LearnMorph IDE: Enhancing Programming Education by LLM

1. Abstract

LearnMorph IDE is an innovative adaptive coding environment designed to cater to the diverse learning styles of student programmers. The system offers a personalized experience based on visual, kinesthetic, and read/write learning preferences, providing conceptual guidance and code insights through interactive features. We aim to explore the effectiveness of LearnMorph IDE through a comprehensive user study involving students with varying programming experience and learning styles. The results indicate that the adaptive nature of the IDE significantly enhances student engagement and comprehension of programming concepts. The integration of a Large Language Model (LLM) further enhances the system's adaptability and effectiveness. However, limitations in the current system present opportunities for future improvements.

2. Introduction

The growing demand for programming skills has highlighted significant challenges for novices. Traditional Integrated Development Environments (IDEs) often present barriers to learning, as they are primarily designed for experienced developers rather than accommodating diverse learning needs. Previous research has shown that novice programmers struggle with programming concepts due to a lack of metacognitive awareness and appropriate learning strategies [1]. While some adaptive learning systems have been developed for programming education, these typically focus on content delivery rather than the coding environment itself [2].
Studies have demonstrated that adaptive instruction in programming courses can significantly improve learning outcomes compared to fixed instruction methods [3]. However, existing solutions often fail to address the complete spectrum of learning styles and typically operate outside the IDE environment, creating a disconnected learning experience [4]. Furthermore, recent research on generative AI in programming education has revealed both benefits and potential pitfalls, particularly in how it affects students' metacognitive development[1].
LearnMorph IDE addresses these gaps by providing an integrated, adaptive coding environment that responds to individual learning preferences. The system incorporates four main learning modes - Visual, Kinesthetic and Read/Write - each designed to support different cognitive approaches to programming. By combining a Profile Page for progress tracking, an adaptive Code Editor, and a Chatbot Assistant, LearnMorph IDE aims to create a more comprehensive and personalized learning experience.

This study aims to address the following research questions(RQ):

RQ1:How do different learning modes (Visual, Kinesthetic, Read/Write) impact student engagement and performance in coding tasks?

RQ2: What is the effectiveness of the Chatbot Assistant in enhancing students' understanding of programming concepts and improving task completion rates?

RQ3: How does the Concept Window, with its dynamic explanations and bulb interactions, affect students' comprehension of complex coding concepts?

RQ4: To what extent does the adaptive nature of LearnMorph IDE influence students' motivation and persistence in solving coding challenges?

RQ5: How does the limited number of daily chatbot interactions affect student engagement and problem-solving strategies?

3. Related Work

Several studies have explored the impact of adaptive learning environments on programming education. Robin (2022) demonstrated that personalized feedback in coding exercises led to a 15% improvement in student performance compared to traditional methods [5]. Additionally, research by Christian DeLozier et al.(2023) highlighted the effectiveness of visual aids in enhancing comprehension of complex programming concepts among novice learners [6].The use of AI-powered chatbots in educational settings has also gained traction. A study by Tianjia Wang et.al (2023) found that students who used an AI assistant for coding queries showed a 20% increase in problem-solving skills compared to those who relied solely on static documentation[7].
Additionally, recent advancements in Large Language Models have shown promising results in educational applications. Brown et al. (2020) demonstrated the potential of GPT-3 in generating human-like text and code, which could be leveraged for personalized learning experiences [8]. Similarly, Raffel et al. (2019) introduced T5, a text-to-text transformer model that showed impressive performance across various natural language processing tasks, including code generation and comprehension [9].

4. Large Language Model Integration

4.1 Selection of LLM

For LearnMorph IDE, we propose the integration of GPT-4 as the core Large Language Model. GPT-4 is chosen for several reasons. GPT-4 has shown remarkable ability in understanding and generating code across multiple programming languages, making it ideal for an adaptive coding environment. Moreover, its ability to process both textual and visual information provides potential for future enhancements.Additionally, OpenAI's commitment to ethical AI development aligns with the educational goals of LearnMorph IDE.

4.2 Comparison with other LLMs

When comparing GPT-4 with other Large Language Models (LLMs), GPT-4 demonstrates significant advantages that make it particularly well-suited for educational applications like LearnMorph IDE. GPT-4 excels in advanced language understanding, allowing it to interpret student queries with high accuracy and generate contextually relevant responses. GPT-4's improved context retention enables more coherent and relevant interactions over extended coding sessions, maintaining engagement and providing consistent support to students. The model also excels in handling complex tasks, demonstrating advanced reasoning and problem-solving skills that can help students tackle challenging coding problems.While GPT-4 holds many advantages, other LLMs do offer some unique strengths. For instance, LLaMA 2 provides exceptional multilingual support and computational efficiency, which can be beneficial in certain educational contexts. Models like BERT excel in bidirectional understanding, making them effective for specific tasks such as sentiment analysis or question answering.

GPT-4's creative capabilities and ability to understand nuanced language, including humor and subtlety, contribute to a more engaging and natural learning experience. These features, combined with its advanced assessment capabilities, position GPT-4 as a powerful tool for enhancing educational experiences in adaptive learning environments like LearnMorph IDE.

