International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 1, 2026

Codegen: An AI-Powered Visual Studio Code Extension

Dr R Dinesh Kumar[1],Ch Advaitha[2],A Manasa[3],K Abhinaya Yadav[4]

[1]Associate Professor; Department Of Computer Science And Engineering Bhoj Reddy Engineering College for Women Hyderabad India.

[2,3,4]B.Tech Students; Department Of Computer Science And Engineering Bhoj Reddy Engineering College for Women Hyderabad India.

Mail Id; me.dineshkumar@gmail.com[1],ammurao54@gmail.com[2],manasaalladi1@gmail.com[3], abhinaya.yadav106@gmail.com[4]

Abstract

Artificial intelligence is increasingly transforming the way software developers write, understand, and maintain code. This paper presents CodeGen, an intelligent extension for Visual Studio Code that provides context-aware programming assistance using locally deployed AI models. The system integrates the DeepSeek Coder model to translate natural language descriptions into executable code snippets, enabling developers to generate code simply by describing the intended functionality.Beyond code generation, CodeGen incorporates several advanced development features including automated code explanation, debugging assistance, context-aware autocomplete, code refactoring support, and automatic test case generation. These capabilities not only accelerate the software development process but also support learning by helping developers better understand generated code and programming concepts.A key distinguishing aspect of CodeGen is its fully offline architecture. Unlike many existing AI-based coding assistants that rely on cloud services, CodeGen runs entirely on locally deployed models. This design ensures enhanced data privacy, removes dependence on internet connectivity, and minimizes latency, making the tool suitable for environments where security and reliability are critical. Additionally, the extension supports multiple programming languages and integrates seamlessly within the Visual Studio Code ecosystem to maintain a smooth developer workflow.By combining advanced artificial intelligence techniques with local model execution, CodeGen addresses major limitations associated with traditional AI development tools, such as privacy risks, subscription-based access, and workflow disruption caused by external services. The proposed system demonstrates how offline AI integration can improve productivity while maintaining secure and efficient development environments. This work contributes toward the development of intelligent, privacy-preserving programming assistants for modern software engineering.

Keywords: *Artificial Intelligence, Code Generation, DeepSeek Coder, Visual Studio Code Extension, Offline AI Models, Context-Aware Programming Assistance, Automated Code Explanation, Debugging Assistance, Code Refactoring, Automatic Test Case Generation, Privacy-Preserving AI, Software Development Productivity.*

Introduction

The rapid advancement of artificial intelligence has significantly influenced modern software development practices by enabling intelligent tools that assist programmers during coding. This research introduces **CodeGen**, an AI-driven extension developed for **Visual Studio Code** that enhances the programming workflow by providing context-aware assistance directly inside the integrated development environment. The extension is designed to simplify coding tasks and improve developer productivity through automated and intelligent support mechanisms.

CodeGen incorporates the **DeepSeek Coder** model to interpret natural language instructions and convert them into executable code snippets in real time. Through this capability, developers can describe their programming requirements in plain language and instantly obtain relevant code implementations. In addition to code generation, the system offers several advanced capabilities including detailed code explanations, automated debugging support, context-aware autocomplete suggestions, and automatic generation of test cases.

A notable characteristic of CodeGen is its **offline operational capability**. In contrast to many existing AI coding assistants that depend on cloud infrastructure and constant internet access, CodeGen runs locally using a deployed AI model. This approach enhances **data security and privacy**, eliminates reliance on external servers, and reduces response delays caused by network communication. As a result, developers can use the tool efficiently even in restricted or low-connectivity environments while maintaining a secure development workflow.

Scope

The scope of the proposed system focuses on improving developer productivity and learning efficiency by integrating artificial intelligence directly into the coding environment. The main functional objectives include:

- Enabling automatic **code generation** based on natural language instructions provided by the developer.

- Providing **step-by-step explanations** of generated or existing code to improve comprehension and support learning.
- Incorporating **intelligent debugging mechanisms** capable of identifying errors and suggesting potential fixes.
- Implementing **context-aware autocomplete features** that predict relevant code structures and programming patterns.
- Supporting **automated test case creation** to verify functionality and improve software reliability.
- Ensuring the system operates **entirely offline**, thereby protecting developer data and removing dependency on external internet services.

