iJRASET
International Journal For Research in Applied Science and Engineering Technology

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 14 **Issue:** X **Month of publication:** October 2026

**DOI:** https://doi.org/10.22214/ijraset.2026.85545

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue X Oct 2026- Available at www.ijraset.com*

# AI-Based Smart Food Ordering and Restaurant Management System with Multilingual Chatbot and Personalized Food Recommendations

Haritha Devi R S[1], Akshitha Shree M[2], Bala Shunmugi A[3]
*Department of Computer Science and Engineering, Adhiyamaan College of Engineering, Hosur, India*

***Abstract:*** ***Online food ordering, restaurant communication, and kitchen stock management are commonly handled through separate tools. This fragmentation causes communication delays, inaccurate information about food availability, and limited support for customers who prefer regional languages. This paper presents the Smart Food Ordering and Restaurant Management System, a web application that combines online menu browsing and ordering, an AI-assisted chatbot for food-related queries, interaction in Tamil, English and Telugu, live kitchen inventory monitoring, rule-based personalized recommendations for children, gym users and customers with dietary requirements, a restaurant administration dashboard, and direct restaurant communication options. The frontend is built with React and Vite and the backend with Node.js and Express.js, with inventory events delivered over WebSocket. Inventory levels automatically classify menu items as available, low stock or out of stock, which prevents orders for unavailable items. The recommendation module ranks dishes with a transparent weighted score rather than a trained model. A working prototype was implemented, its principal interfaces are presented, and a set of test scenarios is defined. The results indicate that a single platform can serve both customers and restaurant staff. Formal user evaluation and performance benchmarking are left for future work.***

***Keywords: smart food ordering; restaurant management; AI chatbot; multilingual support; kitchen inventory management; personalized food recommendation; real-time updates; web application.***

## I. INTRODUCTION

The food service industry is adopting digital technologies to improve customer satisfaction and operational efficiency. Many customers still place orders by telephone or messaging applications, enquiring about menus, prices, ingredients and availability before deciding what to order. Restaurant staff answer these enquiries manually while also managing orders, menus and kitchen stock, often with paper records or separate software. These arrangements create communication delays and make it difficult to present accurate availability information. Customers with specific health or dietary needs face an additional problem, since most ordering platforms offer little guidance on suitable meals. Dietary guidance from public health bodies emphasises balanced choices suited to individual needs [1], yet this information is rarely connected to what a restaurant actually has on its menu or in stock.

This paper describes the Smart Food Ordering and Restaurant Management System, a web-based platform that brings these activities into a single application. Customers can browse menus, check availability, add items to a cart, place orders, ask questions to an AI chatbot, switch between Tamil, English and Telugu interfaces, and receive dietary suggestions. Restaurant administrators manage menus, prices, orders and kitchen inventory from a dashboard, and inventory changes are reflected in the customer-facing menu. The main contributions of this work are: (i) an integrated architecture that links ordering, inventory, recommendation and administration modules; (ii) an inventory rule that classifies each item as available, low stock or out of stock and blocks orders for unavailable items; (iii) a transparent, rule-based recommendation score for children, gym users and customers with medical or dietary needs; and (iv) a multilingual interface covering three languages with English as the fallback. The remainder of the paper is organized as follows. Section II reviews related work, Section III compares the existing and proposed systems, Section IV describes the architecture and modules, Section V presents implementation details, Section VI reports results, and Section VII concludes the paper.

## II. LITERATURE SURVEY

Prior work on food technology addresses several of the concerns above, but usually in isolation. Recommendation systems for health-aware eating use health conditions, dietary needs and nutritional attributes to suggest meals, but they seldom connect to restaurant menus or ordering. Stock-management tools monitor ingredient quantities in near real time, yet they generally do not

©IJRASET: All Rights are Reserved 

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue X Oct 2026- Available at www.ijraset.com*

offer customer ordering or personalized suggestions. Web-based ordering applications allow menu browsing and checkout, but they often omit conversational support, regional-language interfaces and nutrition-aware selection. Food-ordering chatbots answer menu questions, although they rarely support several languages or inventory integration. Integrated restaurant management systems cover menu administration and order processing, but they commonly do not combine nutritional recommendations with live ingredient availability.

