Appendix

Table of Contents

A Broader Impacts

This paper presents a novel method of using Large Language Models (LLMs) to execute web actions through hierarchical prompting and few-shot examples. The broader impact of our work spans both technological and societal dimensions.

Technological Impact: By enabling LLMs to perform web actions, a vast range of applications can be automated and improved, leading to greater efficiencies and new capabilities. For instance, customer service can be improved by automating frequently performed web actions, thereby increasing productivity and customer satisfaction. The automation can streamline processes and free up human resources for more complex, nuanced interactions.

The idea of hierarchical prompting opens up new avenues for artificial intelligence research, especially in the field of natural language understanding and web interaction. As LLMs become more and more capable, we need scalable approaches for decision making that allow these models handle a diverse array of tasks while being able to fail gracefully. Approaches that build on ideas around task decomposition, modularity and reusability would enable improved scalability and failure handling.

Societal Impact: Equipping LLMs with the ability to carry out web tasks opens up a a variety of possibilities for societal benefits and change. For instance, the technology could transform accessibility to digital services, in particular for the elderly or individuals with certain disabilities.This technology also serves as a powerful assistant to humans by automating repetitive tasks hence reducing cognitive burden on employees across different sectors such as customer care and IT services.

However, while these advancements promise substantial benefits, they also present challenges that need to be diligently addressed. The potential for misuse of this technology, such as unauthorized data access, privacy breaches, or malicious automation, is of serious concern. It is essential to develop robust security measures and clear ethical guidelines for the use of LLMs that respect users' privacy. Moreover, care must be taken that the deployment of such technologies is to eventually augment humans and help free them up for more creative and nuanced interactions. This calls for careful planning and implementation, including strategies for re-skilling or up-skilling.

B Ablations

We conduct two ablations. The first studies how the performance of all methods vary with model architectures. The second looks at the performance gains from adding chain-of-thought reasoning to all prompts.

B.1 Effect of Chain-of-Thought Reasoning

While we initially did not have chain-of-thought reasoning [4], we found that adding it in helped `HeaP Few-shot` rationalize a particular action well and boost performance. We wanted to understand which tasks in particular were helped by chain-of-thought reasoning, and what the trade-offs were.

We wanted to test the following hypothesis for `HeaP Few-shot` with and without chain-of-thought:

1. **Hypothesis 1: Chain-of-thought reasoning helps across all tasks.** Even though chain-of-thought reasoning makes prompts slightly longer, the added step of rationalizing actions should always boost performance.
2. **Hypothesis 2: Chain-of-thought reasoning particularly helps in multi-step tasks.** Multi-step tasks often require breaking down a problem into a set of steps and executing each step. While demonstrations certainly show how to break down task, adding chain-of-thought better rationalizes this breakdown and helps generalize to new tasks not covered in the demonstrations.

We compared `HeaP Few-shot` with two versions - having chain of thought, and not having. Fig. 10 shows a plot of success rate for each of the 3 clusters of tasks - single, composite, multi.

Hypothesis 1: Chain-of-thought reasoning helps across all tasks.. We find this to be true since for all tasks, chain-of-thought performs either equally or better. This confirms that the extra tokens consumed by the reasoning does not hurt performance and in fact helps significantly in some cases.

Hypothesis 2: Chain-of-thought reasoning particularly helps in multi-step tasks. We find this to be true as well. Looking at the multi-step tasks, chain-of-thought has the largest performance gains compared to any other cluster. The performance gains are he largest in book-flight and search-engine where the horizon length is the

[4] Chain-of-Thought Prompting Elicits Reasoning in Large Language Models https://arxiv.org/abs/2201.11903

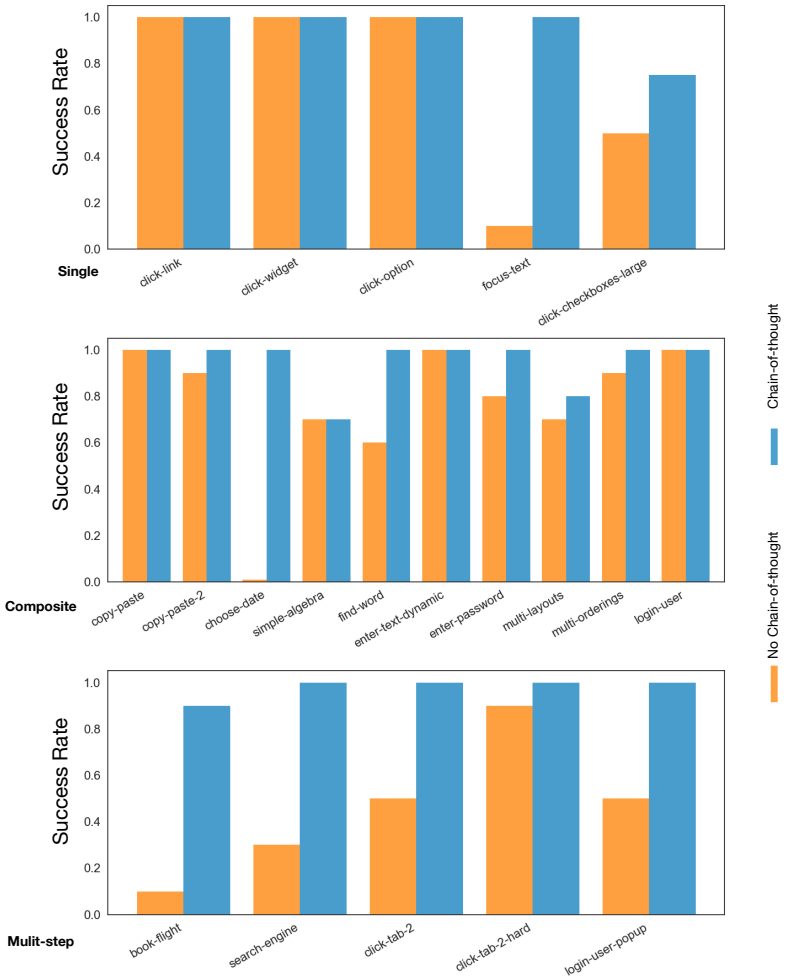

Figure 10: Success rate of `HeaP Few-shot` with both chain-of-thought and without.

largest. In comparison, for single and composite the performace gains vary, being higher for certain tasks like choose-date and find-word and zero for others like click tasks.

B.2 Model Scaling

While we developed the prompts with `text-davinci-003`, we wanted to compare how performance varies with newer models `gpt-3.5-turbo` and `gpt-4`. `gpt-3.5-turbo` is also an InstructGPT model optimized for chat and trained at 1/10th the price of `text-davinci-003`. `gpt-4` is a significantly larger model, and capable of solving more complex tasks.

We wanted to test the following hypothesis:

1. **Hypothesis 1: GPT-4 improves performance across all methods, but both hierarchical prompting and few-shot examples help.** GPT-4 is a powerful model and with an exhaustive set of instructions in the prompt, it should be able to achieve perfect performance. However, designing such exhaustive prompts for all web tasks is challenging. We hypothesize that hierarchical prompting helps break down and scope instructions leading to better performance for GPT-4. Similarly, few-shot examples helps GPT-4 ground instructions in the webpage.
2. **Hypothesis 2: `gpt-3.5-turbo` slightly worse than `text-davinci-003` given few-shot examples.** Practioners have noted that while `gpt-3.5-turbo` has better 0 shot performance, `text-davinci-003` is trained on a more diverse set of tasks and performs better with k-shot learning `https://scale.com/blog/chatgpt-vs-davinci#Introduction`. Since 0 shot performance for web-tasks is challenging without exhaustive instructions, we hypothesize that `text-davinci-003` will perform better.

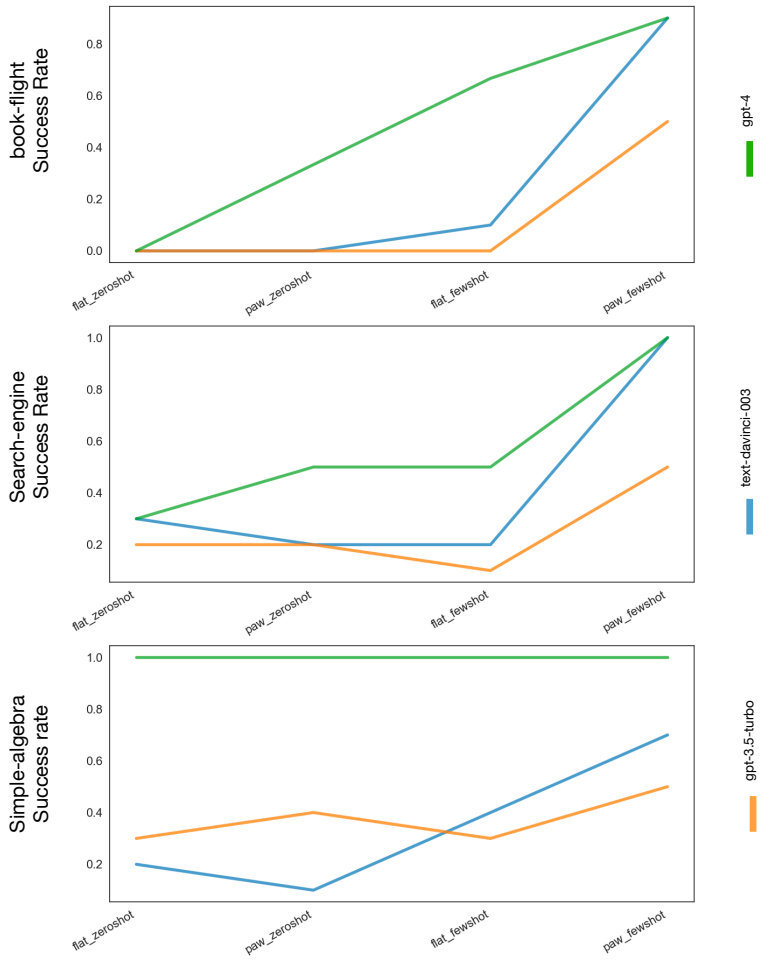

Figure 11: Success rate of all methods with different models.

We compare three language models `text-davinci-003`, `gpt-3.5-turbo` and `gpt-4` for all baselines on 3 tasks from MiniWoB++ - book-flight, search-engine, simple-algebra. Fig. 11 shows a plot for each of these tasks.

