iJRASET
International Journal For Research in Applied Science and Engineering Technology



# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 14 **Issue:** XI **Month of publication:** November 2026

**DOI:** https://doi.org/10.22214/ijraset.2026.85009

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 XI Nov 2026- Available at www.ijraset.com*

# AI-Powered Content Summarization System for Coding Platforms

K. Bala Deekshikesh[1], K. Ravi Kumar[2], P. Varun Goud[3], N. Mysura Reddy[4], O. Tharun[5]
*Department of Computer Science and Engineering (Data Science), Hyderabad Institute of Technology and Management (HITAM), India*

***Abstract:*** ***Coding platforms are important environments for programming practice and technical interview preparation, but learners can face difficulty understanding code, identifying errors, and progressing through problems without depending on complete solutions. This paper presents an AI-powered browser extension designed to provide learning-oriented coding assistance directly within supported coding platforms. The proposed system integrates four assistance modes: Explain Code, Give Hint, Debug Code, and Ask AI. The extension extracts the coding problem statement, user-submitted code, programming language, and other relevant context, and combines this information with the learner's request before sending it to a selected AI model. The project report considers ChatGPT, Ollama, and Grok as possible AI-model integrations. The system is intended to reduce context switching, provide understandable explanations, offer progressive guidance, and support interactive programming learning. Evaluation is structured around response accuracy, relevance, response time, usability, and explanation quality. The proposed approach emphasizes assistance that supports independent problem solving rather than simply presenting complete solutions.***

***Keywords:*** ***artificial intelligence, browser extension, code explanation, programming education, large language models, automated hints, debugging assistance.***

## I. INTRODUCTION

Coding platforms are widely used for programming practice, skill development, and technical interview preparation. However, learners—particularly beginners—may struggle to understand complex program logic, identify programming errors, or determine how to proceed when they become stuck.

The project report identifies a further usability issue: obtaining explanations from external websites or separate AI tools interrupts the coding workflow and requires repeated context switching. An integrated AI assistant can address this problem by presenting programming guidance inside the coding environment itself.

Research on large language models (LLMs) has demonstrated their potential for generating explanations of programming code, while also showing that explanation quality can vary with model selection and prompting strategy [1], [2]. Automated hints have also been studied as a way of guiding learners without immediately revealing complete solutions [4]. In programming education, AI-driven coding assistance has been associated with opportunities for explanation, error clarification, and personalized support, although concerns about accuracy and over-dependence remain [3], [5].

Based on these observations, the proposed project develops a browser extension that combines multiple learning-oriented assistance functions in a single interface. Instead of treating explanation, hints, debugging, and conversational assistance as separate tools, the system provides them as coordinated interaction modes within the coding environment. The central objective is to make coding practice more understandable and interactive while encouraging learners to reason about problems themselves.

## II. RELATED WORK

### *A. LLM-Based Code Explanation*

The project report cites work showing that LLMs can generate useful explanations of programming code. Lekshmi-Narayanan et al. investigated explanations of code examples in introductory programming courses and examined the relationship between LLM-generated and human explanations [1]. Oli et al. further examined how prompting and model configuration affect the behaviour of LLMs when generating code explanations [2]. These studies motivate the use of LLMs as an explanation layer for learners, while also indicating that the quality of generated explanations depends on how the request is formulated.

©IJRASET: All Rights are Reserved 

### *B. Automated Programming Hints*

Automated hints are another important component of learning-oriented programming support. Bui and Syed Mustapha investigated data-driven hint generation in intelligent tutoring systems for code-writing and highlighted the role of hints in guiding learners when they are stuck [4]. A hint-based interaction differs from simply displaying a completed answer because it can provide a progressive path toward a solution. This principle is reflected in the proposed Give Hint function.

### *C. AI Assistance in Programming Education*

Cambaz and Zhang reviewed AI-driven code-generation models in programming education and described their potential for explanation, error clarification, and learning assistance [3]. Kasneci et al. discussed broader opportunities and challenges associated with LLMs in education, including the need to consider reliability and over-reliance [5]. These findings support an educational design in which AI is used as an assistant for understanding and problem solving rather than as a replacement for the learner's reasoning.

