OOAD LAB MVC - 1

ARVIN NOOLI
PES1UG21CS112

1. USE OF MVC CONCEPTS:

1. My model is the 'Movie' class, which represents data for a movie entity. It includes information such as title, release year, genre, director, and average rating that describe the current status of a film. This class normally has getter and setter methods for accessing and modifying its attributes. In MVC, the model is in charge of handling the application's data and business logic, and the 'Movie' class serves this purpose by representing movie-related data.

2. View: the 'index.html' file in the 'static' subdirectory serves as the view component. The view is in charge of delivering the user interface and engaging with the users. The HTML and any client-side scripts in the 'index.html' file define the user interface's structure and look.

3. The MovieController class serves as our project's controller component.
It handles incoming HTTP requests from clients, interacts with the model to retrieve or alter data, and chooses the best view to render as a response. Each method in the MovieController is linked to a specific endpoint (URL) in the application's API. For example, the getAllMovies() method obtains a list of all movies from the database (model) and returns it to the client. Similarly, the addMovie() method accepts a new movie object from the client, stores it in the database, and returns the saved movie.

4. When a user interacts with the application by sending an HTTP request (for example, GET or POST), the DispatcherServlet receives the request. The DispatcherServlet routes the request to the appropriate controller method based on the URL and HTTP method. The controller method communicates with the model to carry out any necessary business logic, such as retrieving or updating data. After processing the request, the controller method produces a response, usually in the form of data (for API endpoints) or a view name (for web pages). If the answer contains a view name, the DispatcherServlet selects the relevant view template (for example, an HTML file) and returns it to the client.

2. CODE

Model:

```
package com.example.demo.model;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
```

```java
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name = "movies")
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 avgRating;

    public Movie() {
    }

    public Movie(Long id, String title, int releaseYear, String genre, String director
, Double avgRating) {
        this.id = id;
        this.title = title;
        this.releaseYear = releaseYear;
        this.genre = genre;
        this.director = director;
        this.avgRating = avgRating;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public int getReleaseYear() {
        return releaseYear;
    }
    public void setReleaseYear(int releaseYear) {
        this.releaseYear = releaseYear;
    }
    public String getGenre() {
        return genre;
    }
    public void setGenre(String genre) {
        this.genre = genre;
    }
    public String getDirector() {
        return director;
    }
    public void setDirector(String director) {
        this.director = director;
    }

    public Double getAvgRating() {
        return avgRating;
    }

    public void setAvgRating(Double avgRating) {
        this.avgRating = avgRating;
    }
}
```

```java
package com.example.demo.model;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.OneToOne;
import jakarta.persistence.Table;

@Entity
@Table(name = "ratings")
public class Rating {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @ManyToOne
    @JoinColumn(name = "movie_id")
    private Movie movie;
    @OneToOne
    @JoinColumn(name = "user_id")
    private User user;
    private double rating;
    public Rating() {
    }

    public Rating(Long id, Movie movie, double rating, User user) {
        this.id = id;
        this.movie = movie;
        this.rating = rating;
        this.user = user;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public Movie getMovie() {
        return movie;
    }

    public void setMovie(Movie movie) {
        this.movie = movie;
    }

    public double getRating() {
        return rating;
    }

    public void setRating(double rating) {
        this.rating = rating;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }
}
```

```java
package com.example.demo.model;
```

```java
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import jakarta.persistence.OneToOne;
import jakarta.persistence.Table;

@Entity
@Table(name = "reviews")
public class Review {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @ManyToOne
    @JoinColumn(name = "movie_id")
    private Movie movie;
    @OneToOne
    @JoinColumn(name = "user_id")
    private User user;
    private String reviewText;

    public Review() {
    }

    public Review(Long id, Movie movie, String reviewText,User user) {
        this.id = id;
        this.movie = movie;
        this.reviewText = reviewText;
        this.user = user;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public Movie getMovie() {
        return movie;
    }

    public void setMovie(Movie movie) {
        this.movie = movie;
    }

    public String getReviewText() {
        return reviewText;
    }

    public void setReviewText(String reviewText) {
        this.reviewText = reviewText;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }
}
```

```java
package com.example.demo.model;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table(name = "users")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String username;
    private String password;
    private String email;

    public User() {
    }

    public User(Long id,String username, String password, String email) {
        this.id = id;
        this.username = username;
        this.password = password;
        this.email = email;
    }
    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;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

Controller:

