Train Reservation Simulation Developer’s guide

INDEX

[1. Objective 2](#_Toc519778545)

[2. Target Audience 3](#_Toc519778546)

[3. Architecture 4](#_Toc519778547)

[3.1 Security 4](#_Toc519778548)

[3.2 Database 5](#_Toc519778549)

[3.3 Testing 6](#_Toc519778551)

[3.4 Deployment 6](#_Toc519778552)

[4. Development 7](#_Toc519778553)

[4.1 Prerequisites 7](#_Toc519778554)

[4.2 Maven POM 7](#_Toc519778555)

[4.3 Application Properties 7](#_Toc519778556)

[4.4 Unit Tests 8](#_Toc519778557)

[5. Deployment 9](#_Toc519778558)

[6. References 10](#_Toc519778559)

# Objective

This document provides detailed information about the Train Reservation Simulation application in technical terms to allow developers to extend the application and understand various functionalities of the app in detail.

# Target Audience

This document is targeted at anyone who is interested in using this application and modify its features.

# Architecture

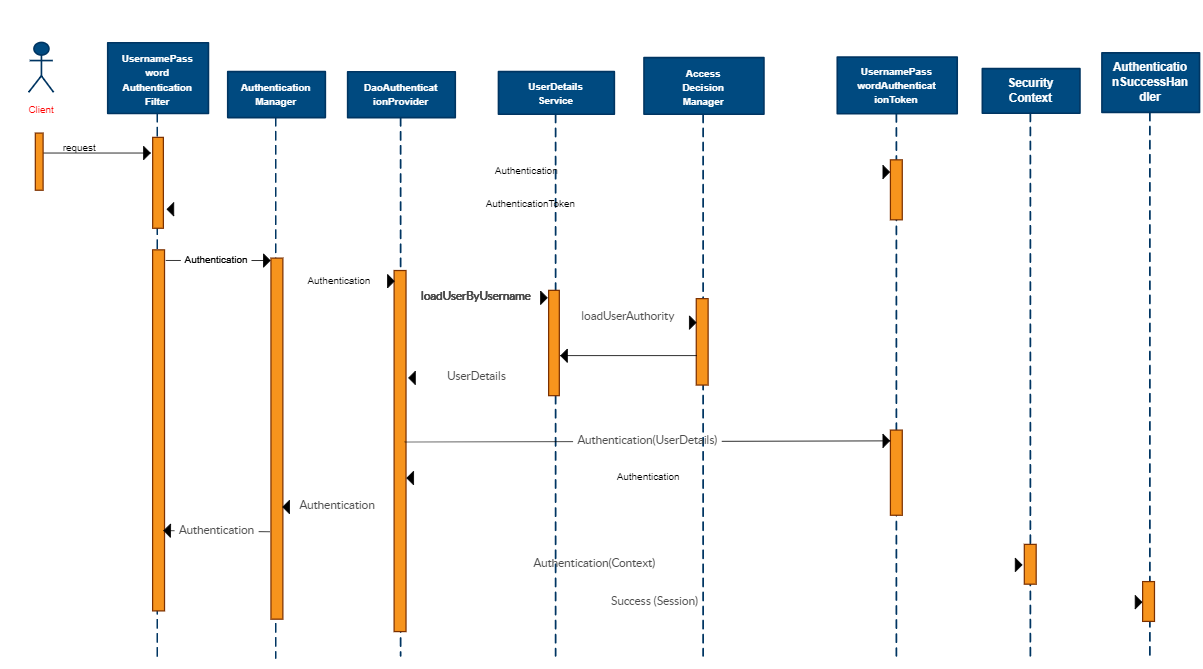
The application uses Spring Boot 1.3.5 along with some other open source libraries as the base framework. The application till now is built with the focus on the following areas.

* Security – Default Spring Security and related validation.
* Database – MySQL
* Testing – Using Junit, Mockito.
* Deployment – (Standalone) Tomcat
* (CI/CD) Jenkins
* (Static code analysis) Sonar

The application has a Maven parent POM. The parent POM defines a minimum set of required and security scanned libraries for building Spring Rest services .

## Security

Spring Security is used to satisfy the authentication requirement of this phase. The required dependencies should be added to the pom.xml file as mentioned in the later section. The sequence diagram below describes the steps in spring security based authentication.



