ISSN 2093-0542




KOREAN SOCIETY FOR INTERNET INFORMATION

The 21st International Conference on Intelligent Computing (INTCOM 2026)

July 03-06, 2026, O_2 universum, Prague, Czech Republic

https://intcom.kr

Proceedings of INTCOM 2026

| Organized by |

Korean Society for Internet Information (KSII)

| Co-Organized by |

Technical University of Kosice, Slovakia

| Sponsored by |

CzechTourism

Contents

An LLM-based Web Navigator (Guider) for Effective Information Search on Complex Websites

Sungho Park[1], Seong Jin Park[2], Janghwan Kim[3], Chaeyun Seo[4] and R. Young Chul Kim[5*]
Department of Software and Communication Engineering, Hongik University
Sejong, South Korea
[e-mail: andytjdgh@gmail.com, sjjw1111@naver.com, {lentoconstante, chaeyun, bob}@hongik.ac.kr]
*Corresponding author: R. Young Chul Kim

Abstract

Many large websites have complex structures. Users often struggle to find the right information they are searching for. Most complex sites, such as E-commerce, finance, public administration, and healthcare sites, pack too many menus on a single screen. Due to the overly complex menu, most seniors and regular users can't find the information they need. Even existing chatbots work only on a single site. General AI assistants cannot read the current screen context. To address this problem, we propose an LLM-based web navigation assistant, Guider. The guider shows a search path to reach the exact spot the user must click. We built Guider as a Chrome extension, so it works on any site. Our approach, Guider, converts a natural-language question into a click path through four steps: 1) DOM extraction, 2) LLM semantic reasoning, 3) validation, and 4) visual highlighting. We tested Guider on 20 scenarios from "the Government24 website" on the Korean government sites. Our Guider achieved an average accuracy of 80%. Our navigator, Guider, is designed to help seniors and other regular users find hidden menus quickly and easily.

Keywords: LLM, Chrome Extension, DOM Analysis, Web Accessibility

1. Introduction

Modern websites offer more functions and menus across many fields. These fields include e-commerce, finance, public administration, and healthcare. This trend makes site structures complex. Many sites also contain menus with similar names. As a result, users often spend more than five minutes to reach their goal [1]. This usability problem makes digitally underserved people give up on the service. It also causes time loss and higher bounce rates for general users [2].

Three existing solutions exist: built-in chatbots, general AI assistants, and automation agents. Each solution has clear limits. First, a built-in chatbot works only on its own site. Each site needs a separate build. Second, a general AI assistant cannot read the current screen. It gives only general answers. Third, an automation agent clicks on the user's behalf. This method removes the learning effect and reduces user control [3].

We propose Guider to solve these problems. Guider is an LLM-based web navigation assistant. It works on any website without site dependency. It marks the click path directly on the user's screen. The user asks a question in natural language. The Guider then analyzes the current page's DOM. It extracts the clickable elements. The LLM matches the user intent with each element's role. It infers the click path. Finally, Guider shows the path on the screen.

We selected "Government24" as our main test site. Government24 has a multi-level menu structure. It also contains many items with similar names. These features clearly show the usability problem. Government24 is a public service that every Korean user can access. This fact gives our evaluation strong reproducibility and value.

This paper has the following structure. Section 2 reviews related work and compares Guider with OpenAI Codex. Section 3 describes the system architecture and the four-step pipeline. Section 4 analyzes the accuracy results on Government24 scenarios. Section 5 presents the conclusion and future work.

2. Related Work

2.1 Agentic Web Automation: OpenAI Codex

Recent LLM-based assistants now act directly inside the web browser. OpenAI Codex is a leading example [4]. Codex started as a coding agent in 2025. In 2026, it added computer use, a built-in browser, and a Chrome extension. Codex now runs inside a live browser session. It reads the page DOM, clicks elements, fills forms, and extracts information on its own. This makes Codex a strong agentic web-automation tool.

2.2 Guidance versus Automation

Guider shares the same technical base as Codex. Both tools run as a Chrome extension. Both work on a live browser session. Both read the page DOM. This common ground makes a direct comparison fair. The two systems split on one key point. Codex performs the task for the user. It clicks and extracts data by itself. This is an automation approach. Earlier web agents, such as AutoWebGLM, follow the same automation line [3]. The Guider does not click for the user. It only shows where the user must click. The user makes the final action. This is a guidance approach.