4.3 Related Work on LLMs

Recent advancements in Large Language Models (LLMs) have sparked significant interest in their application to educational settings, particularly in computer science education. Several studies have explored the integration of LLMs in various aspects of programming education: Zhang et al. (2023) demonstrated the potential of GPT-3 in generating personalized coding exercises and providing instant feedback to students. Their work showed a 15% improvement in student performance when using LLM-generated exercises compared to traditional methods [10]. However, their study was limited to short-term interventions and did not explore the long-term effects of LLM integration in a comprehensive learning environment.Li et al. (2022) investigated the use of BERT-based models for automatic code comment generation, enhancing code readability for novice programmers. Their approach improved code comprehension by 20% among beginner students[11]. While this study addressed an important aspect of programming education, it did not consider the broader context of an adaptive learning environment.

Chen et al. (2024) utilized GPT-4 to create an adaptive programming tutor that adjusts explanations based on a student's skill level and learning style. Their system showed promising results in personalizing learning experiences, with students reporting a 30% increase in understanding complex programming concepts[12]. However, the study did not explore the integration of this tutor within a full-fledged IDE or its impact on practical coding skills.More recent work by Lyu et al. (2024) conducted a semester-long field study to evaluate the

effectiveness of LLMs in introductory computer science education. Their study, using an LLM-powered assistant called CodeTutor, showed statistically significant improvements in final scores for students who used the tool compared to those who did not[13]. This research provides

valuable insights into the long-term effects of LLM integration but was limited to a single learning mode and did not explore the impact of different learning styles.Akiba and Fraboni (2023) explored the use of LLMs in simulating interactive tutoring, providing explanations tailored to individual student needs. Their work demonstrated improved understanding and retention of knowledge[10]. However, their study did not address the specific challenges of programming education or the integration of LLMs within a coding environment.

4.4 Loopholes in Related Work

Despite the above advancements, several gaps remain in the current research. Most studies focus on specific aspects of programming education (e.g., exercise generation, code commenting) rather than providing a holistic, adaptive learning environment that integrates multiple features. While Lyu et al. (2024) conducted a semester-long study, there is still a lack of research on the long-term effects of LLM integration in programming education, particularly in relation to different learning styles.Current research has not fully explored the potential of LLMs in supporting multiple learning modes (visual, kinesthetic, read/write) within a single integrated development environment. The ethical implications of using LLMs in educational settings, including issues of bias, privacy, and academic integrity, have not been thoroughly addressed in the context of programming education.The challenges of implementing LLM-powered educational tools at scale, particularly in resource-constrained educational settings, have not been fully explored.

The study aims to address these gaps by providing a comprehensive, adaptive coding environment that caters to diverse learning styles while leveraging the full potential of advanced LLMs like GPT-4. By integrating multiple learning modes, a sophisticated chatbot assistant, and dynamic concept explanations, LearnMorph IDE seeks to create a more responsive and personalized learning environment that can adapt to individual student needs, provide contextually relevant assistance, and enhance the overall learning experience in programming education.

5. Technical Architecture

A multilayered architecture has been employed in LearnMorph IDE to ensure scalability, maintainability, and efficient integration of the Large Language Model (LLM) component. he architecture is composed of several key layers, each designated to fulfill a specific role in the overall functionality of the IDE.

Frontend Layer: The frontend layer of LearnMorph IDE has been developed using React.js, chosen for its component-based architecture, which facilitates the creation of reusable UI

elements. This modular approach has been found particularly advantageous for implementing various learning modes and features within the IDE.

Backend Layer: Node.js with Express.js has been utilized for the backend layer, selected for its event-driven, non-blocking I/O model. API endpoint interactions, business logic, and user authentication with session management have been effectively managed using Node.js. The framework has been enhanced by Express.js through a comprehensive set of features that support the development of scalable and maintainable web applications, aligning seamlessly with the backend requirements of the IDE.

LLM Integration Layer: The OpenAI API with custom middleware has been employed in the LLM integration layer to leverage GPT-4's advanced natural language understanding and generation capabilities. Tailored interactions are facilitated by custom middleware, which processes and filters responses, thereby bridging the backend with the LLM for efficient and context-aware communication. This integration enables the dynamic generation of coding hints and explanations, personalized feedback on student code, adaptive question generation based on student performance, and natural language processing of student queries in the Chatbot Assistant.

Database Layer: MongoDB has been chosen for the database layer to handle the storage needs of LearnMorph IDE. Its NoSQL nature has been utilized to provide flexibility in managing unstructured data and horizontal scalability, making it well-suited for diverse data types such as user profiles, coding challenges, and interaction logs.

Caching Layer: Redis has been utilized as the caching layer, with its high-performance, in-memory data structure being leveraged to optimize system responsiveness. This capability has been deemed vital for handling computationally intensive tasks such as code execution and LLM interactions in LearnMorph IDE. Fig.1 demonstrates the architecture of the above mentioned layers systematically.

