iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** II **Month of publication:** February 2026

DOI: https://doi.org/10.22214/ijraset.2026.77322

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 II Feb 2026- Available at www.ijraset.com

Crewmate: Design and Implementation of Web-Based Task Management System Using React and Vite

Prof. Atul Akotkar[1], Chaitanya Wasnik[2], Himanshu Nagpure[3], Avish Walde[4], Reena Kakde[5], Nikita Dhote[6]
[1]*Professor Department Of Computer Science And Engineering, Nagarjuna Institute Of Engineering Technology And Management, Maharashtra, India*
[2, 3, 4, 5, 6]*UG Student, Department Of Computer Science And Engineering, Nagarjuna Institute Of Engineering Technology And Management, Maharashtra, India*

Abstract: ***In modern organizational and academic environments, effective task management and role-based coordination are essential for improving productivity and accountability. This paper presents the design and implementation of Crewmate, a web-based task management system developed using React.js and Vite. The proposed system provides a structured platform that enables administrators to create, assign, and monitor tasks, while employees can view and update task statuses through a user-friendly interface. The application follows a modular and component-based architecture, ensuring maintainability, scalability, and efficient state management. Vite is utilized as the build tool to enhance development speed and optimize application performance. The system demonstrates key functionalities such as role-based dashboards, task categorization, and real-time interface updates, making it suitable for small organizations and academic use. The results indicate that the proposed solution offers a lightweight, responsive, and effective approach to task management using modern frontend web technologies.***

Keywords: Task Management System, React.js, Vite, Web Application, Frontend Development, Dashboard, JavaScript, Role-Based Access.

I. INTRODUCTION

In today's fast-paced organizational and academic environments, effective task organization and accountability are essential for maintaining productivity and collaboration. Although numerous task management solutions are available, many of them rely on complex backend infrastructures and extensive configurations, which can make them less suitable for lightweight academic projects or small-scale organizational use. This paper presents the design and implementation of *Crewmate*, a web-based task management system developed using React.js and Vite. The system focuses on providing a clean, intuitive, and role-based interface that enables administrators to assign and monitor tasks, while allowing employees to view and update task progress efficiently. By leveraging modern frontend technologies and client-side state management, the application achieves responsive performance, modularity, and ease of maintenance. The proposed system demonstrates how contemporary web development frameworks can be used to build effective task management solutions for academic and organizational scenarios.

II. PROBLEM STATEMENT

Many existing task management systems are designed for large-scale enterprises and depend on complex backend services, third-party integrations, and continuous network connectivity. Such systems can introduce unnecessary complexity for academic projects, small teams, or organizations that require a simple and efficient task tracking solution. The challenge addressed in this work is to design and implement a lightweight web-based task management system that supports user authentication, role-based dashboards, and intuitive task handling features. The system should provide clear task organization, status tracking, and usability while maintaining a modular architecture that is easy to understand, extend, and maintain.

III. OBJECTIVES

The primary objectives of this project are:

1) To design and develop a web-based task management system using React.js and Vite
2) To implement role-based access for administrators and employees through dedicated dashboards.
3) To utilize client-side storage and state management techniques for handling application data and task persistence.

4) To enable efficient task creation, assignment, categorization, and status tracking within the system
5) To ensure a modular, responsive, and user-friendly interface suitable for academic and small organizational environments.
6) To provide a scalable foundation that allows future enhancements such as backend integration, analytics, and real-time features.

IV. LITERATURE REVIEW

Existing task management systems such as Trello, Asana, and ClickUp offer rich features but depend on cloud services and complex APIs. Studies on lightweight web applications emphasize the importance of client-side frameworks like React for rapid development and local storage mechanisms for quick prototyping and offline functionality. Prior works also highlight that Vite offers enhanced development performance through optimized bundling and hot module replacement, making it ideal for modern JavaScript-based systems. This project integrates these insights to create a minimal but fully functional task management system.