```
public class MovieController {
    @Autowired
    private MovieService movieService;

    @GetMapping
    public List<Movie> getAllMovies() {
        return movieService.getAllMovies();
    }

    @GetMapping("/{id}")
    public ResponseEntity<Movie> getMovieById(@PathVariable Long id) {
        Optional<Movie> movie = movieService.getMovieById(id);
```

```java
        if (movie.isPresent()) {
            return new ResponseEntity<Movie>(movie.get(), HttpStatus.OK);
        } else {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }
}
```

```java
package com.example.demo.controller;
import java.util.List;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.example.demo.model.Movie;
import com.example.demo.service.MovieService;

@RestController
@RequestMapping("/movies")
public class MovieController {
    @Autowired
    private MovieService movieService;

    @GetMapping
    public List<Movie> getAllMovies() {
        return movieService.getAllMovies();
    }

    @GetMapping("/{id}")
    public ResponseEntity<Movie> getMovieById(@PathVariable Long id) {
        Optional<Movie> movie = movieService.getMovieById(id);
        if (movie.isPresent()) {
            return new ResponseEntity<Movie>(movie.get(), HttpStatus.OK);
        } else {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }
}
```

```java
package com.example.demo.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.example.demo.model.Movie;
import com.example.demo.model.Review;
import com.example.demo.model.ReviewRequest;
import com.example.demo.service.ReviewService;

@RestController
@RequestMapping("/reviews")
public class ReviewController {

    @Autowired
    private ReviewService reviewService;

    @GetMapping("/{movieId}")
    public List<Review> getReviewsForMovie(@PathVariable Long movieId) {
        Movie movie = new Movie();
        movie.setId(movieId);
        return reviewService.getReviewsForMovie(movie);
    }

    @PostMapping("/add")
    public ResponseEntity<String> addReviewForMovie(@RequestBody ReviewRequest
reviewRequest) {
        // Check if movieId and reviewText are present in the request body
        if (reviewRequest.getMovieId() == null || reviewRequest.getReviewText() ==
null) {
            return ResponseEntity.badRequest().body("Movie ID and review text must be
provided");
        }

        // Retrieve movie by ID
        Movie movie = new Movie();
        movie.setId(reviewRequest.getMovieId());

        // Create a Review object
        Review review = new Review();
        review.setMovie(movie);
        review.setReviewText(reviewRequest.getReviewText());

        boolean success = reviewService.addReview(review);
        if (success) {
            return ResponseEntity.status(HttpStatus.CREATED).body("Review added
successfully");
        } else {
            return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Failed to add review");

    }
    }
}
```

```
package com.example.demo.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import com.example.demo.model.User;
import com.example.demo.service.UserService;

@RestController
@RequestMapping("/users")
public class UserController {

    @Autowired
    private UserService userService;

    // Endpoint for creating a new user
    @PostMapping
    public ResponseEntity<String> createUser(@RequestBody User user) {
        userService.createUser(user);
        return ResponseEntity.status(HttpStatus.CREATED).body("User created
successfully");
    }

    // Endpoint for retrieving user details by id
    @GetMapping("/{id}")
    public ResponseEntity<User> getUserById(@PathVariable Long id) {
        User user = userService.getUserById(id);
        if (user != null) {
            return ResponseEntity.ok(user);
        } else {
            return ResponseEntity.notFound().build();
        }
    }
    @GetMapping("/validate")
    public ResponseEntity<Long> validateUser(
            @RequestParam(name = "username") String username,
            @RequestParam(name = "password") String password) {

        User user = userService.getUserByUsername(username);
        if (user != null && user.getPassword().equals(password)) {
            return ResponseEntity.ok(user.getId());
        } else {
            return ResponseEntity.notFound().build();
        }
    }
    // Endpoint for updating user details
    @PutMapping("/{id}")
    public ResponseEntity<String> updateUser(@PathVariable Long id, @RequestBody User
user) {
        user.setId(id);
        userService.updateUser(user);
        return ResponseEntity.ok("User updated successfully");
    }

