IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

A Locally Running AI System That Automatically Plans And Executes Tasks On A System With Minimum Human Interaction

Pranay V. kale, Himanshu V. Shivde, Chetan S. Chavhan,
Sandip D. Satpute, Dr. R. S. Durge
Department of Computer Science and Engineering
DRGIT&R College of Engineering Amravati

Abstract: *Recent advancements in Artificial Intelligence (AI) have significantly transformed the way computer systems interact with users and perform complex operations. Traditional systems rely heavily on continuous human input, resulting in inefficiencies and increased time consumption. In contrast, modern intelligent systems emphasize automation, aiming to minimize human intervention while maximizing productivity. This research focuses on the design and development of a locally running AI system capable of autonomously planning and executing tasks.*

Keywords: Artificial Intelligence, Local AI Systems, Task Automation, Machine Learning, Natural Language Processing, Autonomous Systems, Intelligent Planning, Task Scheduling, Edge Computing, Privacy-Preserving AIotocols

I. INTRODUCTION

Artificial Intelligence (AI) has emerged as one of the most transformative technologies of the modern era, reshaping how computer systems interact with users and handle complex operations. Traditional computing systems largely depend on continuous human input for executing tasks, which often leads to inefficiencies, increased time consumption, and limited scalability. As digital environments grow more complex, there is an increasing demand for intelligent systems that can automate processes, reduce manual intervention, and enhance overall productivity.

In response to this demand, intelligent AI-driven systems have evolved to perform tasks autonomously by understanding user intent and making informed decisions. A significant development in this domain is the concept of locally running AI systems software frameworks that operate entirely on a user's device without relying on remote cloud infrastructure. Unlike cloud-based solutions, these systems offer several advantages, including reduced latency, improved responsiveness, enhanced data privacy, and the ability to function without an active internet connection. Such characteristics make them especially suitable for applications where real-time performance and data security are critical.

Locally deployed AI systems combine multiple disciplines, including Machine Learning (ML), Natural Language Processing (NLP), and intelligent decision making algorithms, to create a cohesive framework capable of interpreting user commands, decomposing them into smaller tasks, and executing them efficiently. This capability opens up new possibilities in areas such as personal computing, workflow automation, system administration, and intelligent virtual assistants.

II. LITERATURE SURVEY

The development of modern web and software based systems for automation and intelligent interaction has been supported by several key technologies and frameworks. One of the widely used front-end libraries in modern application development is React. React is a JavaScript library developed and maintained for building user interfaces,

DOI: 10.48175/IJARSCT-34614

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 20, April 2026 Impact Factor: 8.2

particularly for single-page applications. It enables developers to create reusable UI components and manage application state efficiently, which is essential for building responsive and dynamic systems.[React has been extensively adopted in the development of interactive dashboards, web-based control systems, and AI-integrated interfaces due to its component-based architecture and high performance. [1]

React is a widely used JavaScript library designed for building user interfaces, particularly for single-page applications and dynamic web systems. It provides a component-based architecture that allows developers to create reusable UI components, improving code maintainability and development efficiency. According to the React documentation, the library offers a comprehensive reference of APIs and core functionalities that support advanced UI development and state management in modern web applications. The React framework is particularly effective in building responsive and interactive user interfaces for complex systems such as automation tools and AI-based applications. Its virtual DOM mechanism optimizes rendering performance by updating only the necessary components, thereby enhancing application speed and efficiency. This makes it suitable for real-time applications where frequent data updates and user interactions are required.[2]

Node.js is a widely used open-source runtime environment that enables the execution of JavaScript code on the server side. Developed under the OpenJS Foundation, Node.js is built on a non-blocking, event driven architecture, which allows it to handle multiple concurrent requests efficiently. According to the official Node.js documentation, the platform provides a comprehensive set of APIs for building scalable and high-performance network applications. One of the key advantages of Node.js is its asynchronous programming model, which enhances system performance by preventing blocking operations during execution. This makes it particularly suitable for real-time applications, data-intensive systems, and backend services that require high responsiveness. In the context of intelligent and automated systems, Node.js plays a crucial role in managing communication between different system components, handling user requests, and coordinating task execution.[3]

Recent advancements in Artificial Intelligence have enabled the development of systems capable of autonomous task execution with minimal human intervention. The concept of intelligent agents, as described in Artificial Intelligence: A Modern Approach, forms the foundation for such systems. Techniques from Machine Learning and Natural Language Processing allow systems to understand user input and make decisions.