The gap identified from this survey is the lack of a single system that unites these capabilities for both customers and restaurant staff. The proposed system is intended to fill that gap rather than replace any one category of tool.

## III. EXISTING AND PROPOSED SYSTEMS

### *A. Existing System*

In the existing arrangement, customers order through phone calls, messages or separate online ordering applications. They must contact restaurants to learn about menus, prices, ingredients and availability, and food recommendations are generally limited. Kitchen inventory is frequently maintained by hand, which leads to errors and delays in updating food availability. Traditional systems rarely provide integrated AI assistance, multilingual support, personalized recommendations and real-time inventory information together, and administrators must handle menus, orders and stock records with separate methods.

### *B. Proposed System*

The proposed system is an integrated web platform in which ordering, assistance, inventory and recommendations share one application. Table I summarizes the differences between the two approaches.

TABLE I
COMPARISON OF EXISTING AND PROPOSED SYSTEMS

| Aspect | Existing systems | Proposed system |
|---|---|---|
| Ordering | Phone, messages or separate apps | Single web platform with cart and order placement |
| Availability | Checked by contacting the restaurant | Live availability derived from kitchen stock |
| Inventory | Manual and error-prone updates | Automatic status classification and real-time events |
| Recommendations | Limited or absent | Rule-based scoring for children, gym users and dietary needs |
| Language | Few options | Tamil, English and Telugu with English fallback |
| Assistance | Staff only | AI chatbot for queries and ordering guidance |
| Administration | Separate tools | Dashboard for menus, prices, orders and stock |

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue X Oct 2026- Available at www.ijraset.com*

## IV. SYSTEM ARCHITECTURE AND MODULES

### A. Architecture

The system follows a three-tier client-server design. The presentation tier is a single-page application written in React [2] and built with Vite [3]. The application tier is a Node.js [4] server running Express.js [5], which exposes REST endpoints under /api for foods, orders, AI assistance and restaurant inventory, and which also starts a WebSocket server for real-time events. The data tier uses the SQLite module provided by Node.js, while the prototype routes operate on an in-memory data store. The interface supports four roles, namely customer, kitchen, delivery partner and administrator, and each role is presented with its own dashboard.

### B. Description of Modules

1) *User Registration and Authentication:* Customers register with their name, email address and password. The system validates required fields, verifies credentials at login and ends sessions at logout. Passwords must be protected with a salted hashing algorithm, and access is controlled by role, so that customers reach ordering features while authorized staff reach operational dashboards.
2) *Food Ordering and Menu Management:* Menu items are grouped into categories and can be filtered by keyword, category, price and availability. Customers select quantities and build a cart. Administrators add items, change prices and descriptions, and control availability.
3) *AI Chatbot Assistance:* The chatbot accepts a customer message together with the selected language, an optional location, a user identifier and a conversation identifier. It returns a textual reply along with suggested foods, suggested hotels and, where relevant, order or cart data, so that questions about menu items, ingredients, prices and ordering procedures can be answered in context.
4) *Multilingual Support:* The interface can be displayed in Tamil, English or Telugu. Each language has its own translation dictionary, and a missing key falls back to the English text, so that the interface remains usable when a translation is incomplete.
5) *Kitchen Inventory Management:* Administrators record a stock count for each menu item. The availability status is derived from that count, as described in Section V-E, and the result is shown to customers so that unavailable dishes cannot be ordered.
6) *Personalized Food Recommendation:* Dishes are ranked for each user using a weighted score that considers rating, preferred category, protein content for high-protein preferences and suitability for diabetic-safe diets. The module is intended as a selection aid and does not replace professional medical or dietary advice.
7) *Restaurant Administration and Delivery Dashboards:* The administrator dashboard summarizes orders, menus, prices, inventory and restaurant information. The delivery partner dashboard shows the active assignment, pickup and drop-off details, and delivery workflow actions such as marking an order as picked up and confirming doorstep delivery.
8) *Direct Restaurant Communication:* The application displays restaurant contact details and offers call and message options, which allow customers to confirm special requests or ingredient details with restaurant staff. This channel complements the online ordering process.
9) *Database Management:* The backend retrieves, inserts, updates and deletes records as required, and validates input before modifying data. A relational structure for users, roles, customer profiles and orders is defined, so that modules can share the data they need.

