Testing - US.1D

Project Name: Generative Pre-trained Transformer (GPT) Web Application

Test Document: Product Quality Assurance Testing for the semester 2 2023 delivery

Test Date: [2023/9/18].

Tester: [Chuansheng Zhou & Yuchen Zhang].

Version: [2023 semester 2] **.

**1. Introduction

Product Quality Assurance testing is intended to ensure that GPT Web applications meet the design and performance requirements to satisfy customer needs and provide a superior user experience. This document will document the scope, methodology, test cases, and test plan for the test.

2. Scope of Testing

This test will cover the following areas:

Case ID	Objective	Description	Step	Expected Result
TC_LOGIN_001	User can successfully login into the application	Users access the login page, enter valid usernames and passwords, and then click the login button.	1. Access the application's login page. 2. Enter valid usernames and passwords. 3. Click the login button.	Users should successfully log in to the application's main page.
TC_LOGIN_002	Users cannot log in when providing invalid usernames and passwords.	Users access the login page, enter invalid usernames and passwords, and then click the login button.	1. Access the application's login page. 2. Enter invalid usernames and passwords. 3. Click the login button.	Users should receive an error message and should not be able to log in to the application.
TC_LOGIN_003	To verify that users cannot log in with an missing input.	Users access the login page with a missing of some information, and then click the login button.	1. Visit the login page. 2. Enter an invalid username and a valid password. 3. Click the "Login" button.	An error message should be displayed, indicating that information is missing.
TC_LOGIN_004	To verify that users cannot log in with empty username and password fields.	Users access the login page with an empty input and then click the login button.	1. Visit the login page. 2. Leave the username and password fields empty. 3. Click the "Login" button.	An error message should be displayed, indicating that both fields are required.
TC_LOGIN_005	To verify that users have not make the SQL injection.	Users cannot access with the input with SQL structure information.	1. Access the application's login page. 2. Enter the user name with SQL inside. 3. Click the login button.	An error message should be displayed, indicating that there should be SQL injection.
TC_USER_INTERFACE_001	To verify that users can navigate between pages smoothly.	Test page navigation functionality.	1. Visit the login page. 2. Log in with valid credentials. 3. Navigate to the main page.	The user should be able to navigate between pages without errors, and each page should display correctly.
TC_USER_INTERFACE_002	To verify that the user interface is responsive and adapts to different screen sizes.	Test the responsiveness of the user interface.	1. Access the application from different devices and screen sizes (e.g., desktop, tablet, mobile). 2. Resize the browser window on a desktop.	The user interface should adapt to different screen sizes, and content should remain readable and functional.

TC_USER_INTERFACE_003	To verify that fonts and styling are consistent and visually appealing.	Test font styles and overall styling.	1. Inspect the fonts, text sizes, and colors used throughout the application. 2. Verify that fonts are consistent and legible. 3. Ensure that styling aligns with the design guidelines.	Fonts should be consistent and easy to read, and styling should match the design specifications.
TC_USER_INTERFACE_004	To verify that images are displayed correctly.	Test image display.	1. Visit pages with images. 2. Verify that images load properly. 3. Check for broken image links.	Images should load correctly, and there should be no broken image links.
TC_USER_INTERFACE_005	To verify that error messages are displayed correctly.	Test error message display.	1. Perform actions that trigger errors (e.g., submitting an incomplete form). 2. Verify that error messages appear in a clear and noticeable manner.	Error messages should be displayed near the relevant fields and be easy to understand.
TC_FUNCTION_PERFORMANCE_001	To verify that the application provides acceptable response times	Test the application's response time for typical user interactions.	1. Access the application with a single user. 2. Perform actions such as logging in, loading a page, or retrieving data.	The application should provide responses within an acceptable time frame for standard user interactions.
TEST_COMMUNICATION_PROTOCOL_001	To verify the application's ability to handle HTTP requests.	Test the application's response to HTTP requests and validate if it handles requests correctly.	1. Send an HTTP GET request to a valid endpoint of the application. 2. Monitor the response status code and headers. 3. Verify that the response contains the expected data or content.	The application should correctly handle the HTTP request, respond with an appropriate status code, and provide the expected content.
TEST_COMMUNICATION_PROTOCOL_002	To assess the application's handling of API rate limiting.	Test how the application responds when API rate limits are exceeded.	1. Send a series of API requests to a specific endpoint that exceed the defined rate limit. 2. Monitor the application's response and status codes.	The application should respond with an appropriate status code (e.g., 429 Too Many Requests) when API rate limits are exceeded, indicating that rate limiting measures are in place.
TEST_COMMUNICATION_PROTOCOL_003	To verify that the application correctly handles valid JSON data.	Test the application's ability to read and process valid JSON data from a file.	1. Provide a JSON file containing well-formed, valid JSON data. 2. Instruct the application to read and parse the JSON file. 3. Verify that the application successfully parses the JSON data without errors. 4. Check if the application extracts and uses the data as intended.	The application should correctly parse and utilize the valid JSON data, and there should be no parsing errors or exceptions.
TEST_COMMUNICATION_PROTOCOL_004	To assess the application's response to invalid JSON data.	Test how the application handles JSON files containing malformed or invalid JSON data.	1. Provide a JSON file containing invalid JSON data with syntax errors or missing required elements. 2. Instruct the application to read and parse the JSON file. 3. Monitor the application's response and error handling.	The application should detect and report the presence of invalid JSON data, providing clear error messages or logging details about the issues encountered.

3.Overall Test Methodology

This test will be conducted using the following methods:

- Manual Testing:The tester will manually execute the test cases and record the results.

4. Whole Test Plan

This test plan will include the schedule of the test, the test environment and the allocation of test resources.

- Test Environment: NA (Manual Testing)
- Test Resources: NA (Manual Testing)