train Information Microservice**: This microservice will call the Station Details microservice to provide complete train information, including retrieved station details.

Tools and Technologies

The project will be implemented using the following tools and technologies:

- **Maven**: For project management and build automation.
- **Spring Boot**: For developing the microservices.
- **Lombok**: To reduce boilerplate code in the application.
- **MySQL**: To store and retrieve train and station data.
- **Eureka Server**: To enable service discovery and registration.

Functional Requirements

The system must provide the following functionality:

Station Details Microservice

1. Expose an API to retrieve station details based on a Train ID.
2. Retrieve station details from the database using the Train ID.
3. Return the station details in a structured format (e.g., JSON).

Train Information Microservice

1. Expose an API to retrieve complete train information, including station details, based on a Train ID.
2. Call the Station Details microservice to retrieve station details.
3. Combine station details with train information.
4. Return the complete train information in a structured format (e.g., JSON).

Database Design

The system will use MySQL to store train and station data. The database schema will include tables for storing train information and station details.

Architecture

The microservices will be developed using the Spring Boot framework and will follow a microservices architecture. Service discovery and registration will be implemented using Eureka Server.

Non-functional Requirements

1. The system should be highly available and scalable.
2. Proper error handling and validation should be implemented in the APIs.
3. Security measures should be in place to protect sensitive data.
4. API documentation should be available for developers.

Deliverables

The project should deliver the following:

1. Source code for both microservices.
2. Database schema and scripts to create the necessary tables.
3. API documentation for both microservices.
4. Deployment configurations for Maven, Spring Boot, and Eureka Server.
5. Documentation on how to set up and run the microservices.

Project-2: Train Information Consumer Project

Overview

The goal of this project is to create a Spring Boot application that consumes the Train Information Microservices, which provide information about trains, including their Train Number, Name, and Stations from the source to the destination. This application will act as a consumer, retrieving and displaying train information to end-users in a user-friendly manner using the Model-View-Controller (MVC) architectural pattern.

Functional Requirements

The system must provide the following functionality:

1. **Retrieve Train Information**: The application should make HTTP calls to the Train Information Microservices to fetch train information based on a Train ID.
2. **Display Train Information**: Display the retrieved train information, including station details, to the end-users through a web-based interface.
3. **User Interface**: Develop a user-friendly web interface that allows users to input a Train ID, fetch train information, and display it in a structured manner.

4. **Error Handling**: Implement proper error handling for cases where the microservices are unavailable or encounter errors.

Tools and Technologies

The project will be implemented using the following tools and technologies:

- **Spring Boot**: For building the application.
- **Spring MVC**: For implementing the Model-View-Controller architecture.
- **RestTemplate**: For making HTTP calls to the Train Information Microservices.
- **Frontend technologies (HTML, CSS, JavaScript)**: For building the user interface.

Architecture

The project will follow the MVC architectural pattern:

- **Model**: Represents the data and the business logic to retrieve and process train information from the microservices.
- **View**: Represents the user interface, which will display train information.
- **Controller**: Acts as an intermediary between the Model and View. It handles user input, interacts with the RestTemplate to fetch data, and updates the View with the retrieved information.

Non-functional Requirements

1. The application should be responsive, user-friendly, and easy to navigate.
2. Proper error messages and validation should be implemented in the user interface.
3. The application should gracefully handle cases where the Train Information Microservices are unavailable.

Deliverables

The project should deliver the following:

1. Source code for the Spring Boot application.
2. Web pages for user interface using a templating engine.
3. Documentation on how to set up and run the application.
4. Proper error handling and validation in the user interface.