Object Oriented Analysis and Design using Java - UE21CS352B

NAME: SIRI GOWRI H

SRN: PES1UG21CS599

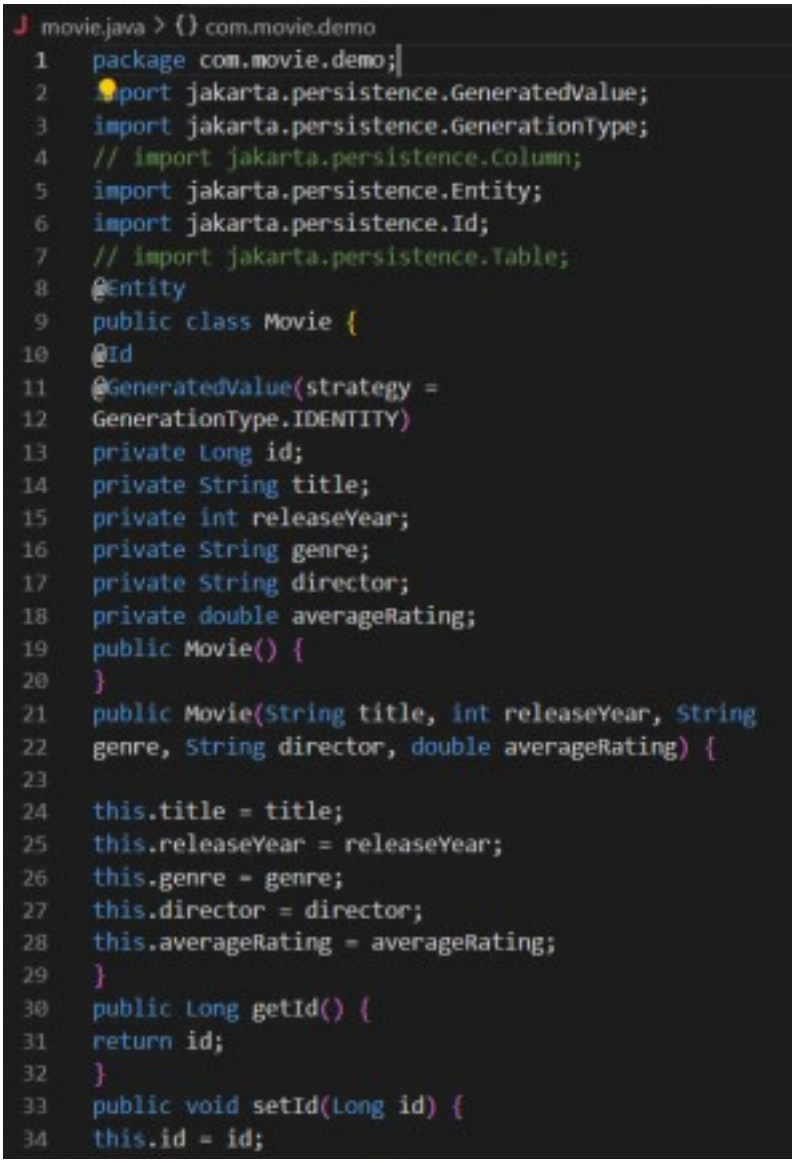
Q) Design a simple movie database website allows users to browse and rate movies (think an extremely simplified version of letterboxd). It follows the MVC pattern, separating the application into three main components: the Model, the View, and the Controller. A write up on how you have used MVC within your project, highlighting the respective concepts.

