Automated Test Script Documentation

1. Framework and Language Choice

- Framework: SpecFlow with Playwright
- Programming Language: C#
  Reasoning:

- SpecFlow: SpecFlow is a behavior-driven development (BDD) framework that promotes collaboration among developers, testers, and business stakeholders. It allows the creation of feature files with scenarios written in Gherkin syntax, making the tests more readable and understandable.
- Playwright: Playwright is a powerful end-to-end testing framework for web applications. It supports multiple browsers, provides a simple and expressive API, and allows for headless and browser contexts.

In addition to the inherent advantages offered by the chosen programming language and framework, another compelling reason for my preference is the practical experience gained through daily use in my current job. Utilizing these tools in my day-to-day tasks has not only deepened my understanding but also bolstered my confidence and familiarity with their functionalities.

2. Overview

In the course of completing the test assignment, I want to address a specific practice I employed,even though it's not considered the best or my usual way of doing things and surely doesn't align with best practices. During the execution of automated tests, I found it necessary to incorporate Task.Delay in several instances. This departure from conventional testing methodologies was driven by the inherent instability of the test application and inconsistencies in the Document Object Model (DOM).

Upon encountering failures in the actual run of the tests—despite successful execution in debug mode—I embarked on a troubleshooting process. It became apparent that subtle delays introduced through Task.Delay rectified the issues. It's crucial to note that this approach deviates from the standard practices I adhere to in the professional testing environment.

I also want to note that, for the overall purpose of the tests, I created a pre-verified user. The credentials for this user are retrieved from the testsettings.json file. This user is used in certain scenarios to ensure smoother and more controlled test execution.

I want to highlight that this is not my typical approach at work, and I only did it

because of the unique challenges faced during this test assignment. In the future, I'll stick to the standard practices and try to avoid using delays unless absolutely necessary.

3. Test Scenarios:

New User Registration:

Positive Test Cases:

1. **Scenario:** New user successfully creates an account.
    - **Steps:**
        1. Navigate to the registration page.
        2. Enter valid name, email, and password.
        3. Optional: Provide company and address details.
        4. Click on the "Create Account" button.
    - **Expected Result:** User account is created successfully.
2. **Scenario:** New user creates an account with the minimum required information.
    - **Steps:**
        1. Navigate to the registration page.
        2. Enter valid name, email, and password.
        3. Click on the "Create Account" button.
    - **Expected Result:** User account is created successfully.

Negative Test Cases:

3. **Scenario:** New user fails to provide a required field.
    - **Steps:**
        1. Navigate to the registration page.
        2. Leave one of the required fields (name, email, or password) empty.
        3. Click on the "Sign Up" button.
    - **Expected Result:** Validation message appears indicating the missing required field.
4. **Scenario:** New user provides an invalid email format.
    - **Steps:**
        1. Navigate to the registration page.
        2. Enter a valid name, invalid email, and valid password.
        3. Click on the "Create Account" button.

- **Expected Result:** Validation message appears indicating the invalid email format.

Project Management:

Positive Test Cases:

5. **Scenario:** User successfully adds a new project.
   - **Steps:**
     1. Navigate to the projects section.
     2. Click on the "Create Project" button.
     3. Enter a valid project name.
     4. Click on the "Save" button.
   - **Expected Result:** New project is added successfully.
6. **Scenario:** User edits an existing project.
   - **Steps:**
     1. Navigate to the projects section.
     2. Click on the "Edit" button of an existing project.
     3. Modify the project name.
     4. Click on the "Save" button.
   - **Expected Result:** Project is edited successfully.

Negative Test Cases:

7. **Scenario:** User attempts to add a project with a duplicate name.
   - **Steps:**
     1. Navigate to the projects section.
     2. Click on the "Create Project" button.
     3. Enter a project name that already exists.
     4. Click on the "Save" button.
   - **Expected Result: Not specified by the acceptance criteria**
     Validation message appears indicating the duplicate project name/
     The project successfully created
8. **Scenario:** User deletes a project with associated tasks.
   - **Steps:**
     1. Navigate to the Dashboard section.
     2. Click on the "Delete" button of a project with tasks.
   - **Expected Result:** Confirmation message appears, preventing deletion

Task Management:

Positive Test Cases:

9. **Scenario:** User adds a new task to a project.
    - **Steps:**
        1. Navigate to the Dashboard section.
        2. Under a specific project, click on the "Add Task" button.
        3. Enter valid task details.
        4. Click on the "Create" button.
    - **Expected Result:** New task is added successfully.
10. **Scenario:** User edits an existing task.
    - **Steps:**
        1. Navigate to the tasks section of a specific project.
        2. Click on the "Edit" button of an existing task.
        3. Modify task details.
        4. Click on the "Update" button.
    - **Expected Result:** Task is edited successfully.

