mlangles

CloudAngles

mlangles Generative AI

Test Automation

(The App in App-design feature that generates the test strategy)

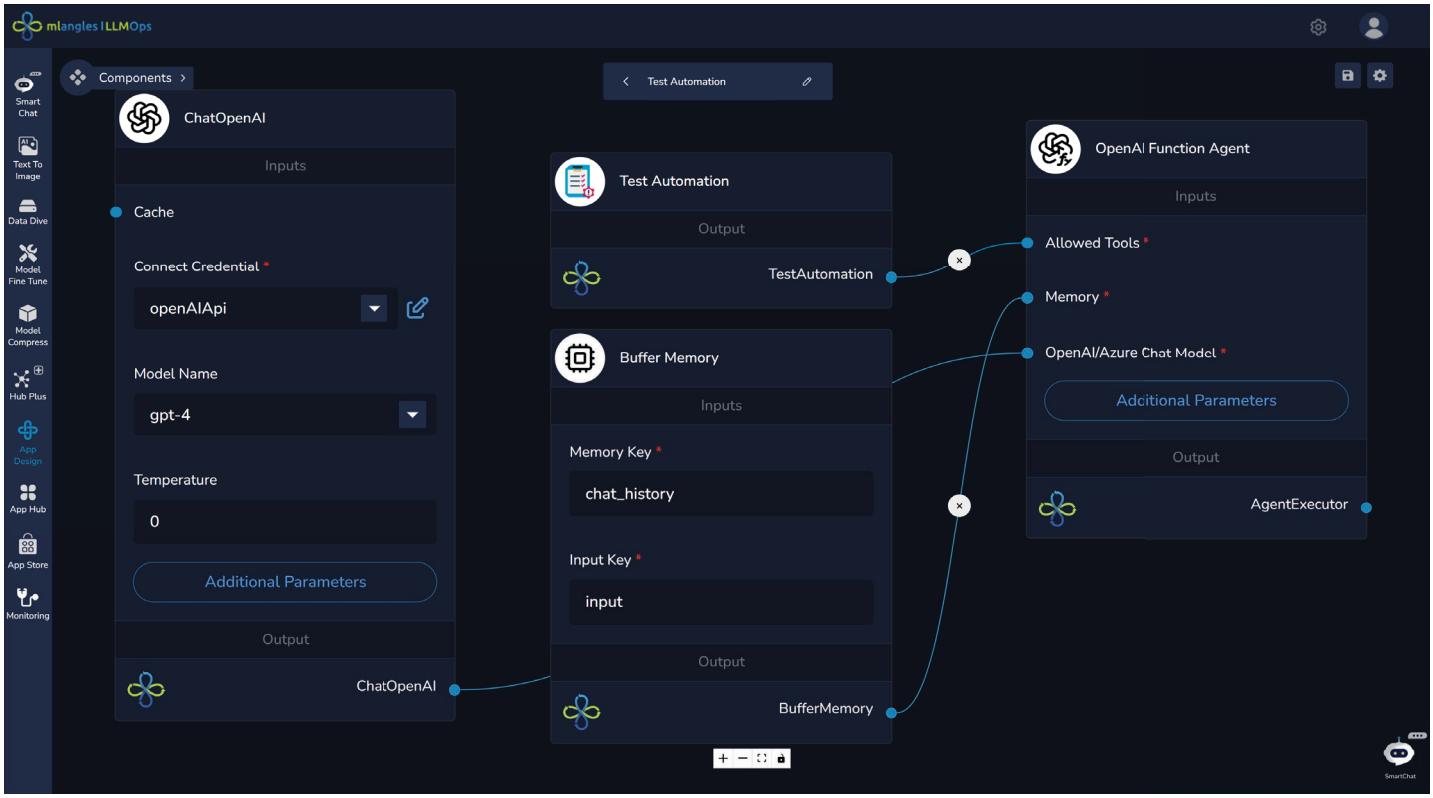
Description:

This application uses LLMs to generate the test strategy document, a comprehensive strategy to ensure through testing of an application across various dimensions, ensuring its functionality, performance, and reliability. The test cases, it gives a comprehensive view of what is to be tested in the component, and the test scripts, they are selenium scripts based on the test cases, for each of the possible components are also generated.