Existing System

In the current software development ecosystem, programmers rely on a variety of tools to assist with coding and debugging tasks. Traditional integrated development environments typically provide fundamental features such as syntax highlighting, basic autocomplete, and built-in debugging utilities. While these tools improve coding efficiency to some extent, they do not provide advanced intelligent assistance capable of understanding developer intent.

Recently, cloud-based artificial intelligence solutions such as **GitHub Copilot** have gained popularity by offering AI-generated code suggestions. However, these tools usually require continuous internet connectivity and often involve subscription-based pricing models. Additionally, developers frequently utilize general-purpose conversational AI platforms for coding guidance, which requires switching between applications and disrupts the development workflow.

For learning and code understanding, programmers commonly consult online resources including official documentation, programming forums, and tutorial platforms. Although these resources are valuable, they can be time-consuming to search and may not always provide solutions tailored to the specific context of a developer's project. Similarly, debugging tasks often depend on manual code inspection, logging statements, and conventional debugging techniques, which can require considerable time and experience to identify and resolve issues effectively.

Proposed System

The proposed **CodeGen** system aims to overcome the limitations of existing development tools by introducing a comprehensive AI-powered coding assistant integrated directly within the Visual Studio Code environment. The system leverages the **DeepSeek Coder** model, which has been specifically optimized for software development tasks, to provide intelligent programming support.

Unlike many AI assistants that rely on remote cloud services, CodeGen operates entirely on the user's local machine. This architecture improves **data privacy, system responsiveness, and reliability**, while also enabling developers to use the system without internet access. The model processes natural language instructions provided by the user and converts them into functional code implementations across multiple programming languages.

Furthermore, the system includes an advanced code explanation module that analyzes generated or existing code and provides clear, step-by-step descriptions of the underlying logic and structure. This feature is particularly useful for educational purposes, collaborative code review, and understanding complex algorithms. The intelligent autocomplete functionality enhances traditional suggestion systems by analyzing the coding context and recommending complete functions, classes, or design patterns rather than simple keyword completions. Through these capabilities, CodeGen aims to create a more efficient, intelligent, and developer-friendly programming environment.

Requirement Analysis

Requirement analysis is an important phase in software development that identifies the capabilities and limitations of a proposed system. It helps define what the system should accomplish and establishes the conditions necessary for its successful operation. In the CodeGen project, requirement analysis focuses on identifying both functional and non-functional aspects of the AI-powered coding assistant. These requirements guide the development process and ensure that the system fulfills the needs of developers who will use the extension within Visual Studio Code. The analysis also determines the necessary computational resources and development process model required to implement the system effectively.

Functional Requirements

Functional requirements describe the specific operations and services that the CodeGen system must perform to support developers. The system includes two primary modules: the **User Module** and the **Extension Administrator Module**.

The **User Module** represents the main interaction interface between developers and the CodeGen extension. Through this module, developers can enter natural language prompts to request various operations such as code generation, code explanation, debugging, and code optimization. The system interprets these prompts using the integrated AI model and produces relevant results directly within the development environment. Users are able to execute generated code through the Visual Studio Code terminal and evaluate its functionality. Additionally, the extension allows users to review AI-generated suggestions and either accept or reject them based on their requirements. The module also supports automatic test case generation, enabling

developers to validate the correctness and reliability of the produced code.

The **Extension Administrator Module** is responsible for maintaining and improving the system. This module is primarily used by the developers or maintainers of the CodeGen extension. It includes managing backend services such as the Flask-based server and integrating the DeepSeek AI model for processing user requests. The administrator is also responsible for updating or fine-tuning the AI model with new datasets to improve accuracy and performance. In addition, this module involves packaging and publishing the extension to the Visual Studio Code Marketplace so that it can be accessed by a wider developer community. Continuous monitoring of system performance, handling error logs, and implementing bug fixes are also essential responsibilities of this module.

Non-Functional Requirements

Non-functional requirements define the quality attributes and operational constraints of the CodeGen system. These requirements ensure that the system performs efficiently, securely, and reliably during use. The system must provide quick responses to developer prompts, allowing tasks such as code generation, explanation, and debugging to be completed with minimal delay. Scalability is another important requirement, as the architecture should be capable of supporting multiple users without reducing system performance.

Usability is a key consideration in the design of the CodeGen extension. The interface should be intuitive and easy to use, allowing developers to interact with AI features directly within the Visual Studio Code environment without requiring complex configurations. Reliability is also critical, and the system should maintain stable operation with minimal downtime. Mechanisms for automatic error handling and recovery should be incorporated to ensure uninterrupted functionality.

Security requirements focus on protecting user data and system integrity. Secure authentication mechanisms, encrypted communication channels, and protection of API keys or configuration data must be implemented to safeguard sensitive information. Compatibility is another important aspect, ensuring that the system can operate effectively across different operating systems and development environments, including Windows-based systems and other commonly used platforms.

Computational Resources

The successful implementation of the CodeGen system requires appropriate computational resources that include both software tools and hardware infrastructure. These resources enable efficient development, deployment, and execution of the AI-powered extension.

Software Requirements

The CodeGen project utilizes widely adopted development tools and programming technologies to build the system. The primary development platform used for implementing the extension is Visual Studio Code running on the Windows 11 operating system. Multiple programming languages are employed to support different aspects of the system. Python is used for implementing backend services and AI model integration, while Node.js is used for extension-related functionalities. Java may also be used for supporting additional system components where required. Data storage and management are handled using lightweight database systems such as SQLite or MySQL. The central intelligence of the platform is powered by the DeepSeek Coder 1.3B model, which is capable of generating code, understanding programming structures, and providing explanations for developers. These software tools collectively enable the development of an efficient and intelligent coding assistant.

Hardware Resources

Hardware requirements represent the physical computing resources necessary to run the CodeGen system effectively. Since the AI model is executed locally, adequate processing power and memory are essential to maintain acceptable system performance. A system equipped with an Intel Core i7 processor of the 11th generation or an equivalent processor is recommended to ensure efficient computation. The system should also include at least 8 GB of DDR4 RAM to support the execution of development tools and AI inference tasks. Storage requirements include a minimum of 512 GB solid-state drive (SSD) to provide sufficient space for system files, development tools, and model data while maintaining fast read and write performance.

Software Process Model

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

Fig 1. Software Process Model

A software process model provides a structured framework that guides the development of a software system from the initial planning stage to deployment and maintenance. It helps development teams organize tasks, manage resources, and ensure that quality standards are maintained throughout the project lifecycle. For the development of the CodeGen system, the Spiral Model is adopted as the primary development methodology.The Spiral Model combines elements of both iterative development and risk management. In this model, the development process is represented as a spiral consisting of multiple loops, where each loop corresponds to a specific phase of the project. Each cycle generally includes planning, risk analysis, development, and testing activities. During the planning stage, project objectives and requirements for the current iteration are identified. The risk analysis stage evaluates potential technical challenges and identifies strategies to mitigate them. In the development stage, system components are designed and implemented based on the defined requirements. Finally, the testing stage verifies system functionality and collects feedback for further improvements.The Spiral Model is particularly suitable for complex projects such as AI-based development tools because it allows gradual refinement of system features through multiple iterations. By continuously evaluating risks and incorporating user feedback, the model helps improve system quality while reducing the likelihood of project failure.

Design

The design phase focuses on transforming system requirements into a structured technical solution. It involves planning how the system components will interact, determining appropriate algorithms, and defining the architecture that supports system functionality. In the CodeGen project, the design process includes defining the architecture of the extension, identifying relevant features for the AI model, and establishing strategies for training and evaluating the model's performance.The design also considers user interface elements to ensure that developers can easily interact with the system within the Visual Studio Code environment. Proper deployment planning is also essential so that the extension integrates seamlessly with existing development workflows. A well-defined design helps manage system complexity and ensures that the final solution meets both functional and performance expectations.

