Agents4Gov: Privacy-Preserving Web Agents Using Open LLMs for Public Sector Tasks

João Victor de Castro Oliveira[1], Nathan Rufino[1], Silvio Levcovitz[1,3], Matheus Yasuo Ribeiro Utino[1],Fábio Lobato[1,4], Marcelo Turine[1,2], Solange Rezende[1], Antonio Fernando Lavareda Jacob Junior[5], Ricardo Marcondes Marcacini[1]

[1]Institute of Mathematics and Computer Sciences (ICMC) – University of São Paulo (USP)

[2]Faculty of Computing (FACOM) - Federal University of Mato Grosso do Sul (UFMS)

[3]Procuradoria-Geral da Fazenda Nacional (PGFN)

[4]Federal University of Western Pará (UFOPA)

[5]State University of Maranhão (UEMA)

agents4gov@icmc.usp.br

Abstract. *Effective public governance requires an organizational culture that values transparency, technical competence, and ethical behavior, while also addressing persistent challenges related to efficiency and compliance in public sector institutions. The digital transformation of the public sector, particularly through the integration of Large Language Models (LLMs), represents a strategic shift toward data-driven governance, process automation, and scalable innovation. These emerging technologies offer unprecedented opportunities to enhance inclusiveness, transparency, security, efficiency, accessibility, and citizen-responsive public services. However, while closed-source LLMs can automate digital tasks, their adoption is limited by concerns over privacy and cost. We present Agents4Gov, a privacy-focused and cost-effective framework for automating browser tasks using open-source LLMs. Built on Browser-Use, it combines a powerful model for planning with a lightweight model for execution, supports asynchronous task requests via email, and can run on scheduled intervals to avoid the need for always-on infrastructure. Evaluated on MiniWoB++, Agents4Gov with LLaMA 4 models achieves competitive results compared to traditional Browser-Use agents powered by GPT-4o-based LLMs.*

1. Introduction

The rapid advancement of digital technologies has significantly transformed public sector operations, particularly in the delivery of services to citizens. Governments worldwide are increasingly adopting digital transformation strategies to improve administrative efficiency, enhance service responsiveness, and strengthen transparency and accountability. This shift toward digital governance is driven by growing societal expectations for faster, more reliable, and more accessible public services. However, the implementation of such transformation remains fragmented across regions and policy domains, highlighting persistent structural and institutional challenges [Cordella and Bonina 2012,

Alhosani and Alhashmi 2024, Aryatama et al. 2024]. Additionally, excessive bureaucracy remains a persistent issue, with studies showing that over 78% of citizens perceive it as a major source of inefficiency[1].

In Brazil, public governance plays a important role as the framework of leadership, strategic planning, and oversight mechanisms that guide the formulation and implementation of public policies. It encompasses a set of best practices, processes, and institutional arrangements designed to ensure the effective, transparent, and accountable allocation of public resources, as well as the delivery of services aligned with the public interest [Brasil 2019]. According to the Organisation for Economic Co-operation and Development (OECD) and the Brazilian Federal Court of Accounts (TCU), public governance is a core element for ensuring that public resources are used efficiently, corruption is mitigated, and sustainable development is promoted. It requires the integration of ethical leadership and accountability into governance structures, together with robust mechanisms for oversight, control, and performance auditing [Nardes et al. 2018, Mergel et al. 2024].

Recent initiatives in Brazil demonstrate the potential of Artificial Intelligence (AI) to enhance public sector efficiency. For instance, the National Institute of Social Security (INSS) has implemented AI systems to automate the granting of social benefits[2]. Another example is the initiative of the Supreme Federal Court, which adopted an AI tool named MarIA[3] to assist in legal drafting and case management. In the context of digital service delivery, Melo [Melo 2024] proposes a methodology for integrating AI into public services through a structured approach that combines Lean Office, Design Sprint, and Large Language Models (LLMs). The methodology was applied to the Gov.br portal, where a generative agent was developed to assist citizens by answering questions and routing service requests. Regarding virtual agents, Scutella et al. [Scutella et al. 2024] explore how citizens derive value from interacting with such systems in public sector settings, reinforcing the importance of designing agent-based solutions that are not only technically competent but also socially and communicatively adaptive.

