Course : CJV805 & Database Connectivity using JAVA	**Digital Video Store Web Application 2 of 2**	**Contribution: 20%**
Instructor: Mehrnaz Zhian	**Date Given: July17,2025 Date Due: Aug7 @ 11:59 pm**	

Notes for the Student: This Project is designed to give you practical experience in building Restful APIs and Server Side applications using Java's Spring Boot

Background: You will need to have access to a code editor. You will also need a thorough understanding of Java, OOP, API design, REST, Spring Boot

Assignment Submission Requirements

- Your source code and git-hub link must be uploaded to Blackboard

Assignment Regulations

- This assignment must be done individually.
- A virtual "in-person" demonstration of this project is required. **The date of the presentation would be in week** .
- Failure to answer questions regarding foundational concepts about Spring Boot and how the said concepts were used within your code would result in 0.
- **Please review Seneca's policies on Academic Integrity, specifically:**

  "Each student should be aware of the College's policy regarding Cheating and Plagiarism. Seneca's Academic Policy will be strictly enforced.To support academic honesty at Seneca College, all work submitted by students may be reviewed for authenticity and originality, utilizing software tools and third party services. Please visit the Academic Honesty site on http://library.senecacollege.ca for further information regarding cheating and plagiarism policies and procedures.
  ." ***Thus, ensure that your code or any part of it is not duplicated by another student(s). This will result in a percentage of zero (0%) assigned to all parties involved.***

Technical Requirements

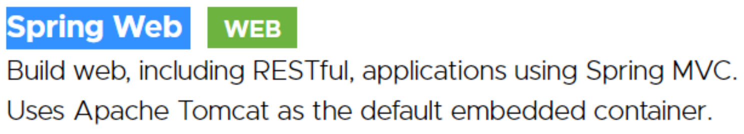
- Your API must be created using Spring Boot.

Detailed App Specification

This assignment is a continuation of Assignment 1 and thus all the requirements for this assignment are to be made “on top” of your Assignment 1. Please note, **you are only required to build the RESTful API for this assignment. For this assignment, you are NOT required to connect it to your React Front-End app.**

Framework

Your RESTful API **MUST** be built using the Spring Boot Java Framework. Create a starter package using the Spring initializer tool: https://start.spring.io/ . Remember to add the Spring Web Dependency to your project



Spring Web WEB

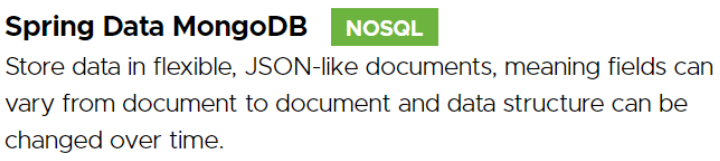
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Database

Your Restful API **MUST** be connected to a MongoDB database.

Regarding your database functionality, the following rules must be followed :

1. Setup and configure a MongoDB cloud service using MongDB Atlas https://www.mongodb.com/cloud/atlas.
2. Connect your API to your mongoDB database using the **Spring Data MongoDB dependency**



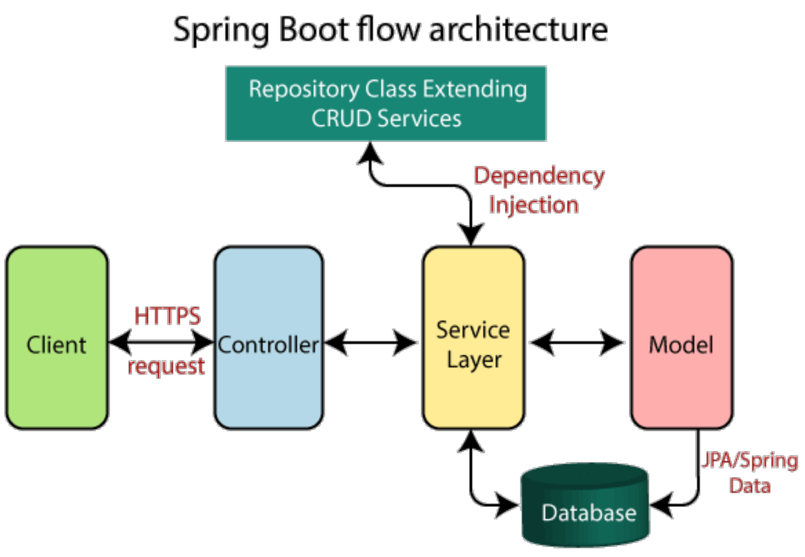
Spring Data MongoDB NOSQL

Store data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time.

