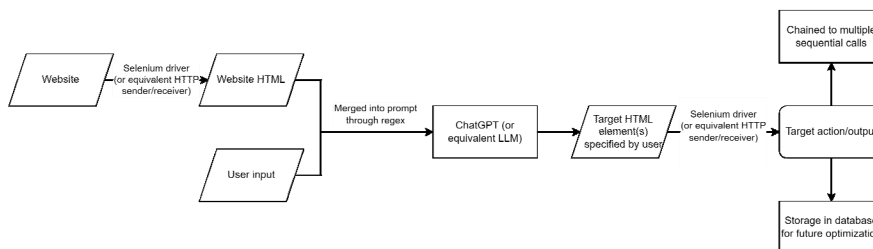


# HCI using GPT4

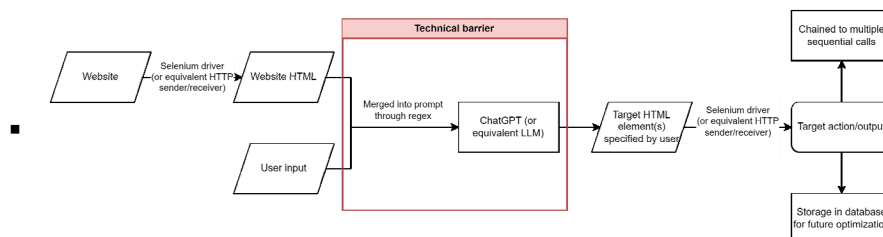
March 17, 2023 10:28 PM

## General

- Project Scope
  - We want to understand the feasibility of building an AI-powered Browser Automation Bot. This Bot will takes a user's natural language input and execute actions on the user's behalf on a browser.
- Bot spec:
  - Based on user instructions, Bot should be able to go to a specified URL and execute browser automations on the user's behalf
  - Bot should be able to know how to navigate the browser based on commands e.g. "select name field", "type in bananas", "press confirm"
  - Bot should be able to navigate to different pages and continue to execute the original user command even after the page has changed e.g. search on Google "bananas" and give back the first 10 results
  - Bot should be able to remember how to execute a set of instructions after it's been taught.
  - Bot should be able to take in certain parts of a user's inputs as variables. And execute them against a template
- Outputs:
  - Understand whether an AI-powered Browser Automation Bot is possible
    - *Theoretically, it should be possible to build a browser automation bot but it's precision/accuracy (and hence usability) would be determined by technical limitations of the components involved*
    - *The framework of such a bot would be as follows:*



- *However, a technical barrier exists at the point of using the Website HTML as input to the LLM:*
  - *Specifically, the LLM cannot accommodate the large input size of the Website HTML. This issue is expanded in the points below*



- Complexities in building the bot
  - *Navigating to URLs and interact with HTML elements are built-in functions of Selenium. The crux of this problems lies with the steps in between (i.e., input to the LLM and correct interpretation of the user instructions with respect to the Website HTML)*
  - *Input limitation to LLM*

- Current OpenAI's APIs for their GPT models have a token limit of ~4000. HTMLs of common web interfaces (e.g., Google home page) often have upwards of 10 000+ tokens depending on the complexity/how dynamic the frontend is
- There are various workarounds for the token limit, with common solutions being:
  - ◆ Segmentation/division of prompt - theoretically would perform on complex HTML structures where certain elements span beyond the 4K token limit resulting in incomplete HTML syntax being input to the LLM. This method works better with natural languages than for formal languages like HTML
  - ◆ Compression of HTML - while there are no standardized tool for compression of HTML into only displayed elements (i.e., removing JS and styling elements), it is possible to build an in-house compressor looking for only displayed/interactable elements and feed this into the LLM which could significantly reduce token consumption
  - ◆ [https://www.reddit.com/r/GPT3/comments/w1e44z/has anyone found a workaround to the max token/?utm\\_source=share&utm\\_medium=ios\\_app&utm\\_name=iossmf](https://www.reddit.com/r/GPT3/comments/w1e44z/has_anyone_found_a_workaround_to_the_max_token/?utm_source=share&utm_medium=ios_app&utm_name=iossmf)
  - ◆ [https://www.reddit.com/r/MachineLearning/comments/zjf45w/d\\_getting\\_around\\_gpt3s\\_4k\\_token\\_limit/?utm\\_source=share&utm\\_medium=ios\\_app&utm\\_name=iossmf](https://www.reddit.com/r/MachineLearning/comments/zjf45w/d_getting_around_gpt3s_4k_token_limit/?utm_source=share&utm_medium=ios_app&utm_name=iossmf)
- Ultimately, this technical limitation may be overcome when models with larger input token limits (i.e., GPT4 API) are released although it remains an economical consideration and a component of the bot that could be further optimized by sub-processes (e.g., HTML compression)
- Interpretation of HTML
  - GPT4 appears to be able to parse simple HTML pages containing simple search bar elements and return the appropriate text input element and text search button element as per user request.
  - However, it is difficult to predict whether this would scale to real-world common websites which many times have odd formatting to accommodate backend requests. This also does not take into consideration that some websites have dynamically loaded elements via AJAX and hence would first need a triggering action (e.g., hover over) to generate the searchable text input box)
- Documented test cases to understand the bot's performance and robustness
  - A few test cases were performed using GPT3.5 via ChatGPT interface. Code available in links listed below
    - Interestingly, GPT3.5 seems to be able to detect that there are multiple text inputs implicitly in the third example. The module receiving the LLM output should take into consideration this type of variation in output (i.e., not always expecting a single HTML element and know how to deal with multiple returns)
  - <https://codepen.io/lfigu042/pen/mgYGBG>

```

</form>
</div>
</section>