    // Endpoint for deleting a user by id
    @DeleteMapping("/{id}")
    public ResponseEntity<String> deleteUser(@PathVariable Long id) {
        userService.deleteUser(id);
        return ResponseEntity.ok("User deleted successfully");
    }
}
```

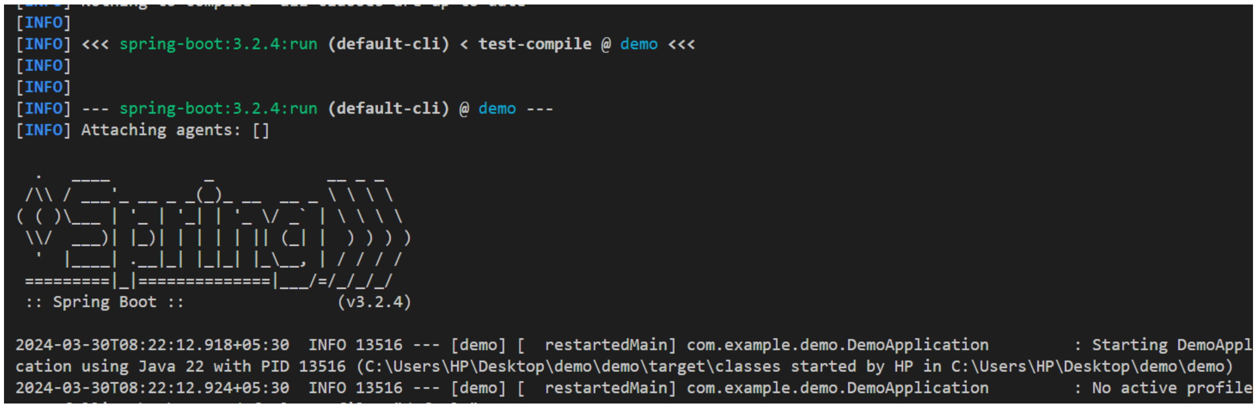
View:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f2f2f2;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            margin: 0;
        }
        .container {
            width: 300px;
            padding: 20px;
            background-color: #fff;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        h2 {
            text-align: center;
            color: #333;
        }
        label {
            display: block;
            margin-bottom: 5px;
            color: #555;
        }
        input[type="text"],
        input[type="password"],
        button {
            width: 100%;
            padding: 10px;
            margin-bottom: 10px;
            border: 1px solid #ccc;
            border-radius: 3px;
            box-sizing: border-box;
        }
        button {
            background-color: #007bff;
            color: #fff;
            cursor: pointer;
        }
        button:hover {
            background-color: #0056b3;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Login</h2>
        <form id="loginForm" onsubmit="login(event)">
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" required>
            <label for="password">Password:</label>
            <input type="password" id="password" name="password" required>
            <button type="submit">Login</button>
        </form>
    </div>
    <script>
        function login(event) {
            event.preventDefault(); // Prevent the default form submission

            // Get the username and password from the form
            var username = document.getElementById('username').value;
            var password = document.getElementById('password').value;

            // Make a Fetch request to the validate endpoint
            fetch('http://localhost:8080/users/validate?username=' +
encodeURIComponent(username) + '&password=' + encodeURIComponent(password))
                .then(response => {
                    if (response.ok) {
                        // If the response is successful, redirect to movieslist.html
                        window.location.href = 'movieslist.html';
                    } else {
                        alert('Invalid username or password');
                    }
                })
                .catch(error => {
                    console.error('Error:', error);
                    alert('An error occurred. Please try again.');
                });
        }
    </script>
</body>
</html>
```

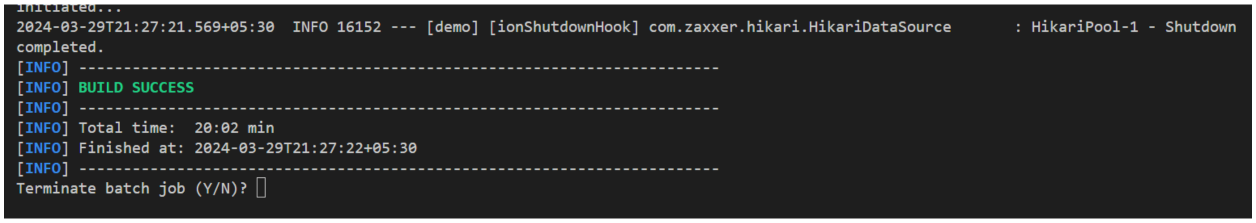
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Movies List</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f2f2f2;
        }
        .container {
            width: 600px;
            margin: 20px auto;
            padding: 20px;
            background-color: #fff;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        h2 {
            color: #333;
            text-align: center;
        }
        ul {
            list-style-type: none;
            padding: 0;
        }
        li {
            margin-bottom: 10px;
        }
        a {
            text-decoration: none;
            color: #007bff;
        }
        a:hover {
            text-decoration: underline;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Movies List</h2>
        <ul>
            <li><a href="/movies/1">Movie 1</a></li>
            <li><a href="/movies/2">Movie 2</a></li>
            <li><a href="/movies/3">Movie 3</a></li>
        </ul>
    </div>