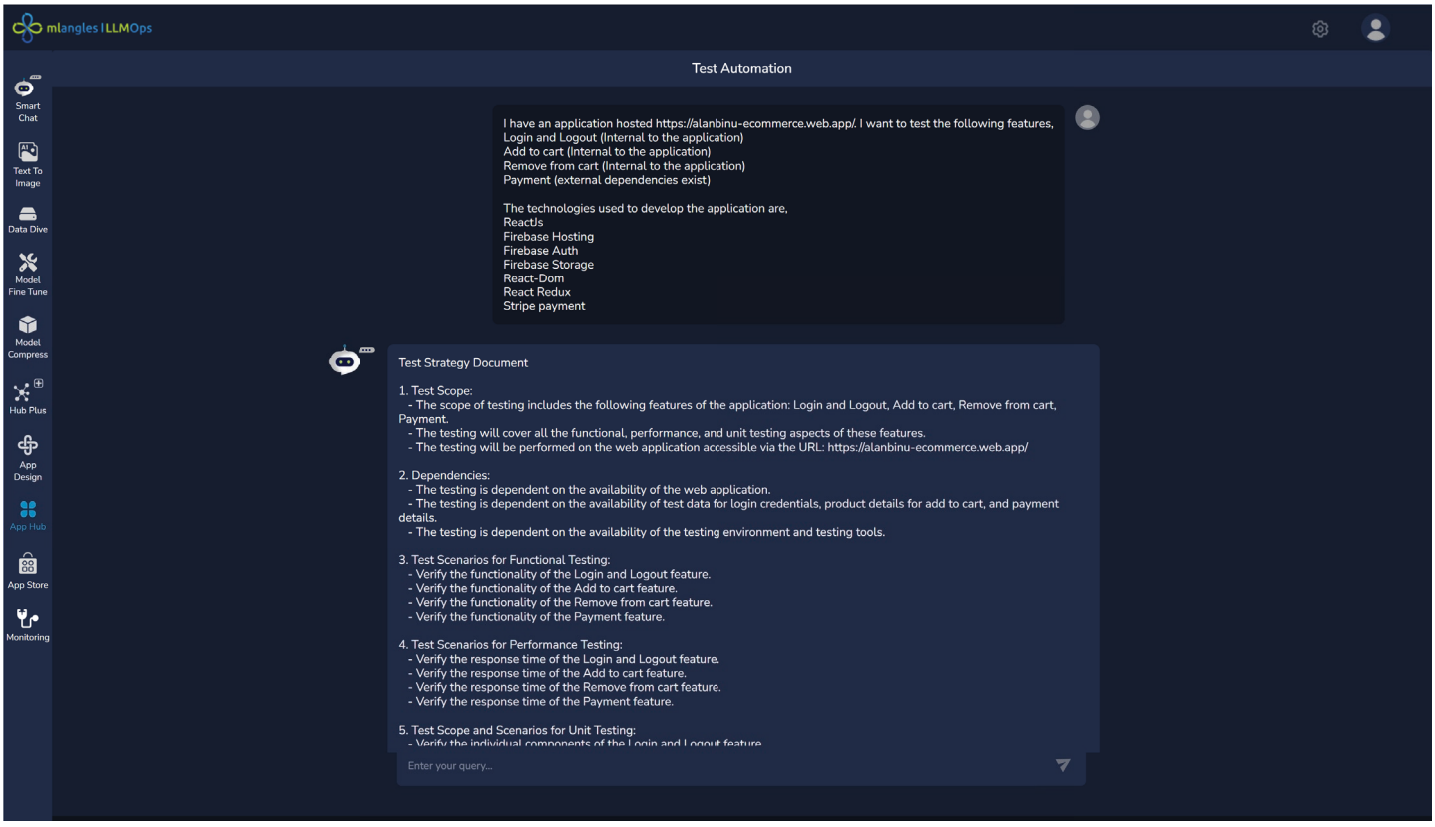
If enterprises have applications that require testing. The test automation tool can provide a structured method to test the components in their application. This eases the process of testing the application, saves cost and mitigate risks for the application.

Explanation:

- The user must create a chat flow in the app-design feature and create the chat flow in the following manner:



- To create the above chat flow in app design the user should click on add new on the top left and give a name to the app with adequate description.
- After that click on the components on the top left and a drop down is presented where the user must drag and drop the following:
  - Chat model: The user must select the large language model. Let's say in the above it is ChatOpenAI and the user has to give the model credentials that includes the open-ai API key. Then the name of the model they want to use and the temperature.
  - Tool: The user must select the test automation tool.
  - Memory:The user must also drag and drop their preferred buffer memory.
  - Agent:Finally, the user must select OpenAI-Function Agent that combines the model, buffer memory and the tool enabling the model to leverage the test-automation tool.
- After dragging and dropping the user must create adequate connections which is to connect the model, test-automation tool and the memory to the agent.
- After this is done the user must save the model for which the option is on the top right. This makes the test-automation app to be available in the app-hub.
- Now in the app-hub the user must go to an app with same name and description of the chat flow created and enter the open-ai API key.
- The chat is presented to the user and the user can chat with the model.



