AI-Powered Web Code Generator Using Project History

Mrs. R. Sivagami
Associate professor,
, Department of Information Technology
SNS College of Technology
Coimbatore, India
sivagami.r.ad@snsce.ac.in

Harini K
Student,
Department of Information Technology
SNS College of Technology
Coimbatore, India
harinikandasamy06@gmail.com

Dharanish N
Student,
Department of Information Technology
SNS College of Technology
Coimbatore, India
dharanish06n@gmail.com

Harivarshan UG
Student,
Department of Information Technology
SNS College of Technology
Coimbatore, India
harivarshanganapathi09@gmail.com

Dr. P. Sumathi
Head of the Department
Department of Information Technology
SNS College of Technology
Coimbatore, India
psumathi.it@gmail.com

ABSTRACT: ***In the age of rapid software development, intelligent automation tools are transforming the way applications are conceptualized, designed, and implemented. This paper introduces a web-based platform that enables users to generate functional software projects or applications based on natural language prompts. By integrating AI code generation with organizational codebase history, the system offers tailored code snippets reflective of the company's prior projects. Users can sign in via GitHub to receive code as a repository or receive TypeScript files directly through the interface. This platform aims to streamline prototyping and enhance developer productivity, especially in startup and fast-paced environments. The solution promotes code reusability and consistency while reducing the overhead of manual setup. Its seamless integration of AI and developer workflows exemplifies the next evolution in intelligent software tooling.***

Keywords: ***AI code generation, Prompt-based development, Web-based development platform, GitHub integration, TypeScript code generation, Code reuse, Developer productivity, Intelligent automation, Organizational codebase learning, Rapid prototyping***

I. INTRODUCTION

In recent years, the need for rapid and efficient software development has grown significantly across various industries. With increasing demand for customized applications and quick prototyping, many developers and businesses are looking for smarter ways to start projects without building everything from scratch. One common challenge developers face is setting up the basic structure of an application, especially when working with specific company standards or technology stacks. Writing boilerplate code repeatedly not only consumes time but can also lead to inconsistencies and slower development cycles.

To address this, we have built a web-based AI-powered platform that helps users generate complete software projects based on simple natural language prompts. Instead of writing code manually or reusing outdated templates, users can describe what they want, and the system will automatically generate a project that aligns with their requirements. The platform is trained on previous projects completed by the organization, ensuring that the code output follows internal guidelines and best practices.

Users can sign in using their GitHub account to receive the generated code directly in a GitHub repository. Alternatively, they can download the code as a TypeScript project through the web interface. The platform supports features like GitHub login, session-based access for guests, and delivers well-structured frontend or full-stack projects. Built with modern technologies like React, NodeJS, and AI language models, this tool focuses on saving time, reducing repetitive coding tasks, and improving consistency in project creation especially for teams, startups, and new developers who want to move fast without compromising on quality.

II. RELATED SYSTEM

In the past few years, several tools have been developed to assist developers in generating code and bootstrapping new projects. Platforms like GitHub Copilot and Replit Ghostwriter are widely known for offering real-time code suggestions based on AI. GitHub Copilot integrates directly with code editors and helps by predicting the next line of code, but it mainly focuses on in-editor support and does not generate full project structures. Similarly, Replit Ghostwriter supports AI suggestions while coding but lacks support for organization-specific templates or code reuse.

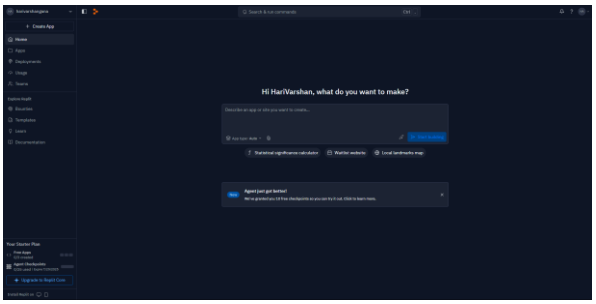

Fig 1. Replit Interface

Other tools like Create React App, Next.js CLI, and Yeoman offer scaffolding capabilities to quickly set up project environments. While useful, they are static in nature and do not understand natural language prompts. They also require manual customization, which can be time-consuming for beginners or those working under tight deadlines.

Some AI platforms like OpenAI Codex and CodeT5 allow developers to convert text prompts into code snippets, but these tools are mostly limited to generating smaller blocks of code and do not provide end-to-end application structures tailored to previous company projects.