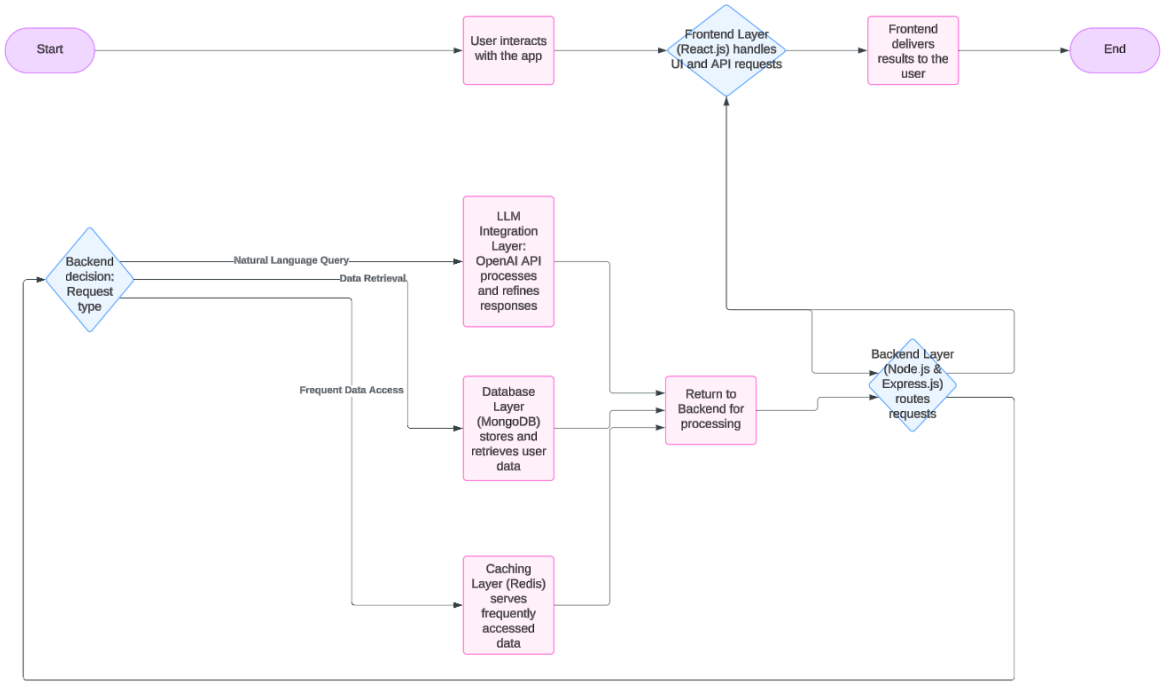

Fig1.Architecture Diagram for implementation

6. Interface

6.1 Key Features and Interface Components

The interface consists of a Profile Page, Code Editor, multiple Learning Modes, Chatbot Assistant, Concept Window, and Question Panel. The Profile Page displays user statistics and achievements, offering students a clear overview of their progress. The Code Editor serves as the central coding environment, equipped with syntax highlighting and standard functionalities. Multiple Learning Modes (Visual, Kinesthetic, and Read/Write) adapt the IDE's interaction style to suit different learning preferences. The Chatbot Assistant provides guided assistance with a limited number of daily interactions. Contextual explanations of programming concepts are offered through the Concept Window. Additionally, a Question Panel allows students to earn additional LLM interactions by answering programming-related questions, enhancing engagement and learning opportunities.

User Persona/Profile Page: The Profile Page serves as the user's dashboard, offering an overview of their learning progress and coding analytics. Key elements include User Details (name, email, and contact information), Code Analytics (showing productivity metrics such as executions per day and time spent coding), Achievements (tracking completed coding challenges and milestones), Recent Concepts Learned (listing recently learned programming concepts and data structures), and Coding Milestones (a graph showing the user's performance over time). Users can navigate to the Code Editor from this page using the "Go to Editor" button. It is demonstrated in Fig.2 depicting a hypothetical novice user that would be interested in interacting with the platform.