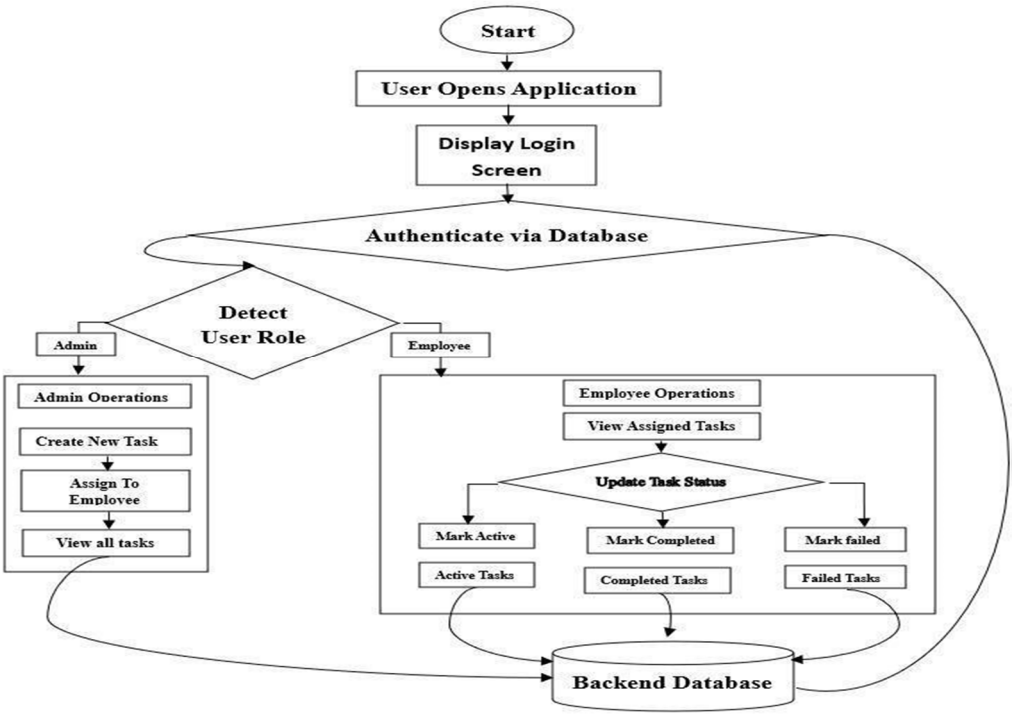
V. METHODOLOGY

The system follows a component-based architecture, typical of React applications, where each UI element (login, dashboard, task cards, etc.) is encapsulated into reusable components. The local storage acts as the data layer for user and task information, while React Context API handles state management across the application.

A. Core Modules

1) Authentication Module: Manages login functionality for both Admin and Employees using predefined credentials stored in local storage.
2) Admin Dashboard: Allows the admin to create, assign, and view all tasks using AdminDashboard and Createtasksection components.
3) Employee Dashboard: Displays employee-specific tasks categorized by their status through the Dashboard and TaskList components.
4) Task Management Module: Organizes tasks into subcomponents (Accepttask, Newtasklist, Completetasklist, and Failedtasklist) to visualize progress and completion.
5) Context Management: Implemented using the AuthProvider component, which shares user data across the app through the Authcontext.

VI. SYSTEM DESIGN



Workflow

1) Login: Users log in as either admin or employee.
2) Role Detection: The app checks stored credentials and redirects users to the appropriate dashboard.
3) Data Fetching: Task and employee data are retrieved from local storage.
4) Dashboard Rendering: Based on user role, relevant components are displayed dynamically.
5) Task Management: Admins create or edit tasks; employees update or view their assigned tasks.

VII. IMPLEMENTATION DETAILS

A. *Technologies Used*

1) Frontend Framework: React.js
2) Build Tool: Vite
3) Language: JavaScript (ES6)
4) Styling: Tailwind CSS
5) State Management: React Context API
6) Data Handling: Browser LocalStorage - Environment: Node.js with npm

B. *Logic Overview*

The App.jsx file controls application flow by rendering the appropriate dashboard after verifying user credentials. The AuthProvider.jsx sets up context-based state management and initializes default employee data through the Localstroge.jsx utility. Task categorization logic is implemented in TaskList.jsx, where tasks are filtered and displayed based on status flags (active, newTask, completed, failed).