Unlike the tools mentioned above, our system is designed for both developers and non-developers who want to create complete project setups using simple English prompts. It leverages the company's past projects to ensure the generated code is familiar and production-ready. The platform also supports GitHub authentication for repository delivery or provides code as TypeScript files, offering a flexible and beginner-friendly solution suitable for fast-moving teams and individuals.

III. PROPOSED SYSTEM

The proposed system is designed to simplify and accelerate the process of creating software projects by using artificial intelligence and organizational code history. Traditionally, developers start new applications by setting up boilerplate code, choosing libraries, and manually aligning the structure with company standards. This process is repetitive, time-consuming, and often inconsistent—especially for new developers or fast-moving teams.

In our system, that entire process is replaced by a single step the user simply enters a prompt describing the type of application they want to build. Once submitted, the system processes the prompt and uses AI to understand the requirements. It then searches through the company's historical codebase and project templates to generate a new project that matches both the request and the company's coding style.

If the user logs in via GitHub, the system automatically creates a private repository and pushes the generated code there. For users who prefer not to log in, the code is provided as a downloadable TypeScript project. This makes it easy for both professional developers and beginners to use the tool without any complicated setup.

Fig 2. The homepage of the application, where users create applications in their preferred format

The AI engine is trained to recognize common patterns in the organization's previous projects. It generates components, folders, configuration files, and even starter logic that matches existing best practices. The platform also includes an AI formatter, which reads the formatting style (indentation, naming, comments) and applies it across the codebase to keep everything neat and professional.

In addition to project generation, the system includes prompt history tracking, allowing users to revisit or reuse previously entered prompts. This is useful for iterative development or when building multiple modules with similar structures. The frontend, built using ReactJS, provides a clean and responsive interface that guides users step by step through the process. Users receive real-time feedback and can regenerate code if the output does not meet expectations.

The backend is developed using NodeJS and Express, and all project data, user sessions, and prompt logs are securely stored in MongoDB. Security measures such as token-based authentication, encrypted session storage, and access-controlled APIs ensure that user data remains safe. The integration of AI not only improves speed but also helps users learn how different components are structured, encouraging better understanding of the codebase.

This tool is especially helpful for students, early-career developers, and startup teams who need to quickly launch new ideas with high-quality, consistent code. It removes the hassle of manual setup and delivers clean, usable output with just one prompt—making project development faster, smarter, and more accessible.

IV. METHODOLOGY

The proposed system is built using a full-stack architecture with modern web technologies and AI capabilities to assist users in generating complete software projects from simple prompts. Unlike traditional methods where developers manually set up a new codebase or use limited scaffolding tools, this system creates a personalized project based on previous organizational standards and natural language input.

The flow begins when the user visits the web platform and enters a project prompt describing the desired application. The backend, developed using Node.js and Express.js, receives this prompt and interacts with an AI model trained to understand programming patterns and code generation. Based on the prompt, it generates a structured project using company-specific templates and best practices stored from past projects.

If the user signs in using GitHub, the platform automatically creates a private repository in the user's GitHub account and uploads the generated code. For non-authenticated users, the system provides a downloadable TypeScript project. All generated files are structured into frontend and backend folders (if required), including components, pages, configuration files, and basic documentation.

To maintain consistency, the AI analyzes prior formatting styles such as indentation, file organization, and naming conventions. The Gemini AI API (or equivalent LLM) is used to enhance the quality of generated code and ensure alignment with modern standards. Tailwind CSS and Framer Motion are used on the frontend to ensure a clean, animated, and responsive user experience.The system is modular and contains five major components described below:

Module 1: Prompt-Based Project Generation Module

This is the core module of the system, responsible for transforming user prompts into fully structured software projects. Instead of requiring users to upload files or fill out lengthy forms, the platform simplifies the process—users just enter a short, natural language prompt such as "A to-do app with user login and dashboard." The backend interprets this prompt using advanced AI models trained on historical company projects and generates a complete, ready-to-run codebase. This includes appropriate file structures, routes, reusable components, and sample data. For authenticated GitHub users, the code is automatically pushed to a new private repository, ensuring seamless and secure integration.