This single difference leads to different outcomes. Codex targets developer productivity and task automation. The user delegates control to the agent. The user also loses the chance to learn the path. Guider keeps the user in full control. The user reaches the target by their own action. So the user learns the menu path as they go. This learning effect matters most for digitally underserved users. For these users, doing the task themselves builds real access ability. A fully automatic agent cannot give this benefit. **Table 1** compares the two systems.

Table 1. Comparison between OpenAI Codex and Guider

Aspect	OpenAI Codex	Guider
Delivery form	Chrome extension	Chrome extension
Operating environment	Live browser session	Live browser session
Page understanding	DOM reading	DOM reading
Execution agent	The agent acts for the user	User acts, with guidance
User control	Delegated to the agent	Fully retained
Learning effect	Lost	Preserved
Output	Performed actions and extracted data	On-screen click location
Main target	Developer productivity	General and underserved users
Error handling	Permission prompts and human review	Client-side DOM validation

3. Guider System Architecture

Fig. 1 shows the overall structure of the Guider system. We implemented the system as a Chrome browser extension. The system processes a natural-language query through a four-step pipeline. The four steps are DOM extraction, LLM semantic reasoning, client validation, and visual highlighting.

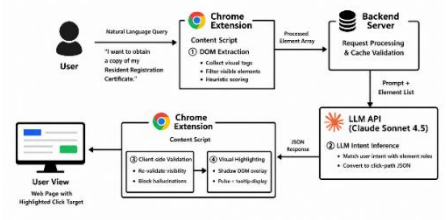

Fig. 1. Overall architecture of the Guider system

3.1 DOM Extraction Stage

The Content Script collects clickable elements from the current page. It uses semantic HTML tags such as a, button, [role="button"], and input. The system then applies visibility filtering. This filter checks display, opacity, and viewport position. It keeps only the elements that appear on the screen. Next, the system applies heuristic scoring. This step selects the candidates that match the user intent [5]. The system gives bonus

points to call-to-action words such as “issue” and “apply”. It gives penalty points to detour paths, such as search and login. This rule makes the LLM focus on the core click candidates [6]. The system also removes sensitive elements such as passwords. This step blocks the risk of personal data exposure. This pre-filtering reduces the number of tokens sent to the LLM. It also improves response speed and cost.

3.2 LLM Semantic Reasoning Stage

The system sends the filtered element list to the Claude Sonnet 4.5 model through a backend API [7]. The system prompt defines rules such as direct matching, menu priority, and CTA priority. The model matches the user intent with each element's role. It returns the click path as JSON. This method matches an informal phrase to an official term. For example, it matches “moving report” to “resident transfer report” [8]. The response also includes a confidence score. The system uses this score for later validation. A Redis cache handles repeated requests for the same domain. This cache greatly shortens the response time. The user enters a question through the Side Panel UI in **Fig. 2**. The user then checks the step-by-step guide.

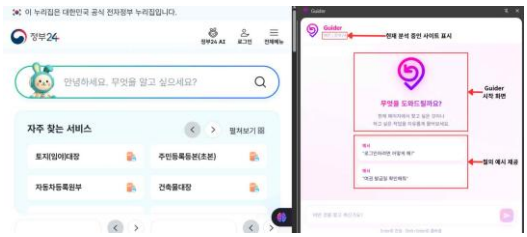

Fig. 2. Guider Side Panel user interface

3.3 Client Validation Stage

The system does not apply the LLM response directly. The client re-checks the element index in the response. It confirms that the index points to a real, visible element. This step blocks wrong guidance from LLM hallucination [9,10]. The system marks the result clearly when the confidence score falls below the threshold.

3.4 Visual Highlighting Stage

The system shows the validated element with a Shadow DOM overlay. **Fig. 3** shows a pulse animation, a number badge, and an arrow indicator. These elements help the user easily recognize the click position. The system uses a MutationObserver to detect page changes. It then automatically proceeds to the next step [11].