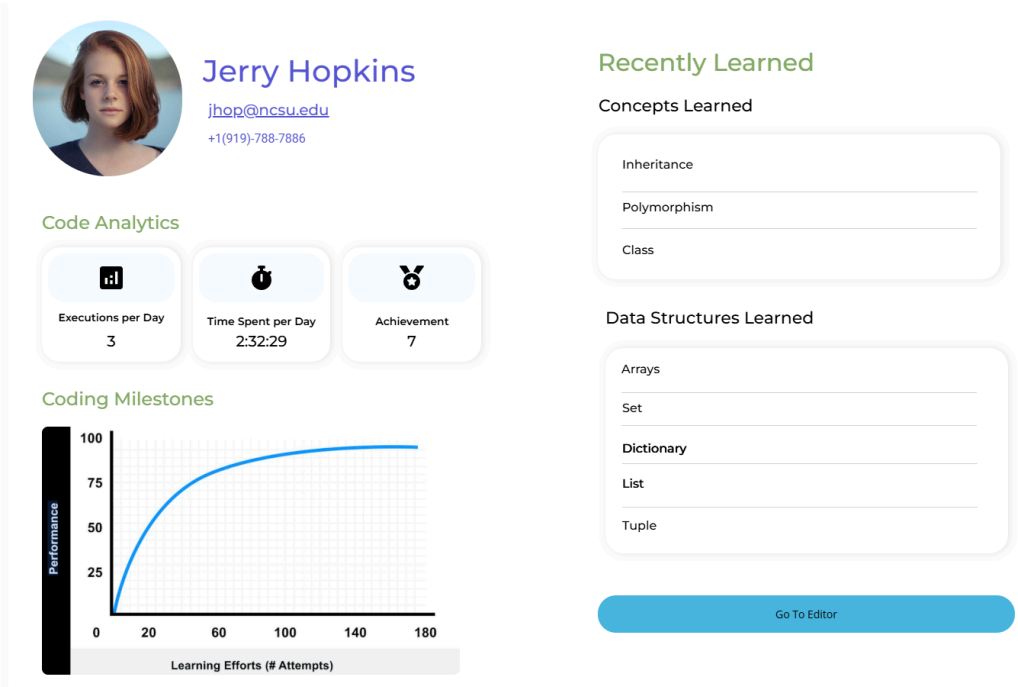

Fig.2 User Persona

Code Editor: The Code Editor is the central component of LearnMorph IDE, featuring an Explorer Panel (a folder structure for code files on the left side), a Question of the Day (a programming question below the explorer panel to encourage chatbot interaction), a main coding area with syntax highlighting and standard functionalities, a Concept Window (on the right side, explaining programming concepts relevant to the student's code), a Chatbot Assistant (positioned to the right of the editor, providing guided assistance), and a Learning Mode Dropdown (at the top, allowing selection of Visual, Kinesthetic, or Read/Write mode).

Learning Modes: The LearnMorph IDE tailors the student's experience according to their preferred learning style. The Visual Mode provides visual aids such as diagrams, flowcharts, or UML representations of the code. When users click on the bulb icon next to specific code elements, visual explanations appear in the Concept Window.If a user wants to change the learning mode they can click on the drop down list shown as visual in the below Fig.3

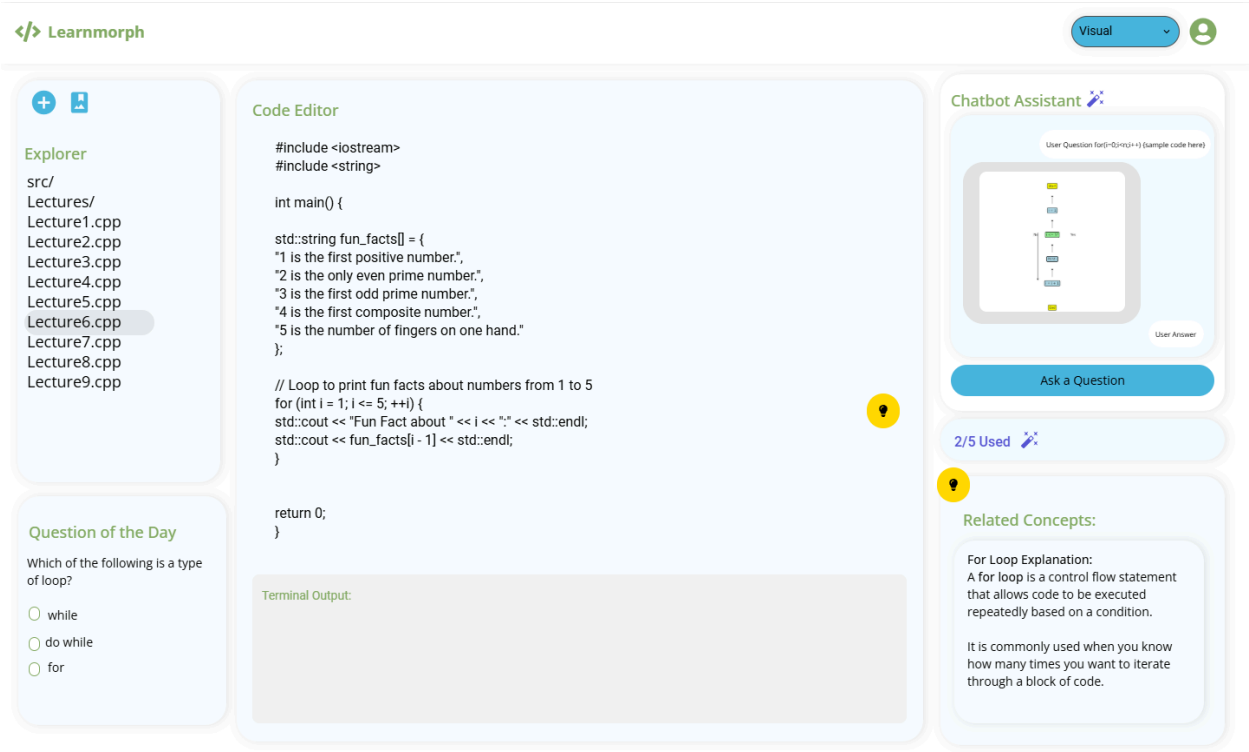

Fig.3 LearnMorph IDE for visual learning mode

The Kinesthetic Mode as shown in fig.4 Emphasizes hands-on interaction with the code. The Chatbot Assistant suggests interactive tasks, prompting students to modify variables or functions directly in the code to observe outcomes. The Read/Write Mode Offers text-based explanations and writing prompts. The Concept Window provides written explanations of coding concepts, and students are encouraged to document and explain their code.