C. *User Interface*

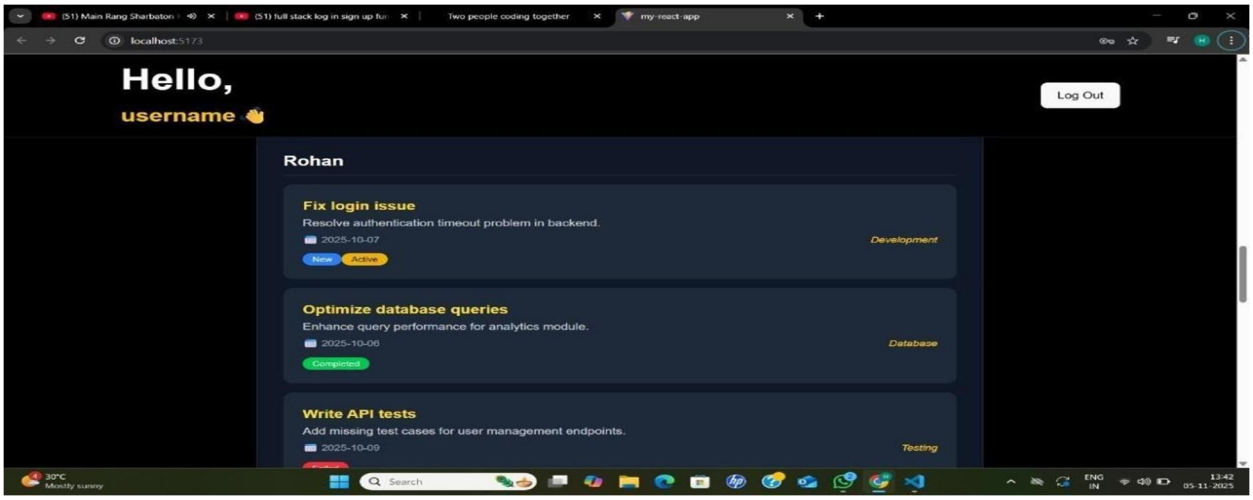
The UI employs a dark theme with white text for readability. Each dashboard includes a header (Headers.jsx), task counters (TaskListNumber.jsx), and task creation forms (CreatTasksection.jsx). Horizontal scrollable task cards provide a modern and compact view of multiple tasks simultaneously.