Fig. 3. Visual highlighting based on Shadow DOM

The system does not depend on a specific site. It works on any standard HTML website. The same pipeline applies without extra development for each site. This feature is the key difference from existing built-in chatbots.

4. Experiments

We performed two quantitative evaluations in this section. These evaluations test the accuracy of reasoning and the user convenience of Guider. The first evaluation measures the reasoning accuracy on Government24. The second evaluation compares menu search efficiency with 50 real users. Government24 is a typical complex website. It has a multi-level menu structure and many similar item names. This site gives our results strong generality and reproducibility.

4.1 Reasoning Accuracy Evaluation

We selected 20 common civil-service scenarios on the Government24 website. The set has 10 simple scenarios and 10 multi-step scenarios. A simple scenario needs one click. Examples include a copy of a resident registration and a family relationship certificate. A multi-step scenario needs two or three clicks. Examples include a business registration copy, a transfer report, and a driver's license renewal. We prepared the correct click path for each scenario in advance. We then checked whether Guider's first recommendation matched the first step of the correct path.

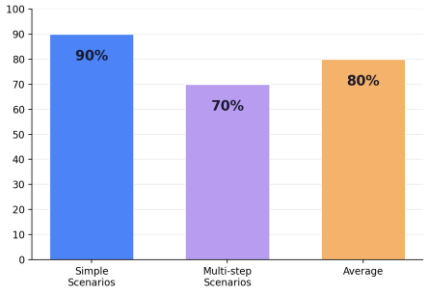

Fig. 4. First-recommendation accuracy by scenario type on Government24

Fig. 4 shows the results. The simple scenarios reached 90% first-recommendation accuracy. The multi-step scenarios reached 70%. The overall average was 80%. We analyzed the wrong cases. Most errors came from similar-menu confusion. For example, the system confused "transfer report" with "household member lookup". We can fix these errors with richer pre-indexing data for each site.

4.2 User Menu Search Efficiency Evaluation

We tested real usability with 50 users in their 20s and 30s. Each participant performed the "business registration copy" task twice on Government24. The first run used no Guider. The second run used Guider. We measured five metrics: task completion time, average click count, average wrong-click count, success rate, and subjective satisfaction on a five-point scale.

Without a guide, the average completion time was 5 minutes 30 seconds. 38 of the 50 users failed to find the menu and gave up. These users clicked an average of 17.3 times. 6.5 of these clicks were wrong clicks off the correct path. This result shows strong confusion among similar menu names. With Guider, the average completion time dropped to 1 minute 10 seconds. This change is about a 79% reduction. Every user finished the task, and no one gave up. The average click count dropped to 3.8, about a 78% reduction. The wrong clicks dropped to 0.4, about a 94% reduction. Users reached the correct path with almost no confusion. The subjective satisfaction rose from 2.4 to 4.6. This result confirms that Guider clearly reduces the burden of menu search.

Table 2. Menu search efficiency before and after the Guider

Metric	Without Guider	With Guider	Improvement
Avg. completion time	5 m 30 s	1 m 10 s	↓ 79%
Avg. clicks	17.3	3.8	↓ 78%
Avg. wrong clicks	6.5	0.4	↓ 94%
Success rate	24% (12/50)	100% (50/50)	↑ 76%p
Subjective satisfaction	2.4 / 5.0	4.6 / 5.0	↑ 91.7%

4.3 Semantic Reasoning Benefit Analysis

One result deserves special attention. The LLM correctly matched an informal phrase to the official term. It guided users to menus that a simple search could not find. An administrative service like Government24 labels menus with legal and official terms. So a general user often cannot recall the exact search word. This gap is a basic barrier. **Fig. 5** summarizes the informal-phrase matching cases from our experiment. For example, a user typed "moving report" and "get a copy". The LLM matched these phrases to "resident transfer report" and "resident registration copy". The LLM also matched "license renewal" to "driver aptitude test application". It matched "vaccine certificate" to "vaccination certificate issuance". These cases show that the LLM closes the word gap between user intent and official names. A simple text-matching tool relies on surface keyword overlap. Such a tool cannot reach this result. The LLM understands synonyms, slang, and short forms. This ability gives real value in web navigation. This semantic matching lets users reach a service without the exact term. So it improves access to information for digitally underserved people.

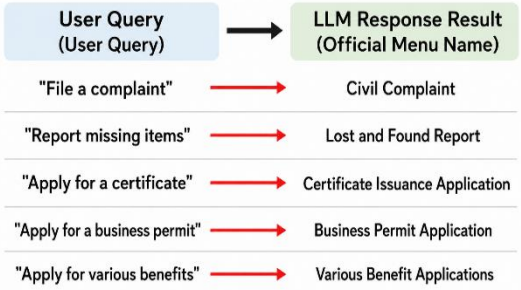

Fig. 5. Informal-expression matching through LLM semantic reasoning

5. Conclusions

In this paper, we designed and implemented Guider. Guider is an LLM-based visual web navigation assistant. It works on any website without site dependency. The system converts a natural-language query into an on-screen click path. It uses a four-step pipeline: DOM extraction, LLM semantic reasoning, client validation, and visual highlighting.

The accuracy evaluation used 20 Government24 scenarios. Guider achieved an average first-recommendation accuracy of 80%. The usability evaluation used 50 users in their 20s and 30s. The average task time dropped from 5 minutes 30 seconds to 1 minute 10 seconds, a reduction of about 79%. The success rate rose from 24% to 100%. The LLM also matched informal phrases such as “moving report” and “get a copy” to official terms. It guided users to menus that a simple search could not find.

This study makes four contributions. First, we built a general navigation assistant. It works on any standard HTML website without site dependency. Second, we chose visual guidance instead of AI automation. This choice preserves user control and the learning effect. Third, we added a client-side validation mechanism to mitigate LLM hallucinations. This mechanism blocks harm from wrong guidance. Fourth, we showed that heuristic scoring cuts the LLM token cost. It also focuses the reasoning on meaningful candidates.

In future work, we will expand the evaluation to more domains such as e-commerce, finance, and healthcare. This plan will quantitatively test the system's generality. We will also run a detailed usability study with more than 50 external users. This study will further confirm the social value. We will also add a Vision model. This model will extend support to Canvas-based sites and Closed Shadow DOM. Current DOM analysis cannot reach these areas.

Acknowledgment

This study is the result of a capstone design project by undergraduate students in the Department of Software Convergence at Hongik University. Additionally, this study is the result of a project by the 6th cohort trainees of the Metaverse Convergence SW Academy at Hongik University (Project Number: 202301830004).

References

[1] Squer, Bridging the Digital Divide: Why Accessibility Thinking and Empathy Matter, Squer Blog, 2024.

[2] WebAIM, The WebAIM Million – An annual accessibility analysis of the top 1,000,000 home pages, WebAIM Report, 2024.

[3] H. Lai et al., “AutoWebGLM: A Large Language Model-based Web Navigating Agent,” *arXiv preprint arXiv:2404.03648*, 2024.

[4] OpenAI, Introducing Codex, OpenAI, 2025.

[5] J. Liu et al., “WEPO: Web Element Preference Optimization for LLM-based Web Navigation,” in *Proc. of the AAAI Conference on Artificial Intelligence*, vol.39, no.25, pp.26614-26622, 2026.

[6] S. Anupam et al., “BrowserArena: Evaluating LLM Agents on Real-World Web Navigation Tasks,” *arXiv preprint arXiv:2510.02418*, 2025.

[7] Anthropic, The Claude 3 Family: Context, Reasoning, and Guardrails, Anthropic White Paper, 2024.

[8] X. Deng et al., “MindAct: Towards Computing with LLM-based Web Agents,” *arXiv preprint arXiv:2302.10666*, 2023.

[9] C. Jin et al., “HalluClear: Diagnosing, Evaluating and Mitigating Hallucinations in GUI Agents,” *arXiv preprint arXiv:2604.17284*, 2026.

[10] L. Huang et al., “A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions,” *ACM Transactions on Information Systems*, vol.43, no.2, pp.1-55, Mar. 2025.

[11] W3C, DOM Standard: Mutation Observers, WHATWG, 2024.