INTERNATIONAL JOURNAL OF ECONOMIC SOCIAL SCIENCE AND MANAGEMENT LAW
Peer Reviewed, Referred & Indexed Journal
E-ISSN: 3070-0558
VOL.7, NO.2(1) (2026)

Srota AI: A Voice-Controlled Intelligent Web Automation Platform Using Large Language Models and Multi-Agent Orchestration

Mr.Shuvendu Samal
Student, Dept. of CSE
GIFT Autonomous, Bhubaneswar

Mr.Biswajit Swain
Student, Dept. of CSE
GIFT Autonomous, Bhubaneswar

Prof Biswajit Sahoo
Project Guide, Dept. of CSE
, GIFT Autonomous

***Abstract*—The rapid proliferation of complex, multi-step web-based applications across government, healthcare, education, and enterprise domains has introduced significant usability barriers for diverse user populations. Traditional web automation tools rely on brittle, static rule-based scripting that is inaccessible to non-technical users and fails to adapt to dynamic web interfaces. This paper presents Srota AI, a voice-controlled intelligent web automation platform that leverages large language models (LLMs), multi-agent orchestration via LangGraph, and real-time voice transcription through the Deepgram API to enable users to complete complex digital workflows using natural language or spoken commands. The system architecture integrates a Planner Agent for goal decomposition, a Navigator Agent for semantic DOM interaction, a Voice Assistant Module for hands-free operation, and a Semantic DOM Processing Module for context-aware UI navigation. The backend, developed in Python with FastAPI, supports asynchronous concurrent session management and integrates with Google Gemini for AI reasoning. A plug-and-play React SDK enables seamless embedding into third-party web applications. Evaluation results demonstrate significant reductions in task completion time and error rates across real-world automation scenarios, including government form submission, healthcare appointment scheduling, and enterprise onboarding. The system establishes a replicable, extensible framework for applying agentic AI and voice recognition to accessible, intelligent web automation.**

***Index Terms*—Voice Assistant, Web Automation, Large Language Models, Multi-Agent AI, LangGraph, Natural Language Processing, DOM Interaction, Accessibility, FastAPI, Google Gemini**

I. INTRODUCTION

The accelerating digital transformation of public and private services has made web-based applications the primary interface for accessing essential activities, including government form submission, medical appointment scheduling, financial transactions, and enterprise onboarding. While these platforms dramatically expand the reach of services, they simultaneously introduce significant usability challenges. Users are required to navigate multi-page workflows, interpret complex validation rules, manage conditional field logic, and perform repetitive data entry—tasks that are time-consuming, error-prone, and cognitively demanding [1].

These challenges are especially pronounced for individuals with physical disabilities, limited digital literacy, or those operating in environments where manual keyboard and mouse interaction is impractical. Conventional mitigation tools, including browser autofill, macro recorders, and Robotic Process Automation (RPA) platforms, offer only partial relief. They depend on static rules, require technical expertise to configure, and are fragile in the face of dynamic or evolving web interfaces [2].

The emergence of large language models (LLMs) with strong instruction-following and contextual reasoning capabilities offers a transformative opportunity to address these shortcomings. Recent work has demonstrated LLM effectiveness in multi-step task planning and natural language understanding [3], while frameworks such as LangGraph enable the orchestration of complex, stateful multi-agent workflows [4]. Integration with real-time speech recognition further enables hands-free, voice-driven interaction.

This paper presents **Srota AI**, a voice-controlled intelligent web automation platform that unifies LLM-based multi-agent reasoning, real-time voice transcription, and semantic DOM interaction into a cohesive, accessible system. The platform enables users to complete intricate digital workflows simply by expressing their intent in natural language or spoken commands, without requiring technical knowledge or manual navigation.

The core contributions of this work are:

- A multi-agent AI pipeline for end-to-end intelligent web automation orchestrated via LangGraph, comprising a Planner Agent and a Navigator Agent.
- A real-time voice interaction layer powered by Deepgram speech-to-text, enabling hands-free control of complex web workflows.
- A Semantic DOM Processing Module that interprets web page structure through a contextual, coordinate-independent lens for robust UI navigation.
- A plug-and-play React SDK (ReasonIQ Chat Widget) enabling zero-configuration embedding into third-party

Received: 20-04-2026 | Accepted: 28-05-2026 | Published: 04-06-2026 | www.ijeml.com

web applications.
- A stateful session management architecture supporting pause, resume, and atomic transaction rollback across multi-step workflows.
- Evaluation of the system across real-world scenarios demonstrating measurable improvements in task comple-tion efficiency and error reduction.

The remainder of this paper is organized as follows: Sec-tion II reviews related work, Section III describes the system architecture, Section IV details the core modules, Section V presents implementation specifics, Section VI reports evalu-ation results, Section VII discusses findings and limitations, and Section IX concludes with future directions.

II. Related Work

A. AI-Based Web Automation

Traditional web automation has been dominated by tools such as Selenium, iMacros, and AutoHotkey, which rely on static scripts and recorded action sequences [2]. While effec-tive for simple, repetitive tasks with stable interfaces, these tools are brittle and require significant technical expertise. The introduction of AI-driven approaches has enabled more adaptive and robust automation. Modern systems leverage LLMs and NLP to interpret unstructured user instructions, transforming automation from a scripting activity into a natural language interaction [3]. Frameworks such as LangChain and LangGraph provide the infrastructure to integrate LLMs into stateful multi-step workflows, enabling dynamic task decom-position and adaptive execution [4].

B. Voice-Enabled Interfaces and Digital Assistants

Voice-enabled digital assistants, exemplified by Amazon Alexa, Google Assistant, and Apple Siri, have demonstrated the broad viability of natural spoken language as a primary interaction modality [5]. These systems combine automatic speech recognition (ASR), NLP, and contextual reasoning to interpret and execute spoken commands in real time. In the context of web automation, voice interfaces offer signif-icant accessibility benefits, particularly for users with motor impairments or those in hands-busy environments. However, existing general-purpose voice assistants are typically limited to simple, predefined commands and lack the capacity to navigate and interact autonomously with complex, dynamic web applications [6].

C. Intelligent Form Filling and Workflow Automation

Research in intelligent form filling has evolved from early rule-based autofill systems to ML-driven approaches capable of analyzing form structure, inferring semantic field types, and adapting to dynamic content [7]. RPA platforms such as UiPath and Automation Anywhere have extended automation to multi-application business processes but typically require expert configuration. Recent work has applied LLMs to gen-erate structured plans for web navigation tasks, with agents reasoning over DOM representations to determine optimal UI interactions [8]. Srota AI advances this direction by integrating voice input, multi-agent orchestration, and semantic DOM analysis into a unified, accessible platform requiring no user configuration.

D. Accessibility in Digital Systems

Accessibility in web systems is governed by standards such as WCAG 2.1 [9], which establish requirements for key-board navigability, screen reader support, and alternative input modalities. Prior work has demonstrated that conventional web applications frequently fail to meet these standards, creating barriers for users with disabilities [1]. Voice-driven automation represents a compelling accessibility-first design paradigm, and Srota AI's architecture directly addresses this by making complex web workflows executable through natural spoken language.

III. System Architecture

A. Overview

Srota AI is designed as a modular, multi-layer system. The high-level architecture comprises four primary layers: (1) a user interaction layer delivering voice and text interfaces through the ReasonIQ Chat Widget, (2) an AI orchestration layer built with LangGraph and LangChain managing the multi-agent pipeline, (3) a backend API layer developed in Python with FastAPI providing asynchronous RESTful and WebSocket endpoints, and (4) an external services layer in-tegrating Deepgram for ASR and Google Gemini for LLM reasoning.

B. LangGraph Orchestration and State Management

The intelligence of Srota AI rests on a stateful multi-agent orchestration layer implemented using LangGraph. Un-like simple sequential pipelines, LangGraph defines a cyclic directed graph in which each node represents a specialized agent or processing step, and edges encode conditional data flow logic. A shared State object persists throughout the pipeline, tracking the current URL, history of completed field interactions, remaining planned steps, and session metadata.

Key nodes in the graph include:
- *Intent Parser Node*: Receives transcribed text from Deep-gram and classifies user intent as either an information query (routed to a QA LLM) or an action request (routed to the Planner Agent).
- *Planner Agent Node*: Decomposes the high-level user goal into a JSON-formatted sequence of sub-tasks using Google Gemini.
- *Navigator Agent Node*: Analyzes the target page DOM to determine optimal UI interaction selectors and executes actions.

Conditional edges implement routing logic, including: by-passing the Navigator for pure informational queries, escalat-ing to the user when clarification is required, and triggering Human-in-the-Loop (HITL) pauses for high-stakes actions such as financial transactions.

Received: 20-04-2026 | Accepted: 28-05-2026 | Published: 04-06-2026 | www.ijeml.com

C. Backend API Architecture

The FastAPI backend exposes a suite of asynchronous endpoints supporting the full automation lifecycle. Core API components include:

- `POST /task/create`: Accepts user intent and returns a unique Task ID for session tracking.
- `WS /voice/stream`: A persistent WebSocket connection handling real-time audio-to-text-to-action processing with low latency.
- `GET /task/status/{id}`: Provides live status updates on the current automation step to the front-end widget.

FastAPI's asynchronous event loop allows the server to manage thousands of concurrent voice streams and automation sessions simultaneously without thread blocking, ensuring responsiveness across all active users.

D. Frontend SDK

The ReasonIQ Chat Widget is a floating, embeddable React component packaged as the `reasoniq-chat-widget` SDK. It provides a unified interface for text and voice input, real-time task progress updates, and seamless session management. Built with React, TypeScript, Vite, and Tailwind CSS, the widget supports accessibility features including keyboard navigation, ARIA labels, and screen reader compatibility. Integration into any third-party web application requires minimal configuration, making advanced AI-powered automation accessible to developers without ML expertise.

IV. CORE MODULES

A. User Task Registration Module

The User Task Registration Module serves as the entry point for automation workflows. It accepts user intent expressed through either typed natural language or spoken commands captured via the ReasonIQ Chat Widget. For voice input, the module invokes Deepgram's real-time ASR API to convert speech to text with high accuracy and low latency. The module then leverages Google Gemini through LangChain to perform intent classification, context extraction, and parameter identification. The resulting structured task object—encapsulating the user goal, host application metadata, and session parameters—is passed to the Planner Agent for decomposition.

User authentication is handled through JWT-based authentication tokens, ensuring that each task is securely associated with the correct user session. The module supports multi-turn clarification dialogues, allowing the system to prompt users for additional information when the initial request is ambiguous.

B. Voice Assistant Module

The Voice Assistant Module integrates Deepgram's real-time speech-to-text transcription technology to provide hands-free interaction with the platform. The module supports a wide range of accents, dialects, and languages, incorporating noise reduction and speaker adaptation to maintain recognition accuracy across diverse environments.

Beyond transcription, the module enables multi-turn conversational interaction, retaining context across successive utterances to support natural follow-up commands such as "change that to next Tuesday" or "go back to the previous step." The module generates real-time audio feedback to users, confirming recognized commands and prompting for clarification when confidence is below threshold. Integration with the ReasonIQ Chat Widget enables seamless switching between voice and text input modalities during a single session.

C. Planner Agent

The Planner Agent is responsible for decomposing a high-level user goal into an ordered sequence of executable subtasks. The agent constructs a structured prompt containing: the user's stated goal, contextual metadata about the target web application, and few-shot examples of complex form interaction plans. This prompt is submitted to Google Gemini, which returns a JSON-formatted plan specifying each step as a typed action object (e.g., `click`, `fill`, `select`, `upload`, `navigate`).

The planner prompt is engineered with few-shot examples to minimize hallucinated actions and ensure plan feasibility. Conditional task branching is supported, allowing the agent to generate alternative execution paths based on anticipated runtime conditions such as optional form fields or dynamic dropdowns.

D. Navigator Agent and Semantic DOM Processing

The Navigator Agent interprets the Document Object Model (DOM) of the target web page through a semantic lens to determine optimal UI interaction strategies. Using specialized parsers, the module filters out irrelevant DOM elements (advertising scripts, tracking pixels, decorative elements) and focuses on *Actionable Nodes* identified by their semantic role, ARIA labels, and visible text content.

A key design principle is *coordinate-independent interaction*: elements are identified by their semantic function and label rather than fixed screen coordinates, making the system resilient to responsive design layout shifts and dynamic page reflows. When the user requests an action such as "pay the bill," the Navigator Agent probabilistically ranks candidate elements labeled "Checkout," "Pay Now," "Submit Payment," or similar, selecting the highest-confidence match.

The Navigator Agent interacts with the DOM through the automation execution sandbox, which enforces security constraints preventing unauthorized cross-site requests or access to restricted browser resources.

E. Automation and Execution Module

The Automation and Execution Module drives the end-to-end execution of planned tasks. It implements advanced session resilience through a State Management Engine that tracks completed steps and supports hydrated session recovery in the event of network interruption—ensuring that actions such as form submissions are not duplicated. Actions are grouped into atomic transaction units; if a constituent action

Received: 20-04-2026 | Accepted: 28-05-2026 | Published: 04-06-2026 | www.ijeml.com

fails, the system rolls back to the last known safe state and prompts the user for guidance.

In complex workflows, the module supports multi-agent concurrency, spinning up parallel sub-agents to execute independent tasks simultaneously (e.g., fetching reference data from one browser context while populating a form in another), significantly reducing total execution latency.

Human-in-the-Loop (HITL) triggers are embedded for high-stakes actions, pausing execution and requiring explicit user confirmation before irreversible operations such as payment submission or data deletion are performed.

F. Admin Management Module

The Admin Management Module provides system administrators with tools for platform configuration, observability, and agent lifecycle management. Integrated observability is delivered through Langfuse, which provides granular trace analysis of LLM interactions, enabling identification of latency bottlenecks and reasoning errors across the multi-agent pipeline. Administrators can perform A/B testing of agent configurations, stage new domain-specific agents in sandbox environments, and perform hot-swap updates to agent models without service interruption.

Security is enforced through Role-Based Access Control (RBAC), JWT authentication, integration with Azure Key Vault for secret management, and configurable rate limiting per user and per agent to control operational costs and prevent resource exhaustion.

V. IMPLEMENTATION

A. Technology Stack

The Srota AI platform is implemented using a modern, cloud-native technology stack. Table I summarizes the key components and their roles.

TABLE I
SROTA AI TECHNOLOGY STACK

Component	Technology
Backend Language	Python 3.13
API Framework	FastAPI (async)
LLM Orchestration	LangChain, LangGraph
Large Language Model	Google Gemini
Speech-to-Text	Deepgram Real-Time ASR
Frontend Framework	React, TypeScript
Styling	Tailwind CSS
Company Website	Next.js 16
Observability	Langfuse
Secret Management	Azure Key Vault
Containerization	Docker
Testing	pytest, httpx

B. Prompt Engineering Architecture

High accuracy across diverse web automation scenarios requires carefully engineered system prompts for each agent. The Planner Agent prompt instructs the LLM to output a strict JSON-formatted action sequence from a high-level goal description. The Navigator Agent prompt provides a sanitized DOM representation and requests the most relevant CSS selector for the next interaction. Both prompts include few-shot examples drawn from representative complex workflows to reduce hallucination rates. The Intent Parser prompt uses a compact classification format to minimize token consumption and latency in the initial routing step.

C. Security and Compliance Framework

All backend API interactions are secured using JWT authentication. Sensitive credentials for external services (Google Gemini, Deepgram) are stored in Azure Key Vault rather than in application source code or environment files accessible to the runtime. Voice data is treated as ephemeral: audio inputs are processed in real time for transcription and intent extraction but are not persisted. The platform implements GDPR-compliant data handling through configurable retention policies and explicit user consent management. All data transmitted between the client widget and the backend is encrypted in transit using TLS.

VI. EVALUATION AND RESULTS

A. Evaluation Methodology

The Srota AI system was evaluated across three dimensions: (1) functional correctness and task completion accuracy, (2) system performance and latency, and (3) accessibility and usability. A comprehensive test suite was executed covering five primary functional areas, and performance profiling was conducted using Langfuse trace analysis across representative workload patterns.

B. Functional Test Results

Table II presents the results of the comprehensive functional test suite executed against the Srota AI platform. All five core test cases passed, covering form navigation, voice intent routing, error recovery, accessibility compliance, and end-to-end latency.

TABLE II
COMPREHENSIVE FUNCTIONAL TEST SUITE RESULTS

ID	Feature	Expected Output	Result
TC-101	Multi-page form navigation	Seamless page transition with state retention	Pass
TC-102	Voice intent routing	Medical Bot module invocation	Pass
TC-103	Error recovery	AI detects and requests missing field info	Pass
TC-104	Screen reader accessibility	Correct ARIA labeling of widget elements	Pass
TC-105	End-to-end latency	Processing and response within 2 seconds	Pass

Received: 20-04-2026 | Accepted: 28-05-2026 | Published: 04-06-2026 | www.ijeml.com

C. Performance Analysis

System performance profiling revealed that the FastAPI asynchronous architecture enables effective concurrent handling of multiple simultaneous voice streams without thread blocking. Langfuse trace analysis identified that approximately 70% of total end-to-end system latency is attributable to LLM inference time within the Planner and Navigator Agents during goal decomposition and DOM analysis. External API calls to Deepgram for ASR account for approximately 15% of latency, while DOM parsing and action execution contribute the remaining 15%.

To address the primary LLM latency bottleneck, the team is implementing model quantization—reducing the numerical precision of LLM weights to decrease memory footprint and accelerate inference without substantial degradation in reasoning accuracy. This optimization is particularly critical for the Voice Assistant Module, where latency directly impacts the naturalness of conversational interaction.

D. Real-World Scenario Evaluation

The platform was evaluated against three representative real-world automation scenarios: government license renewal, healthcare appointment scheduling, and university admission form completion. In each scenario, task completion times were significantly lower than comparable manual workflows, and error rates (defined as incorrect field entries requiring correction) were substantially reduced. User feedback indicated high satisfaction with the voice interaction modality, with participants specifically citing the reduction in cognitive load and the accessibility benefits for users with motor impairments.

VII. DISCUSSION

A. Key Findings

The evaluation results confirm that integrating a multi-agent LLM pipeline with real-time voice interaction produces a qualitatively superior automation experience compared to conventional rule-based tools. The semantic DOM processing approach—based on element function and label rather than screen coordinates—proved particularly effective in maintaining automation reliability across responsive and dynamically rendered web interfaces. The few-shot prompting strategy employed for both the Planner and Navigator Agents substantially reduced hallucinated actions compared to zero-shot baselines. The HITL trigger mechanism for high-stakes actions was validated as essential for user trust, particularly in financial and medical workflow contexts where irreversible mistakes carry significant consequences.

B. Limitations

Several limitations warrant acknowledgment. First, LLM inference latency remains the dominant bottleneck; while the platform is functional for workflows tolerant of 1–3 second per-step delays, highly time-critical scenarios may require further optimization. Second, the Navigator Agent's DOM-based approach assumes that web pages expose sufficient semantic markup (ARIA labels, descriptive button text); pages constructed without accessibility standards may reduce automation accuracy. Third, the current system primarily supports English-language input; multilingual support would broaden accessibility for global users. Fourth, CAPTCHA mechanisms on some web applications represent an ongoing challenge requiring vision-language model integration.

C. Ethical Considerations

The deployment of autonomous web agents raises important ethical considerations. Srota AI mitigates misuse risks through: execution sandboxes that prevent unauthorized access to system resources, HITL confirmation for irreversible actions, comprehensive audit logging of all agent decisions, and JWT-based user authentication ensuring actions are always performed on behalf of an authenticated, consenting user. Organizations deploying Srota AI should implement additional domain-specific safeguards appropriate to their regulatory context, particularly in healthcare and financial applications.

VIII. FUTURE WORK

Several promising directions for future development have been identified:

- **Vision-Language Model Integration**: Incorporating VLMs will enable the Navigator Agent to interpret visual context, handle CAPTCHA challenges, and process handwritten document uploads—transforming the system into a comprehensive spatial and visual assistant.
- **Multimodal Input**: Extending input modalities to include image and gesture recognition will allow users to upload screenshots of target forms or use camera input to guide automation tasks.
- **Multi-Agent Collaborative Networks**: Deploying specialized parallel sub-agents (e.g., a Form Agent maintaining session state while a Research Agent independently retrieves reference data) will eliminate sequential bottlenecks and support complex multitasking workflows.
- **Adaptive Personalization**: Learning from historical user interactions to anticipate preferences, suggest shortcuts, and pre-fill information based on prior entries will enhance both efficiency and user satisfaction.
- **Domain-Specific Agents**: Developing specialized modules for healthcare, finance, legal, and government domains—with domain-appropriate compliance and validation logic—will broaden the platform's applicability.
- **Multilingual Support**: Extending LLM prompting and ASR configuration to support non-English workflows will make the platform globally accessible.

IX. CONCLUSION

This paper presented Srota AI, a voice-controlled intelligent web automation platform that integrates multi-agent LLM orchestration via LangGraph, real-time speech-to-text through Deepgram, and semantic DOM interaction into a unified, accessible system. By enabling users to express complex digital workflow goals in natural language or spoken

Received: 20-04-2026 | Accepted: 28-05-2026 | Published: 04-06-2026 | www.ijeml.com

commands—and autonomously executing those goals across any web application—Srota AI overcomes the fundamental limitations of conventional rule-based automation tools and general-purpose voice assistants.

The platform's modular architecture, comprising the Planner Agent, Navigator Agent, Voice Assistant Module, Semantic DOM Processing Module, and Admin Management Module, establishes a replicable framework for applying agentic AI to the challenge of accessible web interaction. Evaluation results demonstrate functional correctness across diverse automation scenarios, practical latency performance suitable for real-world deployment, and meaningful accessibility benefits for users with diverse needs and abilities.

As web-based digital services continue to grow in complexity, intelligent platforms like Srota AI offer a pathway toward more equitable, efficient, and user-centric digital access—ensuring that the benefits of digital transformation are available to all users, regardless of technical expertise or physical ability.

ACKNOWLEDGMENT

The authors thank Prof. Biswajit Sahoo, Gandhi Institute for Technology, Bhubaneswar, for guidance and supervision throughout this project. The authors extend gratitude to Prof. Smruti Smaraki Sarangi, Head of the Department of Computer Science and Engineering, for institutional support. The authors also acknowledge the open-source communities behind LangChain, LangGraph, FastAPI, React, and the Deepgram platform, whose tools made this work possible.

REFERENCES

[1] W. Yeager, "Web Accessibility: Web Standards and Regulatory Compliance," *Friends of ED*, 2006.
[2] A. Choudhary, A. Porwal, and R. Goyal, "A Comprehensive Survey on Automated Software Testing," in *Proc. Int. Conf. Intelligent Computing, Information and Control Systems (ICICCS)*, 2019, pp. 706–711.
[3] T. Brown et al., "Language Models are Few-Shot Learners," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 1877–1901, 2020.
[4] H. Chase, "LangChain: Building applications with LLMs through composability," GitHub repository, 2022. [Online]. Available: https://github.com/langchain-ai/langchain
[5] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 3rd ed. Pearson Education, 2023.
[6] M. Lo´pez Noceda and B. V. Daudenarde, "Conversational Agents and Voice Assistants for Web Accessibility," in *Proc. W4A '21: Web for All Conference*, 2021.
[7] R. S. Cabral, E. Bosshard, and A. Bast, "Intelligent Form Understanding and Filling Using Neural Networks," in *Proc. ACM SIGCHI Conf. Human Factors in Computing Systems (CHI)*, 2022.
[8] S. Yao et al., "WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022.
[9] World Wide Web Consortium (W3C), "Web Content Accessibility Guidelines (WCAG) 2.1," W3C Recommendation, June 2018. [Online]. Available: https://www.w3.org/TR/WCAG21/
[10] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson, 2021.
[11] Google Developers, "Gemini API Documentation," 2024. [Online]. Available: https://ai.google.dev
[12] Deepgram, "Deepgram Speech-to-Text API Documentation," 2024. [Online]. Available: https://developers.deepgram.com