HOLLA! The private Messenger

Software Engineering

Group X

Gagana kusuma K

Rakesh Kumar Sah

Introduction to the Project

“HOLLA!” Private Messenger Application is a communication tool designed for secure and private messaging between users. It is intended to facilitate seamless communication within a closed group or team environment, ensuring confidentiality and privacy of conversations.

Test Cases Overview:

The following test cases have been implemented to verify the functionality of the LoginBackend.authentication method:

1. **Successful Authentication**
    - Validates successful login with correct credentials.
2. **Failed Authentication Due to Incorrect Password**
    - Verifies that authentication fails when an incorrect password is provided.
3. **Failed Authentication Due to Non-existent User**
    - Ensures authentication fails when attempting to authenticate with a non-existent username.
4. **SQLException Handling**
    - Tests the handling of SQL exceptions when an invalid database URL is provided.

Fixes Implemented

- **Resolved NullPointerException**: Handled null parameters in the authentication method to prevent crashes.
- **SQLException Handling**: Enhanced database connection management to handle SQLExceptions properly.
- **Test Data Corrections**: Fixed SQL statements in test data insertion methods to match schema requirements, preventing SQL integrity constraint violations.

Results

All test cases have been executed successfully after implementing the fixes. The authentication method now operates as expected under various scenarios, ensuring robust performance and error handling.

Objective: Verify the functionality and UI components of the AfterLogin class

Test Environment:

- Java 11
- JUnit 5
- Swing (for UI components)

Test Setup:

1. **AfterLogin Class:**
    - Represents a JFrame that displays user data and provides interaction with various UI components such as buttons and text fields.
    - Includes methods for fetching usernames from a database, performing searches, opening group message prompts, and editing profiles.
2. **Test Class: AfterLoginTest**
    - Contains JUnit test methods to validate the behavior and properties of UI components within the AfterLogin frame.
    - Utilizes reflection to access private fields and methods of the AfterLogin class for testing purposes.

Test Cases:

1. **Buttons Creation**
    - **Objective:** Ensure that buttons (usernameButton, groupMessageButton) are created and displayed correctly.
    - **Steps:**
        - Create an instance of AfterLogin for testing.
        - Use the findComponent method to locate buttons by their expected names.
        - Assert that buttons are not null and verify their text content matches the expected values.
2. **Search Panel Components**
    - **Objective:** Verify the presence and functionality of search-related components (searchField and searchButton) in the AfterLogin frame.
    - **Steps:**
        - Retrieve the content pane of the AfterLogin instance.
        - Locate the search components using the findComponent method.
        - Assert that the search field and button are not null and validate their properties (e.g., text content).

Utility Method: findComponent(Container container, Class<?> clazz, String name)

- **Purpose:** Recursively searches for a component by its class and name within a given container and its children.

- **Usage:** Utilized in test methods to locate specific UI components within the AfterLogin frame for validation.

Conclusion: The test cases ensure that the AfterLogin class functions correctly by validating the creation, properties, and functionality of its UI components. By employing reflection and utility methods like findComponent, the tests verify that the application behaves as expected, providing a robust user interface for interacting with user data.

Test Case Summary

1. **testUIComponents**
    - **Purpose:** Verify the existence and initialization of UI components (messagebox and chatscreen).
    - **Expected Outcome:** Assert that the UI components are not null and initialized correctly. Also, ensure the correct title of the chat window.
2. **testSendMessage**
    - **Purpose:** Test the functionality of sending a message.
    - **Steps:** Set text in the messagebox, click the "Send" button, and verify that the messagebox is cleared after sending.
    - **Verification:** Assert that the messagebox is empty after sending a message.
3. **testSendEmoji**
    - **Purpose:** Verify the behavior of the emoji button.
    - **Steps:** Click the emoji button, verify that the emoji appears in the messagebox.
    - **Verification:** Assert that the emoji text appears correctly in the messagebox.
4. **testRefreshMessages**
    - **Purpose:** Validate the refreshing of messages from the backend.
    - **Steps:** Set backend messages, invoke the refresh method, and verify that chatscreen displays the updated messages.
    - **Verification:** Assert that the chatscreen contains the expected messages after refreshing.

Results

All test cases have been executed successfully after implementing the fixes. The authentication method now operates as expected under various scenarios, ensuring robust performance and error handling.