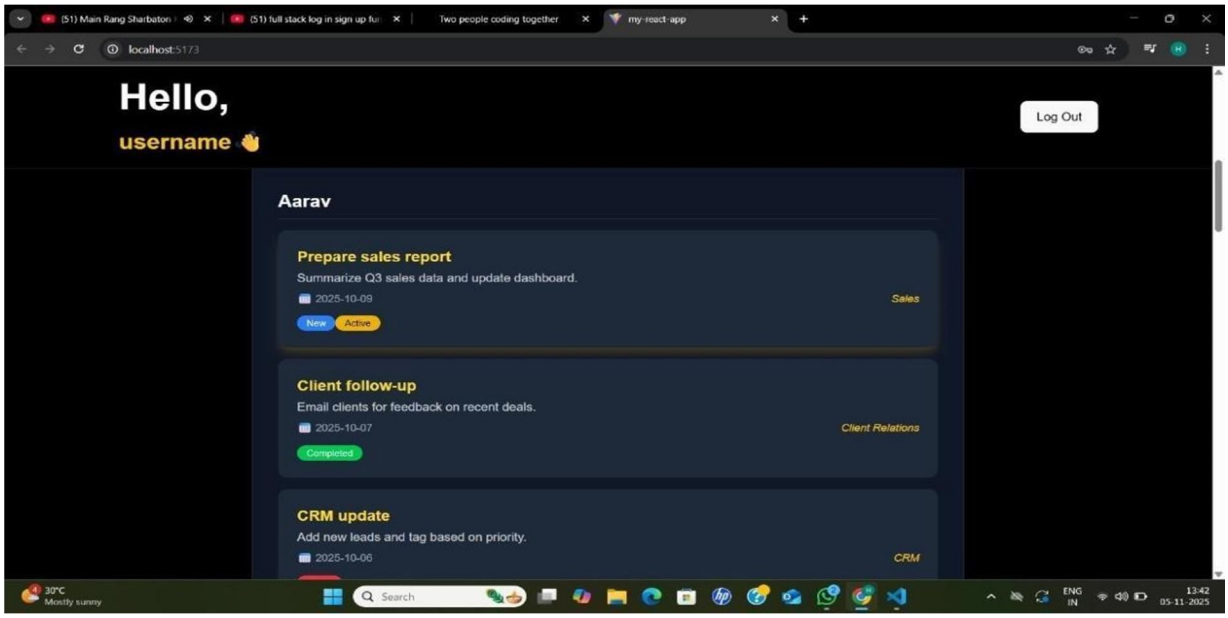
VIII. RESULTS & DISCUSSION

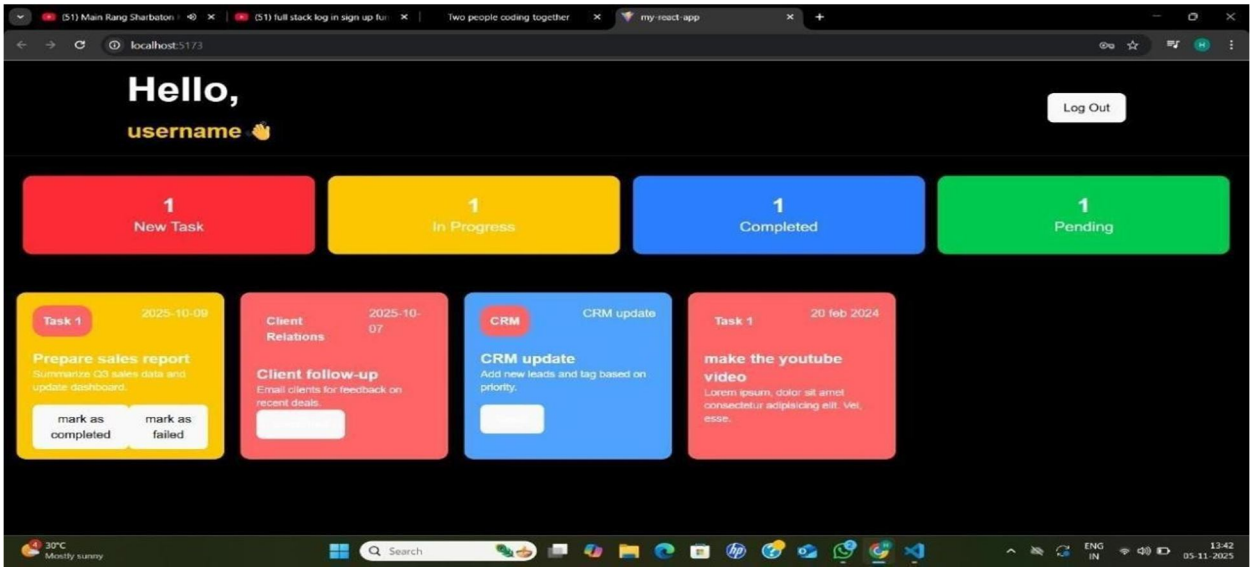
The resulting system successfully differentiates between administrator and employee roles.

1) Admin Panel: Displays tools to create and manage all employee tasks.
2) Employee Panel: Shows categorized tasks, providing visual clarity of active, completed, and failed tasks.
3) Performance: The app loads instantly due to Vite's optimized bundling and React's component reusability.
4) Storage: All data persists across sessions via local storage, validating its offline usability.

User testing within a local environment demonstrated smooth navigation, quick rendering, and proper functionality without backend dependencies.







IX. CONCLUSION

This project achieves a self-contained, efficient, and responsive task management solution using modern frontend technologies. It demonstrates that React and Vite can build production-grade interfaces even without external databases or APIs. The system is ideal for small organizations, personal productivity tracking, or academic demonstrations, highlighting the practical application of React Context API and LocalStorage in real-world projects.

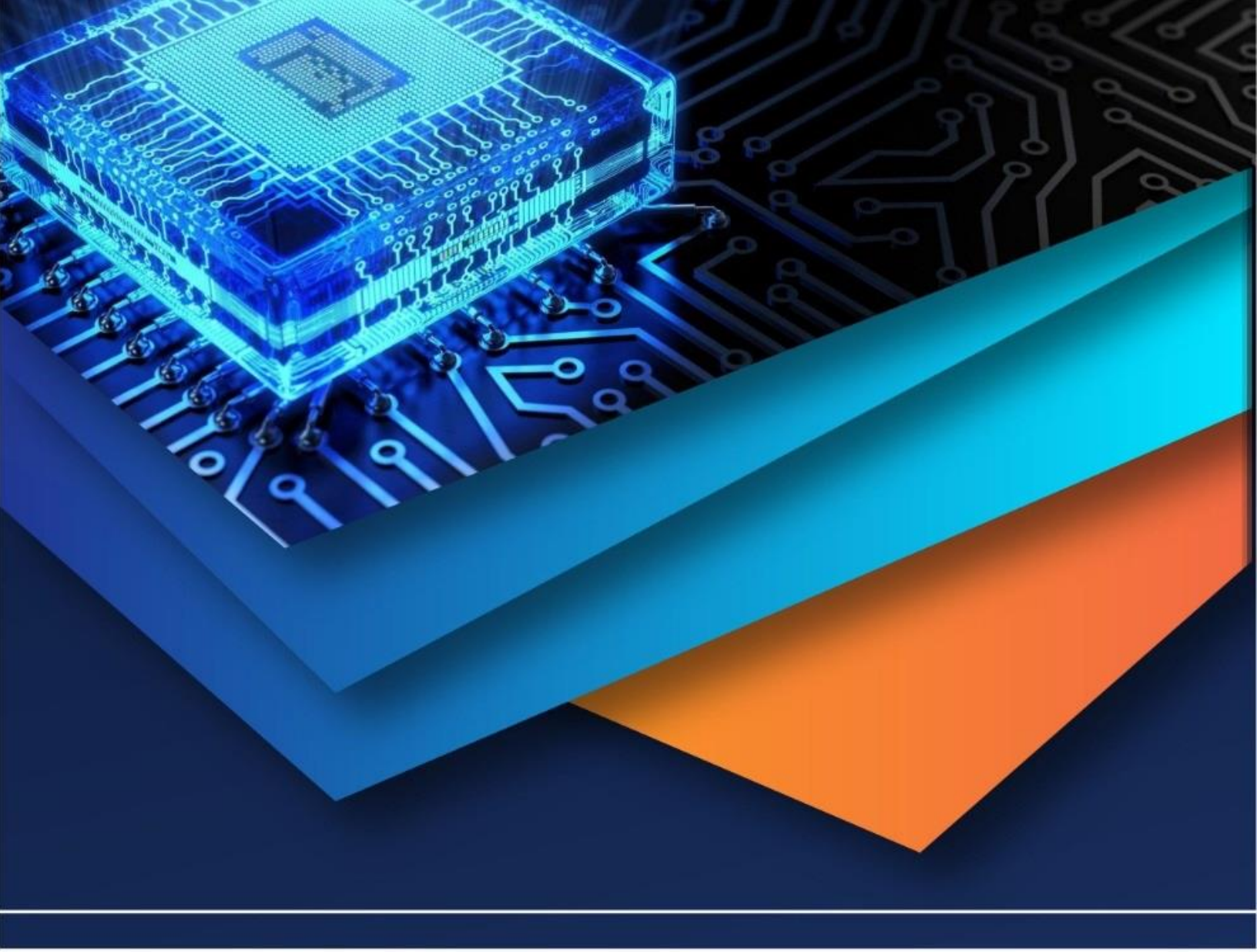
X. FUTURE SCOPE / ENHANCEMENTS

Future enhancements may include:

1) Integration with a backend API and database (e.g., Firebase, MongoDB).
2) Addition of real-time updates and notifications using WebSockets.
3) Role-based user registration and authentication systems.
4) Exportable reports and analytics dashboards.
5) Cloud storage synchronization for multi-device access.

REFERENCES

[1] S. Kumar and A. Hernandez, Journal of Advanced Web Systems, ―Evaluating Client-Side Storage Technologies for Modern Web Applications: A Comprehensive Review,‖ vol. 11, pp. 56–70, 2025.
[2] Cloud Native Computing Foundation, CNCF Technical Report, ―Best Practices for Deploying and Hosting Decoupled Web Applications,‖ vol. 9, pp. 1–20, 2024.
[3] J. Smith, L. Johnson, and P. Brown, Journal of Web Engineering, ―A Comparative Analysis of Frontend Frameworks: React.js, Angular, and Vue.js for Building Scalable Web Applications,‖ vol. 23, pp. 112–125, 2023.
[4] T. Wilson and R. Davis, IEEE International Conference on Data Engineering, ―A Performance- Centric Comparison Between MongoDB and MySQL Databases in Web Applications,‖ vol. 31, pp. 334–348, 2023.
[5] D. Evans and B. Garcia, Conference on Software Development, ―State Management in Large-Scale React Applications: Context API vs. Redux,‖ vol. 12, pp. 67–80, 2023.
[6] H. Clarke and F. Ito, Journal of Information Security, ―Security Vulnerabilities in Node.js REST APIs and Their Mitigation,‖ vol. 15, pp. 145–162, 2023.
[7] K. Zhang and M. O. Lee, International Conference on Software Engineering, ―Performance and Developer Experience of Modern Frontend Build Tools: Vite vs. Webpack,‖ vol. 18, pp. 45–59, 2022.
[8] C. Miller, T. Anderson, R. Gupta, and S. Lee, ACM Computing Surveys, ―Architectural Patterns for Modern Web Applications: A Review of Microservices and Monoliths,‖ vol. 54, pp. 98–120, 2022.
[9] R. Martin and S. Chen, International Conference on Software Maintenance and Evolution, ―The Impact of Clean Code and Modular Architecture on Software Maintainability,‖ vol. 27, pp. 230– 245, 2022.
[10] Roberts and S. Patel, International Journal of Computer Applications, ―Designing Scalable RESTful Web Services with Node.js and Express.js,‖ vol. 29, pp. 210–220, 2021.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY