Object Oriented Analysis and Design using Java

Self Learning Hands-on Assignment: MVC Framework

MVC (Model-View-Controller) is one of the most widely used software architecture patterns for app and web development. The Model View Controller pattern separates concerns into model, view and controller. The Model contains data, View has the presentation logic and Controller has the backend code for application logic. Using this framework, we can separate the business logic from presentation logic.

Problem statement:

Design a simple movie database website allows users to browse and rate movies (think an extremely simplified version of letterboxd). It follows the MVC pattern, separating the application into three main components: the Model, the View, and the Controller. Use any Java framework of your choice, including but not limited to :

− Spring

− Grails

− Vaadin

− Play

− Struts

− Java Server Faces

Make sure your project directory contains the Model, View and Controller subdirectories. In case of some frameworks like Spring, you'll also have to include an extra Service class that handles business logic.

Features:

1. **Movie Catalog (View) :**

Users can view a list of movies with details such as title, release year, genre, director, and average rating.

2. **Data Handling (Model):**

   The Model component manages the application's data, including movies, user ratings, reviews, and user accounts.

   Movies and user data are stored in a database (e.g., MySQL, PostgreSQL).

   The Model component provides methods for CRUD operations (Create, Read, Update, Delete) on movies, ratings, reviews, and user accounts.

3. **Controller:**

   The Controller component handles user requests, processes input from the View, and interacts with the Model to retrieve or modify data.

   It routes requests to the appropriate handlers based on the URL and HTTP method (e.g., GET requests for retrieving movies, POST requests for adding reviews).

Implementation:

Given below are templates for how your code must be structured.

Keep in mind these are generalized guidelines (each framework consists of their own conventions) and you'll have to enhance it with additional features, error handling, and database integration based on the frameworks you've chosen, your specific requirements and environment.

Model (Data Layer): Implement the data model for movies, user ratings, and reviews. Define relationships between entities and methods for read and write operations.

```
public class Movie {



   private Long id;

   private String title;

   private int releaseYear;

   private String genre;

   private String director;

   private double averageRating;



   // Constructors, getters, and setters



}
```

View (Presentation Layer): Create a simple frontend for displaying movies and movie details.

The View layer can be implemented using your preferred framework to render the frontend interface for displaying movies and movie details.

Controller (Application Logic): Develop the application logic to handle user requests, validate input, interact with the Model, and render appropriate responses. Use a web framework that supports MVC architecture.

```
import java.util.ArrayList;

import java.util.List;



public class MovieController {



   private List<Movie> movieList;



   public MovieController() {



       movieList = new ArrayList<>();

       movieList.add(new Movie(1L, "Movie 1", 2022, "Action", "Director 1", 4.5));

       movieList.add(new Movie(2L, "Movie 2", 2019, "Comedy", "Director 2", 3.8));

       movieList.add(new Movie(3L, "Movie 3", 2020, "Drama", "Director 3", 4.2));

       // Add more movies as needed

   }



   public List<Movie> getAllMovies() {

       // Return the list of all movies

       return movieList;

   }







   public void rateMovie(Long id, double rating) {

       // Update the rating of the movie with the specified ID
```

```
        for (Movie movie : movieList) {
            if (movie.getId().equals(id)) {
                // Update the average rating (implementation details may vary)
                movie.setAverageRating((movie.getAverageRating() + rating) / 2);
                break;
            }
        }
    }

    // Other controller methods for CRUD operations (Create, Update, Delete) can be added here

}
```

Objectives:

- Understand the structure and benefits of the MVC architectural pattern in web development.
- Learn how to integrate front-end and back-end components using a web framework.

For submission, create a PDF document with the following

1. A write up on how you have used MVC within your project, highlighting the respective concepts.
2. Complete screenshot of code written by you
3. Screenshot of console with application running
4. Screenshot of UIs related to the two scenarios with values, outputs, errors (if any)
5. Screenshot of database with data items