Negative Test Cases:

11. **Scenario:** User adds a task with missing required fields.
    - **Steps:**
        1. Navigate to the tasks section of a specific project.
        2. Click on the "Add Task" button.
        3. Leave one of the required fields empty.
        4. Click on the "Create" button.
    - **Expected Result:** Validation message appears indicating the missing required field.
12. **Scenario:** User attempts to delete a task.
    - **Steps:**
        1. Navigate to the tasks section of a specific project.
        2. Click on the "Delete" button of a task.
    - **Expected Result:** A confirmation message appears to confirm the task deletion.

TaskDB Functionality:

Positive Test Cases:

13. **Scenario:** User views all tasks belonging to their projects.
    - **Steps:**
        1. Navigate to the TaskDB section.
        2. Verify that all tasks belonging to the user's projects are displayed.
    - **Expected Result:** All tasks are visible.
14. **Scenario:** User performs a successful search on tasks.
    - **Steps:**
        1. Navigate to the TaskDB section.
        2. Enter a search query in the search bar.
    - **Expected Result:** Relevant tasks matching the search query are displayed.
15. **Scenario:** User successfully sorts tasks.
    - **Steps:**
        1. Navigate to the TaskDB section.
        2. Click on the sorting option (by summary descending/ascending).
    - **Expected Result:** Tasks are sorted based on the selected criteria.

Negative Test Cases:

16. **Scenario: User performs a search with no matching results.**
    - **Steps:**
        1. Navigate to the TaskDB section.
        2. Enter a search query that does not match any tasks.
    - **Expected Result:** No tasks are displayed

4. Automated test cases

Following, there are the Features incorporating automate test cases for the above scenarios**:**

Feature: User_Registration

1. Scenario: New users can create an account with optional company and address.
    - Covered Test Cases:
        - Positive case for creating an account with valid information, including optional company and address.

(Note that before executing this specific scenario, ensure that you replace 'YOUR_NEW_EMAIL@example.com' with an actual, unused email address. This step is crucial as the test checks for the uniqueness of the email, and using an already registered email may lead to test failures.)

2. Scenario: Attempt to create an account without providing a required field (name, email, or password).
    - Covered Test Cases:
        - Negative cases for each required field (name, email, password) and combinations (nameAndEmail, nameEmailPassword).
        - Checks that appropriate validation messages are displayed when attempting to create an account without required information.

Feature: TaskDB_Operations

3. Scenario: Users can view all tasks belonging to their projects in TaskDB and perform searching.
    - Covered Test Cases:
        - Positive case for viewing all tasks belonging to the user's projects.
        - Positive case for searching for a specific task.
        - Positive case for sorting tasks based on different criteria.

Feature: Project_Management

4. Scenario: Users can Add / Edit / View / Delete projects.

- Covered Test Cases:
  - Positive cases for adding, editing, and deleting a project.
  - Checks that the project is successfully added, updated, and removed.

5. Scenario: Attempt to create a project without providing a required field (name, or description).
   - Covered Test Cases:
     - Negative cases for each required field (name, description, nameAndDescription).
     - Checks that appropriate validation messages are displayed when attempting to create a project without required information.

Feature: TaskManagement

6. Scenario: Users can Add / Edit / View / Delete tasks for a specific project.
   - Covered Test Cases:
     - Positive cases for adding, viewing, editing, and deleting a task.
     - Checks that the task is successfully added, viewed, updated, and removed.

7. Scenario: Attempt to update a task without providing a required field (name, or description).
     - Covered Test Cases:
       Negative cases for updating a task without providing required informationThis scenario covers negative cases for each required field (name, description, nameAndDescription) when attempting to update a task

I chose to automate these specific test cases because they represent critical and frequently repeated scenarios in the application. The selected cases cover a range of functionalities, including user registration, taskDB operations, project management, and task management. By automating these cases, we ensure thorough and efficient testing of core features that are essential for the application's functionality. Automation allows us to quickly and consistently execute tests, providing scalability and repeatability in our testing efforts. Additionally, these cases cover both positive and negative scenarios, offering a comprehensive validation of the application's behavior in various situations. This strategic selection aligns with the goal of achieving robust test coverage and

ensures that the application meets specified requirements across different user interactions

- Instructions on how to execute the test cases are the readMe file.