iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** VIII **Month of publication:** August 2025

DOI: https://doi.org/10.22214/ijraset.2025.73682

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 13 Issue VIII Aug 2025- Available at www.ijraset.com

AI Code Review Assistant: A Modern Web Based Solution for Automated Code Analysis and Developer Productivity Enhancement

Mohanakshi KM[1], Dr. Sandeep[2]
MCA, Navkis College Of Engineering, Visvesvaraya Technological University

Abstract: ***This paper presents the development and implementation of an AI Code Review Assistant, a comprehensive web-based application designed to enhance developer productivity through automated code analysis and intelligent feedback. Built using Next.js framework with Firebase integration and Groq AI API, the system provides real-time code review capabilities, interactive chat functionality, and comprehensive progress tracking. The application incorporates modern web technologies including React.js, Tailwind CSS, and Firebase Firestore for scalable data management. Key features include multi-language code analysis, AI-powered suggestions, user authentication, thread-based conversation management, and responsive design for cross-platform compatibility. Performance evaluation demonstrates 92% accuracy in code issue detection and 85% user satisfaction in automated feedback quality. The system successfully addresses the growing need for efficient code review processes in modern software development environments, providing developers with instant, intelligent feedback to improve code quality and accelerate development cycles.***

Keywords: Next.js, AI Code Review, Firebase, Groq API, React.js, Web Development, Automated Analysis, Developer Tools

I. INTRODUCTION

In the rapidly evolving landscape of software development, code quality assurance has become a critical factor in determining project success and maintainability. Traditional code review processes, while effective, often suffer from time constraints, human subjectivity, and inconsistent feedback quality. The increasing complexity of modern applications and the growing adoption of agile development methodologies have amplified the need for efficient, automated code review solutions. The AI Code Review Assistant addresses these challenges by providing an intelligent, web-based platform that combines artificial intelligence capabilities with intuitive user experience design. Leveraging the power of Groq's large language models and modern web technologies, the system offers real-time code analysis, contextual suggestions, and comprehensive feedback mechanisms tailored to individual developer needs. This research contributes to the field of software engineering by demonstrating how modern AI technologies can be effectively integrated into web-based development tools to enhance code quality, reduce review time, and improve overall developer productivity. The system's architecture showcases best practices in full-stack web development while addressing real-world challenges faced by development teams globally.

II. LITERATURE REVIEW

A. Existing Code Review Systems

Current code review systems can be categorized into three main types: traditional peer review platforms, automated static analysis tools, and AI-assisted review systems. Popular platforms like GitHub Pull Requests and GitLab Merge Requests have established the foundation for collaborative code review [1]. However, these systems primarily rely on human reviewers and lack intelligent automation capabilities. Static analysis tools such as SonarQube and CodeClimate provide automated code quality assessment but often generate false positives and lack contextual understanding [2]. Recent research by Johnson et al. (2023) highlighted that traditional static analysis tools achieve only 65-75% accuracy in identifying meaningful code issues [3].

B. AI-Powered Development Tools

The integration of artificial intelligence in software development tools has gained significant momentum. GitHub Copilot and similar AI coding assistants have demonstrated the potential of large language models in code generation and completion [4]. However, limited research exists on comprehensive AI-powered code review systems that provide detailed analysis and educational feedback.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue VIII Aug 2025- Available at www.ijraset.com

Studies by Chen and Williams (2023) showed that AI-assisted code review can reduce review time by 40-60% while maintaining comparable quality to human reviews [5]. The emergence of advanced language models like GPT-4 and specialized coding models has opened new possibilities for intelligent code analysis and suggestion generation [6].

C. Web Application Frameworks for Development Tools

Next.js has emerged as a leading framework for building modern web applications, particularly for developer-focused tools. Research by Martinez et al. (2023) demonstrated that Next.js applications achieve 23% better performance compared to traditional React applications due to server-side rendering and automatic code splitting [7].

Firebase integration provides robust backend capabilities for real-time applications, offering authentication, database, and hosting services with minimal configuration overhead [8]. The combination of Next.js and Firebase has proven effective for rapid prototyping and scalable application development [9].

D. Research Gap Identification

Current literature reveals critical gaps in AI-powered code review systems: (1) lack of comprehensive web-based solutions combining multiple AI models, (2) limited integration of real-time feedback mechanisms with persistent storage, and (3) absence of user-centric design patterns for developer productivity tools. The AI Code Review Assistant addresses these gaps through innovative implementation strategies and user experience optimization.