Hypothesis 1: GPT-4 improves performance across all methods, but both hierarchical prompting and few-shot examples help. We find this to be true. GPT-4 is unambiguously better than any other model and improves all methods. However, as shown in book-flights and search-engine, both hierarchical prompting (`HeaP`) and few-shot examples boost performance. In simpler reasoning tasks like simple algebra, it gets it correct right away.

Hypothesis 2: `gpt-3.5-turbo` slightly worse than `text-davinci-003` given few-shot examples. We also find evidence for this. `text-davinci-003` with few-shot examples always outperforms `gpt-3.5-turbo`. Although, in simple-algebra, zero shot performance of `gpt-3.5-turbo` is better than `text-davinci-003`, matching what other practitioners have observed in other tasks.

B.3 Task-wise Performance on MiniWoB++

Table 4 shows a task-wise performance breakup on the MiniWoB++ benchmark for various models. We choose a set of 45 tasks that don't require visual reasoning. HeaP Few-shot has competitive performance to prior works like WGE Liu et al. (2018), CC-Net Humphreys et al. (2022), WebN-T5 Gur et al. (2022), WebGUM Furuta et al. (2023) while using $1000x$ or more lower data. See Table 1 for aggregate metrics on success rates and dataset sizes.

The gap in HeaP Few-shot and Flat Few-shot performance is particularly pronounced on tasks involving multiple steps and composing different low-level actions, demonstrating that hierarchy is helpful in solving more complex tasks. Few-shot variants of both `HeaP` and Flat generally leads to improved success rates and

	Task	WGE %suc↑	CC-Net (SL+RL) %suc↑	WebN-T5 %suc↑	WebGUM (HTML) %suc↑	Flat Zero-shot %suc↑	Flat Zero-shot #act↓	Flat Few-shot %suc↑	Flat Few-shot #act↓	HeaP Zero-shot %suc↑	HeaP Zero-shot #act↓	HeaP Few-shot %suc↑	HeaP Few-shot #act↓
simple	click-link	1.00	0.99	1.00	1.00	0.94	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	click-option	1.00	0.99	0.37	1.00	0.76	3.68	1.00	2.62	0.80	2.94	1.00	1.94
	focus-text	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	click-button	1.00	1.00	1.00	0.98	0.98	1.02	1.00	1.00	1.00	1.00	1.00	1.00
	click-button-sequence	0.99	1.00	1.00	1.00	0.96	2.00	1.00	2.00	1.00	2.00	1.00	2.00
	click-dialog	1.00	1.00	1.00	1.00	1.00	1.06	1.00	1.20	1.00	1.28	1.00	1.02
	click-dialog-2	1.00	1.00	0.24	0.34	0.98	1.00	1.00	1.00	0.98	1.00	1.00	1.02
	click-tab	0.55	1.00	0.74	1.00	1.00	1.00	1.00	1.00	0.98	1.00	1.00	1.04
	click-test	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	click-test-2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	enter-text	1.00	1.00	0.89	1.00	1.00	2.50	1.00	2.00	1.00	2.10	1.00	2.00
	focus-text-2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	enter-text-dynamic	1.00	1.00	0.98	1.00	0.98	2.44	1.00	2.00	0.98	2.06	1.00	2.00
	enter-password	0.99	1.00	0.97	1.00	1.00	3.08	1.00	3.00	1.00	3.20	1.00	4.52
	login-user	0.99	1.00	0.82	1.00	0.96	3.42	1.00	3.00	1.00	3.06	1.00	3.00
	click-pie	0.32	0.97	0.51	0.99	1.00	3.00	1.00	3.00	1.00	3.00	1.00	3.00
	enter-date	0.00	1.00	0.00	1.00	1.00	3.00	1.00	2.00	0.00	4.08	1.00	2.00
	grid-coordinate	1.00	1.00	0.49	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	click-widget	0.93	1.00	1.00	1.00	0.94	1.00	0.94	1.00	0.94	1.00	1.00	1.00
complex	email-inbox	0.43	1.00	0.38	0.99	0.40	7.00	0.70	4.50	0.00	3.00	0.90	5.20
	email-inbox-nl-turk	0.77	1.00	0.23	0.92	0.40	7.20	0.50	6.00	0.00	2.90	1.00	4.58
	email-inbox-forward-nl-turk	-	1.00	0.33	1.00	0.30	5.08	0.40	4.80	0.00	3.50	0.90	4.30
	multi-orderings	0.99	1.00	0.88	1.00	0.56	3.60	0.98	3.98	0.76	4.28	1.00	4.00
	choose-date	0.00	0.97	0.00	0.08	0.20	3.00	0.94	3.60	0.20	5.80	1.00	5.40
	click-collapsible-2	0.65	0.98	0.00	0.94	0.60	3.64	0.74	4.14	0.34	3.34	0.80	4.50
	simple-arithmetic	-	0.86	0.00	0.00	0.82	2.12	0.92	2.12	0.54	2.66	1.00	2.00
	click-tab-2	0.64	0.98	0.18	0.94	0.76	5.58	0.88	4.62	0.88	2.84	1.00	2.24
	click-tab-2-hard	-	0.98	0.12	0.57	0.68	3.36	0.88	3.84	0.76	3.06	1.00	2.42
	multi-layouts	0.99	1.00	0.83	1.00	0.42	4.46	0.72	3.94	0.66	4.82	0.94	4.00
	copy-paste	-	0.79	0.00	0.00	0.14	2.14	0.70	3.48	0.98	2.72	1.00	2.00
	click-collapsible	1.00	1.00	0.00	1.00	0.54	1.76	0.68	1.88	0.86	1.88	1.00	2.00
	choose-date-easy	-	0.99	0.03	1.00	0.74	2.74	0.62	2.62	0.20	10.18	1.00	3.10
	copy-paste-2	-	0.63	0.00	0.00	0.54	7.66	0.56	4.00	0.48	3.84	0.96	2.04
	simple-algebra	-	0.75	0.00	0.00	0.14	8.80	0.30	6.78	0.04	4.38	0.74	2.00
	click-checkboxes	0.98	0.98	0.96	1.00	0.40	4.90	0.44	5.94	0.74	3.20	0.90	3.14
	click-checkboxes-transfer	0.64	0.99	0.63	0.99	0.40	4.80	0.40	3.90	0.54	3.20	0.94	2.84
	login-user-popup	-	1.00	0.72	0.97	0.46	6.28	0.46	3.52	0.46	5.82	1.00	4.88
	click-checkboxes-soft	0.51	0.95	0.54	1.00	0.00	7.00	0.00	7.30	0.04	6.94	0.54	5.64
	enter-text-2	-	0.98	0.00	0.00	0.00	2.60	0.40	5.20	0.40	2.00	1.00	2.00
	email-inbox-forward-nl	-	1.00	0.60	1.00	0.10	5.04	0.10	4.58	0.00	3.24	0.74	4.74
	search-engine	0.26	1.00	0.34	0.59	0.38	3.64	0.38	3.16	0.26	4.46	1.00	4.30
	find-word	-	0.88	0.00	0.00	0.22	2.62	0.26	5.18	0.12	2.92	0.98	2.00
	choose-date-medium	-	0.99	0.00	0.57	0.32	2.90	0.20	2.76	0.20	9.26	1.00	3.86
	click-checkboxes-large	0.68	0.71	0.22	0.98	0.00	8.40	0.20	8.40	0.00	7.00	1.00	6.20
	book-flight	0.00	0.87	0.00	0.48	0.00	16.00	0.10	11.10	0.00	13.52	0.90	9.14
	Mean	0.77	0.96	0.56	0.91	0.62	3.70	0.72	3.38	0.60	3.43	0.96	2.89

Table 4: Task-wise performance breakup on MiniWoB++ benchmark on a set of 45 tasks.

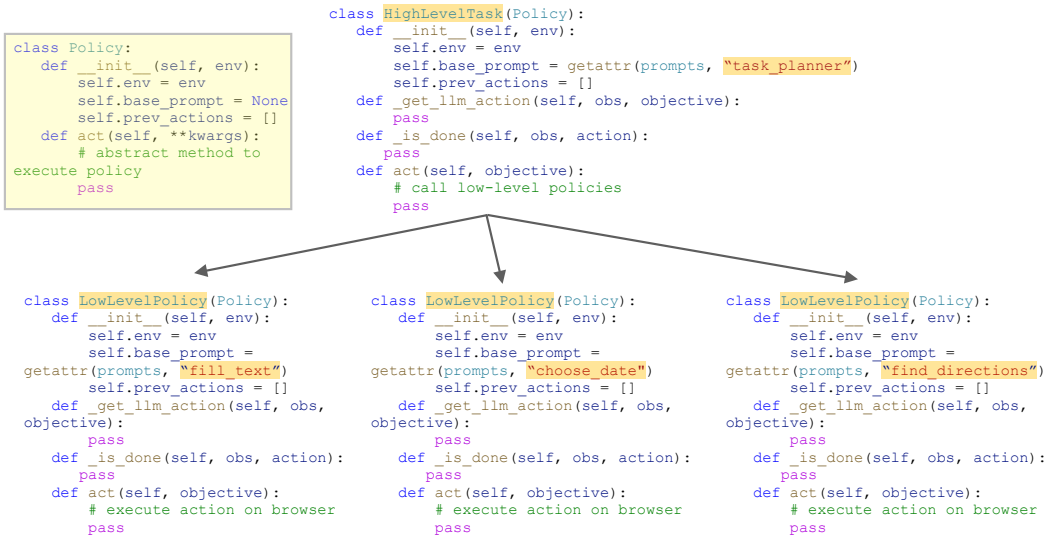

Figure 12: HeaP Planner Policy Code Architecture

efficiency, as evident from the reduced number of actions needed for task completion. This underlines the importance of having a handful of demonstrations to ground the task on the UI.

C Experimental Details

C.1 HEAP Planner Policy Code Architecture

Fig. 12 shows the hierarchical architecture as a code skeleton consisting of a high-level task planner and various low-level policies for executing web actions. The high-level planner is represented by the `HighLevelTask` class, which inherits from a generic `Policy` class. `LowLevelPolicy` also inherits from the generic Policy class, but is instantiated with different base prompts to serve as specialized policies for various sub-tasks.

`HighLevelTask` is responsible for orchestrating the execution of tasks by calling upon different low-level policies as needed to accomplish specific sub-tasks. It does so by instantiating the `LowLevelPolicy` class with different base prompt parameters, like `fill_text`, `choose_date`, `find_directions`, etc. This helps tailor the low-level policy for a specialized sub-task like filling text fields, selecting dates, finding directions. This design allows for the easy addition of new policies to handle complex web tasks, creating a library of skills, hence providing a dynamic and flexible framework for executing a wide array of web tasks.