3. Name your database and collections appropriately.

Application Architecture

Your RESTful API **MUST** be built in accordance with the MVC (with service layer) design pattern. Thus, you must create Controller classes, Service Classes and Model or Repository Clases

1. 

Endpoints

Test all the below endpoints using Postman or any other API Client to ensure that each endpoint is functional and works.

Ensure that **ALL** endpoints return pertinent JSON data and a status code.

Customer endpoints

1. Create an endpoint that will allow a user to register. The endpoint must accept the same data (in JSON) that was indicated in Assignment 1's registration form and then insert the data into your MongoDB database.

   Also, send an appropriate status code and message (in JSON) to the client if the operation was successful. This endpoint must also provide validation logic (missing data, data in incorrect format,etc) and return the appropriate status code and messages if the incoming data does fail the validation criteria.

   Note, **passwords must not be stored in plain text in the database, thus your application must store passwords in an encrypted format. You can use the Bcrypt Library or any encryption library of your choice.**

2. Create an endpoint that retrieves a specific customer and all it's associating information.This endpoint should also provide validation logic, specifically for endpoints that do not contain a valid customer id.

Movie and TV Show endpoints

1. Create a endpoint that will create movies/tv shows to be added to the database.The below is the data that must be added when a movie is created:
    a. Movie or TV Show name,
    b. Movie/TV Show price,
    c. Synopsis of the movie or tv show,
    d. A value to represent if the item is a movie or tv show
    e. Movie/TV show small poster (image path of the movie/tv show)
    f. Movie/TV show large poster (image path of the movie /tv show)
    g. The price to rent the movie or tv show
    h. The price to purchase the movie or tv show outright.
    i. A field to determine if the movie or tv show is a featured movie or tv show

2. Create an endpoint that retrieves all the movies in the database.

3. Create an endpoint that retrieves all the tv shows in the database.

4. Create an end point that allows a user to supply a title of a movie and/or tv show and allow the API to return a list of movies and/or tv shows that CONTAINS the supplied title

5. Create an endpoint that will retrieve the featured movies in the database. Your end point MUST contain a query string

6. Create an endpoint that will retrieve the featured tv shows in the database. Your end point MUST contain a query string

7. Create an endpoint that will retrieve a specific movie or tv show in the database. This endpoint should also provide validation logic, specifically for endpoints that do not contain a valid movie id.

8. Create an endpoint that will update and change an existing movie in the database. For example, **changing movie rent prices, etc.** This endpoint should also provide validation logic, specifically for endpoints that do not contain a valid movie id and missing incoming data.

9. Create an endpoint that will delete an existing movie or tv show in the database. This endpoint should also provide validation logic, specifically for endpoints that do not contain a valid movie id.

User and Authentication endpoints

1. Create an endpoint that will create users. The below is the data that must be added when a user is created:
    a. First Name
    b. Last Name
    c. Email
    d. Password

2. You are required to create an endpoint that will authenticate a user.A successful authentication occurs when the email and password pair, passed to the endpoint, exists in the database. An unsuccessful authentication occurs when the email and password pair does not exist in the database.

Rubric

-Home View - Your Home view in your React App is connected to your Spring Boot Back-End. It pulls the data from the pertinent endpoint and populates the Featured Movie Section & the Featured TV Show Section .	**Not Implemented 0**	**Partially Implemented 3**	**Fully Implemented 6**
Movie/TV Show Listing view - Your Movie & TV Show Listing view in your React App, pull its data from the appropriate end point in your Back-End	**Not Implemented 0**	**Partially Implemented 2.5**	**Fully Implemented 5**
Movie/TV SHow Description View - Your Movie/TV SHow Description view in your React App, gets its data from the appropriate end point in your	**Not Implemented 0**	**Partially Implemented 2.5**	**Fully Implemented 5**
Movie Search -Ensure that when the user searches for a movie and/or tv show, the list of movies/tv shows is returned from the appropriate end point in your Back-End and renders the data in your React App	**Not Implemented 0**	**Partially Implemented 3**	**Fully Implemented 6**

User Registration- When the user submits the registration form in the React app, their data is posted to the appropriate end point in Back-End. Once validated, their data should then be inserted into the database.	**Not Implemented 0**	**Partially Implemented 3**	**Fully Implemented 6**
Log-in - Ensure that when the user submits the log-in form in the React app, their data is posted to the appropriate end point in Back-End. Once validated, the user should be navigated to a dashboard showing their personal information. The dashboard page can only be	**Not Implemented 0**	**Partially Implemented 3**	**Fully Implemented 6**

Total : 34 MARKS

THE END