</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Movie View</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f2f2f2;
        }
        .container {
            width: 600px;
            margin: 20px auto;
            padding: 20px;
            background-color: #2d817e;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
        h2 {
            color: #333;
        }
        p {
            color: #f4efef;
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Movie Details</h2>
        <p><strong>Movie Title:</strong> Movie 2</p>
        <p><strong>Release Year:</strong> 2019</p>
        <p><strong>Genre:</strong> Comedy</p>
        <p><strong>Director:</strong> Director 2</p>
        <p><strong>Average Rating:</strong> 3.8</p>

    </div>
</body>
</html>
```

3. CONSOLE SCREENSHOT

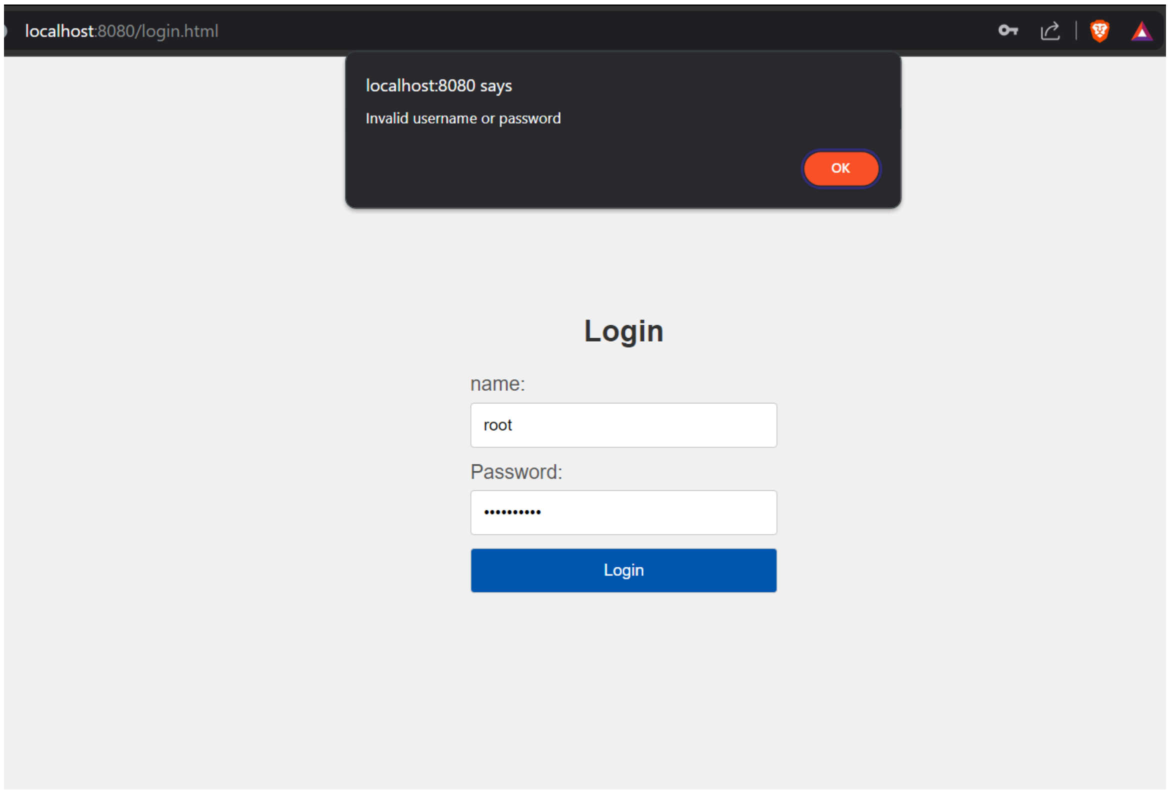


```
[INFO]
[INFO] <<< spring-boot:3.2.4:run (default-cli) < test-compile @ demo <<<
[INFO]
[INFO]
[INFO] --- spring-boot:3.2.4:run (default-cli) @ demo ---
[INFO] Attaching agents: []

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v3.2.4)

