Take home assignment:

Due Monday March 24: 11:59 pm

Note: There will be no tutorial on March 24. You can use the lab time to wrap up your assignment delivery.

In this assignment you will be doing a project-related activity. The activity consists of two QA tasks: end-2-end manual testing and code review.

Activity 1 steps:

1. As a team, in a document (called EECS2311-SectionName-Team#-TakeHomeAssignment.pdf) list the fully implemented and unit/integration tested user stories in your project up to now that have GUI as well.

2. In the document, explicitly identify who are the main developers of each user story.

3. Evenly distribute all stories to team members such that nobody is assigned to a user story that they are heavily involved in its implementation.

4. Write down who is now responsible to test which stories and place this document in your team's GitHub.

5. Next, each student, independently, make a copy of the document and name it EECS2311-SectionName-Team#-YourFullName-TakeHomeAssignment.pdf. In this document, under each assigned story to you, write down several end-2-end manual test cases (equivalent to customer tests) to validate the assigned user stories.

    a. An end-2-end test is a step-by-step precise instruction to follow a scenario that is described by the user story.
    b. Make sure your tests are precise enough to be replicable.
    c. The tests are derived from the user stories from the perspective of customers. Think of different ways they could likely cover the story.
    d. You need multiple tests per user story so that not only the main scenario is tested but also corner cases.
    e. Each test is end to end. That is, it uses the GUI, the business layer, as well as all external libraries (if any) and the DB.

6. Continue with writing test and running them until you have a good coverage of the scenarios related to your assigned stories.

7. When you find bugs and issues, report them following the format explained in the lecture on bug reports.

8. Report the bugs both on the team's Github as well as the TakeHomeAssignment Document.

Activity 2: The second activity is code review.

This time you are supposed to read the code for the assigned stories. Again, report issues in the Github following the bug report format and also write them down in the document.

a. You can report both bugs and code/design smells, in this step.
b. Check out the list of code smells here for more information:
i. http://sourcemaking.com/refactoring
ii. https://refactoring.guru/refactoring/smells
iii. https://pragmaticways.com/31-code-smells-you-must-know/

9. Beside reporting the bugs and issues on the GitHub, write them down in the TakeHomeAssignment document as well. In the document explain what code smells you checked and which ones you detected.

Due:
This activity starts today and ends on March 24 (must be submitted in eClass and GitHub by 23:59).

Deliverable:

Each student submits their "TakeHomeAssignment" document, Individually, on eClass and reports the bugs and issues on the team's GitHub. The document includes all bugs and issues as they are reported plus the actual test cases and the code smells they checked and those they detected.

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

Deliverable

Part 1: Test Cases

Test Case 1: Successful Login

Steps:

1. Open the login page.
2. Enter a valid username and password.
3. Click the "Log In" button.

Expected Result:

A success message appears.

Test Case 2: Invalid Login (Wrong Password)

Steps:

1. Open the login page.
2. Enter a valid username but an incorrect password.
3. Click "Log In".

Expected Result:

A message box appears saying "Invalid credentials!". The login screen also remains open.

Test Case 3: Invalid Login (Non-Existent User)

Steps:

1. Open the login page.
2. Enter a username that doesn’t exist in the system.
3. Enter any password.
4. Click "Log In".

Expected Result:

A message box appears saying "Invalid credentials!".

Test Case 4: Empty Username or Password

Steps:

1. Open the login page.
2. Leave the username and/or password field empty.
3. Click "Log In".

Expected Result:

A message appears prompting the user to fill in both fields.

Test Case 5: Register Button Functionality

Steps:

1. Open the login page.
2. Click the "Don't have an account? Register" button.

Expected Result:

The current window closes and the RegisterPage window opens.

Part 2: Bug Reports

Bug #1:

Bug ID: BUG-001

Reported by: Suneel Denneny

Date Reported: March 23, 2025

Program Component: LoginPage.java

Version: 1

Environment: Windows 11, Java 17, Eclipse 2024-03

Bug Type: Coding Error

Can Reproduce? Yes

Severity: Medium

Priority: Medium

Problem Summary: Login does not have unique message for empty password

Keywords: Login, password, validation, UI bug

Problem Description:

1. Open the Login Page.

2. Leave both username and password fields empty.

3. Click "Log In".

4. Expected Result: The system should display an error message: "Username and password cannot be empty!"

5. Actual Result: The system outputs default message

Suggested Fix: Add a separate error message and exception for empty username/password sections

Status: Open

Resolution: Pending

Bug #2:

Bug ID: BUG-002

Reported by: Suneel

Date Reported: March 23, 2025

Program Component: LoginPage.java

Version: 1

Environment: Windows 11, Java 17, Eclipse 2024-03

Bug Type: Security Issue

Can Reproduce? Yes

Severity: Critical

Priority: High

Problem Summary: Password is stored as a String instead of char[]

Keywords: Security, password, memory leak

Problem Description:

1. The application retrieves the password using `new String(passwordField.getPassword())`.

2. Strings remain in memory longer and cannot be securely wiped.

3. Expected Result: The password should be handled as `char[]` and wiped from memory after use.

4. Actual Result: Passwords remain in memory as Strings.

Suggested Fix: Use a `char[]` to store passwords and overwrite it after authentication.

Status: Open

Resolution: Pending

Bug #3:

Bug ID: BUG-003

Reported by: Suneel Denneny

Date Reported: March 23, 2025

Program Component: LoginPage.java

Version: 1

Environment: Windows 11, Java 17, Eclipse 2024-03

Bug Type: UI Bug

Can Reproduce? Yes

Severity: Low

Priority: Low

Problem Summary: Register button does not set `RegisterPage` as visible

Keywords: UI, navigation, register button

Problem Description:

1. Open the Login Page.

2. Click the "Register" button.

3. Expected Result: The `RegisterPage` window is not very clear to the user

4. Actual Result: The window is instantiated but looks nearly identical to login page

Suggested Fix: Update the action listener to include `registerPage.setVisible(true);`.

Status: Open

Resolution: Pending

Part 3: Code Smells

Issue ID: SMELL-001

Reported by: Suneel Denneny

Date Reported: March 23, 2025

Program Component: LoginPage.java

Type of Code Smell: Tight Coupling

Problem Summary: `LoginPage` directly calls `User.loginUser()`, making the UI dependent on the backend logic.

Description of the Issue:

- The `LoginPage` calls `User.loginUser(username, password)` and `new MovieMainMenu(username)`, making UI changes dependent on authentication logic.

- If the authentication method changes (e.g., switching from `.dat` files to a database), the UI needs to be rewritten.

- Should make UI function independently

Proposed Fix:

- Use a separate `AuthController` class to handle authentication logic.

Status: Open

Issue ID: SMELL-002
Reported by: Suneel Denneny
Date Reported: March 23, 2025
Program Component: LoginPage.java
Type of Code Smell: Long Method
Problem Summary: handleLogin() performs multiple actions, making it hard to read.
Description of the Issue:

- Handle login is responsible for many functions such as Retrieving user input, validating input etc.
- This makes it much harder to debug

Proposed Fix:

- Break up handleLogin() into smaller methods

Status: Open

Issue ID: SMELL-003
Reported by: Suneel Denneny
Date Reported: March 23, 2025
Program Component: LoginPage.java
Type of Code Smell: Magic Numbers & Hardcoded UI Properties
Problem Summary: Colors, fonts, and sizes are hardcoded, which will make changing them in the future extremely difficult

Description of the Issue:

- UI properties such as colors and fonts are hardcoded throughout the class, making global style changes difficult.

Proposed Fix:

- Make constants in a separate class

Status: Open