AN INTERACTIVE LEARNING ENVIRONMENT FOR PROGRAMMING LANGUAGES WITH GENERATIVE AI

W. P. G. M. Weerasinghe

239370J

MSc in Computer Science

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

June 2025

AN INTERACTIVE LEARNING ENVIRONMENT FOR PROGRAMMING LANGUAGES WITH GENERATIVE AI

W. P. G. M. Weerasinghe

239370J

Thesis/Dissertation submitted in partial fulfillment of the requirements for the degree

MSc in Computer Science

Department of Computer Science and Engineering

Faculty of Engineering

University of Moratuwa

Sri Lanka

June 2025

DECLARATION

I declare that this is my own work and this thesis does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other University or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date: 27/06/2025

The above candidate has carried out research for the Master's thesis under my supervision. I confirm that the declaration made above by the student is true and correct.

Name of Supervisor: Prof. Indika Perera

Signature of the Supervisor: Date: 27/06/2025

DEDICATION

In loving memory of my father whose unwavering support, wisdom and encouragement set the foundation for my academic career. His memory will always inspire me to achieve greater heights.

To my mother who grants me freedom and offers unwavering support, even during the most difficult moments. Thank you for your sacrifices, understanding and endless love. This achievement is as much yours as it is mine.

This research is dedicated to all individuals who are passionate about education and curious about the impact of AI technology.

May this work contribute, in some small way, to the ongoing dialogue about the transformative potential of AI technology in education.

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to all those who have contributed to the successful completion of this project.

First and foremost, I would like to extend my heartfelt gratitude to my research supervisor, Prof. Indika Perera for his immense support and invaluable guidance given to me from the beginning of the project. His expertise and encouragement have been instrumental in shaping the direction and quality of this project.

I would be grateful to the faculty and staff of Department of Computer Science for initial understanding and guidance we were given and for the immense support given throughout the project.

Last but not least, I want to express my appreciation to my family and friends for their unwavering support, understanding and encouragement. Their confidence in my abilities has been a constant source of inspiration.

This research journey has been a collaborative effort and I am sincerely thankful to all who have played a part, no matter how big or small.

ABSTRACT

Generative Artificial Intelligence (GAI) has emerged as a transformative force in education, offering the potential to revolutionize learning experiences. This study focuses on the design and implementation of an interactive learning environment for Python programming language, leveraging GAI, specifically the Gemini API, to deliver personalized and adaptive education. This system aims to address the limitations of traditional programming education such as lack of individualization, delayed feedback and limited interactivity by offering dynamic content generation, real-time AI-driven feedback and interactive learning experience. The evaluation of the developed system revealed high user experiences with 100% of user satisfaction, with most finding it easier to understand compared to traditional learning materials. Further they marked explanations/ contents as accurate and code suggestions as useful, while engagement features like daily challenges and AI-generated teaching tips were frequently highlighted as motivational. By integrating GAI, this study explores the opportunities and challenges of utilizing advanced AI technologies to enhance programming education, contributing to the development of future-oriented educational tools. Ultimately, this study contributes meaningfully to the ongoing dialogue on leveraging AI technologies to enhance teaching and learning experiences in education system. As we move to the new era of Generative Artificial Intelligence, this study attempts to design a future where technological advancements enhance educational achievements with seamless integration of AI.

Keywords: Generative Artificial Intelligence, Gemini API, Interactive Learning Environment, Programming Education, Adaptive Learning, Personalized Education.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
AI	Artificial Intelligence
GAI	Generative Artificial Intelligence
TAM	Technology Acceptance Model

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

Generative Artificial Intelligence (GAI) has emerged as a powerful technology with the potential to transform various industries, including education. One prominent application of GAI is the development of chatbots such as Gemini, ChatGPT, DeepSeek which can simulate human-like conversations and provide automated assistance to students and educators. The adoption of GAI in the education sector has gained significant attention globally as well as in Sri Lanka. As the education system seeks to keep steps with technological advancements, understanding the impact of GAI and devising strategies to adapt to its integration becomes crucial. On the other hand, the recent shift to online education due to the COVID-19 pandemic has accelerated the adoption of digital platforms for learning [1]. This transformation leads to accelerate examine of the tools and technologies supporting education and explore the adoption of GAI in the education context. Hanse exploring how GAI, fits into the educational context becomes even more pertinent.

The integration of these technologies into education system presents both opportunities and challenges. On one hand, GAI has the potential to enhance student engagement, improve learning outcomes and streamline administrative processes. Even though GAI has the capacity to serve as an assistant for the educators (e.g., to generate course materials and make suggestions) and a virtual tutor for students (e.g., to answer questions and facilitate collaboration), there have been some challenges with its use. Therefore, immediate action is needed to improve evaluation techniques and institutional rules in schools and other educational institutions. Further responding to the impact of GAI on the educational environment requires instructor training and student education [2].

The adaptation of education to the growing presence of GAI requires a comprehensive understanding of its impact on various stakeholders, including students, educators, administrators and policymakers. Additionally, ethical considerations and the development of guidelines for responsible use are vital to ensure the technology is leveraged effectively. This research aims to investigate the impact of GAI in the university education system and its opportunities and challenges in successful integration. Moreover, our research takes another step toward the development of an interactive and personalized learning environment for students learning programming languages specially focusing on Python language. This personalized learning environment functions as a dynamic companion, according to the specific learning requirements of each individual student. Our goal is to create a dynamic learning environment that adapts to the

changing landscape of education, particularly in the modern era of Generative Artificial Intelligence.

This effort arose from the awareness that education is changing and that students benefit from personalized, technology-driven solutions, particularly in the field of programming languages. Our commitment is to improve the learning experience by keeping up to date with cutting-edge technologies such as Generative Artificial Intelligence, specifically utilizing the Gemini API.

By conducting a thorough literature review, this study seeks to provide insights into the benefits and challenges associated with GAI integration with a focus on creating an interactive learning environment. It will explore its effects on student engagements, learning outcomes and administrative processes. The literature review examines existing research on the use of GAI in education to establish a solid foundation of knowledge. This review highlights the potential benefits and challenges associated with integrating GAI in the education system.

The expected outcomes of this research are twofold. Firstly, it aims to deliver a comprehensive analysis of the impact of GAI on student engagement, learning outcomes, academic processes and its associated opportunities and challenges within the educational system. Secondly, the research aims to develop an interactive and personalized learning environment for students to learn programming languages, leveraging the Gemini API to enhance the learning experience.

1.2 Research Problem

The integration of Generative Artificial Intelligence (GAI) technologies into education presents both opportunities and challenges. While GAI has the potential to enhance student engagement, improve learning outcomes and provide personalized learning experiences, several critical issues remain. One primary challenge is the accuracy and reliability of AI-generated educational content which can sometimes produce inaccurate or misleading information. Additionally, there are concerns about the lack of real-time, adaptive support in traditional educational settings making it difficult to cater to individual student needs. Furthermore, ethical considerations and the impact on academic integrity must be carefully addressed to ensure responsible and effective use of GAI in education system.

This research is crucial as it seeks to provide insights into the benefits and challenges associated with GAI implementation in education. Also, this provides a key focus to develop an interactive and personalized learning environment for students to learn Python programming language. The traditional approach to programming education often lacks individualized support, restricts learning experiences of some students. By addressing this aspect, the study hopes to transform programming education. This tailored interactive learning environment will address each student's specific needs, providing targeted assistance, real-time

feedback and a dynamic learning environment. This emphasis on individualized support is consistent with the expanding educational landscape, ensuring that Gemini API integration extends beyond basic adoption to provide students with a personalized, engaging and interesting programming learning experience.

1.3 Research Objectives

The expected outcomes of this research are twofold. Firstly, it aims to provide a comprehensive understanding of the impact of GAI on student engagement, learning outcomes and academic processes. Further to identify opportunities in teaching and learning. Secondly, the research aims to develop an interactive and personalized learning environment for students and educators to learn Python programming language optimizing the identified opportunities of GAI.

The research objectives of this study are:

1. Assess the impact of Generative AI on the education system.
2. Analyze and address the challenges in the current programming educational environments.
3. Develop an interactive and personalized learning environment for students to learn python programming language.

The findings of this study will contribute to the ongoing dialogue on leveraging GAI in education and provide insights for adapting the education system to this rapidly evolving technology. By actively developing an interactive and personalized learning environment to assist students in learning programming languages, the study aims to bridge the gap between technology and education, ensuring a seamless and beneficial integration of GAI, specifically through the Gemini API, into the education system. Ultimately, the goal is to enhance the quality of teaching and learning experiences in the era of GAI.

1.4 Chapter Synopsis

Chapter 1: Introduction

This chapter presents the research by providing the background information on the transformational capability of Generative AI (e.g., ChatGPT, Gemini) in education. It presents the research problem focusing on issues such as lack of personalization, poor interactions and accessibility constraints in traditional computer programming education. The chapter is focused on discussing how Generative AI can solve these issues and improve learning outcomes. Lastly, the chapter outlines the research objectives.

Chapter 2: Literature Review

This chapter provides an extensive literature review in the field of AI and its use in education. The chapter starts with the introduction to AI, where focus is given to Generative AI technologies like ChatGPT, GPT-4 and Gemini and a comparison of

performance between them. Development of AI in education, historically speaking, from birth to development and influence, is researched. A theoretical framework is introduced, bringing together learning theory and AI models for the management of the research. The chapter then delves into the role of Generative AI in programming education, discussing its opportunities and challenges. Finally, the chapter examines Gemini's potential in education, highlighting its opportunities for teaching and learning as well as the challenges it poses.

Chapter 3: Methodology

This chapter presents the research approach utilized in achieving the objectives of the study. It starts by presenting the data collection process which involves an in-depth literature review, user testing and the collection of feedback. The procedure for participant selection is presented including questionnaires design and dissemination. Data analysis procedures are presented including qualitative and quantitative analysis and validation and verification procedures. The chapter also enumerates the instruments used in the research including survey instruments (e.g., Google Forms), development tools (e.g., React.js, Node.js) and literature databases (e.g., IEEE Xplore). The approach promotes systematic and rigorous handling of the research objectives.

Chapter 4: System Design and Implementation

The chapter outlines the design and implementation of the interactive and personalized learning environment. It starts by describing the purpose and objectives of the system which are synchronized with the research objectives. The system architecture is described; consisting of its central components and attributes such as an interactive coding interface, real-time feedback and learning paths. The chapter describes the functionalities of the system, i.e., user login, code execution and tracking progress. Functional and non-functional requirements are described. Implementation facts are given, for example, about the technology stack (e.g., React.js, Node.js, MongoDB), system architecture (e.g., three-tier architecture) and integration with Generative AI (e.g., Gemini API). The chapter concludes by describing user interface design.

Chapter 5: Results and Evaluation

This chapter concludes the findings of the study and pinpoints the success of the developed interactive and personalized learning environment. The chapter starts with a synopsis of the findings determined from the literature review, emphasizing the influence of Generative AI on education as well as limitations in formal education systems. Test results and feedback collected are then presented, including participants' comments regarding usability, interest and efficacy of the learning environment. Qualitative data collected through interviews or focus groups are also presented, giving insight into user experience in greater detail. The chapter ends by assessing the system's performance against its strengths, weaknesses and points of improvement.

Chapter 6: Discussion and Future Work

The chapter concludes with an overview of the major findings and contributions of the research. The chapter starts off by reminding readers of the study's objectives and summarizing how they were addressed in the study. The chapter emphasizes the key contributions, including observations regarding the contribution of Generative AI to education, establishing an interactive learning environment and empirically grounded recommendations for the incorporation of AI into teaching computer programming. The study limitations, limits of scope and challenges are addressed. Lastly, the chapter describes potential future research directions like expanding the system to accommodate various programming languages, incorporating higher-level AI modules such as virtual mentors etc. The chapter concludes with a consideration of wider implications of the research for education in the future.

CHAPTER 2

LITERATURE REVIEW

The utilization of artificial intelligence (AI) in educational contexts is not a new concept, but the broad use of GAI applications has recently brought it to the forefront. Despite GAI relatively recent entrance into practical applications, the volume of literature on the topic has grown significantly. The incorporation of AI-powered chatbots in the educational setting has attracted substantial interest as a viable option to promote student engagement and improve learning outcomes [3]. In recent years, numerous research has been conducted to explore the efficiency of chatbots in various educational environments. This study intends to investigate the effects of GAI on educational processes and their consequence.

The integration of AI-powered chatbots in educational activities has emerged as a valuable domain for fostering student engagement and facilitating learning processes. Studies have consistently shown that chatbot technologies have the potential to enhance student interaction and improve learning experiences [4]. These technologies have been found to positively impact student success in higher education by potentially boosting motivation, engagements and learning outcomes. However, there is still ongoing debate and lack of consensus among educators regarding the specific impact of GAI in educational contexts [5].

Interactive learning environments have gained prominence in modern education for their hands-on approach, enabling learners to write and execute code while receiving immediate feedback, which fosters deeper comprehension and retention of programming concepts [1]. The integration of generative AI into these platforms further amplifies their effectiveness by creating customized learning paths, generating relevant educational content and adapting to students' unique strengths and weaknesses [2][3].

With these advances and the release of Generative AI tools, there are concerns regarding academic integrity in higher education. Some suggest that AI tools such as Gemini, ChatGPT can improve student learning and educators should adjust their teaching and evaluation procedures accordingly. Further, there is a shortage in scholarly literature on GAI tools [6]. They also evaluated the news reports to investigate the influence of GAI on higher education including academic integrity concerns, university responses, limitations of AI tool outputs and opportunities for student learning. The analysis finds divided public conversation, with a focus on integrity issues and novel assessment design. The potential for GAI tools to support students from disadvantaged backgrounds and the student perspective are underrepresented. The article emphasizes the need for a student-led discourse and involving students in policy development. According to the world university news on 04 March 2023, considering academic integrity and potential issues related to plagiarism, a number of prestigious universities, including Oxford and Cambridge,

have implemented bans on the use of ChatGPT. Among the elite Russell Group universities, eight out of 24 have explicitly notified their students that employing the AI bot for assignments would be considered as a form of academic misconduct. These universities include Manchester, Bristol, Edinburgh and Oxbridge. Additionally, numerous other universities across the country, such as Durham, Liverpool and Northampton, are now hastening to reassess their plagiarism policies in preparation for the upcoming assessments, prompted by the sudden prominence of ChatGPT in November. But there is a debate that these bans will actually serves its purpose. Since the wide range of adaptations are already take places and will be widely used. Furthermore, Microsoft has integrated artificial intelligence into its most well-known productivity applications, such as Outlook, PowerPoint, Excel and Word, with the goal of transforming how millions of people work every day. This indicates a growing awareness and proactive response from educational institutions to address the potential challenges posed by the use of AI in academic settings.

This paper [7] illustrates that, other than the language mastery, GAI can solve unique and challenging problems in coding, medicine, mathematics, law, psychology, vision and other fields without the need for specific prompting. Furthermore, in all of these tasks, the performance of GPT-4 is unexpectedly similar to the human-level performance and it typically considerably exceeds earlier models such as ChatGPT [7]. One of their evaluations, they have employed GPT-4 on LeetCode's Interview Assessment platform[1], which is designed to simulate coding interviews for software engineer positions at leading technology companies. Remarkably, GPT-4 successfully tackled all the questions from the three interview rounds (online assessment, phone interview and on-site interview) within a mere 10 minutes, despite being allotted a generous 4.5 hours. According to LeetCode's assessment criteria, GPT-4 achieved impressive scores of 8.96/10, 8.69/10 and 10/10 in the respective rounds. Furthermore, it outperformed a significant majority of users, surpassing 93%, 97% and 100% of them, respectively. It is important to note that these scores take into account various factors such as time taken, test cases solved per question and other relevant considerations.

In order to address the lack of opportunities for English as a Foreign Language (EFL) students during their learning journey, this research explores the possibility of utilizing a GAI tool [8]. The goal of the research is to evaluate GAI efficacy in generating high quality dialogue materials for an EFL chatbot system. The results show that conversations created by AI are appropriate for students at level A2 of the Common European Framework of Reference for Languages (CEFR), offering easily understood content and encouraging vocabulary learning. In order to keep

[1] LeetCode is a technology skills and interview preparation platform that students, professional programmers and employers use to study, practice and evaluate coding and programming abilities. (https://leetcode.com/)

users from becoming bored, the study also investigates the usage of alternate responses and assesses the readability of generated materials using a variety of metrics. The research indicates that using AI-powered chatbots in EFL instruction, especially for CEFR A2 competency, has the potential to overcome challenges and providing valuable learning resources to students.

Another study revealed the potential misuse of GAI, highlighting the concern that this AI could successfully complete university courses without genuine knowledge acquisition. Taking a broader perspective on IT security, we explored various methods to validate students' understanding and found that AI tools pose a significant threat to academic integrity. Restrictive measures alone are unlikely to suffice, as detection tools may be lacking and punishing misuse may be challenging. Instead, it is imperative to prepare students for responsible and effective use of this technology. Considering the substantial investments already announced, the widespread adoption of AI is inevitable. Therefore, it is crucial to continue fostering qualities such as critical thinking, independent work skills and creativity in students [9].

2.1 Artificial Intelligence

Artificial Intelligence (AI) is a transformative field of computer science that focuses on creating intelligent systems capable of performing tasks that normally require human intelligence [10]. These include learning, thinking, solving problems, understanding natural language and adapting with the changing times. Robots, computer vision, natural language processing and machine learning are some of the technologies that fall under artificial intelligence (AI).

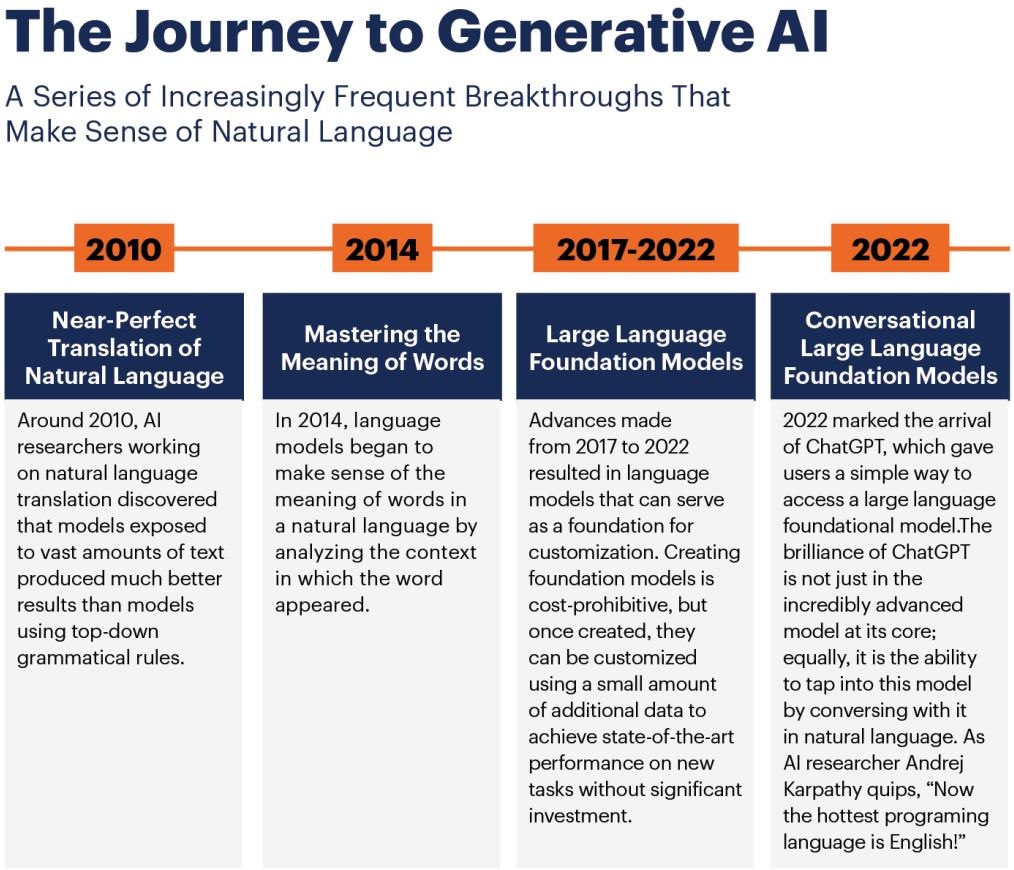
Algorithms and models that allow machines to process data, learn from it and make decisions is the basis of artificial intelligence. A subset of artificial intelligence called machine learning enables systems to gain experience and eventually improve the perform [11]. Role of AI in our daily lives and in various businesses is being shaped by ongoing research and technological developments that push the boundaries of what AI is capable of.

Further it is affecting education, focusing on how technology may help understand students' challenges, foster group creativity and provide new learning opportunities. A review of current solutions shows that although traditional educational systems continue to exist, artificial intelligence is changing the nature of education through its integration into systems like intelligent systems for assessment, massively open online courses and personalized learning environments. Personalized lessons, digital profiles and effective assessment are all made possible by the use of AI, which boosts productivity and flexibility. The study comes to the conclusion that AI technologies are already incorporated into a number of aspects of education, including teacher-student communication and material/ content development and that they bring important new perspectives to the field of technology-enhanced learning [12].

2.1.1 Generative Artificial Intelligence

Generative Artificial Intelligence (GAI) is a paradigm shift in the vast field of artificial intelligence (AI) that introduces computers that can produce materials which resembles human-like writing and have natural language conversations. This breakthrough has created new opportunities for dialogue systems, natural language understanding and the creative content generation.

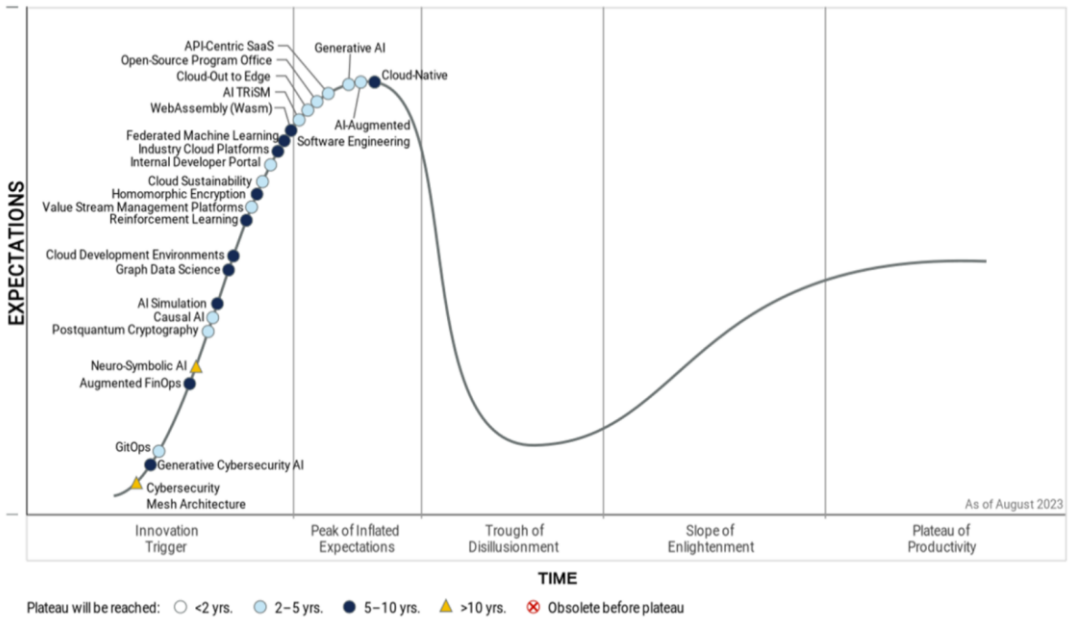
Figure 1 The journey to Generative Artificial Intelligence



Source - Gartner.com (2023)

GAI has reached the peak of inflated expectations on Gartner's Hype Cycle for Emerging Technologies, 2023, with an anticipated revolutionary benefit within two to five years [13].

Figure 2 Hype Cycle for Emerging Technologies, 2023



Source - Gartner.com (August 2023)

The introduction of OpenAI's ChatGPT in late 2022 aroused popular curiosity and pushed generative AI into the mainstream. Gartner compares the potential of generative AI as a general-purpose technology with the revolutionary inventions like steam engine, electricity and the internet. Gartner predicts that the influence of generative AI will increase as people and businesses find more creative uses for the technology in everyday life and work, even while the initial excitement will inevitably fade as implementation realities set in. This acknowledgment highlights how generative artificial intelligence is becoming more and more significant in influencing the direction of technology [14].

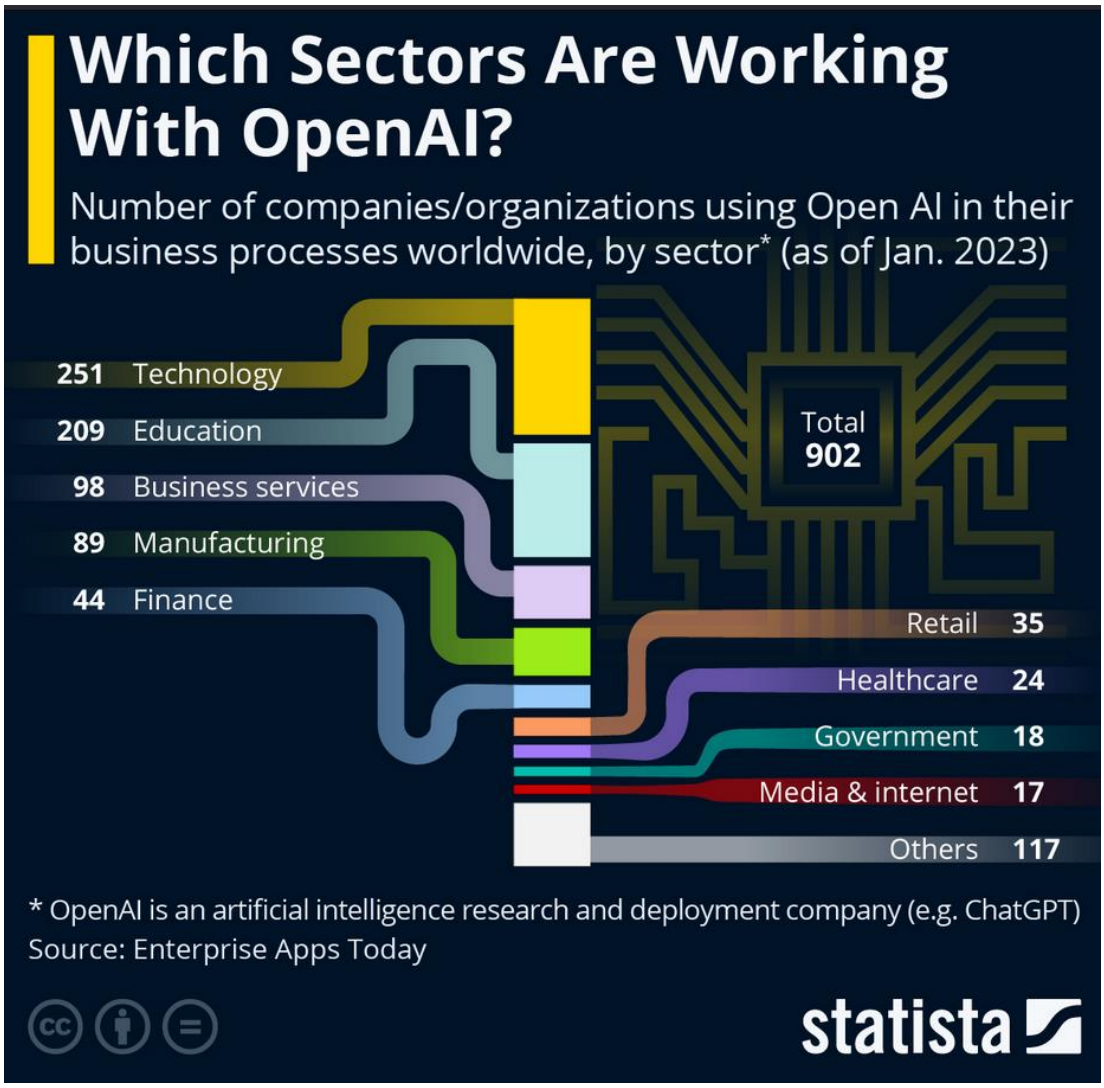
2.1.2 ChatGPT and GPT-4

The innovative AI research organization OpenAI has played a crucial part in expanding the capabilities of language models. ChatGPT is an important achievement that demonstrates the power of generative models by generating texts that are both coherent and contextually appropriate from user inputs. GPT-4 is the most recent version in the series and it boasts both scale and capability enhancements, improving language interpretation, context preservation and subtle text production [15].

According to OpenAI's website statistical data, with 17.2 million visitors in September 2022, the platform continues to demonstrate its popularity. With users from 156 countries, OpenAI's global accessibility is evident with 21.1 million monthly active users. ChatGPT has reached over 100 million active users as of January 2023. Because ChatGPT users have increased by 9,900% in just 60 days, the platform has been known to have the fastest growth in history. There were almost 173 million active ChatGPT users as of April 2023, a 576.6% growth in website visits to chat.openai.com. Each user spends approximately 8 minutes and 32 seconds on average. [16].

When it comes to industries making use of OpenAI, the technology sector leads with the greatest number of participating businesses. The education sector uses OpenAI in 209 institutions, which is a close second [17].

Figure 3 The number of companies working with OpenAI in different sectors as of January, 2023



Source- statcdn.com

2.1.3 Gemini

GAI has rapidly evolved, with models like Gemini pushing the boundaries of what is possible in language understanding, reasoning and code generation. Gemini, developed by Google, represents a significant advancement in this field, offering enhanced capabilities across various benchmarks.

Recent data indicates a strong and growing interest in Gemini. The average session length of 4 minutes and 40 seconds suggests that users are finding satisfactory conversational outcomes and achieving reasonable task success. Furthermore, a sharp upward trend in search traffic related to gemini.google.com has been noticed. Notably, in January 2025, visits rose to 13 million, an increase of 2.8 million from the previous month and traffic value increased by 2.7 million [18].

Figure 4 Search traffic related to gemini.google.com

Source – Sentisight.ai

2.1.4 Performance of Gemini compared to GPT

Google's CEO, Sundar Pichai, has expressed confidence in Gemini's capabilities, stating that it has surpassed OpenAI's ChatGPT in performance. Google has also set an ambitious goal of reaching over 500 million users for its AI applications by the end of 2025. A key strategy for achieving this goal is the integration of Gemini into Samsung smartphones, which provides a significant pathway to widespread adoption [19].

Figure 5 provides a comparative overview of Gemini Ultra's performance against GPT-4 across key benchmarks. These benchmarks are crucial for evaluating the capabilities of large language models (LLMs) in areas directly relevant to our research on interactive learning environments for programming languages.

Figure 5 Performance of Gemini compared to GPT-4

TEXT

Capability	Benchmark Higher is better	Description	Gemini Ultra	GPT-4 API numbers calculated where reported numbers were missing
General	MMLU	Representation of questions in 57 subjects (incl. STEM, humanities, and others)	90.0% CoT@32*	86.4% 5-shot** (reported)
Reasoning	Big-Bench Hard	Diverse set of challenging tasks requiring multi-step reasoning	83.6% 3-shot	83.1% 3-shot (API)
	DROP	Reading comprehension (F1 Score)	82.4 Variable shots	80.9 3-shot (reported)
	HellaSwag	Commonsense reasoning for everyday tasks	87.8% 10-shot*	95.3% 10-shot* (reported)
Math	GSM8K	Basic arithmetic manipulations (incl. Grade School math problems)	94.4% maj1@32	92.0% 5-shot CoT (reported)
	MATH	Challenging math problems (incl. algebra, geometry, pre-calculus, and others)	53.2% 4-shot	52.9% 4-shot (API)
Code	HumanEval	Python code generation	74.4% 0-shot (IT)*	67.0% 0-shot* (reported)
	Natural2Code	Python code generation. New held out dataset HumanEval-like, not leaked on the web	74.9% 0-shot	73.9% 0-shot (API)

* See the technical report for details on performance with other methodologies
** GPT-4 scores 87.29% with CoT@32 - see the technical report for full comparison

Source – blog.google

General Capabilities (MMLU): Gemini Ultra demonstrates a high level of proficiency in understanding and answering questions across a wide range of subjects, indicating its potential for providing comprehensive educational support.

Reasoning (Big-Bench Hard, DROP, HellaSwag): Gemini Ultra's strong performance in reasoning tasks highlights its ability to handle complex, multi-step problems, which is essential for providing effective feedback and guidance in programming education.

Math (GSM8K, MATH): Gemini Ultra excels in mathematical reasoning and problem-solving, suggesting its potential for assisting students with mathematical aspects of programming and related concepts.

Code (HumanEval, Natural2Code): Gemini Ultra showcases strong code generation capabilities, demonstrating its ability to assist students with code writing, debugging and optimization within the interactive learning environment.

These capabilities, combined with the increasing adoption of Gemini, are particularly effective in implementing interactive learning environment.

2.2 Historical Context of AI in Education

A brief historical overview of artificial intelligence in education follows its progress from the first intelligent tutoring system to the dominance of sophisticated AI applications today. Beginning in the 1960s with initiatives such as "PLATO," which introduced computer-assisted instruction [20], important turning points were reached in the following decades. Expert systems, such as MYCIN in medical diagnostics, were created in the 1980s [21], showcasing AI's ability to mimic human competence. Intelligent tutoring programs like AutoTutor first appeared in the 1990s and provided individualized instruction [22]. This study offers an overview of the evolution of computer tutors based on Anderson's ACT theory [23] over a ten-year period. These tutors were developed for courses including LISP, geometry and algebra and showed remarkable accomplishment benefits, with the best-case scenarios demonstrating equivalent proficiency to conventional instruction in significantly less time [24]. Data-driven methods emerged in the 21st century and sites like Khan Academy use AI to provide adaptive learning. The paradigm shifts towards machine learning, shown by Duolingo's use of AI for language instruction, is a characteristic of the modern era [25]. This historical trend demonstrates how AI is still evolving and influencing learning environments.

2.3 Theoretical Framework

The theoretical framework serves as a conceptual foundation for understanding the impact of GAI in educational settings. Drawing on the "Technology Acceptance Model (TAM)," as proposed by Davis, is a well-established theoretical framework commonly employed in the field of information systems and technology adoption [26].

When considering the implementation of AI in education, TAM emerges as a useful theoretical framework. Perceived ease of use refers to the degree to which individuals believe that using a particular technology would be free from effort

[26]. When it comes to GAI, instructors and students might consider how simple it is to include the chatbot into their lesson plans. This adoption is more likely if users believe it to be user-friendly and simple to integrate into current educational workflows. Conversely, perceived usefulness describes the degree to which individuals believe that using a specific technology would enhance their job performance [26]. Stakeholders may evaluate if GAI significantly improves student engagement, learning outcomes, or instructional efficiency in the classroom. For example, educators could assess it's value in helping students with individualized support during distance learning.

Using the Technology Acceptance Model (TAM) as a framework, this quantitative cross-sectional study investigates how English as a Foreign Language (EFL) learners use ChatGPT and how they perceive it [27]. According to an analysis of 405 EFL learners' responses, perceived ease of use indirectly affects learners' attitudes through perceived usefulness. However, positive attitudes regarding the usefulness of GAI are correlated with a higher intention to use the platform, which significantly predicts actual usage in English learning outside of the classroom. The study highlights GAI's role in promoting innovative and effective engagement within the Computer Assisted Language Learning (CALL) paradigm and offers empirical data in support of the tool's potential as an effective language-learning tool.

2.4 Role of GAI in Programming Education

Generative AI is increasingly transforming the educational landscape by enhancing teaching and learning processes. This innovative technology not only serves as a powerful tool but also acts as a substance for creative and personalized educational experiences. Generative AI focuses on creating new content; such as text, images and music based on patterns learned from extensive datasets, distinguishing it from traditional AI, which primarily analyses existing data to make predictions or classifications [28].

According to Biswas [29], GAI models can be used for a range of programming-related tasks. It can be used for text-to-code generation, document generation, chatbot development, code completion, correction, prediction, bug fixing and technical question answering. This has the ability to help developers and users understand complicated programming concepts, identify technical problems, debugging assistance, bug prediction, bug explanation and improve their overall satisfaction with support services by offering explanations, examples and guidance [29], [30]. Developers can leverage GAI models to streamline their coding process, save time and enhance the accuracy of their code, allowing them to focus on more critical tasks. Further as both researchers and practitioners have found, this can correctly solve a range of introductory programming assignments and appropriately explain the contents of code [31].

This study [32], which focused on the Object-Oriented Programming II course over an eight-week period, involved 41 undergraduate students from the Computer Technology and Information Systems department of a public institution. Weekly project assignments related to the course were given to the students, who were also encouraged to use ChatGPT, a GAI model to help them solve problems. Open-ended questions were used in the data collecting process and content analysis was used for the analysis. The conclusions of the research provide insight into the benefits and drawbacks in students' perspective. Students cited advantages like better thinking abilities, easier debugging, faster and generally more accurate answers to questions and increased self-confidence. Nevertheless, various drawbacks were also noted, such as worries about encouraging laziness, the sporadic incapacity to provide accurate answers to specific queries and inducing professional anxiety in students.

Some studies have highlighted the drawbacks of currently developed AI integrated tools in terms of programming education. The fundamental drawback is the possibility of producing erroneous results without quality assurance [33]. This can lead to errors in code or faulty explanations that appear plausible to novices. These tools may generate code that is not suitable, lacks robustness in edge instances, may include security flaws [34], or differs from what students are studying in a specific class. These technologies only have knowledge from their training data, which is an outdated snapshot of the web. As a result, they may be unable to assist with recently released things, such as a new JavaScript library published within the last week. Despite frequent retraining, attempts are undertaken to supplement them with web searches [35], [36]. Novice users may have difficulty delivering high-quality output using simple instructions [37]. Generating efficient and dependable prompts needs some amount of expertise knowledge [38]. Also, AI technologies can produce diverse results even when given the identical instruction. While there are parameters to reduce randomness, modifications to the underlying AI models may make the findings non-reproducible. AI-generated outputs may contain negative biases [39], [40]. For example, AI-generated code examples may include inappropriate stereotypes in variable names or strings [41], [42].

The decision between standard debugging tools and GAI models requires consideration of a number of factors [30]. Traditional debugging tools, which are often included in Integrated Development Environments (IDEs) and debuggers, can be costly. In contrast, GAI tools frequently uses a more adaptable and cost-effective cloud-based service framework. Another important consideration is speed, it provides quick and efficient bug explanations, but traditional debugging tools may take longer to process. The accuracy of issue predictions and explanations is critical, with each GAI models' performance influenced by the quality of its training data, whereas traditional debugging tools may provide a better level of accuracy but require a more thorough understanding of the code. Customizability favors traditional debugging tools, which are recognized for being extremely adjustable, over GAI's out-of-the-box design, which may lack the same amount of

customization. GAI model's ease of use is a plus, as it uses natural language generation for developer-friendly comprehension, whereas traditional debugging tools may have a steeper learning curve. Traditional debugging tools excel in integrating with other systems, however GAI tools may have disadvantages in this regard. Finally, scalability is a significant advantage for GAI models, which can debug code at scale, whereas traditional debugging tools may struggle with significant and complicated codebases.

These studies offer important information about how GAI might be used to create individualized learning environment for students learning computer programming languages. Its capacity to forecast code snippets, correct errors, optimize code and explanation ability is implying that it has the potential to help students better understand complicated programming concepts while enhancing their learning environment.

2.4.1 Opportunities in Programming Education

- Provide sample solutions

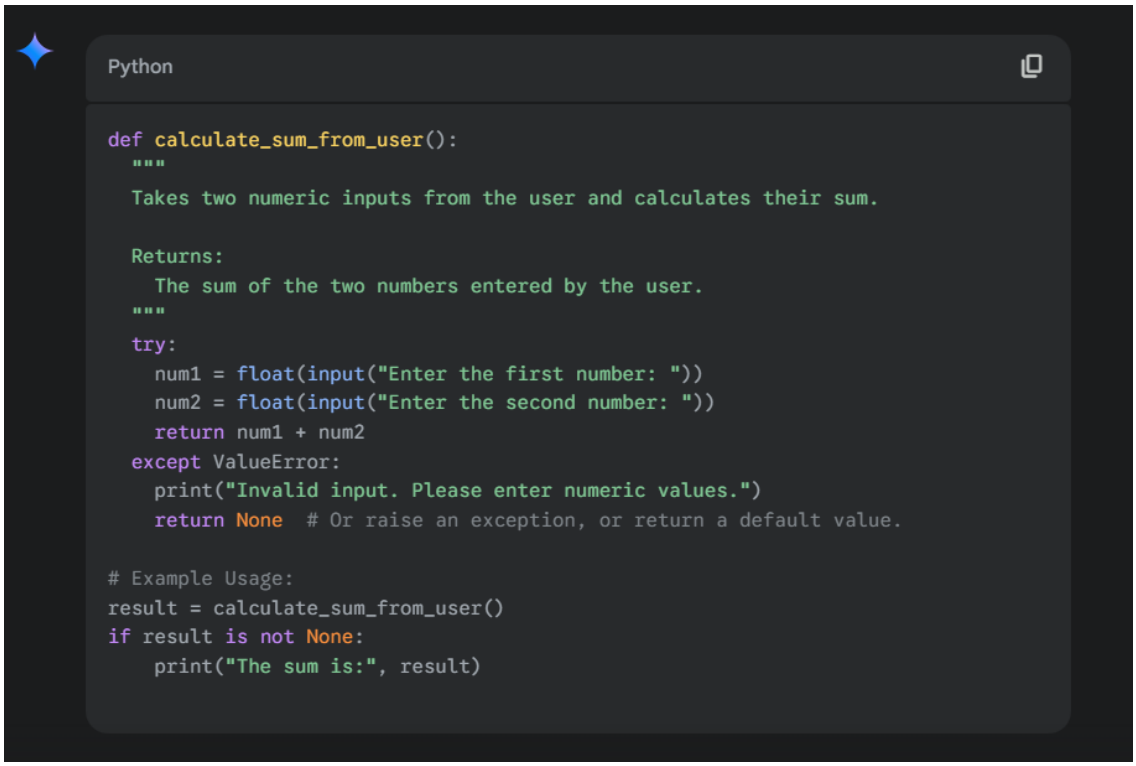
AI-generated solutions are a cost-effective solution for students studying programming who frequently complete auto-graded short problems. In the context of programming education, students may have difficulties in completing these tasks and even if they succeed, they may seek model solutions for greater comprehension. Instructors may not have the time to develop and publish model solutions for all exercises, quizzes and examinations. AI-generated answers bridge this gap by giving students with easily accessible sample solutions, allowing them to review their work during practice sessions [42].

- Generate solutions based on natural language descriptions

As an example, this enables code generation based on plain language descriptions of required functionality. For example, when given a description such as " *Write a function to take two numeric inputs and then calculate sum. This should be a C# function*," tools can automatically generate the necessary code [43].

The response that we received to the query " *Write a function to take two numeric inputs from the user and then calculate sum. This should be a Python function*." is shown below (Gemini 2.0 Flash - February 19, 2025).

Figure 6 Generate solutions based on natural language descriptions



Python

```
def calculate_sum_from_user():
  """
  Takes two numeric inputs from the user and calculates their sum.

  Returns:
    The sum of the two numbers entered by the user.
  """
  try:
    num1 = float(input("Enter the first number: "))
    num2 = float(input("Enter the second number: "))
    return num1 + num2
  except ValueError:
    print("Invalid input. Please enter numeric values.")
    return None  # Or raise an exception, or return a default value.

# Example Usage:
result = calculate_sum_from_user()
if result is not None:
    print("The sum is:", result)
```

Explanation Provided by Gemini;

Explanation:

1. ***def calculate_sum_from_user():***:
    - *Defines a function named calculate_sum_from_user.*
2. ***try...except ValueError:***:
    - *This block handles potential errors if the user enters non-numeric input.*
    - *try: The code within the try block is executed.*
    - *except ValueError: If a ValueError occurs (which happens when float() cannot convert the input), the code within the except block is executed.*
3. ***num1 = float(input("Enter the first number: "))***:
    - *input("Enter the first number: "): Prompts the user to enter the first number.*
    - *float(...): Converts the user's input to a floating-point number. This allows for decimal values. If you only want integers, use int().*
4. ***num2 = float(input("Enter the second number: "))***:
    - *Similar to the previous line, prompts for and converts the second number.*

5. ***return num1 + num2****:*
    - *Calculates and returns the sum.*
6. ***print("Invalid input. Please enter numeric values.")****:*
    - *Displayed if the user enters non-numeric input.*
7. ***return None****:*
    - *If there is invalid input, the function returns None. This lets the calling code know that the calculation failed. Alternatively, you could raise an exception, or return a default value like 0.*
8. ***Example Usage:***
    - *calls the function and stores the result in the result variable.*
    - *Checks if the result is not None, ensuring that the input was valid, then prints the result.*

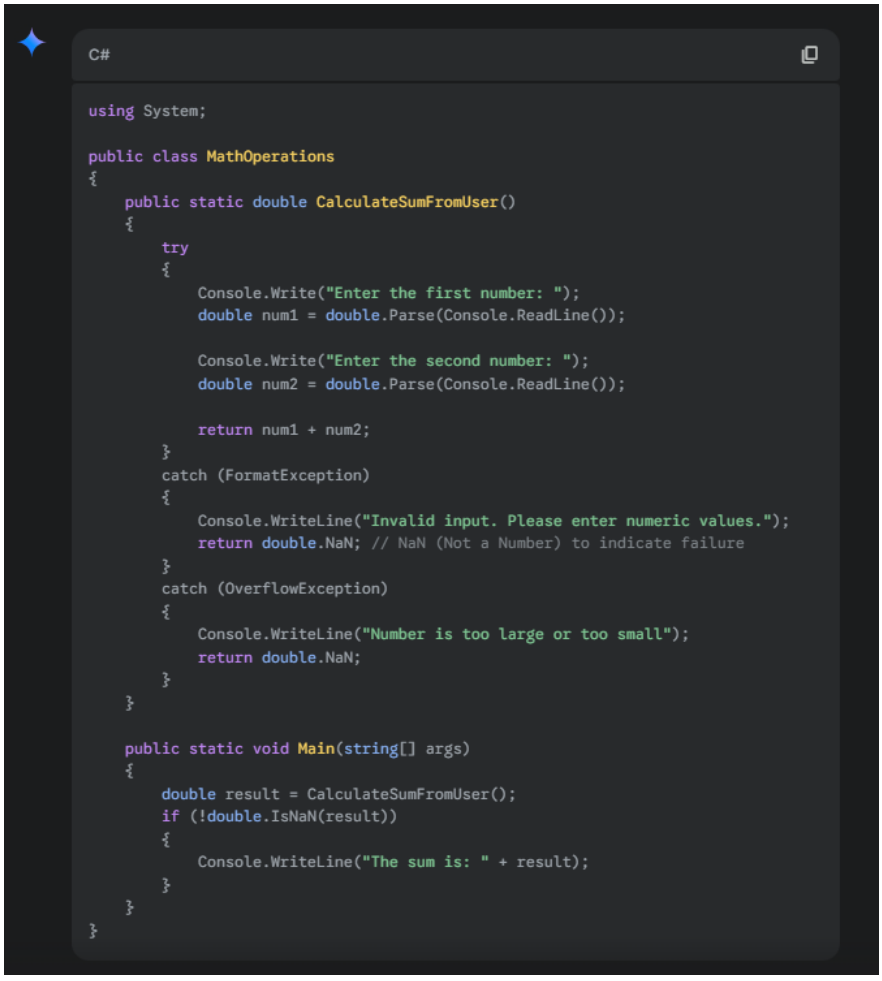
- Language translation

These AI tools can translate code from one programming language to another (although imperfectly). They can also translate references to human languages within code [43].

For the above sum calculation function, we asked it to write it in C#.

“Write above function in C#”
(Gemini 2.0 Flash - February 19, 2025).

Figure 7 Language translation



C#

```
using System;

public class MathOperations
{
    public static double CalculateSumFromUser()
    {
        try
        {
            Console.Write("Enter the first number: ");
            double num1 = double.Parse(Console.ReadLine());

            Console.Write("Enter the second number: ");
            double num2 = double.Parse(Console.ReadLine());

            return num1 + num2;
        }
        catch (FormatException)
        {
            Console.WriteLine("Invalid input. Please enter numeric values.");
            return double.NaN; // NaN (Not a Number) to indicate failure
        }
        catch (OverflowException)
        {
            Console.WriteLine("Number is too large or too small");
            return double.NaN;
        }
    }

    public static void Main(string[] args)
    {
        double result = CalculateSumFromUser();
        if (!double.IsNaN(result))
        {
            Console.WriteLine("The sum is: " + result);
        }
    }
}
```

Explanation Provided by Gemini;

Explanation:

1. ***using System;****:*
    - *Includes the necessary System namespace for input/output and number parsing.*
2. ***public static double CalculateSumFromUser()****:*
    - *Defines a public static method that returns a double (floating-point number).*
3. ***try...catch (FormatException)...catch(OverflowException)****:*
    - *Handles potential errors during input parsing.*
    - *try: The code within the try block is executed.*
    - *catch (FormatException): Catches errors when the user enters non-numeric input.*
    - *catch (OverflowException): catches errors when the number the user enters is too large or too small for the double type.*

4. ***Console.Write("Enter the first number: ");***:
    - *Prompts the user to enter the first number.*
5. ***double num1 = double.Parse(Console.ReadLine());***:
    - *Console.ReadLine(): Reads the user's input as a string.*
    - *double.Parse(...): Converts the string to a double.*
6. ***Console.Write("Enter the second number: ");***:
    - *Prompts for the second number.*
7. ***double num2 = double.Parse(Console.ReadLine());***:
    - *Parses the second number.*
8. ***return num1 + num2;***:
    - *Returns the sum.*
9. ***Console.WriteLine("Invalid input. Please enter numeric values.");***:
    - *Displays an error message if the input is invalid.*
10. ***return double.NaN;***:
    - *Returns double.NaN (Not a Number) to indicate that the calculation failed. This is a common way to signal errors in numeric functions.*
11. ***public static void Main(string[] args)***:
    - *The Main method is the entry point of the program.*
12. ***double result = CalculateSumFromUser();***:
    - *Calls the CalculateSumFromUser function and stores the result.*
13. ***if (!double.IsNaN(result))***:
    - *Checks if the result is not NaN.*
14. ***Console.WriteLine("The sum is: " + result);***:
    - *Prints the sum if the input was valid.*

- Diverse problem-solving approaches

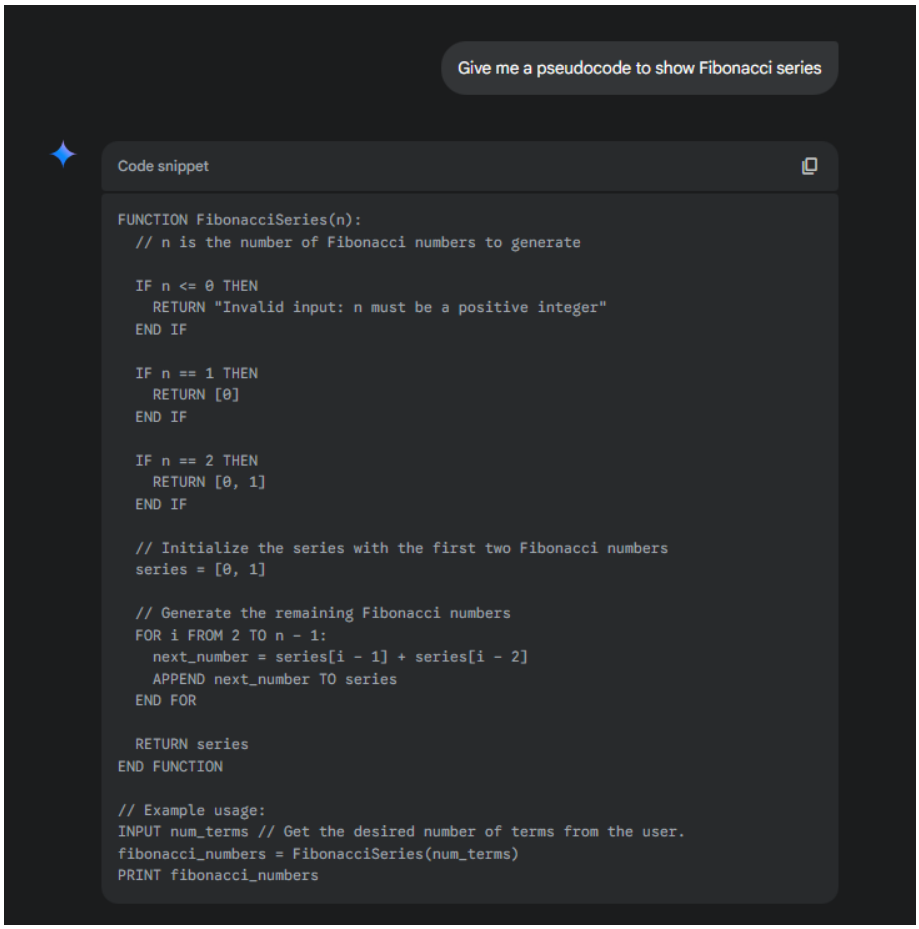
AI-based solutions, are critical for exposing students to a variety of problem-solving strategies. Many programming issues have several solutions, but novice learners may not completely grasp this diversity. It is necessary to provide learners with proper variance in programming teaching to help them identify the efficiencies and differences between coding styles [44]. Code generation tools, particularly for non-trivial problems, help to achieve this goal by creating a wide range of right answers, demonstrating the various ways a problem can be properly addressed [42].

- Conversational enhancement

Chatbots, like Gemini improve user participation by enabling multi-turn conversations. Unlike standalone interfaces, users can have back-and-forth conversations with the AI, allowing them to return to earlier context without re-entering the full request. For example, a user may say, "Now rewrite this code using more descriptive variable names," and Gemini, which retains discussion history, recognizes the reference to previously written code [43].

The response that we received to the query "*Give me a pseudocode to show Fibonacci series*" and "*Now write this in python*" are shown below (Gemini 2.0 Flash - February 19, 2025).

Figure 8 Conversational approach Figure 1



Explanation Provided by Gemini;

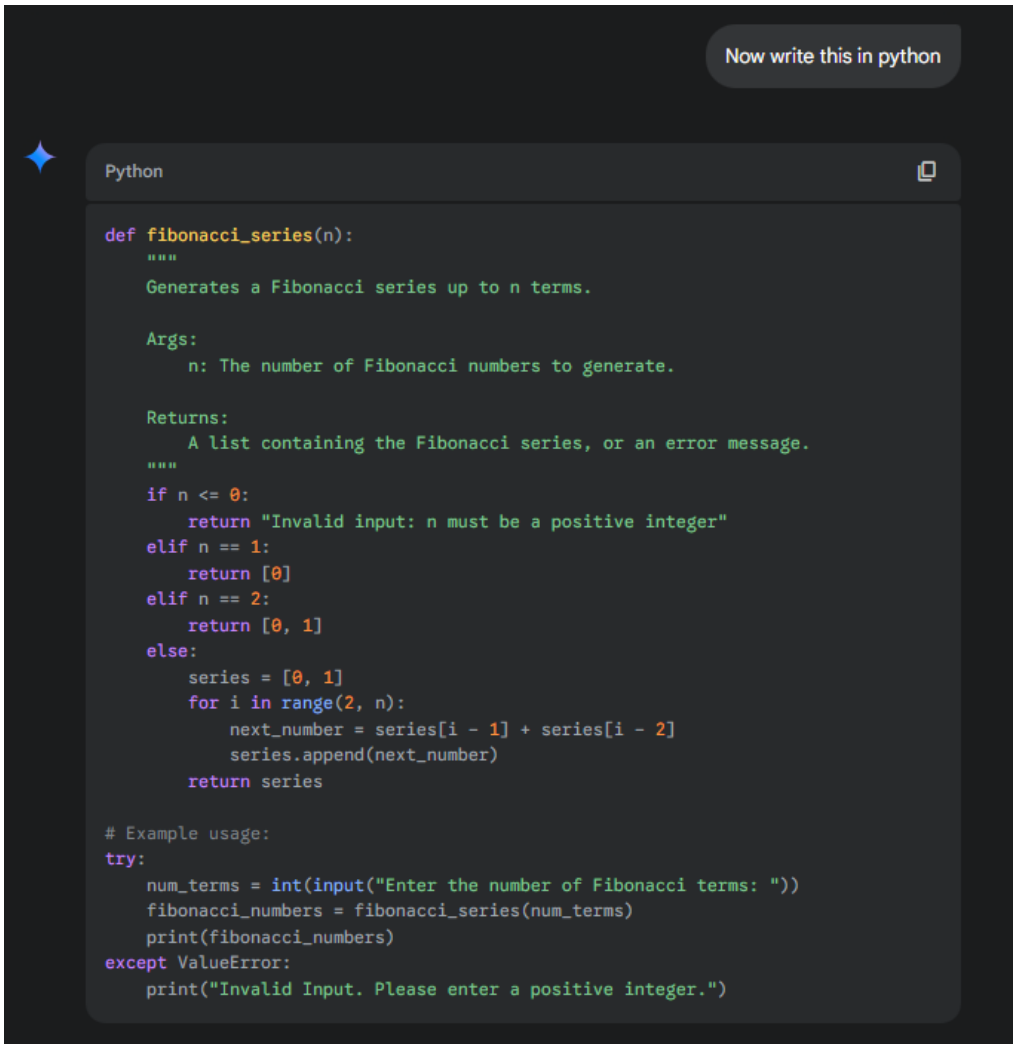
Explanation:

1. ***FUNCTION FibonacciSeries(n):***
    - *Defines a function named FibonacciSeries that takes an integer n as input.*
2. ***Input Validation:***

- *Checks if n is a positive integer. If not, it returns an error message.*
- *Handles the base cases where n is 1 or 2.*

3. ***Initialization:***
   - *Creates a list (or array) called series and initializes it with the first two Fibonacci numbers, 0 and 1.*
4. ***Iteration:***
   - *A FOR loop iterates from 2 up to n - 1 (inclusive).*
   - *Inside the loop:*
     - *next_number is calculated by adding the previous two Fibonacci numbers in the series.*
     - *next_number is appended to the series.*
5. ***Return Value:***
   - *The function returns the series containing the Fibonacci numbers.*
6. ***Example Usage:***
   - *Gets input from the user.*
   - *Calls the FibonacciSeries function.*
   - *Prints the resulting Fibonacci series.*

Figure 9 Conversational approach Figure 2



Explanation Provided by Gemini;

Explanation:

1. ***def fibonacci_series(n):****:*
    - *Defines the function fibonacci_series that takes an integer n as input.*
2. ***Input Validation:***
    - *if n <= 0:: Checks if n is less than or equal to 0. If so, it returns an error message.*
    - *elif n == 1: and elif n == 2:: Handle the base cases for n equal to 1 or 2.*
3. ***Initialization:***
    - *series = [0, 1]: Initializes the series list with the first two Fibonacci numbers.*
4. ***Iteration:***

- *for i in range(2, n):: Loops from 2 up to n - 1.*
- *next_number = series[i - 1] + series[i - 2]: Calculates the next Fibonacci number.*
- *series.append(next_number): Adds the calculated number to the series.*

5. ***Return Value:***
    - *return series: Returns the complete Fibonacci series as a list.*
6. ***Example Usage:***
    - *try...except ValueError:: Handles potential errors if the user enters non-integer input.*
    - *num_terms = int(input("Enter the number of Fibonacci terms: ")): Gets the number of terms from the user.*
    - *fibonacci_numbers = fibonacci_series(num_terms): Calls the function to generate the series.*
    - *print(fibonacci_numbers): Prints the resulting series.*
    - *print("Invalid Input. Please enter a positive integer."): Handles the case where the user enters non integer data.*

- Code review

In the context of introductory programming classes, provide more attention to code accuracy rather than code quality and style. Automatically generating syntactically valid solutions allows for differentiation and evaluation of solution style and quality. Code generation models can create different quality solutions for assessment problems, engaging students at higher levels of Bloom's taxonomy[2] by focusing on code quality evaluation [42].

- Code and concept explanations and feedback

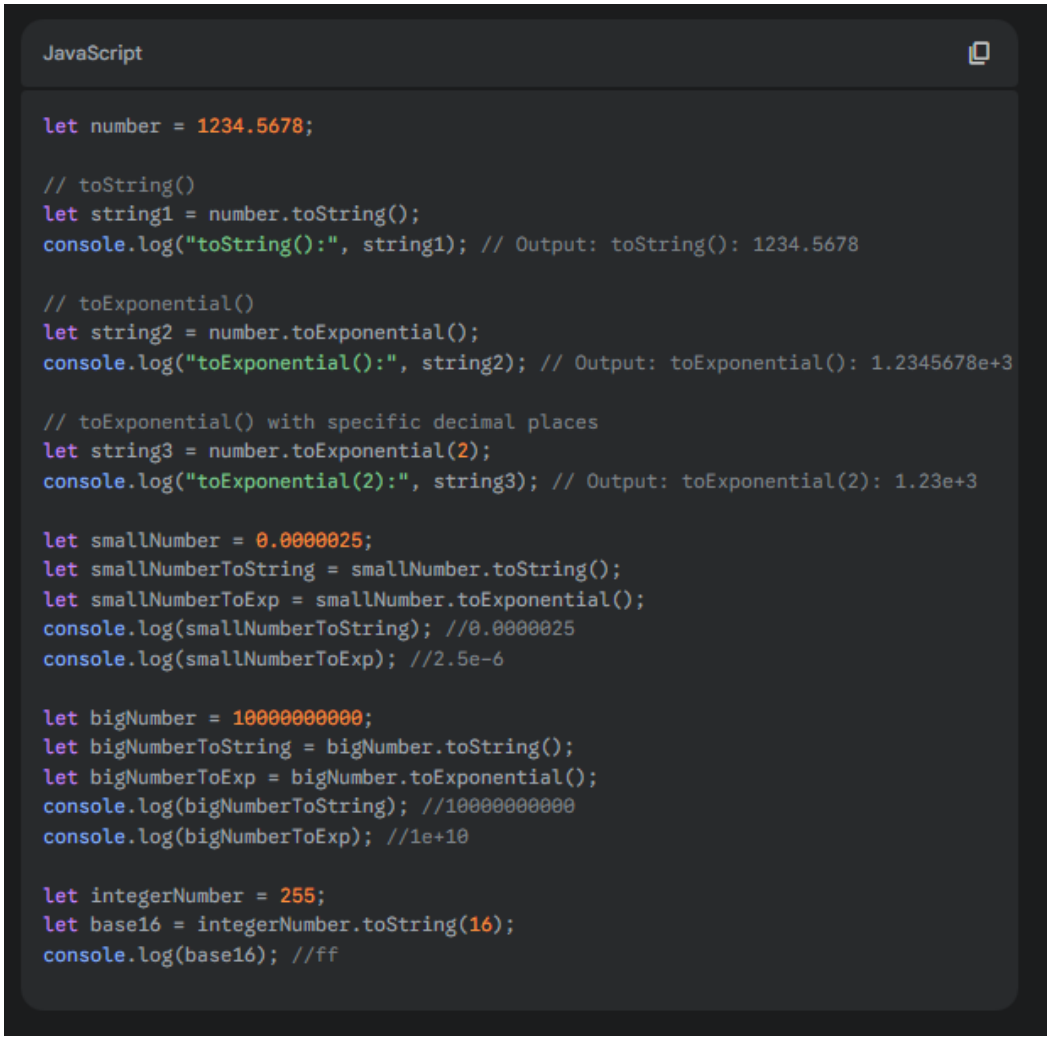
Code explanations and feedback are critical for improving students' understanding of programming topics. Also, users can ask the tool to clarify programming concepts with example codes, exactly like a human instructor [43]. AI enabled tools noted for its capacity to generate high-quality explanations for complicated code parts, makes a significant contribution to this educational resource [42], [45].

The response that we received to the query "*What is different between toString() and toExponential() in JavaScript. Explain it using an example*?" is shown below (Gemini 2.0 Flash - February 19, 2025).

---

[2] Bloom's taxonomy is a set of three hierarchical models for classifying educational learning objectives based on their complexity and specificity. The three lists include learning objectives in the cognitive, affective and psychomotor domains.
(https://en.wikipedia.org/wiki/Bloom%27s_taxonomy)

Figure 10 Code and concept explanations



```javascript
let number = 1234.5678;

// toString()
let string1 = number.toString();
console.log("toString():", string1); // Output: toString(): 1234.5678

// toExponential()
let string2 = number.toExponential();
console.log("toExponential():", string2); // Output: toExponential(): 1.2345678e+3

// toExponential() with specific decimal places
let string3 = number.toExponential(2);
console.log("toExponential(2):", string3); // Output: toExponential(2): 1.23e+3

let smallNumber = 0.0000025;
let smallNumberToString = smallNumber.toString();
let smallNumberToExp = smallNumber.toExponential();
console.log(smallNumberToString); //0.0000025
console.log(smallNumberToExp); //2.5e-6

let bigNumber = 10000000000;
let bigNumberToString = bigNumber.toString();
let bigNumberToExp = bigNumber.toExponential();
console.log(bigNumberToString); //10000000000
console.log(bigNumberToExp); //1e+10

let integerNumber = 255;
let base16 = integerNumber.toString(16);
console.log(base16); //ff
```

Explanation provided by Gemini;

Both toString() and toExponential() in JavaScript are methods used to convert a number to a string, but they do so in different ways, leading to different string representations.

toString():

- *Converts a number to a string in its standard decimal representation (or in a specified radix/base).*
- *It simply represents the number as we usually write it.*

toExponential():

- *Converts a number to a string in exponential notation (scientific notation).*
- *This is particularly useful for representing very large or very small numbers.*
- *It provides control over the number of digits after the decimal point in the exponential part.*

Key Differences Summarized:

- ***Representation:***
    - *toString(): Normal decimal representation.*
    - *toExponential(): Exponential (scientific) notation.*
- ***Use Cases:***
    - *toString(): General-purpose conversion to a string.*
    - *toExponential(): For very large or very small numbers, or when you specifically need exponential notation.*
- ***Control:***
    - *toString(): You can specify the radix (base) for integer numbers.*
    - *toExponential(): You can specify the number of digits after the decimal point in the exponential part.*
- ***Readability of very large or small numbers:***
    - *toString(): Can become hard to read for extremely large or small numbers.*
    - *toExponential(): Maintains readability for extremely large or small numbers.*
- Code refactoring and simplification

Users can utilize code refactoring and simplification to improve readability, style and maintainability of their code. For example, "*Refactor this function to use smaller helper functions.*" Or "*Can this be simplified*?" Engaging with ChatGPT and answering its queries can lead to improved code generation [43].

- Generate test cases

Users can use AI technologies to produce test cases, such as unit/regression tests. Copilot's TestPilot functionality allows for interactive test generation and refinement depending on user feedback. These tools can provide tests for unique edge scenarios that beginners may not consider [43].

2.4.2 Challenges in Programming Education

- Inaccurate data

Limitations of these technologies include the potential for erroneous outputs and lack of quality guarantees. This can lead to inaccuracies in code and explanations that novices may believe [33], [43].

- Code quality issues

Non-ideal code, lack of robustness to edge cases, security vulnerabilities, or misalignment with student learning objectives [43].

- Academic misuse

AI-based code generators have showed proficiency in scoring high on exams and producing proper code. Concerns regarding contract cheating arise, since AI-generated solutions provide a low-risk/high-reward option for students wanting immediate grades [42].

- Learning curve

Novice users may struggle to provide high-quality results when given simple suggestions. Creating effective and dependable prompts requires competence [43].

2.4.3 Challenges in Current Programming Learning Tools

- Lack of Personalized Content

Most learning resources for programming are one-size-fits-all, presenting the same material to all users regardless of their interest, skill level or experience level [46]. This is frustrating and demotivating because the content is too advanced for beginners and too simple for experienced people [47].

Therefore, learners are unable to progress at their own pace and the lack of customization reduces the effectiveness of the learning experience.

- Need to Refer to Multiple Resources

Students may need to go for additional resources (such as Stack Overflow, YouTube and documentation) in order to try to grasp a concept because the current tools do not adequately provide thorough explanations or examples. [46]. This fragmented approach can be time-consuming and overwhelming.

Learners spend more time searching for information than actually learning, which can lead to frustration and a lack of focus.

- Lack of Real-Time Feedback

Most programming platforms are not in a position to offer students timely feedback after submitting code or bugs. Students must wait for human grading or static error messages that do not instruct them on how to correct the error.

Without instant feedback, students are not able to learn from their mistakes immediately, which slows down the learning process and lowers the level of interest.

- Limited Interactivity

Interactive features such as live coding environments, interactive quizzes or hands-on exercises are not available in most online tools. Thus, learning is passive and

dull. Without direct interaction or active participation, students are less likely to acquire or develop practical coding skills [48].

For example, a course that does not provide video lectures but also does not include coding exercises or interactive features does not engage students actively.

- Inadequate Support for Debugging

Debugging is a critical skill in programming, but many tools do not provide adequate support for identifying and fixing errors [48]. Learners are often left to figure out issues on their own, which can be discouraging.

Learners struggle to develop problem-solving skills and may become demotivated when faced with persistent errors.

- Lack of Gamification and Motivation

Gamification elements such as badges, leaderboards and progress tracking, which may motivate learners to remain active and complete courses are generally not part of the programming tools.

The students will be more likely to lose interest in the subject or abandon the courses if they remain unmotivated.

- Limited Accessibility and Inclusivity

It is difficult for differently abled students (hearing or vision impairment) to employ the majority of technology because they were not designed to be used by them. Additionally, some instruments may not be equipped with other languages or cultural settings.

It excludes many potential students, thereby reducing the overall reach and effectiveness of the tool.

- Outdated or Irrelevant Content

Technologies and programming languages evolve rapidly, but there are few tools that regularly update their content. This may lead to students gaining knowledge that is outdated or useless.

Learners may acquire skills that are no longer in demand or struggle to adapt to modern programming practices.

- High Cost of Premium Tools

Because of the price of a subscription or a license, students with limited financial resources cannot even afford to buy the majority of advanced programming platforms and tools.

There are inequality in learning opportunities because those students who are from low-income backgrounds cannot use quality materials.

- Lack of Community and Peer Support

Some platforms do not provide the student with the opportunity to interact with peers or mentors, which may turn learning into a lonely process that is not as productive.

Learners miss out on the benefits of collaborative learning, such as sharing knowledge, solving problems together and receiving encouragement.

2.5 Gemini in Education

In the investigation of Gemini's role in education, this part explores into a thorough study of both challenges and potential opportunities. The review includes a variety of perspectives and scholarly insights, providing a complex implication of incorporating Gemini into educational contexts. By combining data from diverse sources, we receive useful insights on the potential benefits of Gemini in improving educational procedures. Simultaneously, the review identifies challenges, ethical considerations and areas for caution that educators and organizations must manage when implementing this technology in educational settings. Through this investigation, we hope to provide a comprehensive understanding of Gemini's impact on education, both positive and negative.

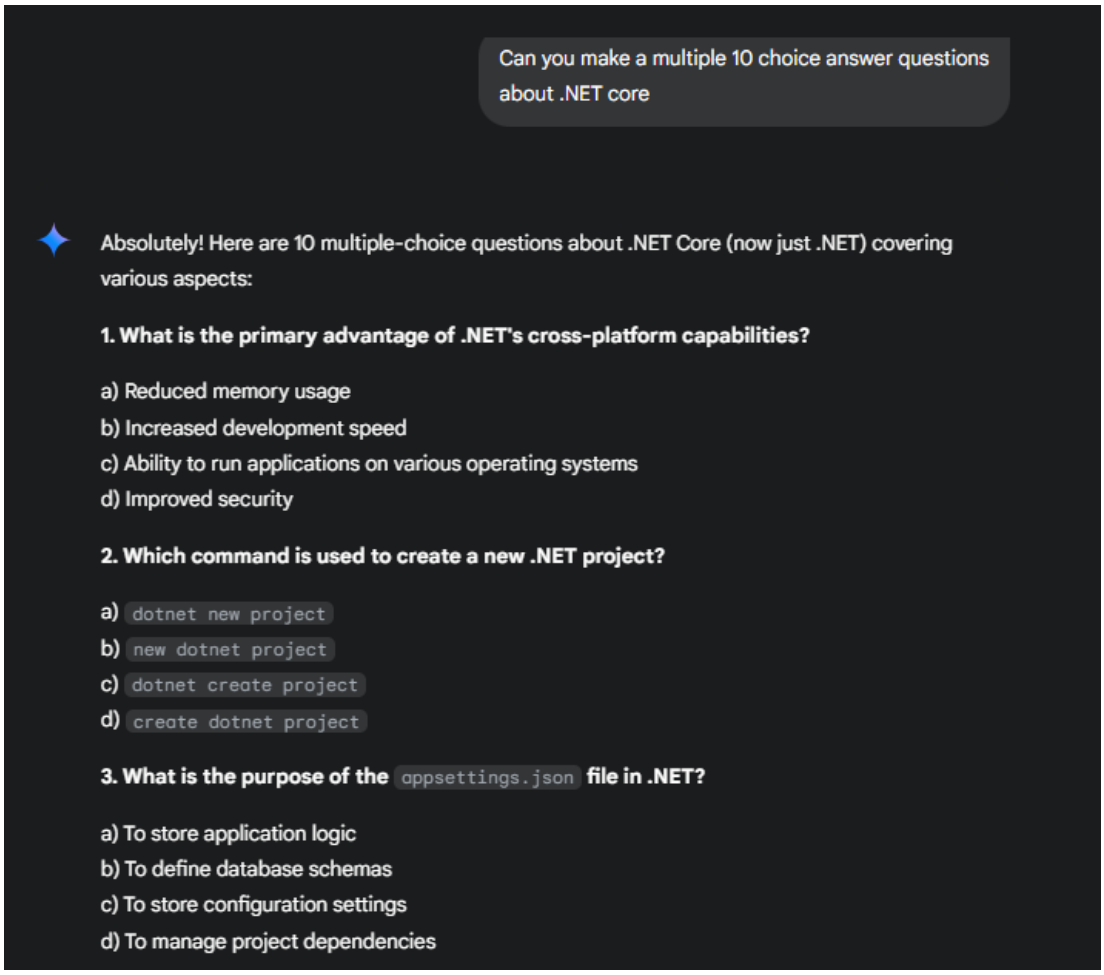
2.5.1 Opportunities in Teaching

- Generate and provide course materials

One major opportunity is the creation and delivery of course materials. GAI models can be used to efficiently generate educational content such as lecture notes, assignments and supplemental materials [49], [50], [51]. This not only simplifies the content generation process for instructors, but also allows for the personalization of materials depending on the specific needs and preferences of students [52].

The response that we received to the query "*Can you make a multiple 10 choice answer questions about .NET core?"* is shown below (Gemini 2.0 Flash - February 19, 2025).

Figure 11 The quiz generated by Gemini



Full quiz generated by Gemini;

1. What is the primary advantage of .NET's cross-platform capabilities?

a) Reduced memory usage
b) Increased development speed
c) Ability to run applications on various operating systems
d) Improved security

2. Which command is used to create a new .NET project?

a) dotnet new project
b) new dotnet project
c) dotnet create project
d) create dotnet project

3. What is the purpose of the appsettings.json file in .NET?

a) To store application logic
b) To define database schemas
c) To store configuration settings
d) To manage project dependencies

4. What does the acronym "CLR" stand for in .NET?

a) Common Language Runtime
b) Compiled Language Registry
c) Central Language Repository
d) Code Library Resource

5. Which of the following is a key feature of ASP.NET Core (now just ASP.NET) for web development?

a) Windows Forms integration
b) Razor Pages and MVC support
c) Direct hardware access
d) Legacy COM component compatibility

6. What is NuGet used for in .NET?

a) Code compilation
b) Package management
c) Database migrations
d) User interface design

7. What is the purpose of the dotnet build command?

a) To run the application
b) To create a new project
c) To compile the application's source code
d) To install NuGet packages

8. What is the role of the Startup.cs (or Program.cs in newer versions) file in an ASP.NET Core (ASP.NET) application?

a) To define database connections
b) To configure middleware and services
c) To manage project dependencies
d) To handle user authentication

9. What is the purpose of the DbContext class in Entity Framework Core?

a) To define user interfaces
b) To manage database connections and entities
c) To handle network requests
d) To store application settings

10. What is a key characteristic of .NET's garbage collector?

a) Manual memory management
b) Automatic memory management
c) Fixed memory allocation
d) Static memory allocation

- Perform language translation

The demonstrated language translation capabilities, making it a useful tool for transforming instructional resources into multiple languages. This feature expands the reach of educational content, making it more accessible to a diverse audience [45], [53]. Additionally, the model's capability to operate in over 40 languages promotes inclusivity and ensures that non-native English speakers can engage effectively with educational content [54].

- Evaluate student performance

The adaptability extends to the evaluation of student assessments, which is a potential answer for automating grading operations [49], [53]. This feature has the potential to reduce teachers' workload, allowing them to devote more time to other important areas of teaching [45].

- Inspiring Creativity: Fresh Ideas for Engaging Lessons

Gemini helps to inspire a educator's creativity by offering fresh ideas and new approaches to content, lessons and assignments [54].

2.5.2 Opportunities in Learning

- Provide answers

GAI models including Gemini is proficient at providing relevant responses, making it an invaluable tool for quick reference and self-study [9][53].

- Summarize information

It demonstrates commendable capabilities in processing information, distilling it and presenting it verbally [49].

- Collaborative learning

The potential of generating diverse scenarios enables the students to collaborate in problem solving and goal achievement [9]. GAI facilitates interactive learning by allowing students to converse with a virtual tutor [45].

- Adaptive learning

Utilizing GAI in the development of adaptive learning systems allows for personalized teaching methods that adapt to a student's progress and performance. These adaptive systems enhanced support for students learning programming, leading to notable improvements in their performance on programming assessments [45].

When it comes to Gemini one of the most transformative aspects of Gemini in education is its ability to personalize learning experiences. By tailoring content to individual student needs, Gemini enables teachers to provide customized support that caters to diverse learning preferences and paces [54][55]. This personalized

approach not only boosts student engagement but also enhances motivation, as learners feel more connected to the material being taught [55][56].

- Advanced Reasoning Capabilities

Unlike many earlier models that focus primarily on language generation, Google Gemini includes advanced reasoning capabilities. This allows it to tackle more complex and nuanced educational tasks, providing deeper insights and problem-solving assistance for students [55].

- Multimodal Capabilities

Gemini's ability to process and generate both text and visuals allows it to present information in various formats. This multimodal approach enables students to interact with content through text, images, charts and even videos, which can significantly enhance comprehension and retention [55].

- Self-learning tool

A review of Gemini's capabilities revealed its effectiveness as a reference and self-learning tool. Its advanced reasoning and access to vast information make it a valuable resource for independent study and knowledge enhancement [57].

- Providing personalized real time feedback

Gemini’s potential extends to providing students with immediate feedback as well as real-time assignment grading [55], [56]. Further it enables customized instruction and feedback for students depending on their learning needs and progress [5][45].

- 24/7 Availability and Accessibility

One of the key benefits of Gemini is its 24/7 availability. Unlike human teaching assistants who have set office hours, Gemini can offer support to students at any time, making it a convenient resource for both in-person and online learning environments [56].

2.5.3 Challenges

- Biased data

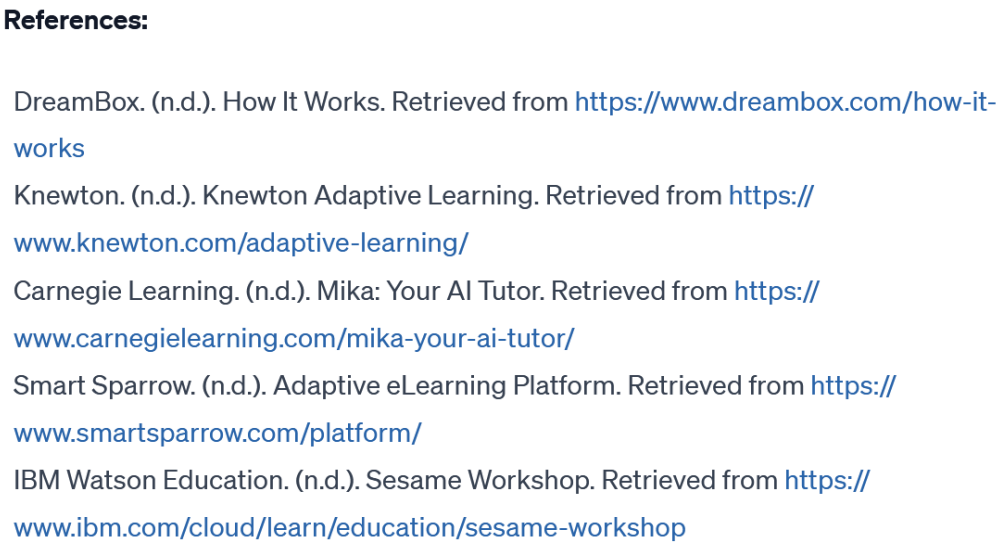
Concerns have been expressed about biases in GAI tools' responses; the biases that exist in training data and potential for chatbots to continue harmful biases and discrimination. [53][58].

- Reliability of generated contents

A notable drawback is its ability to generate erroneous, inaccurate information, such as inaccurate facts, non-existent article, a non-functional URL, quotes or citations. These inaccurate data generated by chatbots can affect the validity and reliability of the learning and teaching results. Therefore, it is recommended not to use as your only source for research [33][58], [59][51].

Below is the inaccurate, non-functional URLs provided by ChatGPT (ChatGPT 3.5 - January 04, 2024) related to AI applications in education

Figure 12 Inaccurate references



References:

1. DreamBox. (n.d.). How It Works. Retrieved from https://www.dreambox.com/how-it-works
2. Knewton. (n.d.). Knewton Adaptive Learning. Retrieved from https://www.knewton.com/adaptive-learning/
3. Carnegie Learning. (n.d.). Mika: Your AI Tutor. Retrieved from https://www.carnegielearning.com/mika-your-ai-tutor/
4. Smart Sparrow. (n.d.). Adaptive eLearning Platform. Retrieved from https://www.smartsparrow.com/platform/
5. IBM Watson Education. (n.d.). Sesame Workshop. Retrieved from https://www.ibm.com/cloud/learn/education/sesame-workshop

- Lack of contextual awareness and understanding the nature

A lack of contextual awareness and understand regarding the nuances nature of tasks is a common issue associated with AI systems. Generative models may provide improper or irrelevant responses due to the lack of context and situational understanding [45], [58].

- Academic integrity

Many articles focused on the academic integrity concerns such as cheating, academic dishonesty in AI tools usage. Some concerns that "students can now outsource their essay writing to the chatbot", "write college application essays" and even "pass MBA exams" and tests required for professional licenses in sectors such as medicine and business[6][9][51].

Current automatic detection algorithms for academic integrity violations are often unreliable, lacking definitive evidence for violations [60]. This unreliability could lead to mistrust between students and instructors, complicating the educational dynamic. Moreover, there is a growing need to revise academic integrity codes to clarify the acceptable use of GAI, ensuring that faculty can determine when its use is permissible or prohibited.

Depending on the use case, using AI technologies to fulfill academic activities may create ethical concerns about the legitimacy of the results.

- Plagiarism

Students may utilize GAI tools to create content for assignments without fully understanding or acknowledging the material's source [6] [61]. On the other hand these tools can avoid plagiarism detection programs like Turnitin by generating unique content [59]. Research suggests that students who used GAI tools were more likely to plagiarize compared to those who did not. This creates a significant challenge to academic legitimacy, validity and fair assessment of student learning [51], [59], [62], [63].

- Over-Reliance on Technology

Dependence on GAI tools may also foster a culture of technological dependency, making educational systems vulnerable to potential technical failures [64]. This over-reliance can diminish critical thinking skills among students and educators, as they risk becoming passive recipients of information rather than active participants in the learning process. Furthermore, the automation of problem-solving could stifle creativity and innovative thinking, undermining the fundamental objectives of education [60].

- Lack of human interaction

Gemini and related generative models may be less effective than real professors or tutors because to the lack of human connection. This emphasizes the importance of personal connection and emotional cues in the learning process, which may be hampered by the lack of true human interaction in AI-based teaching technologies [45].

- Lack of creativity

Generative models are limited in their creativity and originality as they can only produce responses based on training data patterns [45].

- Privacy

Researchers have expressed concerns about the privacy implications of using Gemini and other generative AI models[64][45]. While these models have impressive capabilities, it is crucial to remember that they cannot fully replace human teachers and tutors. Responsible use involves utilizing these tools together with human guidance and support, highlighting the importance of taking a balanced and ethical approach to integrating AI into educational environments [5].

CHAPTER 3

METHODOLOGY

The research methodology for this study is designed to comprehensively investigate the impact of GAI in the education system, emphasizing the development and evaluation of an interactive learning environment for students learning Python programming language. The methodology incorporates a mixed-methods approach, primarily focusing on qualitative data collection through surveys and literature review and with the development and validation of the novel interactive learning environment leveraging Gemini API.

3.1 Data Collection

3.1.1 Detailed Literature Review

This comprehensive literature review aims to investigate diverse impact of GAI on education. The primary objective is to obtain a comprehensive awareness of the current state of affairs, theoretical foundations and real-world implications of integrating GAI tools into the educational contexts.

The study includes a review of research papers, academic articles and relevant publications that provide insight into the many aspects of GAI deployment. By reviewing the existing studies, this study aims to gain insight into the potential benefits and challenges of GAI in educational context. Furthermore, the research review looks at the challenges related to GAI in educational settings.

By conducting this extensive literature review, the study hopes to not only identify current gaps and controversies, but also to synthesize a thorough knowledge of the impact of GAI on university education. The review results guide the development of interactive learning environment for learning programming languages.

3.1.2 User Testing and Feedback

The developed student learning support system will be deployed within a controlled environment and will conduct user testing involving learning enthusiasts. Collect the user feedback on the system's functionality, user interface and overall user experience.

The focus of these will be to collect feedback on:

- System Functionality: Assessing the effectiveness of interactive coding examples, assistance and concept explanations.
- User Interface: Evaluating the design for ease of navigation and engagement.

- Overall User Experience: Understanding the users' perception of the system's impact on their learning process.

3.1.3 Participant Selection

The developed system will be used by students, educators who are enthusiastic to learn programming. The diverse sample includes participants from various academic disciplines to capture a broad variety of responses. Thus, this approach ensures the collection of a diverse range of responses that reflects the varied perspectives of developed system.

3.1.4 Questionnaires Design and Distribution

In this study, questionnaires will be employed to collect data from end users of the developed system. The questionnaires will be developed to capture the experiences and perspectives of students concerning the usability, functionality and other perspective of the developed system. The surveys aim to gather qualitative and quantitative data including the effectiveness, appropriateness and broader implications for student engagement, learning outcomes and academic processes.

The reason for this study's focus on questionnaire format is that it is the most appropriate technique to gather data from a huge number of participants, as obtaining information from a large number of participants through interviews takes significantly longer. Numerous research has employed questionnaires as a primary means of data acquisition, benefiting from their expeditious data collection, cost-effectiveness and heightened objectivity compared to alternative methods. Despite these advantages, questionnaires also present drawbacks, such as respondents randomly choosing answer options without a full grasp of the question and the inability for researchers to convey more opinions about the issue due to the lack of a relevant question.

3.1.5 Data Analysis

Employ qualitative data analysis and statistical methods to analyse quantitative data such as ratings and survey responses, to derive meaningful insights and patterns.

According to the nature of the questions, questionnaires can be classified as quantitative or qualitative methods. Questions that are closed-ended and with multiple choice answer options are analysed quantitatively, using bar graphs, pie charts and percentages whereas open-ended questionnaire questions are analysed qualitatively, using discussions and critical analyses rather than numbers and calculations.

- Qualitative Data Analysis:

Utilize thematic analysis to identify recurring themes and patterns in qualitative data gathered from open-ended survey questions.

Conduct content analysis to categorize and interpret qualitative data based on predefined themes related to system usability, functionality and user experience.

- Statistical Methods for Quantitative Data:

Calculate statistics to summarize quantitative data from rating scales and Likert-type survey questions.

- Integration of Qualitative and Quantitative Insights:

Use qualitative and quantitative findings to provide a comprehensive understanding of user perspectives and system performance. Use data visualization techniques like charts, graphs and diagrams to present key findings and trends derived from both qualitative and quantitative analyses.

3.1.6 Validation and Verification

The developed interactive learning environment will be evaluated using the user feedbacks and predefined metrics, including user satisfaction, improvements in learning outcomes and system performance.

Furthermore, a comparative analysis will be undertaken, aligning the findings from the literature review with the data collected from the system evaluation. This process aims to verify the correspondence of the developed student learning support system with observed trends and challenges in the integration of GAI in educational settings.

3.2 Resources

3.2.1 Survey Tools

Online survey platforms facilitate the creation, distribution and collection of survey responses in a digital format. These tools offer user-friendly interfaces for designing surveys, managing participant responses and generating analytical reports.

Examples: Popular online survey tools include Google Forms, SurveyMonkey, Typeform and Qualtrics.

Among these tools most widely used Google Forms will be utilized to design and distribute surveys to the university students and the academic staff, efficiently collecting both qualitative and quantitative data on the impact of ChatGPT.

3.2.2 Development Tools

Programming and development tools are software applications that help create, test and manage the personalized support system for students to learn programming

languages. These technologies offer an integrated environment for developing, testing and delivering software solutions.

Examples: Depending on the programming language and platform chosen, tools like Visual Studio Code, PyCharm, Eclipse or Jupiter Notebooks may be employed.

These tools will be used to develop the interactive support system, incorporating features like interactive coding examples, debugging assistance and explanations of programming concepts.

3.2.3 Literature Databases

Academic databases, journals and repositories have a variety of scholarly articles, publications and theoretical frameworks related to the integration of ChatGPT in educational settings. Accessing these resources is crucial for conducting a thorough literature review.

Examples: Databases such as IEEE Xplore, Sci-hub and Google Scholar provide access to a vast array of academic publications.

Relevant literature will be investigated and collected to provide a solid theoretical foundation, identify existing gaps and synthesize knowledge about ChatGPT's impact on university education.

CHAPTER 4

SYSTEM DESIGN AND IMPLEMENTATION

4.1 Purpose and Goals

An interactive and personalized learning environment for students to learn Python programming language has been developed. This is a learning system developed with Generative AI to revolutionize students' learning experience of programming languages. This platform offers an individualized, interactive and optimized learning experience via the capability of Generative AI (GAI) specifically Gemini API.

- Personalized Education: Provide customizable learning experiences that tailored to the individual needs, preferences and skills of each student.
- Enhanced Engagement: To make learning programming more interactive and enjoyable through dynamic content, gamification and real-time feedback.
- Effective Learning: To improve learning outcomes by providing immediate support and comprehensive feedback.
- Accessibility: To make high-quality programming education accessible to a wider audience through a user-friendly, web-based platform.

4.2 System Features

This system will have interactive coding examples, debugging assistance and descriptions of programming concepts etc. The development process will follow learning objectives and user-centred design principles. Additionally, the effectiveness of the learning environment will be tested through user feedbacks.

The interactive learning environment aims to address key challenges identified in the integration of GAI into programming language education.

It will feature:

- Personalized Lesson Plans: Combined of AI-generated lessons based on user skill level, learning goals and preferences.
- Interactive Code Editor: Real-time code execution and AI-driven code suggestions.
- Gamification: Challenges to motivate and engage learners.
- User Dashboard: Progress tracking, performance analytics and personalized recommendations.
- Interactive coding examples: Providing hands-on coding exercises to reinforce theoretical concepts.

- Real-time guidance: Offering guidance during coding sessions to help students identify and rectify errors.
- Explanations of programming concepts: Providing detailed explanations of programming principles to support conceptual understanding.

The system development will adhere to educational objectives, ensuring alignment with curriculum requirements and prioritize a user-friendly design for optimal usability.

4.3 System Functionalities

1. User Registration and Onboarding
    - User Registration: Users can register with basic details (e.g., email, password).
    - Onboarding Survey: Collects user data to personalize the learning experience:
        - Nickname: A friendly name for the user.
        - Skill Level: Beginner, Intermediate, or Educator (If the user is unsure, the system provides a Python questionnaire to assess their skill level).
        - Purpose of Learning:
            - For Education
            - For Fun
            - Upskilling in Tech
            - Career Switch
            - Other.
        - Coding Knowledge: None, a bit, quite a bit.
        - Priority/Preference: Users can select topics/areas of interest (e.g., data analysis, web development, automation).
2. Personalized Lesson Plan Creation
    - Lesson Plans:
        - The system creates lessons based on collected onboarding data.
        - Lessons are pulled from a predefined content but AI customizes
        - Use AI to:
            - Explain concepts in different ways based on user preferences.
            - Generate additional examples or exercises if the user needs more practice.

- Adaptive Learning:
    - User can get further explanations on where the user needs more clarification.
    - Provide alternative explanations for difficult concepts using AI.
    - Generate additional examples upon request.

3. Interactive Learning Environment
    - Interactive Code Editor: The system includes an embedded interactive code editor with:
        - Real-time execution
        - AI provides suggestions and sample solutions.
4. User Dashboard
    - Welcome Message: Greeting (e.g., "Welcome back, Gaya!").
    - Progress Tracking:
        - Tracks completed lessons and quizzes.
        - Displays a progress bar.
    - Motivational Tips: AI-generated tips to keep users motivated based on progress. (eg: Keep going! You're just 3 lessons away from mastering Python Basics.)
    - Next Lesson: The next lesson based on progress.
    - Quiz Performance: Shows average score and highlights areas for improvement.
    - Gamification:
        - Displays a daily challenge to keep users engaged.
        - It is a new coding challenge generated by AI every day. (eg: Write a program to reverse a string.)
        - Provides sample solutions.
5. End-of-Lesson Feedback
    - Questionnaire: At the end of each lesson, users complete a quiz to assess understanding.
        - Provides detailed feedback on quiz
        -

4.4 Functional Requirements

Table 1 Functional Requirements

ID	Requirement	Description
FR-1	User Registration	The system shall allow users to register with basic details (e.g., email, password).
FR-2	Onboarding Survey	The system shall collect user data (e.g., skill level, purpose of learning, coding knowledge, preferences) during onboarding.
FR-3	Skill Level Assessment	The system shall determine the user's skill level through a Python questionnaire if unsure.
FR-4	Personalized Lesson Plan	The system shall create a personalized lessons based on user data, using predefined content and AI-generated explanations/examples.
FR-5	Interactive Code Editor	The system shall provide an interactive code editor with real-time execution.
FR-6	AI-Generated Feedback	The system shall provide real-time feedback on quizzes using AI.
FR-7	Dashboard	The system shall display a dashboard with progress tracking, motivational tips, next/previous lesson, daily challenge and sample solutions, quiz performance
FR-8	End-of-Lesson Questionnaire	The system shall provide a quiz at the end of each lesson to assess understanding.
FR-9	Feedback	The system shall provide feedback on quiz performance.

4.5 Non-Functional Requirements

Table 2 Non-Functional Requirements

ID	Requirement	Description
NFR-1	Usability	The system should have an intuitive and user-friendly interface suitable for beginners and educators.
NFR-2	Compatibility	The system should work on Windows, macOS and Linux operating systems and modern web browsers (e.g., Chrome, Firefox).
NFR-3	Ethical Considerations	The system should inform users when AI-generated content is provided.
NFR-4	Localization	The system should support English as the primary language.

4.6 Implementation Details

This section provides a detailed overview of the implementation process, including the technologies used, the system architecture and the integration of the Google Gemini AI model into the interactive learning environment.

4.6.1 Technology Stack

The interactive learning environment was developed using a modern and scalable technology stack to ensure high performance, flexibility and ease of maintenance. The key components of the technology stack are as follows:

- Frontend: The user interface was built using React, a popular JavaScript library for building dynamic and responsive web applications. React was chosen for its component-based architecture, which allows for reusable and modular code. This ensures a smooth and interactive user experience.
- Backend: The server-side logic was implemented using Node.js, a runtime environment that enables JavaScript to be used for server-side programming. Node.js was selected for its non-blocking I/O model, which ensures high performance and scalability, especially when handling multiple concurrent requests.
- Database: MongoDB, a NoSQL database, was used to store user data, learning progress and interaction logs. MongoDB was chosen for its flexibility in handling unstructured data and its ability to scale horizontally.

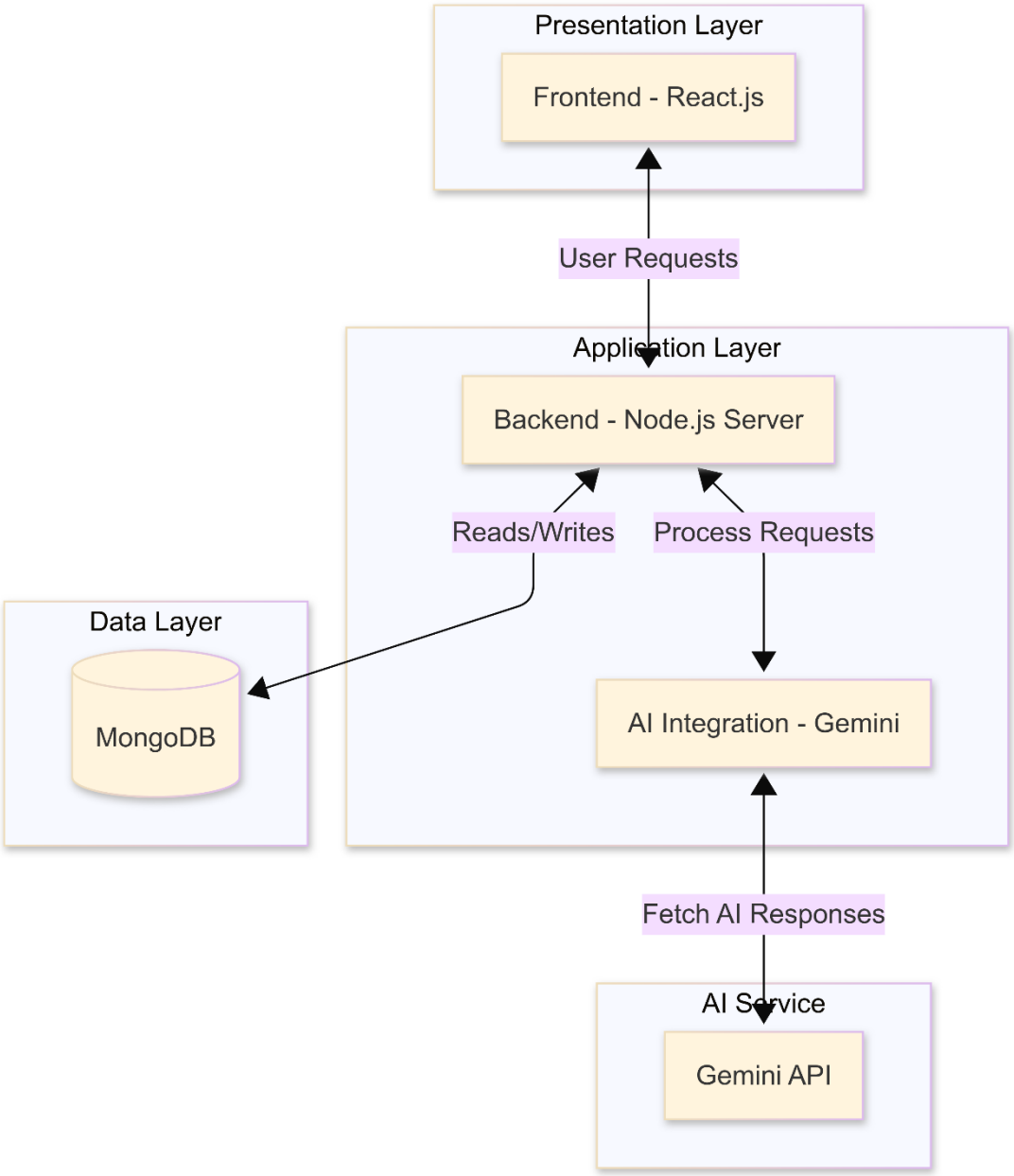
It also integrates seamlessly with Node.js, making it ideal for this application.

- Hosting: The application was hosted on the Microsoft Azure cloud platform. Azure provides a robust and scalable infrastructure for deploying web applications along with built-in tools for monitoring, security and management.

4.6.2 System Architecture

The system architecture follows a three-tier architecture, consisting of the presentation layer (frontend), application layer (backend) and data layer (database).

Figure 13 System Architecture Diagram



The above diagram (Figure 13) represents the three-tier architecture for the Interactive Learning Environment. It consists of four main components:

- Presentation Layer (Frontend - Client)
- Application Layer (Backend - Server)
- Data Layer (Database)
- AI Service (Gemini API Integration)

1. Presentation Layer (Client - Frontend)

Component:

Frontend (React.js): The user interface of the application, built using React.js. It handles the user interface and user interactions. It communicates with the backend via RESTful APIs to fetch data and perform operations. The frontend is designed to enhance the user experience and user friendliness.

It includes features such as:

- A code editor for interactive coding.
- A dashboard for tracking learning progress.
- Real-time feedback.

Interactions:

- Users interact with the frontend (UI) to read contents, practice code, view feedback and track progress.
- The UI sends user requests to the backend for processing via API calls.
- If real-time feedback is required, the UI establishes a connection with backend get responses through AI integration model.

2. Application Layer (Server - Backend)

The backend server implementation using Node.js. This is act as a middleware between the frontend and the database. The business logics are implemented in this layer. It handles user requests and communicates with Gemini AI integration model.

Components:

- Backend (Node.js Server): The server that processes user requests.
- API Endpoints: RESTful APIs that allow the frontend to communicate with the backend.
- AI Integration Module: Connects the backend with Gemini AI for intelligent feedback.

Interactions:

- The backend processes requests received from the frontend.

- API Endpoints handle user authentication, course progress and learning module interactions.
- If AI-powered suggestions are required (such as code reviews or explanations), the backend calls the AI Integration module.

3. Data Layer (Database - MongoDB)

MongoDB stores all persistent data, including user profiles, learning modules, progress data, etc.

Component:

- MongoDB Database: Stores all user data, including:
  - User profiles
  - Course content
  - Learning progress
  - Logs and analytics

Interactions:

- The backend reads/writes data to the database based on user actions.
- User progress is saved in MongoDB to track learning history.

4. AI Service (Gemini API Integration)

Gemini API handles all the AI powered requests.

Component:

- Gemini API: Provides AI-powered feedback, code suggestions and personalized explanations.

Interactions:

- The AI Integration module in the backend sends queries to Gemini API when AI-driven feedbacks are required.
- The Gemini API returns suggestions, explanations etc which are sent back to the frontend.

4.6.3 Generative AI Integration with Google Gemini

The core functionality of the interactive learning environment is powered by Google Gemini, a state-of-the-art Generative AI model. Gemini provides real-time feedback, code suggestions and explanations to learners, making it an essential component of the system. The integration process involved the following steps:

1. Model Selection

Google Gemini was selected as the AI model due to its advanced natural language processing (NLP) capabilities and its ability to generate high-quality, context-

aware responses. Gemini is particularly well-suited for educational applications because of its ability to understand and explain complex programming concepts in a beginner-friendly manner.

2. API Integration

Gemini was integrated into the backend using its REST API. When a user submits a query or code snippet, the backend sends the input to the Gemini API and receives the generated response.

The API was configured to handle various types of requests, such as:

- Code Explanation: Providing detailed explanations of Python code snippets.
- Concept Explanation: Providing detailed explanations of Python concepts.
- Generate Questions: Generating questions based on the progress.
- Code Suggestions: Offering optimized or alternative solutions to programming problems.
- Generate motivational message: Generating motivational message based on the progress.

The API responses were formatted as JSON objects, making it easy for the backend to parse and process them.

3. Real-Time Interaction

The frontend displays the AI-generated response in real-time, allowing learners to interact with the system seamlessly. For instance, if a learner submits a code snippet with errors, Gemini provides suggestions for fixing the errors and if s/he needs further clarifications it explains the underlying concepts further.

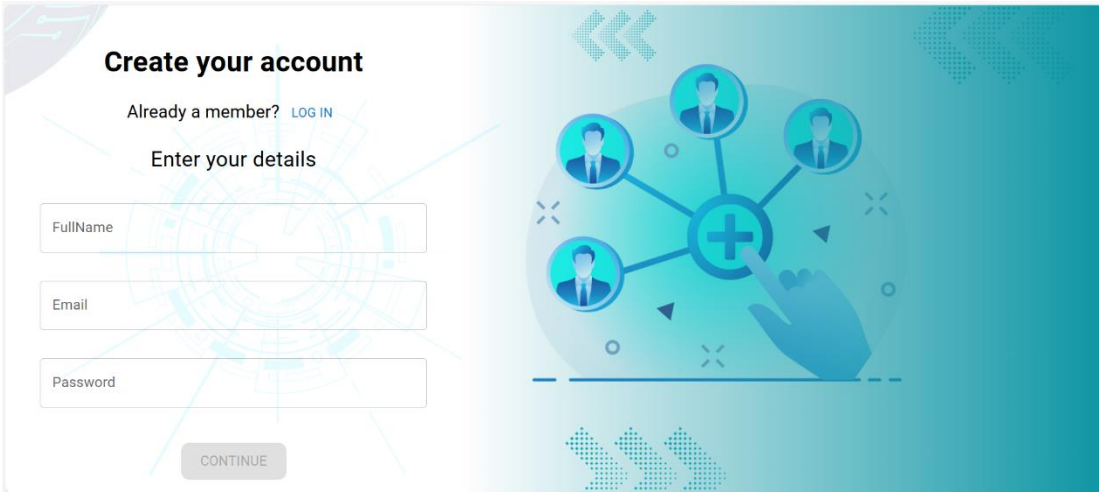
The system also includes features like interactive code execution, where learners can run their code directly in the browser and receive instant feedback from Gemini.

4.6.4 User Interfaces

User Registration

Users can create their own account by entering a valid email and a password.

Figure 14 User registration UI 1



Skill level should be selected. If the user unsure about it, user can take an assessment to measure the skill level.

Figure 15 User registration UI 2

Select purpose of learning to have better experience

Figure 16 User registration UI 3

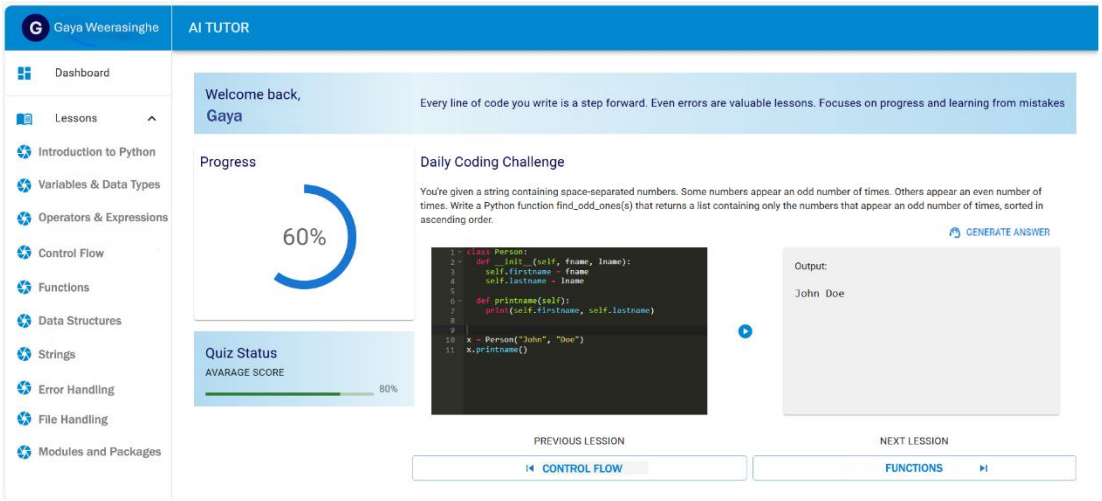
Select what you prefer to study.

Figure 17 User registration UI 4

Sample User Dashboard which shows; lesson plan, progress, motivational quote, the daily challenge, quiz status etc.

Generate Answer option will provide user a sample answer for the daily challenge.

Figure 18 Sample user dashboard



Every lesson is structured in six main parts.

1. Prior Requirements – Indicate prerequisites that need to be met before starting the relevant lesson.
2. Learning Outcomes – What are the expected outcomes by the end of each lesson.
3. Content – Content based on user level.
4. Recap – Summary of each lesson.
5. Practice Exercise – Exercise based on each lesson.
6. What's Next – What you will learn in next lesson.

Figure 19 Lesson - Prior requirements and learning outcomes

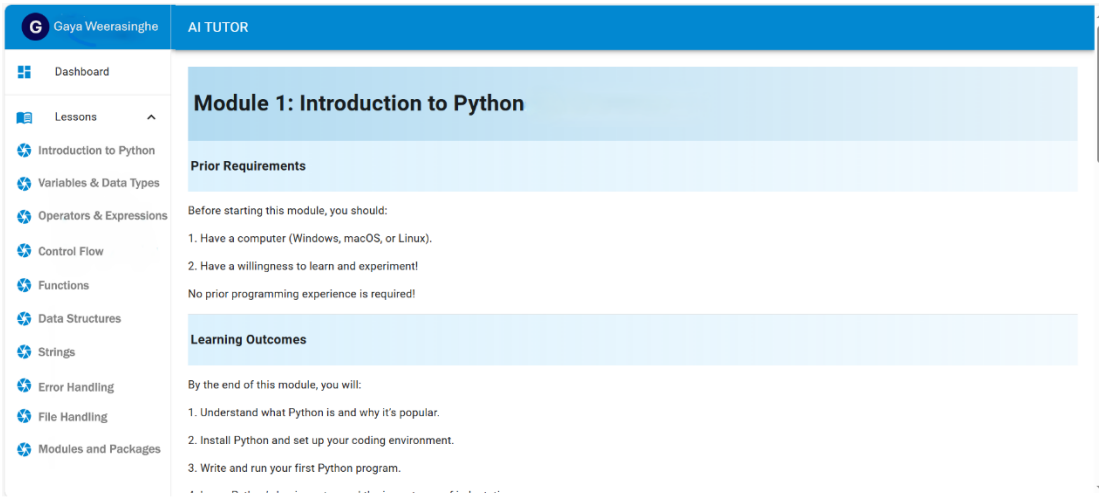
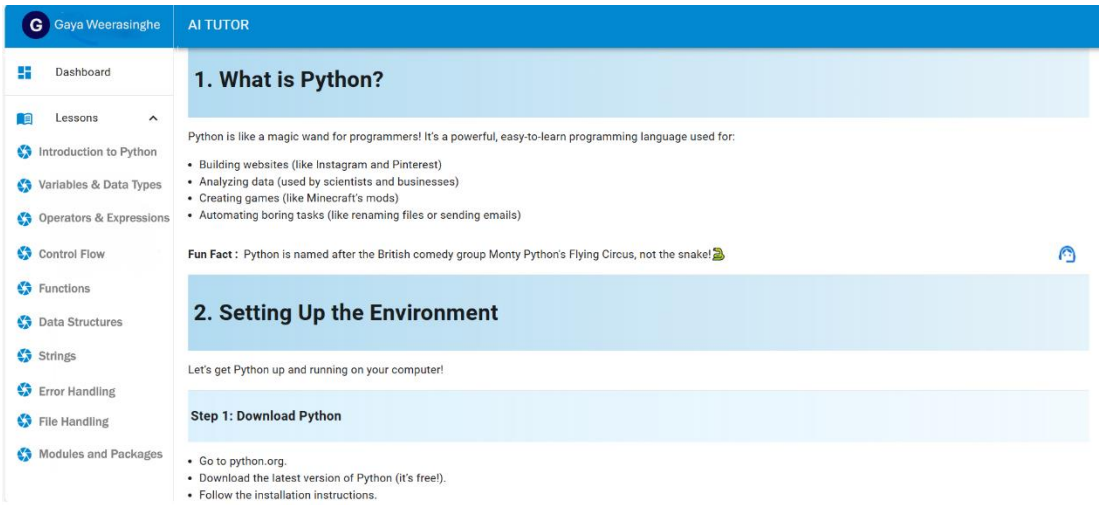
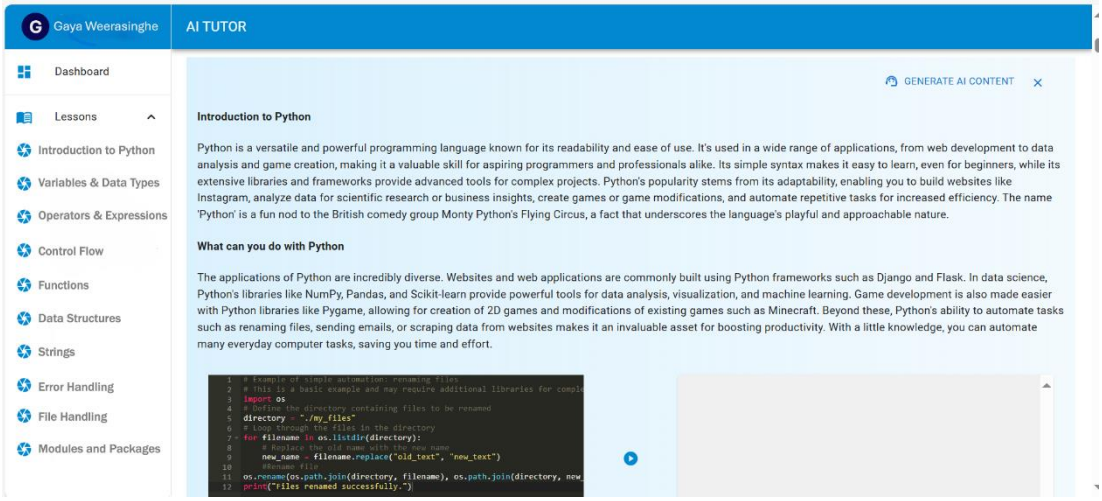


Figure 20 Lesson for a beginner



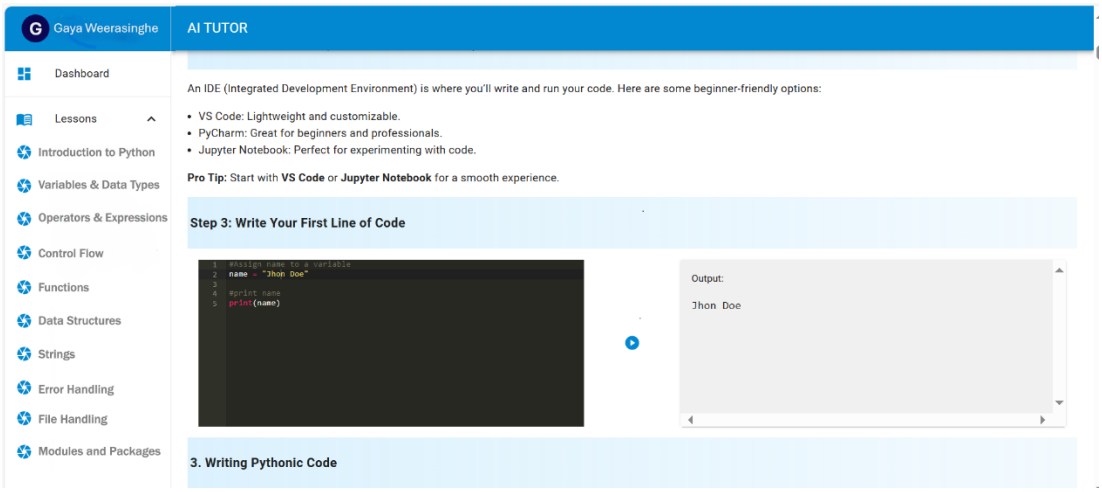
When a user clicks the 'bot' icon in any section, it delivers tailored, in-depth explanations specifically designed to meet their needs. Figure 21 shows sample explanation content.

Figure 21 Further explanations using GAI



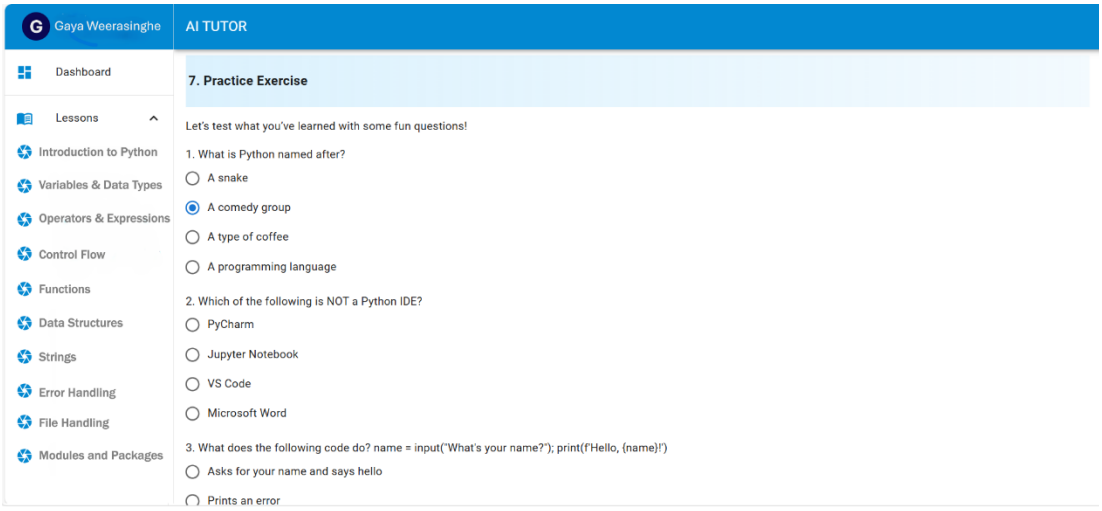
Example codes can be executed in real time, allowing users to instantly see results and interact with the functionality, enhancing their understanding and engagement.

Figure 22 Real time execution



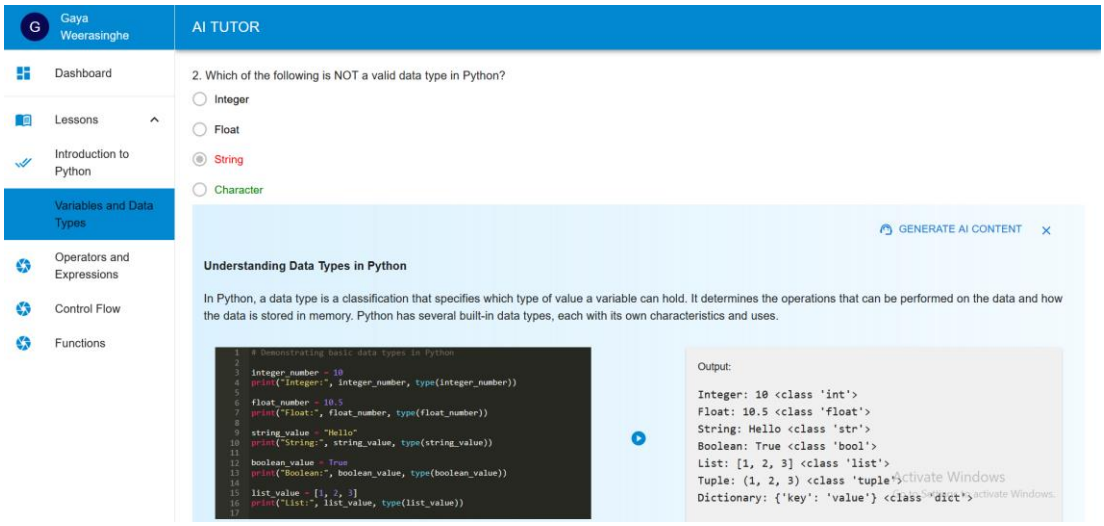
At the end of each lesson, there is a practice exercise (Figure 23) designed to reinforce the concepts covered and help users apply their knowledge effectively.

Figure 23 Practice exercise



After completing a practice exercise, you can get explanations for each question.

Figure 24 Practice exercise explanations



4.6.5 Deployment for Testing

The system includes two major components: a Node.js backend API server and a React.js frontend application. These components were deployed to Microsoft Azure using the Azure App Service for web application hosting.

The deployment was carried out in two phases:

1. Backend Deployment:
   Hosting the Node.js RESTful API on Azure App Service (Linux). The Node.js backend includes routes for authentication, topic handling, user operations, dashboards and a code execution API. It uses Express.js and connects to MongoDB via Mongoose.

2. Frontend Deployment:
   Hosting the React.js frontend (build folder) as a web application on a separate Azure App Service instance.

CHAPTER 5

RESULT AND EVALUATION

5.1 User-Centred Evaluation

The data gathered with the help of the survey tool (Google Forms) that was used to measure the interactive learning environment for Python programming is considered in this chapter. The results are outlined in various categories depending on goals and requirements of the system. Both quantitative and qualitative data have been used to evaluate the usability, functionality, engagement, satisfaction and comparative value of the system.

5.1.1 Participant Demographics

The survey collected responses from 8 participants, with the following demographic distribution:

1. Age Group

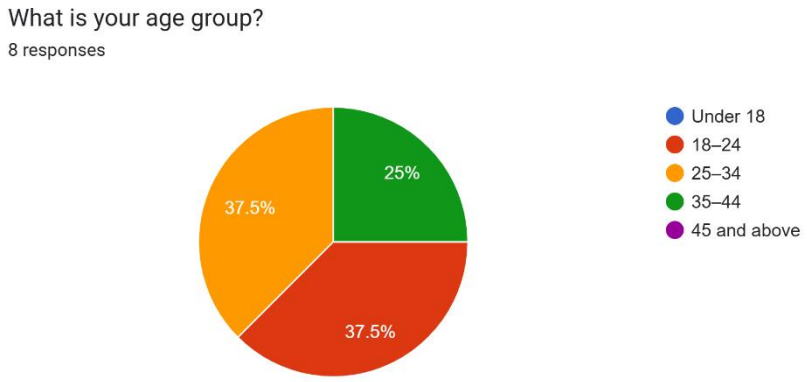


Figure 25 Age group pie chart

Most respondents (75%) were between the ages of 18 and 34, indicating a target audience primarily composed of university students or early-career professionals.

2. Educational Background

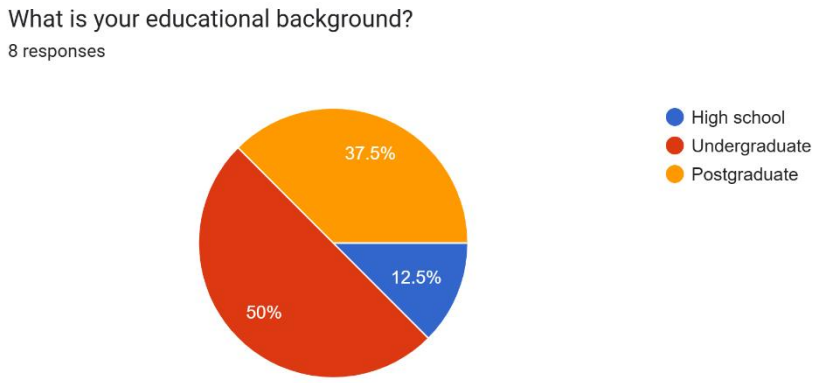


Figure 26 Educational background pie chart

Most respondents held undergraduate or postgraduate qualifications. This indicates that the tested users have a formal education background and may be comfortable using educational technology platforms.

3. User Role

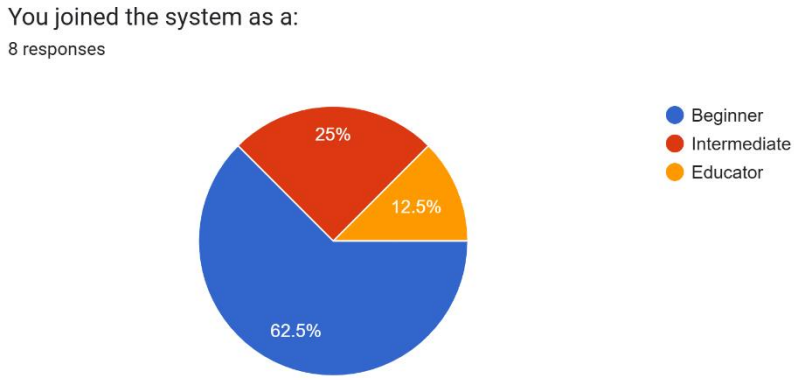


Figure 27 User role pie chart

Respondents joined the system as Beginners (63%), Intermediates (25%) and Educators (13%). This spread supports the evaluation of the platform from multiple user perspectives.

4. Programming Experience

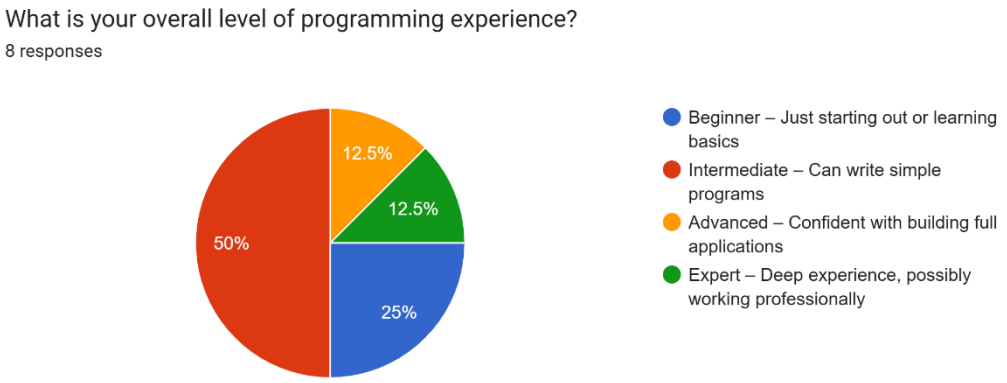


Figure 28 Programming experience pie chart

While many users self-identified as beginners or intermediates, some responses highlighted discrepancies between their joined role and actual experience. This is may be because they were not familiar with Python programming language even though they have programming experience. This reinforces the need for skill assessments during onboarding.

5.1.2 Usability of the System

Usability was measured through two metrics: ease of navigation and interface clarity.

1. Ease of Navigation

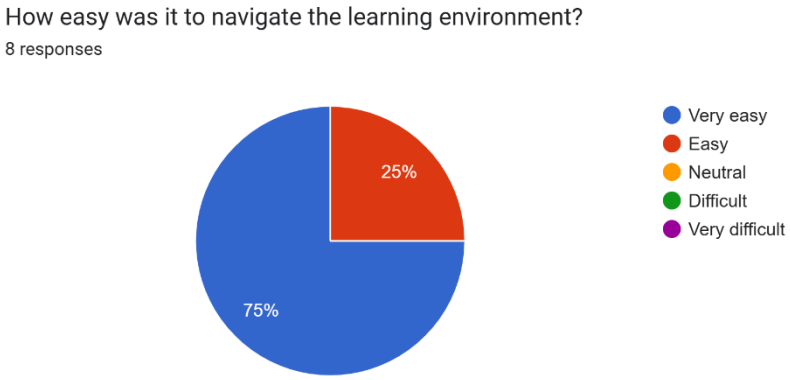


Figure 29 Ease of navigation pie chart

75% of respondents found the system "Very easy" to navigate, while the remaining 25% selected "Easy." No users found it difficult to navigate the system which demonstrates the user-friendly interface.

2. **User Interface (UI) Clarity**

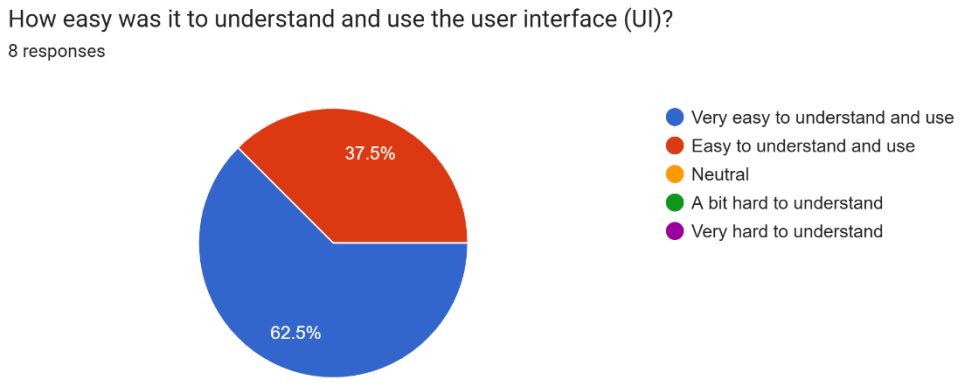


Figure 30 User interface clarity pie chart

63% found the interface "Very easy to understand and use" while 38% selected "Easy to understand and use"

These results confirm the system's adherence to its non-functional requirement NFR-1 (Usability).

5.1.3 System Functionality

Respondents were asked to evaluate feedback effectiveness, code suggestions, explanation accuracy and adaptability to learning style:

1. **Personalized Feedback**

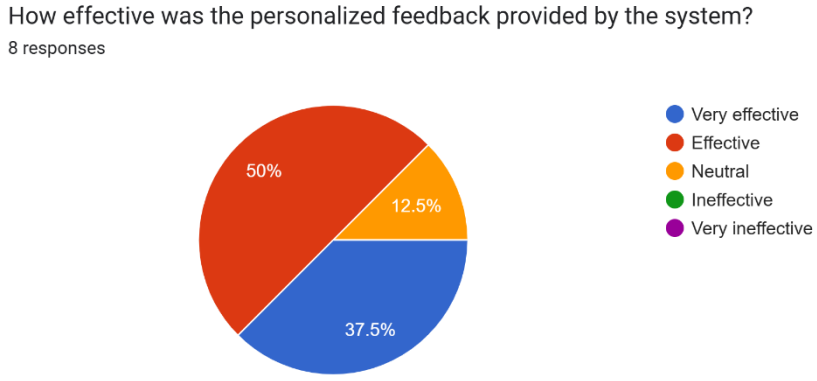


Figure 31 Personalized feedback pie chart

Most users rated the system as "Effective" or "Very effective" in delivering real-time feedback. 88% of respondents mentioned it was effective while 13% were neutral.

2. Code Suggestions and Hints

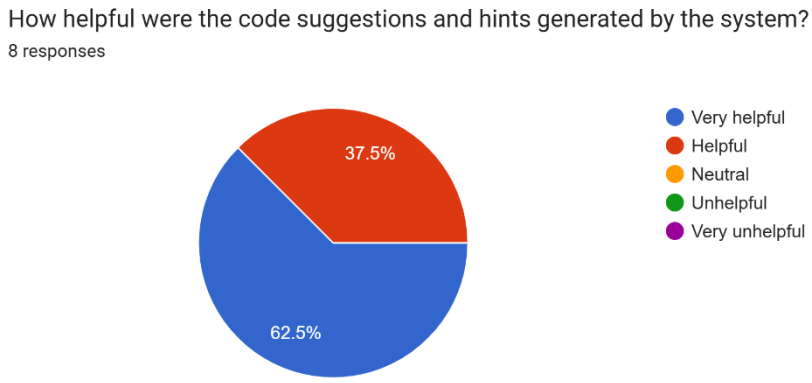


Figure 32 Code suggestions and hints pie chart

All participants found them either "Helpful" or "Very helpful" showcasing the effectiveness of AI-driven suggestions.

3. Explanation Accuracy

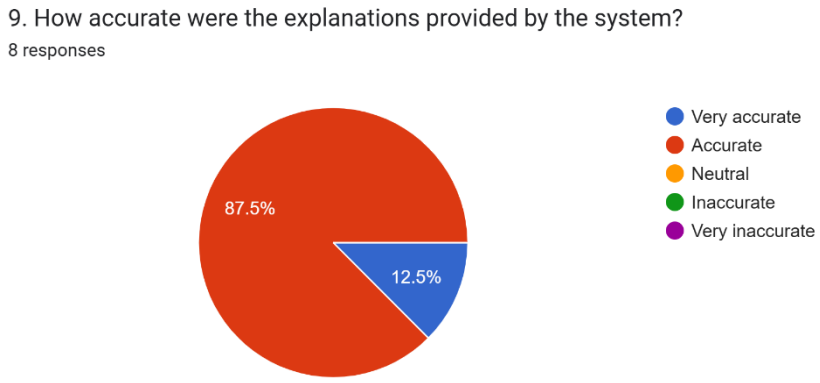


Figure 33 Accuracy pie chart

All respondents marked explanations as "Accurate" or "Very accurate" indicating that the AI-generated content was reliable and clear.

4. **Learning Pace Adaptability**

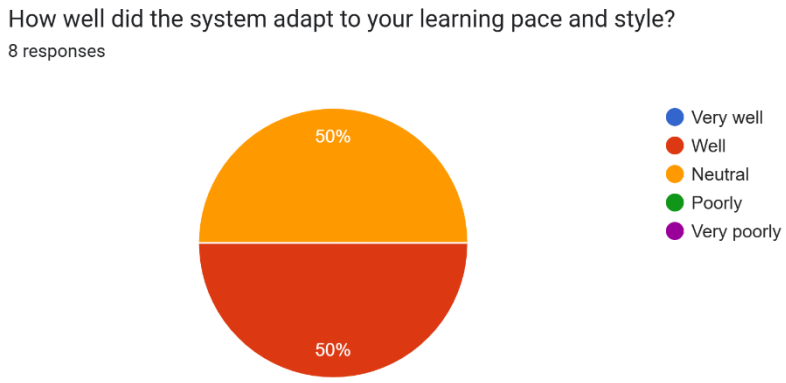


Figure 34 Learning pace adaptability pie chart

50% of users indicate the system well adapted to their learning style. While others indicated the system adapted "Neutral" to their individual learning styles.

5.1.4 Engagement and Motivation

The system's engagement level was a key aspect evaluated:

1. **Engagement**

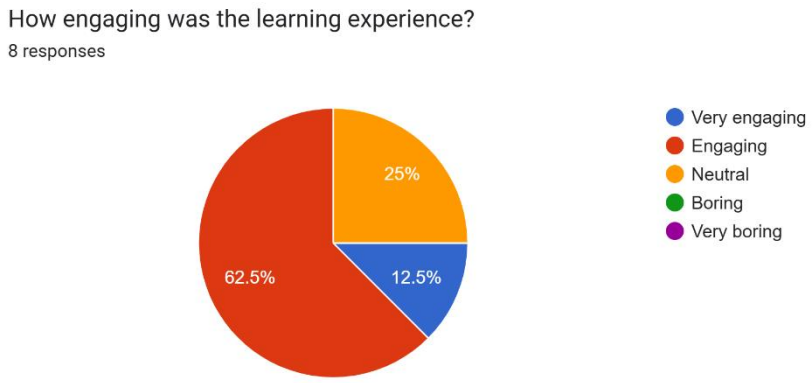


Figure 35 Engagement pie chart

75% of respondents found the platform to be "Very engaging" or "Engaging" while 25% of respondents were neutral about it.