2024-03-30T08:22:12.918+05:30  INFO 13516 --- [demo] [  restartedMain] com.example.demo.DemoApplication         : Starting DemoAppl
cation using Java 22 with PID 13516 (C:\Users\HP\Desktop\demo\demo\target\classes started by HP in C:\Users\HP\Desktop\demo\demo)
2024-03-30T08:22:12.924+05:30  INFO 13516 --- [demo] [  restartedMain] com.example.demo.DemoApplication         : No active profile
```



```
2024-03-29T21:27:21.569+05:30  INFO 16152 --- [demo] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown
completed.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  20:02 min
[INFO] Finished at: 2024-03-29T21:27:22+05:30
[INFO] ------------------------------------------------------------------------
Terminate batch job (Y/N)?
```

4. UIs SCREENSHOT



localhost:8080/login.html

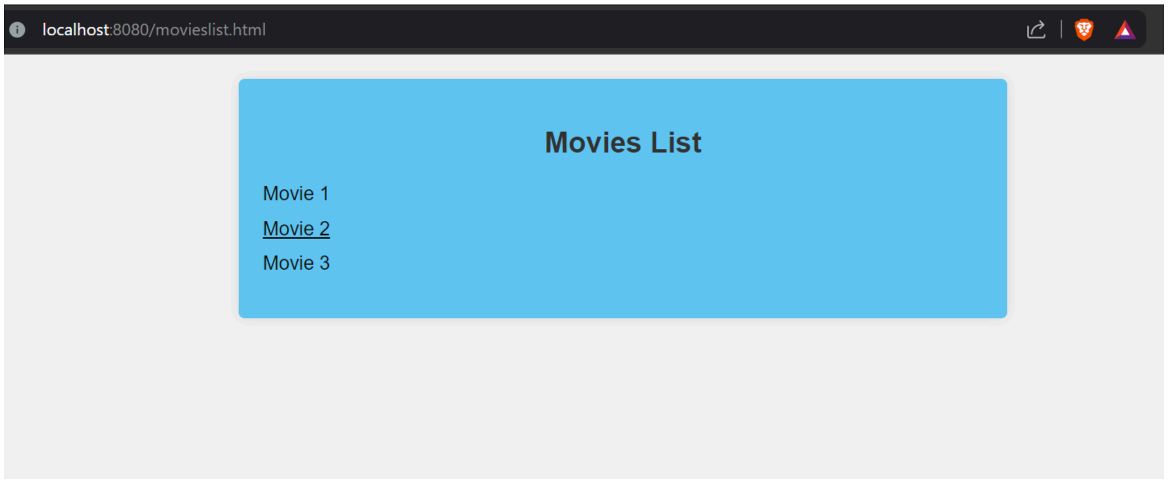
localhost:8080 says

Invalid username or password

Login

name: root

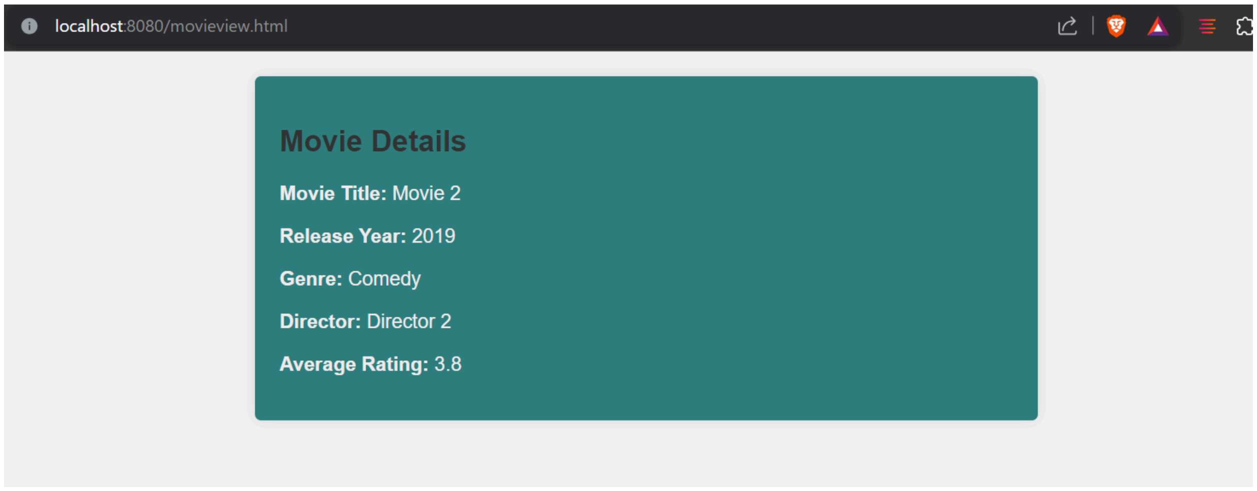
Password: ••••••••••



localhost:8080/movieslist.html

Movies List

Movie 1
Movie 2
Movie 3



