© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719198

Gen-AI Code Reviewer: An AI-Assisted Pull Request Review Platformfor Faster First-Pass Software Quality Analysis

VEDANG PANDITRAO[1], FAIZAHMED PATHAN[2], SHAHU SAKHARE[3], OM PATIL[4], ADITYA MUKE[5]
[1, 2, 3, 4, 5] *Department of Computer Engineering P.E.S. Modern College of Engineering, Pune, India Affiliated to Savitribai Phule Pune University Academic Project Paper*

Abstract- ***Code review remains one of the most important quality-assurance activities in modern software engineering, yet manual review of pull requests is often delayed by reviewer workload, inconsistent practices, and growing repository activity. This paper presents Gen-AI Code Reviewer, a full-stack web platform that automates first-pass pull request analysis using a large language model integrated with GitHub-based development workflows. The system authenticates users through GitHub OAuth, retrieves repository and pull request metadata through the GitHub REST API, extracts changed-file diffs, and submits structured review prompts to an AI model for contextual analysis. The generated output includes a summary, risk score, severity- tagged comments, and suggested fixes, which are persisted in a PostgreSQL database and displayed through a dashboard interface. The platform is implemented using Next.js, TypeScript, Prisma ORM, tRPC, React Query, Better Auth, and Inngest for asynchronous workflow execution. The proposed system is designed to reduce review turnaround time, improve consistency of early feedback, and support human reviewers by surfacing potentially risky changes before manual inspection. The paper describes the motivation, architecture, workflow, implementation strategy, and practical relevance of the prototype, and discusses limitations related to model reliability, repository context, and prompt-length constraints.***

Index Terms— ***Code Review, Pull Request Analysis, Generative AI, Github, Large Language Models, Software Quality, Developer Tooling, Next.Js***

I. INTRODUCTION

Code review is a widely adopted practice for detecting defects, improving maintainability, and preserving code quality in collaborative software projects. In Git-based workflows, pull requests act as the primary unit of review before code integration. However, manual pull request review is time-consuming and depends heavily on reviewer availability, project familiarity, and sustained attention. As teams scale, these constraints can increase review latency and make it harder to maintain consistent first-pass feedback.

Recent progress in generative AI has created new oppor- tunities to assist developers in tasks that require contextual reasoning over source code. While traditional static analysis tools are effective for rule-based checks, they do not always provide natural-language explanations or repository-facing feedback aligned with how developers review pull requests in practice.

This gap motivates the design of AI-assisted review systems that can examine changed code and produce actionable comments in a form familiar to software teams.
This paper presents Gen-AI Code Reviewer, a web-based platform that automates preliminary review of GitHub pull requests. Rather than replacing human reviewers, the system acts as an initial review layer that highlights suspicious changes, summarizes overall risk, and records structured feedback for later inspection.

The work focuses on practical integration with existing developer workflows and demonstrates how a modern full-stack architecture can combine GitHub APIs, asynchronous processing, persistent storage, and large language model inference in one coherent system.

II. PROBLEM STATEMENT AND MOTIVATION

Manual review remains essential for software correctness and design quality, but the early stages of review often involve repetitive inspection for common issues such as risky edits, poor coding

practices, insecure patterns, and maintainability concerns. This introduces several recurring problems:

- Delayed review cycles due to reviewer workload and timezone differences.
- Inconsistent feedback quality across repositories and team members.
- Difficulty prioritizing risky pull requests when many changes are awaiting review.
- Limited contextual explanation from conventional automated tools for changed diffs.

The motivation behind the proposed system is to reduce this bottleneck by automating the first-pass review stage. The goal is not to provide final approval, but to generate useful review assistance quickly enough that developers can act on it before detailed manual inspection begins.

III. OBJECTIVES AND CONTRIBUTIONS

The major objectives of the project are as follows:

- Develop a web-based AI-assisted platform for pull request analysis.
- Integrate GitHub authentication and repository access with minimal setup friction.
- Extract pulls request diffs and analyze them using a large language model.
- Produce structured output containing summaries, risk indica- tors, and actionable comments.
- Persist reviews and expose them through an interface for future reference.

The main contributions of this work are:

- A practical architecture for AI-assisted review over GitHub pull requests.
- A structured review schema combining summary, severity, category, file path, line reference, and suggested fix.
- An asynchronous processing pipeline for scalable review execution.
- A prototype dashboard that makes generated reviews easy to inspect and revisit.

IV. RELATED WORK

Existing code review support tools generally fall into two broad categories: static-analysis-based systems and AI-assisted systems. Static analyzers are valuable for detecting syntactic, security, and style violations, but they often lack contextual narrative feedback tailored to a pull request's changed diff. Human review platforms such as GitHub provide collaboration and discussion primitives, but the interpretive effort remains manual.

