**Project Overview:**

The automation project involves using Python, pytest, and Selenium to automate the testing of a given web application: https://prompthero.com/. The application's functionality includes Scrapping various features like generation parameters, link testing, user name, content prompts and model used from Web Application. The project structure includes different Python files for different modules, along with corresponding test files for testing each module's functionality using pytest.

**Approach:**

**Project Structure:**

**Python Files:**

* + **LinkTest.py**: Extracts all links present in the given Web Page.
  + **ModelUsed.py**: Extracts the model used data from given Webpage.
  + **Generationparameters.py**: Extracts the Generation Parameters data from given Webpage.
  + **PromptContent.py**: Extracts all lines where the line contains prompt word from given Webpage.
  + **UserName.py**: Extracts Username from given Webpage.
  + **ExtractedFile.py**: Contains functions to extract information from files.
  + **LoginPage.py**: Implements user login functionality for given Webpage

**Pytest Files**

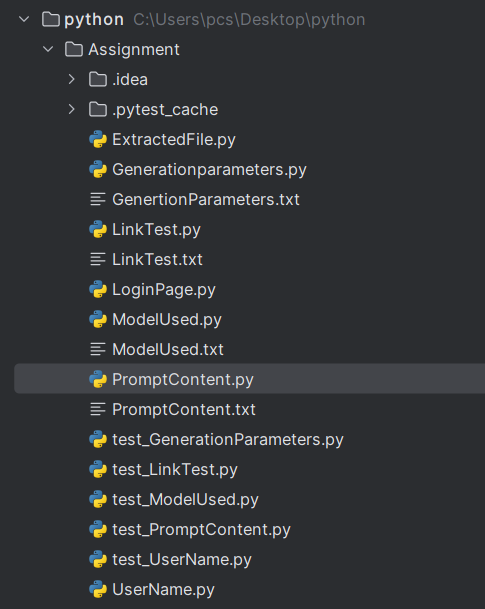
* + **test\_GenerationParameters.py**: Contains pytest test cases for Generationparameters.py.
  + **test\_LinkTest.py**: Contains pytest test cases for LinkTest.py.
  + **test\_ModelUsed.py**: Contains pytest test cases for ModelUsed.py.
  + **test\_PromptContent.py**: Contains pytest test cases for PromptContent.py.
  + **test\_UserName.py:** Contains pytest test cases for UserName.py.
* **Automation Process:**
  + For each module (generation parameters, link testing, model used, prompt content, and user name), there are corresponding Python files containing functions to interact with the web application and perform the required actions.
  + For each module, there are also separate test files that use pytest to run test cases against the functions in the corresponding Python files.
  + Selenium is used to interact with the web application, simulate user actions, and validate expected outcomes.

**How to run the Project:**

* **Install python libraries:**
  + 1. Pip install pytest
    2. pip install selenium
* **Navigate to Directory:**
  + 1. cd OpenAgent\_AutomationAssignment
    2. Run the below command from current directory to run the project **Pytest**
    3. After execution output.txt will be saved inside current directory where all scraped data is available
* **Get git clone** : https://github.com/MandaGayatri/OpenAgent\_AutomationAssignment
* **Challenges Faced and Solutions Implemented:**
  + Dynamic Content: The web application has dynamically changing content, making it challenging to write stable tests.
    - Solution: Used waits in Selenium to ensure elements are loaded before interacting with them.
  + Test Data Management: Managing test data and configuration files for different modules are complex.
    - Solution: Keeping sample data or configuration files separate and version-controlled. Used proper file-naming conventions for clarity.

**Conclusion:**

The automation project successfully utilizes Python, pytest, and Selenium to automate testing for various modules of the web application. By structuring the project effectively and using proper automation techniques, challenges related to dynamic content and test data management are addressed.



To run this project run the below command:

pytest