1. **Model:** In our movie database project, the Model component serves as the backbone, embodying both the data structure and the core business logic. The Movie class encapsulates essential information about each movie, while CRUD operations are meticulously implemented to ensure data integrity and seamless interaction with the underlying database. By maintaining a clear focus on data management and manipulation, the Model facilitates the reliable storage and retrieval of movie-related information, forming the foundation upon which the entire application rests.
2. **View:** As the face of our application, the View layer is entrusted with presenting the movie catalog in an engaging and user-friendly manner. Here, users can explore various movie details such as title, release year, genre, director, and average rating without being encumbered by technical complexities. Decoupled from the underlying logic, the View fosters adaptability and flexibility in presentation, accommodating different front-end frameworks or technologies seamlessly. Through its intuitive interface, the View empowers users to navigate the movie database effortlessly, enhancing their overall browsing experience.
3. **Controller:** Acting as the conductor of our application's symphony, the Controller layer orchestrates the interaction between the View and the Model. It serves as the intermediary, translating user actions into meaningful operations within the application. By receiving and processing user requests, such as retrieving movie data or submitting ratings, the Controller ensures proper flow of control while safeguarding the View from the intricacies of data manipulation. Through its judicious invocation of Model methods, the Controller enables seamless communication between different components, fostering a cohesive and responsive user experience.

2. Complete screenshot of code written by you.

movieController.java > ...

```java
package com.movie.demo;
import
org.springframework.beans.factory.annotation.Autowired
;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import
org.springframework.web.bind.annotation.GetMapping;
import com.movie.demo.repository.MovieRepository;
// import com.movie.demo.Movie;
import java.util.List;

@Controller
public class MovieController {
@Autowired
private MovieRepository movieRepository;
@GetMapping("/movies")
public String getAllMovies(Model model) {
List<Movie> movies =
movieRepository.findAll();

model.addAttribute("movies", movies);
return "movies";
}
}
```


movie.java > {} com.movie.demo
package com.movie.demo;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
// import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
// import jakarta.persistence.Table;
@Entity
public class Movie {
@Id
@GeneratedValue(strategy =
GenerationType.IDENTITY)
private Long id;
private String title;
private int releaseYear;
private String genre;
private String director;
private double averageRating;
public Movie() {
}
public Movie(String title, int releaseYear, String
genre, String director, double averageRating) {
this.title = title;
this.releaseYear = releaseYear;
this.genre = genre;
this.director = director;
this.averageRating = averageRating;
}
public Long getId() {
return id;
}
public void setId(Long id) {
this.id = id;

```
35    }
36    public String getTitle() {
37    return title;
38    }
39
40    public void setTitle(String title) {
41    this.title = title;
42    }
43    public int getReleaseYear() {
44    return releaseYear;
45    }
46    public void setReleaseYear(int releaseYear) {
47    this.releaseYear = releaseYear;
48    }
49    public String getGenre() {
50    return genre;
51    }
52    public void setGenre(String genre) {
53    this.genre = genre;
54    }
55    public String getDirector() {
56    return director;
57    }
58    public void setDirector(String director) {
59    this.director = director;
60    }
61    public double getAverageRating() {
62    return averageRating;
63    }
64    public void setAverageRating(double averageRating)
65    {
66
67    this.averageRating = averageRating;
68    }
69    }
```

review.java > ...

```java
package com.movie.demo;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
@Entity
public class Review {
@Id
@GeneratedValue(strategy =
GenerationType.IDENTITY)
private Long id;
private Long movieId;
private Long userId;
private String comment;
public Review() {
}
public Review(Long movieId, Long userId, String
comment) {

this.movieId = movieId;
this.userId = userId;
this.comment = comment;
}

public Long getId() {
return id;
}
public void setId(Long id) {
this.id = id;
}
public Long getMovieId() {
return movieId;
}
public void setMovieId(Long movieId) {
this.movieId = movieId;
}
public Long getUserId() {
```

```java
}
public Long getUserId() {
return userId;
}
public void setUserId(Long userId) {
this.userId = userId;
}
public String getComment() {
return comment;
}
public void setComment(String comment) {
this.comment = comment;
}
}
```

user.java > ...

```java
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
@Entity
public class User {
@Id
@GeneratedValue(strategy =
GenerationType.IDENTITY)
private Long id;
private String username;
private String password;
public User() {
}
public User(String username, String password) {
this.username = username;
this.password = password;
}
public Long getId() {
return id;
}
public void setId(Long id) {
this.id = id;
}

public String getUsername() {
return username;
}
public void setUsername(String username) {
this.username = username;
}
public String getPassword() {
return password;
}
public void setPassword(String password) {
this.password = password;
}
```

userRating.java > ...

