QA Task

E-commerce and Note API Testing with Playwright with Typescript

Prepared By:

Mohamed Moustafa Metwally

❖ Table of Contents

1. Introduction:

- **Project Name:**
  E-commerce Platform and Note API Testing
- **Purpose:**
  To automate the end-to-end testing of an e-commerce platform's UI functionalities and API functionalities for note management.
- **Scope:**
  Automated testing for both UI and API, using Playwright with typescript. Covers functional scenarios for products, counters, user management, and note creation.

---

2. Objectives:

- Verify that user interface elements (such as product add, edit, search, and delete) work as expected.
- Validate that the API operations (e.g., user registration, note creation) meet requirements.
- Ensure that the application functions properly under defined scenarios and responds correctly.

---

3. Test Scope:

- In-Scope:
  - UI Tests: Product addition, deletion, editing, and counter functionality.
  - API Tests: User registration, login, profile update, password change, note CRUD operations.

- Out-of-Scope:
  - Load testing, Performance testing, and security testing (handled separately).

---

4. Test Approach:

- **Automation Framework**: Playwright with Typescript, and API testing with Playwright with Typescript.
- **Execution**: Tests will run on chromium .
- **Reporting**: Playwright-report, with Playwright screenshots for failed tests.

---

5. Test Environment:

- **Browsers:** Chromium for UI testing.
- **Tools and Technologies:** Playwright, JSON**.**
- **Environment:** Testing against staging or development environment URLs.

6.Test Cases

Test Case ID	Description	Steps	Expected Result
TC01	Add a new product	1-Navigate to the e-commerce platform. 2-Click the "Add Product" button. 3-Enter product details (title, price). 4-Save the product.	The product is added to the list, and the UI displays it.
TC02	Edit an existing product	1- Search for the existing product by name. 2- Select the product and click the "Edit" button. 3-Update the product name and/or price. 4-Click "Save"	The product's updated details are displayed
TC03	Delete a product	1- Search for the product to delete. 2- Select the product and click the "Delete" button.	The product no longer appears in the product list.
TC04	Search for a product	1- Enter a search term in the search bar. 2- Observe the displayed results.	Only products matching the search term appear in the list.
TC05	Counter increment functionality	1- Note the initial counter value. 2- Click the "+" button.	The counter value increases by 1.
TC06	Register a new user	Request: POST /users/register body : { "username": "Mhmd", "password": "123456" }	Status code 201 with a user ID in the response.
TC07	Login with registered user	Request: POST /users/login body : { "username": "Mhmd", "password": "123456" }	Status code 200 and a token in the response.
TC08	Update user profile	Request: PATCH /users/profile body: { "email": "Mhmd@gmail.com", "name": "Mhmd" }	Status code 200 and updated profile data in the response.
TC09	Change user password	Request: PATCH /users/change-password Body: { "oldPassword": "123456", "newPassword": "321654" }	Status code 200 with confirmation of the password change.
TC10	Create a new note	Request: POST /notes Body: { " category ": "Sample note category" }	Status code 201 with the note ID in the response.
TC11	Retrieve all notes	Request: GET /notes	Status code 200 with a list of notes.
TC12	Update an existing note	Request: PUT /notes/{noteId} Body: { "category": "Updated note category " }	Status code 200 and updated note category in the response.

TC13	Delete a note	Request: DELETE /notes/{noteId}	Status code 204, and the note no longer appears in subsequent

7.Test Cases Overview:

- **UI Functional Tests:**
  - Add, edit, search, delete product
  - Increment counter
- **API Functional Tests:**
  - User registration, login
  - Update profile, change password
  - Create, update, delete note

---

8. Test Deliverables:

  - Automated test scripts, reports, and logs.
  - Traceability Matrix to map requirements to test cases.

---

9. Roles and Responsibilities:

- **Tester:** Responsible for creating and executing test cases, analyzing results, and reporting defects.
- **Developer:** Responsible for fixing identified defects.

---

10. Entry and Exit Criteria:

- **Entry Criteria:**
  - Access to the testing environment.
  - All relevant components are deployed and stable.
- **Exit Criteria:**
  - All high-priority test cases have been executed and passed.
  - No critical defects remain unresolved.

End of Document