### *D. Research Gap and Novelty*

The project report identifies a gap in the integration of these capabilities. Existing work has explored code explanation, automated hints, AI-assisted programming education, and LLM-based coding support as related but often separate functions. The proposed system combines Explain Code, Give Hint, Debug Code, and Ask AI in one browser-extension environment intended to operate directly within coding platforms. This integrated design is the primary novelty described by the project.

## III. SYSTEM DESIGN AND METHODOLOGY

### *A. System Overview*

The proposed system is a browser extension that operates as an AI assistance layer over a supported coding platform. Its workflow begins by obtaining the relevant coding context, continues with interpretation of the learner's request, and ends with presentation of the generated assistance inside the extension. The methodology described in the project report can be organized into four major stages: context extraction, context preparation, request processing, and AI response generation.

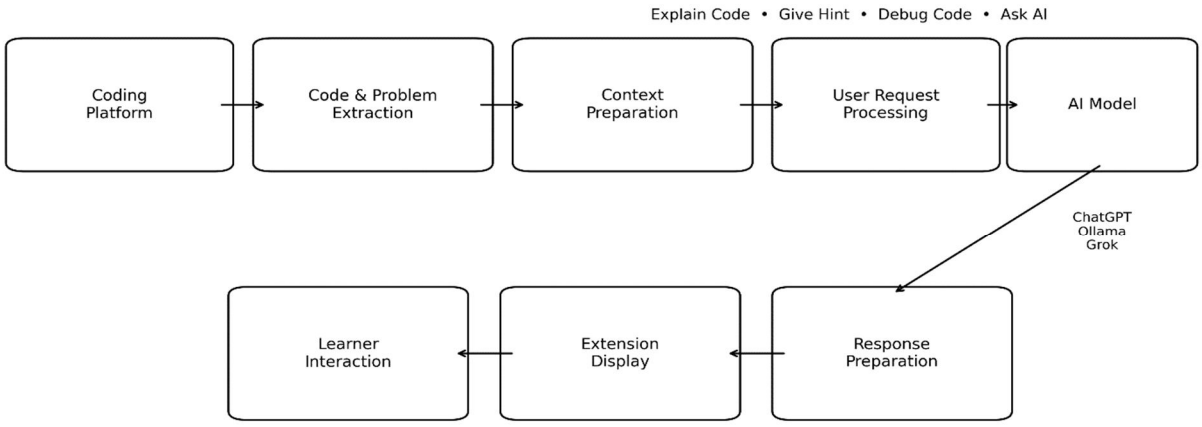



Fig. 1 Proposed workflow of the AI-powered coding assistance extension.

### *B. Context Extraction and Preparation*

The first stage extracts information from the supported coding platform. The extension is designed to detect the problem statement, the user's submitted code, the programming language, and other relevant information available in the coding environment. The extracted information is then prepared as contextual input for the selected AI model. Context preparation is important because the usefulness of an AI response depends on the model receiving sufficient information about both the problem and the learner's code.

### C. User Request Processing

The extension provides four assistance options: Explain Code, Give Hint, Debug Code, and Ask AI. When the learner selects an option or submits a question, the system combines the request with the relevant problem and code context. The request-processing stage therefore connects the learner's immediate need with the information extracted from the coding platform.

### D. AI Model Integration

The report identifies ChatGPT, Ollama, and Grok as AI models that can be integrated into the proposed system. The problem statement, submitted code, programming language, and user query are sent to the selected model. The generated response is then prepared for display through the extension. This architecture allows the project to explore different model configurations while maintaining a common user-facing assistance workflow.

## IV. FUNCTIONAL ASSISTANCE FEATURES

### A. Explain Code

Explain Code is intended to translate programming logic into simple, understandable language. The function can describe the overall approach, important statements, algorithms, and the relationship between different parts of the submitted program. Its purpose is to help a learner understand existing code rather than merely obtain another version of the code.

### B. Give Hint

Give Hint provides progressive guidance for a coding problem. Instead of directly revealing a complete solution, the feature is designed to provide a learner with a direction for the next step. This design follows the learning-oriented principle of supporting independent problem solving and is consistent with the automated-hint motivation described in the project report [4].

### C. Debug Code

Debug Code focuses on identifying possible errors in the learner's submission, explaining the likely causes of those errors, and suggesting improvements. The feature is intended to support understanding of mistakes, rather than only presenting corrected code. This can help learners connect an observed error with the underlying programming concept.

### D. Ask AI

Ask AI provides an interactive conversational interface for programming-related questions. A learner can ask about code, concepts, errors, or problem-solving approaches, and the system uses the available coding context to generate a response. Displaying the interaction within the extension reduces the need to leave the coding environment and search for separate explanations.

TABLE I
FUNCTIONAL MODULES AND THEIR INTENDED ASSISTANCE

| Module | Primary Input | Intended Assistance |
|---|---|---|
| Explain Code | Problem, code, language | Simple explanation of logic, statements, algorithms, and approach |
| Give Hint | Problem and current code | Progressive guidance without immediately exposing the complete solution |
| Debug Code | Problem and submitted code | Possible errors, causes, and suggested improvements |
| Ask AI | Problem, code, language, user query | Interactive programming support and conceptual clarification |

## V. EVALUATION FRAMEWORK

### A. Evaluation Criteria

The project report proposes evaluating the system using response accuracy, relevance, response time, usability, and the quality of code explanations. These criteria cover both technical behavior and the learner's experience.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 14 Issue XI Nov 2026- Available at www.ijraset.com*

Accuracy concerns whether the generated assistance correctly addresses the programming context. Relevance concerns whether the response matches the learner's request. Response time captures the responsiveness of the extension, while usability considers how easily learners can interact with the assistance features. Explanation quality concerns whether the response is understandable and useful for learning.

TABLE II
EVALUATION CRITERIA FOR THE PROPOSED SYSTEM

| **Criterion** | **What It Examines** | **Purpose** |
|---|---|---|
| Response Accuracy | Correctness of generated guidance | Assess whether the assistance is technically useful |
| Relevance | Alignment with problem and user request | Assess contextual appropriateness |
| Response Time | Time required to generate and display assistance | Assess interaction responsiveness |
| Usability | Ease of using the extension features | Assess learner interaction and workflow |
| Explanation Quality | Clarity and comprehensibility of explanations | Assess learning value |

### *B. Learning-Oriented Evaluation*

In addition to technical response quality, the system is designed around an educational objective: helping learners understand programming problems and their own code. Explain Code and Debug Code emphasize understanding, while Give Hint is explicitly intended to encourage the learner to continue solving the problem. Ask AI extends this interaction by allowing follow-up questions within the same environment. Together, these functions provide a framework for evaluating whether AI assistance can be integrated without turning the coding workflow into a simple solution-retrieval process.

### *C. Current Evidence and Scope of Evaluation*

The supplied project report defines the evaluation criteria but does not provide numerical benchmark results, user-study results, or comparative measurements for the integrated extension. Therefore, this paper presents the evaluation framework and expected outcomes without inventing accuracy, latency, or usability values. A complete publication study would require controlled test cases, representative coding problems, user evaluation, and measured results for the selected AI models.

## VI. SDG ALIGNMENT AND EXPECTED OUTCOMES

### *A. Alignment with SDG 4*

The project report aligns the proposed system with Sustainable Development Goal 4 (Quality Education). The alignment is based on making programming concepts easier to understand, providing personalized AI-based assistance, encouraging independent problem solving, and offering immediate guidance and feedback. By embedding these capabilities in the coding environment, the project aims to make programming practice more interactive and accessible.

©IJRASET: All Rights are Reserved 

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 XI Nov 2026- Available at www.ijraset.com*

*B. Architecture*

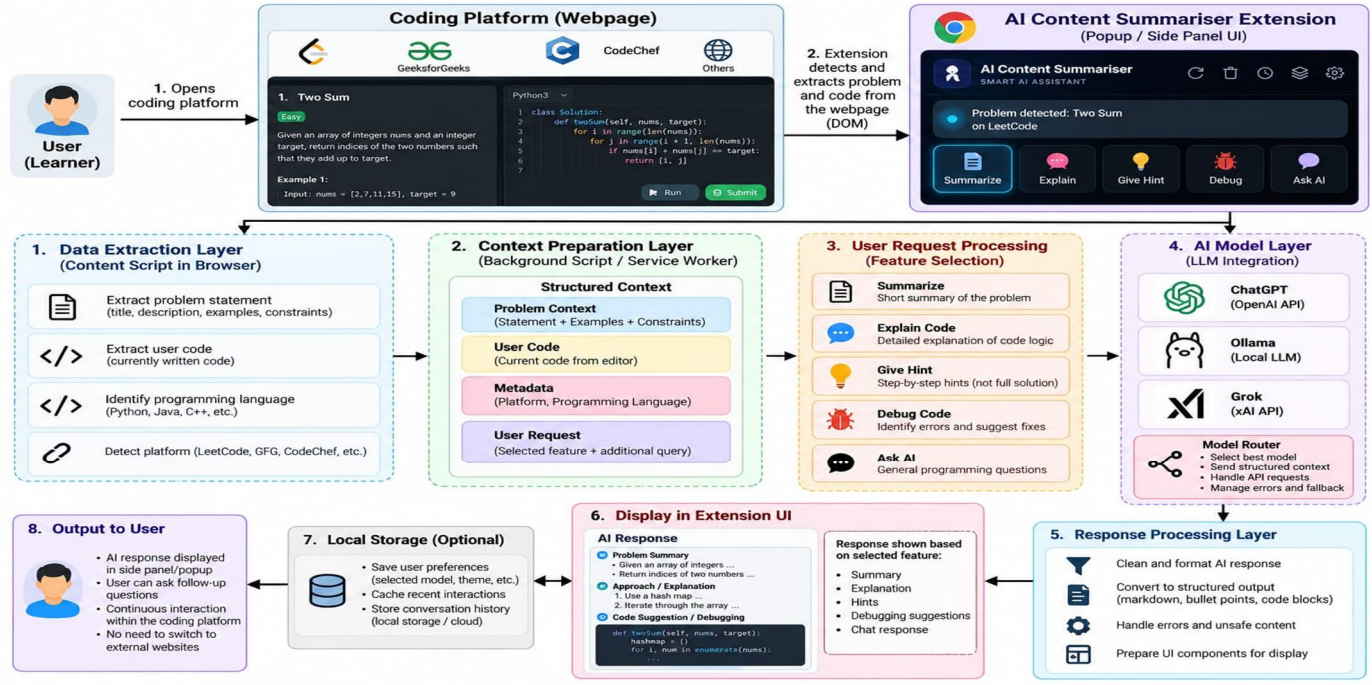



Fig. 2 Architecture of the Proposed AI-Powered Coding Assistance Extension.

*C. Result*

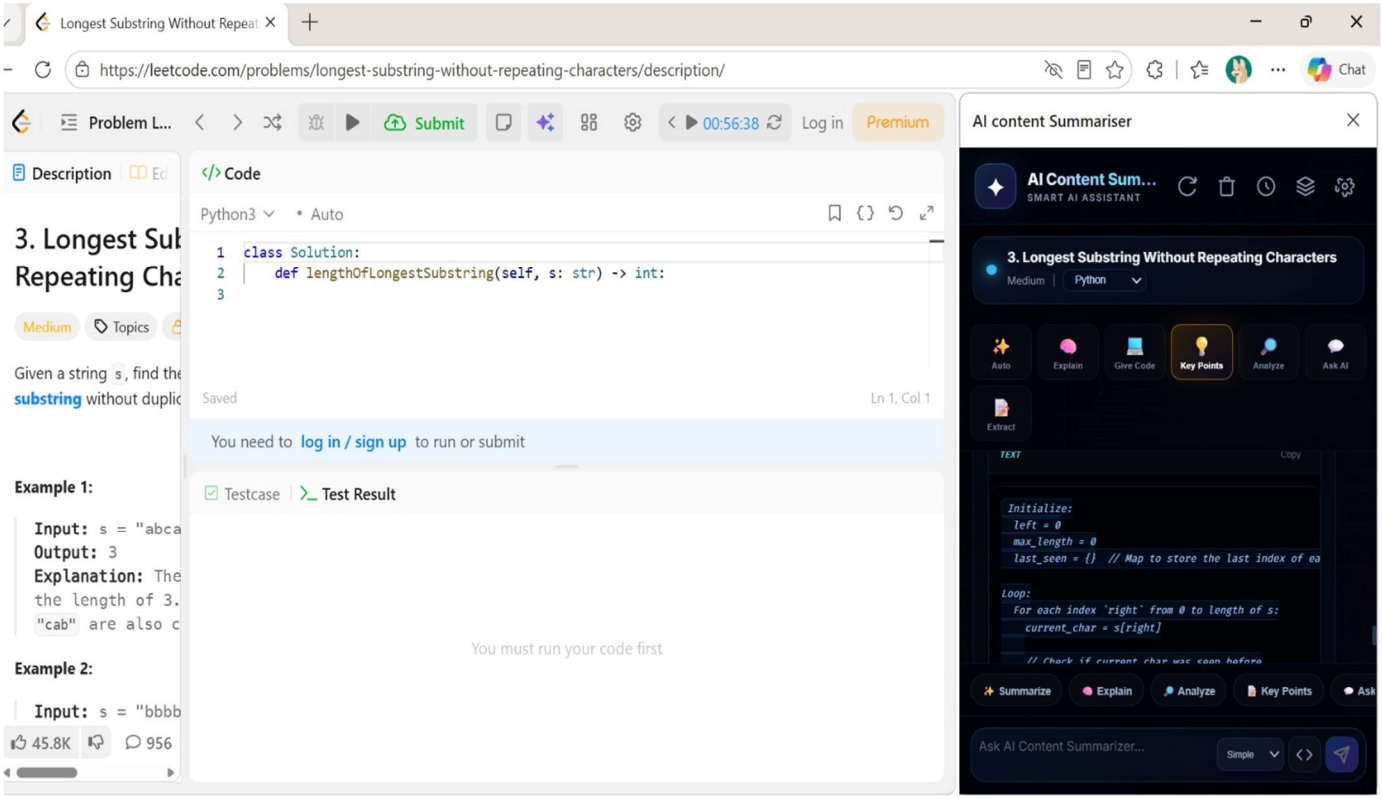



Fig. 3 Result of the AI-Powered Content Summarization Extension on a Coding Platform.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

## VII. FUTURE SCOPE

The project can be extended to support additional coding platforms and programming languages. Future versions can also incorporate time and space complexity analysis, code-optimization suggestions, alternative solution approaches, and more personalized assistance. The report additionally identifies the exploration of further AI models and locally hosted models as a possible direction. These extensions could increase the flexibility of the system and broaden its applicability across programming-learning environments.

## VIII. CONCLUSION

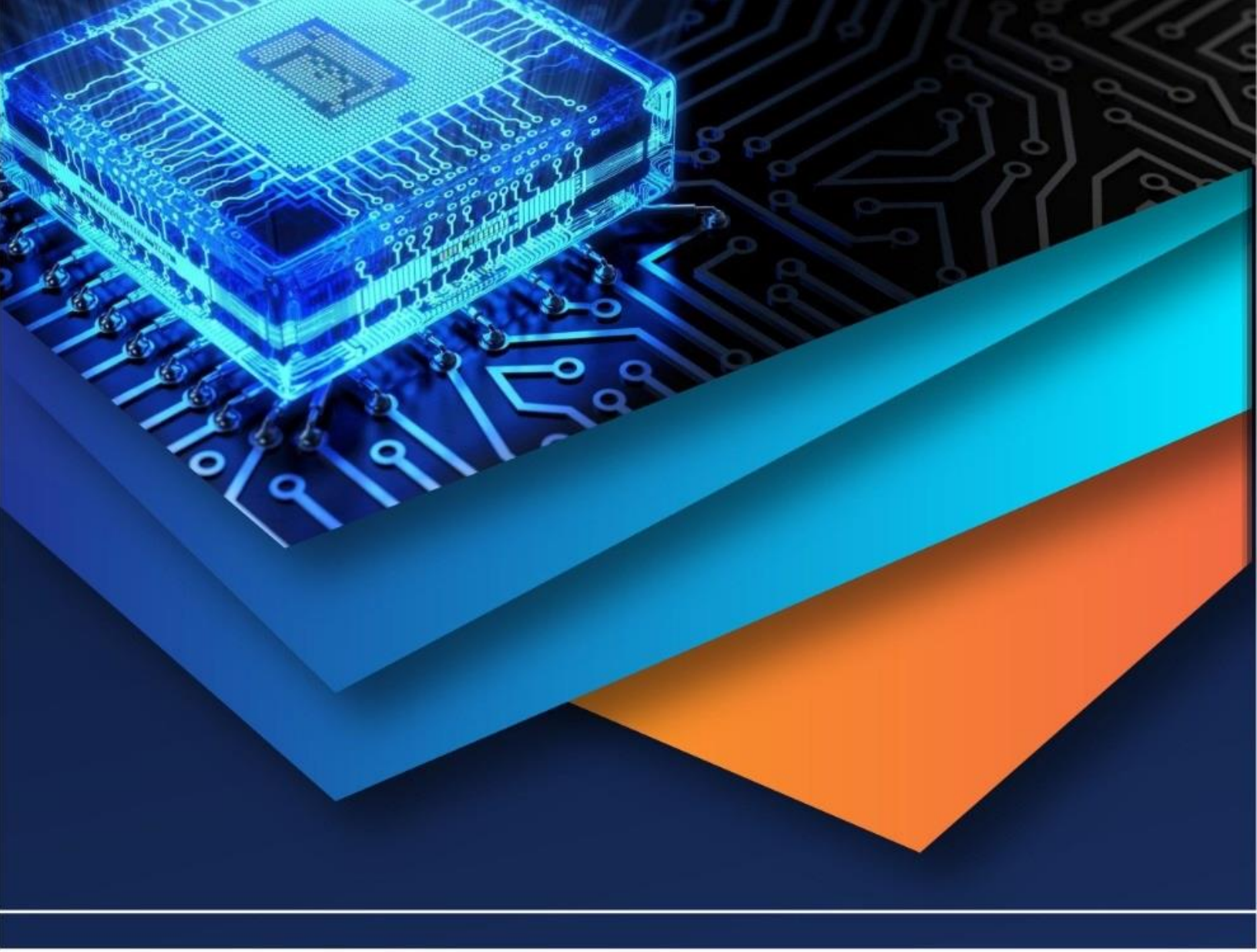
This paper presents an AI-powered browser-extension approach for providing integrated coding assistance within coding platforms. The proposed system combines Explain Code, Give Hint, Debug Code, and Ask AI so that learners can obtain explanations, progressive guidance, debugging support, and conversational assistance without repeatedly switching to external tools. The methodology extracts the coding problem, submitted code, programming language, and relevant context before processing the learner's request through a selected AI model. The design is motivated by prior work on LLM-based code explanations, automated hints, and AI-supported programming education [1]–[5]. The project emphasizes understandable assistance and independent problem solving rather than simply providing complete solutions. The current report establishes the system concept, methodology, evaluation criteria, and expected outcomes; quantitative validation remains an important requirement for a full empirical publication.

## IX. ACKNOWLEDGMENT

The authors acknowledge their academic institution and project mentors for providing the environment and guidance required to develop and document the proposed system.

## REFERENCES

[1] A. B. Lekshmi-Narayanan et al., “Explaining Code Examples in Introductory Programming Courses: LLM vs Humans,” Proceedings of Machine Learning Research, vol. 257, pp. 107–117, 2024.
[2] P. Oli, R. Banjade, J. Chapagain, and V. Rus, “The Behavior of Large Language Models When Prompted to Generate Code Explanations,” NeurIPS 2023 Workshop on Generative AI for Education, 2023.
[3] D. Cambaz and X. Zhang, “Use of AI-driven Code Generation Models in Teaching and Learning Programming: A Systematic Literature Review,” Proc. 55th ACM Technical Symposium on Computer Science Education (SIGCSE 2024), pp. 172–178, 2024.
[4] H. T. Bui and S. M. F. D. Syed Mustapha, “Automated Data-Driven Hint Generation in Intelligent Tutoring Systems for Code-Writing,” International Journal of Emerging Technologies in Learning (iJET), 2018.
[5] E. Kasneci et al., “ChatGPT for Good? On Opportunities and Challenges of Large Language Models for Education,” Learning and Individual Differences, vol. 103, 102274, 2023.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)