In this context, a new class of intelligent agents powered by LLMs is emerging to automate complex yet repetitive tasks [Wang et al. 2024]. These agents can interpret natural language instructions, plan actions, and interact with digital systems. Practical examples include policy analysis [Cao et al. 2024], generation of official documents such as declarations and certificates [Musumeci et al. 2024], and support in legal workflows by drafting or reviewing legal texts [Mamalis et al. 2024]. It is important to recognize that, despite the promise of digitalizing public services with LLM-based agents to improve efficiency, several limitations still hinder their widespread adoption. First, the high operational and licensing costs associated with proprietary LLMs pose significant challenges, especially for government agencies with limited resources. In addition, relying on external providers for model execution raises concerns about autonomy and the long-term sustainability of public digital infrastructure. Most critically, the use of LLMs in public service contexts often involves handling sensitive personal data, which introduces privacy risks. This practice can also conflict with data protection regulations such as Brazil's Gen-

[1]https://www.fiesp.com.br/indices-pesquisas-e-publicacoes/pulsos/pulso_burocracia_versao-3/

[2]https://policyreview.info/articles/analysis/balancing-efficiency-and-public-interest-ai

[3]https://noticias.stf.jus.br/postsnoticias/stf-lanca-maria-ferramenta-de-inteligencia-artificial-que-dara-mais-agilidade-aos-servicos-do-tribunal/

eral Data Protection Law (LGPD), which sets strict rules on how sensitive information should be collected, stored, and processed. These challenges raise a research question: *how can we design LLM-based agents that ensure service efficiency while guaranteeing privacy, legal compliance, and feasibility for local deployment in the public sector?*

In this paper, we introduce *Agents4Gov*, a privacy-preserving framework for autonomous browser agents powered by open-source LLMs. In the context of accelerating digital transformation, public institutions are increasingly pressured to modernize their internal operations and service delivery models. Thus, this work is motivated by the growing need in public institutions to automate internal operations that are repetitive, time-consuming, and of low to medium complexity tasks, such as consulting multiple government systems, extracting data, downloading reports from portals, completing structured digital forms, submitting routine requests, issuing certificates or declarations, and navigating step-by-step governmental platforms, to name just a few. These tasks often burden administrative staff, reducing the time available for higher-level decision-making and generating dissatisfaction with the delay in responding to the public. Our contributions are manyfold, including:

- a hybrid approach that separates the use of LLMs according to the agent's operational stages: high-capacity models are used for planning and evaluation, while lightweight models handle browser interactions, reducing computational cost without significantly compromising performance;
- an asynchronous execution pipeline that allows agents to process task requests via institutional email, thereby reducing infrastructure demands and avoiding constant GPU usage;
- a comparative analysis of our framework against commercial and open-source LLM agents, achieving competitive results while preserving sensitive data by using open-source models deployable in local environments;
- the release of the current version of *Agents4Gov* as an open-source project, publicly available at `https://github.com/LABIC-ICMC-USP/agents4gov`.

We carried out an experimental evaluation of Agents4Gov using 55 tasks from the MiniWoB++ benchmark. We compared four configurations: GPT-4o, LLaMA 4 Maverick, LLaMA 4 Scout, and our hybrid approach, Agents4Gov, which uses LLaMA 4 Maverick for planning and LLaMA 4 Scout for browser execution. Task completion success rates were: GPT-4o (0.964), Maverick (0.891), Scout (0.673), and *Agents4Gov* (0.818), showing a good balance between performance and efficiency using only local, open-source models.

2. Background and Related Work

Agents are computational entities capable of perceiving an environment, reasoning about goals, and executing actions to accomplish tasks [Wooldridge and Jennings 1995]. In our context, agents are systems capable of perceiving their environment through sensors, processing information to make decisions, and acting upon the environment to achieve specific objectives [Wooldridge 2009]. A common formulation in the literature defines an agent as a tuple (P, E, A), where P represents the perception function that captures the

current state of the environment, E is the decision-making or planning function that maps states and goals into actions, and A is the set of possible actions executable by the agent.

