**Test Scenarios and Test Cases for UI testing**

**Test Scenarios and test cases:**

Based on the requirements provided, identified the scenarios are as follow:

* User registration and login.
* Product browsing and selection.
* Adding products to the shopping cart.

**a. User Registration and Login:**

- Test Case 1: Verify that a new user can successfully register with valid credentials.

- Test Case 2: Verify that the registration form validates input fields and displays appropriate error messages for invalid data.

- Test Case 3: Verify that a registered user can log in with valid credentials.

- Test Case 4: Verify that the login form validates input fields and displays appropriate error messages for invalid data.

- Test Case 5: Verify that a new user can un-successful register empty mandatory field

- Test Case 6: Verify that user enters name and password which is long text means exceed data limits.

username\_input.send\_keys("dgfuafguaefhuerfhyeurfheriuteruityeruityeruitreuiteruithuertruetreuituertuerterufhgsdfhdsadsghgdfhejgfhegfe")

[name as:’dgfuafguaefhuerfhyeurfheriuteruityeruityeruitreuiteruithuertruetreuituertuerterufhgsdfhdsadsghgdfhejgfhegfe’]

[Password as:”dhfgdshgfadghdslf32yp473561375jbdjkfhay4q3r5ernmsdnfcdsjfheuruierhfdsjfhweuiry834yerehjfnsdmfqeiwulr”]

- Test Case 7: Verify that user enter mobile wrong way with special characters.

mobile\_input.send\_keys("23123@#$$%^29391238129")

Mobile number as[23123@#$$%^29391238129]

Mobile number as[2312329391238129534253245234532452345324523453245]

- Test Case 8: Verify that the login with empty mobile number and password

- Test Case 9: Verify that the login with long password or exceed limits.

- Test Case 10: Verify that the login with unregister number.

**b. Product Browsing and Selection:**

- Test Case 11: Verify that the user can browse different categories of products.

- Test Case 12: Verify that the user can search for products using the search functionality.

- Test Case 13: Verify that the user can view details of a selected product.

- Test Case 14: Verify that the user can navigate back to the previous page after viewing a product.

- Test Case 15: Verify that the user can scroll up and down page to see complete page on product view.

- Test Case 16: Verify that the user can select product at bottom of page.

- Test Case 17: Verify that the user can select only one product at once and add to cart.

- Test Case 18: Verify that the user can select same product many times.

- Test Case 19: Verify that the user can navigate to sub folder of products.

- Test Case 20: Verify that the user can filter item and add to cart.

**c. Adding Products to the Shopping Cart:**

- Test Case 21: Verify that the user can add a product to the shopping cart from the product details page.

- Test Case 22: Verify that the shopping cart updates correctly when a product is added.

- Test Case 23: Verify that the user can remove a product from the shopping cart.

- Test Case 24: Verify that the shopping cart reflects the correct total price of all added products.

**Setup Test Environment:** Implemented scripts in python. All scenarios could not able to automate due time constraints.

- Setup Selenium WebDriver

Make sure you have Selenium WebDriver installed and configured with python programming language.

- Run the script with command from path of your project .

C:\Users\ui\_test> python.exe .\tests\test\_script.py:python.exe .\tests\test\_script.py

**Test plan and test cases for API testing**

**Test scenarios and test cases**

Based on the requirements provided, identified the scenarios are as follow:

**a. User Authentication:**

Test Case 1: Verify successful user login with correct credentials.

Test Case 2: Verify unsuccessful user login with incorrect credentials.

Test Case 3: Verify successful user registration with valid details.

Test Case 4: Verify unsuccessful user registration with existing email or username.

Test Case 5: Verify successful logout of a logged-in user.

Test Case 6: Verify access to authenticated endpoints is restricted for unauthenticated users.

**b. Product Retrieval and Details:**

Test Case 1: Verify retrieval of all products from the database.

Test Case 2: Verify retrieval of a specific product by its ID.

Test Case 3: Verify retrieval of product details including name, description, price, and availability.

Test Case 4: Verify correct handling of non-existing product IDs.

**c. Cart Manipulation (Add/Remove Items):**

Test Case 1: Verify successful addition of a product to the cart.

Test Case 2: Verify correct updating of the cart quantity when adding multiple quantities of the same product.

Test Case 3: Verify successful removal of a product from the cart.

Test Case 4: Verify correct handling of removing a non-existing product from the cart.

Test Case 5: Verify the accuracy of the cart total after adding and removing products.

**d. Order Placement:**

Test Case 1: Verify successful placement of an order with items in the cart.

Test Case 2: Verify correct calculation of the order total including taxes and shipping charges.

Test Case 3: Verify that the inventory is updated after placing an order.

Test Case 4: Verify proper handling of empty cart while placing an order.

Test Case 5: Verify that the user receives an order confirmation email after successful order placement.

**Setup Test Environment:**

API testing using Python's requests library for the JSONPlaceholder API for user authentication, product retrieval and details, cart manipulation, and order placement.

- Run the script with command from path of your project .

C:\Users\ui\_test> python.exe .\tests\test\_script.py:python.exe .\tests\test\_db.py

**Test plan and test cases for Database testing**

**Test scenarios and test cases for data Base:**

Based on the requirements provided, identified the scenarios are as follow:

**Data Integrity Validation:**

-Ensure that all required fields are not null when inserting a new record into the database.

-Test uniqueness constraints to ensure that duplicate records cannot be inserted.

-Check referential integrity by verifying that foreign key constraints are enforced.

**Data Retrieval:**

-Retrieve a specific record from the database and verify that all its fields match the expected values.

-Test retrieval of multiple records based on different criteria (e.g., by name, by ID) and ensure the correct results are returned.

-Validate that appropriate error handling occurs when attempting to retrieve a non-existent record.

**Data Modification:**

-Update an existing record in the database and verify that the changes are reflected accurately.

-Test updating multiple records simultaneously and ensure that all records are updated correctly.

-Verify that appropriate error handling occurs when attempting to update a non-existent record.

**Error Handling:**

-Test scenarios where database operations fail (e.g., due to connectivity issues, database constraints violations) and ensure that appropriate exceptions are raised or error messages are provided.

**Performance Testing:**

-Test the performance of data retrieval operations, especially when dealing with large datasets, and ensure that they meet acceptable performance criteria.

-Measure the time taken for insertion, retrieval, and modification operations and ensure they are within acceptable limits.

**Concurrency Testing:**

-Simulate concurrent access to the database by multiple clients and verify that data integrity is maintained and race conditions are handled appropriately.

-Test scenarios where multiple clients attempt to modify the same record simultaneously and ensure that conflicts are resolved correctly.

**Boundary Testing:**

-Test boundary conditions for data fields (e.g., maximum and minimum values for numeric fields, maximum length for string fields) and ensure that the database handles them correctly.

-Validate behavior when attempting to insert or update records with boundary values.

**Data Migration Testing:**

-Test scenarios where the database schema is altered (e.g., adding new columns, dropping columns) and ensure that existing data is migrated correctly without loss or corruption.

-Validate that data modification operations continue to work as expected after schema changes.

To perform database testing with appropriate tools and libraries such as JDBC for Java and SQLAlchemy for Python, you can follow these steps:

**Setup the Sample Database:** Before writing test cases, you need to have a sample database to work with. You can either create a new database or use an existing one for testing purposes.[sqlite:///example.db]

**Setup Test Environment:** Install the necessary testing frameworks and libraries for Python, used pytest with SQLAlchemy.

- Run the script with command from path of your project .

C:\Users\ui\_test> python.exe .\tests\test\_script.py:python.exe .\tests\test\_db.py