localhost:8080/movieview.html

Movie Details

Movie Title: Movie 2

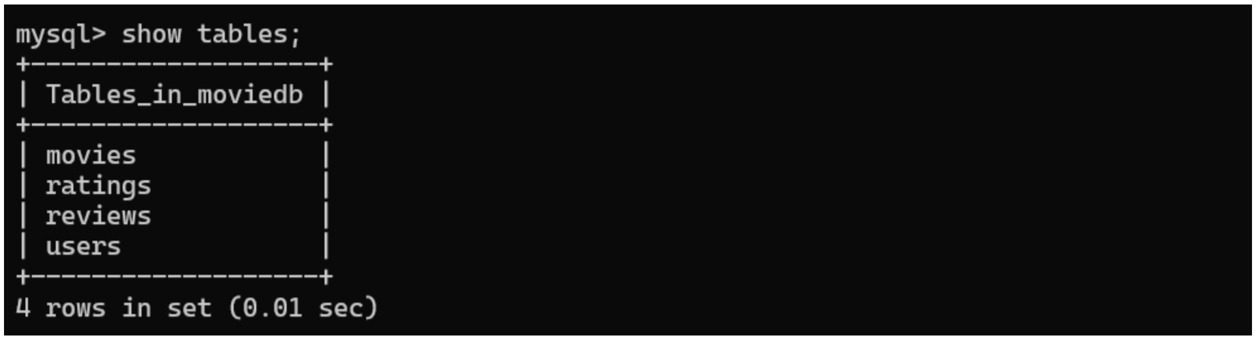
Release Year: 2019

Genre: Comedy

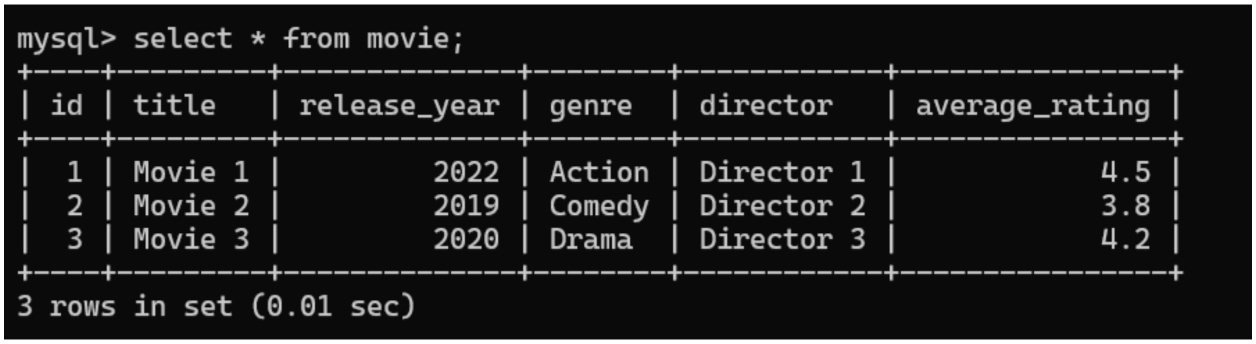
Director: Director 2

Average Rating: 3.8

5. DATABASE SCREENSHOT



```
mysql> show tables;
+--------------------+
| Tables_in_moviedb |
+--------------------+
| movies             |
| ratings            |
| reviews            |
| users              |
+--------------------+
4 rows in set (0.01 sec)
```



```
mysql> select * from movie;
+----+---------+--------------+--------+------------+----------------+
| id | title   | release_year | genre  | director   | average_rating |
+----+---------+--------------+--------+------------+----------------+
|  1 | Movie 1 |         2022 | Action | Director 1 |            4.5 |
|  2 | Movie 2 |         2019 | Comedy | Director 2 |            3.8 |
|  3 | Movie 3 |         2020 | Drama  | Director 3 |            4.2 |
+----+---------+--------------+--------+------------+----------------+
3 rows in set (0.01 sec)
```