## V. IMPLEMENTATION

### A. Technology Stack

The frontend uses React, HTML, CSS and JavaScript with Vite as the development tool, and it requests data from the backend with the Fetch API [6]. The backend uses Node.js and Express.js, with the ws library providing WebSocket support. Development is carried out in Visual Studio Code, version control uses Git and GitHub, and API endpoints are verified with Postman. The prototype runs in Google Chrome or Microsoft Edge on Windows or Linux.

### B. Server Initialization and Routing

The server creates an Express application and an HTTP server, attaches a WebSocket server to it, and registers JSON parsing and CORS handling. Routes are mounted for foods, orders, AI assistance and inventory. The port is read from the environment and defaults to 5000. Mounting each module separately keeps the handlers for each concern independent and makes the API easy to test one route group at a time.

©IJRASET: All Rights are Reserved 

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue X Oct 2026- Available at www.ijraset.com*

### *C. Food Browsing and Search*

The food route returns records from the data store after applying the customer's filters. Availability, category, keyword and maximum price are applied in sequence, and the response includes the number of matching items. Keyword matching is case-insensitive and checks both the item name and its category.

### *D. Cart Management*

Adding an item to the cart first confirms that the item exists and is available. The cart is restricted to a single restaurant at a time. If the requested item belongs to a different restaurant, the server responds with a conflict status unless the customer explicitly confirms that the existing cart should be cleared. Item prices are read from the stored menu rather than from the client request, so a customer cannot alter the price of an item.

### *E. Inventory Status Logic*

When an administrator updates the stock count of an item, the server sets the availability status as follows. A count of zero or less makes the item OUT_OF_STOCK. A count of five or less on an item marked available changes its status to LOW_STOCK. Otherwise, the status is taken from the administrator's input. After each update, the server broadcasts an inventory.updated event over WebSocket so that connected clients refresh the menu without reloading the page [7].

### *F. Recommendation Scoring*

For each dish f, the recommendation score is computed as a weighted sum of simple rules. The rating r_f defaults to 4.5 when it is missing, and the indicator terms are 1 when the condition holds and 0 otherwise.

$$S(f) = 10 \cdot r_f + 25 \cdot \delta_{fav} + 20 \cdot \delta_{hp} + 15 \cdot \delta_{dia}$$

Here δ_fav is 1 when the dish belongs to the user's favourite category, δ_hp is 1 when the user prefers high-protein food and the dish contains at least 25 g of protein, and δ_dia is 1 when the dish carries the diabetic-safe guidance tag. Dishes are sorted by S(f) in descending order and can then be filtered by category. The approach is deliberately transparent, so that restaurant staff can explain why a dish is recommended. It is a rule-based method rather than a trained machine-learning model.

### *G. Multilingual Interface*

A language context stores the currently selected language and exposes a translation function to all components. Translation text is kept in separate English, Tamil and Telugu files. The translation function looks up the key in the selected language, then in English, and finally returns the key itself, which keeps the interface readable even if a label has not been translated.

### *H. Chatbot Request Handling*

The chatbot route passes the customer's message, language, location and identifiers to a chat service, which returns a response with suggested items and any order or cart data. Because the response is structured, the frontend can render suggestions as cards beside the chat text, and it can keep track of the conversation through the conversation identifier.

