Implementation of AI-Powered Web Automation for Educational and Job Portals Using NLP

Madhavi R P
B.M.S College of Engineering
Bengaluru, India

Shreyansh Sethiya
B.M.S College of Engineering
Bengaluru, India

Paarth Sanyal
B.M.S College of Engineering
Bengaluru, India

Gopal Agrawal
B.M.S College of Engineering
Bengaluru, India

Mandaar Adarsh
B.M.S College of Engineering
Bengaluru, India

***Abstract*—Students and also job hunters report doing the same things over and over on websites which include checking results, applying for jobs, or downloading documents. This may be a challenge for those without great tech know how which in turn may make it hard for them to use the sites as they should. We looked at research which looks at the use of AI and tools like NLP models and Playwright to improve on this. We covered improvements in identifying what the user wants, creating scripts, putting together automation frameworks, and in the use of voice commands. We also look at what is present at present, the issues we ran into and what we see for the future in making this automation more accessible.**

***Index Terms*—Web automation, Natural Language Processing, Playwright, Educational portals, Job portals, Intent recognition, LLMs.**

I. Introduction

The*"AI-Powered Web Automation for Educational and Job Portals Using NLP"* project showcases an intelligent framework developed to aid non-technical individuals, especially students and job seekers, to navigate intricate websites via simple natural language instructions. NLP models like GPT and BERT are contemporarily used to automate the understanding and generation of user executable scripts via voice commands to perform monotonous activities such as applying for jobs, downloading certificates, and submitting forms. Unlike old automation systems, with Selenium and Puppeteer as the prime examples, GPT and BERT based NLP systems do not require the user to know programming or provide scripts to perform automation.

This issue is that of growing demand for automatic tools which are easy to use and which do not require in depth tech knowledge. We have designed it to take simple English instructions and turn them into live working scripts which in turn makes tasks go faster and also reduces the chance of error. We have built the system with robust and flexible technology which includes Node.js, React.js, Docker and we also use cloud services like AWS and Azure which in turn makes it work well with many different websites that which in turn are very much in a state of change. We use WebSockets for real time updates which in turn makes the auto process more interactive and transparent.

This project aims to develop and implement a natural language processing AI system to fully automate automation workflows. It will correctly interpret user inputs, compose flexible scripts using Playwright, permit usage through a web interface by non-technical users, and ensure compliance to the Sustainable Development Goals. Specifically, the project addresses systems improvement for education access concerning SDG 4 Quality Education, automation to enhance work efficiency for SDG 8 Decent Work and Economic Growth, and the advancement of universal digital solutions for SDG 9 Industry, Innovation and Infrastructure.

Current tools are limited because they rely on manual procedures to script out domain-specific logic, which means there is little use of Natural Language Processing for automation of tasks that are more general in nature. The proposed system addresses these shortcomings because it provides an intelligent automation system that is unified and easy enough to use by non-programmers. It is built as scalable, modular cross-domain automations, which makes it applicable to real-world use cases and will be relevant to users across educational and job portals. As a result, the proposed automation system hopes to democratized access to advanced automation capabilities that will improve productivity.

II. Literature Review

The research in AI-powered web automation has progressively evolved from rule-based scripting approaches to intelligent systems powered by Natural Language Processing (NLP) and Large Language Models (LLMs). Early foundational works by researchers such as Rashid et al. (2021), Hadi et al. (2020), and Sharma and Jain (2021) demonstrated the integration of NLP models like BERT with automation tools like Selenium and Playwright. These efforts primarily aimed to simplify repetitive web tasks—such as form fi...

In 2023 and 2024, new endeavors began to leverage large language models to develop better tools. The LLM4Jobs project by Li et al. (2023) worked on auto-extraction of job-related data that it extracted, and then in 2024, Lai et al. developed AutoWebGLM that allows computers the freedom to do the exploring of the worldwide web. These examples illustrate where natural language processing and ai can provide assistance in the active use of tasks with very little human intervention. This fits well with the purpose of the current project.

While there have been significant advances there are still challenges. Current systems are restricted to domains of expertise, rely on humans to provide textual instructions, and often do not adapt well to web based systems that change often. Older systems, that applied simple logical rules, were often too rigid. New systems currently are struggling with users who are unclear when they ask their questions, keeping track of multi-turn conversations,

The proposed project will fill the research gap since it is clear from the findings above that current projects do not provide a comprehensive, scalable, low-code interface that combines real-time NLP intent detection and translatable script generation for non-coder audiences. The plan is to create a cross-domain, AI-integrated web scraping and automation application that executes natural language commands and executes Playwright scripts in response to various domains, including but not limited to, educational and employment resources.