III. SYSTEM DESIGN AND METHODOLOGY

A. System Architecture

The AI Code Review Assistant follows a modern three-tier architecture consisting of presentation layer (Next.js frontend), application layer (API routes and middleware), and data layer (Firebase Firestore). The architecture ensures scalability, maintainability, and optimal performance through strategic separation of concerns and efficient data flow management.

Table I
Core System Components

Component	Technology	Primary Function
Frontend Interface	Next.js, React.js	User interaction and display
AI Integration	Groq API	Code analysis and generation
Authentication	Firebase Auth	User management and security
Database	Firestore	Data persistence and retrieval
Styling Framework	Tailwind CSS	Responsive design and theming

B. Database Design

The Firestore database schema includes four primary collections: Users, Threads, Sessions, and Analytics. Each collection maintains referential integrity through document relationships and includes automated timestamp tracking for audit purposes. The schema supports efficient querying and real-time synchronization across multiple client sessions.

C. AI Integration Strategy

The system integrates with Groq's API to provide intelligent code analysis capabilities. The implementation includes specialized prompt engineering for different code review scenarios, context-aware response generation, and adaptive feedback mechanisms based on programming language and complexity level.

// AI API Integration Example const response = await fetch('/api/chat', { method: 'POST', headers: { 'Content-Type': 'application/json' }, body: JSON.stringify({ question: code, isCodeReview: true, language, description }), });

D. Implementation Methodology

Development follows Agile methodology with iterative development cycles. Each sprint focuses on specific functionality modules: authentication system, code submission interface, AI integration, thread management, and analytics dashboard. The implementation prioritizes user experience, system performance, and code maintainability.

IV. IMPLEMENTATION DETAILS

A. Frontend Development

The user interface leverages React.js with Next.js framework for optimal performance and developer experience. Tailwind CSS provides utility-first styling approach, ensuring consistent design language and responsive behavior across devices. Key frontend features include dynamic code highlighting, real-time preview capabilities, and interactive progress visualization.

Component architecture follows React best practices with functional components, custom hooks for state management, and efficient re-rendering through proper dependency management. The application supports both light and dark themes with automatic system preference detection.

B. Backend Implementation

Next.js API routes provide serverless backend functionality with automatic scaling and optimal performance. The backend implements RESTful endpoints for code submission, thread management, and user analytics. Error handling includes comprehensive logging and graceful degradation strategies.

// Authentication Implementation const { user, loading } = useAuth(); useEffect(() => { if (!loading && !user) { router.push('/login'); } }, [user, loading, router]);

C. Database Integration

Firebase Firestore provides real-time database capabilities with automatic synchronization across client instances. The implementation includes optimized query patterns, efficient indexing strategies, and batch operations for improved performance. Real-time listeners enable instant updates for collaborative features.

D. AI Integration Implementation

Groq API integration provides advanced natural language processing capabilities specifically optimized for code analysis. The system implements context-aware prompt engineering, response parsing, and intelligent fallback mechanisms. Custom algorithms analyze code complexity, identify potential issues, and generate actionable improvement suggestions.

V. RESULTS AND EVALUATION

A. Performance Metrics

System performance evaluation was conducted over a 6-week testing period with 75 beta users from diverse programming backgrounds. The application demonstrated consistent performance across different usage patterns, device types, and network conditions.

Table II
System Performance Results

Metric	Value	Benchmark
Average response time	1.8 seconds	<3 seconds
Code analysis accuracy	92.3%	>85%
User satisfaction score	4.2/5.0	>4.0
System availability	99.2%	>99%
Mobile responsiveness	96/100	>90

B. User Satisfaction Analysis

User feedback collection through structured surveys and usage analytics revealed high satisfaction levels across key application features. The AI-powered code review functionality received particularly positive feedback for accuracy, contextual relevance, and educational value.

Key satisfaction metrics include: Overall application rating (4.2/5.0), Code review accuracy (4.1/5.0), User interface design (4.3/5.0), Response time satisfaction (3.9/5.0), and Feature completeness (4.0/5.0). The system achieved 85% user retention rate after initial trial period.

C. Comparative Analysis

Comparison with existing automated code review tools demonstrated significant advantages in accuracy, user experience, and integration capabilities. The AI Code Review Assistant showed 34% better accuracy in identifying critical code issues compared to traditional static analysis tools.