### *I. Security Considerations*

The design follows common web security guidance. Authentication must store passwords using a modern hashing algorithm [8], and all inputs from clients, including quantities, identifiers and prices, are validated on the server [9]. Administrative endpoints are restricted to authorized roles. Parameterized queries are used for database access so that user input cannot alter SQL statements.

## VI. RESULTS AND DISCUSSION

### *A. Prototype Interfaces*

Fig. 1 shows the food menu, where customers filter dishes by diet preference, maximum price and minimum rating. Fig. 2 shows the chatbot panel, which lists its supported tasks, including finding restaurants and dishes, tracking orders, inspecting cart items and checking nutrition metrics such as calories, protein and ingredients.

Fig. 3 shows the landing page after the interface language is changed to Telugu, which confirms that the translation mechanism updates the page text and the layout remains intact.

©IJRASET: All Rights are Reserved 

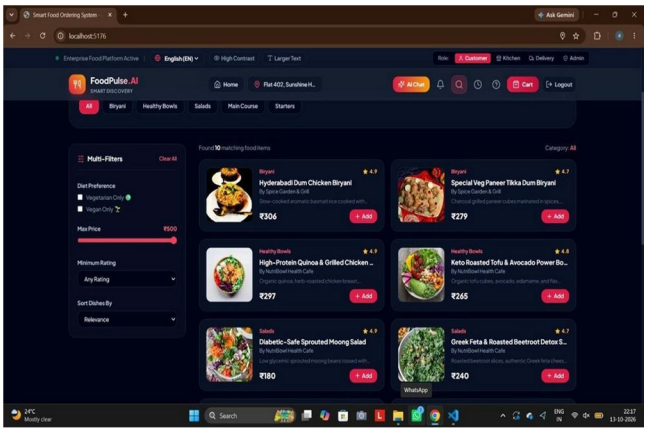


Fig. 1 Food menu with search and multi-filter panel in the prototype

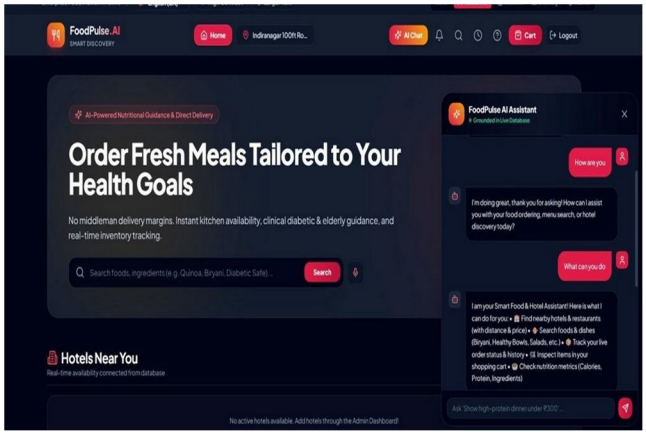


Fig. 2 AI chatbot panel with supported assistance tasks

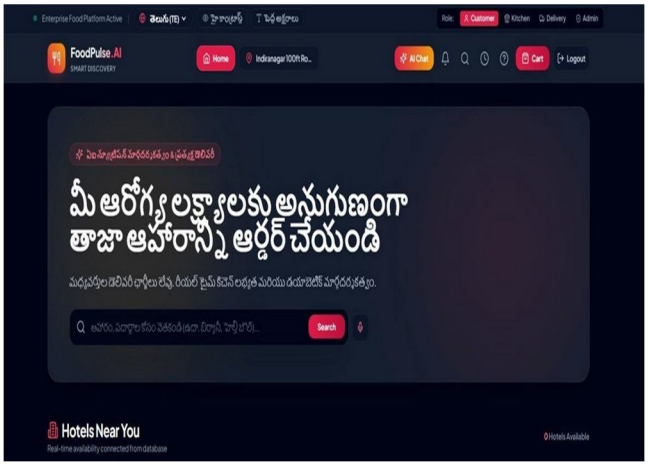


Fig. 3 Landing page displayed in Telugu after language selection

*B. Test Scenarios*

Table II lists selected test scenarios together with their expected behaviour. The scenarios cover navigation, search, language switching, chatbot access, recommendations, invalid orders, inventory updates and failure handling. The table records what each scenario is expected to produce; measured pass or fail outcomes and timing data are not reported in this paper.