Example Input:

I have an application hosted https://alanbinu-ecommerce.web.app/. I want to test the following features,
Login and Logout (Internal to the application)
Add to cart (Internal to the application)
Remove from cart (Internal to the application)
Payment (external dependencies exist)

The technologies used to develop the application are,
ReactJs
Firebase Hosting
Firebase Auth
Firebase Storage
React-Dom
React Redux
Stripe payment

- In the input prompt the user must give the web link and the features in it they want to test and the tech stack on which the application is developed on.
- Based on this test strategy document, test cases and the test scripts for the test cases are generated. Finally, a pdf is also presented that includes the test strategy document that was given in the output. This is the output that was generated for the above example:

1. Test Scope:

The scope of testing includes the following features of the application: Login and Logout, Add to cart, Remove from cart, Payment.
The testing will cover all the functional, performance, and unit testing aspects of these features.
The testing will be performed on the web application accessible via the URL: https://alanbinu-ecommerce.web.app/

2. Dependencies:

The testing is dependent on the availability of the web application.
The testing is dependent on the availability of test data for login credentials, product details for add to cart, and payment details.
The testing is dependent on the availability of the testing environment and testing tools.

3.Test Scenarios for Functional Testing:

- Verify the functionality of the Login and Logout feature.
- Verify the functionality of the Add to cart feature.
- Verify the functionality of the Remove from cart feature.
- Verify the functionality of the Payment feature.

4.Test Scenarios for Performance Testing:

- Verify the response time of the Login and Logout feature.
- Verify the response time of the Add to cart feature.
- Verify the response time of the Remove from cart feature.
- Verify the response time of the Payment feature.

5. Test Scope and Scenarios for Unit Testing:

- Verify the individual components of the Login and Logout feature.
- Verify the individual components of the Add to cart feature.
- Verify the individual components of the Remove from cart feature.
- Verify the individual components of the Payment feature.

6. Performance Test Types:

- **Load Testing:** To verify the application's performance under the expected load.
- **Load Testing:** To verify the application's performance under the expected load.
- **Endurance Testing:** To verify the application's performance over a long period of time.

7. Performance Metrics:

- **Response Time:** The time taken by the application to respond to a user action.
- **Throughput:** The number of transactions processed by the application per unit time.
- **Resource Utilization:** The amount of resources (CPU, Memory, Disk I/O, Network I/O) used by the application.

8.Test Environment:

- The testing will be performed on the web application accessible via the URL: https://alanbinu-ecommerce.web.app/
- The testing will be performed on different browsers (Chrome, Firefox, Safari, Edge) and different devices (Desktop, Laptop, Tablet, Mobile).

9. Testing Tools:

- **Functional Testing:** Selenium WebDriver for automating the browser actions.
- **Performance Testing:** Apache JMeter for simulating the load and measuring the performance metrics.
- **Unit Testing:** JUnit for testing the individual components of the application.

Test Cases

Test Case 1:

Test Case ID: TC_001

2)**Test Scenario:** Verify the functionality of the Login feature.

3)**Test Steps:**

a) Open the application URL: https://alanbinu-ecommerce.web.app/

b) Click on the Login button.

c) Enter valid login credentials.

d) Click on the Submit button.

4)**Expected Result:** User should be successfully logged in to the application.

Test Case 2:

Test Case ID: TC_002

2)**Test Scenario:** Verify the functionality of the Login feature.

3)**Test Steps:**

a) Login to the application.

b) Click on the Logout button.

4)**Expected Result:** User should be successfully logged out from the application.

Test Case 3:

Test Case ID: TC_003

2)**Test Scenario:** Verify the functionality of the Add to cart feature.

3)**Test Steps:**

a) Login to the application.

b) Select a product.

c) Click on the Add to cart button.

4)**Expected Result:** Selected product should be added to the cart.

Test Case 4::

Test Case ID: TC_004

2)**Test Scenario:** Verify the functionality of the Remove from cart feature.

3)**Test Steps:**

a) Login to the application.

b) Go to the cart.

c) Click on the Remove button for a product.