C.2 Hyper-parameters

We use OpenAI API for model calls, `text-davinci-003` as the default model with a temperature of 0.3 and set the number of calls to 3. For `gpt-3.5-turbo` `gpt-4` we use the same temperature and number of calls. Below are the exact API calls,

Listing 1: Hyper-parameters for different models

```
if (model_type == "gpt-3.5-turbo"):
    response = openai.ChatCompletion.create(
        model=model_type,
        messages=[{"role": "user", "content": prompt}],
        temperature=0.3,
        top_p=1,
        n=3,
        max_tokens=max_tokens
    )
    response = response.choices[0].message.content.strip()

elif (model_type == "gpt-4"):
    response = openai.ChatCompletion.create(
        model=model_type,
        messages=[{"role": "user", "content": prompt}],
        temperature=0.3,
        top_p=1,
        n=3,
        max_tokens=max_tokens
    )
    response = response.choices[0].message.content.strip()

elif (model_type == "text-davinci-003"):
    response = openai.Completion.create(
        model=model_type,
        prompt=prompt,
        temperature=0.3,
        best_of=3,
        n=3,
        max_tokens=max_tokens
    )
    response = response.choices[0].text.strip()
```

D Autolabeling and Prompt Generation

We first collect a dataset of human demonstrations. We then "autolabel" each timestep with the low-level policy that was being executed. We convert the labelled datasets to prompts for both high-level planner and low-level policy. Finally, we augment prompts with chain-of-thought reasoning.

D.1 Collect and Autolabel demonstrations

We collect demonstrations from a human user who is given a conversation context ϕ and performs a sequence of actions in the browser to complete the task. We design a browser plugin to record webpage DOM d and events such as clicks and types. Each demonstration is parsed as text by converting the DOM tree into a list of salient web elements like links, buttons, inputs. Each element is represented as text, e.g. `<button`

`id=18 title="Travelers">1 Adult />`. Action events are parsed as a `CLICK <id>` or `TYPE <id> <val>`. The parsed demonstration dataset is represented as $\mathcal{D} = \{(\phi, s_1, a_1, \ldots, s_T, a_T)\}$.

We then "autolabel" each timestep t with a low-level policy π_t and instruction ψ_t to create a labeled dataset $(\phi, s_1, a_1, \pi_1, \psi_1, \ldots, s_T, a_T, \pi_T, \psi_T)$. Ideally, to label full sequences, we would select the most likely sequence of policies given the entire state-action sequence $\arg\max_{\pi_{1:T}, \psi_{1:T}} \log P(\pi_1, \psi_1 \ldots, \pi_T, \psi_T | s_1, a_1, \ldots, s_T, a_T)$. However, this becomes impractical given demonstrations are too long to fit in to context length. Instead, we relax the problem to predicting the current policy given the previous policy, i.e. $\arg\max_{\pi_t} P(\pi_t | \pi_{t-1}, s_t, a_t)$.

We construct an autolabeling prompt (Appendix H.5) that consists of an overall instruction along with a small set of human-annotated labels as in-context examples. Each in-context example has the following format: `{input: [state` s_t`, action` a_t`, previous policy` π_{t-1}`], output: current policy` π_t`}`.

In-context examples are chosen such that they cover every low-level policy. We note that the auto-labeller can generalize to new tasks not contained in the prompt as long as the constituent low-level policies are covered (e.g. Search Flight → Cancel Booking)

D.2 Autogenerate prompts from labeled demonstrations

Once we have a dataset of labeled demonstrations $\mathcal{D}_{label} = \{(\phi, s_0, a_0, (\pi_0, \psi_0), \ldots, s_T, a_T, (\pi_T, \psi_T))\}$, we can use that to generate base prompts for both the high-level task planner and low-level policies. For the task planner, we concatenate the following input–output pairs as in-context examples in the base prompt: `{input: [context` ϕ`, initial state` s_0`]}` and `{output: sequence of web policy calls` $(\pi_1, \psi_1), (\pi_2, \psi_2) \ldots, (\pi_T, \psi_T)$`}`. For each web policy, we search the dataset $\mathcal{D}_{label}$ for all instances of the policy and create a prompt by concatenating examples of `{input: [instruction` ψ_t`, current state` s_t`, previous actions` $a_{t:t-k}$`]}` and `{output: next action` a_t`}`.

D.3 Augmenting prompts with chain-of-thought reasoning

Empirically, we observed that directly going from input browser content to output web actions can be difficult or error prone for the LLM. Chain-of-thought prompting Wei et al. (2022) has shown to be effective to mitigate this. We additionally include a reasoning section between the input and output sections of the prompt that forces the LLM to generate a series of short sentences that provide justifications for the actions it predictions. We found this to uniformly improve performance for both task and policy prompts (Appendix B). We subsequently added in a simple reason generator prompt (Appendix H.6) that works across both high-level tasks and low-level policies. The prompt takes the current browser content s_t, previous actions $a_{t-1:t-k}$, current action a_t and generates a reason r.

D.4 Autolabeling Examples

We include the following set of in-context examples for autolabeling that allows us to cover the different low-level policies:

1. **MiniWoB:** In-context examples include 1 book flight demo, 1 search engine demo. This lets us cover `CHOOSE_DATE()`, `FILL_TEXT()`, `FIND_AND_CLICK_SEARCH_LINK()`, `FIND_AND_CLICK_TAB_LINK()` policies.
2. **Airline CRM:** In-context examples include 1 demo of search flight, covering `CHOOSE_DATE()`, `FILL_TEXT()` policies.
3. **LiveWeb:** In-context examples include 1 demo of book flight, covering `CHOOSE_DATE()`, `FILL_TEXT()`

E Airline CRM Environment

We constructed a mock Airline CRM environment based on typical airline call-center workflows, and modelled on public airline websites (e.g., jetblue.com, aa.com). The website is built as a standard single-page web application, using a light-weight web development stack. Figs 13, 14 shows sample screenshots from the CRM.

This CRM allows us to test scenarios that are more complex than those in MiniWoB++, and scenarios that cannot practically be run on public websites (e.g., cancelling a flight). The scenarios currently supported are described below.

1. **Find one-way or return flight.** This is a single step task, that requires the source & destination airports and flight dates to be entered in the UI, and the search button to be clicked.

Figure 13: *search flight* screen of the mock airline CRM.

2. **Book flight** This is a four step task:
   (a) Find flight (scenario 1),
   (b) Select desired outward and return flights & click *Confirm*,
   (c) Enter passenger details (*Title, first name, last name, gender, date-of-birth*) & click *Save*,
   (d) Enter payment card details (*card number, expiry, CVC/CVV*) & click *Book flight*.
3. **Find a booking** This is a single step task - enter booking reference & click *Search*.
4. **Cancel a booking** This is a three step task:
   (a) Find booking (scenario 3),
   (b) Click *Cancel*,
   (c) Confirm cancellation by re-entering booking reference & click *Cancel*.
5. **Modify passenger details on an existing booking** This is a three step task:
   (a) Find booking (scenario 3),
   (b) Click *Modify*,
   (c) Change any required passenger details & click *Save*.
6. **Modify flights on an existing booking** This is
   (a) Find booking (scenario 3),
   (b) Click *Modify*,
   (c) Find flight (scenario 1),

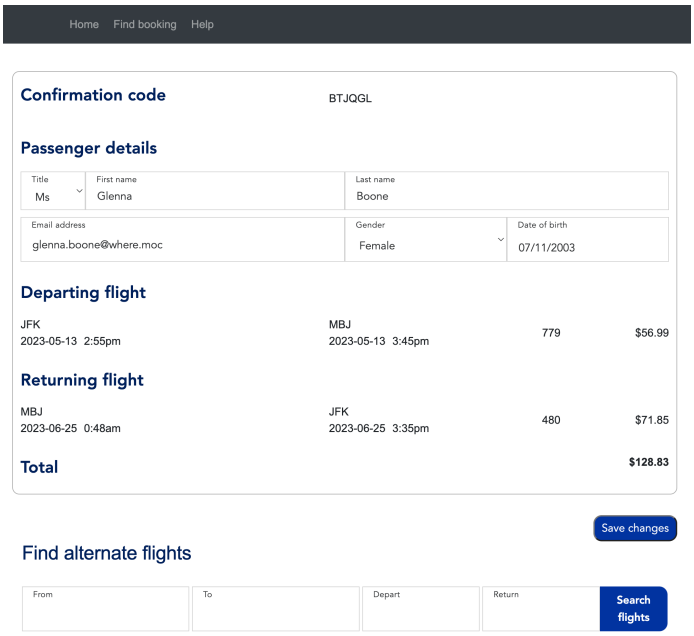

Figure 14: *find and modify booking* screen of the mock airline CRM.

(d) Select desired outward and return flights & click *Save*,

In addition to supporting the above scenarios, the CRM also exposes a few helper APIs that make running and evaluating experiments easier. Two of these are of interest here:

- `https://<base-url>/generate-random-scenario`. This API returns a scenario randomly selected from those listed above, along with all of the data required for completing that scenario on the UI. Shown below is an example of a scenario returned by this API. In addition to the task specific data, the scenario includes a unique id, and a unique URL on which the task can be executed.


```
{
    "scenario": "TASK_FIND_FLIGHT",
    "id": "ylmjd3iuqpdc3gdrvspq",
    "url": "https://<base-url>/?scenario=ylmjd3iuqpdc3gdrvspq",
    "details": {
        "flight": {
            "from": "JFK",
            "to": "FLL",
            "departure": "2023-07-07",
            "return": "2023-09-13",
            "outward-departure-time": "7:01pm",
            "outward-arrival-time": "0:13pm",
            "return-departure-time": "6:00am",
            "return-arrival-time": "8:43am"
        }
    }
}
```


- `https://<base-url>/evaluate?scenario=<id>` This API provides a means of automatically evaluating the *success rate* and *task progress* metrics for scenarios generated by the API above. Specifically, if the UI actions are performed on the unique URL returned by the *generate-random-scenario* API, calling the *evaluate* API afterwards with the scenario id will return the metrics. These metrics are calculated with reference to the gold standard actions required for the given scenario.

