Test Execution Summary Report

This report summarizes the results of automated tests conducted on the **Hiring Hotspot** application using Selenium with Python and Pytest. The tests cover key functionalities: logging in, sending a message in the chat, and viewing a contract. The report also highlights issues encountered during testing and suggests improvements for enhancing the test suite.

---

1. Test Results

- **Total Tests:** 3
- **Passed:** 3
- **Failed:** 0
- **Execution Time:** 35 seconds

---

2. Test Details

- **Login Test:**
    - **Description:** Successfully logged in with the email `k213916@nu.edu.pk` and navigated to the dashboard.
    - **Assertion:** Verified that the URL no longer contains "login".
    - **Screenshot:** `login_success.png`
- **Send Message Test:**
    - **Description:** Searched for the user "Hassan" in the chat, selected "Hassan Haneef", and sent the message "Hello from Selenium!".
    - **Assertion:** Verified that the message "Hello from Selenium!" appears on the screen.
    - **Screenshots:** `chat_search.png` (after searching), `message_sent.png` (after sending the message)
- **View Contract Test:**
    - **Description:** Navigated to the contract page at `/contracts/14` and verified the presence of a contract title.
    - **Assertion:** Verified that the contract title is non-empty.
    - **Screenshot:** `contract_viewed.png`

---

3. Issues Found

During the test execution, the following issues were identified:

- **Flaky Tests Due to `time.sleep()`:**
    - The tests rely on `time.sleep()` for waiting, which can lead to inconsistent results. The timing might not always align with the application's state, causing tests to fail intermittently.
    - **Affected Tests:** `test_send_message` (waiting for search results and message appearance).
- **Limited Assertions:**
    - The assertions in the tests are minimal and may not fully validate the application’s state.
    - **Login Test:** Only checks that "login" is not in the URL but does not confirm the presence of specific dashboard elements.
    - **View Contract Test:** Only checks that the contract title is non-empty but does not verify the actual content or other elements on the page.
- **Hardcoded Values:**
    - The tests use hardcoded values (e.g., email, password, user search term, contract ID), which limits their flexibility and reusability.
    - **Affected Tests:** All tests.