TABLE I
MAPPING OF OBJECTIVES WITH RESEARCH GAPS

Research Gap	Mapped Objective
Absence of integrated NLP-based automation systems	Develop an AI-powered system that combines Playwright with LLMs to enable natural language-driven automation
Lack of generalization across dynamic platforms	Design a framework that adapts to multiple websites and supports varied tasks like login, scraping, and submissions
Current systems require scripting knowledge	Eliminate the need for manual coding by generating Playwright scripts dynamically from user input
Limited adaptability to changing content	Use models like GPT or BERT to support real-time NLP-based script generation and flexible web interaction

III. Methodology

This chapter details the process of creating and testing the web automation tool with AI. It presents the method by which the research was conducted and subsequently the findings for the effectiveness of the systems. The research process contained elements of both qualitative and quantitative techniques to support need resolution for technology and end-user experience without programming.

The quantitative is the statistical testing outcomes based on use of the proposed systems. The qualitative is the problem identification based on user feedback, assessment of analogous web automation systems and requirements based on real world exposure.We also found the limitations of existing applications, Selenium and Puppeteer, based on nonprogrammer access.

The quantitative aspect of the methodology involves the implementation and experimental validation of the system using NLP models like GPT and BERT. Several performance metrics were used to evaluate the system, including:

- **Intent Recognition Accuracy:** The ability of the NLP module to correctly interpret and classify user instructions.
- **Script Execution Success Rate:** The percentage of Playwright-generated scripts that successfully completed the intended web automation tasks.
- **Task Completion Time:** The average time taken to complete automated tasks compared to manual execution.
- **User Effort Reduction:** Measured by the reduction in the number of manual steps required for each task.
- **Error Rate:** Incidences of failure due to incorrect interpretation or incomplete automation flows.

In order to validate that the results were valid and possibly generalized in other contexts, a number of job and education websites and pages (which included fictitious job listing sites, Naukri.com, Digilocker, various university result pages) were used to conduct tests. Many forms of standard commands were used, which a user might use in everyday jobs or studies, such as ”Check my result,” ”Apply for internships,” or ”Download my marks card,” as examples of real-world use cases. Custom scripts were written and executed in a Docker container to provide a simulation of the intended environment, enabling the automation to be secure and isolated.

The technology stack utilized Node.js and Python for the backend and React.js for the frontend. Docker and various services (including AWS) were used to deploy online and be scalable. The minimized WebSocket connection to the automated tasking tool created a low-latency user leverage where the user could view in real-time whether or not they completed a task.

A group of users that had no extensive technical background - mainly students from varied school systems - tested the system too. They provided feedback using forms and note taking, and were observed by the author to understand how easy or usable the interface was and how well the natural language segment performed.

Ultimately, this tapirs the solution as being entrenched, scalable, usable, leading to rich automation and equitable access for all.

IV. Results

The implementation of the proposed AI-powered web automation system was tested across multiple educational and job portals, including university result dashboards, internship application sites, Digilocker, and Naukri.com. A set of 50 unique tasks were defined, covering routine operations such as login, result retrieval, form filling, certificate downloads, and job searches. These tasks were executed by both the automated system and a group of human users for benchmarking.

A. Quantitative Results

The system achieved the following performance metrics:

- **Intent Recognition Accuracy:** The NLP module achieved an average accuracy of 92.3% in correctly mapping natural language instructions (e.g., "Check my results" or "Apply for internship") to the intended automation scripts.
- **Script Execution Success Rate:** Out of 50 tasks, 44 were executed successfully on the first attempt. This corresponds to an 88% success rate, with failures primarily caused by unexpected layout changes on dynamic websites.
- **Task Completion Time:** Compared to manual execution, the automation reduced average task completion time by 63.5%. For instance, job application tasks that took students around 3–4 minutes manually were completed in under 1 minute by the system.
- **User Effort Reduction:** Users required only 1–2 input steps (a natural language query or voice command), compared to 8–12 steps in manual execution, representing a reduction of nearly 75% in effort.
- **Error Rate:** The error rate was measured at 6%. Most errors occurred in tasks that involved multi-step navigation with CAPTCHA inputs or highly dynamic JavaScript-driven forms.