F LIVE WEBSITES DATASET

F.1 COLLECTING HUMAN TASK DEMOS

We collected a dataset of human agents searching for flights across three websites: https://www.jetblue.com/, https://www.aa.com/, https://www.united.com/en/us. For each of the websites, a human agent was given a set of 10 task specifications as short conversations, e.g. "Book a flight departing from <>, arriving at <>, leaving on <> and returning on <>". The actions of the human agent, e.g. the click and types, were recorded for each episode. Along with the actions, the raw DOM of the website was also recorded. This is crucial as over the course of the paper, the DOM would change as a result of updates to the website. This also goes to show that simply memorizing actions is unlikely to succeed for these live website tasks.

F.2 PARSING BROWSER CONTENT

Given a data point of raw website DOM (Document Object Model) and action, we make use of playwright https://playwright.dev to parse the DOM and extract a list of web elements. This process involves traversing the DOM tree and extracting salient nodes, i.e. nodes with actionable elements like <input>, <button>, <link>. We also propagate up text attributes from child nodes to the salient parent nodes since the text label for an element may occur inside the subtree. Every web element in the list is represented by an `id` and optionally a `value`. The list of all such elements is concatenated to represent the browser content in natural text form. This is input as the browser observation in the LLM context. Natural language descriptions attached to different web elements helps it generalize across different websites since LLMs have been pre-trained on natural language data. This text form is included under Browser Content in the LLM context. We also convert the demonstrated actions to `CLICK <id>` or `TYPE <id> "TEXT"`.

F.3 AUGMENTING WITH CONVERSATIONS

Each episode of the dataset is represented by the task, the sequence of browser content and actions. Since we wanted to test how models can generalize to different conversations, we used an LLM to automatically generate conversations between a customer and agent that converted the task specification into a short conversational context. We specified the conversations to be diverse and set the temperature to be high (0.6) which naturally resulted in different conversations for every task.

G EVALUATION WITH LLAMA2 MODEL

We also compare `Flat` and `HeaP` with open-source models like `LLaMA2-Chat-{7B, 13B}`. The models are all pre-trained instruction following models without any additional fine-tuning. The prompts used are the same as `Flat Zero-shot` and `HeaP Zero-shot`.

Overall, we find the performance to be lower than `gpt-{3,3.5,4}`. The drop in performance could be due to a number of reasons such as the model size or the training data on which the models are trained. However, we find that `HeaP` still outperforms `Flat` for many tasks, which we discuss below.

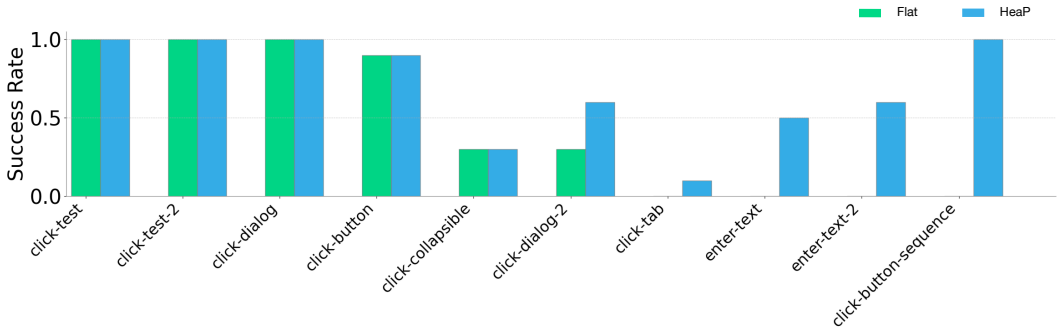

Figure 15: Evaluation with LlaMA2-Chat-13B on a subset of MiniWob++ tasks. Success rate breakdown of 10 tasks averaged over 10 seeds over task.

Figure. 15 shows a comparison of `Flat` and `HeaP` on a subset of MiniWob++ tasks. We see that `Flat` matches `HeaP` for simple CLICK tasks, it fails for slightly more complex tasks that require multiple actions, e.g. multiple CLICK or TYPE.

We also observe that adding few-shot examples did not improve performance. Hence, we chose a subset of MiniWoB++ tasks that don't require being grounded in few-shot examples. We believe that finetuning the model on a larger dataset of demonstrations can help address some of the failure modes that we discuss below and leave that as interesting future work.

We cluster common failure modes for both `Flat`, `HeaP` below:

1. Repeating the same action in multi-step tasks. For tasks that require a sequence of actions, the model tends to get stuck in the first step repeating the same action over and over. It fails to interpret the previous actions or take into account updated states, i.e. changing browser contents. This is more pronounced for `Flat` since `HeaP` has a task planner that can generates a sequence of action and track progress along that sequence.

For example, in the `click-button-sequence` task, for the following input

```
CONTEXT:
Click button ONE, then click button TWO.
BROWSER CONTENT:
<body id=1 val= />
<div id=2 val=wrap />
<div id=3 val=area />
<button id=4 val=ONE />
<button id=5 val=TWO />
PREVIOUS ACTIONS:
CLICK 4
```

the model produces the following output

```
REASON:
To achieve the goal of clicking button ONE and then button TWO, we need to first
click button ONE to activate it, and then click button TWO to activate it.
ACTION:
CLICK 4
```

when the correct output would have been `CLICK 5`, given that the previous actions show that `CLICK 4` has already been executed.

In contrast, HeaP uses the high-level task planner to predict a list of actions "CLICK 4, CLICK 5" which are executed in sequence to solve the task.

Similarly, for the enter-task task, for the following input

```
CONTEXT:
Enter "Agustina" into the text field and press Submit.
BROWSER CONTENT:
<body id=1 val= />
<div id=2 val=wrap />
<div id=3 val=area />
<div id=4 val=form />
<input_text id=5 val=AgustinaAgustina />
<button id=6 val=Submit />
PREVIOUS ACTIONS:
TYPE 5 "Agustina"
TYPE 5 "Agustina"
```

the model gets stuck producing the output TYPE 5 "Agustina",

```
REASON:
To achieve the goal of entering "Agustina" into the text field and
submitting the form, we need to first click on the input field to
focus it and then type "Agustina" into it. Once we have entered the text,
we can click on the Submit button to submit the form.
ACTION:
TYPE 5 "Agustina"
```

and not taking into account changing browser content, i.e. <input_text id=5 val=AgustinaAgustina />

2. Skipping intermediate actions When faced with longer-horizon tasks, Llama2-chat-13B skips over intermediate actions. For example, on a book flight task with browser content

```
... <h2 id=5 val=Book Your One-Way Flight />
<input_text id=7 val=flight-from /> ...
```

the model generates an action like CLICK 5 directly before filling in the empty flight-from input box.

A task like this does require showing examples to the model to ground the task in the UI. Unlike gpt-3.5/4, the model does not seem to pick up on 1-2 few-shot examples given in-context. Finetuning on a larger set of demonstrations should be able to address such issues.

3. Not following the desired action formats. We observe that Llama2-chat-13B often times fails to follow the specified action formats for CLICK and TYPE. This problem occurs for both Flat and HeaP. We apply post-processing to get structured formats (using a gpt-3.5 prompt) to catch some of these cases.

For example, given the following browser content

```
... <div id=2 val=wrap />\n<link id=4 val=justo. />...
```

the models predict CLICK #justo instead of CLICK 4.

Similarly, the model predicts TYPE "Ignacio" instead of TYPE 5 "Ignacio".

H LIBRARY OF PROMPTS

We list out prompts for the four different methods `Flat Zero-shot`, `Flat Few-shot`, `HeaP Zero-shot`, `HeaP Few-shot` that we use to evaluate across all MiniWoB++ tasks in Table 2. We also include the train time auto labeling prompt for that was used to annotate human-collected demonstrations for the search-flight task in the LiveWeb dataset.