---

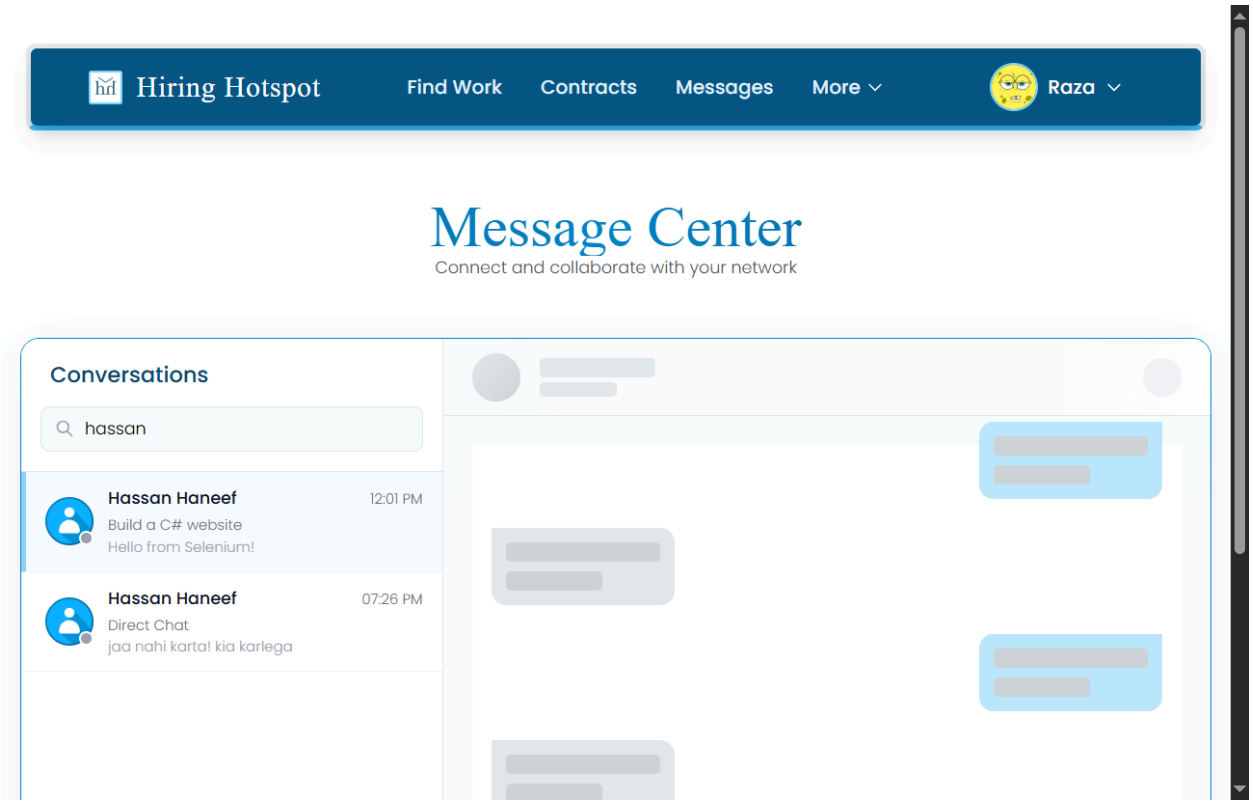
4. Possible Improvements

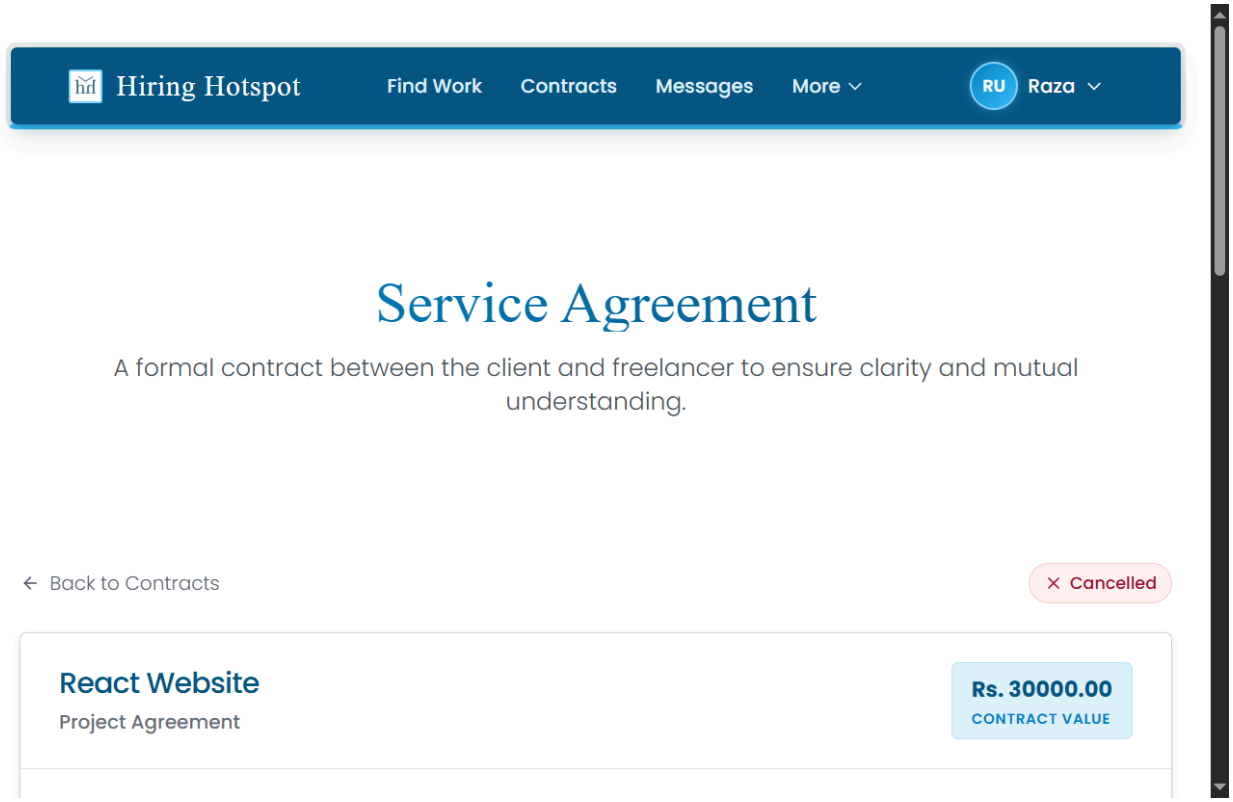
To enhance the reliability, maintainability, and coverage of the test suite, the following improvements are recommended:

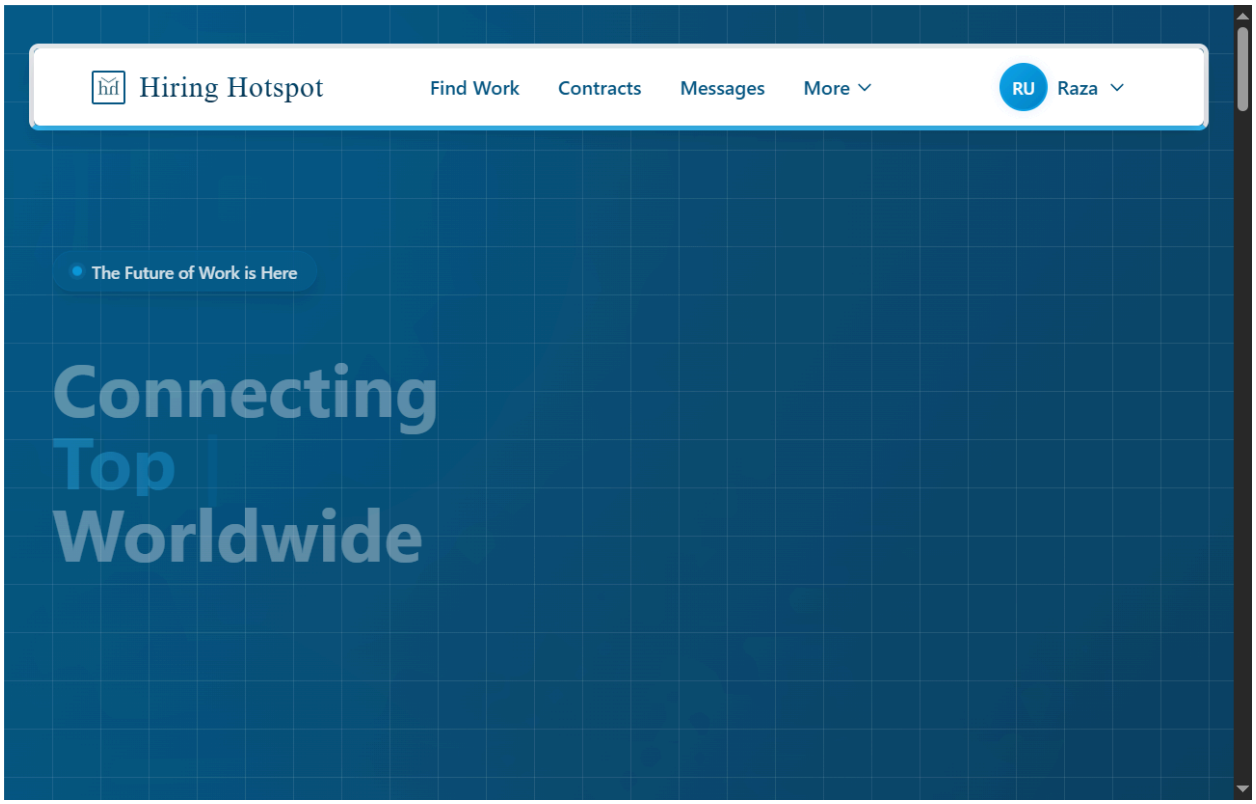
- **Replace `time.sleep()` with Explicit Waits:**
    - Use Selenium's `WebDriverWait` with conditions (e.g., `EC.visibility_of_element_located`) to wait for elements dynamically, reducing flakiness.
    - **Example:** Replace `time.sleep(2)` in `test_send_message` with a wait for the user result to be clickable.
- **Enhance Assertions:**
    - Add more comprehensive assertions to ensure the application is in the expected state.
    - **Login Test:** Verify the presence of a specific dashboard element (e.g., a welcome message or user profile).
    - **View Contract Test:** Check for specific contract details (e.g., contract ID, status) in addition to the title.
- **Use More Specific Locators:**
    - Ensure locators are unique and less prone to change. Consider using `data-testid` attributes in the React frontend for stable test identifiers.
    - **Example:** If the application uses `data-testid="contract-title"`, update the locator to `By.CSS_SELECTOR, "[data-testid='contract-title']"`.
- **Parameterize Tests:**
    - Use Pytest's parameterization to test multiple scenarios (e.g., different users, messages, or contract IDs).
    - **Example:** Test sending messages to different users or viewing multiple contracts.
- **Error Handling and Logging:**
    - Add try-except blocks to capture screenshots on failure and log detailed error messages.
    - **Example:** Capture a screenshot if an assertion fails and log the exception for debugging.
- **Test Data Management:**
    - Create test data programmatically (e.g., via API calls) before running tests to ensure the required users and contracts exist.
    - **Example:** Use the application's API to create a test user and contract before executing the tests.

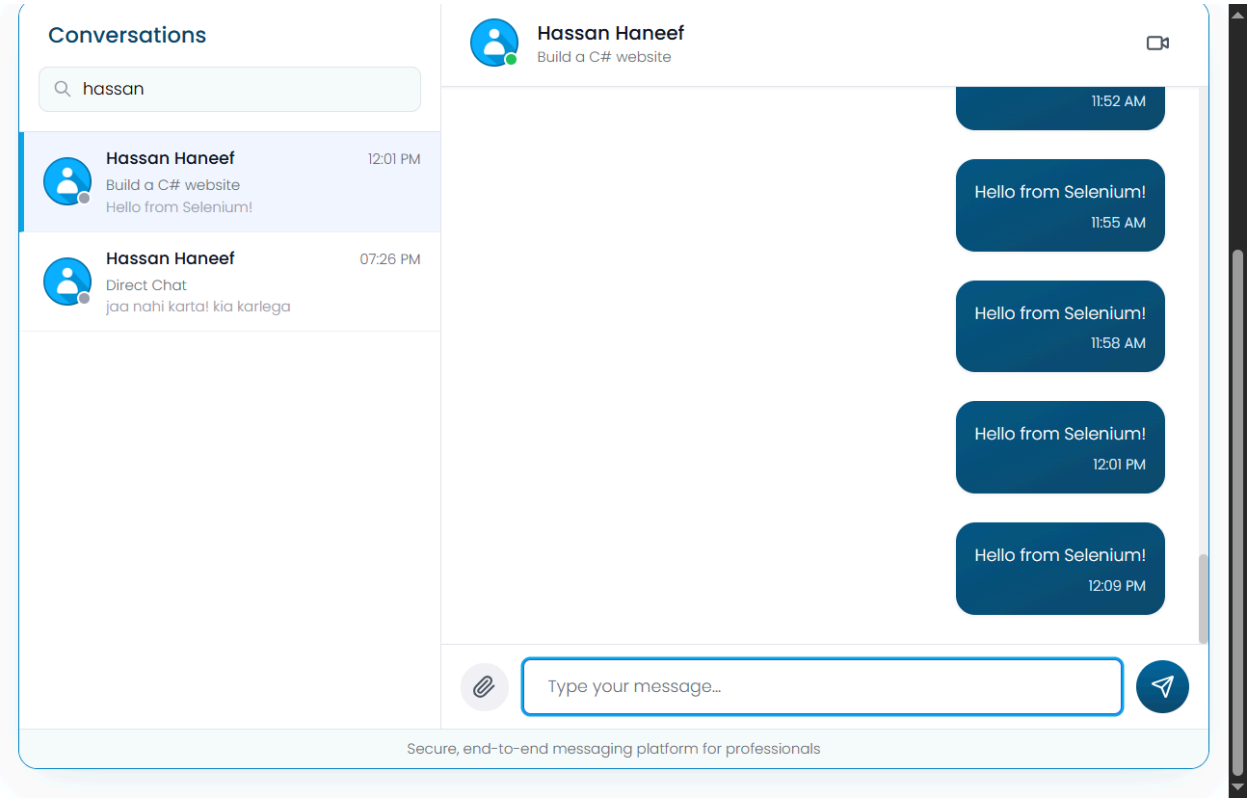
5. Attachments

- **Test Report:** `report.html` (generated by Pytest)
- **Screenshots:**




Hiring Hotspot
Find Work
Contracts
Messages
More
RU
Raza
Service Agreement
A formal contract between the client and freelancer to ensure clarity and mutual understanding.
Back to Contracts
Cancelled
React Website
Project Agreement
Rs. 30000.00
CONTRACT VALUE


Hiring Hotspot
Find Work
Contracts
Messages
More
RU
Raza
The Future of Work is Here
Connecting
Top
Worldwide


Conversations
hassan
Hassan Haneef
12:01 PM
Build a C# website
Hello from Selenium!
Hassan Haneef
07:26 PM
Direct Chat
jaa nahi karta! kia karlega
Hassan Haneef
Build a C# website
11:52 AM
Hello from Selenium!
11:55 AM
Hello from Selenium!
11:58 AM
Hello from Selenium!
12:01 PM
Hello from Selenium!
12:09 PM
Type your message...
Secure, end-to-end messaging platform for professionals