```

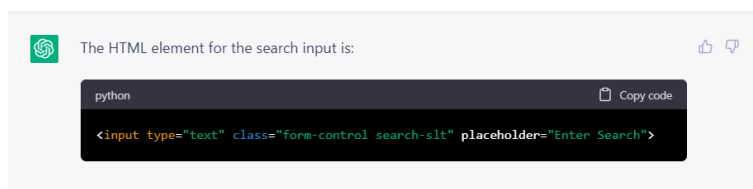
```

</body>

```

Obtain the HTML element for the search input from the above HTML document:

□



```

</body>

```

Obtain the HTML element for the search button from the above HTML document:

</body>

Obtain the HTML element for the search button from the above HTML document:



The HTML element for the search button is:



python

Copy code

```
<button type="button" class="btn btn-danger wrn-btn">Search</button>
```

It can be found in the `<div>` with class `col-lg-3 col-md-3 col-sm-12 p-0` inside the `<form>` element.

- <https://codepen.io/billzhao/pen/wzxMbW>

```
</div>  
</div>
```

```
</body>  
</html>
```

Obtain the HTML element for text input from the HTML above:



The HTML element for text input is:



python

Copy code

```
<input type="text" class="form-control" name="x" id="search" placeholder="Search">
```

It is located inside the `<div>` with the class `input-group`.

```
</body>  
</html>
```

Obtain the HTML element for the search button from the HTML above:



The HTML element for the search button is:




php

Copy code

```
<button class="btn btn-default" type="button">  
<span class="glyphicon glyphicon-search"></span>  
</button>
```

- <https://bootsnipp.com/snippets/WMlMa>

Obtain the HTML element for the search input from the above HTML document:

 There are three search input elements in the above HTML document. Here are their HTML elements:

1. First search input element:

python

Copy code

```
<input type="text" class="form-control search-slt" placeholder="Enter Pickup City">
```

2. Second search input element:

python

Copy code

```
<input type="text" class="form-control search-slt" placeholder="Enter Drop City">
```

3. Third search input element:

php


Copy code

```
<select class="form-control search-slt" id="exampleFormControlSelect1">
  <option>Select Vehicle</option>
  <option>Example one</option>
  <option>Example one</option>
  <option>Example one</option>
  <option>Example one</option>
  <option>Example one</option>
  <option>Example one</option>
  <option>Example one</option>
</select>
```

So, there are two input elements of type `text` and one input element of type `select`.

```
</div>
</section>
```

Obtain the HTML element for the search button from the above HTML document:

 The HTML element for the search button is:

html

Copy code

```
<button type="button" class="btn btn-danger wnn-btn">Search</button>
```

- Alternatives to consider
  - *Computer vision (or models involving CV) would be another alternative to LLM as the interpreter to websites and user input although NLP will likely still need to be used to parse human input and translate it into spatial coordinates for the CV model to act on.*
- Demonstration of the bot
- Source code
  - [https://github.com/SteveHQiu/HCI\\_GPT4](https://github.com/SteveHQiu/HCI_GPT4)
- If using selenium, will there be a way that hosts can not know whether automation is being used?
  - <https://stackoverflow.com/questions/33225947/can-a-website-detect-when-you-are-using-selenium-with-chromedriver>
- Proposed Approach
  - GPT3.5 <> LangChain <> Selenium library
  - GPT3.5 as the LLM
  - LangChain as the framework to interact with Python
    - Create the "Browser Agent"

- Create the "Browser Automation Tool"
  - Question: Can the Browser Automation Tool be a universal tool for any URL? Or is the tool deterministic to a specific URL? I.e. can we dynamically generate python script based on user instructions?

•