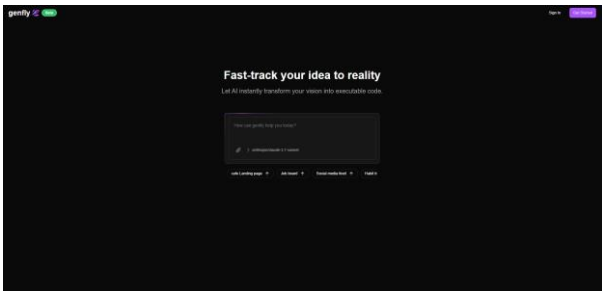

Fig 3. . The project creation screen where users enter prompts and choose GitHub or direct download

Module 2: AI-Powered Code Structuring and Enhancement Module

After the initial code generation, this module further enhances the project using AI-driven refinement techniques. It thoroughly analyzes the generated code structure and adjusts it to align with organizational best practices, such as standardized folder naming, creation of reusable components, and inclusion of essential configuration files. The system also automatically identifies and adds missing elements like README documentation, .env templates, license files, and basic API handlers. Using the Gemini API, the module rewrites or restructures any sections that deviate from clean coding standards. This process ensures the final output is consistent, maintainable, scalable, and ready for real-world deployment scenarios.

Module 3: Code Formatter and Style Enforcer Module

This module ensures that the entire codebase maintains a consistent and professional coding style. It analyzes existing files to detect patterns such as indentation preferences (tabs or spaces), naming conventions (camelCase, PascalCase), and comment formatting styles. Based on this analysis, the system automatically applies the same style uniformly across all generated files. Even if the original data came from different developers with varied styles, the formatter harmonizes everything to follow a single, clean standard. It also removes unnecessary imports, reformats lengthy lines, organizes code blocks, and optimizes readability, ensuring that every file is polished, consistent, and ready for deployment.

Module 4: Authentication and Repository Integration Module

This module handles user authentication and ensures secure access to the platform. Users have the option to log in using their GitHub account via OAuth or sign up using a traditional email and password method. GitHub authentication enables the system to directly access the user's repositories and create new ones for project deployment. For email signups, user credentials are securely encrypted and stored in MongoDB. In case of a forgotten password, an OTP-based recovery system is implemented, sending a verification code to the user's registered email. All application routes are protected using access tokens and middleware, ensuring user data and project history remain private and secure

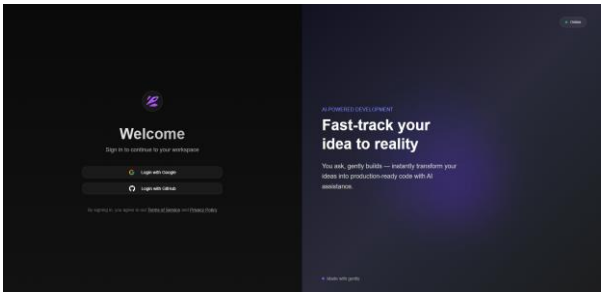

Fig 4. The login page supporting GitHub, Google, and email authentication

Module 5: UI Interaction and Output Delivery

This module controls how the output is displayed and downloaded. After project generation is completed, users are taken to a summary screen where they can view file structures, download the ZIP file, or navigate to the GitHub repo. The frontend is developed using **React.js**, styled with **Tailwind CSS**, and animated with **Framer Motion** for a modern and responsive interface. The project preview and status indicators help users understand the generation process clearly. Even users with little coding knowledge can use the system and understand what each file does through the AI-generated documentation included in the project.

V. SYSTEM ARCHITECTURE

The architecture of this project is designed to deliver fast, reliable, and intelligent application generation, combining a full-stack web system with advanced AI services. It is structured into four primary layers: the user interface layer, the server-side logic layer, the AI and code generation layer, and the database layer. Each layer interacts with others through secure, modular APIs, ensuring smooth and scalable performance.

1. User Interface Layer:
The frontend is where users interact with the platform. Built using ReactJS, it offers a dynamic and highly responsive experience. Users can enter a prompt to describe the application they want, monitor project generation progress, and access generated codebases. It also handles user authentication through GitHub OAuth or email-based login. Tailwind CSS is used for sleek styling, while Framer Motion adds fluid animations, resulting in a professional, modern user experience.

2. Server-Side Logic Layer
This layer is built using Node.js and Express.js. It handles incoming API requests from the frontend, such as cloning repositories, scanning files, and sending them for processing. This layer also validates GitHub links, manages authentication logic, and handles communication between the database and AI services. It acts as the main controller of the entire system.

3. AI and Code Processing Layer

At the core of the system lies the AI and Code Generation Layer. Once a user submits a prompt, this layer interprets the request using AI models trained on historical organizational projects. It builds a full, functional project structure, including components, routes, APIs, and configuration files. The Gemini API is integrated to optimize and refine the generated code, ensuring real-world deployment standards. AI formatting tools also ensure code consistency, cleaning up styles, removing unused elements, and maintaining best practices across the project.