Modern models like GPT-4 enhance the ability to interpret and generate human-like responses. Research in task planning methods, such as hierarchical planning, supports breaking complex tasks into smaller steps. Cloud-based AI systems offer scalability but face issues related to latency and data privacy. In contrast, locally running AI systems provide better security and offline functionality. Frameworks like TensorFlow Lite enable efficient on-device model deployment.[4] A locally running AI system that automatically plans and executes tasks with minimal human interaction has gained increasing attention in recent research. Frameworks like Express.js provide a lightweight backend environment that supports modular and scalable AI integration. The use of middleware such as CORS plays a crucial role in enabling secure communication between the AI system and external services when required. Studies highlight that autonomous agents rely on well-defined APIs and controlled access mechanisms to perform tasks without manual intervention. By incorporating CORS, systems can enforce strict access policies, ensuring that only authorized resources are utilized during execution. This is particularly important for maintaining data privacy and system security in local AI deployments. Additionally, middleware-based architectures support efficient orchestration of multiple components, allowing AI systems to plan, execute, and adapt tasks dynamically. Overall, the integration of Express.js middleware contributes to building robust, secure, and autonomous AI systems.[5]

III. PROPOSED SYSTEM

The proposed system is a locally running intelligent automation framework developed using concepts from Artificial Intelligence, Machine Learning, and Natural Language Processing. It is designed to automatically plan and execute tasks with minimal human intervention while ensuring privacy, low latency, and independence from cloud infrastructure.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

Step 1: User Input Acquisition The process begins with the user providing input in natural language form. This input represents the task or instruction that the system needs to perform. The system is designed to accept flexible and human-friendly commands, making interaction intuitive and accessible.

Step 2: Input Processing and Understanding In this step, the Input Processing Module applies Natural Language Processing techniques to analyze the user input. It performs tasks such as tokenization, parsing, and intent recognition to convert unstructured text into a structured representation. This structured data serves as the foundation for further processing.

Step 3: Intent Recognition and Information Extraction The system identifies the user's intent and extracts relevant entities, keywords, and parameters from the processed input. This step ensures that the system clearly understands what action needs to be performed and what resources are required.

Step 4: Task Planning and Decomposition The Task Planning Module uses Artificial Intelligence planning techniques to break down the identified task into smaller, manageable sub-tasks. It determines the logical sequence of operations required to achieve the final goal. This structured decomposition improves efficiency and reduces execution errors.

Step 5: Execution Strategy GenerationBased on the generated plan, the system formulates an execution strategy. This includes selecting appropriate system resources, tools, and methods required for each sub-task. The strategy ensures optimal use of available local resources.

Step 6: Task Execution The Execution Engine carries out the planned actions by interacting directly with the operating system. It performs operations such as file management, application control, data processing, and system monitoring. The system ensures that each step is executed accurately and in the correct sequence.

Step 7: Monitoring and Feedback Mechanism During execution, the Feedback and Monitoring Module continuously tracks progress. It detects errors, exceptions, or deviations from the plan and takes corrective actions when necessary. It also provides feedback to the user regarding the current status of the task.

Step 8: Learning and Adaptation The Learning Module incorporates Machine Learning algorithms to improve system performance over time. By analyzing past tasks, execution patterns, and user interactions, the system adapts and becomes more efficient, accurate, and responsive in handling future requests.

Step 9: Task Completion and Output Delivery After successful execution, the system delivers the final output to the user. The results are presented in a clear and understandable format, ensuring that the user's objective is achieved without requiring continuous supervision.

IV. RESULTS AND DISCUSSIONS

The proposed locally running intelligent automation system was successfully designed and evaluated based on its ability to interpret user commands, generate execution plans, and perform tasks autonomously. The system demonstrated effective integration of Artificial Intelligence, Machine Learning, and Natural Language Processing techniques.

IV. RESULTS

The system was tested on multiple task scenarios such as file management, application launching, and basic system operations. The results indicate that:

- The Input Processing Module accurately interpreted natural language commands with a high success rate, converting them into structured representations suitable for further processing.
- The Task Planning Module effectively decomposed complex user requests into smaller actionable steps, ensuring logical sequencing and task efficiency. - The Execution Engine successfully executed planned tasks with minimal errors, demonstrating reliable interaction with the operating system.
- The Learning Module showed gradual improvement in task handling by adapting to repeated user commands and optimizing execution strategies. - The Feedback and Monitoring Module provided real-time updates and handled exceptions, improving system reliability and user trust.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 20, April 2026 Impact Factor: 8.2

Performance analysis revealed that local execution significantly reduced response time compared to cloudbased approaches. The absence of network dependency enabled faster processing and uninterrupted task execution. Additionally, data remained confined to the local device, ensuring enhanced privacy and security.

Discussion

The results confirm that a locally running intelligent automation framework is a feasible and efficient alternative to cloud-based systems. By eliminating reliance on external servers, the system achieves lower latency and greater control over user data, which is crucial in privacy-sensitive environments. The use of AI-based task planning proved to be a key strength of the system. It enabled structured problem-solving by breaking down complex instructions into manageable steps, thereby improving accuracy and execution efficiency. Furthermore, the inclusion of a learning mechanism enhanced adaptability, allowing the system to improve performance over time.

VI. CONCLUSION

This work presented the design and development of a locally running intelligent automation system capable of understanding user commands, planning tasks, and executing them with minimal human intervention. By integrating techniques from Artificial Intelligence, Machine Learning, and Natural Language Processing, the system demonstrates an effective approach to building autonomous and efficient computing solutions. One of the key achievements of the proposed system is its ability to operate entirely on the user's device, eliminating dependency on cloud infrastructure. This local execution model significantly improves data privacy, reduces latency, and ensures uninterrupted functionality even without internet connectivity. The system successfully interprets natural language input, generates structured execution plans, and performs tasks reliably through its modular architecture.

The inclusion of a learning component allows the system to adapt over time, improving its accuracy and efficiency based on past interactions. This makes the system more user-friendly and capable of handling repetitive and evolving tasks. Additionally, the feedback and monitoring mechanism enhances transparency and reliability by tracking execution progress and handling errors.Despite its advantages, the system faces limitations related to hardware constraints, scalability, and handling highly complex tasks. These challenges highlight the need for further optimization and advanced techniques to improve performance in resource-limited environments.

Overall, the proposed system demonstrates strong potential in advancing intelligent automation through local processing. It provides a foundation for future research in developing more robust, scalable, and efficient offline AI systems.

VII. ADVANTAGES

The proposed locally running intelligent automation system offers several significant advantages over traditional manual and cloud-dependent automation solutions:

1. Enhanced Data Privacy and Security Since all processing is performed locally on the user's device, sensitive data does not need to be transmitted to external servers. This greatly reduces the risk of data breaches, unauthorized access, and privacy violations.

2. Reduced Dependency on Internet Connectivity The system operates independently of internet access, allowing users to execute tasks even in offline environments. This makes it highly reliable in areas with poor or unstable connectivity.

3. Low Latency and Faster Execution Local processing eliminates network delays associated with cloud communication. As a result, the system responds quickly to user commands and executes tasks in real time.

4. Improved User Control

Users retain full control over their data and system operations. There is no reliance on third-party cloud providers, ensuring greater transparency and autonomy.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 20, April 2026
Impact Factor: 8.2

5. Cost Efficiency

By avoiding cloud infrastructure and subscription-based services, the system reduces long-term operational costs for users.

6. Continuous Learning and Adaptability The integration of a learning module enables the system to improve performance over time by learning from user behavior and past task executions.

7. Automation of Complex Tasks

The system can break down complex instructions into simpler steps and execute them systematically, reducing human effort and improving productivity.

8. Offline Accessibility

All core functionalities are available without external dependencies, making the system suitable for secure, isolated, or restricted environments.

REFERENCES

[1]. React, "React," React Documentation. [Online].Available: https://react.dev/. [Accessed: Apr. 20, 2026].
[2]. React, "React Reference Overview," React Documentation. [Online]. Available: https://react.dev/reference/react. [Accessed: Apr. 20, 2026].
[3]. OpenJS Foundation, "Node.js Documentation,"
[4]. Node.js. [Online]. Available: https://nodejs.org/api/documentation.html. [Accessed: Apr. 20, 2026].
[5]. Express.js, "Express Routing," Express Documentation. [Online]. Available: https://expressjs.com/en/guide/routing.html. [Accessed: Apr. 20, 2026].
[6]. Express.js, "cors," Express Resources and Middleware. [Online]. Available: https://expressjs.com/en/resources/middleware/cors.htm.[Accessed: Apr. 20, 2026].