

NeoShop: A Voice-Enabled E-Commerce System Using MERN Stack

Sujay Kumar¹, Tushar Sharma²

¹Computer Science and Engineering & RKGIT

²Computer Science and Engineering & RKGIT

Abstract - NeoShop is a voice-enabled e-commerce web application developed using the MERN stack to simplify user interaction with online shopping platforms. The system integrates browser-based speech recognition to allow users to navigate, search, and perform actions through simple voice commands instead of relying entirely on traditional click-based interfaces. A rule-based logic engine processes spoken input and maps it to application routes, enabling fast and predictable responses without external AI dependencies. The backend, built with Node.js and Express, manages product data, authentication, and order processing, while MongoDB stores dynamic user and product information. During implementation, the system showed reduced interaction time for common tasks like opening the cart or searching products. In situations where users are multitasking or unable to use touch input, voice navigation proved noticeably useful. The overall design keeps performance lightweight while improving accessibility, especially for users who struggle with complex menus. The project demonstrates that practical, browser-native solutions can enhance user experience without adding unnecessary system complexity.

Key Words: Voice Navigation, MERN Stack, Web Speech API, E-commerce System, Accessibility, Single Page Application.

1. INTRODUCTION

Online shopping platforms have grown quite complex over time. What started as simple product listing pages now includes layered menus, filters, dashboards, and multiple interaction steps just to complete a single purchase. While this adds functionality, it also increases the effort required from the user. Even during testing of our own system, it was noticeable how many clicks were needed just to move between basic sections like cart, profile, or orders.

Most existing e-commerce websites still rely heavily on graphical user interfaces, where users must visually scan and manually interact using a mouse or touch. This works fine in general cases, but it creates friction in situations where quick interaction is needed or when users are not comfortable navigating complex layouts. For example, someone browsing while cooking or holding items cannot easily interact with small buttons on a screen. Similarly, elderly users often find multi-level menus confusing and tiring to use.

NeoShop approaches this problem slightly differently. Instead of replacing the traditional interface, it adds a voice-based interaction layer on top of it. The system is built using the MERN stack MongoDB, Express.js, React.js, and Node.js forming a responsive Single Page Application. The key addition is a voice navigation module that uses the browser's Web Speech API to capture and process user commands in real time. When a user speaks commands like "open cart" or

"go to home," the system interprets the input and triggers the corresponding navigation instantly.

One important design choice was to avoid complex AI models and instead use a rule-based command mapping. This keeps the system fast, predictable, and easier to maintain. During development, this approach also helped reduce delays that usually come with external API calls.

Overall, the idea is not to completely change how e-commerce works, but to make interaction smoother and more flexible. By combining a standard web architecture with lightweight voice control, NeoShop tries to reduce unnecessary effort in navigation while keeping the system simple and usable.

2. Body of Paper

The main content of this paper is organized into structured sections, each focusing on a specific part of the system design and implementation. This helps in maintaining clarity while also making it easier to connect different parts of the work. For instance, the system architecture discussed in Sec. 3 builds directly on the requirements identified earlier, and the implementation details further explain how those ideas were translated into working modules.

The NeoShop system is developed as a full-stack web application using the MERN stack, which stands for MongoDB (database), Express.js (backend framework), React.js (frontend library), and Node.js (runtime environment). This combination allows the application to handle dynamic data efficiently while keeping the user interface responsive. During development, separating the frontend and backend layers made debugging simpler and allowed parallel progress on different modules.

One of the key components is the voice interaction module, which relies on the Web Speech Application Programming Interface (API). This module captures audio input, converts it into text, and then processes it through a simple rule-based logic engine. Instead of predicting user intent using complex Natural Language Processing (NLP) models, predefined command patterns are matched with application routes. This design choice ensures faster response time and avoids unnecessary dependencies.

The backend exposes Representational State Transfer (REST) APIs for handling user authentication, product management, and order processing. Data is stored in MongoDB using flexible document schemas, which proved useful when handling different types of products without rigid structure changes. On the frontend, React components manage the user interface, and state management is handled using Context API to keep track of user sessions and cart data.