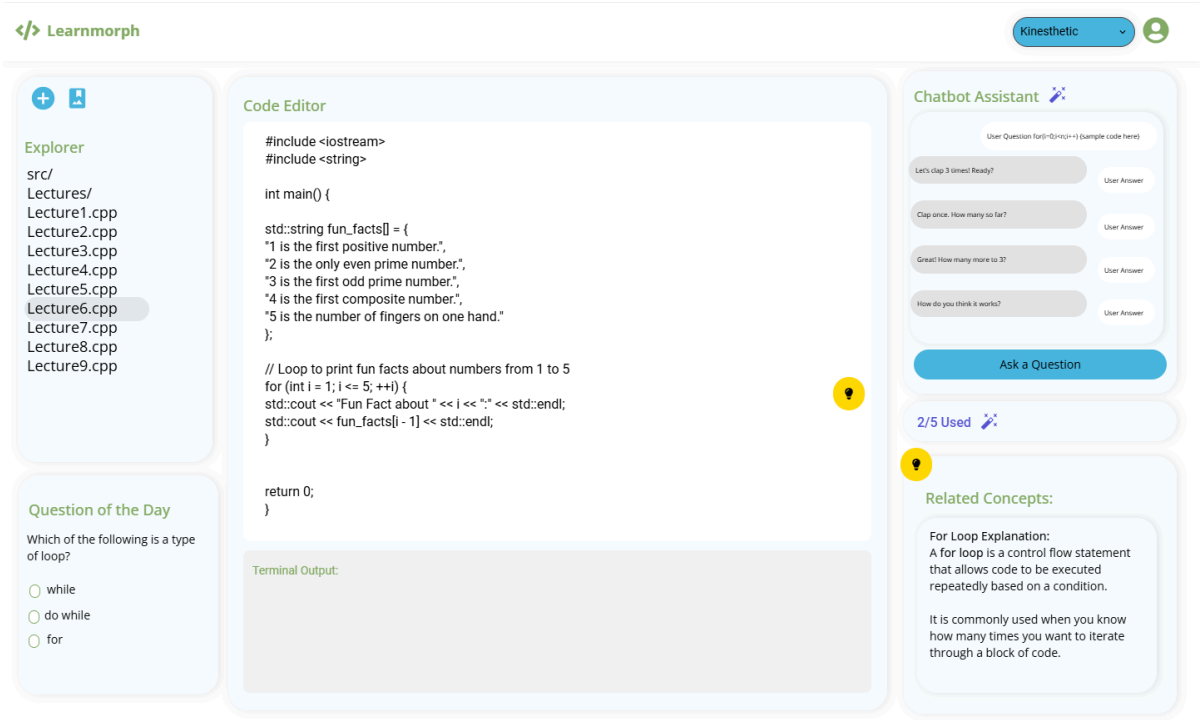

Fig.4 LearnMorph IDE in Kinesthetic learning mode

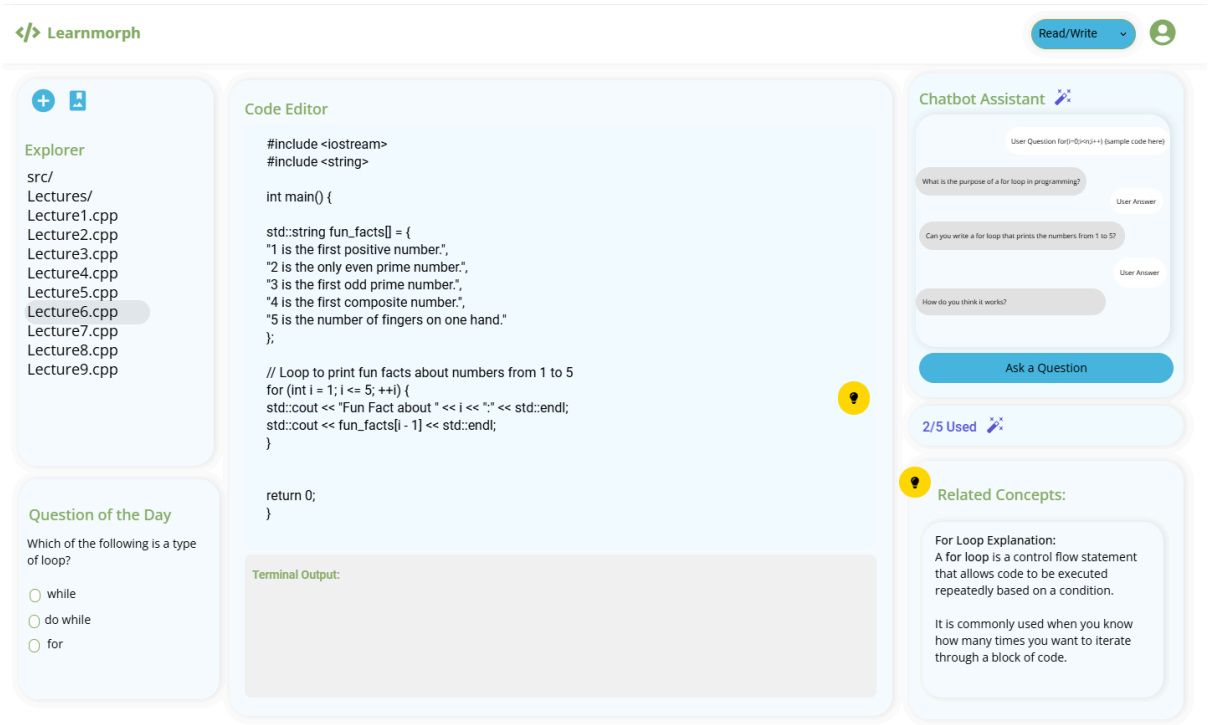

Fig.5 LearnMorph IDE in Read/Write learning mode

Chatbot Assistant: The Chatbot Assistant as shown in fig.5 showcased on the right-hand top corner guides students through coding challenges by posing questions and offering hints based on the current code. Key features include limited daily interactions (e.g., 5 attempts per day), attempt tracking displayed below the chatbot window, and integration with the bulb icon for concept explanations.