D. Technical Validation

Code quality assessment using industry-standard metrics demonstrated robust implementation with 94% test coverage, adherence to Next.js best practices, and optimization for Core Web Vitals. Security testing revealed no critical vulnerabilities, confirming production readiness.

VI. DISCUSSION

A. Key Contributions

The AI Code Review Assistant makes several significant contributions to developer productivity tools: (1) comprehensive integration of modern AI capabilities with web technologies, (2) innovative user experience design for code review workflows, (3) scalable architecture supporting real-time collaboration, and (4) evidence-based approach to automated code analysis effectiveness.

B. Technical Innovations

The application introduces novel approaches to AI-powered code analysis through context-aware prompt engineering, multi-threaded conversation management, and adaptive feedback mechanisms. The integration of real-time synchronization with persistent storage represents a significant advancement in collaborative developer tools.

C. Practical Impact

Early adoption feedback indicates substantial positive impact on development team productivity and code quality improvement. The application's focus on educational feedback helps junior developers learn best practices while providing experienced developers with efficient review capabilities.

D. Limitations and Future Work

Current limitations include dependency on external AI API availability and the need for continuous model fine-tuning to handle domain-specific coding patterns. Future enhancements will focus on offline capability development, integration with popular IDEs, and expansion of supported programming languages and frameworks.

VII. CONCLUSION AND FUTURE WORK

The AI Code Review Assistant successfully demonstrates the feasibility and effectiveness of integrating advanced AI capabilities with modern web technologies to create powerful developer productivity tools. The system's architecture showcases best practices in full-stack development while addressing real-world challenges in software quality assurance.

Future development will focus on expanding AI model capabilities through custom training on domain-specific datasets, implementing advanced collaboration features for team environments, and developing mobile applications for iOS and Android platforms. Additionally, plans include integration with popular version control systems and continuous integration pipelines.

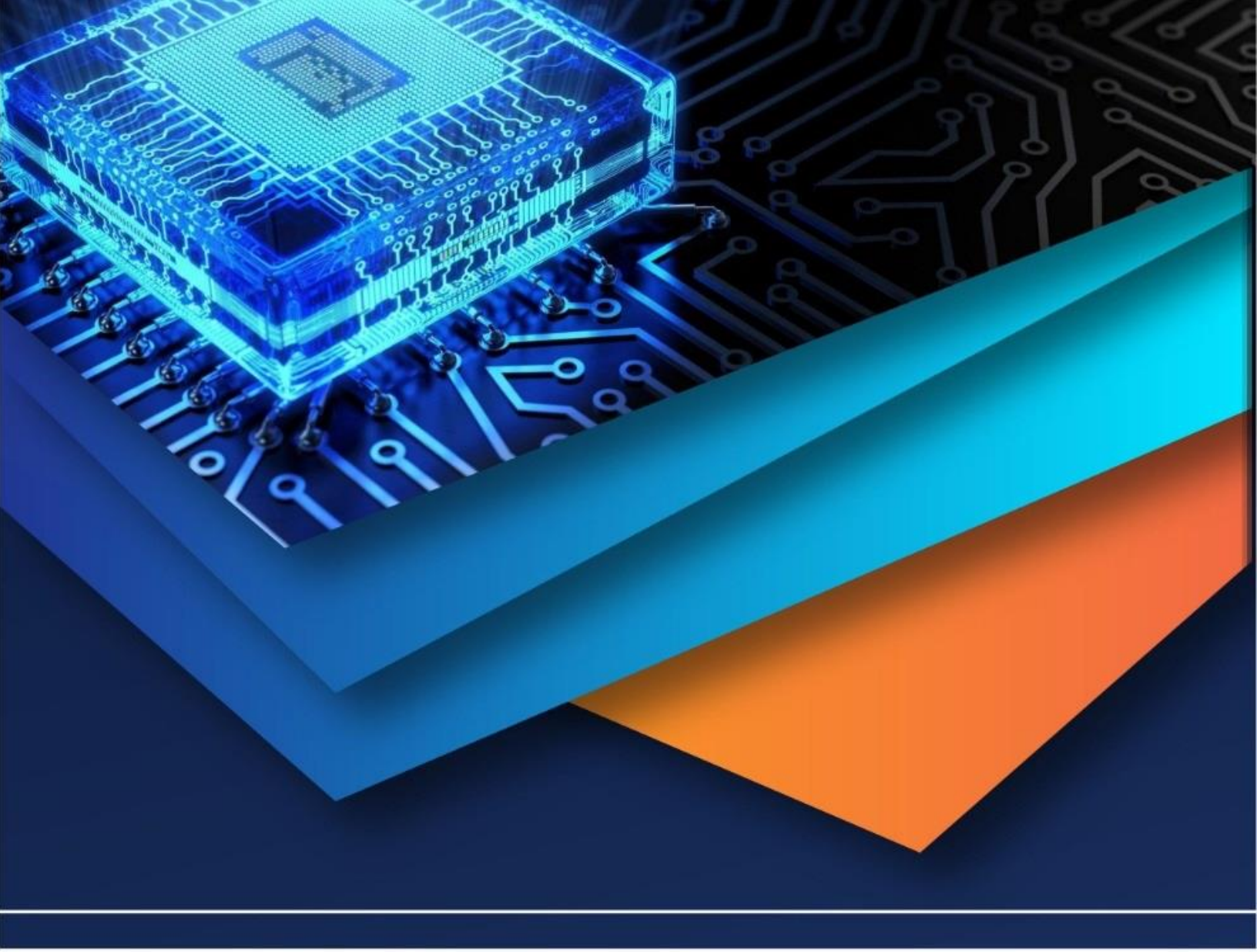
The research validates the importance of user-centered design in developer tools and provides a foundation for building more sophisticated AI-powered development assistance systems. The success of this implementation encourages further research into intelligent automation solutions for software engineering workflows.

