Spring Framework/Boot Edition

Sample Movie Database

API Services

SOFTWARE REQUIREMENTS SPECIFICATION

December 2021

Table of Contents

1 Introduction 3
1.1 Prerequisites 3
1.2 Technology Stack 3
2 Functional Requirements 4
2.1 Product Perspective 4
2.2 Functionality 4
2.2.1 Domain Model 4
2.2.2 Generate Sample Data 4
2.2.3 Basic Architecture 4
2.2.4 Search 4
2.2.5 Data Backup 5
2.2.6 Reports 5
3 Non-functional Requirements 6
3.1 Software Design 6
3.2 Performance 6
3.3 Logging 6
3.4 Software Quality 6
4 Deliverables 7
4.1 Presentation Date 7
4.2 Source Code Delivery 7

1 Introduction

The introduction of the Sample Movie Database (SMDB) Software Requirements Specification(SRS) provides an overview of the purpose, scope, definitions, and overview of the SRS. This document aims to provide a well-defined set of requirements the SMDB site should implement. The detailed requirements of the SMDB site are provided in this document.

1.1 Prerequisites

You should create a public repository in GitHub where the source code should be pushed.

1.2 Technology Stack

The technology stack you have at your disposal consists of the following components:

- Java 11
- Spring Boot latest production release
- Maven latest production release
- H2 database server

The development IDE of our choice is IntelliJ IDEA. Nevertheless, if you consider the use of another IDE will help, you are free to use that IDE.

To call your endpoints, you may utilize one of the following tools:

- Postman Application
- Intellij HTTP Client
- Curl command-line tool

You are free to use any library as long as it is declared as a dependency via Maven.

2 Functional Requirements

2.1 Product Perspective

Sample Movie Database (SMDB) is an online database of information related to films, television programs – including cast, production crew, plot summaries, genre, ratings, among others.

Based on the fact that we are only covering the API services of the site, every feature will be considered complete upon the successful return of a valid JSON response.

In the following section, we will describe all required functionality.

2.2 Functionality

2.2.1 Domain Model

Our system consists of several entities with the most basic being the films, TV shows (sitcoms) and of course the people (actors, directors, producers etc.) You are free to develop whatever domain you think is appropriate and include whatever attributes per domain class.

As taught, you should base the database model generation in your domain classes.

2.2.2 Generate Sample Data

Once you have concluded the domain model, you should generate some corresponding sample data to feed the services under development. The approach you should follow must be able to re-run at any given moment according to the development team needs.

2.2.3 Basic Architecture

For every domain class, you should create the corresponding class in every basic architecture layer. You should support all basic actions and at least three retrieving actions per domain class.

2.2.4 Search

The site's visitor should be able to freely search throughout the entire set of domain classes. The results though should indicate the type of returning information, i.e. <SEARCH_TOKEN> (Movie), <SEARCH_TOKEN> (Actor)Data Retrieval

Every domain class should have a dedicated controller to serve the respective content. Depending on the domain class, it should also return all associated content.

- In case of returning a film, we need to include all its attributes along with the list of actors and other contributors such as directors, producers.
- In the case of actors, we need to include all the movies and tv shows the actor has participated in.

2.2.5 Data Backup

The system should provide a functionality to export the database data to CSV files. This functionality should be triggered by a REST call. Once completed, we should report back the number of rows exported per domain class.

2.2.6 Reports

The system should support the following reports:

1. Return the top X high-rated content.
2. Return all content associated with a given individual regardless of hir/her contributing role.
3. Return all content associated with a given individual for a given contributing role.
4. Return all content for a given genre.
5. Return the number of shows per genre
6. Return the numbers of shows per year per genre
7. Return all content associated with a given individual organized per genre.

3 Non-functional Requirements

3.1 Software Design

All functionality should be implemented by following the practices covered during the sessions. You should make use of interfaces and abstract classes wherever it seems appropriate and use proper java packaging nomenclature.

3.2 Performance

When it comes to performance, all calls should have a maximum response time of 1 second.

3.3 Logging

There should be a well-defined logging policy maintaining all information considered mandatory to be able to trace user activity and debug erroneous calls. By logging policy, we are referring to the definition of the logging level per specific functionality, potentially more than one log file along with a clear rotation policy.

3.4 Software Quality

When it comes to the quality of the implementing product, we need to make sure every call should always return a valid JSON document including the cases where something went wrong, due to either a development bug or system reasons. The end-user should not see default error pages.

Depending on the type of the call (CRUD actions), the proper HTTP code should be always returned:

- 200 OK
- 201 Created
- 202 Accepted
- 204 No Content
- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 500 Internal Server Error

As far as the quality of the source code is concerned, you should utilize any type of formatting rules you consider appropriate and indicative of your coding style.

Code.Hub

4 Deliverables

4.1 Presentation Date

The project's presentation date is Thursday 13th of January 2022.

4.2 Source Code Delivery

The Github repository hosting your source code is the actual delivery. Although technically, you can push the entire source code at once, we urge you to have multiple commits as this approach will allow us to track your thinking and the approach you followed to develop the SMDB site.

Once ready and besides the team's presentation, you should send an email to c.giannacoulis@codehub.gr containing:

1. Your team's members full names
2. Github repository URL
3. A collection of calls to test your endpoints