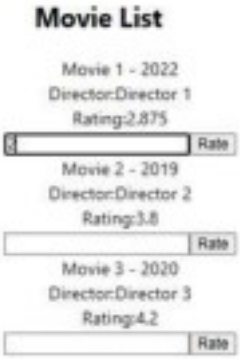
```java
public package com.movie.demo;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
@Entity
public class UserRating {
@Id
@GeneratedValue(strategy =
GenerationType.IDENTITY)
private Long id;

private Long movieId;
private Long userId;
private double rating;
public UserRating() {
}
public UserRating(Long movieId, Long userId,
double rating) {

this.movieId = movieId;
this.userId = userId;
this.rating = rating;
}
public Long getId() {
return id;
}
public void setId(Long id) {
this.id = id;
}
public Long getMovieId() {
return movieId;
}
public void setMovieId(Long movieId) {
this.movieId = movieId;
}
```

```java
public Long getUserId() {
return userId;
}

public void setUserId(Long userId) {
this.userId = userId;
}
public double getRating() {
return rating;
}
public void setRating(double rating) {
this.rating = rating;
}
} {

}

```

movie.html > ...

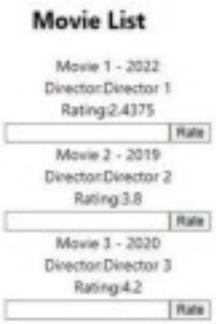
```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width,
initial-scale=1.0" />
<title>Movies</title>
</head>
<body>
<h1>Movie Catalog</h1>
<table>
<thead>
<tr>
<th>Title</th>
<th>Release Year</th>
<th>Genre</th>
<th>Director</th>
<th>Average Rating</th>

</tr>
</thead>
<tbody>
<tr th:each="movie : ${movies}">
<td th:text="${movie.title}"></td>
<td th:text="${movie.releaseYear}"></td>
<td th:text="${movie.genre}"></td>
<td th:text="${movie.director}"></td>
<td th:text="${movie.averageRating}"></td>
</tr>
</tbody>
</table>
</body>
</html>
```

3. Screenshot of console with application running



4. Screenshot of UIs related to the two scenarios with values, outputs, errors (if any) **Before Rating**

After Rating



5.Screenshot of database with data items

movies.sql

```sql
-- Create 'movies' table
CREATE TABLE movies (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    releaseYear INT NOT NULL,
    genre VARCHAR(100) NOT NULL,
    director VARCHAR(255) NOT NULL,
    averageRating DOUBLE DEFAULT 0
);

-- Create 'ratings' table
CREATE TABLE ratings (
    id BIGINT AUTO_INCREMENT PRIMARY KEY,
    userId BIGINT NOT NULL,
    movieId BIGINT NOT NULL,
    rating DOUBLE NOT NULL,
    FOREIGN KEY (movieId) REFERENCES movies(id)
);

-- Sample data for 'movies' table
INSERT INTO movies (title, releaseYear, genre, director, averageRating) VALUES
('The Shawshank Redemption', 1994, 'Drama', 'Frank Darabont', 9.3),
('The Godfather', 1972, 'Crime, Drama', 'Francis Ford Coppola', 9.2),
('The Dark Knight', 2008, 'Action, Crime, Drama', 'Christopher Nolan', 9);

-- Sample data for 'ratings' table
INSERT INTO ratings (userId, movieId, rating) VALUES
(1, 1, 9),
(2, 1, 10),
(1, 2, 8),
(2, 3, 9);

-- Ensure the database uses UTF-8 encoding to support international characters
ALTER DATABASE your_database_name CHARACTER SET = utf8mb4 COLLATE = utf8mb4_unicode_ci;
ALTER TABLE movies CONVERT TO CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
ALTER TABLE ratings CONVERT TO CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```