H.1 HEAP FEW-SHOT PROMPT

Listing 2: `HeaP Few-shot` Task Prompt

```
You are an AI assistant performing tasks in a web browser on behalf of a human agent. To do
     this, you will be given specific information and allowed to issue certain actions that
     will get you closest to achieving your objective.

You are given:
1. CONTEXT: The goal you need to achieve, either explicitly stated or implied from a
     conversation between a customer (CUS) and agent (REP).
2. CURRENT BROWSER CONTENT: A simplified text description of the current browser content,
     without formatting elements.
3. CURRENT URL: The current webpage URL.
4. PREVIOUS ACTIONS: A list of your past actions.

You can only interact with web elements like links, inputs, and buttons in the browser content
     . You can issue these actions:
- CLICK <element>: Click on the specified element.
- FILL_TEXT <element> "TEXT": Fill a text box with the given text.
- CHOOSE_DATE <element> "DATE": Select a date value from a datepicker.
- FIND_AND_CLICK_SEARCH_LINK "LINK_NUMBER": Find and click a link the LINK_NUMBER from search
     results.
- FIND_AND_CLICK_TAB_LINK "LINK": Find and click a link "LINK" by switching between tabs.
- DONE: Once you finish issuing all actions.

In the <element> field above, you may find either an <id>, a <value>, or both. These are used
     to identify the specific web element in the browser content.

Here are some examples:

EXAMPLE 1:
================================================
CONTEXT:
Find the 9th word in the paragraph, type that into the textbox and press "Submit".
<p id=4 val=Egestas lectus sit commodo turpis ultrices ut malesuada vestibulum />
<input_text id=5 val=answer-input />
<button id=6 val=Submit />
------------------
CURRENT URL:
https://
------------------
PREVIOUS ACTIONS:
------------------
REASONING:
I have to find the 9th word in the paragraph and type it into the textbox and press "Submit".
Looking at the browser content, I see a paragraph with id 4 and a textbox with id 5.
The paragraph contains the following words: 1: Egestas, 2: lectus, 3: sit, 4: commodo, 5:
     turpis, 6:ultrices, 7:ut, 8:malesuada, 9:vestibulum.
The 9th word is "vestibulum".
I have to fill the textbox with "vestibulum"
I have to click Submit
Finally, I have to issue DONE
YOUR ACTION:
TYPE 5 answer-input "vestibulum"
CLICK Submit
DONE
================================================

EXAMPLE 2:
================================================
CONTEXT:
Book the shortest one-way flight from: LEB to: RDG on 12/26/2016.
------------------
CURRENT BROWSER CONTENT:
<body id=1 val= />
<div id=2 val=wrap />
<div id=3 val=area />
<div id=4 val=menu />
<h2 id=5 val=Book Your One-Way Flight />
<div id=6 val= />
```

```
<input_text id=7 val=flight-from />
<div id=8 val= />
<input_text id=9 val=flight-to />
<div id=10 val= />
<div id=11 val=Departure Date />
<div id=12 val= />
<input_text id=13 val=datepicker />
<div id=14 val= />
<button id=15 val=Search />
<div id=16 val= />
<div id=23 val=1 result is available, use up and down arrow keys to navigate. />
<div id=18 val= />
<div id=27 val=1 result is available, use up and down arrow keys to navigate. />
------------------
CURRENT URL:
https://
------------------
PREVIOUS ACTIONS:
------------------
REASONING:
I have to book the shortest flight from LEB to RDG on 12/26/2016
Looking at the browser content, I have to fill in the following fields
I have to fill flight-from with "LEB"
I have to fill flight-to with "RDG"
I have to choose date from the datepicker as 12/26/2016
I have to click Search
Finally, I have to issue DONE
YOUR ACTION:
FILL_TEXT flight-from "LEB"
FILL_TEXT flight-to "RDG"
CHOOSE_DATE datepicker "12/26/2016"
CLICK Search
DONE
================================================

EXAMPLE 3:
================================================
CONTEXT:
Book the shortest one-way flight from: Cincinnati, OH to: LEX on 10/16/2016.
------------------
CURRENT BROWSER CONTENT:
<label id=362 val=Depart: />
<div id=364 val=2:50 PM />
<div id=365 val=Sun Oct 16 2016 />
<div id=366 val=Cincinnati, OH (CVG) />
<label id=369 val=Arrives: />
<div id=371 val=5:32 AM />
<div id=372 val=Mon Oct 17 2016 />
<div id=373 val=LEX />
<label id=376 val=Duration: />
<div id=378 val=14h 42m />
<button id=380 val=Book flight for $379 />
<div id=383 val= />
<label id=406 val=Depart: />
<div id=408 val=11:06 PM />
<div id=409 val=Sun Oct 16 2016 />
<div id=410 val=Cincinnati, OH (CVG) />
<label id=413 val=Arrives: />
<div id=415 val=2:27 AM />
<div id=416 val=Mon Oct 17 2016 />
<div id=417 val=LEX />
<label id=420 val=Duration: />
<div id=422 val=3h 21m />
<button id=424 val=Book flight for $209 />
<div id=425 val= />
<label id=428 val=Depart: />
<div id=430 val=3:23 AM />
<div id=431 val=Sun Oct 16 2016 />
<div id=432 val=Cincinnati, OH (CVG) />
<label id=435 val=Arrives: />
<div id=437 val=5:19 AM />
<div id=438 val=Mon Oct 17 2016 />
<div id=439 val=LEX />
<label id=442 val=Duration: />
<div id=444 val=25h 56m />
<button id=446 val=Book flight for $1197 />
------------------
CURRENT URL:
https://
------------------
PREVIOUS ACTIONS:
```

```
------------------
REASONING:
I have to book the shortest flight from  Cincinnati, OH to: LEX
Looking at the browser content, I see different flight options with flight durations and
     prices
Since I have to book the shortest flight, I should look at duration
The shortest duration is 3h 21m corresponding to id 422
To book this flight, I have to click on button below corresponding to id 424
Finally, I have to issue DONE
YOUR ACTION:
CLICK <button id=424 val=Book flight for $209 />
DONE
==================================================

The current context, url, and browser content follow. Reply with your reasoning and next
     action to the browser.

==================================================
CONTEXT:
{context}
------------------
CURRENT BROWSER CONTENT:
{browser_content}
------------------
CURRENT URL:
{url}
------------------
PREVIOUS ACTIONS:
{previous_actions}
------------------
REASONING:
```

Listing 3: `HeaP Few-shot` FILL_TEXT() Policy Prompt

```
You are an AI assistant performing tasks in a web browser on behalf of a human agent. Your
     goal is to fill a text box.

You are given:
1. CONTEXT: An instruction like FILL_TEXT <element> <text>, where <element> corresponds to a
     web element and <text> corresponds to text to fill.
2. CURRENT BROWSER CONTENT: A simplified text description of the current browser content,
     without formatting elements.
3. CURRENT URL: The current webpage URL.
4. PREVIOUS ACTIONS: A list of your past actions.

You can only interact with web elements like links, inputs, and buttons in the browser content
     . You can issue these actions:
- TYPE <element> "TEXT" - Type "TEXT" into the input element.
- CLICK <element> - Click on the specified element.
- DONE - Once you finish issuing all actions.

In the <element> field above, you may find either an <id>, a <value>, or both. These are used
     to identify the specific web element in the browser content.

Before selecting an action, provide reasoning.

Here are some examples:

Example 1:
==================================================
CONTEXT:
FILL_TEXT 10 "Bowen"
------------------
CURRENT BROWSER CONTENT:
<body id=1 val= />
<div id=2 val=wrap />
<div id=3 val=area />
<div id=4 val=Movie Search />
<table id=5 val= />
<tbody id=6 val= />
<th id=8 val=Director />
<input_text id=10 val= />
<th id=12 val=Genre />
<input_text id=14 val= />
<th id=16 val=Year />
<input_text id=18 val= />
<div id=19 val=Submit />
------------------
CURRENT URL:
https://
```

```
------------------
PREVIOUS ACTIONS:
------------------
REASONING:
I have no previous actions.
I have to first type "Bowen" in id 10
YOUR ACTION:
TYPE 10 "Bowen"
==================================================

Example 2:
==================================================
CONTEXT:
FILL_TEXT 10 "Bowen"
------------------
CURRENT BROWSER CONTENT:
<body id=1 val= />
<div id=2 val=wrap />
<div id=3 val=area />
<div id=4 val=Movie Search />
<table id=5 val= />
<tbody id=6 val= />
<th id=8 val=Director />
<input_text id=10 val= Bowen/>
<th id=12 val=Genre />
<input_text id=14 val= />
<th id=16 val=Year />
<input_text id=18 val= />
<div id=19 val=Submit />
------------------
CURRENT URL:
https://
------------------
PREVIOUS ACTIONS:
TYPE 10 "Bowen"
------------------
REASONING:
I have already typed "Bowen" in id 10
There seems to be no dropdown text
I am done filling text
YOUR ACTION:
DONE
==================================================

Example 3:
==================================================
CONTEXT:
FILL_TEXT flight-from "LEB"
------------------
CURRENT BROWSER CONTENT:
<body id=1 val= />
<div id=2 val=wrap />
<div id=3 val=area />
<div id=4 val=menu />
<h2 id=5 val=Book Your One-Way Flight />
<div id=6 val= />
<input_text id=7 val=flight-from />
<div id=8 val= />
<input_text id=9 val=flight-to />
<div id=10 val= />
<div id=11 val=Departure Date />
<div id=12 val= />
<input_text id=13 val=datepicker />
<div id=14 val= />
<button id=15 val=Search />
<div id=16 val= />
<div id=17 val= />
------------------
CURRENT URL:
https://
------------------
PREVIOUS ACTIONS:
------------------
REASONING:
I have no previous actions.
I have to first type "LEB" in the field flight-from corresponding to id 7
YOUR ACTION:
TYPE 7 flight-from "LEB"
==================================================

Example 4:
```

```
================================================
CONTEXT:
FILL_TEXT flight-from "LEB"
------------------
CURRENT BROWSER CONTENT:
<body id=1 val= />
<div id=2 val=wrap />
<div id=3 val=area />
<div id=4 val=menu />
<h2 id=5 val=Book Your One-Way Flight />
<div id=6 val= />
<input_text id=7 val=flight-from />
<div id=8 val= />
<input_text id=9 val=flight-to />
<div id=10 val= />
<div id=11 val=Departure Date />
<div id=12 val= />
<input_text id=13 val=datepicker />
<div id=14 val= />
<button id=15 val=Search />
<ul id=18 val=ui-id-1 />
<li id=19 val= />
<div id=20 val=Hanover, NH (HNV) />
<li id=21 val= />
<div id=22 val=Lebanon, NH (LEB) />
<li id=23 val= />
<div id=24 val=White River, VT (WHR) />
<div id=16 val= />
<div id=25 val=3 results are available, use up and down arrow keys to navigate. />
<div id=17 val= />
------------------
CURRENT URL:
https://
------------------
PREVIOUS ACTIONS:
TYPE 7 flight-from "LEB"
------------------
REASONING:
I have already typed in "LEB" in id 7
There is a corresponding dropdown text in "Lebanon, NH (LEB)" in id 22
I have to click on id 22 Lebanon, NH (LEB)
YOUR ACTION:
CLICK 22 Lebanon, NH (LEB)
================================================

Example 5:
================================================
CONTEXT:
FILL_TEXT flight-from "LEB"
------------------
CURRENT BROWSER CONTENT:
<body id=1 val= />
<div id=2 val=wrap />
<div id=3 val=area />
<div id=4 val=menu />
<h2 id=5 val=Book Your One-Way Flight />
<div id=6 val= />
<input_text id=7 val=flight-from />
<div id=8 val= />
<input_text id=9 val=flight-to />
<div id=10 val= />
<div id=11 val=Departure Date />
<div id=12 val= />
<input_text id=13 val=datepicker />
<div id=14 val= />
<button id=15 val=Search />
<div id=16 val= />
<div id=25 val=3 results are available, use up and down arrow keys to navigate. />
<div id=17 val= />
------------------
CURRENT URL:
https://
------------------
PREVIOUS ACTIONS:
TYPE 7 flight-from "LEB"
CLICK 22 Lebanon, NH (LEB)
------------------
REASONING:
I have already typed in "LEB" in id 7
I have clicked on the dropdown text in id 22 Lebanon, NH (LEB)
I am done filling text
```

```
YOUR ACTION:
DONE
==================================================

The current context, url, and browser content follow. Reply with your reasoning and next
     action to the browser.

==================================================
CONTEXT:
{context}
------------------
CURRENT BROWSER CONTENT:
{browser_content}
------------------
CURRENT URL:
{url}
------------------
PREVIOUS ACTIONS:
{previous_actions}
------------------
REASONING:
```

