SOFTWARE TEST AND REFACTOR

HOLLA

Holla is a private messenger application 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.

The initial goals and objectives as outlined in the project assignment are;

1. Ensure secure and private messaging between users.
2. Enable users to create accounts and log in securely, with passwords stored securely in the database.
3. Implement a user-friendly messaging interface for sending and receiving text messages in real-time
4. Store messages and user data securely and persistently in a database, ensuring data integrity and availability.

Testing Approach

A. Black-box testing

A software testing method where the functionality of an application is evaluated without looking at the internal code structure, implementation details. It solely focuses on the inputs and outputs of the software ensuring that it works as expected based on its requirements and specifications. This testing plays a critical role in ensuring the quality, usability, and reliability of software applications.

TEST CASE SUMMARY

Six out of the 15 test cases exhibited failure. The failed ones are 2, 3, 4, 6, 10 and 12.

1. **Functional**

These test cases focus on verifying that specific functionality of the application work as intended.

- Test Case 1: Verify that the signup form is displayed correctly.
- Test Case 5: Verify that the login form is displayed correctly.
- Test Case 7: Verify that the user is redirected to the correct page after a successful login.
- Test Case 8: Verify that the chat interface is displayed correctly.
- Test Case 9: Verify that a message can be sent successfully.
- Test Case 11: Verify that the search results window is displayed correctly.
- Test Case 13: Verify that the edit profile window is displayed correctly.
- Test Case 14: Verify that the Group message works correctly.
- Test Case 15: Verify that the logout works correctly

2. **Validation**

   These test cases verify that the application handles incorrect or invalid inputs properly and displays appropriate error messages.

- Test Case 2: Ensure that incorrect signup credentials display an appropriate error message.

- Test Case 3: Verify that an error is returned when trying to register with an existing username.
- Test Case 4: Verify that an error is returned when trying to register with empty username or password fields.
- Test Case 6: Verify that an error message is shown for empty username or password fields.
- Test Case 10: Verify that sending an empty message displays an appropriate error

3. **Usability**

   These test cases ensure that the user interface elements are displayed correctly and are user-friendly.

   - Test Case 1: Verify that the signup form is displayed correctly.
   - Test Case 5: Verify that the login form is displayed correctly.
   - Test Case 8: Verify that the chat interface is displayed correctly.
   - Test Case 11: Verify that the search results window is displayed correctly.
   - Test Case 13: Verify that the edit profile window is displayed correctly.

4. **Navigation**

   These test cases ensure that users are directed to the correct pages or states within the application based on their actions.

   - Test Case 7: Verify that the user is redirected to the correct page after a successful login.

- Test Case 15: Verify that the logout works correctly

5. **Layout and Responsiveness**

   These test cases verify that the application's layout adjusts correctly to different screen sizes and window resizing actions.

   - Test Case 12: Verify that resizing the application window adjusts the layout correctly.

B. White-box testing

A method of testing software that involves looking inside the software's internal structure, design, and implementation. White box testing requires knowledge of the internal code and architecture. Testers use this information to design test cases that ensure every possible path through the code is executed and tested.

TEST CASE SUMMARY

Seven out of the thirty test cases failed. The failed cases are 6, 7, 14, 15, 17, 24 and 26.

1. **UI Component Initialization Tests**

   - Test Case 1: Verify UI components initialization (‘AppGUI’ instance)
   - Test Case 4: Verify UI components initialization (‘ChatFrontend’ instance)
   - Test Case 11: Verify UI components initialization (‘LoginFrontend’ instance

- Test Case 21: Verify UI components initialization (‘SignupFrontend’ instance)
- Test Case 27: Verify password field initialization

2. **Action Tests**

    - Test Case 2: Test login button action
    - Test Case 3: Test signup button action
    - Test Case 5: Test sending a message
    - Test Case 6: Test sending an emoji
    - Test Case 22: Verify ‘actionPerformed’ method with valid input

3. **Input Handling Tests**

    - Test Case 12: Valid login credentials
    - Test Case 13: Invalid login credentials

4. **Functional Tests**

    - Test Case 9: Verify fetching messages
    - Test Case 10: Verify searching messages

5. **Error Handling Tests**

    - Test Case 14: Verify successful authentication with correct credentials
    - Test Case 15: Verify failed authentication due to incorrect password
    - Test Case 17: Verify SQL exception handling

6. **Authentication and Authorization Tests**

- Test Case 14: Verify successful authentication with correct credentials
- Test Case 15: Verify failed authentication due to incorrect password

7. **Database Interaction Tests**

- Test Case 8: Verify message storing
- Test Case 9: Verify fetching messages
- Test Case 16: Verify clearing of USERS1 table before each test

8. **Behavioural Test**

- Test Case 7: Test refreshing messages

9. **Window and Layout Test**

- Test Case 18: Verify window properties
- Test Case 19: Verify search results display in the UI
- Test Case 20: Verify initial visibility of the window

10. **Initialization and Setup Tests**

- Test Case 16: Verify clearing of USERS1 table before each test