EECS2311 - Lab 03 - Team 7
Take Home Assignment
Name: Gjergj Kroqi
Date: March 20, 2025
User Story Tested: Genre Filtering Feature

Reported by: Gjergj Kroqi
Program/Component: MovieMainMenu.java
Version: v1.0
Configuration: Windows 11, Java 17, SQLite, Eclipse
Can Reproduce: Yes

End-to-End Testing

Test Case 1: Filter by Action Genre

Steps to Reproduce:

1. Launch the Movie Catalog application.
2. Log in (if required).
3. Click the “Genres” button.
4. Click the “Action” genre.

Expected Result:
Only movies that have “Action” in their genres are displayed in the grid.
Actual Result:
All movies that have “Action” in their genres were correctly displayed.
Pass/Fail: Pass

Test Case 2: Filter by Comedy Genre

Steps to Reproduce:

1. Launch the Movie Catalog application.
2. Click the “Genres” button.
3. Click the “Comedy” genre.

Expected Result:
Only movies that have “Comedy” in their genres are displayed in the grid.
Actual Result:
All movies that have “Comedy” in their genres were correctly displayed.
Pass/Fail: Pass

Test Case 3: Filter by Drama Genre

Steps to Reproduce:

1. Launch the Movie Catalog application.
2. Click the “Genres” button.
3. Click the “Drama” genre.

Expected Result:
Only movies that have “Drama” in their genres are displayed in the grid.
Actual Result:
All movies that have “Drama” in their genres were correctly displayed.
Pass/Fail: Pass

Test Case 4: Filter by Science Fiction Genre

Steps to Reproduce:

1. Launch the Movie Catalog application.
2. Click the “Genres” button.
3. Click the “Science Fiction” genre.

Expected Result:
Only movies that have “Science Fiction” in their genres are displayed in the grid.
Actual Result:
All movies that have “Science Fiction” in their genres were correctly displayed.
Pass/Fail: Pass

Test Case 5: Filter by Fantasy Genre

Steps to Reproduce:

1. Launch the Movie Catalog application.
2. Click the “Genres” button.
3. Click the “Fantasy” genre.

Expected Result:
Only movies that have “Fantasy” in their genres are displayed in the grid.
Actual Result:
All movies that have “Fantasy” in their genres were correctly displayed.
Pass/Fail: Pass

Test Case 6: Filter by Mystery Genre

Steps to Reproduce:

1. Launch the Movie Catalog application.

2. Click the “Genres” button.

3. Click the “Mystery” genre.

Expected Result:
Only movies that have “Mystery” in their genres are displayed in the grid.
Actual Result:
All movies that have “Mystery” in their genres were correctly displayed.
Pass/Fail: Pass

Code Review:

File: MovieMainMenu.java
Reviewed by: Gjergj Kroqi
Date: March 22, 2025
Version: v1.0 (Sprint 2)
Component: View Layer (GUI)
Type: Design and Maintainability Review

1. Long Method – `showMainMenu()`

- **Type:** Long Method
- **Location:** `showMainMenu()`
- **Smell Description:** This method is over 100 lines long and performs many responsibilities
- **Impact:** Low readability, hard to test, violates Single Responsibility Principle (SRP).
- **Suggested Refactor:** Decompose into smaller methods:
    - `topBar()`
    - `filterPanel()`
    - `scrollPaneWithGrid()`

2. Hardcoded Data – Genre List

- **Type:** Hardcoded Value
- **Location:** `setupSidePanel()`

Smell Description: Genre list is manually defined as:

```
String[] genres = {"Action", "Comedy", "Drama", "Science Fiction",
"Fantasy", "Mystery"};
```

- **Impact:** Difficult to update genres without modifying code. Not scalable.

- **Suggested Refactor:** Load genres dynamically from the database using `SELECT DISTINCT` or a `Movie.getAllGenres()` method.

3. Feature Envy / Logic in UI Layer

- **Type:** Feature Envy
- **Location:** Genre button listeners

Smell Description: Filtering logic is embedded inside the GUI layer:

```
movie.getGenres().contains(genre)
```

- **Impact:** Violates separation of concerns. UI and business logic are tightly coupled.

Suggested Refactor: Move filtering logic to a service or controller class:

```
MovieService.getMoviesByGenre(String genre);
```

4. Lack of Null Safety – `getGenres()`

- **Type:** Missing Null Check
- **Location:** Genre filter and display methods
- **Smell Description:** Code assumes `movie.getGenres()` is never null, which can lead to `NullPointerException`.
  Impact: Runtime crash risk if data is corrupt or missing.

Suggested Refactor: Safeguard with:

```
if (movie.getGenres() != null && movie.getGenres().contains(genre))
```

5. Duplicated UI Styling – Genre Buttons and Colors

- **Type:** Duplicated Code / Magic Numbers
- **Location:** Genre buttons and UI styling code
- **Smell Description:** Colors, button dimensions, and fonts are repeated throughout the codebase.
- **Impact:** Higher maintenance cost, inconsistency risks.
- **Suggested Refactor:** Extract common styles into constants or a helper UI utility class.

6. God Class Tendencies – `MovieMainMenu`

- **Type:** God Class
- **Location:** Entire Class
- **Smell Description:** `MovieMainMenu` handles:
  - UI layout
  - Event handling
  - Business logic
  - DB-interacting model logic indirectly
- **Impact:** Violates SRP, makes testing and extension difficult.
- **Suggested Refactor:** Extract:
  - A `MovieFilterService`
  - A `UIBuilder` or `MoviePanelFactory`
  - Possibly separate `GenrePanel` component