Listing 4: `HeaP Few-shot` CHOOSE_DATE() Policy Prompt

```
You are an AI assistant performing tasks in a web browser on behalf of a human agent. Your
     goal is to choose a date from a datepicker by navigating to the right month.

You are given:
1. CONTEXT: An instruction like CHOOSE_DATE <element> <date>, where <element> corresponds to a
      web element and <date> corresponds to date to choose.
2. CURRENT BROWSER CONTENT: A simplified text description of the current browser content,
     without formatting elements.
3. CURRENT URL: The current webpage URL.
4. PREVIOUS ACTIONS: A list of your past actions.

You can only interact with web elements like links, inputs, and buttons in the browser content
     . You can issue these actions:
- CLICK <element> - Click on the specified element.
- DONE - Once you finish issuing all actions.

In the <element> field above, you may find either an <id>, a <value>, or both. These are used
     to identify the specific web element in the browser content.

Before selecting an action, provide reasoning.

Here are some examples:

Example 1:
==================================================
CONTEXT:
CHOOSE_DATE datepicker "11/03/2016"
------------------
CURRENT BROWSER CONTENT:
<body id=1 val= />
<div id=2 val=wrap />
<div id=3 val=area />
<p id=4 val= />
<t id=-1 val=Date: />
<input_text id=5 val=datepicker />
<button id=6 val=Submit />
------------------
CURRENT URL:
https://
------------------
PREVIOUS ACTIONS:
------------------
REASONING:
I have no previous actions.
I have to first click on a datepicker.
YOUR ACTION:
CLICK 5 datepicker
==================================================

Example 2:
==================================================
CONTEXT:
CHOOSE_DATE datepicker "11/03/2016"
------------------
CURRENT BROWSER CONTENT:
<div id=8 val= />
```

```
<a id=9 val= />
<span id=10 val=Prev />
<a id=11 val= />
<div id=13 val= />
<span id=14 val=December />
<span id=15 val=2016 />
<a id=40 val=12/1/2016 />
<a id=42 val=12/2/2016 />
<a id=44 val=12/3/2016 />
<a id=47 val=12/4/2016 />
<a id=49 val=12/5/2016 />
<a id=51 val=12/6/2016 />
<a id=53 val=12/7/2016 />
<a id=55 val=12/8/2016 />
<a id=57 val=12/9/2016 />
<a id=59 val=12/10/2016 />
<a id=62 val=12/11/2016 />
<a id=64 val=12/12/2016 />
<a id=66 val=12/13/2016 />
<a id=68 val=12/14/2016 />
------------------
PREVIOUS ACTIONS:
CLICK 5 datepicker
------------------
REASONING:
I have already clicked on datepicker.
Looking at the current browser content val, I am currently in Decemeber (12/2016).
I have to go to November (11/2016).
Since 11 < 12, I have to click on Prev
YOUR ACTION:
CLICK 10 Prev
==================================================

Example 3:
==================================================
CONTEXT:
CHOOSE_DATE datepicker "11/03/2016"
------------------
CURRENT URL:
https://
------------------
CURRENT BROWSER CONTENT:
<tbody id=33 val= />
<a id=40 val=11/1/2016 />
<a id=42 val=11/2/2016 />
<a id=44 val=11/3/2016 />
<a id=47 val=11/4/2016 />
<a id=49 val=11/5/2016 />
<a id=51 val=11/6/2016 />
<a id=53 val=11/7/2016 />
<a id=55 val=11/8/2016 />
<a id=57 val=11/9/2016 />
<a id=59 val=11/10/2016 />
<a id=62 val=11/11/2016 />
<a id=64 val=11/12/2016 />
<a id=66 val=11/13/2016 />
<a id=68 val=11/14/2016 />
<a id=70 val=11/15/2016 />
------------------
PREVIOUS ACTIONS:
CLICK 5 datepicker
CLICK 10 Prev
------------------
REASONING:
I have already clicked on datepicker.
Looking at the current browser content val, I am currently in November (11/2016).
I have to go to November (11/2016).
Since 11 = 11, I am in the correct month.
I have to click on the id corresponding to 11/3/2016
YOUR ACTION:
CLICK 44 11/3/2016
==================================================

Example 4:
==================================================
CONTEXT:
CHOOSE_DATE datepicker "11/03/2016"
------------------
CURRENT URL:
https://
------------------
```

```
CURRENT BROWSER CONTENT:
<body id=1 val= />
<div id=2 val=wrap />
<div id=3 val=area />
<p id=4 val= />
<t id=-1 val=Date: />
<input_text id=5 val=datepicker />
<button id=6 val=Submit />
------------------
PREVIOUS ACTIONS:
CLICK 5 datepicker
CLICK 10 prev
CLICK 44 11/3/2016
------------------
REASONING:
I am done selecting the dates.
YOUR ACTION:
DONE
==================================================


The current context, url, and browser content follow. Reply with your reasoning and next
     action to the browser.

==================================================
CONTEXT:
{context}
------------------
CURRENT BROWSER CONTENT:
{browser_content}
------------------
CURRENT URL:
{url}
------------------
PREVIOUS ACTIONS:
{previous_actions}
------------------
REASONING:
```

H.2 FLAT FEW-SHOT PROMPT

Listing 5: `Flat Few-shot` Prompt

```
You are an AI assistant performing tasks in a web browser on behalf of a human agent. To do
     this, you will be given specific information and allowed one action at a time that get
     you closest to achieving your objective.

You are given:
1. CONTEXT: The goal you need to achieve, either explicitly stated or implied from a
     conversation between a customer (CUS) and agent (REP).
2. CURRENT BROWSER CONTENT: A simplified text description of the current browser content,
     without formatting elements.
3. CURRENT URL: The current webpage URL.
4. PREVIOUS ACTIONS: A list of your past actions.

You can only interact with web elements like links, inputs, and buttons in the browser content
     . You can issue any one of these actions:
- CLICK <id> - Click on the specified element.
- TYPE <id> "TEXT" - Type "TEXT" into the input element.
- DONE - Once you finish issuing all actions.

Example: CLICK 7, TYPE 11 "New York"

Before selecting an action, provide reasoning.

Here are some examples:

Example 1:
==================================================
CONTEXT:
Book the shortest one-way flight from: LEB to: RDG on 12/26/2016.
------------------
CURRENT BROWSER CONTENT:
<body id=1 val= />
<div id=2 val=wrap />
<div id=3 val=area />
<div id=4 val=menu />
<h2 id=5 val=Book Your One-Way Flight />
```

```
<div id=6 val= />
<input_text id=7 val=flight-from />
<div id=8 val= />
<input_text id=9 val=flight-to />
<div id=10 val= />
------------------
CURRENT URL:
https://
------------------
PREVIOUS ACTIONS:
------------------
REASONING:
I have no previous actions.
I have to first type "LEB" in the field flight-from corresponding to id 7
YOUR ACTION:
TYPE 7 flight-from "LEB"
==================================================

Example 2:
==================================================
CONTEXT:
Book the shortest one-way flight from: LEB to: RDG on 12/26/2016.
------------------
CURRENT BROWSER CONTENT:
<body id=1 val= />
<div id=2 val=wrap />
<div id=3 val=area />
<div id=4 val=menu />
<h2 id=5 val=Book Your One-Way Flight />
<div id=6 val= />
<input_text id=7 val=flight-from />
<div id=8 val= />
<div id=14 val= />
<button id=15 val=Search />
<ul id=18 val=ui-id-1 />
<li id=19 val= />
<div id=20 val=Hanover, NH (HNV) />
<li id=21 val= />
<div id=22 val=Lebanon, NH (LEB) />
<li id=23 val= />
<div id=24 val=White River, VT (WHR) />
<div id=16 val= />
<div id=25 val=3 results are available, use up and down arrow keys to navigate. />
<div id=17 val= />
------------------
CURRENT URL:
https://
------------------
PREVIOUS ACTIONS:
TYPE 7 flight-from "LEB"
------------------
REASONING:
I have already typed in "LEB" in the field "flight-from"
There is a corresponding dropdown text in "Lebanon, NH (LEB)" in id 22
I have to click on id 22 Lebanon, NH (LEB)
YOUR ACTION:
CLICK 22 Lebanon, NH (LEB)
==================================================

Example 3:
==================================================
CONTEXT:
Book the shortest one-way flight from: LEB to: MOT on 12/26/2016.
------------------
CURRENT BROWSER CONTENT:
<body id=1 val= />
<div id=2 val=wrap />
<div id=3 val=area />
<div id=4 val=menu />
<h2 id=5 val=Book Your One-Way Flight />
<div id=6 val= />
<input_text id=7 val=flight-from />
<div id=8 val= />
<input_text id=9 val=flight-to />
<div id=10 val= />
<div id=11 val=Departure Date />
<div id=12 val= />
<input_text id=13 val=datepicker />
------------------
CURRENT URL:
https://
```

```
------------------
PREVIOUS ACTIONS:
TYPE 7 flight-from "LEB"
CLICK 22 Lebanon, NH (LEB)
TYPE 9 flight-to "MOT"
CLICK 22 Montana, MOT
------------------
REASONING:
I have already typed in "LEB" in the field "flight-from"
I have already clicked on id 22 Lebanon, NH (LEB)
I have already typed in "MOT" in the field "flight-to"
I have already clicked on id 22 Montana, MOT
I have to now choose a date by clicking on the datepicker
YOUR ACTION:
CLICK 13 datepicker
================================================

Example 4:
================================================
CONTEXT:
Book the shortest one-way flight from: LEB to: MOT on 12/26/2016.
------------------
CURRENT BROWSER CONTENT:
<a id=47 val=12/21/2016 />
<a id=49 val=12/22/2016 />
<a id=51 val=12/23/2016 />
<a id=53 val=12/24/2016 />
<a id=55 val=12/25/2016 />
<a id=57 val=12/26/2016 />
<a id=59 val=12/27/2016 />
<a id=62 val=12/28/2016 />
<a id=64 val=12/29/2016 />
<a id=66 val=12/30/2016 />
<a id=68 val=12/31/2016 />
------------------
CURRENT URL:
https://
------------------
PREVIOUS ACTIONS:
TYPE 7 flight-from "LEB"
CLICK 22 Lebanon, NH (LEB)
TYPE 9 flight-to "MOT"
CLICK 22 Montana, MOT
CLICK 13 datepicker
------------------
REASONING:
I have already typed in "LEB" in the field "flight-from"
I have already clicked on id 22 Lebanon, NH (LEB)
I have already typed in "MOT" in the field "flight-to"
I have already clicked on id 22 Montana, MOT
I have already clicked on a datepicker
I have to now choose the correct date 12/26/2016.
YOUR ACTION:
CLICK 57 12/26/2016
================================================

The current context, url, and browser content follow. Reply with your reasoning and next
     action to the browser.

================================================
CONTEXT:
{context}
------------------
CURRENT BROWSER CONTENT:
{browser_content}
------------------
CURRENT URL:
{url}
------------------
PREVIOUS ACTIONS:
{previous_actions}
------------------
REASONING:
```

