ISSN: 2582 - 6379
IJISEA Publications
International Journal for Interdisciplinary Sciences and Engineering Applications
IJISEA - An International Peer- Reviewed Journal
2025, Volume 6 Issue 2
www.ijisea.org

CODEMATE
A Coding Chatbot using Generative AI

P.Chiranjeevi[1], T.Poojitha[2] ,R.N.HemanthKumar[3],AB.Sameer[4],B.Sai Dharma Teja[5]
Professor[1] , UG Students[2,3,4,5]
Computer Science Department
Amrita Sai Institute of Science & Technology
Paritala , Andhra Pradesh , India

thondepupoojitha@gmail.com

ABSTRACT

CodeMate is an AI-powered coding assistant intended to enhance the developer experience by the pursuit of pretrained Simple language model (SLMs), such as Microsoft's Phi 3. It offers a competent user interface, reminiscent of popular AI conversational interfaces like ChatGPT, where handlers can engage in natural language colloquies to seek coding assistance, debug issues, and gain sagacity on programming concepts.CodeMate's prime emphasis is to boost the coding procedure, consenting users to ask complex questions, obtain code endorsements, and rally inclusive thruput. By amalgamating the SLM's indulgence of innumerable programming languages and development environments, CodeMate can aid numerous developers, oscillating from beginners to professionals. The project aspires to bridge the gap between human creators and machine learning models, accomplishing coding more comprehensible and resourceful. By incorporating AI-driven insights, CodeMate not only expediates competent coding practices but also nurtures a unfathomable supportive of programming moralities, eventually vesting users to befall more skillful and assured in their coding adeptness.

Key words: AI-driven insights, AI-powered coding assistant,CodeMate, Coding assistance,Human-AI collaboration,Natural language interface,Simple Language Models (SLMs)

Abbreviations:AI, Artificial Intelligence; IDE, Integrated Development Environment; NLP, Natural Language Processing; SLM, Simple Language Model;UI, User Interface;

I.INTRODUCTION

CODEMATE – REENVISIONING DEVELOPER PROFICIENCY THROUGH AI

In today's rapidly evolving technological landscape, coding has become a core competency across multiple domains—from software engineering to data science. CodeMate leverages advancements in Natural Language Processing (NLP) and Small Language Models (SLMs), such as Microsoft's Phi-3, to transform the way developers receive support. Acting as a virtual coding assistant, CodeMate allows users to pose natural language queries, debug code, and receive real-time, context-aware suggestions across multiple programming languages including Python, Java, and JavaScript. Designed for developers of all experience levels, CodeMate accelerates the learning curve, boosts productivity, and simplifies complex programming concepts.

IMPACT AND CONTRIBUTION TO THE DEVELOPER COMMUNITY

CodeMate is more than just a coding assistant—it's a collaborative learning companion. By simplifying complex coding challenges and reducing the time spent on debugging, it enhances productivity and fosters continuous learning. Its ability to provide immediate, relevant, and reliable coding assistance addresses a critical gap in modern software development. Ultimately, CodeMate empowers developers to write cleaner, more efficient code while fostering a more inclusive and dynamic programming environment.

II.RELATED WORK

"Evaluating Large Language Models Trained on Code," arXiv preprint, A. Chen, P. Jain, and K. Lee [1]

This paper sightsees the depiction of large language models (LLMs) distinctively trained on programming languages. The authors scrutinize how transformer-based models such as OpenAI Codex can abet with engendering code snippets, completing functions, and proffering documentation sustenance. Their verdictsvalidate how these models can suggestivelypep talk developer productivity when cohesive into tools like GitHub Copilot. This study propounds indispensable acumens for the expansion of intelligent coding assistants like CodeMate, collateralizing the real-world pertinency of LLMs in software engineering workflows.

"Code Optimization with Phi-2: Small Language Models with Big Potential," arXiv preprint, T. Nijkamp, B. Pang, and C. Zhang [2]

This research bestows Phi-2, a Small Language Model (SLM) advanced by Microsoft, intended for efficient coding and cognitive tasks. Despite its smaller size, Phi-2 dispenses strong performance athwart a variety of programming challenges. The study emphasises the potential of compact models to endorse coding tasks effectually without the heavy resource stipulates of larger LLMs. This allies meticulously with CodeMate's architecture, which clouts SLMs to dispense lightweight, responsive, and intelligent coding assistance on sundry platforms.

"Expectations, Outcomes, and Challenges of Modern Code Completion Tools," ACM SIGSOFT ISSTA, B. Vaithilingam, P. Francis, and S. Pradel [3]

This customer-focused scrutinize inspects how developers intermingle with contemporary code completion tools, embracing their potentials, apprehensions about reliability, and trust in AI-generated innuendos. The authors equate vital challenges such as be delirious outputs and erratic code quality. Their verdicts are predominantlygermane to CodeMate, which targets to bid not just smart accomplishments, but trustworthy and attribute-focusedsustenance, exclusivelyimperative for facilitating beginners contrive poise and write healthier code.