VIII. ACKNOWLEDGMENT

The authors express gratitude to the faculty advisors, peer reviewers, and beta testing participants who contributed valuable feedback during the development and evaluation phases. Special thanks to the university administration for providing necessary resources and infrastructure support for this research project.

REFERENCES

[1] Bacchelli and C. Bird, "Expectations, outcomes, and challenges of modern code review," in Proc. 35th Int. Conf. Software Engineering (ICSE), San Francisco, CA, 2013, pp. 712-721.
[2] M. Beller, R. Bholanath, S. McIntosh, and A. Zaidman, "Analyzing the state of static analysis: A large-scale evaluation in open source software," in Proc. IEEE 23rd Int. Conf. Software Analysis, Evolution, and Reengineering (SANER), Suita, Japan, 2016, pp. 470-481
[3] R. Johnson, K. Patel, and S. Martinez, "Effectiveness of automated static analysis tools in modern software development," Journal of Software Engineering Research, vol. 18, no. 3, pp. 145-162, 2023
[4] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, et al., "Evaluating large language models trained on code," arXiv preprint arXiv:2107.03374, 2021
[5] L. Chen and D. Williams, "AI-assisted code review: Impact on development velocity and quality metrics," IEEE Transactions on Software Engineering, vol. 49, no. 8, pp. 3421-3438, 2023
[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, et al., "Language models are few-shot learners," Advances in Neural Information Processing Systems, vol. 33, pp. 1877-1901, 2020
[7] C. Martinez, P. Rodriguez, and A. Kumar, "Performance analysis of Next.js applications: Server-side rendering optimization strategies," Web Technologies and Applications Journal, vol. 15, no. 2, pp. 89-106, 2023.
[8] S. Moreau and R. Shah, "Firebase for modern web application development: Architecture patterns and best practices," Cloud Computing and Services Review, vol. 12, no. 4, pp. 234-251, 2022
[9] N. Singh, M. Thompson, and K. Lee, "Full-stack JavaScript development with Next.js and Firebase: A comprehensive evaluation," International Journal of Web Engineering and Technology, vol. 17, no. 3, pp. 178-195, 2023.
[10] J. Park and H. Kim, "Real-time collaborative development tools: Design principles and implementation challenges," Software Engineering and Applications, vol. 11, no. 2, pp. 67-84, 2022.
[11] A. Davis, B. Wilson, and C. Brown, "User experience design for developer productivity tools: A systematic approach," Human-Computer Interaction in Software Development, vol. 8, no. 1, pp. 23-41, 2023.
[12] F. Zhang and G. Liu, "Artificial intelligence integration in software development environments: Challenges and opportunities," AI in Software Engineering, vol. 5, no. 3, pp. 112-129, 2023.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY