Test Plan for Basic Functionality of ChatGPT

1. Introduction

Based on my expertise, the end-to-end tests provided in Problem 1 do not adhere to standard testing principles. The four distinct features should be tested individually, each requiring a tailored testing strategy.

For example, if the system under test is the login mechanism, tests should focus solely on the login functionality. Large end-to-end tests with complex call flows often lead to flakiness and can obscure the identification of bugs. Each test should be independent and capable of being executed multiple times without issues.

If the goal is to validate the context of responses, API testing would be more appropriate, allowing for more detailed context checks. Additionally, comparing responses with those from a different AI API could verify the model's accuracy, as static validation can lead to the pesticide paradox.

With this in mind, I have developed a test plan to effectively test these four distinct functionalities.

2. Test Objectives

1. Verify login functionality
2. Validate that users can log out successfully.
3. Ensure that text-based exchanges function correctly.
4. Confirm that image uploads and corresponding descriptions are handled correctly.

3. Test Scope

- Functional testing of the ChatGPT interface focusing on
    - Login
    - logout
    - text messaging
    - image upload functionalities.

- For the scope of this document we will focus on functional testing and not test any non functional aspects as performance or security.

4. Test Environment

- Browser: Latest version of Chrome/Firefox.
- Automation Tool: Python Playwright.
- System: Any OS with Python and Playwright installed.
- Network: Stable internet connection.
- Repository management will be handled by git

5 Test Cases

5.1 Verify login functionality

Test Case 1: Login Functionality

- **Objective**: Verify that a user can log in successfully.
- **Steps**:
    1. Navigate to the ChatGPT login page.
    2. Enter valid credentials (username and password).
    3. Click the login button.
    4. Verify successful login by checking for a specific element that appears only after logging in.
- **Expected Result**: User is successfully logged in and redirected to the main interface.
- **Iterations**
    1. Login with email address
    2. Login through Google Authentication
    3. Login through Microsoft Authentication
    4. Login through Apple Authentication

Test Case 2 Login Functionality Failing Scenario

- **Objective**: Verify that an invalid user cannot log.
- **Steps**:
    1. Navigate to the ChatGPT login page.
    2. Enter invalid credentials (username and password).
    3. Click the login button.
    4. Verify that the user cannot login
    5. **Expected Result**: User cannot log in and get a verbose error message.
- **Iterations**
    1. Login with an unauthorized email address
    2. Login with an invalid email address
    3. Try sql injections

5.2 Validate that users can log out successfully.

Test Case 1: Logout Functionality

- **Objective**: Verify that a user can log out successfully.
- **Steps**:
    1. Log in to ChatGPT.
    2. Click the logout button.
    3. Verify that the user is redirected to the login page.
- **Expected Result**: User is successfully logged out.
- **Iterations**
    1. Login with email address
    2. Login through Google Authentication
    3. Login through Microsoft Authentication
    4. Login through Apple Authentication

Test Case 2: Logout Functionality

- **Objective**: Verify that a user is logged out even after refresh
- **Steps**:
    1. Log in to ChatGPT.
    2. Click the logout button.
    3. Make refresh to page
    4. Verify that the user is redirected to the login page.
- **Expected Result**: User is successfully logged out.

5.3 Ensure that text-based exchanges function correctly.

Test Case 1 Text-based Exchange

- **Objective**: Ensure that text messages can be sent and responses are received correctly.
- **Steps**:
    1. Log in to ChatGPT.
    2. Send a text message (e.g., "Hello, ChatGPT!").
    3. Verify that a response is received.
    4. Verify also the context of response
- **Expected Result**: ChatGPT responds to both text messages appropriately.
- **Iterations**
    1. Send a valid text in order to check the response is correct
    2. Send random text in order to verify that chat gpt replies accordingly
    3. Send a message in different language
    4. Try Sql injection
    5. Try at maximum characters limit
    6. Special Characters Handling
    7. Multi line text
    8. Emojis

Test Case 2 Text-based Exchange failing scenario

- **Objective**: Ensure that text messages can be sent and responses are received correctly.
- **Steps**:
    1. Log in to ChatGPT.
    2. Send a text message
    3. Verify that a response is received.
    4. Verify an error response is sent
- **Expected Result**: ChatGPT responds to both text messages appropriately.
- **Iterations**
    1. Try Sql injection
    2. Try above maximum characters
    3. Try empty text.

Test Case 3 Text-based Exchange multiple messages

- **Objective**: Ensure that text messages can be sent and responses are received correctly.
- **Steps**:
    1. Log in to ChatGPT.
    2. Send a text message (e.g., "Hello, ChatGPT!").
    3. Verify that a response is received.
    4. Send a second text message (e.g., "Hello, ChatGPT again!").
    5. Verify also the context of response
- **Expected Result**: ChatGPT responds to both text messages appropriately.

Test Case 4 Text-based Exchange multiple messages simultaneously handling

- **Objective**: Ensure that text messages can be sent and responses are received correctly.
- **Steps**:
    1. Log in to ChatGPT.
    2. Send a text message (e.g., "Hello, ChatGPT!").
    3. Try to send simultaneously a second message
    4. Verify that the page handles this
- **Expected Result**: ChatGPT responds to both text messages appropriately.

5.4 Confirm that image uploads and corresponding descriptions are handled correctly.

Test Case 1: Image Upload and Description

- **Objective**: Validate the image upload functionality and the description provided by ChatGPT.
- **Steps**:
    1. Log in to ChatGPT.
    2. Upload an image.
    3. Ask ChatGPT to describe the image.
    4. Verify the description provided by ChatGPT.
- **Expected Result**: Image is uploaded successfully, and ChatGPT provides an accurate description.
- **Iterations**
    1. Upload a file from pc
    2. Upload a file from google drive
    3. Upload a file from Microsoft One drive
    4. Upload various file formats.
    5. Upload maximum allowed file 20 MB

Test Case 2: Image Upload and Description failing scenario

- **Objective**: Validate the image upload functionality and the description provided by ChatGPT.
- **Steps**:
    1. Log in to ChatGPT.
    2. Upload an image.
    3. Ask ChatGPT to describe the image.
    4. Verify an error message appears
- **Expected Result**: Image is not uploaded successfully, and ChatGPT provides an accurate description.
- **Iterations**
    1. Upload a file of restricted format
    2. Upload maximum allowed file more than 20 MB
    3. Upload image and also use above maximum characters limit

6. Risks

- Network issues affecting login and message exchange.
- High performance impact on Chat Gpt due to extreme load
- Changes in the ChatGPT interface affecting element locators.
- File upload restrictions or issues with specific image formats.
- Cloudflare security check for bots can provide
- Limitation on using the functionality can lead to flakiness

7. Challenges

- My first challenge is to work on python and playwright since it is not the language or the framework I work on my job. I had forgotten how many things can go wrong with virtual environments and imports.
- My Second challenge was the site itself. It had a lot of performance impact. There were elements that needed multiple times to be clicked in order to react and had chaotic behavior
- Cloudflare check also proved to be a problem as it needed to reduce the speed in order not to be recognised as bot.
- I had a limit of tries to upload a file and the mechanism as described on the internet and official documentation did not seem to work losing 2 days to make this happen.
- Playwright Although it has some more clean ways to check locators I found it a bit more difficult to use page object model and do customizations. Also Setup and teardown needed to be on conftest.py file which felt a bit restricted.
- When you get used in a compiled language interpreter languages look a bit more fluid concerning object types.
- The locators where kind of challenging testwise

8. Summary

Problem 1

For Problem 1, I developed a basic solution using Python and Playwright, adhering to the Page Object Model (POM) to enhance maintainability. I implemented the two scenarios as outlined in your instructions, creating a foundational structure for the test suite.

To ensure robustness, I integrated a logger and enabled video recording for test failures. Although I considered adding screenshot capture on failure, this feature is not included in the current implementation. I utilized hooks for test setup and teardown to facilitate better control and code reusability. Additionally, I incorporated wrappers for actions and assertions to provide enhanced control and increased logging, particularly useful for headless execution.

My primary focus was on creating modular, scalable, and reusable code blocks. For data management, I avoided hardcoded values by using a structured approach, although I acknowledge there is room for improvement in handling static data within tests.

Last but not least i had a huge issue with the playwright upload mechanism. I managed to do it with javascript on my last attempt but could not finalize my test because of the chat gpt daily limit.

Lastly, I chose not to employ any encryption methods for handling credentials. Instead, I opted to use environment variables. This approach allows any user to set credentials on their local machine or a test agent securely. By leveraging environment variables, we can maintain flexibility and security without embedding sensitive information directly into the codebase.

Problem 2

Regarding Problem 2, given that it involves a different AI chat platform, we can apply the same principles, organizing elements and functionalities into separate folders. While introducing abstraction to classes could be beneficial, it may not be necessary due to the distinct login flow of the Anthropic platform. Any other platform-specific variations can be addressed similarly without overcomplicating the structure.

Project 3

For Project 3, Playwright suffices for automating the VS Code web editor. However, testing desktop applications like VS Code requires a different tool, such as Spectron, to simulate user actions. Currently, I am not well-versed in desktop application testing and would need to conduct a proof of concept (POC) to evaluate tools and develop the implementation. This task, however, falls outside the scope of an interview assignment.