Recent advances in LLMs have greatly expanded the potential of autonomous agents [Li et al. 2024]. These models enable agents to interpret user instructions, decompose them into actionable steps, and interact directly with environments that contain textual elements, such as HTML-based pages. This paper focuses on LLM-based web-use agents, a class of autonomous systems designed to perform complex tasks in web browsers [Müller and Žunič 2024, OpenAI 2025, S. B. LLC 2025, Browserbase 2024].

These agents use LLMs to understand instructions, plan actions, and interact with web content through clicks, form filling, and navigation. Unlike traditional rule-based scripts, they can adapt to dynamic layouts and changing content, combining natural language understanding, strategic planning, and interaction browsers [Hu et al. 2024]. Systems such as Proxy [Convergence 2025] and Operator [OpenAI 2025] have achieved high performance on interactive web benchmarks like WebVoyager, surpassing 85% task completion in many cases [Mudryi et al. 2025]. Proxy, developed by Convergence AI, emphasizes action-oriented execution by enabling agents to perform real-time interactions on web interfaces, such as clicking, scrolling, and navigating pages. The Operator, in turn, integrates GPT-4o with a specialized model known as the Computer-Using Agent (CUA), trained via reinforcement learning to interact directly with Graphical User Interfaces (GUIs) similarly to human users.

We are particularly interested in open-source alternatives, such as *Browser Use* [Müller and Žunič 2024]. Its architecture integrates reasoning and planning in a unified feedback loop that continuously refines actions based on the current browser state. The agent operates primarily through Document Object Model (DOM) extraction, providing structured access to page elements for the language model. Although *Browser Use* is open-source, its current implementation offers limited support for local LLMs, and the use of different LLMs across agent stages remains underexplored. In this paper, we build on the *Browser Use* architecture as the foundation for Agents4Gov, extending it with privacy-preserving adaptations tailored to public sector scenarios. Our approach explores the use of local LLMs with a separation of agent stages, allowing using more sophisticated models (usually, proprietary) for the planning and reasoning phase, while assigning lighter models to handle browser tool execution and page interaction.

3. Agents4Gov Framework

We introduce the proposed Agents4Gov framework in this section. The overall process is illustrated in Figure 1, which summarizes how user requests are received, interpreted, executed through planning and browser actions, and returned as structured email responses.

Following the workflow given in 1, a user request is submitted via email, which is interpreted and decomposed into subgoals by the planner model M_p. The executor model M_e performs each subgoal in the browser environment. After completion, M_p validates the result and generates a final response returned via email. In the next sections, we first formally define the problem addressed by the framework (Section 3.1). Then, we describe deployment strategies designed to ensure both privacy and cost efficiency (Section 3.2). Finally, we detail the agent's execution loop, explaining how the planning and execution models interact, and present examples of the system prompts used to guide the agent's

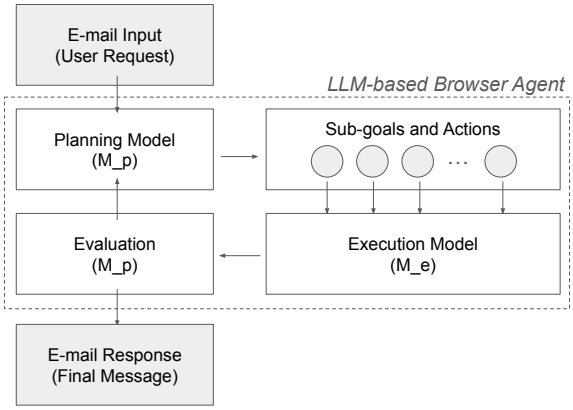

Figure 1. Overview of the Agents4Gov framework.

behavior (Section 3.3).