Architecture

System architecture provides a high-level representation of the components involved in the project and describes how they interact during the processing of user requests. It explains how data flows between modules and how each component contributes to the overall functionality of the system. In the CodeGen project, the architecture is divided into two main categories: software architecture and technical architecture.

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

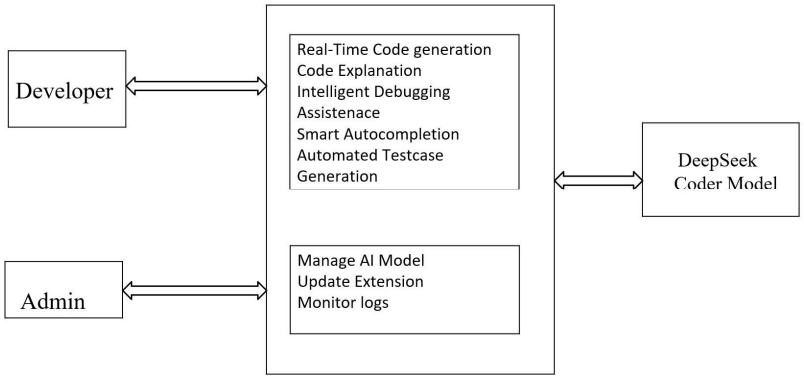


Fig 2;Software Architecture

Software architecture describes the logical structure of the system and defines the relationships between different software components. A well-designed software architecture helps developers build systems that are secure, efficient, and easy to maintain. It enables early identification of potential software defects and security vulnerabilities during the design stage. By analyzing the architecture, developers can evaluate possible threats, detect design weaknesses, and implement corrective measures before deployment. This process improves system reliability and reduces the likelihood of errors during runtime.

Technical Architecture

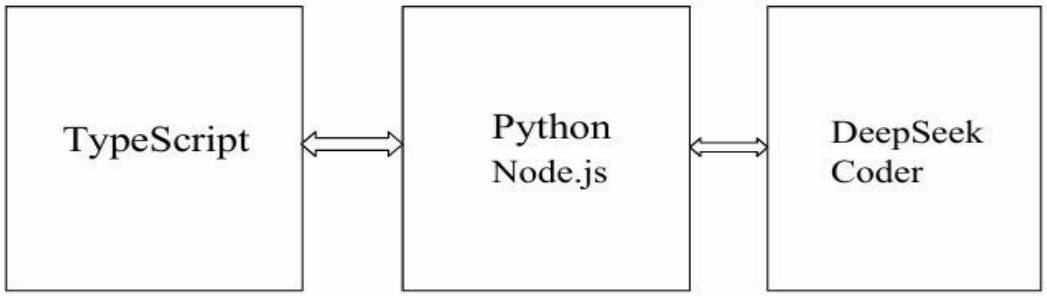


Fig 3;Technical Architecture

Technical architecture focuses on the infrastructure and technological framework required to implement the system. It describes how hardware resources, software platforms, and communication mechanisms interact to support system operations. In the CodeGen system, the technical architecture includes the Visual Studio Code extension interface, the backend processing layer implemented using a Flask server, the locally deployed DeepSeek AI model, and the database used for storing configuration data and logs. These components work together to process user requests, generate code responses, and deliver results directly within the integrated development environment.

UML Diagrams

Unified Modeling Language (UML) is a standardized visual modeling language used to represent the structure and behavior of software systems. UML diagrams help developers understand how different components of a system interact and provide a graphical representation of system architecture. These diagrams are widely used in software engineering to illustrate actors, system processes, components, and database structures. By using UML diagrams, developers can analyze the design of the system more effectively and communicate design concepts clearly among project stakeholders.

Use Case Diagram

A use case diagram illustrates the interaction between external actors and the system. It identifies the functionalities that the system provides and shows how users interact with these services. Actors represent external entities such as developers or administrators who interact with the system. Use cases represent specific functionalities provided by the system, such as generating code, debugging programs, or explaining code. The relationships between actors and use cases indicate how users access different system capabilities. Use case diagrams are particularly useful for understanding system functionality from the user's perspective.