Recent AI-based tools and research prototypes have shown that large language models can assist with code summarization, bug detection, and review comment generation. However, many such systems either operate outside the standard pull request workflow or do not emphasize end-to-end integration with repository authentication, background job orchestration, and persisted review history. The proposed work builds on this direction by combining contextual AI review with a web application architecture suitable for practical developer use.

V. SYSTEM OVERVIEW

A. Workflow

The proposed platform follows a straightforward operational pipeline:

1) The user signs in using GitHub OAuth.
2) The system fetches repositories accessible to the authenti- cated user.
3) A selected pull request is queried through the GitHub REST API.
4) Changed files and textual diffs are transformed into a structured AI prompt.
5) The AI model returns a machine-readable review response. 6) The response is validated, stored, and displayed on the review dashboard.

B. Architecture

The platform is implemented as a modular full-stack appli- cation. The frontend is built with Next.js and React, while the backend uses TypeScript, tRPC, Prisma ORM, and PostgreSQL. Authentication is handled with Better Auth using GitHub OAuth. Review generation is executed asynchronously through Inngest

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719198

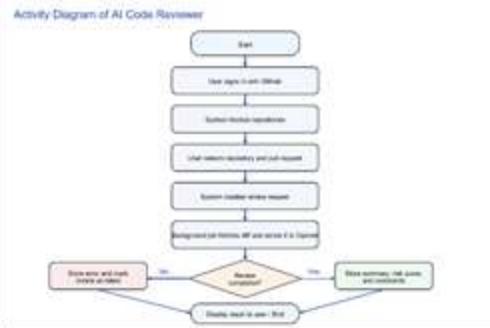

Fig. 1. Operational workflow of the Gen-AI Code Reviewer platform.

so that large or slow model calls do not block the interactive user flow.

At a high level, the architecture contains five cooperating layers:

- presentation layer for dashboard pages and review views,
- integration layer for GitHub repository and pull request retrieval,
- orchestration layer for queued and background review execu- tion,
- intelligence layer for prompt construction and model infer- ence,
- persistence layer for review history, metadata, and generated comments.

Figure 1 illustrates the overall project workflow used in the implemented prototype, showing the movement from repository selection and pull request retrieval to AI analysis and result presentation.

VI. REVIEW GENERATION METHODOLOGY

The quality of an AI-generated review depends strongly on the structure of the prompt and the predictability of the output schema. In the proposed system, pull request title metadata and changed-file patches are combined into a single prompt that asks the model to inspect code for correctness, security, performance, style, and maintainability issues.

If the pull request title is denoted by$T$ and the set of diff patches is denoted by $D = \{d1, d2, \ldots, dn\}$, then the constructed review context can be expressed as:

$$P=T\cup D$$

where P is the complete prompt payload sent for model reasoning.

The generated review output is normalized into a structured format containing:

- a concise natural-language summary,
- a numerical risk score$R$in the range $0\leq R\leq 100$,
- a set of comments$C= \{c1, c2, \ldots, cm\}$,
- severity labels such as Critical, High, Medium, and Low,
- optional suggested fixes when the model can provide a clear correction.

The review prioritization logic can be interpreted using a weighted severity model:

$$R = \alpha|C_{critical}| + \beta|C_{high}| + \gamma|C_{medium}| + \delta|C_{low}|$$

where$\alpha>\beta>\gamma>\delta>0$ and the final score is normalized to the [0, 100] interval. Although the current system stores the model-produced score directly, this formulation provides a rational basis for severity-driven prioritization.

VII. IMPLEMENTATION DETAILS

The implementation adopts technologies commonly used in modern production-grade TypeScript systems.

A. Frontend and User Experience

The frontend uses Next.js for routing and rendering, together with React for the interface layer. Users can connect repositories, browse pull requests, trigger AI reviews, and inspect completed results from a centralized dashboard. The review pages display status transitions such as pending, processing, completed, and failed, which improves transparency during asynchronous execution.

B. Backend Services

The backend coordinates authentication, GitHub API inter- action, prompt generation, schema validation, and persistence. tRPC provides type-safe communication between client and server.

© JUN 2026 | IRE Journals | Volume 9 Issue 12 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I12-1719198

Prisma ORM maps review entities to PostgreSQL tables and simplifies querying of prior review history.

C. Asynchronous Processing

Because AI inference may take multiple seconds and may depend on network-bound API calls, review execution is dispatched through Inngest background functions. This avoids blocking the request-response cycle and allows the system to scale more gracefully when multiple reviews are queued.

VIII. PROTOTYPE OUTPUT CHARACTERISTICS

The prototype produces review outputs with the following practical characteristics:

- a single pull request can be transformed into a concise review summary,
- comments are associated with file context and issue category,
- a risk score helps prioritize manual reviewer attention,
- completed reviews remain available for later comparison and auditing.

In a representative project scenario, the platform combines repository connectivity, pull request retrieval, structured analy- sis, asynchronous processing, and persistent result management into one coherent workflow. The observed system behavior suggests that the platform is suitable for first-pass triage and for surfacing potentially problematic changes early in the review lifecycle.

Figure 2 presents a high-level system build-up view using the project data-flow representation. It highlights how users, GitHub services, the AI review engine, and persistence components cooperate during review generation.

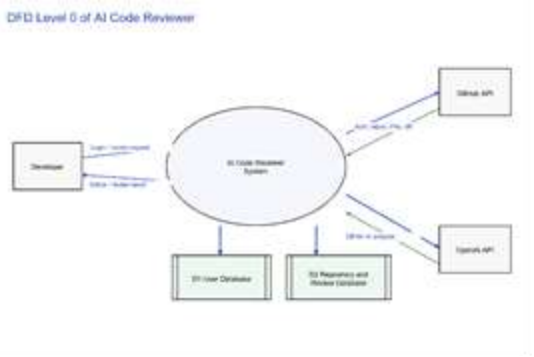

Fig. 2. High-level system build-up and data-flow view of the proposed platform.

IX. DISCUSSION

The proposed approach offers several advantages over purely manual first-pass review. It can generate rapid preliminary feedback, preserve consistency across pull requests, and reduce repetitive reviewer effort on routine checks. By operating on changed diffs instead of full repositories, the system also aligns naturally with the pull request abstraction used by software teams.

At the same time, the system has important limitations. Model output quality depends on prompt quality and may vary across pull request size and code complexity. Repository-wide architectural context is not always available when only changed diffs are analyzed.

Extremely large pull requests may exceed practical prompt limits, and binary or non-textual changes remain outside the scope of useful AI review. For these reasons, the platform should be viewed as a review assistant rather than a replacement for human judgment.

X. PRACTICALRELEVANCE

The project is relevant to both academia and industry. From an academic perspective, it demonstrates the integration of web engineering, OAuth-based security, database-backed persistence, asynchronous workflows, and applied AI in a single software artifact. From an industrial perspective, it addresses the practical problem of pull request review latency and proposes a lightweight mechanism for early issue

surfacing without requiring teams to abandon existing GitHub workflows.

Potential application areas include student-team repositories, startup engineering teams, internal enterprise review pipelines, and open-source projects that need quicker first-pass inspection before deeper human review.

XI. FUTURE WORK

Several extensions can improve the current prototype:

- support for additional platforms such as GitLab and Bit- bucket,
- direct publishing of AI comments back to pull request discussion threads,
- organization-specific review policies and customizable prompt rules,
- repository-wide context retrieval beyond the local diff,
- notification integration with email or collaboration tools,
- comparative analysis of repeated reviews across pulls request revisions.

XII. CONCLUSION

This paper presented Gen-AI Code Reviewer, an AI-assisted platform for preliminary GitHub pulls request review. The system combines GitHub integration, asynchronous processing, full- stack web technologies, persistent review storage, and large language model analysis into a practical developer-support tool.

The resulting prototype demonstrates that generative AI can be integrated meaningfully into software engineering workflows to reduce first-pass review effort, improve consistency of early feedback, and help reviewers focus attention on risky changes more efficiently.

While limitations remain in context depth, prompt scale, and model reliability, the proposed system offers a strong foundation for future research and deployment-oriented refinement in AI-assisted software quality assurance.

REFERENCES

[1] GitHub, Inc., “Collaborating with Pull Requests,” GitHub Documenta- tion, 2024. [Online]. Available: https://docs.github.com/en/pull-requests/ collaborating-with-pull-requests

[2] Vercel, Inc., “Next.js Documentation—The React Framework for the Web,” Next.js Official Documentation, 2024. [Online]. Available: https: //nextjs.org/docs

[3] Prisma Data, Inc., “Prisma ORM — Next-Generation Node.js and TypeScript ORM,” Prisma Official Documentation, 2024. [Online]. Available: https://www.prisma.io/docs

[4] xAI Corp., “Grok AI Model API Documentation,” xAI Developer Documentation, 2024. [Online]. Available: https://docs.x.ai/docs

[5] Better Auth Contributors, “Better Auth Documentation,” 2024. [Online]. Available: https://www.better-auth.com/docs

[6] T. Fu and M. Zhang, “AICodeReview: Advancing Code Quality with AI- Enhanced Reviews,” SSRN Electronic Journal, 2023. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4575968

[7] Inngest, Inc., “Inngest—Durable Workflow Orchestration Documen- tation,” Inngest Official Documentation, 2024. [Online]. Available: https://www.inngest.com/docs

[8] tRPC Contributors, “tRPC—End-to-End Type-Safe APIs for TypeScript,”tRPC Official Documentation, 2024. [Online]. Available: https://trpc.io/ docs

[9] TanStack Contributors, “TanStack Query—Asynchronous State Manage- ment for React,” TanStack Official Documentation, 2024. [Online]. Avail- able: https://tanstack.com/query/latest/docs/framework/react/overview