4)**Expected Result:** Selected product should be removed from the cart.

Test Case 5:

Test Case ID: TC_005

2)**Test Scenario:** VVerify the functionality of the Payment feature.

3)**Test Steps:**

a) Login to the application.

b) Add products to the cart.

c) Go to the cart.

d) Click on the Checkout button.

e) Enter valid payment details.

f) Click on the Pay button.

4)**Expected Result:** Payment should be successful and products should be ordered.

Total number of test cases generated: 5

Test Case 6:

Test Case ID: TC_006

2)**Test Scenario:** Verify the response time of the Login feature.

3)**Test Steps:**

a) Open the application URL: https://alanbinu-ecommerce.web.app/

b) Click on the Login button.

c) Enter valid login credentials.

d) Click on the Submit button.

4)**Expected Result:** The response time for the Login feature should be within the acceptable

Test Case 6:

Test Case ID: TC_006

2)**Test Scenario:** Verify the response time of the Login feature.

3)**Test Steps:**

a) Open the application URL: https://alanbinu-ecommerce.web.app/.

b) Click on the Login button.

c) Enter valid login credentials.

d) Click on the Submit button.

4)**Expected Result:** The response time for the Login feature should be within the acceptablelimit..

Test Case 7:

Test Case ID: TC_007

2)**Test Scenario:** Verify the response time of the Logout feature.

3)**Test Steps:**

a) Login to the application.

b) Click on the Logout button.

4)**Expected Result:** he response time for the Logout feature should be within the acceptablelimit.

Test Case 8:

Test Case ID: TC_008

2)**Test Scenario:** Verify the response time of the Add to cart feature.

3)**Test Steps:**

a) Login to the application.

b) Select a product.

c) Click on the Add to cart button.

4)**Expected Result:** The response time for the Add to cart feature should be within theacceptable limit.

Test Case 9:

Test Case ID: TC_009

2)**Test Scenario:** Verify the response time of the Remove from cart feature.

3)**Test Steps:**

a) Login to the application.

b) Go to the cart.

c) Click on the Remove button for a product.

4)**Expected Result:** The response time for the Remove from cart feature should be within theacceptable limit.

Test Case 10:

Test Case ID: TC_0010

2)**Test Scenario:** Verify the response time of the Payment feature.feature.

3)**Test Steps:**

a) Login to the application.

b) Add products to the cart.

c) Go to the cart.

d) Click on the Checkout button.

e) Enter valid payment details.

f) Click on the Pay button.

4)**Expected Result:** The response time for the Payment feature should be within the acceptablelimit.

Total number of test cases generated: 10

Test Case 11:

Test Case ID: TC_0011

2)**Test Scenario:** Verify the individual components of the Login feature.

3)**Test Steps:**

a) Open the application URL: https://alanbinu-ecommerce.web.app/

b) Click on the Login button.

c) Check the visibility and functionality of the username and password fields.

d) Check the visibility and functionality of the Submit button.

4)**Expected Result:** All the individual components of the Login feature should work asexpected.

Test Case 12:

Test Case ID: TC_0012

2)**Test Scenario:** Verify the individual components of the Logout feature.

3)**Test Steps:**

a) Login to the application.

b) Check the visibility and functionality of the Logout button.

4)**Expected Result:** All the individual components of the Logout feature should work asexpected.

Test Case 13:

Test Case ID: TC_0013

2)**Test Scenario:** Verify the individual components of the Add to cart feature.

3)**Test Steps:**

a) Login to the application.

b) Select a product.

c) Check the visibility and functionality of the Add to cart button.

4)**Expected Result:** All the individual components of the Add to cart feature should work asexpected.

Test Case 14:

Test Case ID: TC_0014

2)**Test Scenario:** Verify the individual components of the Remove from cart feature.

3)**Test Steps:**

a) Login to the application.

b) Go to the cart.

c) Check the visibility and functionality of the Remove button for a product.

4)**Expected Result:** All the individual components of the Remove from cart feature should workas expected.

Test Case 15:

Test Case ID: TC_0015

2)**Test Scenario:** Verify the individual components of the Payment feature.

3)**Test Steps:**

a) Login to the application.

b) Add products to the cart.

c) Go to the cart.

d) Check the visibility and functionality of the Checkout button.

e) Check the visibility and functionality of the payment details fields.

f) Check the visibility and functionality of the Pay button.

4)**Expected Result:** All the individual components of the Payment feature should work asexpected.

Total number of test cases generated: 15

Here are the automation test scripts for the given test cases using Selenium for functional testing:

Test Case 1: Verify the functionality of the Login feature.

```python
# filename: test_login.py
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
driver = webdriver.Chrome()
driver.get("https://alanbinu-ecommerce.web.app/")
login_button = driver.find_element_by_name("login")
login_button.click()
username = driver.find_element_by_name("username")
password = driver.find_element_by_name("password")
username.send_keys("testuser")
password.send_keys("testpassword")
submit_button = driver.find_element_by_name("submit")
submit_button.click()
assert "Logout" in driver.page_source
driver.quit()
```

Test Case 2: Verify the functionality of the Logout feature.

```python
# filename: test_logout.py
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
driver = webdriver.Chrome()
driver.get("https://alanbinu-ecommerce.web.app/")
login_button = driver.find_element_by_name("login")
login_button.click()
username = driver.find_element_by_name("username")
password = driver.find_element_by_name("password")
username.send_keys("testuser")
password.send_keys("testpassword")
submit_button = driver.find_element_by_name("submit")
submit_button.click()
logout_button = driver.find_element_by_name("logout")
logout_button.click()
assert "Login" in driver.page_source
driver.quit()
```

Test Case 3: Verify the functionality of the Add to cart feature.

```python
# filename: test_add_to_cart.py
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
driver = webdriver.Chrome()
driver.get("https://alanbinu-ecommerce.web.app/")
login_button = driver.find_element_by_name("login")
login_button.click()
username = driver.find_element_by_name("username")
password = driver.find_element_by_name("password")
username.send_keys("testuser")
password.send_keys("testpassword")
submit_button = driver.find_element_by_name("submit")
submit_button.click()
product = driver.find_element_by_name("product")
product.click()
add_to_cart_button = driver.find_element_by_name("add_to_cart")
add_to_cart_button.click()
assert "Product added to cart" in driver.page_source
driver.quit()
```

Test Case 4: Verify the functionality of the Remove from cart feature.

```python
# filename: test_remove_from_cart.py
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
driver = webdriver.Chrome()
driver.get("https://alanbinu-ecommerce.web.app/")
login_button = driver.find_element_by_name("login")
login_button.click()
username = driver.find_element_by_name("username")
password = driver.find_element_by_name("password")
username.send_keys("testuser")
password.send_keys("testpassword")
submit_button = driver.find_element_by_name("submit")
submit_button.click()
cart = driver.find_element_by_name("cart")
cart.click()
remove_button = driver.find_element_by_name("remove")
remove_button.click()
assert "Product removed from cart" in driver.page_source
driver.quit()
```

Test Case 5: Verify the functionality of the Payment feature.

```python
# filename: test_payment.py
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
driver = webdriver.Chrome()
driver.get("https://alanbinu-ecommerce.web.app/")
login_button = driver.find_element_by_name("login")
login_button.click()
username = driver.find_element_by_name("username")
password = driver.find_element_by_name("password")
username.send_keys("testuser")
password.send_keys("testpassword")
submit_button = driver.find_element_by_name("submit")
submit_button.click()
product = driver.find_element_by_name("product")
product.click()
add_to_cart_button = driver.find_element_by_name("add_to_cart")
add_to_cart_button.click()
cart = driver.find_element_by_name("cart")
cart.click()
checkout_button = driver.find_element_by_name("checkout")
checkout_button.click()
payment_details = driver.find_element_by_name("payment_details")
payment_details.send_keys("testpaymentdetails")
pay_button = driver.find_element_by_name("pay")
pay_button.click()
assert "Payment successful" in driver.page_source
driver.quit()
```

Note: The above scripts are just examples and may not work as expected because the actual names of the elements in your application may be different. You need to replace the names of the elements with the actual names in your application. Also, the scripts assume that the application displays the messages "Product added to cart", "Product removed from cart", and "Payment successful" when the respective actions are performed. If the application does not display these messages, you need to replace these strings with the actual strings that the application displays.

Here are the automation test scripts for the remaining test cases using Selenium for functional testing:

Test Case 11: Verify the individual components of the Login feature.

```python
# filename: test_login_components.py
from selenium import webdriver
driver = webdriver.Chrome()
driver.get("https://alanbinu-ecommerce.web.app/")
login_button = driver.find_element_by_name("login")
assert login_button.is_displayed()
login_button.click()
username = driver.find_element_by_name("username")
password = driver.find_element_by_name("password")
submit_button = driver.find_element_by_name("submit")
assert username.is_displayed()
assert password.is_displayed()
assert submit_button.is_displayed()
driver.quit()
```

Test Case 12: Verify the individual components of the Logout feature.

```python
# filename: test_logout_components.py
from selenium import webdriver
driver = webdriver.Chrome()
driver.get("https://alanbinu-ecommerce.web.app/")
login_button = driver.find_element_by_name("login")
login_button.click()
username = driver.find_element_by_name("username")
password = driver.find_element_by_name("password")
username.send_keys("testuser")
password.send_keys("testpassword")
submit_button = driver.find_element_by_name("submit")
submit_button.click()
logout_button = driver.find_element_by_name("logout")
assert logout_button.is_displayed()
driver.quit()
```

Test Case 13: Verify the individual components of the Add to cart feature.

```python
# filename: test_add_to_cart_components.py
from selenium import webdriver
driver = webdriver.Chrome()
driver.get("https://alanbinu-ecommerce.web.app/")
login_button = driver.find_element_by_name("login")
login_button.click()
username = driver.find_element_by_name("username")
password = driver.find_element_by_name("password")
username.send_keys("testuser")
password.send_keys("testpassword")
submit_button = driver.find_element_by_name("submit")
submit_button.click()
product = driver.find_element_by_name("product")
assert product.is_displayed()
add_to_cart_button = driver.find_element_by_name("add_to_cart")
assert add_to_cart_button.is_displayed()
driver.quit()
```

Test Case 14: Verify the individual components of the Remove from cart feature.

```python
# filename: test_remove_from_cart_components.py
from selenium import webdriver
driver = webdriver.Chrome()
driver.get("https://alanbinu-ecommerce.web.app/")
login_button = driver.find_element_by_name("login")
login_button.click()
username = driver.find_element_by_name("username")
password = driver.find_element_by_name("password")
username.send_keys("testuser")
password.send_keys("testpassword")
submit_button = driver.find_element_by_name("submit")
submit_button.click()
cart = driver.find_element_by_name("cart")
assert cart.is_displayed()
remove_button = driver.find_element_by_name("remove")
assert remove_button.is_displayed()
driver.quit()
```

Test Case 15: Verify the individual components of the Payment feature.

```python
# filename: test_payment_components.py
from selenium import webdriver
driver = webdriver.Chrome()
driver.get("https://alanbinu-ecommerce.web.app/")
login_button = driver.find_element_by_name("login")
login_button.click()
username = driver.find_element_by_name("username")
password = driver.find_element_by_name("password")
username.send_keys("testuser")
password.send_keys("testpassword")
submit_button = driver.find_element_by_name("submit")
submit_button.click()
cart = driver.find_element_by_name("cart")
assert cart.is_displayed()
checkout_button = driver.find_element_by_name("checkout")
assert checkout_button.is_displayed()
payment_details = driver.find_element_by_name("payment_details")
pay_button = driver.find_element_by_name("pay")
assert payment_details.is_displayed()
assert pay_button.is_displayed()
driver.quit()
```

Note: The above scripts are just examples and may not work as expected because the actual names of the elements in your application may be different. You need to replace the names of the elements with the actual names in your application.

For performance testing, we will use Apache JMeter. Here are the test scripts for the performance test cases:

Test Case 6: Verify the response time of the Login feature.

In JMeter, you would create a test plan as follows:

1. Add a Thread Group with the number of threads (users) you want to simulate.
2. Add a HTTP Request Sampler with the server name or IP and path to your login API.
3. Add parameters for username and password in the Parameters tab.
4. Add a Response Assertion to check that the login was successful (e.g., check for a specific response code or text in the response).
5. Add a Duration Assertion to check that the response time is within the acceptable limit.
6. Add a View Results Tree Listener to view the results of the test.

Repeat the above steps for the other performance test cases (Test Cases 7 to 10), replacing the HTTP Request Sampler with the appropriate API for each feature and adjusting the assertions as necessary.

Test Case 7: Verify the response time of the Logout feature.
Test Case 8: Verify the response time of the Add to cart feature.
Test Case 9: Verify the response time of the Remove from cart feature.
Test Case 10: Verify the response time of the Payment feature.

Note: The above steps are a general guide and may need to be adjusted based on the specifics of your application and APIs. Also, JMeter is a GUI-based tool, so the "scripts" are actually test plans that you create using the tool.

I'm glad you found the information helpful.
If you have any more questions or need further clarification, feel free to ask.

Happy Testing!