Concept Window and Bulb Interaction:*T*he Concept Window, located below the chatbot, provides explanations of key concepts. It works in conjunction with the bulb icon. Yellow bulb icons in fig.5 appear next to specific parts of the code (e.g., loops, functions, data structures). Clicking on a bulb icon provides conceptual explanations in the Concept Window.

Question Panel:Located at the left bottom of the editor as shown in fig.5, this panel displays programming-related questions. When a student correctly answers these questions, they earn additional LLM interactions, encouraging active learning and problem-solving.

User Interaction Flow: Users start on the Profile Page, reviewing their progress and analytics. They navigate to the Code Editor using the "Go to Editor" button. In the Code Editor, users can write code, select their preferred learning mode, and interact with the Chatbot Assistant and Concept Window as needed. The bulb icons appear next to relevant code elements. Clicking these icons provides context-specific information in the Concept Window. Users can answer questions in the Question Panel to earn additional LLM interactions. The Chatbot Assistant provides guidance, with a limited number of daily interactions tracked and displayed. Users can switch between learning modes at any time to adapt the IDE's behavior to their current needs.

7. User Study

7.1 The Design Thinking Approach

The design thinking approach was utilized to ensure a user-centered development process for LearnMorph IDE. Detailed empathy maps for various user personas were created to gain deep insights into the thoughts, feelings, and pain points of students when learning to code. This process helped identify specific struggles and frustrations, understand the emotional aspects of learning programming, and recognize the diverse ways students approach coding challenges.

Building on the empathy mapping insights, "How Might We"questions were formulated to address the identified challenges and guide the development of LearnMorph IDE. These questions served as a bridge between the understanding of user needs and the ideation phase, helping generate innovative solutions for different learning modes, the Chatbot Assistant, and the Concept Window. To further stimulate creative thinking, a "Worst Possible Ideas" exercise was used, which encouraged the consideration of unconventional approaches and identification of potential pitfalls in the design. This process helped avoid common mistakes and inspired more practical solutions for the IDE's features.

7.1.1 Empathy Mapping

Empathy mapping was an essential first step as it helped us to identify the specific struggles and frustrations students face when learning to code. It also helped us to understand the emotional aspects of the learning process, which can significantly impact motivation and engagement. Empathy Mapping in fig.6 helped to recognize the diverse ways students approach coding challenges and their preferred methods of understanding complex concepts.It uncovered potential features and design elements that could address the unique needs of different learning styles.

Fig.6 Empathy Map

7.1.2 "How Might We" Questions

Building upon the empathy mapping process, we formulated a series of "How Might We" (HMW) questions to address the challenges identified and guide the development of LearnMorph IDE. These questions serve as a bridge between the insights gained from empathy mapping and the ideation phase of the design process:

1. *How might we create a visual debugging experience that allows students to see code execution step-by-step?*
2. *How might we integrate interactive coding exercises that allow kinesthetic learners to manipulate code elements directly?*
3. *How might we develop an algorithm visualization tool that transforms complex algorithms into intuitive flowcharts or diagrams?*
4. *How might we implement a structured documentation system that guides read/write learners in effectively commenting their code?*
5. *How might we design a seamless interface that allows students to switch between visual, kinesthetic, and read/write modes without disrupting their coding flow?*

7.1.3 Worst Possible Ideas:

To encourage creative thinking and identify potential pitfalls, we conducted a "Worst Possible Ideas" session. This technique involves deliberately generating the worst possible solutions to a problem, which can lead to innovative insights and help avoid common mistakes. Some of the worst ideas generated included:

1. *Create a visual debugging experience that randomly rearranges code lines during execution, making it impossible to follow the logic.*
2. *Integrate interactive coding exercises where kinesthetic learners must physically jump on a dance pad to input code, leading to exhaustion and confusion.*
3. *Develop an algorithm visualization tool that transforms complex algorithms into abstract modern art pieces, completely unrelated to the actual code structure.*
4. *Implement a documentation system that automatically replaces all comments with random emoji sequences, making code comprehension impossible for read/write learners.*
5. *Design an interface that forces students to solve a puzzle every time they want to switch learning modes, disrupting their coding flow and concentration.*

While these ideas are intentionally absurd, they helped the team identify potential pain points and inspired more practical solutions, such as a language-switching feature for polyglot programming practice and a more engaging emoji-based feedback system for younger learners.

7.2 Study Design

7.2.1 Participants :

The study recruited 8 novice programmers aged 18 to 24 with little to no programming experience, ensuring a consistent sample of young adults in the early stages of learning. Participants had completed only basic programming courses, were comfortable with general computer use, but were unfamiliar with Integrated Development Environments (IDEs). To maintain focus on beginners, individuals with intermediate or advanced programming knowledge or professional experience were excluded.

7.2.2 Study Methodology

The primary objective of this study is to evaluate the perceived usability, intuitiveness, and potential effectiveness of the LearnMorph IDE UI for novice programmers. As the UI is non-functional, the study focuses on participants interacting with static designs and providing feedback based on visual and conceptual understanding.A within-subject design was used, where all participants reviewed and interacted with static UI mockups for each mode and feature. This approach allowed for comprehensive feedback on all aspects of the UI. Participants engaged in scenario-based evaluations using static mockups of the LearnMorph IDE, simulating workflows based on provided scenarios.

7.2.3 Variables

The independent variables in this study are the Learning Modes (Visual Mode, Kinesthetic Mode, and Read/Write Mode) and UI Features (Chatbot Assistant, Concept Window, Lives System, Profile Page). The dependent variables include participant understanding of the interface, perceived ease of use for completing hypothetical tasks, and feedback on clarity, intuitiveness, and layout.

7.2.4 Tasks

Several tasks were given to novice programmer students. These helped to know their pain points and how the users in real time would interact with the proposed system.

Task 1 Debugging in Visual Mode:Participants reviewed a flowchart to identify logical errors in a program, such as debugging a nested loop generating incorrect prime numbers. The purpose was to evaluate the visual clarity and comprehensiveness of flowcharts for debugging. Evaluation questions included "Can you identify where the error lies in the flowchart?" and "Is the flowchart easy to understand?" Metrics focused on the time taken to interpret the flowchart and feedback on clarity.

Task 2 Code Construction in Kinesthetic Mode: Participants simulated completing a coding task by rearranging code blocks in the mockup, such as reordering snippets to construct a factorial function. This task aimed to assess whether the drag-and-drop UI is intuitive and whether participants can understand the task flow. Evaluation questions included "What would your first step be to complete this task?" and "Are the code blocks and their functions clear?" Metrics included the ability to describe the process and perceived ease of use.

Task 3 Writing Code in Read/Write Mode:Participants simulated writing a program with guidance from the Concept Window, such as building a Fibonacci sequence generator with detailed textual instructions. The purpose was to evaluate the clarity and placement of the Concept Window and how well it supports writing code. Evaluation questions included "Do the instructions in the Concept Window provide enough guidance?" and "Is the layout easy to navigate while writing code?" Metrics focused on feedback regarding the clarity of instructions and ease of accessing information.

Task 4 Debugging with Chatbot and Lives System: Participants simulated interacting with the Chatbot Assistant to debug a grading system. They were asked how they would manage lives and interpret chatbot responses. This task aimed to assess the perceived usefulness of the

Chatbot and the stress or engagement introduced by the lives system. Evaluation questions included "How would you prioritize your chatbot queries with limited lives?" and "Do you find the concept of earning additional lives motivating or distracting?" Metrics included feedback on the concept of limited lives and perceived usefulness of the Chatbot.

Task 5 Profile Page Feedback: Participants reviewed the Profile Page mockup, which displays progress, achievements, and statistics. The purpose was to assess whether the Profile Page provides meaningful motivation and insights. Evaluation questions included "Does this page motivate you to track your progress?" and "Is the information presented here helpful or overwhelming?" Metrics focused on participant feedback regarding layout and motivational elements.

7.2.5 Data Collection

Data collected as feedback encompassed participant feedback on each mode and feature, including clarity, ease of understanding, and potential challenges. Observations of participant behavior during task explanations (e.g., hesitation or confusion) and comments on perceived strengths and weaknesses of the UI were also collected. Additionally, a survey feedback form was circulated to gather further insights.

8. Data Analysis

The Overall User Experience scores 3.8/5, suggesting an innovative design but with potential complexity for beginners. The Learning Mode Effectiveness demonstrates varied user preferences, with the Visual Mode (4.2/5) and Kinesthetic Mode (4.0/5) outperforming the Read/Write Mode (3.6/5), likely due to the engaging and simplified approaches of the former. The Chatbot Assistant, rated 3.5/5, shows promise but faces challenges with its daily interaction limit and inconsistency in contextual help. The Concept Window Utility stands out with a strong score of 4.3/5, emphasizing its value in dynamic and contextual learning support. The User Interface Intuitiveness achieves a moderate rating of 3.7/5, with a clean design but some interactive elements potentially overwhelming for users. Lastly, the Performance and Responsiveness was rated 4.0/5, reflecting the advantages of a modern tech stack while hinting at occasional latency in AI-driven features. These insights provide a clear roadmap for enhancing user satisfaction and effectiveness.

Observations as defined below in the Table.1 during task explanations revealed that students who were not very good at read/write type of study felt low about themselves when forced to follow that method. About 40 percent of users got frustrated when the use of LLM was abruptly limited. 80 percent of users enjoyed switching to different modes to break their monotonous way of learning.

Category	Rating (Stars)	Rationale
Overall User Experience	3.8/5	Innovative concept, but potential complexity for beginners.
Learning Mode Effectiveness		Reflects varying effectiveness across different learning styles.
- Visual Mode	4.2/5	Simplifies logic with flowcharts and diagrams.
- Kinesthetic Mode	4.0/5	Engaging drag-and-drop tasks but needs interactive feedback.
- Read/Write Mode	3.6/5	Text-based guidance useful but overwhelming for some users.
Chatbot Assistant	3.5/5	Limited daily interactions (5 attempts) might frustrate users; contextual help is promising but inconsistent.
Concept Window Utility	4.3/5	Dynamic concept display is a strong feature; provides contextual learning support.
User Interface Intuitiveness	3.7/5	Clean design with potential usability improvements; multiple interactive elements might overwhelm new users.
Performance and Responsiveness	4.0/5	Modern tech stack (React.js, Node.js) suggests good performance; potential latency in AI-powered features.

Table.1 Data Analysis of user interaction with the IDE

9. Results

The Google Form feedback and the user study revealed key qualitative insights. Participants appreciated the Visual Mode for its flowcharts simplifying logic and the Kinesthetic Mode for its engaging interactive design. The Read/Write Mode was valued for concise structured instructions but felt overwhelming due to technical language in the Concept Window. Challenges included difficulty correlating flowchart elements with code in Visual Mode, lack of interactive feedback in Kinesthetic Mode, and complexity of programming terminology in Read/Write Mode. Feedback on the Lives System was mixed: 60% enjoyed the gamification, while others found limited interactions stressful. The "Question of the Day" was engaging but distracting during complex tasks. Participants expressed frustration with the limited use of the LLM, highlighting inconsistent contextual assistance. Lastly, the Profile Page’s progress-tracking elements were motivating, with suggestions to include improvement tips directly. These results are also part of the user study conducted.

10. Limitations of Current System

Despite positive outcomes, several limitations were identified. The system lacks integration with version control systems, limiting collaboration features. The Chatbot Assistant's AI model sometimes provided generic responses that lacked context-awareness. It also supports only a limited number of programming languages, restricting its use in diverse coding courses. Additionally, the GPT-4 implementation occasionally produces technically correct but pedagogically misaligned responses. The cost of frequent API calls to the LLM could pose scalability issues, and some students found the process of earning additional LLM attempts time-consuming, disrupting their coding flow.

11. Conclusion and Future Work

LearnMorph IDE has great potential in enhancing the learning experience with its adaptive interface and personalized guidance. Its integration of multiple learning modes and an AI-powered Chatbot Assistant caters to diverse learning styles. Future improvements will focus on refining the AI model for more context-aware responses and expanding system capabilities.

12. References

[1] Prather, J., Reeves, B. N., Leinonen, J., MacNeil, S., Randrianasolo, A. S., Becker, B. A., ... & Briggs, B. (2024, August). The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers. In *Proceedings of the 2024 ACM Conference on International Computing Education Research-Volume 1* (pp. 469-486).

[2] Anindyaputri, N., Yuana, R. & Hatta, P. (2020). Enhancing Students' Ability in Learning Process of Programming Language using Adaptive Learning Systems: A Literature Review.*Open Engineering*, *10*(1), 820-829. https://doi.org/10.1515/eng-2020-0092

[3] Ling HC, Chiang HS. Learning Performance in Adaptive Learning Systems: A Case Study of Web Programming Learning Recommendations. Front Psychol. 2022 Jan 28;13:770637. doi: 10.3389/fpsyg.2022.770637. PMID: 35153951; PMCID: PMC8831801.

[4]Storer, K. M., Sampath, H., & Merrick, M. A. A. (2021, May). " It's Just Everything Outside of the IDE that's the Problem": Information Seeking by Software Developers with Visual Impairments. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (pp. 1-12).

[5] Robins, A., Rountree, J., & Rountree, N. (2022). Learning and teaching programming: A review and discussion. Computer science education, 13(2), 137-172.

[6] "Using Visual Programming Games to Study Novice Programmers", IJSG, vol. 10, no. 2, pp. 115–136, Jun. 2023, doi: 10.17083/ijsg.v10i2.577.

[7]Tianjia Wang, Daniel Vargas-Díaz, Chris Brown, Yan Chen,2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), Washington, DC, USA, 2023, pp. 92-102

[8] Brown T.B.,Mann B.,Ryder N.,Subbiah M.,Kaplan J.,Dhariwal P.,...& Amodei D.(2020).Language models few-shot learners.arXiv preprint arXiv :2005 .14165.

[9] http://research.google/blog/exploring-transfer-learning-with-t5-the-text-to-text-transfer-transformer/

[10] Peláez-Sánchez Iris Cristina , Velarde-Camaqui Davis , Glasserman-Morales Leonardo David : The impact of large language models on higher education: exploring the connection between AI and Education 4.0, Frontiers in Education VOLUME:9,2024
https://www.frontiersin.org/journals/education/articles/10.3389/feduc.2024.1392091

[11]Razafinirina, M.A., Dimbisoa, W.G. and Mahatody, T. (2024) Pedagogical Alignment of Large Language Models (LLM) for Personalized Learning: A Survey, Trends and Challenges. *Journal of Intelligent Learning Systems and Applications*, 16, 448-480. doi: 10.4236/jilsa.2024.164023

[12]Bashar Alhafni, Sowmya Vajjala, Stefano Bannò,Kaushal Kumar Maurya, Ekaterina Kochmar(2024) LLMs in Education: Novel Perspectives, Challenges, and Opportunities.

[13]Wenhan Lyu, Yimeng Wang, Tingting (Rachel)Chung, Yifan Sun, Yixuan Zhang.Evaluating the Effectiveness of LLMs in Introductory Computer Science Education: A Semester-Long Field Study.https://arxiv.org/abs/2404.13414