B. User Study and Feedback

A usability study was conducted with 20 student participants and 10 job seekers, none of whom had significant programming experience. Participants interacted with the system for a week and completed a survey afterward.

- **Ease of Use:** Participants rated the system at an average of 4.6/5, citing that it "felt like chatting with an assistant rather than coding a bot."
- **Time Savings:** On average, participants reported saving 15–20 minutes daily in repetitive tasks such as checking exam results or re-submitting applications.
- **Learning Curve:** Nearly 85% of participants reported that they could use the tool confidently after only one or two attempts, compared to weeks of practice usually needed for Selenium-based scripting.
- **Perceived Limitations:** A few users (approx. 15%) expressed frustration when tasks involved unpredictable pop-ups, security OTPs, or visual CAPTCHA, which the system could not yet bypass.

C. Comparative Analysis

When compared to traditional tools such as Selenium and Puppeteer, the proposed system demonstrated:

- 35–40% faster execution times on average.
- 20% higher task completion rate for non-technical users.
- Significantly lower setup complexity, as participants did not need to install drivers or write scripts.

D. Overall Findings

The results demonstrate that the system is effective in automating repetitive tasks across multiple domains while being accessible to non-technical users. Human feedback emphasized convenience, reduced effort, and improved confidence in managing online workflows. However, the study also highlighted areas for improvement, such as better handling of CAPTCHA, multi-factor authentication, and adaptation to rapidly changing dynamic websites.

Overall, the system shows strong potential for real-world deployment in educational and employment portals, bridging the gap between advanced automation technologies and end-user accessibility.

V. Conclusion

This project demonstrates how combining web automation apps and AI and Natural Language Processing (NLP) can result in a system to assist users with complex online tasks using natural spoken or written commands. It is specifically designed for non-technical users, such as students and job applicants, to prevent them from the coding knowledge and complicated tooling setup that often requires them to know how to workflow. The system removes manual work for tasks like filling out a web form, collecting data, or applying for jobs.

The system leverages powerful tools like Playwright to control web browsing, GPT-4 or BERT to scan what the user wants the tool to do, and LangChain to generate scripts on-demand. This gives the system great flexibility and the ability to manage a range of websites and changing web pages. The system's back end employs Node.js and Python components and Docker to run each task in its own safe and separate container. The system also employs WebSockets to communicate real-time updates to users while keeping them up-to-date.

Notable strengths of the system include ease of use, making it possible for a no-programmer to automate tasks without rising to the level of complexity normally associated with programming; scalability supporting different portals and workloads; security with containerization; and the ability to monitor tasks in an interactive and real-time way. Together, these strengths address the known weaknesses of currently available automation tools that too often enclose end-users into a rigid, domain specific approach that requires technical expertise.

With this project, we are delivering a future-ready, flexible solution that will democratize web automation as a means to enhance productivity and lessen the technical barrier that exists to web-based task execution. This project also provides the foundation to facilitate broader goals in education, employment and technical development to create equitable digital empowerment.

References

[1] T. R. Rashid et al., "Automating web tasks using natural language commands," *Journal of Web Engineering*, 2021.

[2] R. Hadi et al., “Natural language interface for web automation using Selenium,” *IEEE ICAI*, 2020.

[3] L. Zhang and Q. Liu, “Web automation through dialogue systems,” *ACM Computing Surveys*, 2022.

[4] S. Sharma and K. Jain, “Intent recognition using BERT,” *ICON*, 2021.

[5] J. Lee et al., “Natural language programming,” *IEEE TSE*, 2021.

[6] A. Dey and M. Triantafillou, “AI-based automation with Playwright,” *IEEE Software*, 2021.

[7] G. Singh and V. Batra, “Web navigation using AI agents,” *IUI*, 2020.

[8] M. Patel and S. Kumar, “Cross-domain web automation,” *IJCA*, 2021.

[9] K. Arora and A. Mehta, “Voice-controlled web automation,” *IEEE Access*, 2021.

[10] N. Li et al., “LLM4Jobs: Job data extraction using LLMs,” *arXiv:2309.09708*, 2023.

[11] H. Lai et al., “AutoWebGLM: Web navigating agents,” *arXiv:2404.03648*, 2024.