**Building an Autonomous AI Agent for Form Filling**

**Introduction**

In this test, I developed an autonomous AI agent capable of filling out online forms automatically. The primary objective was to streamline and automate the tedious process of form filling, leveraging the power of artificial intelligence and web automation techniques.

**Tech Stack:**

* **Web Automation:** Our solution utilizes Selenium, a powerful web automation tool, to interact with online forms dynamically. Selenium allows us to simulate user actions such as clicking buttons and entering text, enabling seamless automation of the form-filling process.
* **Natural Language Generation (NLG):** To generate realistic data for form fields, we leverage OpenAI's NLG capabilities. By feeding prompts to OpenAI's language model, we can generate human-like responses for various form inputs, ensuring authenticity and coherence in the filled-out forms.

**My Approach:**

For this test, I adopted a combination of basic Selenium scripting and advanced Selenium techniques. This approach offers the best of both worlds, combining simplicity and reliability with enhanced robustness and flexibility.

**Reasons for Choosing the Combination Approach:**

* **Basic Selenium Scripting:** We opted for basic Selenium scripting due to its simplicity and ease of implementation. This approach allowed for rapid development and prototyping, making it ideal for our project's requirements.
* **Advanced Selenium Techniques:** Additionally, we incorporated advanced Selenium techniques such as explicit waits and dynamic XPath generation to enhance the reliability and stability of our automation script. These techniques ensure smooth execution even in dynamic web environments, minimizing the risk of errors and failures.

**Other Approaches:**

While our chosen combination approach proved to be the most suitable for our project, several other approaches were considered:

**Parallel Test Execution with Selenium Grid:** This approach leverages Selenium Grid to execute multiple instances of the automation script in parallel across different browsers and platforms. By distributing the workload across multiple nodes, the overall form-filling process can be accelerated, improving efficiency and scalability.

**Hybrid Approach with JavaScript Injection:** In this approach, Selenium is used to automate user interactions on the form, while JavaScript injection techniques are employed to dynamically populate form fields with generated data. By combining the strengths of both Selenium and JavaScript, this approach offers flexibility and versatility in handling complex form structures.

These approaches were not chosen for this specific implementation due to their potential complexity and the simplicity of the task at hand. The Basic Selenium Scripting approach sufficed for the requirements of automating form filling using Selenium and API for data generation.

**Implementation:**

The code sets up an automated process to fill out a web form using Selenium and OpenAI's GPT-3.5 for generating dummy data. First, it initializes the Selenium WebDriver with Chrome options and installs the necessary driver using webdriver-manager. The script opens a specified web form URL and allows the page to load.

For data generation, a function using OpenAI's API creates realistic dummy data for each form field, such as names, addresses, and additional information. The generated data is stored in a dictionary, mapping each field's description to its generated value.

To fill out the form, the code uses explicit waits provided by Selenium’s WebDriverWait to ensure each form element is present before attempting to interact with it. This increases the robustness of the automation, making it less likely to fail due to elements not being loaded in time. For each form field, the corresponding generated data is entered sequentially.

The script also uploads a resume from a specified path and submits the form. Finally, it waits for a few seconds to ensure the submission process completes before closing the WebDriver. This combination of web automation and AI-generated data effectively simulates the process of filling out a form as a human would.

**Conclusion:**

In conclusion, by considering alternative approaches and evaluating their suitability, this project ensures a comprehensive exploration of automation strategies. The integration of Selenium for web automation and OpenAI's GPT-3.5 for data generation leads to a more resilient, efficient, and scalable solution for form filling. This approach highlights the potential for AI to streamline and enhance repetitive tasks.

GitHub Link: https://github.com/Rajeshmummidisetti/AI-agent-for-automatic-form-filling

Working Video/Screen Recording: <https://youtu.be/_hwCpejiUcY?si=Q14QxngXro2dkt_L>

Submitted by: Mummidisetti Laxmi Rajesh Gupta