**DevOps Course**

**Demo Day Project**

Department and Year : IT and 3rd Year

Batch Name : Elite

Batch Members : Nivethethaa A S

Manju Sri S

Janani M

Asika M

GitHub Link :

[https://github.com/MANJUSRI08/railway-reservation-system\_devops\_Inlustro](https://github.com/MANJUSRI08/railway-reservation-system_devops_Inlustro" \o "https://github.com/MANJUSRI08/railway-reservation-system_devops_Inlustro)

**Railway Reservation System**

**Aim:**

To demonstrate containerization using Docker, exposing the project metrics through Prometheus, extracting the results from Prometheus and graphically displaying them in Grafana for Railway Reservation System.

**Procedure:**

**Introduction**

The Railway Reservation System is an online platform designed to facilitate the booking of train tickets for passengers. This documentation provides a comprehensive guide to the system's requirements, design, implementation, and usage.

**Requirements Documentation**

**Functional Requirements**

* **User Authentication**: Users should be able to register, log in, and log out of the system.
* **Booking Tickets**: Users should be able to search for trains, view available seats, and book tickets for desired routes and dates.
* **Viewing Bookings**: Users should be able to view their booked tickets, including details such as train name, departure time, seat number, etc.
* **Cancelling Bookings**: Users should be able to cancel booked tickets if needed.
* **Admin Panel**: Administrators should have access to an admin panel to manage trains, stations, bookings, etc.

**Non-functional Requirements**

* **Performance**: The system should be responsive and capable of handling multiple concurrent users without significant delays.
* **Security**: User data should be stored securely and protected against unauthorized access. Passwords should be hashed before storage.
* **Usability**: The user interface should be intuitive and easy to navigate, with clear instructions provided for all functionalities.

**Design Documentation**

**Architecture Overview**

The system follows a client-server architecture, with the frontend implemented using HTML, CSS, and JavaScript, and the backend implemented using Node.js and Express.js. Data is stored in a MySQL database.

**Database Design**

* **Tables**: The database consists of tables for trains, stations, bookings, users, etc.
* **Entity-Relationship Diagram (ERD)**: A diagram illustrating the relationships between different entities and their attributes.

**System Components**

* **Frontend**: HTML templates for user interface components, CSS for styling, and JavaScript for interactivity.
* **Backend**: Node.js and Express.js for server-side logic, handling API requests, and interacting with the database.

**Implementation Documentation**

**Code Structure**

* **Directory Structure**: Overview of project directory structure, including frontend and backend directories.
* **Key Files**: Description of important files and their functions.

**Backend Implementation**

* **Modules**: Description of backend modules, routes, controllers, and middleware functions.
* **Database Interaction**: Code snippets illustrating how the backend interacts with the MySQL database using Sequelize ORM.

**Frontend Implementation**

* **HTML Templates**: Description of HTML templates for different pages (home, login, register, booking, etc.).
* **CSS Styling**: Overview of CSS stylesheets for layout, colors, fonts, etc.
* **JavaScript**: Explanation of client-side JavaScript functions for form validation, AJAX requests, etc.
* **Testing Documentation**

**Test Plan**

* **Testing Approach**: Description of testing approach, including unit testing, integration testing, and system testing.
* **Tools**: List of testing tools and frameworks used (e.g., Mocha, Chai, Postman).

**Test Cases**

* **Test Scenarios**: Detailed test cases covering functional and non-functional requirements.
* **Expected Results**: Expected outcomes for each test case.

**Test Results**

* **Summary**: Summary of test execution results, including passed, failed, and pending tests.
* **Bug Reports**: Description of any bugs encountered during testing, along with steps to reproduce and resolve them.

**Deployment Documentation**

**Deployment Plan**

* **Deployment Steps**: Instructions for deploying the system to a production or staging environment.
* **Dependencies**: List of dependencies and configuration settings required for deployment.

**Server Configuration**

* **Server Environment**: Description of server environment setup, including web server (e.g., Apache), database server (e.g., MySQL), etc.
* **Configuration Files**: Sample configuration files and settings.

**User Guide**

**Passenger Guide**

* **Registration and Login**: Steps to register for a new account and log in to the system.
* **Booking Tickets**: Instructions for searching for trains, selecting seats, and booking tickets.
* **Viewing Bookings**: How to view booked tickets and check booking details.
* **Cancelling Bookings**: Steps to cancel booked tickets if needed.

**Administrator Guide**

* **Accessing Admin Panel**: How to access the admin panel using administrator credentials.
* **Managing Trains**: Instructions for adding, editing, and deleting trains from the system.
* **Managing Bookings**: How to view and manage bookings, including cancelling tickets and updating passenger information.

**Maintenance and Support**

**Maintenance Plan**

* **Updates**: Procedures for applying updates, patches, and bug fixes to the system.
* **Backup and Recovery**: Guidelines for data backup and disaster recovery.

**Support Contact**

* **Technical Support**: Contact information for technical support and assistance.