Since we are using spring boot framework the lines of code required to use spring security is drastically reduced.

## Database

The database in this project is just used for storing user details. During registration user details are added to the database. For every subsequent login, the user needs to provide username and password.

The username field needs to be unique. mysql dependency needs to be added in pom.xml file

Name of the database is db\_example alternatively to use your database change the following line in application.properties file.

* spring.datasource.url=jdbc:mysql://localhost:3306/db\_example

The username of database used is "user" and password "password", alternatively these lines in application.properties can be changed.

* spring.datasource.username=user
* spring.datasource.password=password

Refer application.properties for more details regarding database configuration.

Alternatively

And replace the entire contents of application.properties file with

|  |  |
| --- | --- |
|  |  |
|  |  |
|  | spring.mvc.view.prefix: /  spring.mvc.view.suffix: .jsp |
|  | spring.jpa.hibernate.ddl-auto=create |
|  | spring.jpa.show-sql=true |
|  | spring.messages.basename=validation |

## Testing

Testing can never seem to be complete. Junit and Mockito have been used for testing. Junit and spring security based testing dependencies have been added to pom.xml file . This field is still being worked upon

This section is handled in detail in the coming sections.

## Deployment

**Tomcat**

Tomcat needs to be installed to deploy the war file. Alternatively , spring boot’s embedded Tomcat can be used. Deployment will be covered in detail later

**Jenkins**

Jenkins is used for CI/CD. It provides various plugins to support building, automation and deployment. Jenkins war file needs to be downloaded and kept in web-apps for of tomcat. Now once tomcat is started Jenkins is available at tomcathome/jenkins

**Sonar**

Sonar is used for static code analysis. It scans the static code for any code smells.

# Development

## Prerequisites

* JDK 1.8 or later
* Maven 3 or later
* MySQL 5.6 or later
* Tomcat 8.x or later

## Maven POM

Apache Maven has been chosen as the build and dependency management tool . The following dependencies have been used.

spring-boot-starter-parent -> For creating a spring boot project this dependency is required in parent tag

spring-boot-starter-web -> To create a web project

spring-boot-starter-security -> To enable authority based login

spring-boot-starter-data-jpa -> For integrating any database

mysql-connector-java -> To connect to mySql

spring-boot-starter-tomcat -> For deploying the app on tomcat server

tomcat-embed-jasper -> Embed that in the application

jstl -> Template to display jsp pages

spring-boot-starter-test -> For junit testing

yahoo-weather-java-api -> External API for weather forecast details

## Application Properties

The following lines of code correspond to view displayed. This indicates that jsp pages are used for views

spring.mvc.view.prefix: /

spring.mvc.view.suffix: .jsp

Java Persistence API related configuration is specifies in the following lines.

spring.jpa.hibernate.ddl-auto=none

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

spring.jpa.hibernate.naming-strategy=org.hibernate.cfg.ImprovedNamingStrategy

MySQL related configuration is specified in the following lines

spring.datasource.url=jdbc:mysql://localhost:3306/db\_example

spring.datasource.username=user

spring.datasource.password=password

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.datasource.testWhileIdle=true

spring.datasource.validationQuery=SELECT 1

Validation related configuration is specified

spring.messages.basename=validation

Logging is done using sl4j logger, its related configuration is specified here

logging.level.org.springframework.security= DEBUG

logging.level.org.springframework.web= ERROR

## Unit Tests

Unit tests have been written using junit. Integration tests with the database have also been implemented. Testing with spring security requires MockUser annotation. An abstractTest class with required configuration has been created , any subsequent classes can extend that class.

# Deployment

The war file can be deployed on tomcat . To generate the war file

Do a *mvn install* in the project location

And place this file in webapps folder to deploy it on default port

Alternatively the project can be imported in Spring Tool Suite as an existing Maven project and can be run in its embedded Tomcat (by running it as a spring boot app).

*mvn spring-boot:run*

The application will be available on

* http://localhost:{ default tomcat port}/

the user can register or signin with his credentials

# References

<https://creately.com/diagram/example/imcj5bv4/SpringSecurity>

<https://hellokoding.com/registration-and-login-example-with-spring-security-spring-boot-spring-data-jpa-hsql-jsp/>