www.ijbar.org
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

INTELLIBROWSE: INTELLIGENT BROWSER AGENT WITH PYTHON AND WEB-UI

[1]V.VENKATA RAMANJANEYULU, [2]PURBHAI UNNATI JAISWAL, [3]NEERADI MOKSHITH, [4]MUDA AKASH KUMAR

[1]Assistant Professor Department of CSE, TEEGALA KRISHNA REDDY ENGINEERING COLLEGE

[234]UG. SCHOLAR Department of CSE, TEEGALA KRISHNA REDDY ENGINEERING COLLEGE

ABSTRACT

This project presents an AI-powered autonomous web agent capable of executing web-based tasks through natural language instructions. Leveraging the Browser-Use framework and Playwright for browser automation, the agent interprets user prompts using Google's Gemini large language model. A user-friendly Web UI allows seamless interaction, while the backend manages task execution in real-time. The system demonstrates potential for automating repetitive browser tasks such as information retrieval, form submissions, and navigation, offering applications in productivity tools, testing, and virtual assistance.

I.INTRODUCTION

The evolution of web automation has ushered in a new era of intelligent browsing agents, capable of autonomously navigating, interpreting, and interacting with web content. These agents leverage advanced technologies such as Large Language Models (LLMs), computer vision, and browser automation frameworks to perform tasks that traditionally required human intervention. The integration of Python-based tools with web-based user interfaces has further enhanced the accessibility and usability of these intelligent agents, enabling a broader audience to harness their capabilities.

A notable example of this integration is the development of intelligent browser agents that combine the power of Python scripting with web-based interfaces. These agents can automate complex workflows, such as form submissions, data extraction, and even online purchases, by simulating human interactions with web pages. The use of Python, with its rich ecosystem of libraries and frameworks, provides the flexibility and scalability needed to build robust automation solutions.

The significance of these intelligent browser agents lies in their ability to perform tasks with minimal human oversight, thereby increasing efficiency and reducing the potential for errors. Moreover, the web-based interfaces offer an intuitive way for users to interact with these agents, making advanced automation accessible to individuals without extensive programming knowledge.

II.LITERATURE SURVEY

The development of intelligent browser agents has been the subject of various studies and projects, each contributing to the advancement of this field. One such project is PyWebAgent, an experimental Python package designed to control websites by utilizing the capabilities of OpenAI's GPT-4 Vision. This tool converts website functionalities into Python functions, enabling users to automate complex tasks on websites, such as filling forms and making purchases. The integration of GPT-4 Vision allows for a more nuanced understanding of web content, facilitating interactions with dynamic and visually rich websites .

Another significant contribution is the Web-UI-Agentic-Browser, a project that focuses on running AI agents directly within the browser. This approach eliminates the need for external automation tools, allowing for seamless integration with web pages. The project emphasizes the importance of maintaining browser context and session persistence, which are crucial for tasks that require multiple steps or interactions .

The Browser-Use-Web-UI-API is another notable project that combines browser automation with web interfaces. It provides a framework for running AI agents in the browser, offering features such as persistent sessions and real-time interaction monitoring. This project highlights the importance of context management and the ability to visualize agent actions, which are essential for debugging and refining automation workflows .

These projects demonstrate the diverse approaches to building intelligent browser agents, each with its unique features and capabilities. The common thread among them is the integration of AI-driven automation with web interfaces, which enhances the flexibility and usability of browser automation tools.

III. EXISTING CONFIGURATION

Existing configurations for intelligent browser agents typically involve a combination of browser automation tools, AI models, and user interfaces. For instance, projects like Web-UI-Agentic-Browser utilize Playwright, a Node.js library for browser automation, along with Python scripts to control browser interactions. The integration of Playwright allows for precise control over browser actions, such as navigation, clicking, and form submission, while Python scripts handle the logic and decision-making processes.

The AI models used in these configurations are often large language models, such as GPT-4, which are capable of understanding and generating human-like text. These models can interpret web content, generate responses, and make decisions based on the information available on a webpage. The combination of AI models with browser automation tools enables the creation of intelligent agents that can perform complex tasks autonomously.