3.1. Problem Definition

We define the task of a browser-based LLM agent as the problem of mapping a natural language instruction into a sequence of browser actions that complete a goal-oriented interaction on a web interface.

Let $\mathcal{T}$ denote the set of all possible tasks expressed in natural language, and let $t \in \mathcal{T}$ represent a specific instruction, and let E denote the observable state of the environment, typically represented by the DOM tree of a web page. The agent must produce a sequence of browser actions $\pi = (a_1, a_2, \ldots, a_k)$, where each $a_i \in \mathcal{A}$ is a low-level interaction such as a click, text input, or navigation.

To structure this process, we decompose the agent into two components: a planner model, M_p, and an executor model, M_e. The planner M_p receives the instruction t and the initial environment E, generating a high-level plan $\hat{\pi} = (s_1, s_2, \ldots, s_k)$, where each s_i is a natural language description of an intended sub-action. Then, for each s_i, the executor M_e observes the current DOM state E_i and maps the sub-action s_i into a concrete browser command a_i. As formalized in Equations 1 and 2, the process consists of:

$$\hat{\pi} = M_p(t, E) \tag{1}$$

$$a_i = M_e(s_i, E_i), \quad \text{for each } s_i \in \hat{\pi} \tag{2}$$

The final execution trace $\pi = (a_1, \ldots, a_k)$ is applied to the environment, dynamically updating the DOM state at each step. After executing all actions, the planner model M_p re-evaluates the final environment state E_k, along with the original instruction t, to assess whether the task objective has been successfully completed. The agent is considered successful if this final evaluation indicates that the goal described in t has been achieved in the current state of the browser.

3.2. Privacy and Cost-Efficient Deployment

Our *Agents4Gov* framework is designed to run entirely with local models. This approach avoids sending sensitive data to external providers. However, due to the high cost of

running LLMs, especially given their dependence on GPU acceleration, the framework assumes the use of models with fewer parameters, organized into two stages. A more capable model is used for planning and evaluation, while a lighter model is used during interaction with the browser.

In this sense, the planner M_p (the more capable model) is responsible for understanding the task, generating a strategy, and verifying whether the objective has been reached. The executor M_e (the lighter model) handles interactions with the browser, such as clicking or filling forms. The model M_p requires more reasoning capacity, but it is used only at key moments. It runs at the beginning to produce a plan and at the end to evaluate the result. M_e is simpler and used repeatedly. It runs at each step of the interaction, processing one command at a time based on the current page state.

As part of the privacy-preserving and security strategy, task requests are submitted to the *Agents4Gov* via email. Thus, the agent is linked to an institutional email account created specifically for this purpose. This approach ensures that access control and identity verification remain under the institution's domain. An additional advantage is that the organization can schedule the agent to run at specific times of the day, processing incoming emails in batches and executing the corresponding tasks. This scheduled execution reduces the need for continuous GPUs resource usage, allowing the service scalability. After each execution, the agent sends a reply by email with the task's result, indicating whether it was completed successfully or not.

3.3. Agent Execution Model

Our *Agents4Gov* framework is structured around an automated loop that connects to an institutional email inbox using the IMAP protocol. This loop periodically monitors the inbox for unread messages. For each new message, the email body is extracted and interpreted as a user request R, which triggers a new execution session for the agent. This execution session instantiates a browser automation agent based on the architecture proposed by the *Browser-User* project [Müller and Žunič 2024], as described in Section 2. The agent is guided by a structured system prompt that defines its behavior, internal reasoning loop, and output format. The prompt is loaded from a dedicated markdown template within the framework, as illustrated in Figure 2.

```
You are an AI agent designed to operate in an iterative loop to automate browser tasks.
     Your ultimate goal is accomplishing the task provided in user request R.
```

Figure 2. Excerpt of the system prompt of the browser automation agent.

At every step t, the agent receives a state $S_t = (R, H_{1:t-1}, E_t)$, where R is the user request derived from the email, $H_{1:t-1}$ is the chronological history of previous actions and memory updates, and E_t is the current browser environment state. The reasoning process is decomposed into planning and execution stages. The planning model M_p is invoked with a dedicated planning prompt as shown in Figure 3.