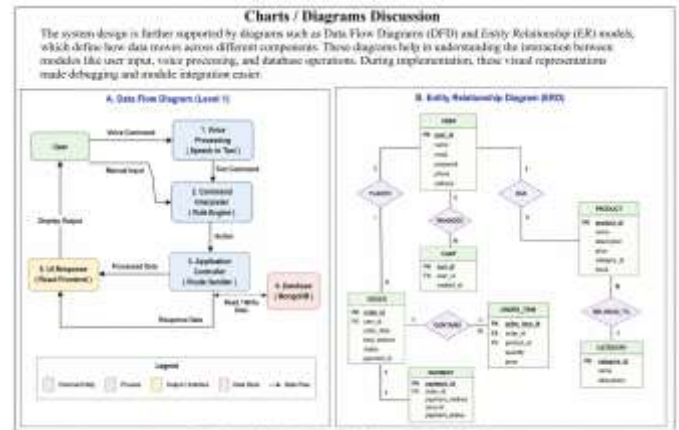
Testing and development followed an iterative approach. Each module authentication, product listing, cart management, and voice navigation was implemented and verified separately before integration. This step-by-step process helped in identifying issues early, especially in voice command handling where small errors in recognition could affect navigation.

Overall, the body of this work reflects a practical implementation where design decisions were influenced not just by theory but also by actual development challenges and observations.

Table -1: Core System Modules and Functions

Table-1: Core System Modules and Functions

	Module Name	Function Description	Technology Used
1.	User Authentication	Handles login, registration, and session management	JWT, Node.js
2.	Product Module	Displays products and manages categories	MongoDB, React
3.	Voice Module	Converts speech to commands for navigation	Web Speech API
4.	Cart System	Manages selected items and quantity updates	Context API
5.	Payment Integration	Processes secure transactions	Stripe/Razorpay
6.	Admin Panel	Manages products and orders	MERN Stack



3. CONCLUSIONS

The development of NeoShop provided a practical look at how small changes in interaction design can impact overall usability. Instead of focusing only on adding features, the project explored how users move through an application and where friction exists. One clear observation during implementation was that even simple actions, like navigating to the cart or checking orders, often require multiple steps in traditional systems. Reducing that effort became a central idea behind the voice-enabled approach.

The integration of a voice module using the Web Speech API turned out to be more effective than expected for controlled commands. Since the system uses a rule-based mapping instead of complex Natural Language Processing, it avoids unnecessary processing overhead and remains responsive. There were some limitations, especially when speech input was unclear or when background noise affected recognition, but within defined conditions, the system performed reliably. It felt more like a practical feature rather than an experimental one.

From a system design perspective, the MERN stack architecture helped maintain a clean separation between components. The frontend handled interaction and state management, while the backend managed logic and data operations. This separation made testing easier, particularly when debugging issues related to API responses or voice-triggered navigation. In a few cases, backend delays did not affect the frontend experience significantly, which showed that the architecture handled failures reasonably well.

In real-world scenarios, such as browsing while multitasking or using a device with limited accessibility, the benefit of voice navigation becomes more noticeable. It does not replace the graphical interface but complements it. That balance is important, because not all users will prefer voice interaction all the time.

There is still scope for improvement. The system currently supports only a fixed set of commands and lacks adaptability to more natural language variations. Expanding this with better intent recognition or multi-language support could make it more flexible. Even with these limitations, the project demonstrates that effective user experience improvements do not always require complex or heavy AI solutions. Sometimes, using existing technologies in a thoughtful way is enough to make the system more intuitive and usable.

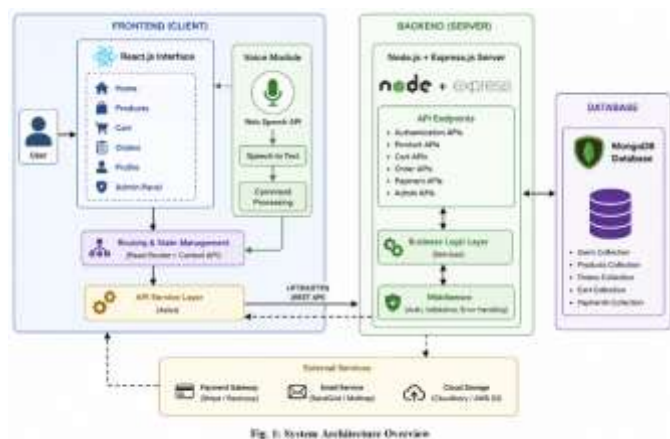


Fig -1: System Architecture Overview

The NeoShop system is designed using a layered architecture that separates the frontend, backend, and database for better clarity and maintainability. The frontend is built using React.js and works as a Single Page Application, allowing users to navigate without full page reloads. This is important for keeping the voice module active throughout the session. The voice component runs on the client side and uses the Web Speech API to capture user commands, convert them into text, and pass them to the routing logic for quick execution.