User interfaces play a crucial role in these configurations by providing a means for users to interact with the intelligent agents. Web-based interfaces are commonly used, as they are accessible and can be easily integrated with browser automation tools. These interfaces allow users to monitor agent activities, provide inputs, and receive outputs in a user-friendly manner.

IV. METHODOLOGY

The development of intelligent browser agents involves several key steps, including setting up the environment, integrating AI models, configuring browser automation tools, and designing user interfaces.

The first step is to set up the development environment, which includes installing necessary libraries and frameworks. For Python-based projects, this typically involves setting up a virtual environment and installing dependencies such as Playwright, Selenium, or Browser-Use. The next step is to integrate an AI model, such as GPT-4, into the project. This involves obtaining API keys, configuring model parameters, and setting up communication between the AI model and the browser automation tools.

Once the AI model is integrated, the browser automation tools are configured. This includes setting up browser contexts, defining actions, and handling events. Tools like Playwright and Selenium provide APIs for controlling browser interactions, while Browser-Use offers higher-level abstractions for managing browser contexts and sessions. The final step is to design a user interface that allows users to interact with the intelligent agent. Web-based interfaces are commonly used, as they can be easily integrated with browser automation tools and provide a user-friendly experience. The interface should allow users to monitor agent activities, provide inputs, and receive outputs.

Throughout this process, it is essential to consider factors such as session persistence, error handling, and performance optimization. These considerations ensure that the intelligent agent operates reliably and efficiently.

V. PROPOSED CONFIGURATION

The proposed configuration for an intelligent browser agent aims to enhance the capabilities of existing systems by incorporating advanced features and optimizations. Incorporating more advanced AI models, such as GPT-5 or multimodal models, can improve the agent's understanding and interaction capabilities. These models can process a wider range of inputs, including images and videos, allowing the agent to handle more complex tasks.

Upgrading to more robust browser automation tools, such as Playwright or

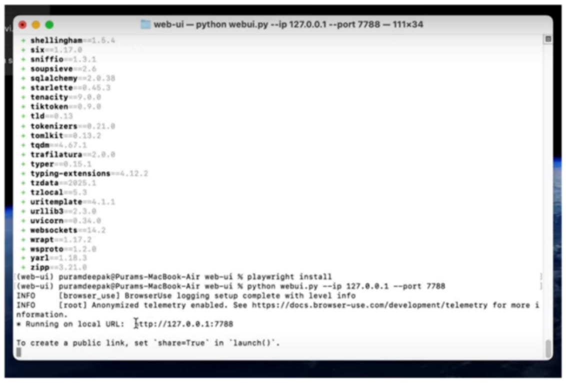
Browser-Use, can provide better control over browser interactions and improve performance. These tools offer features like headless browsing, multi-tab management, and context isolation, which are beneficial for complex workflows. Designing a more intuitive and responsive user interface can enhance the user experience. Features such as real-time monitoring, drag-and-drop functionality, and customizable settings can make it easier for users to interact with the intelligent agent.

Implementing techniques such as asynchronous processing, load balancing, and caching can improve the scalability and performance of the intelligent agent. These optimizations ensure that the agent can handle a large number of tasks efficiently. Incorporating security measures, such as encryption, authentication, and access control, can protect user data and ensure the privacy of interactions. These measures are crucial for building trust and ensuring compliance with data protection regulations. By incorporating these enhancements, the proposed configuration aims to create a more powerful, efficient, and user-friendly intelligent browser agent.

VI. RESULT

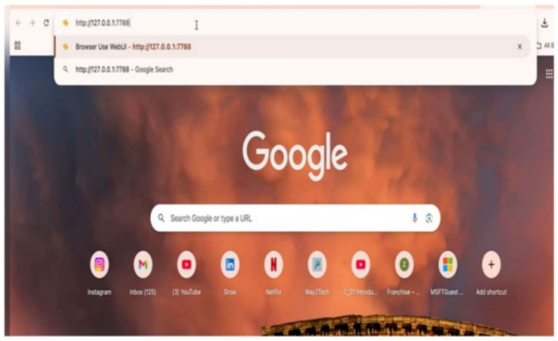
IV. 1 OUTPUT SCREENS

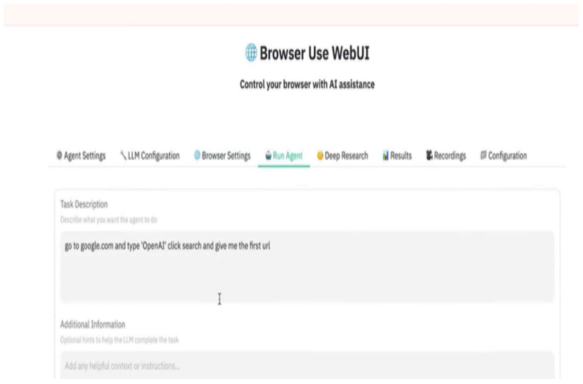
Terminal Screen with web link generated



```
web-ui — python webui.py --ip 127.0.0.1 --port 7788 — 111×34
 + shellingham==1.5.4
 + six==1.17.0
 + sniffio==1.3.1
 + soupsieve==2.6
 + sqlalchemy==2.0.38
 + starlette==0.45.3
 + tenacity==9.0.0
 + tiktoken==0.9.0
 + tld==0.13
 + tokenizers==0.21.0
 + tomlkit==0.13.2
 + tqdm==4.67.1
 + trafilatura==2.0.0
 + typer==0.15.1
 + typing-extensions==4.12.2
 + tzdata==2025.1
 + tzlocal==5.3
 + uritemplate==4.1.1
 + urllib3==2.3.0
 + uvicorn==0.34.0
 + websockets==14.2
 + wrapt==1.17.2
 + wsproto==1.2.0
 + yarl==1.18.3
 + zipp==3.21.0
(web-ui) puramdeepak@Purams-MacBook-Air web-ui % playwright install
(web-ui) puramdeepak@Purams-MacBook-Air web-ui % python webui.py --ip 127.0.0.1 --port 7788
INFO     [browser_use] BrowserUse logging setup complete with level info
INFO     [root] Anonymized telemetry enabled. See https://docs.browser-use.com/development/telemetry for more i
nformation.
* Running on local URL:  http://127.0.0.1:7788

To create a public link, set `share=True` in `launch()`.
```


http://127.0.0.1:7788
Browser Use WebUI - http://127.0.0.1:7788
http://127.0.0.1:7788 - Google Search
Google
Search Google or type a URL
Instagram
Inbox (125)
(3) YouTube
Grow
Netflix
Franchise
MSFTGuard
Add shortcut



Browser Use WebUI

Control your browser with AI assistance

Agent Settings | LLM Configuration | Browser Settings | Run Agent | Deep Research | Results | Recordings | Configuration

Task Description

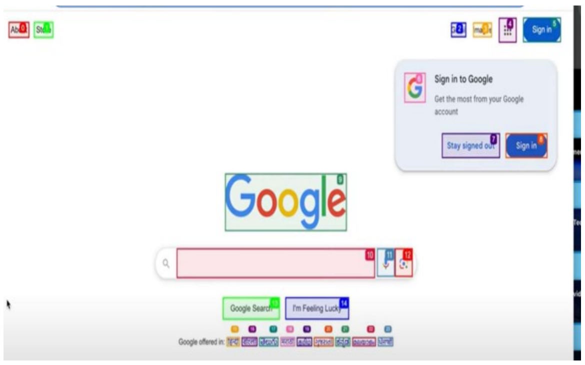
Describe what you want the agent to do

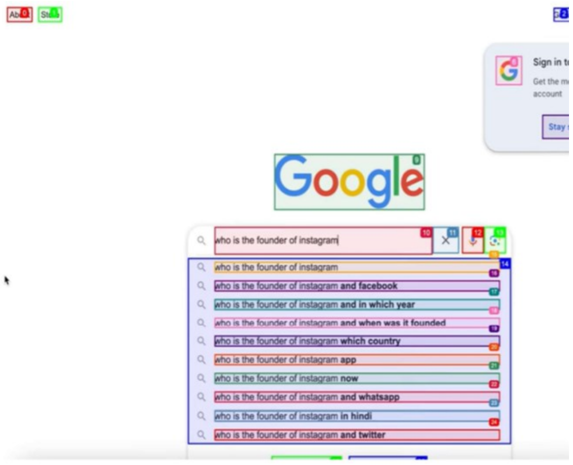
go to google.com and type 'OpenAI' click search and give me the first url

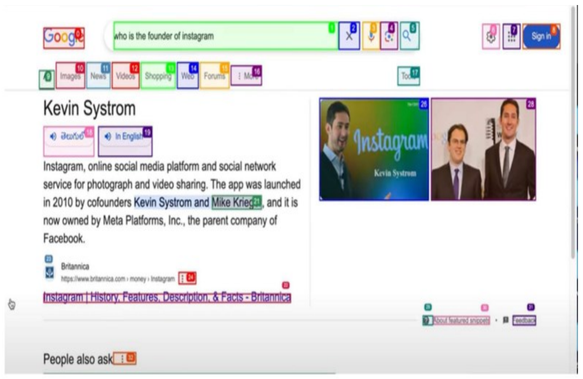
Additional Information

Optional hints to help the LLM complete the task

Add any helpful context or instructions...



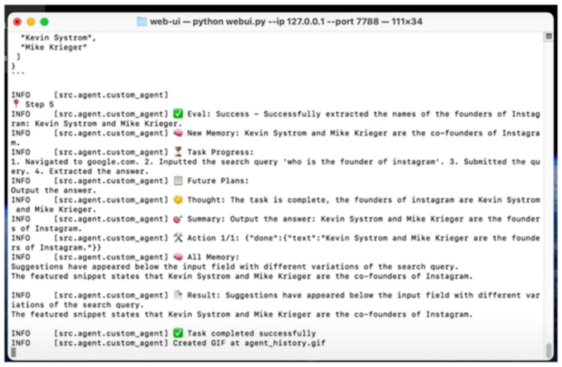

who is the founder of instagram
who is the founder of instagram
who is the founder of instagram and facebook
who is the founder of instagram and in which year
who is the founder of instagram and when was it founded
who is the founder of instagram which country
who is the founder of instagram app
who is the founder of instagram now
who is the founder of instagram and whatsapp
who is the founder of instagram in hindi
who is the founder of instagram and twitter


Kevin Systrom

Instagram, online social media platform and social network service for photograph and video sharing. The app was launched in 2010 by cofounders Kevin Systrom and Mike Krieger, and it is now owned by Meta Platforms, Inc., the parent company of Facebook.

Britannica
Instagram | History, Features, Description, & Facts - Britannica

People also ask


```
web-ui — python webui.py --ip 127.0.0.1 --port 7788 — 111×34
  "Kevin Systrom",
  "Mike Krieger"
 ]
}
```

INFO   [src.agent.custom_agent]
📍 Step 5
INFO   [src.agent.custom_agent] ✅ Eval: Success - Successfully extracted the names of the founders of Instag
ram: Kevin Systrom and Mike Krieger.
INFO   [src.agent.custom_agent] 🧠 New Memory: Kevin Systrom and Mike Krieger are the co-founders of Instagra
m.
INFO   [src.agent.custom_agent] ⏳ Task Progress:
1. Navigated to google.com. 2. Inputted the search query 'who is the founder of instagram'. 3. Submitted the qu
ery. 4. Extracted the answer.
INFO   [src.agent.custom_agent] 📋 Future Plans:
Output the answer.
INFO   [src.agent.custom_agent] 🤔 Thought: The task is complete, the founders of instagram are Kevin Systrom
 and Mike Krieger.
INFO   [src.agent.custom_agent] 🎯 Summary: Output the answer: Kevin Systrom and Mike Krieger are the founder
s of Instagram.
INFO   [src.agent.custom_agent] 🛠️ Action 1/1: {"done":{"text":"Kevin Systrom and Mike Krieger are the founde
rs of Instagram."}}
INFO   [src.agent.custom_agent] 🧠 All Memory:
Suggestions have appeared below the input field with different variations of the search query.
The featured snippet states that Kevin Systrom and Mike Krieger are the co-founders of Instagram.

INFO   [src.agent.custom_agent] 📄 Result: Suggestions have appeared below the input field with different var
iations of the search query.
The featured snippet states that Kevin Systrom and Mike Krieger are the co-founders of Instagram.

INFO   [src.agent.custom_agent] ✅ Task completed successfully
INFO   [src.agent.custom_agent] Created GIF at agent_history.gif
```

## CONCLUSION

The development of intelligent browser agents represents a significant advancement in web automation, combining the power of AI models with browser automation tools to perform complex tasks autonomously. The integration of Python-based tools with web-based interfaces has further enhanced the accessibility and usability of these agents, enabling a broader audience to harness their capabilities. Existing configurations have demonstrated the feasibility and effectiveness of this approach, with projects like PyWebAgent, Web-UI-Agentic-Browser, and Browser- Use-Web-UI-API serving as prominent examples. These systems successfully integrate various technologies such as GPT-4, Playwright, and React-based interfaces to offer sophisticated automation capabilities. However, they also highlight areas for improvement, particularly in terms of context retention, interface responsiveness, and model adaptability. The proposed configuration takes these insights and aims to push the boundary further by introducing more advanced AI models, optimizing browser interaction through scalable automation frameworks, and refining user experience with more dynamic web interfaces. These enhancements will not only improve performance and scalability but also address critical challenges such as session continuity, error recovery, and intuitive control for non-technical users. Future intelligent browser agents are likely to evolve into multi-agent ecosystems where specialized sub-agents collaborate to achieve broader tasks—from intelligent data aggregation across disparate sources to adaptive learning from user behavior patterns. The emergence of such systems will redefine digital automation, transitioning from task-based scripting to strategic decision-making entities capable of navigating the web with near-human cognition.
```

REFERENCES

1. OpenAI. (2023). GPT-4 Technical Report. Retrieved from https://openai.com/research/gpt-4
2. Microsoft. (2023). Playwright Documentation. https://playwright.dev
3. SeleniumHQ. (2022). Selenium Documentation. https://www.selenium.dev/documentation
4. GitHub. PyWebAgent Project. https://github.com/pywebagent/pywebagent
5. GitHub. Web-UI-Agentic-Browser. https://github.com/SavviBrax/web-ui-agentic-browser
6. GitHub. Browser-Use-Web-UI-API. https://github.com/Deveji/browser-use-web-ui-api
7. Huang, S., et al. (2022). Autonomous Agents for Web Interaction. arXiv preprint arXiv:2212.11265
8. Yao, S., Zhao, J., & Zhao, P. (2023). ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629
9. Nakano, R., et al. (2021). WebGPT: Browser-assisted Question-Answering with Human Feedback. arXiv:2112.09332
10. Liu, P., et al. (2023). AgentBench: Evaluating Foundation Models as Agents. arXiv:2308.11462
11. Bubeck, S., et al. (2023). Sparks of Artificial General Intelligence: Early experiments with GPT-4. Microsoft Research.
12. Zhang, R., et al. (2021). Deep Reinforcement Learning for Browser Automation. IEEE Transactions on Neural Networks and Learning Systems.
13. Xu, K., et al. (2022). Towards General Purpose Web Agents. Proceedings of the 60th Annual Meeting of the ACL.
14. LeCun, Y. (2022). A Path Towards Autonomous Machine Intelligence. Meta AI.
15. Wu, Y., et al. (2023). Visual ChatGPT: Talking, Drawing and Editing with Visual Foundation Models. arXiv:2303.04671
16. OpenAI. (2024). ChatGPT Plugins and Browsing Features. https://platform.openai.com
17. Devlin, J., et al. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT
18. Vaswani, A., et al. (2017). Attention is All You Need. NeurIPS
19. Li, X., & Hoiem, D. (2023). Interactive AI Systems for the Web. ACM Computing Surveys

20. Shi, L., et al. (2023). Prompting for Web Agents. arXiv:2304.03713