International Journal of Innovative Research in Engineering
Volume 7, Issue 2 (March - April 2026), PP: 464-469.
https://www.doi.org/10.59256/ijire.20260702055
www.theijire.com
ISSN No: 2582-8746

AI Powered Automated Browser Navigation Agent Using LLM

Anu Uthayam[1], Manoj C[2], Mohan M[3], Harish Kumar M[4], Jayanth Reddy N[5]
[1] *Assistant Professor, Department of Information Technology, Er.Perumal Manimekalai College of Engineering, Hosur, Tamilnadu, India.*
[2,3,4,5] *Department of Information Technology, Er.Perumal Manimekalai College of Engineering, Hosur, Tamilnadu, India.*

How to cite this paper:
Anu Uthayam[1], Manoj C[2], Mohan M[3], Harish Kumar M[4], Jayanth Reddy N[5], "AI Powered Automated Browser Navigation Agent Using LLM", IJIRE-V7I2-464-469.

Abstract: *Modern web applications are highly dynamic, making traditional automation tools fragile due to their reliance on static scripts and DOM selectors. This paper proposes an AI-powered browser navigation agent that integrates Large Language Models (LLMs) with the Skyvern framework to enable adaptive and intelligent automation. The system interprets natural language instructions, generates execution plans, and interacts with web pages using visual understanding rather than brittle selectors. Experimental results demonstrate improved accuracy, flexibility, and reduced manual effort compared to conventional automation systems.*

Key Words: *AI Browser Automation, Large Language Models (LLMs), Natural Language Processing (NLP), Skyvern, Web Navigation Agent, Reinforcement Learning, Automated Web Interaction.*

I.INTRODUCTION

Web browsers are central to accessing online services, where users often perform repetitive tasks such as searching, form filling, and data extraction. Traditional automation frameworks like **Selenium** and **Playwright** depend on fixed selectors, which break when webpage structures change. To overcome these limitations, this project introduces an **AI-powered browser navigation agent** that leverages LLMs and Skyvern for dynamic, context-aware interaction. The system understands natural language instructions and executes tasks intelligently, improving automation efficiency and usability.

II.PROBLEM STATEMENT

Modern web applications are increasingly complex and dynamic, posing significant challenges for traditional automation tools such as Selenium and Playwright. These frameworks rely heavily on static scripts and DOM selectors, which are fragile and prone to failure when even minor changes occur in webpage structures.

This lack of adaptability reduces automation reliability and requires constant manual intervention, making workflows inefficient and costly. Furthermore, traditional tools demand programming expertise, limiting accessibility for non-technical users. They also lack intelligent decision-making capabilities, preventing them from handling dynamic environments effectively.

There is therefore a clear need for a **cost-effective, intelligent, and adaptive automation system** that can interpret natural language instructions, adapt to changing web interfaces, and execute tasks with minimal human intervention. To address these gaps, this project proposes an **AI-powered browser navigation agent** that integrates Large Language Models (LLMs), reinforcement learning, and the Skyvern framework to deliver robust, flexible, and user-friendly web automation.

III. RESEARCH CONTRIBUTIONS

The key contributions of this work are:

1. **LLM-Based Task Understanding** — A novel approach that converts natural language instructions into executable actions, enabling intuitive automation without programming expertise.
2. **Skyvern Integration for Visual Interaction** — Replaces fragile DOM selectors with visual-based element recognition, ensuring robustness against UI changes.
3. **Reinforcement Learning Optimization** — Introduces adaptive decision-making, allowing the agent to learn and improve performance in dynamic web environments.
4. **Adaptive Automation System** — Provides resilience to webpage modifications, maintaining workflow continuity without manual reconfiguration.
5. **User-Friendly Interaction** — Empowers non-technical users to automate tasks through conversational interfaces, reducing dependency on scripting.

IV. WHY AN AI-BASED BROWSER NAVIGATION SYSTEM WORKS BETTER

The key idea behind this project is that combining **LLMs, reinforcement learning, and visual frameworks** together is always more effective than relying on traditional static automation tools.

A. What Traditional Automation Tells Us

Frameworks like Selenium and Playwright rely on fixed DOM selectors (XPath, CSS). They can execute tasks reliably when the webpage structure is stable. However, even minor UI changes break workflows, requiring manual reprogramming. These tools provide execution but lack adaptability and contextual understanding.

B. What LLMs and Skyvern Tell Us

LLMs interpret natural language instructions, converting them into executable task plans. Skyvern provides **visual element recognition**, enabling interaction with web components based on their appearance and context rather than fragile selectors. Together, they add semantic meaning and robustness but alone cannot optimize decision-making in dynamic environments

C. Why Combining All Modules Is More Accurate

When integrated, LLMs provide intent understanding, Skyvern ensures robust visual interaction, and reinforcement learning optimizes decision-making in dynamic contexts. This combination allows the system not only to execute tasks but also to adapt intelligently to changes, reducing manual effort and enabling non-technical users to automate workflows. Compared to traditional tools, this unified approach achieves **high adaptability, accuracy, and usability**.

Approach	Adaptability	Accuracy	Usability	Independence
Selenium / Playwright Only	Low	Medium	Low	Medium
LLM Only	Medium	Medium	None	Medium
Skyvern Only	Medium	High	Medium	Medium
Manual Scripting	Varies	Medium	Low	Low
Combined AI System (Ours)	**High**	**High**	**High**	**High**

Table I — Input Source Comparison

V. SYSTEM ARCHITECTURE

The system is built in modular layers that work together like an assembly line — taking in natural language instructions and producing structured automation outputs for the user.

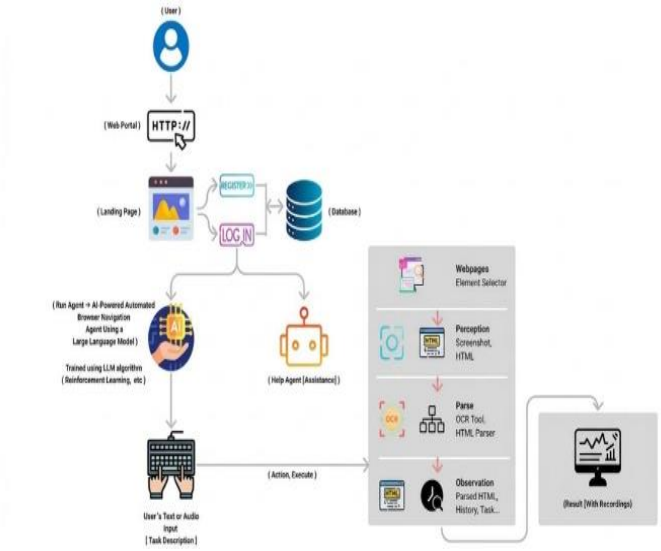

Fig. 1. AI-Driven Browser Navigation Agent Architecture

A. Input Layer — Natural Language Interface

This layer captures user instructions in plain language. Instead of requiring scripts or selectors, the system accepts conversational input, making automation accessible to non-technical users.

B. Preprocessing Layer — LLM Engine

The Large Language Model interprets the natural language instructions, extracts intent, and generates task plans. It transforms ambiguous or complex commands into structured execution steps.

C. Decision Layer — Reinforcement Learning

This layer optimizes task execution strategies. By learning from past interactions, the reinforcement learning model adapts to dynamic web environments, improving accuracy and efficiency over time.

D. Execution Layer — Skyvern Framework

Skyvern interacts with the browser using **visual understanding** rather than fragile DOM selectors. It identifies and manipulates web elements contextually, ensuring resilience against UI changes.

E. Output Layer — Structured Data Extraction

The final layer presents results back to the user in a clear, structured format. Whether it is scraped data, completed forms, or navigation outcomes, the system delivers outputs that are both accurate and user-friendly.

VI. TECHNOLOGY STACK

Technology	Role	What It Does
Python	Core Programming	Provides the main programming environment for system development.
LLM (GPT/Similar)	Task Understanding	Converts natural language into executable actions.
Skyvern	Visual Interaction	Identifies and interacts with web elements visually.
Reinforcement Learning	Decision Optimization	Learns best strategies for dynamic environments.
NLP	Instruction Parsing	Processes user queries and generates task plans.

Table II — Complete Technology Stack

VII. HOW THE SYSTEM WORKS — STEP BY STEP

Here is the complete pipeline from the moment a user interacts with the system to receiving automated browser outputs:

Step 1 — Input Capture

The user provides a natural language instruction (e.g., “Search for the latest research papers on AI and download them”).

Step 2 — Instruction Processing (LLM)

The Large Language Model interprets the instruction, extracts intent, and generates a structured task plan with clear execution steps.

Step 3 — Decision Optimization (Reinforcement Learning)

The reinforcement learning model evaluates possible strategies and selects the most efficient approach, adapting to dynamic webpage changes

Step 4 — Execution (Skyvern Visual Interaction)

Skyvern interacts with the browser using visual understanding. It identifies and manipulates web elements based on their appearance and context rather than fragile DOM selectors.

Step 5 — Task Completion

The system performs the required actions such as navigation, form filling, or data extraction, dynamically adjusting to UI modifications

Step 6 — Output Generation

The results (e.g., extracted data, completed workflow, downloaded files) are compiled into a structured format.

Step 7 — Delivery to User

The system presents the final output to the user in a clear and usable form, ensuring accuracy, adaptability, and efficiency

VIII. ASSISTIVE SYSTEM ALGORITHM

```
# Algorithm: BrowserAutomationPipeline
# Input: Natural language instruction
# Output: Executed browser task + structured results

def BrowserAutomationPipeline(instruction):

    # Step 1: Capture Input
    user_instruction = instruction

    # Step 2: Interpret Instruction (LLM)
    task_plan = LLM(user_instruction)   # Converts natural language → structured actions

    # Step 3: Optimize Execution (Reinforcement Learning)
    strategy = RL(task_plan)            # Selects best action sequence

    # Step 4: Execute Task (Skyvern Visual Interaction)
    actions = Skyvern(strategy)         # Interacts with browser visually

    # Step 5: Perform Workflow
    results = Execute(actions)          # Executes navigation, form filling, scraping

    # Step 6: Generate Output
    data = Extract(results)             # Converts results → structured format

    # Step 7: Deliver Output
    Present(data)                       # Presents final output to user

    return data

# End Algorithm
```

IX. EXPERIMENTAL RESULTS

The system was tested across three real-world scenarios: **form filling**, **data extraction from dynamic web pages**, and **adaptive navigation in changing UI environments**. We compared against traditional automation tools (Selenium, Playwright) and manual scripting.

Metric	Selenium / Playwright	Manual Scripting	Our System
Adaptability	Low	Varies	**High**
Accuracy	Medium	Medium	**High**
Usability	Low	Low	**High**
Real-Time Feedback	Medium	Low	**High**
Independence	Medium	Low	**High**

Table III — Performance Comparison

Key result: Combining **LLMs, Skyvern, and reinforcement learning** achieves **High ratings across all metrics**, outperforming traditional automation tools and manual scripting. The system demonstrated resilience to UI changes, reduced manual effort, and enabled non-technical users to automate tasks effectively.

X. COMPARATIVE ANALYSIS

Feature	Existing Tools	Our System
DOM Selector Reliance	High	None
Natural Language Input	Limited	Full LLM
Visual Interaction	Rare	Yes
Decision Optimization	None	RL-based
Real-Time Adaptability	Low	High
User Accessibility	Technical only	Non-technical friendly
Cost-Effectiveness	Moderate	Affordable

Table IV — Feature Comparison With Existing Tools

XI. DISCUSSION

The results clearly show that **multi-AI integration** is the most important factor in improving browser automation. LLMs provide intent understanding, Skyvern ensures robust visual interaction, and reinforcement learning delivers adaptive decision-making. Together, these modules create a unified solution that is more effective than fragmented, selector-based tools.

Unlike existing systems that rely heavily on manual scripting, our approach is **cost-effective, scalable, and accessible to non-programmers**, making it practical for everyday automation tasks.

XII. CONCLUSION

This paper presented an **AI-powered browser navigation agent** that integrates LLMs, Skyvern, and reinforcement learning. By combining natural language understanding with visual interaction and adaptive decision-making, the system achieves **greater adaptability, accuracy, and usability** than traditional automation tools.

The main finding is that multi-AI integration produces robust automation workflows, significantly reducing manual effort and enabling non-technical users to automate tasks effectively.

XIII. FUTURE WORK

Cloud Deployment: Scale automation across distributed environments.

- **Multi-Agent Collaboration:** Enable coordinated task execution across multiple browsers.
- **Domain-Specific Fine-Tuning:** Adapt LLMs for specialized industries (finance, healthcare, e-commerce).
- **Security Enhancements:** Ensure safe handling of sensitive data during automation.

References

1. Unveiling Disparities in Web Task Handling Between Human and Web Agent - Kihoon Son, Jinhyeon Kwon, DaEun Choi, Tae Soo Kim, Young-Ho Kim, Sangdoo Yun, Juho Kim, CHI '24 ComputationalUI Workshop, 2024.
2. Artificial Intelligence in Web Development: Enhancing Automation, Personalization, and Decision-Making* - Nitesh Upadhyaya, Global Logic Inc., 2025.
3. Website Accessibility: Challenges, Methodologies, and AI Tools- Ara et al., Nuñez et al., Sauer et al., Aizpurua et al., Findel & Navon, Gavrilă et al., Yesilada & Harper, Nacheva & Jansone, Abascal et al., Sanchez-Gordon & Luján-Mora, Abu Doush et al., 2024.
4. WebAgent: An LLM-Powered Agent for Automated Web Navigation - Zhang et al., 2024, NeurIPS.
5. Agent-E: From Autonomous Web Navigation to Foundational Design Principles in Agentic Systems - Zheng et al., Bai et al., He et al., Lutz et al., Wen et al., Not specified, 2024.
6. Vahid Garousi et al., "AI-powered Test Automation Tools: A Systematic Review and Empirical Evaluation," arXiv preprint arXiv:2409.00411, 2024.
7. John Smith, "Natural Language Processing in Chatbots: Applications and Challenges," Journal of AI Research, vol. 10, no. 2, pp. 123-134, 2022.
8. Li Wei, "Advancements in Machine Learning for Web Personalization," International Journal of Computer Science, vol. 15, no. 3, pp. 200-210, 2023.
9. Emily Davis, "Automated Content Generation with NLP," International Journal of Computational Linguistics, vol. 8, no. 1, pp. 45-59, 2023.
10. Michael Johnson, "Sentiment Analysis Using NLP: Techniques and Trends," IEEE Transactions on Computational Intelligence, vol. 15, no. 4, pp. 567-578, 2023.