**Lab Test 5 EECS 2311**

**Activity 1:**

User Stories:

* Login and register
* SQLite database and MySQL database
* Page for individual movies
* Discovery page with movies
* and genre filter options
* Search function

What you’re working on:

* Login and register and the .dat file: Suneel
* SQLite database and MySQL database: Rythem
* Page for individual movies: Sarah
* Genre filter options: Gjergj
* Search function: Andy

**Take-Home Assignment: Movie Page Testing and Code Review**

**Activity 1: End-to-End Testing**

**User Story: Page for Individual Movies**

The individual movie page should allow users to view detailed information about a selected movie, including its title, release date, and other relevant details. The user should also be able to navigate back to the previous screen. Users must be logged in to access movie information.

**End-to-End Manual Test Cases**

**Test Case 1: User Login Required for Movie Access**

**Steps:**

1. Launch the application.
2. Try to navigate to a movie details page without logging in.
3. Observe if the system restricts access.

**Expected Result:**

* The user should be redirected to the login page if they are not logged in.
* After logging in, the user should be able to access the movie page.

**Edge Cases:**

* Test logging in with invalid credentials.
* Try logging in and logging out multiple times to see if session handling is correct.

**Test Case 2: Movie Page Loads Successfully**

**Steps:**

1. Log in to the application.
2. Navigate to the discovery page.
3. Click on a movie to open its detail page.

**Expected Result:**

* The movie page should load, displaying the title and release date without crashing.

**Edge Cases:**

* Try opening the page for a movie with missing details (e.g., no title or release date).
* Test with a movie that has a long title.

**Test Case 3: Back Button Functionality**

**Steps:**

1. Open the movie page.
2. Click the "Back" button.

**Expected Result:**

* The page should navigate back to the discovery page.

**Edge Cases:**

* Try pressing the "Back" button multiple times quickly.
* Verify behavior when using the keyboard's back button (if applicable).

**Test Case 4: UI Validation**

**Steps:**

1. Open a movie page with different movies.
2. Observe the font, colors, and alignment.

**Expected Result:**

* The UI should be readable.
* The title should be in a larger font.
* Colors should maintain contrast.

**Edge Cases:**

* Test on different screen resolutions.
* Check for proper alignment when long titles are displayed.

**Activity 2: Code Review and Bug Reporting**

**Bug Reports**

**Bug 1: Missing Login Check for Movie Page Access**

**Description:** Users should not be able to access movie details without logging in. **Steps to Reproduce:**

1. Open the application without logging in.
2. Try accessing a movie details page via direct URL or discovery page.
3. Observe whether access is granted. **Expected Result:**

* The system should redirect the user to the login page before granting access.

**Bug 2: Missing Error Handling for Null Movie Data**

**Description:** When a movie object has missing fields (e.g., title or release date), the application may throw a null pointer exception. **Steps to Reproduce:**

1. Open the movie details page for a movie with missing data.
2. Observe if the UI fails to render properly or crashes. **Expected Result:**

* The page should gracefully handle missing data by displaying placeholder text or a relevant message.

**Bug 3: Hardcoded UI Styles**

**Description:** Colors and fonts are hardcoded in the MoviePage.java file, which could cause inconsistencies in UI design.**Steps to Reproduce:**

1. Open the movie details page.
2. Compare the UI with other pages. **Expected Result:**

* A centralized theme manager should be used to maintain UI consistency.

**Bug 4: Back Button Navigation Issues**

**Description:** The back button implementation relies on an external Runnable, which could lead to unexpected behavior.**Steps to Reproduce:**

1. Open multiple movie pages in sequence.
2. Click the "Back" button multiple times quickly. **Expected Result:**

* The back navigation should follow a predictable behavior and not cause crashes.

**Code Review Issues**

**1. Hardcoded UI Styles**

* Colors and fonts are hardcoded (e.g., new Color(30, 32, 34), "Segoe UI Semibold").
* Consider using a **centralized theme manager** to maintain consistency.

**2. No Error Handling for Missing Movie Data**

* If movie.getTitle() or movie.getReleaseDate() is null, the UI may break.
* Should include **null checks** before displaying data.

**3. Back Button Logic Could Be Improved**

* The **onBack.run()** mechanism depends on an external function.
* A better approach might be to use a **dedicated back-navigation method** to handle state more cleanly.

**Code Smells Detected**

**1. Long Method**

* The MoviePage constructor handles multiple UI setup tasks in a single method.
* Refactor by extracting UI initialization logic into separate methods for better readability and maintainability.

**2. Duplicate Code**

* UI component styling is repeated across the class.
* Create a helper function to apply consistent styles to components.

**3. Inappropriate Intimacy**

* MoviePage directly accesses properties of Movie without null checks.
* Implement getter methods with validation to ensure safe access to data.

**4. Feature Envy**

* Some UI components manipulate movie data directly rather than delegating this to the model layer.
* Refactor logic so that business logic remains in the model rather than the view.