4.Database layer:

MongoDB is used as the primary database to store user information, project prompts, generated project histories, and system logs. All sensitive information, such as passwords, is securely encrypted. This layer also maintains user-specific data like saved projects and login sessions, enabling a personalized experience for returning users.

5.Integration layer:

The process begins when a user enters a prompt or logs in via GitHub. The frontend securely sends this input to the backend, which triggers the AI module to generate the project. The code is then either pushed to a GitHub repository or returned as downloadable TypeScript files. All communication uses secure REST APIs, and MongoDB stores project history and user data for future access.

DATAFLOW DIAGRAM

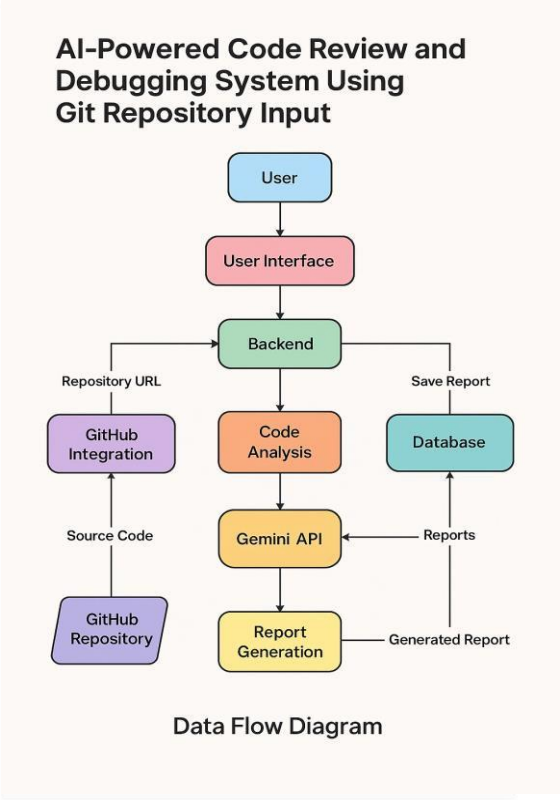

Fig 5. A flow diagram showing the working of the project

VI. CONCLUSION

The main goal of this project was to simplify and accelerate application development by leveraging artificial intelligence and organizational code history. Many developers, especially students and early-stage teams, often struggle to kick-start projects efficiently or maintain consistency across their codebase. This platform solves that by allowing users to generate entire applications from simple prompts or sync code directly via GitHub. From generating boilerplate code to customizing structure based on past projects, the system ensures fast and reliable output with minimal manual effort.

The integration of the Gemini API adds an intelligent layer, refining the code to meet professional standards, while the formatter ensures style consistency across the project. Secure login options through GitHub OAuth and email, OTP-based password recovery, and an intuitive frontend built with ReactJS make the platform accessible to both beginners and experienced users. By automating key tasks like code generation, formatting, and repository management, the system boosts productivity and reduces dependency on senior developers or third-party tools.

Overall, the project combines AI capabilities, code automation, and user-focused design to create a powerful tool for developers looking to quickly prototype or build projects with clean, consistent code. It is designed to be scalable, extensible, and especially valuable for startups, academic projects, and small teams.

VII. REFERENCES

[1] A. Kumar, B. Singh, and S. R. R. K. Varma, "AI-Based Code Review System: A Survey," International Journal of Computer Applications, vol. 184, no. 6, pp. 50–55, Jul. 2024.

[2] S. Sharma, R. Gupta, and P. Shah, "Improving Code Quality Using AI and Static Analysis Tools," Journal of Software Engineering, vol. 22, no. 4, pp. 180–192, Dec. 2024.

[3] M. Patel and D. K. Verma, "Automation of Code Reviews Using Deep Learning Techniques," IEEE Access, vol. 11, pp. 42312–42325, Apr2023. doi:10.1109/ACCESS.2023.3267812.

[4] J. K. Lee and H. S. Kim, "Smart IDE Plugin for Real-Time Bug Detection Using Machine Learning," in Proc. Int. Conf. Artificial Intelligence and Software Engineering (ICAISE), Singapore, 2023, pp. 89–94.

[5] SonarSource, "SonarQube: Continuous Inspection of Code Quality," [Online]. Available: https://www.sonarqube.org

[6] GitHub, "Cloning a Repository," [Online]. Available: https://docs.github.com/en/repositories/creating-and- managing-repositories/cloning-a-repository.

[7] Gemini AI, "Gemini API Documentation," [Online]. Available: https://gemini.google.com/api

[8] DeepSource, "DeepSource: Automated Code Review with Static Analysis," [Online]. Available: https://deepsource.io