The backend is developed using Node.js and Express.js, where all API requests, authentication, and business logic are handled. It acts as a bridge between the frontend and the database. MongoDB is used as the database to store user information, product details, and order records in a flexible format. Communication between layers happens through REST APIs over HTTP.

This architecture helps in distributing workload efficiently and simplifies debugging. During testing, even when certain API calls failed, the frontend continued to function without crashing. In practical situations, such as users navigating while multitasking, the system handled quick transitions smoothly, showing the benefit of combining modular design with lightweight voice processing.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our project supervisor for their constant guidance and timely feedback throughout the development of NeoShop. Their inputs were not just limited to technical corrections but also helped us think more clearly about design decisions and overall system structure. At several points during implementation, especially while working on the voice navigation module, their suggestions helped us simplify approaches that initially seemed more complicated.

We are also thankful to the Department of Computer Science & Engineering for providing a supportive environment where we could work on this project without major limitations. Access to lab facilities, development tools, and learning resources made the process smoother. It gave us the flexibility to test different ideas, fail in some cases, and improve the system gradually.

We would like to acknowledge our institution as well for encouraging project-based learning. Working on NeoShop was not just about completing an academic requirement; it gave us exposure to real-world development practices. Handling backend APIs, managing database schemas, and integrating features like payments and voice interaction helped us understand how systems behave beyond theory.

We are also grateful to our friends and classmates who participated in testing the application. Their feedback, especially on usability and voice command accuracy, helped us identify small issues that we might have overlooked during development. Some of the suggestions, like improving command response feedback, came directly from these testing sessions.

Finally, we would like to thank everyone who indirectly contributed to this project. Whether it was through shared resources, online documentation, or discussions, each input played a role in shaping the outcome.

REFERENCES

- [1] V. Surana, S. Patel, and R. Mehta, "Design and Development of E-Commerce Web Applications Using MERN Stack," *International Journal of Computer Applications*, vol. 183, no. 42, pp. 15–20, 2021.
- [2] A. Banks and E. Porcello, *Learning React: Functional Web Development with React and Redux*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2020.
- [3] M. Cantelon, M. Harter, T. Holowaychuk, and N. Rajlich, *Node.js in Action*, 2nd ed. Shelter Island, NY, USA: Manning Publications, 2017.
- [4] K. Chodorow, *MongoDB: The Definitive Guide*, 3rd ed. Sebastopol, CA, USA: O'Reilly Media, 2019.
- [5] R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Ph.D. dissertation, Univ. of California, Irvine, CA, USA, 2000.
- [6] W3C, "Web Speech API Specification," World Wide Web Consortium, 2023. [Online]. Available: <https://www.w3.org/TR/speech-api/>
- [7] Stripe, "Stripe API Reference," 2024. [Online]. Available: <https://stripe.com/docs/api>
- [8] Razorpay, "Razorpay Payment Gateway Documentation," 2024. [Online]. Available: <https://razorpay.com/docs/>
- [9] M. Richards and N. Ford, *Fundamentals of Software Architecture: An Engineering Approach*, Sebastopol, CA, USA: O'Reilly Media, 2020.

[10] T. Haverbeke, *Eloquent JavaScript: A Modern Introduction to Programming*, 3rd ed. San Francisco, CA, USA: No Starch Press, 2018.

[11] S. Tilkov and S. Vinoski, "Node.js: Using JavaScript to Build High-Performance Network Programs," *IEEE Internet Computing*, vol. 14, no. 6, pp. 80–83, Nov.–Dec. 2010.

[12] Google Developers, "Using the Web Speech API," 2023. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API

[13] OWASP Foundation, "OWASP Top Ten Web Application Security Risks," 2021. [Online]. Available: <https://owasp.org/www-project-top-ten/>

[14] A. Mesbah and A. van Deursen, "Single Page Web Applications: Design, Implementation, and Testing," in *Proc. IEEE Int. Conf. Web Engineering*, 2009, pp. 1–10.

[15] MongoDB Inc., "MongoDB Atlas Documentation," 2024. [Online]. Available: <https://www.mongodb.com/docs/atlas/>