III.METHODOLOGY

Design and Architecture

The CodeMate techniqueenlists a modular, client-server architecture to assureflexibility, scalability, and functional reliability. The client side is erected with HTML, CSS, and JavaScript to expediateproblem solving time, handlerreciprocities and a responsive chat interface. The backend is wroughtby Flask, which copessessions, authentication, and communique with AI models and anorganized knowledge base.

Core components include:

- A user authentication module exploiting Flask sessions to guarantee secure entree.
- A chat interface that cloutsFlask templates and JavaScript for flawlessuser interaction.
- An AI response manager cohesive with the models such as Microsoft Phi 3 via Hugging Face transformers to handleinputs and yieldconnotativelypertinentretorts.

The data flow trails an organized path: user messages are received by the backend, managed through either the knowledge base or the AI model, and retorts are delivered back to the frontend. Security is obligatory through HTTPS and session management, while caching and load balancing sustenance performance and scalability.

Tools and Technologies

Development of CodeMate amalgamatesmultiple technologies:

- Python and Flask form the backend central.
- HTML, CSS, and JavaScript build the user interface.
- AI competences are powered by models such as Microsoft Phi 3 and Hugging Face Transformers.
- JSON is used for lightweight knowledge base management, while SQLite or PostgreSQL handle controls data storage.
- Through Hugging Face and models like Microsoft aidadvanced NLP tasks.
- Testing is buoyed by tools like pytest, and version control is accomplished using Git.
- Security is enhanced through Flask-Login.

Algorithmic Approach

The chatbot’s serviceableness relies on foundational NLP and machine learning techniques. Tokenization, intent recognition, and entity extraction enable interpretation of user inputs. Language models like Microsoft Phi 3 generate dynamic responses, supplemented by a retrieval-based knowledge base. Context management ensures coherent multi-turn conversations. Responses are either generated via AI or selected from predefined templates for common queries.

Key steps:

- Load the knowledge base.
- Initialize AI client.
- Receive and parse user input.
- Retrieve or generate a relevant response.
- Return response to the frontend.

This approach allows the chatbot to dynamically switch between retrieval and generative response strategies based on query context.

Data Collection and Pre-processing

Data for CodeMate is sourced from user interactions, curated documentation, and third-party platforms such as GitHub or Stack Overflow, Scopus. Pre-processing involves removing duplicates, standardizing formats, handling missing values, and tokenizing inputs for downstream tasks and is finetuned. The processed data is securely stored and versioned for continuous model updates and performance monitoring.

Data Analysis and Visualization

To continually refine CodeMate's effectiveness, data is analysed across several dimensions:

- Logs of user interactions and feedback are evaluated to assess usage trends and identify pain points.
- Metrics such as latency, uptime, and feature usage help gauge system performance.
- Visual analytics including dashboards, heatmaps, and time-series plots are created using tools like D3.js.

This framework ensures that CodeMate adapts over time to user behaviour and system performance needs. And this methodology ensures that CodeMate remains lightweight, scalable, and capable of delivering intelligent coding assistance across diverse developer needs and environments.

IV.RESULTS

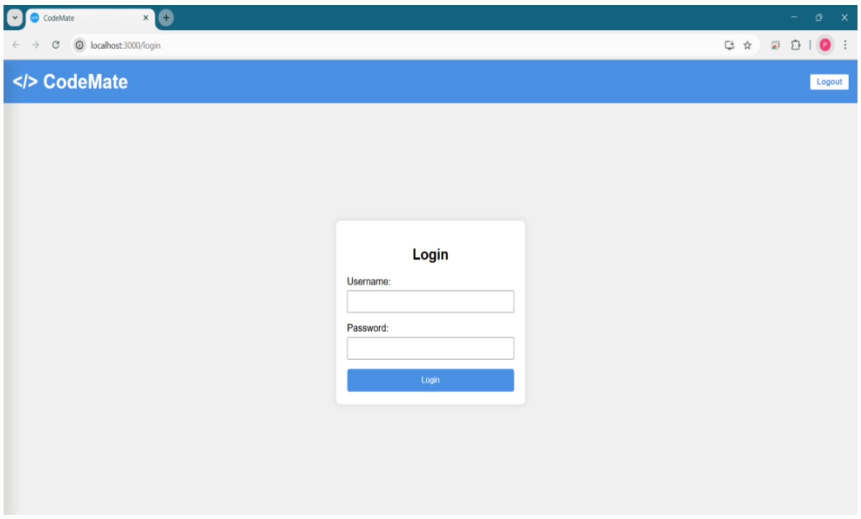
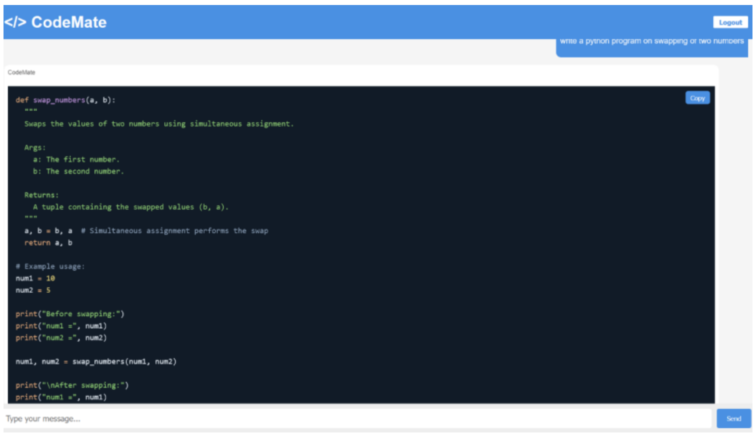