Class Diagram

A class diagram is a structural UML diagram used to represent the static structure of an object-oriented system. It illustrates the classes present in the system along with their attributes and methods. The diagram also shows relationships between classes, such as

associations, inheritance, and dependencies. Class diagrams are commonly used during the system design phase to help developers visualize how different objects interact and how the system components are organized.

Implementation

The implementation phase focuses on transforming the designed architecture into a fully functional system. It involves integrating the frontend interface, backend processing modules, and artificial intelligence components to provide intelligent coding assistance within the development environment. The CodeGen extension is implemented as a Visual Studio Code plugin that communicates with a locally deployed AI model through a backend server. The system architecture ensures seamless interaction between the user interface, request processing layer, and the AI model responsible for generating coding solutions.

Frontend Technologies

The frontend component is responsible for providing an interactive interface through which developers can communicate with the CodeGen extension. Web technologies such as **HTML**, **CSS**, and **JavaScript** are used to design and implement this interface. HTML is used to define the structure and layout of the user interface elements, including the prompt input panel and result display sections. CSS is applied to enhance the visual appearance of the interface and ensure a clean and user-friendly layout that integrates smoothly with the Visual Studio Code environment. JavaScript is used to implement dynamic functionality such as handling user inputs, triggering extension commands, and managing communication between the user interface and backend services. Through these technologies, developers can interact with the system efficiently while writing code.

Backend Technologies

The backend component is responsible for processing user requests and interacting with the artificial intelligence model. The server-side logic is implemented using **Python** with the **Flask framework**, which provides a lightweight and efficient environment for building APIs and handling communication between the extension and the AI model. When the extension sends a request, the Flask server receives the input prompt, processes it, and forwards it to the AI model for code generation or explanation.

The system utilizes the **DeepSeek Coder 1.3B model**, which is specifically designed for software development tasks such as code generation, debugging, and explanation. The model analyzes user prompts and produces relevant programming solutions based on the provided instructions. To store system data such as user prompts, generated outputs, and configuration information, lightweight database systems such as **SQLite** or **MySQL** are used. These databases ensure efficient data storage and retrieval while maintaining system performance.

Implementation Process

The implementation of the CodeGen system follows a sequence of steps that define how user requests are processed and transformed into meaningful outputs. The process begins when the developer installs and activates the CodeGen extension within the Visual Studio Code environment. Once the extension is installed, the user can open any programming file, such as a Python or Java file, and start interacting with the extension. Before processing requests, the backend server must be running locally to enable communication between the extension and the AI model.

Finally, the processed result is sent back to the Visual Studio Code extension. The output can be displayed in two different ways depending on the interaction mode selected by the user. In inline mode, the generated code is inserted directly into the editor at the appropriate position. In WebView mode, the result is displayed within a dedicated output panel where the user can review the generated content before integrating it into the project.

Algorithmic Workflow (Pseudocode Description)

The internal operation of the CodeGen system can be described through a sequence of algorithmic steps that illustrate how the extension interacts with the backend server and AI model.On the backend side, the Flask server loads the DeepSeek Coder model using appropriate machine learning libraries and prepares the input prompt for processing. The model generates a response based on the provided instruction, producing either executable code or descriptive explanations. The generated output is then cleaned and formatted to remove unnecessary text and ensure that the result contains only the relevant code or explanation.Finally, the extension extracts the generated code block from the response and inserts it into the user's editor or displays it in the output interface. This process completes the interaction cycle between the developer and the AI-powered coding assistant.

Testing

Software testing is a process, to evaluate the functionality of a software application with an intent to find whether the developed software met the specified requirements or not and to identify the defects to ensure that the product is defect free in order to produce the quality product. As per the current trend, due to constant change and development in digitization, our lives are improving in all areas. The way we work is also changed. We access our bank online we do shop online; we order food online and many more. We rely on software's and systems. What if these systems turnout to be defective? We all know that one small bug shows huge impact on business in terms of financial loss and goodwill. To deliver a quality product, we need to have Software Testing in the Software

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026

Development Process. Some of the reasons why software testing becomes very significant and integral part in the field of information technology are as follows.

Stages of Testing

During This first round of testing, the program is submitted to assessments that focus on specific units or components of the software to determine whether each one is fully functional. In this phase, a unit can refer to a function, individual program or even a procedure, and White box testing method is usually used to get the job done.

Integration Testing

Integration testing allows individuals the opportunity to combine all of the units within a program and test them as a group. This testing level is designed to find interface defects between the modules/functions. This is particularly beneficial because it determines how efficiently the units are running together. Keep in mind that no matter how efficiently each unit is running, if they properly integrated, it will affect the functionality of the software program.

System Testing

System testing is the first level in which the complete application is tested as a whole. The goal at this level is to evaluate whether the system has complied with all of the outlined requirements and to see that it meets Quality Standards. System testing is undertaken by independent testers who haven't played a role in developing the program. This testing is performed in an environment that closely mirrors production. System Testing is very important because it verifies that the application meets the technical, functional, and business requirements that were set by the customer.

Acceptance Testing

The final level, Acceptance testing (or User Acceptance Testing), is conducted to determine whether the system is ready for release. During the Software development life cycle, requirements changes can sometimes be misinterpreted in a fashion that does not meet the intended needs of the users. During this final phase, the user will test the system to find out whether the application meets their business needs. Once this process has been completed and the software has passed, the program will then be delivered to production. The extensiveness of these tests is just another reason why bringing software testers in early is important. When a program is more thoroughly tested, a greater number of bugs will be detected; this ultimately results in higher quality software.

Types of Testing

Black box testing

It is also called as Behavioral/Specification-Based/Input-Output Testing. Black Box Testing is a software testing method in which testers evaluate the functionality of the software under test without looking at the internal code structure. This can be applied to every level of software testing such as Unit, Integration, System and Acceptance Testing.

White box testing

It is also called as Glass Box, Clear Box, Structural Testing. White Box Testing is based on applications internal code structure. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. This testing usually done at the unit level.

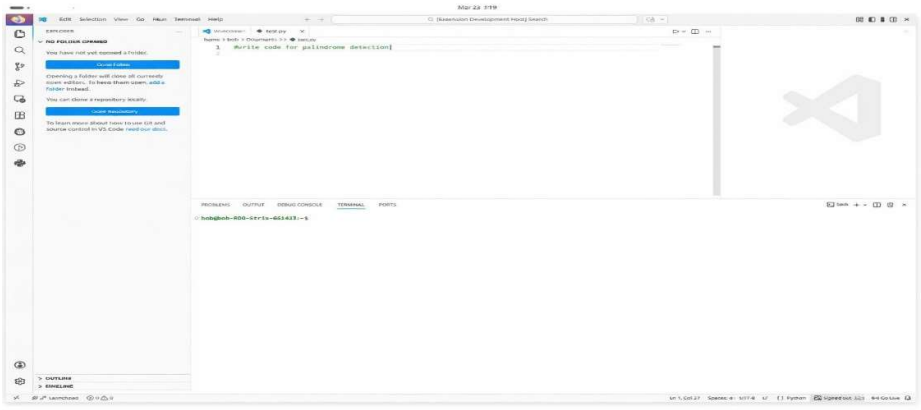
White Box Testing Techniques:

Screenshots

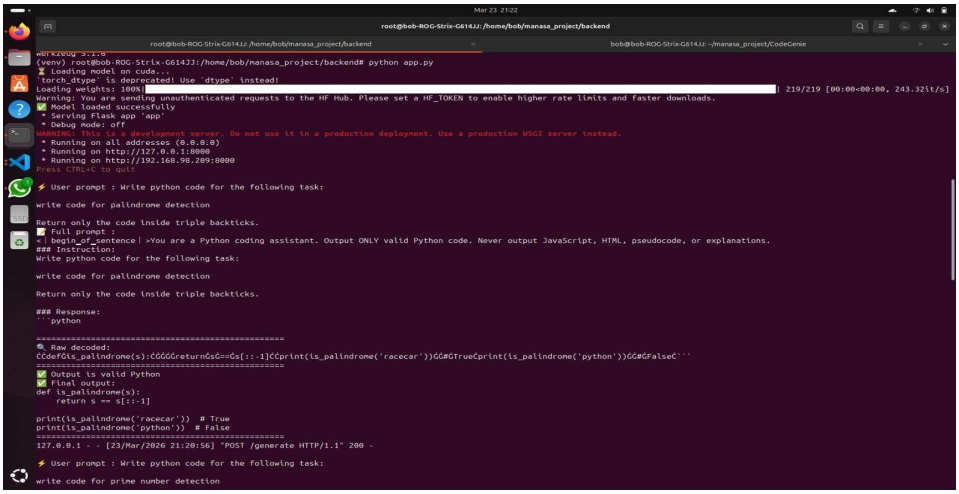
VS Code Extension Window

Screenshot 1;VS Code Extension Window

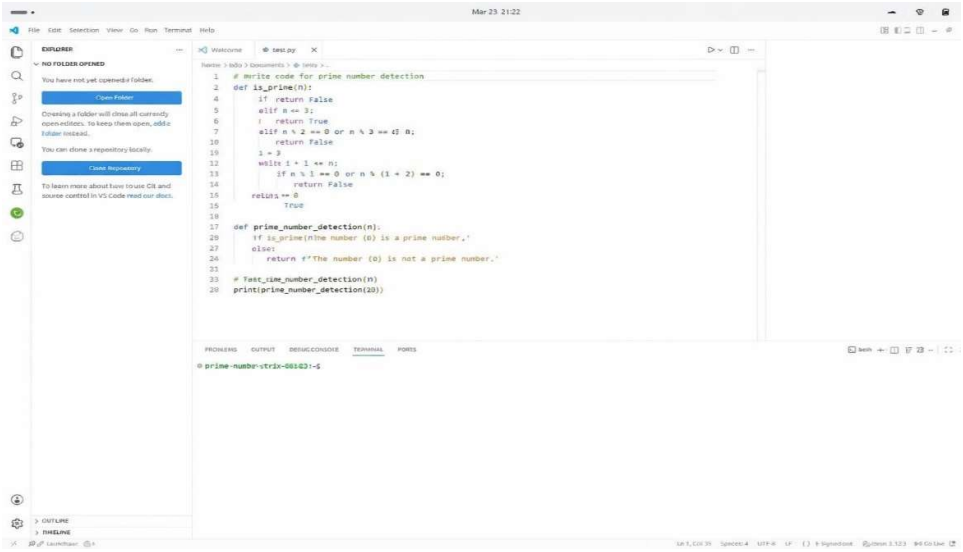
International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026



Screenshot 2;User prompt using

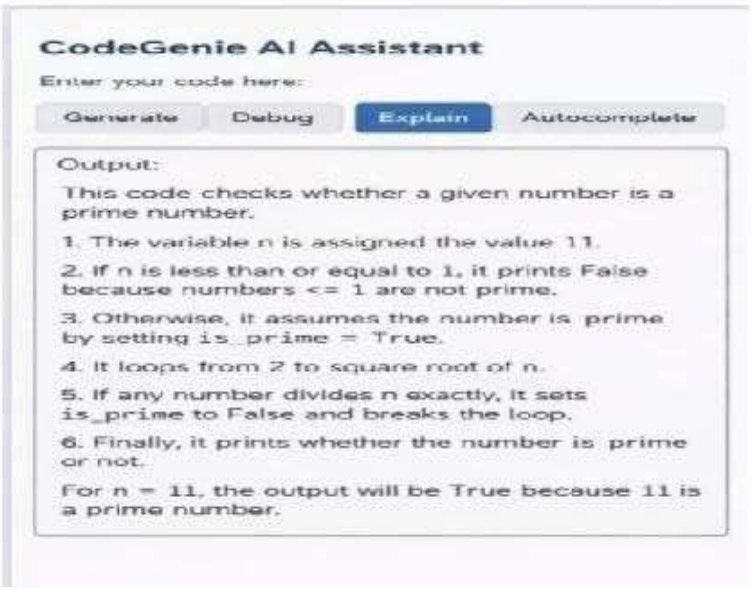


Screenshot 3; Backend Log



Screenshot 4; Generated Code Using Extension

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 1, 2026



Screenshot 6; Explain

Conclusion

The development of **CodeGen**, an AI-powered extension for Visual Studio Code, demonstrates the potential of artificial intelligence to improve modern software development practices. By integrating intelligent coding support directly within the development environment, the system assists programmers in performing various tasks more efficiently. The extension provides features such as natural language–based code generation, automated debugging assistance, structured code explanations, and automatic test case creation. These capabilities enable developers to reduce development time while also improving their understanding of programming logic and implementation techniques.

A key advantage of the proposed system is its ability to operate with locally deployed AI models, which ensures better data privacy and reduces dependence on internet connectivity. This approach allows developers to work in secure environments without exposing source code to external servers. In addition, the seamless integration of the extension within Visual Studio Code helps maintain a smooth workflow, minimizing the need for developers to switch between multiple tools during the coding process.Overall, the CodeGen system highlights how AI-based development assistants can contribute to more intelligent and efficient programming environments. By combining automation, contextual understanding, and offline AI capabilities, the project establishes a strong foundation for future research and development in privacy-focused AI programming tools.

Future Scope

The future development of the CodeGen system offers several opportunities for improvement and expansion as artificial intelligence technologies continue to evolve. One potential enhancement involves integrating more advanced AI models that provide higher accuracy, better contextual understanding, and support for a wider range of programming languages. This improvement would allow the system to handle more complex development tasks and provide more precise code suggestions.

Another important direction for future development is the enhancement of real-time development assistance within Visual Studio Code. Features such as continuous code suggestions, intelligent autocomplete mechanisms, and advanced debugging capabilities could be implemented to provide developers with more proactive support during the coding process.

Further improvements could include the addition of voice-based programming interfaces, allowing developers to interact with the system through spoken commands. Expanding the system to support multiple programming languages and collaborative development environments would also increase its usability for team-based software projects. Additionally, integrating optional cloud-based services for model updates, storage, and deployment could improve system scalability and accessibility while maintaining the flexibility of offline functionality.

By incorporating these advancements, future versions of CodeGen have the potential to evolve into a comprehensive AI-driven development platform that assists programmers throughout the entire software development lifecycle, from code generation and debugging to testing and deployment.

References

[1] M. J. Price, *Programming Visual Studio Code Extensions*. Birmingham, UK: Packt Publishing, 2022.
[2] B. Johnson, *Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers*. New York, NY, USA: Apress, 2024.
[3] L. Moroney, *AI and Machine Learning for Coders: A Programmer's Guide to Artificial Intelligence*. Sebastopol, CA, USA: O'Reilly Media, 2021.
[4] K. Zhang and Z. Lin, "Offline Machine Learning: Deploying Models Locally for Privacy and Speed," IEEE Computer Society, 2023.
[5] A. Raj and M. Patel, *Building AI-Powered Developer Tools: Integrating Machine Learning into IDEs*. Cham, Switzerland: Springer, 2022.
[6] R. Verma, *Visual Studio Extensibility Development: Extending Visual Studio IDE for Productivity, Quality, Tooling, Analysis, and Artificial Intelligence*. Birmingham, UK: Packt Publishing, 2024.
[7] A. Svyatkovskiy, Y. Zhao, S. Fu, and N. Sundaresan, "Pythia: AI-Assisted Code Completion System," in *Proceedings of the ACM Symposium on Programming Language Design and Implementation*, 2019.
[8] B. Donato, L. Mariani, D. Micucci, O. Riganelli, and M. Somaschini, "MultiMind: A Plug-in for AI-Assisted Development Tasks in Integrated Development Environments," 2025.
[9] E. Chen, R. Huang, J. Liang, D. Chen, and P. Hung, "GPTutor: An Open-Source AI Pair Programming Tool for Software Development Assistance," 2023.