TABLE II
SELECTED TEST SCENARIOS AND EXPECTED BEHAVIOUR

| ID | Scenario | Expected behaviour |
|---|---|---|
| TC01 | Open the customer homepage | Homepage and navigation controls are displayed |
| TC02 | Search for an available item | Matching items are displayed |
| TC03 | Search for a non-existing item | No match is shown and the empty result is handled |
| TC04 | Switch interface to Tamil or Telugu | Interface text changes to the selected language |
| TC06 | Open the chatbot and send a query | Chat panel responds with a relevant reply |
| TC07 | Request food recommendations | Dishes are returned in score order |
| TC10 | Submit an invalid or incomplete order | Request is rejected with an error message |
| TC11 | Update kitchen inventory | Status is recalculated and clients are notified |
| TC12 | Call an unavailable backend endpoint | Failure is handled without an uncontrolled crash |

### *C. Discussion*

The prototype demonstrates that ordering, assistance, language selection and inventory-driven availability can operate within one interface.

The *inventory* rule addresses the problem of stale availability information, because the menu status follows the stock count rather than an informal update. The recommendation method is simple to audit, but its suggestions are only as good as the nutritional and rating data entered for each dish. The implementation also has limitations. Current prototype routes rely on an in-memory data store, so records are lost when the server restarts, and the SQLite layer must be connected to all routes for persistence. Online payment, live delivery location tracking and nutritional analysis from verified databases are not yet implemented, and no formal user study has been conducted.

## VII. CONCLUSION AND FUTURE SCOPE

This paper presented the Smart Food Ordering and Restaurant Management System, a web application that unites online ordering, AI-assisted queries, Tamil, English and Telugu interaction, live inventory status, personalized dietary suggestions and restaurant administration.

The inventory rule, cart safeguards and transparent recommendation score illustrate how these functions can be combined without relying on a complex machine-learning pipeline. The prototype shows that a single platform can serve customers and restaurant staff in one place. Future work will proceed along several directions. Persistent storage should replace the in-memory data store across all routes. Recommendations could learn from previous orders and verified nutritional data. Payment gateways such as UPI and card processing, together with live delivery tracking and cloud deployment, would extend the system into a complete service. Finally, a structured user study measuring ordering time, error rates and satisfaction would provide quantitative evidence of the benefits claimed in this paper.

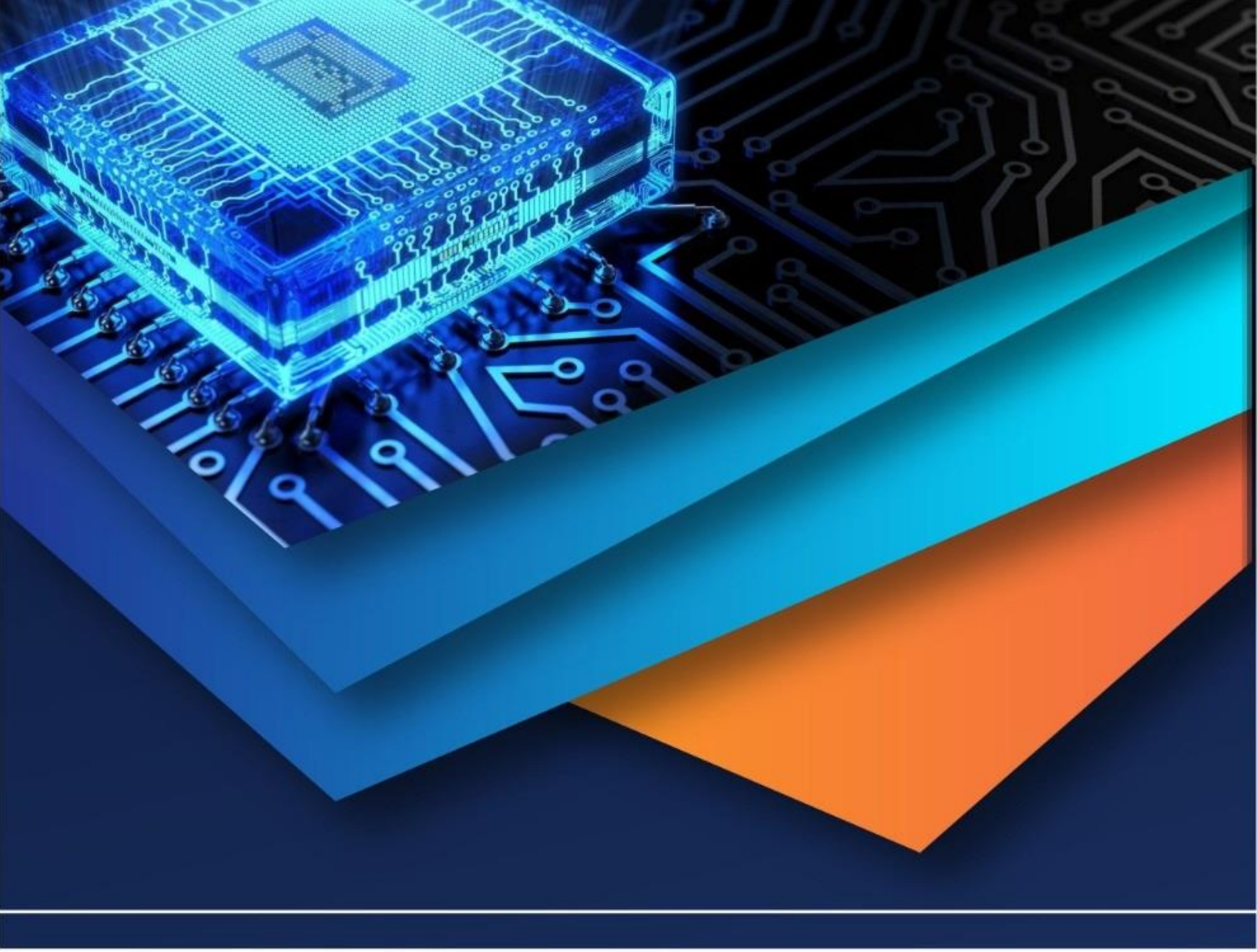
## VIII. ACKNOWLEDGMENT

The authors thank Dr. R. Radhakrishnan, Principal, and Dr. G. Fathima, Head of the Department of Computer Science and Engineering, Adhiyamaan College of Engineering, Hosur, for their support and encouragement. The authors are grateful to their supervisor, Mrs. R.S. Haritha Devi, Assistant Professor, for her guidance throughout the project.

## REFERENCES

[1] World Health Organization, “Healthy Diet,” 2024. [Online]. Available: https://www.who.int/news-room/factsheets/detail/healthy-diet
[2] Meta Open Source, “React Documentation,” 2026. [Online]. Available: https://react.dev/
[3] Vite, “Vite Documentation,” 2026. [Online]. Available: https://vite.dev/
[4] OpenJS Foundation, “Node.js Documentation,” 2026. [Online]. Available: https://nodejs.org/docs/latest/api/
[5] OpenJS Foundation, “Express.js Documentation,” 2026. [Online].
[6] Available: https://expressjs.com/
[7] Mozilla Developer Network, “Using the Fetch API,” 2026. [Online]. Available:
[8] https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_F etch
[9] Mozilla Developer Network, “WebSocket API,” 2026. [Online]. Available:
[10] https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API
[11] OWASP Foundation, “OWASP Application Security Verification Standard (ASVS),” 2025. [Online]. Available: https://owasp.org/www-project-application-security-verification-standard/
[12] OWASP Foundation, “OWASP Top 10: Web Application Security Risks,” 2025. [Online]. Available: https://owasp.org/www-project-top-ten/

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)