Figure 1: shows login page

After successfully logging in using chat interface users can request queries like "writing python program on swapping of two numbers" etc CodeMate results as below.



```python
def swap_numbers(a, b):
  """
  Swaps the values of two numbers using simultaneous assignment.

  Args:
    a: The first number.
    b: The second number.

  Returns:
    A tuple containing the swapped values (b, a).
  """
  a, b = b, a  # Simultaneous assignment performs the swap
  return a, b

# Example usage:
num1 = 10
num2 = 5

print("Before swapping:")
print("num1 =", num1)
print("num2 =", num2)

num1, num2 = swap_numbers(num1, num2)

print("\nAfter swapping:")
print("num1 =", num1)
```

Figure 2: Shows An example of working of CodeMate

In this way users can request queries not only in one programming language but many and even explains the code efficiently.

V.CONCLUSIONS

The CodeMate project highpoints the upwardinfluence of AI in software development by amalgamatinga Flask-based backend, a React frontend, andmodels like Microsoft's Phi NLP. This architecture permits intelligent landscapes such as real-time code analysis, debugging, and optimization, refining both productivity and code quality. Enthused by industry drifts and tools like GitHub Copilot and Tabnine, CodeMate espousalsdevelopers in restructuring workflows while preserving humansuperintendence. Although AI tools may diminish routine tasks, they accentuate the advancingrole of developersnecessitating adaptability, creativity, and critical thinking. CodeMate thus characterizes a step onward in building AI-assisted, developer-centric environments that substitutecompetent and pioneering coding practices.

VI.DISCUSSIONS

The CodeMate project effectuallyestablishes the incorporation of AI into software development using a Flask-React architecture and Microsoft's Phi etc; NLP models. Itvitrines how AI can augment coding thruput through real-time support, debugging, and code optimization. The modular design certifies scalability, while Hugging Face transformersexpediatedefficient language model collaboration.

Crucialtrialsencompassedhandling user context, certifying response accuracy, and fortifying user data. Despite these, CodeMate efficaciouslytransported context-aware support, shimmering trends seen in tools like GitHub Copilot. The project also highpoints the poise between automation and the growing role of developers, accentuation AI as a sustenance tool rather than aauxiliary.

Overall, CodeMate subsidizes to the mounting shift toward intelligent development environments and sets a groundwork for future augmentations.

ACKNOWLEDGEMENT(S)

It gives me immense pleasure to thank my guide Dr. P. Chiranjeevi ph. D, for having been a source of constant inspiration, precious guidance, and generous assistance during the project work. I deem it as a privilege to have worked under his able guidance. Without his close monitoring and valuable suggestions, this work would not have taken this shape. I feel that his help is unforgettable.

I take the privilege to record my thanks to H.O.D of CSE Dr. P. CHIRANJEEVI Ph.D for his encouragement, cooperation, and valuable suggestions to complete project in time.

I am very thankful to honorable Principal Dr M. SASIDHAR Ph. D and I express my sincere gratitude and deep sense of respect for making me available with all the required assistance and for his support and inspiration to carry out this project in the Institute.

I am thankful to the teaching and non-teaching staff of CSE department for their direct as well as indirect help in my project.

REFERENCES:

[1] A. Chen, P. Jain, and K. Lee, “Evaluating Large Language Models Trained on Code,” arXiv preprint arXiv:2107.03374, 2021.

[2] T. Nijkamp, B. Pang, C. Zhang, et al., “Code Optimization with Phi-2: Small Language Models with Big Potential,” arXiv preprint arXiv:2310.01333, 2023.

[3] B. Vaithilingam, P. Francis, and S. Pradel, “Expectations, Outcomes, and Challenges of Modern Code Completion Tools,” in Proc. 27th ACM SIGSOFT Int. Symp. Softw. Test. Anal. (ISSTA), 2022, pp. 42–52.

[4] A. Svyatkovskiy, S. Sundaresan, Y. Zhao, and N. Fu, “Intellicode Compose: Code Generation Using Transformer,” in Proc. 28th ACM Joint Meeting on European Software Engineering Conf. and Symp. Found. Softw. Eng., 2020, pp. 1433–1443.

[5] J. Kocielnik, P. M. Doubleday, M. K. Lee, and S. M. H. H. Teevan, “Guidelines for Human-AI Interaction,” in Proc. 2019 CHI Conf. Human Factors in Computing Systems (CHI), 2019, Paper No. 3.

[6] A. Vaswani, N. Shazeer, N. Parmar, et al., “Attention is All You Need,” in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2017.

[7] T. Wolf, L. Debut, V. Sanh, et al., “Transformers: State-of-the-Art Natural Language Processing,” inProc. 2020 EMNLP: System Demonstrations, 2020, pp. 38–45.