Given the setup presented in Figure 3, the planning model M_p produces a plan according Equation 1 and for each subgoal $s_i \in \hat{\pi}$, the execution model M_e selects and performs an action a_i, guided by the current browser state and execution context as defined

```
You are a planning agent that helps break down tasks into smaller steps and reason
    about the current state.
Your role is to:
1. Analyze the current state and history
2. Evaluate progress towards the ultimate goal
3. Identify potential challenges or roadblocks
4. Suggest the next high-level steps to take

Your output format should always be a JSON object:
{
  "state_analysis": "...",
  "progress_evaluation": "...",
  "challenges": ["..."],
  "next_steps": ["...", "..."],
  "reasoning": "..."
}
```

Figure 3. Excerpt of the planning prompt provided to model M_p to generate structured next steps based on the current state and task history.

in Equation 2. At each step, the agent receives a structured summary of its current state, including the user request from the email, the internal task plan generated earlier, the current step number, the visible browser elements, and a log of previous actions and outcomes. It also includes any recently extracted data from web pages or files. This structured input is used by the execution model M_e to select the next action based on the current context and progress.

The execution continues until the agent completes the task or reaches a stopping condition (e.g., time limit or number of attempts). At the end, M_p is reinvoked to evaluate final success and summarize the execution. The final message follows a structured JSON format as illustrated in Figure 4.

```
{
  "action": [{
    "done": {
      "success": true or false,
      "text": "Summary of the completed task"
    }
  }]
}
```

Figure 4. Final output message summarizing task execution and success status, as produced by the planning model M_p.

The final output is automatically formatted into a reply email, which includes a natural language explanation, selected excerpts from the execution history, and a success indicator. Finally, the email is marked as read in the IMAP system, completing the cycle and freeing the loop to process the next incoming task.

4. Experimental Evaluation

We evaluate the performance of the *Agents4Gov* framework in solving real-world browser automation tasks. Our focus is on assessing whether the proposed hybrid strategy which utilizes separate models for planning M_p and execution M_e can maintain high task success

rates while relying entirely on open-source models. To this end, we compare Agents4Gov with a top-line setup that uses the proprietary GPT-4o model from OpenAI for both planning M_p and execution M_e. We also include two simpler baselines to analyze the impact of each component on the agent's overall performance.

4.1. Dataset

We conduct our experiments using the MiniWoB++ benchmark, a suite of web-based tasks. Each environment simulates a specific browser-based task, such as filling out a form, selecting from a drop-down, or completing multi-step interactions. In our evaluation, we selected 55 tasks from the benchmark that cover a broad range of interaction patterns. These tasks require agents to understand user instructions, extract relevant information from the interface, and execute the correct sequence of browser actions to complete the task.

Table 1. Distribution of the tasks by interaction type and task complexity.

Task Type	Task Complexity			
	Low	**Medium**	**High**	**Total**
Authentication	-	1	2	3
Auto-complete Interaction	1	1	-	2
Click Interaction	5	3	-	8
Data Lookup	1	-	3	4
Email Management	-	-	2	2
List/Option Selection	2	-	1	3
Multi-Step Interaction	-	-	3	3
Random/Choice Interaction	-	1	1	2
Reasoning/Computation	2	1	2	5
Simulated Transaction	-	-	1	1
Tab/Menu Navigation	8	3	1	12
Terminal Commands	-	-	1	1
Text Input & Manipulation	7	1	1	9
	26	11	18	55

Table 1 shows details of the 55 tasks used in our evaluation. Each task was analyzed and categorized by interaction type and complexity level. Complexity levels were manually assigned based on the number of required steps and the need for memory or multi-element coordination. This categorization allows us to analyze results across several scenarios, ranging from simple click-based actions to more complex multi-step workflows and reasoning challenges.

Figure 5 represents one of the tasks following the typical structure of the MiniWoB++ benchmark, where the task is embedded within the page content and requires goal-oriented interactions. In this example: (1) the agent identifies the task description highlighted in yellow; (2) it extracts the keyword and performs the corresponding search; (3) upon realizing that the desired result is in the 9th position and only three items are displayed per page, it handles pagination accordingly; and (4) the agent locates and clicks the correct result, completing the task.

We highlight that, although the MiniWoB++ dataset has been widely used in prior studies on browser automation, to the best of our knowledge, this is the first study to

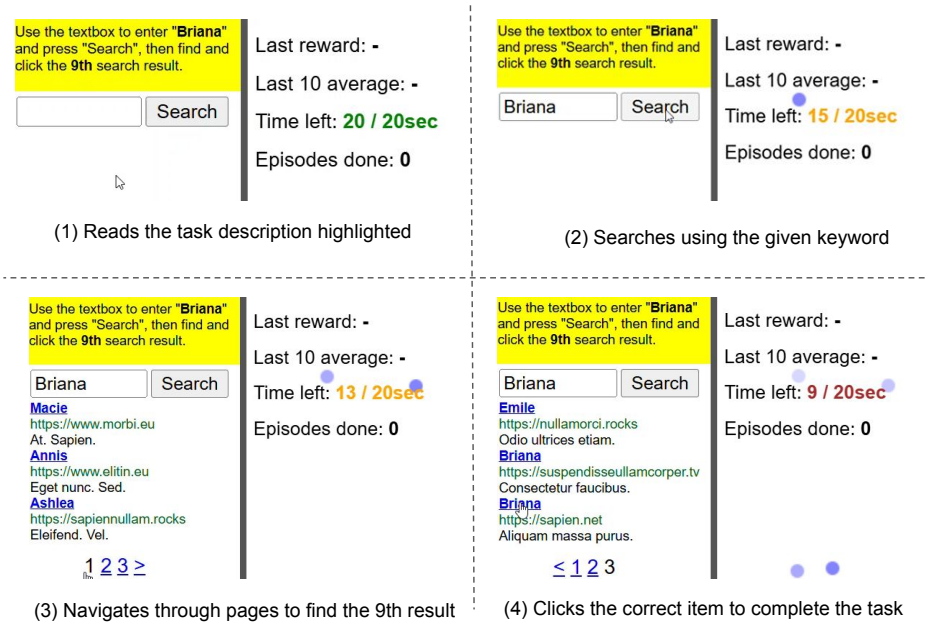

Figure 5. Example of a standard MiniWoB++ task.

evaluate web automation tasks by directly comparing different LLM-based agents.

4.2. Experimental Setup

To establish a performance reference, we defined a top-line model based on OpenAI's proprietary GPT-4o (representing traditional Browser-Use). Ignoring constraints such as asynchronous task submission via email, privacy concerns, or computational cost, GPT-4o, combined with the browser-user interface, represents a strong benchmark, as it is already integrated into a browser-use agent [Müller and Žunič 2024].

The Agents4Gov framework, in contrast, is fully based on open models. For the planning model M_p, we use LLaMA 4 Maverick, a 17B active parameter MoE model with 128 experts. For the execution model M_e, we use LLaMA 4 Scout, another 17B-parameter model, but with 16 experts. Since both models are open and have their weights publicly released, we can apply quantization techniques to reduce memory usage, allowing deployment on NVIDIA H100 GPUs or smaller devices, depending on the quantization level. Additionally, for ablation purposes, we evaluate two agents, each configured with a single model: one fully based on LLaMA 4 Maverick and the other on LLaMA 4 Scout.

Finally, as our evaluation metric, we use the task completion rate, which measures the percentage of tasks successfully completed by each agent.

4.3. Experimental Results

We first present the overall task completion rates achieved by each evaluated agent configuration. The GPT-4o agent achieves the highest success rate, completing 96.4% of the tasks. The LLaMA 4 Maverick agent, which uses the same model for both planning and execution, achieves a success rate of 89.1%. Our proposed *Agents4Gov* configuration, which combines Maverick for planning and Scout for execution, achieves a completion rate of 81.8% of the tasks. Finally, the agent using LLaMA 4 Scout in both stages performs the worst, with a success rate of 67.3%.

To better understand the agent behaviors across different contexts, Table 2 presents the task completion rates by task category. We observe that *Agents4Gov* matches the performance of GPT-4o in categories such as *Click Interaction*, *Email Management*, *Simulated Transaction*, *Tab/Menu Navigation*, and *Text Input & Manipulation*, indicating that its hybrid configuration is effective for direct and structured interactions. However, Agents4Gov showed weaker results in more dynamic scenarios such as *Autocomplete Interaction* and *Random/Choice Interaction*, due to difficulties in how the executor module (M_e) interprets and adapts to the instructions generated by the planner module (M_p)

Table 2. Task completion rates per interaction category for GPT-4o, LLaMA 4 Maverick, LLaMA 4 Scout, and the hybrid *Agents4Gov* setup.

Category	GPT-4o	Maverick	Scout	Agents4Gov
Authentication	100%	67%	67%	67%
Autocomplete Interaction	100%	100%	50%	0%
Click Interaction	100%	100%	63%	100%
Data Lookup	100%	100%	0%	75%
Email Management	50%	100%	50%	100%
List/Option Selection	100%	67%	33%	67%
Multi-Step Interaction	67%	100%	67%	67%
Random/Choice Interaction	100%	100%	100%	50%
Reasoning/Computation	100%	80%	80%	80%
Simulated Transaction	100%	100%	100%	100%
Tab/Menu Navigation	100%	83%	83%	100%
Terminal Commands	100%	0%	0%	0%
Text Input & Manipulation	100%	100%	89%	89%

Table 3 presents the task completion rates grouped by complexity level. All agents performed well on low-complexity tasks, with Agents4Gov achieving 92% accuracy, slightly below the topline GPT-4o and the Maverick-only configuration. For medium-complexity tasks, Agents4Gov maintained an 82% success rate, matching Maverick and significantly outperforming Scout, which dropped to 45%. On high-complexity tasks, performance drops across all agents, with Agents4Gov completing 67% of the tasks.

Table 3. Task completion rates grouped by task complexity level.

Complexity	GPT-4o	Maverick	Scout	Agents4Gov
Low	100%	96%	88%	92%
Medium	100%	82%	45%	82%
High	89%	83%	50%	67%

These results demonstrate that our Agents4Gov performs well in tasks of low and medium complexity, reaching success rates close to those of the best-performing agents. In more complex tasks, we observed that even with a high-capacity planning model, the execution model sometimes failed to follow the subgoal instructions, particularly when the instructions were long or referred to elements from previous interactions. This is the main limitation identified, suggesting that improving the alignment between planning and execution is an important next step. Despite this, we found that more detailed user instructions can help reduce such failures. Since we kept the original MiniWoB++ task descriptions unchanged, this strategy was not applied in this experimental evaluation.

5. Concluding Remarks

In this paper, we presented the Agents4Gov framework, designed to support language model-based agents for automation in the public sector. The main goal was to create an agent that does not rely on proprietary APIs, using only open LLMs to reduce privacy risks, ensure auditability (i.e., allowing third-party inspection, which is essential in public service contexts), and lower operational costs.

Our strategy separates the agent into two models: one for planning and another for execution. This hybrid approach allows the selection of models suited to each task. In Agents4Gov, we utilize cost-efficient open models for execution, while more powerful models are reserved for planning and evaluation. The experimental results showed that Agents4Gov performs well in tasks of low and medium complexity, remaining a viable solution for many real-world scenarios. However, tasks with higher complexity revealed some limitations, especially when the execution model needs to interpret long instructions or handle multiple steps based on prior interactions.

As future work, we plan to explore the use of smaller models as they continue to improve, aiming to reduce memory demands and expand deployment possibilities. Another direction is to fine-tune both the planning and execution models to improve performance in complex workflows.

Acknowledgment: This work was supported by the National Council for Scientific and Technological Development (CNPq) – PQ-316507/2023-7, DT-303031/2023-9, POSDOC-101057/2024-5; and the São Paulo Research Foundation (FAPESP) – 2023/10100-4.

References

Alhosani, K. and Alhashmi, S. M. (2024). Opportunities, challenges, and benefits of ai innovation in government services: a review. *Discover Artificial Intelligence*, 4(1):18.

Aryatama, S., Miswan, M., Fahriyah, F., Pribadi, T., and Suacana, I. (2024). Enhancing governance efficiency through digital transformation in public services: Lessons from global practices. *Global International Journal of Innovative Research*, 2(5):1019–1027.

Brasil (2019). Decreto nº 9.901, de 8 de julho de 2019: Altera o decreto nº 9.203, de 22 de novembro de 2017, que dispõe sobre a política de governança da administração pública federal direta, autárquica e fundacional. Diário Oficial da União.

Browserbase (2024). Open operator: A template for building web agents with stagehand on browserbase. `https://github.com/browserbase/open-operator`. Accessed: Jun 20, 2025.

Cao, C., Zhuang, J., and He, Q. (2024). Llm-assisted modeling and simulations for public sector decision-making: Bridging climate data and policy insights. In *AAAI-2024 Workshop on Public Sector LLMs: Algorithmic and Sociotechnical Design*.

Convergence (2025). Convergence's proxy ahead in top agent benchmark, beats openai and anthropic. `https://convergence.ai/introducingproxy/`. Accessed: Jun 20, 2025.

Cordella, A. and Bonina, C. M. (2012). A public value perspective for ict enabled public sector reforms: A theoretical reflection. *Government information quarterly*, 29(4):512–520.

Hu, X., Xiong, T., Yi, B., Wei, Z., Xiao, R., Chen, Y., Ye, J., Tao, M., Zhou, X., Zhao, Z., et al. (2024). Os agents: A survey on mllm-based agents for computer, phone and browser use.

Li, X., Wang, S., Zeng, S., Wu, Y., and Yang, Y. (2024). A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1(1):9.

Mamalis, M. E., Kalampokis, E., Fitsilis, F., and Tarabanis, K. (2024). A large language model agent based legal assistant for governance applications. In *International Conference on Electronic Government*, pages 286–301. Springer.

Melo, M. K. (2024). Implementing AI for enhanced public services gov.br: A methodology for the brazilian federal government. In *Proceedings of the 20th International Conference on Web Information Systems and Technologies, WEBIST 2024, Porto, Portugal, November 17-19, 2024*, pages 90–101.

Mergel, I., Dickinson, H., Stenvall, J., and Gasco, M. (2024). Implementing ai in the public sector. *Public Management Review*, pages 1–14.

Mudryi, M., Chaklosh, M., and Wójcik, G. (2025). The hidden dangers of browsing ai agents. *arXiv preprint arXiv:2505.13076*.

Musumeci, E., Brienza, M., Suriani, V., Nardi, D., and Bloisi, D. D. (2024). Llm based multi-agent generation of semi-structured documents from semantic templates in the public administration domain. In *International Conference on Human-Computer Interaction*, pages 98–117. Springer.

Müller, M. and Žunič, G. (2024). Browser use: Enable ai to control your browser.

Nardes, J. A. R., Altounian, C. S., and Vieira, L. A. G. (2018). Governança pública. *O desafio do Brasil*, 3.

OpenAI (2025). Introducing operator. `https://openai.com/index/introducingoperator/`. Accessed: Jun 20, 2025.

S. B. LLC (2025). Do browser: Ai browser automation agent. `https://www.dobrowser.io/`. Accessed: Jun 20, 2025.

Scutella, M., Plewa, C., and Reaiche, C. (2024). Virtual agents in the public service: examining citizens' value-in-use. *Public Management Review*, 26(1):73–88.

Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., et al. (2024). A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Wooldridge, M. (2009). *An introduction to multiagent systems*. John wiley & sons.

Wooldridge, M. and Jennings, N. R. (1995). Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(2):115–152.