H.3 FLAT ZERO-SHOT PROMPT

Listing 6: `Flat Zero-shot` Prompt

```
You are an AI assistant performing tasks in a web browser on behalf of a human agent. To do
     this, you will be given specific information and allowed one action at a time that get
     you closest to achieving your objective.

You are given:
1. CONTEXT: The goal you need to achieve, either explicitly stated or implied from a
     conversation between a customer (CUS) and agent (REP).
2. CURRENT BROWSER CONTENT: A simplified text description of the current browser content,
     without formatting elements.
3. CURRENT URL: The current webpage URL.
4. PREVIOUS ACTIONS: A list of your past actions.

You can only interact with web elements like links, inputs, and buttons in the browser content
     . You can issue any one of these actions:
- CLICK <id> - Click on the specified element.
- TYPE <id> "TEXT" - Type "TEXT" into the input element.
- DONE - Once you finish issuing all actions.

Example: CLICK 7, TYPE 11 "New York"

Before selecting an action, provide reasoning.

Here is the template:

================================================
CONTEXT:
(The instruction)
------------------
CURRENT BROWSER CONTENT:
(A list of web ids and links)
------------------
CURRENT URL:
(The current url)
------------------
PREVIOUS ACTIONS:
(A list of previous actions)
------------------
REASONING:
(A rationale for selecting the action below)
YOUR ACTION:
(A single action, e.g.)
CLICK <id>
================================================

The current context, url, and browser content follow. Reply with your reasoning and next
     action to the browser.

================================================
CONTEXT:
{context}
------------------
CURRENT BROWSER CONTENT:
{browser_content}
------------------
CURRENT URL:
{url}
------------------
PREVIOUS ACTIONS:
{previous_actions}
------------------
REASONING:
```

H.4 HEAP ZERO-SHOT PROMPT

Listing 7: `HeaP Zero-shot` Task Prompt

```
You are an AI assistant performing tasks in a web browser on behalf of a human agent. To do
     this, you will be given specific information and allowed to issue certain actions that
     will get you closest to achieving your objective.

You are given:
1. CONTEXT: The goal you need to achieve, either explicitly stated or implied from a
     conversation between a customer (CUS) and agent (REP).
2. CURRENT BROWSER CONTENT: A simplified text description of the current browser content,
     without formatting elements.
3. CURRENT URL: The current webpage URL.
4. PREVIOUS ACTIONS: A list of your past actions.
```

```
You can only interact with web elements like links, inputs, and buttons in the browser content
     . You can issue these actions:
- CLICK <id>: Click on the specified element.
- FILL_TEXT <id> "TEXT": Fill a text box with the given text.
- CHOOSE_DATE <id> "DATE": Select a date value from a datepicker.
- FIND_AND_CLICK_SEARCH_LINK "LINK_NUMBER": Find and click a link the LINK_NUMBER from search
     results.
- FIND_AND_CLICK_TAB_LINK "LINK": Find and click a link "LINK" by switching between tabs.
- DONE: Once you finish issuing all actions.

Example: CLICK 7, FILL_TEXT 11 "New York"

Before selecting an action, provide reasoning.

Here is a template:

==================================================
CONTEXT:
(Some instruction or conversations)
------------------
CURRENT BROWSER CONTENT:
(A list of web ids and links)
------------------
CURRENT URL:
(The current url)
------------------
PREVIOUS ACTIONS:
(A list of previous actions)
------------------
REASONING:
(A rationale for selecting the actions blow()
YOUR ACTION:
(A list of actions to do in this webpage, one action per line, e.g.)
CLICK <id>
FILL_TEXT <id> "TEXT"
DONE
==================================================

The current context, url, and browser content follow. Reply with your reasoning and next
     action to the browser.

==================================================
CONTEXT:
{context}
------------------
CURRENT BROWSER CONTENT:
{browser_content}
------------------
CURRENT URL:
{url}
------------------
PREVIOUS ACTIONS:
{previous_actions}
------------------
REASONING:
```

Listing 8: `HeaP Zero-shot` FILL_TEXT() Policy Prompt

```
You are an AI assistant performing tasks in a web browser on behalf of a human agent. Your
     goal is to fill a text box.

You are given:
1. CONTEXT: An instruction like FILL_TEXT <id> <text>, where <id> corresponds to a web element
      and <text> corresponds to text to fill.
2. CURRENT BROWSER CONTENT: A simplified text description of the current browser content,
     without formatting elements.
3. CURRENT URL: The current webpage URL.
4. PREVIOUS ACTIONS: A list of your past actions.

You can only interact with web elements like links, inputs, and buttons in the browser content
     . You can issue these actions:
- TYPE <id> "TEXT" - Type "TEXT" into the input element.
- CLICK <id> - Click on the specified element.
- DONE - Once you finish issuing all actions.

Before selecting an action, provide reasoning.

Here is the template:

==================================================
```

```
CONTEXT:
(The instruction)
------------------
CURRENT BROWSER CONTENT:
(A list of web ids and links)
------------------
CURRENT URL:
(The current url)
------------------
PREVIOUS ACTIONS:
(A list of previous actions)
------------------
REASONING:
(A rationale for selecting the action blow()
YOUR ACTION:
(A single action)
================================================

The current context, url, and browser content follow. Reply with your reasoning and next
     action to the browser.

================================================
CONTEXT:
{context}
------------------
CURRENT BROWSER CONTENT:
{browser_content}
------------------
CURRENT URL:
{url}
------------------
PREVIOUS ACTIONS:
{previous_actions}
------------------
REASONING:
```

Listing 9: `HeaP Zero-shot` CHOOSE_DATE() Policy Prompt

```
You are an AI assistant performing tasks in a web browser on behalf of a human agent. Your
     goal is to choose a date from a datepicker by navigating to the right month.

You are given:
1. CONTEXT: An instruction like CHOOSE_DATE <element> <date>, where <element> corresponds to a
      web element and <date> corresponds to date to choose.
2. CURRENT BROWSER CONTENT: A simplified text description of the current browser content,
     without formatting elements.
3. CURRENT URL: The current webpage URL.
4. PREVIOUS ACTIONS: A list of your past actions.

You can only interact with web elements like links, inputs, and buttons in the browser content
     . You can issue these actions:
- CLICK <element> - Click on the specified element.
- DONE - Once you finish issuing all actions.

In the <element> field above, you may find either an <id>, a <value>, or both. These are used
     to identify the specific web element in the browser content.

Before selecting an action, provide reasoning.

Here is the template:

================================================
CONTEXT:
(The instruction)
------------------
CURRENT BROWSER CONTENT:
(A list of web ids and links)
------------------
CURRENT URL:
(The current url)
------------------
PREVIOUS ACTIONS:
(A list of previous actions)
------------------
REASONING:
(A rationale for selecting the action blow()
YOUR ACTION:
(A single action)
================================================
```

```
The current context, url, and browser content follow. Reply with your reasoning and next
    action to the browser.

================================================
CONTEXT:
{context}
------------------
CURRENT BROWSER CONTENT:
{browser_content}
------------------
CURRENT URL:
{url}
------------------
PREVIOUS ACTIONS:
{previous_actions}
------------------
REASONING:
```

H.5 AUTO-LABELING PROMPT

Listing 10: `Autolabel` Prompt

```
You are an AI assistant labeling web actions as one of several skills.

You are given:
- a CONTEXT that contains the objective you are trying to achieve. The objective may be
    explicitly stated or be implicit in a conversation between a customer (CUS) and agent (
    REP).
- A simplified text description of the browser content and the current webpage URL. The
    browser content is highly simplified with all formatting elements removed.
- The current web action being performed
- The previous label that was assigned

You can only assign labels from the following types
- FILL_TEXT "desription" "TEXT" - fill a text box
- CHOOSE_DATE "desription" "DATE" - choose a date from a date grid
- CLICK "desription" - click a button

================================================
CONTEXT:
Book a flight from Boston to Chicago departing on Dec 1, 2023 and returning on Dec 12, 2023
------------------
CURRENT BROWSER CONTENT:
<text id=10>JetBlue Home</text>
<text id=11>Where on Earth are you headed?</text>
<button id=12>Flights</button>
<button id=13>Flights + Hotel</button>
<button id=14>Flights + Cruise</button>
<button id=15>Cars</button>
<button id=16>Stays</button>
<button id=17>Roundtrip</button>
<button id=18 title="Travelers">1 Adult</button>
<text id=19>Use TrueBlue points</text>
<text id=20>From</text>
<input id=21 text>Washington D.C. area (WAS)</input>
<button id=22>Reverse origin and destination city or airport</button>
<text id=23>To</text>
<input id=24 text/>
<text id=25>Depart</text>
<input id=26 Depart Press DOWN ARROW key to select available dates/>
<text id=27>Return</text>
<input id=28 Return Press DOWN ARROW key to select available dates/>
<button id=29>Search flights</button>
------------------
CURRENT ACTION:
TYPE 21 Boston
PREVIOUS LABEL:
CURRENT LABEL:
FILL_TEXT From "Boston"
================================================

================================================
CONTEXT:
Book a flight from Boston to Chicago departing on Dec 1, 2023 and returning on Dec 12, 2023
------------------
CURRENT BROWSER CONTENT:
<text id=20>From</text>
<input id=21 text>Boston</input>
```

```
<text id=22>Boston</text>
<text id=23>area</text>
<text id=24>Boston</text>
<text id=25>, MA (BOS)</text>
<text id=26>Providence, RI (PVD)</text>
<text id=27>Worcester, MA (ORH)</text>
<text id=28>Browse by regions</text>
<text id=29>Browse by regions</text>
<text id=30>Mint Service</text>
<text id=31>Partner Airline</text>
<button id=32>Reverse origin and destination city or airport</button>
<text id=33>To</text>
<input id=34 text/>
<text id=35>Depart</text>
<input id=36 Depart Press DOWN ARROW key to select available dates/>
<text id=37>Return</text>
<input id=38 Return Press DOWN ARROW key to select available dates/>
<button id=39>Search flights</button>
------------------
CURRENT ACTION:
CLICK 24
PREVIOUS LABEL:
FILL_TEXT From "Boston"
CURRENT LABEL:
FILL_TEXT From "Boston"
================================================

================================================
CONTEXT:
Book a flight from Boston to Chicago departing on Dec 1, 2023 and returning on Dec 12, 2023
------------------
CURRENT BROWSER CONTENT:
<button id=13>Flights + Hotel</button>
<button id=14>Flights + Cruise</button>
<button id=15>Cars</button>
<button id=16>Stays</button>
<button id=17>Roundtrip</button>
<button id=18 title="Travelers">1 Adult</button>
<text id=19>Use TrueBlue points</text>
<text id=20>From</text>
<input id=21 text>Boston, MA (BOS)</input>
<button id=22>Reverse origin and destination city or airport</button>
<text id=23>To</text>
<input id=24 text/>
<text id=25>Browse by regions</text>
<text id=26>Mint Service</text>
<text id=27>Partner Airline</text>
<text id=28>Depart</text>
<input id=29 Depart Press DOWN ARROW key to select available dates/>
<text id=30>Return</text>
<input id=31 Return Press DOWN ARROW key to select available dates/>
<button id=32>Search flights</button>
------------------
CURRENT ACTION:
TYPE 24 Chicago
PREVIOUS LABEL:
FILL_TEXT From "Boston"
CURRENT LABEL:
FILL_TEXT To "Chicago"
================================================

================================================
CONTEXT:
Book a flight from Boston to Chicago departing on Dec 1, 2023 and returning on Dec 12, 2023
------------------
CURRENT BROWSER CONTENT:
<text id=20>From</text>
<input id=21 text>Boston, MA (BOS)</input>
<button id=22>Reverse origin and destination city or airport</button>
<text id=23>To</text>
<input id=24 text>Chicago</input>
<text id=25>Chicago</text>
<text id=26>, IL (ORD)</text>
<text id=27>Browse by regions</text>
<text id=28>Browse by regions</text>
<text id=29>Mint Service</text>
<text id=30>Partner Airline</text>
<text id=31>Depart</text>
<input id=32 Depart Press DOWN ARROW key to select available dates/>
<text id=33>Return</text>
<input id=34 Return Press DOWN ARROW key to select available dates/>
```

```
<button id=35>Search flights</button>
------------------
CURRENT ACTION:
CLICK 25
PREVIOUS LABEL:
FILL_TEXT To "Chicago"
CURRENT LABEL:
FILL_TEXT To "Chicago"
================================================

================================================
CONTEXT:
Book a flight from Boston to Chicago departing on Dec 1, 2023 and returning on Dec 12, 2023
------------------
CURRENT BROWSER CONTENT:
<button id=17>Roundtrip</button>
<button id=18 title="Travelers">1 Adult</button>
<text id=19>Use TrueBlue points</text>
<text id=20>From</text>
<input id=21 text>Boston, MA (BOS)</input>
<button id=22>Reverse origin and destination city or airport</button>
<text id=23>To</text>
<input id=24 text>Chicago, IL (ORD)</input>
<text id=25>Depart</text>
<input id=26 Depart Press DOWN ARROW key to select available dates/>
<text id=27>Return</text>
<input id=28 Return Press DOWN ARROW key to select available dates/>
<button id=29 aria-label="Previous Month"/>
<text id=30>April 2023</text>
<text id=31>S</text>
<text id=32>M</text>
<text id=33>T</text>
<text id=34>W</text>
<text id=35>T</text>
<text id=36>F</text>
<text id=37>S</text>
<button id=38 aria-label="depart Saturday, April 1, 2023">1</button>
<button id=39 aria-label="depart Sunday, April 2, 2023">2</button>
<button id=40 aria-label="depart Monday, April 3, 2023">3</button>
<button id=41 aria-label="depart Tuesday, April 4, 2023">4</button>
<button id=42 aria-label="depart Wednesday, April 5, 2023">5</button>
<button id=43 aria-label="depart Thursday, April 6, 2023">6</button>
<button id=44 aria-label="depart Friday, April 7, 2023">7</button>
------------------
CURRENT ACTION:
TYPE 26 12/01/2023
PREVIOUS LABEL:
FILL_TEXT To "Chicago"
CURRENT LABEL:
CHOOSE_DATE Depart 12/01/2023
================================================

================================================
CONTEXT:
Book a flight from Boston to Chicago departing on Dec 1, 2023 and returning on Dec 12, 2023
------------------
CURRENT BROWSER CONTENT:
<button id=18 title="Travelers">1 Adult</button>
<text id=19>Use TrueBlue points</text>
<text id=20>From</text>
<input id=21 text>Boston, MA (BOS)</input>
<button id=22>Reverse origin and destination city or airport</button>
<text id=23>To</text>
<input id=24 text>Chicago, IL (ORD)</input>
<text id=25>Depart</text>
<input id=26 Depart Press DOWN ARROW key to select available dates>12/01/2023</input>
<text id=27>Return</text>
<input id=28 Return Press DOWN ARROW key to select available dates/>
<button id=29 aria-label="Previous Month"/>
<text id=30>April 2023</text>
<text id=31>S</text>
<text id=32>M</text>
<text id=33>T</text>
<text id=34>W</text>
<text id=35>T</text>
<text id=36>F</text>
<text id=37>S</text>
<button id=38 aria-label="depart Saturday, April 1, 2023">1</button>
<button id=39 aria-label="depart Sunday, April 2, 2023">2</button>
------------------
CURRENT ACTION:
```

```
TYPE 28 12/12/2023
PREVIOUS LABEL:
CHOOSE_DATE Depart 12/01/2023
CURRENT LABEL:
CHOOSE_DATE Return 12/12/2023
================================================

================================================
CONTEXT:
Book a flight from Boston to Chicago departing on Dec 1, 2023 and returning on Dec 12, 2023
------------------
CURRENT BROWSER CONTENT:
<link id=4>Book</link>
<link id=5>Manage Trips</link>
<link id=6>Check In</link>
<link id=7>Travel Info</link>
<link id=8>TrueBlue</link>
<link id=9 aria-label="Shopping cart (Empty)"/>
<text id=108>Flights available for sale through Mar 19, 2024.</text>
<button id=109>Done</button>
<button id=110>Search flights</button>
------------------
CURRENT ACTION:
CLICK 110
PREVIOUS LABEL:
CHOOSE_DATE Return 12/12/2023
CURRENT LABEL:
CLICK Search flights
================================================

The current context, browser content, action, previous label are below. Reply with your
    current label.

================================================
CONTEXT:
{context}
------------------
CURRENT BROWSER CONTENT:
{browser_content}
------------------
CURRENT ACTION:
{current_action}
------------------
PREVIOUS LABEL:
{previous_label}
------------------
CURRENT LABEL:
```

H.6 REASONING PROMPT

Listing 11: `Reasoning` Prompt

```
You are an AI assistant whose goal is to generate reasoning as to why a particular web action
    was taken. You are given:
1. CONTEXT: An instruction stating the overall goal you need to achieve
2. CURRENT BROWSER CONTENT: A simplified text description of the current browser content,
    without formatting elements.
3. CURRENT URL: The current webpage URL
4. PREVIOUS ACTIONS: A list of your past actions.
5. YOUR ACTION: The current action that you took

Provide reasoning (after REASONING:) for why the current action (YOUR ACTION:) was taken.

Here are some examples:

Example 1:
================================================
CONTEXT:
CHOOSE_DATE Depart "12/03/2023"
------------------
CURRENT BROWSER CONTENT:
<button id=18 title="Travelers">1 Adult</button>
<text id=19>Use TrueBlue points</text>
<text id=20>From</text>
<input id=21 text>New York City area (NYC)</input>
<button id=22>Reverse origin and destination city or airport</button>
<text id=23>To</text>
<input id=24 text>Boston area</input>
```

```
<text id=25>Depart</text>
<input id=26 Depart Press DOWN ARROW key to select available dates/>
<text id=27>Return</text>
<input id=28 Return Press DOWN ARROW key to select available dates/>
<button id=29 aria-label="Previous Month"/>
<text id=30>April 2023</text>
<text id=31>S</text>
<text id=32>M</text>
<text id=33>T</text>
<text id=34>W</text>
<text id=35>T</text>
<text id=36>F</text>
<text id=37>S</text>
<button id=38 aria-label="depart Saturday, April 1, 2023">1</button>
<button id=39 aria-label="depart Sunday, April 2, 2023">2</button>
<button id=40 aria-label="depart Monday, April 3, 2023">3</button>
<button id=41 aria-label="depart Tuesday, April 4, 2023">4</button>
<button id=42 aria-label="depart Wednesday, April 5, 2023">5</button>
<button id=63 aria-label="depart Wednesday, April 26, 2023">26</button>
<button id=64 aria-label="depart Thursday, April 27, 2023">27</button>
<button id=65 aria-label="depart Friday, April 28, 2023">28</button>
<button id=66 aria-label="depart Saturday, April 29, 2023">29</button>
<button id=67 aria-label="depart Sunday, April 30, 2023">30</button>
------------------
CURRENT URL:
https://www.jetblue.com/
------------------
PREVIOUS ACTIONS:
------------------
YOUR ACTION:
TYPE 26 Depart Press DOWN ARROW key to select available dates "12/03/2023"
REASONING:
I have no previous actions.
I have to first type "12/03/2023" in the field Depart which corresponds to input id 26
==================================================

Example 2:
==================================================
CONTEXT:
FILL_TEXT From "Seattle"
------------------
<button id=16>Stays</button>
<button id=17>Roundtrip</button>
<button id=18 title="Travelers">1 Adult</button>
<text id=19>Use TrueBlue points</text>
<text id=20>From</text>
<input id=21 text>Seattle</input>
<text id=22>Seattle</text>
<text id=23>, WA (SEA)</text>
<text id=24>Seattle</text>
<text id=25>/King Country, WA (BFI)</text>
<text id=26>Browse by regions</text>
<text id=27>Browse by regions</text>
<text id=28>Mint Service</text>
<text id=29>Partner Airline</text>
<button id=30>Reverse origin and destination city or airport</button>
------------------
CURRENT URL:
https://www.jetblue.com/
------------------
PREVIOUS ACTIONS:
TYPE 21 "Seattle"
------------------
YOUR ACTION:
CLICK 22 Seattle
REASONING:
I have already typed in "Seattle" in id 21
I should next check if there is a dropdown text below id 21
Yes, there is a corresponding dropdown text "Seattle" in id 22
I have to click on id 22
==================================================

CONTEXT:
{context}
------------------
CURRENT BROWSER CONTENT:
{browser_content}
------------------
CURRENT URL:
{url}
------------------
```

```
PREVIOUS ACTIONS:
{previous_actions}
------------------
YOUR ACTION: {current_action}
REASONING:
```