2. Motivation

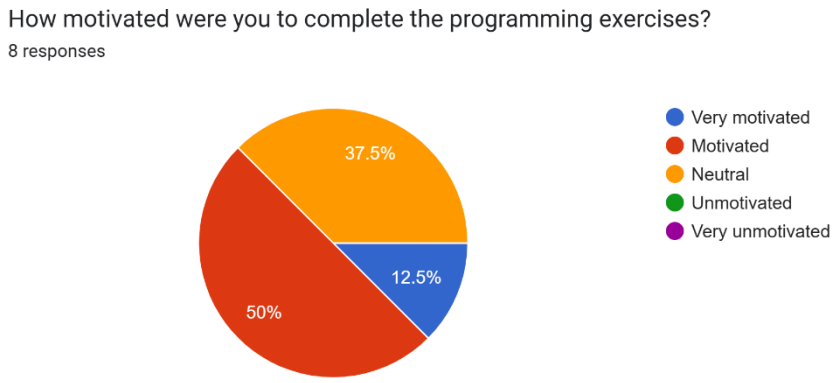


Figure 36 Motivation pie chart

13% of respondents recorded being “Very motivated” and half of the respondents reported being "Motivated" while the 38% selected "Neutral" regarding their motivation to complete the exercises.

3. Enjoyability

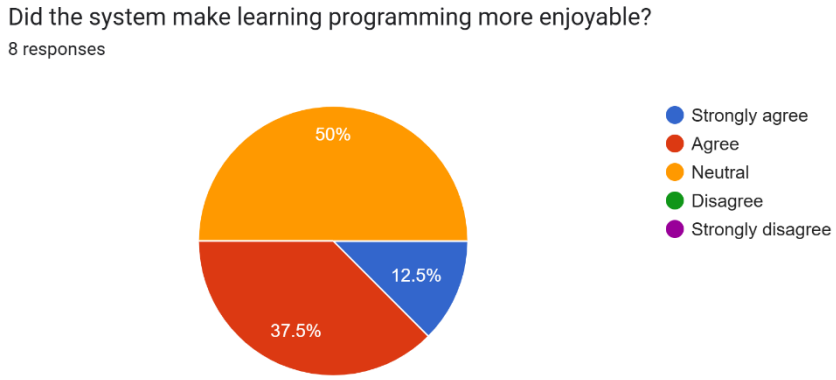


Figure 37 Enjoyability pie chart

Responses have a 50% for “Neutral” on whether the system made learning programming more enjoyable and others “Strongly agree” or “Agree” with the statement.

The daily challenge and motivational tips features were frequently mentioned in open-ended responses as major contributors to engagement.

5.1.5 Learning Resource Comparison

Users were asked about their typical external resources and how the system compares:

1. Common Learning Sources

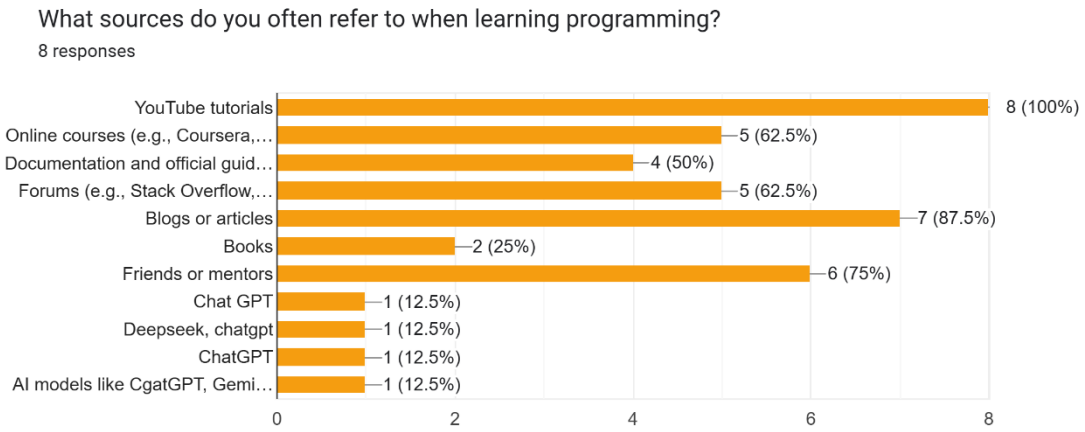


Figure 38 Common learning sources bar chart

YouTube tutorials, Blogs or articles, online courses, documentation and forums like Stack Overflow were the most cited resources. Other than those resources most respondents use AI models such as ChatGPT, Gemini or Deepseek.

2. Comparison to Other Sources

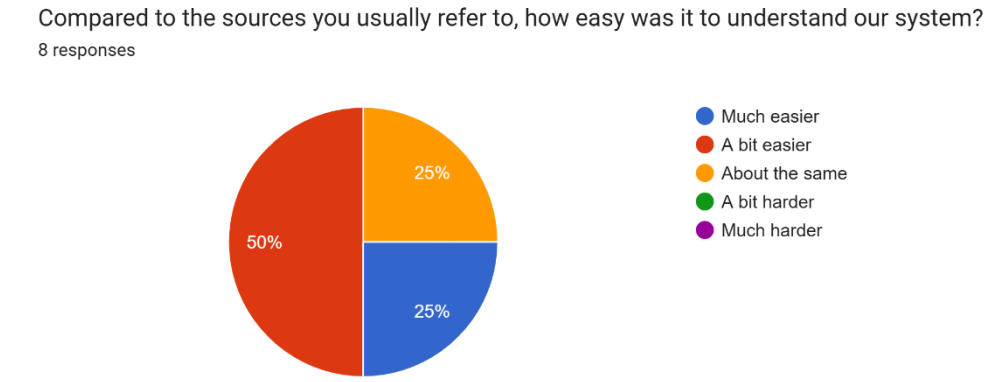


Figure 39 Comparison with other sources pie chart

75% of users reported that the system was "Much easier" or "A bit easier" to understand compared to external sources and none of the respondents indicate it as hard to understand.

3. External Resource Usage

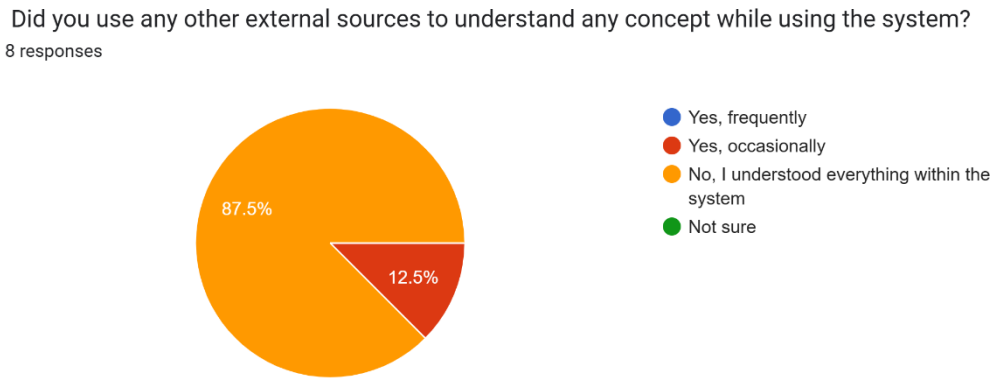


Figure 40 External resource usage pie chart

Most respondents (88%) did not use any external sources while using the system, implying it was largely self-sufficient and one respondent uses video tutorials while referring the system.

5.1.6 Overall Satisfaction

1. Satisfaction

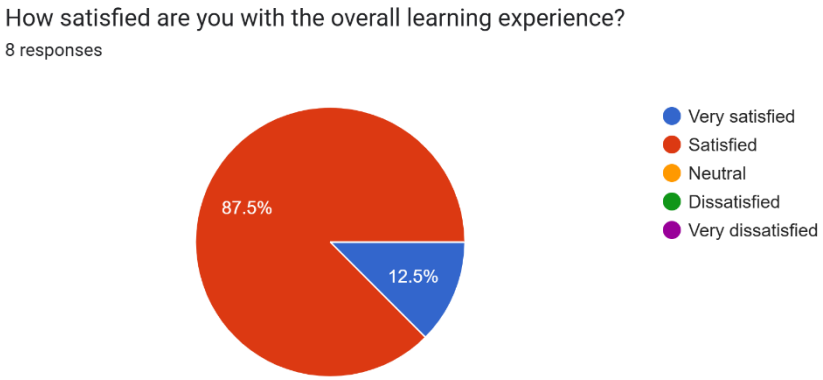


Figure 41 Satisfaction pie chart

All users rated their experience as “Very satisfied” or "Satisfied”. This is evident that the capability of the system to meet the minimum requirements of its users.

2. Recommendation Likelihood

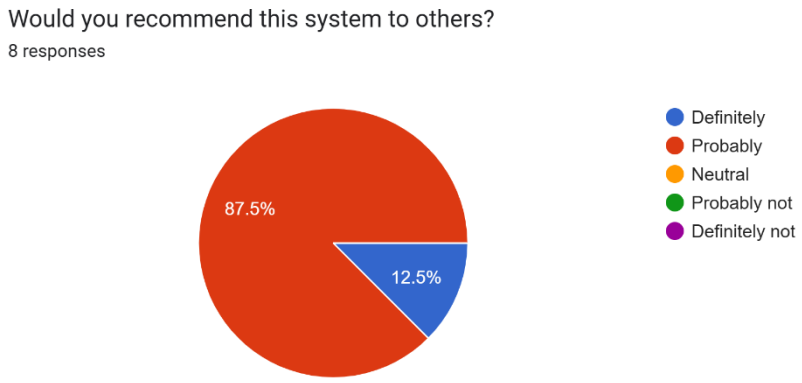


Figure 42 Recommendation likelihood pie chart

All participants indicated they would “Definitely” or "Probably" recommend the platform to others which proves the high level of satisfactory of the participants.

3. Open-Ended Feedback Summary

The open-ended responses collected via the evaluation survey provided valuable qualitative data on user sentiments in relation to the interactive learning environment. These were grouped under three main topics: most appreciated features, areas of dissatisfaction and areas for improvement.

- **Most Appreciated Features**

Figure 43 Thematic analysis for the most appreciated features

Participants were satisfied with most aspects of the system that enhanced their learning experience. Consistent appreciation was for contents being straightforward and uncomplicated, with users commenting that explanations were clear and lacked the need for extra resources to learn basic concepts. *"Also the explanations on quizzes were really helpful", "The content was simple and easy to understand", "getting further clarification from AI was helpful, so I did not want to refer any additional resource"*

Various users found the everyday coding problem and provision of sample solutions to be motivational and interactive. *"I like the daily challenge and the sample solutions provided"*. Another widely appreciated aspect was the immediate access to quiz explanations and personalized teaching tips, which allowed users to identify mistakes and improve their understanding in real time.

Moreover, utilization of the Generative AI by the system in explaining and justifying was also appreciated. One user explicitly mentioned that the AI-driven feedback and additional clarifications reduced their dependence on external learning platforms.

- **Least Appreciated Features**

Figure 44 Thematic analysis for the least appreciated features

Though overall feedback was positive, a number of participants also noted several limitations. Some users indicated that they would like to see more advanced programming modules introduced - *"Should include more advanced topics"* - and that the system might now be better suited to beginners and intermediate learners.

The code editor was another section that came under criticism. Particular criticisms were that the editor could not accept user input while executing and was not very flexible in managing interactive programs. *"All good, but code editor should enhance", "The code editor is not handling taking user inputs".* Some users criticized the editor as being restrictive and suggested improvements to be able to suit more realistic coding exercises.

One significant issue was the lack of real human support or in-app conversation such as a forum ("*There was no option to ask questions or get human support when stuck*"), which left some users constrained in how they could work through tougher questions or get support beyond what the AI could offer.

- **Suggested Improvements**

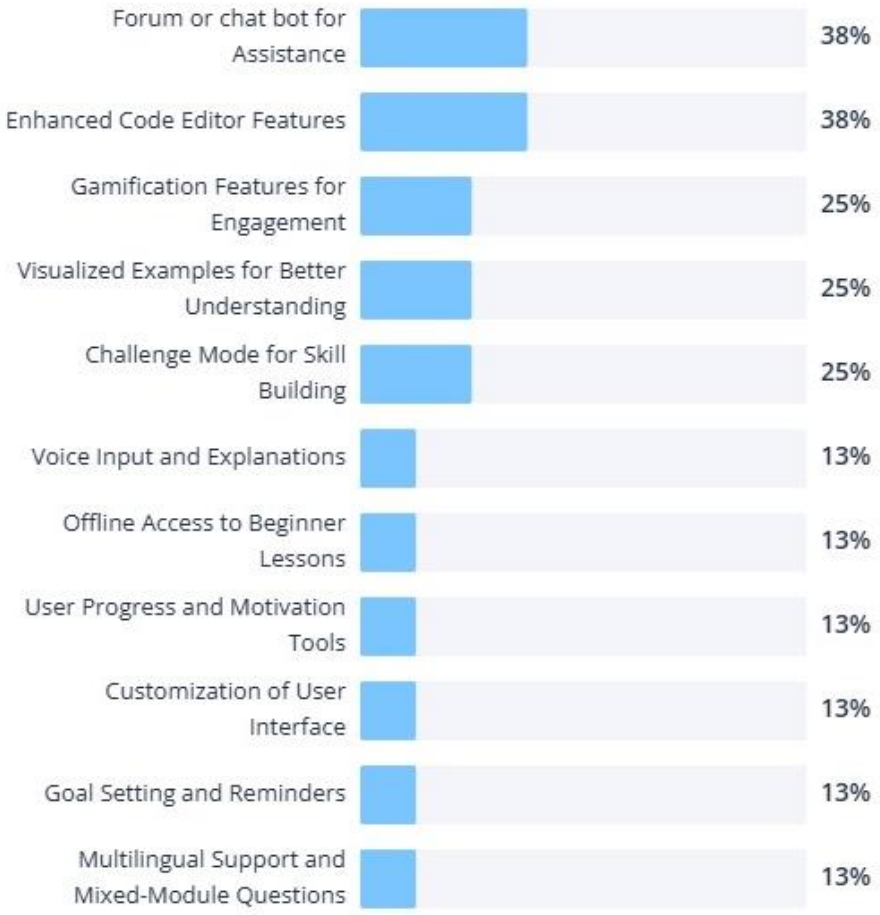


Figure 45 Thematic analysis for the suggested features

Members provided valuable suggestions for improving the learning system. The most frequent recommendation was adding a discussion forum or chatbot, under which learners can ask questions and be answered by peers or experts. *"have a chatbot, so we can ask questions as well."*, *"Include a forum to get human support when stuck".* This would be another feature on top of the AI support and make learning more efficient.

Other recommendations were:

- Improving the code editor, especially support for user input and interactive coding exercise.
- Adding more content to cover advanced programming concepts and adding more graphical learning tools like diagrams, videos and animations.
- Add more modules on popular programming languages and expand the curriculum
- Addition of gamification badges and a tracking mechanism to track daily challenges so that the user can track his/her performance and receive feedback on specific coding exercises.
- Support multi user languages such as Sinhala language.
- Facilitate to get mixed exercises from all completed modules.
- Add a special mode where users complete small challenges under time limits. This would help them think faster and build confidence.
- Notifications/ reminders to complete the lessons

These suggestions imply adding not just more technical elements to the system but also motivational and social elements to make users continue using the platform for extended periods.

5.2 System Evaluation

Since the system was deployed in the Azure environment, azure tools were used to evaluate system performance. This section presents a performance evaluation of the deployed system using Azure Load Testing. The objective is to assess how well the application handles concurrent user requests in terms of responsiveness, reliability and scalability.

5.2.1 Load Test Setup

To evaluate the backend performance and scalability of the developed system, two Azure Load Tests were conducted under simulated real-world conditions. The first test was executed prior to system optimizations, while the second test was conducted after key improvements were applied to enhance performance.

Each test duration is 10 minutes, simulating 50 concurrent virtual users, repeatedly executing four key backend API requests:

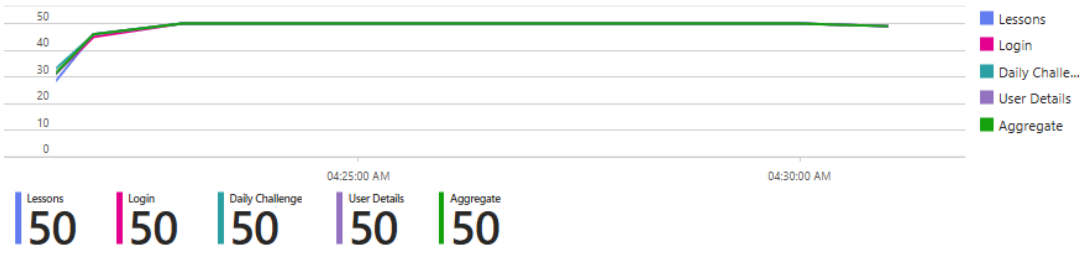
- Login – User login, returns authorization JWL token
- User Details – Get user object
- Daily Challenge – To get daily challenge with a sample answer
- Lessons – Get the lesson plan

5.2.2 Performance Metrics and Interpretation

1. Load Test 1

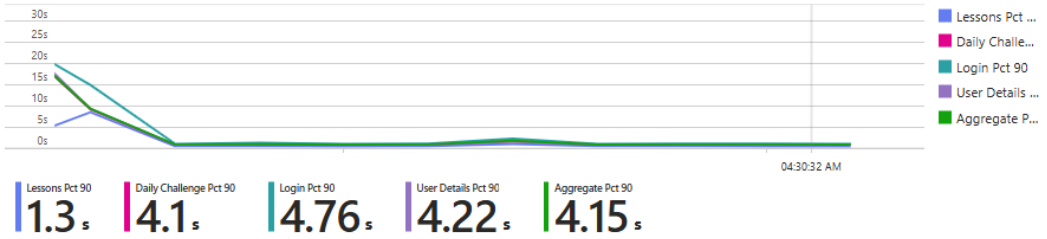
This was conducted using Azure Load Testing with 50 virtual users targeting four key API endpoints over a period of 10 mins 7 secs. A total of 39,746 requests were successfully sent and 0.05% error rate was recorded.

Figure 46 Virtual users (max)



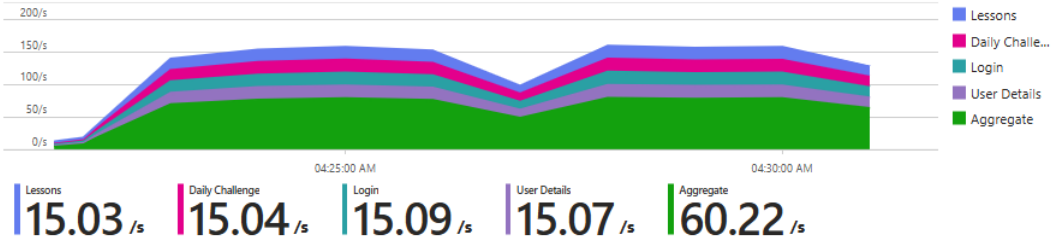
The test reached a peak of 50 concurrent virtual users across all endpoints. This level of concurrency simulates realistic traffic in an educational context, such as multiple students accessing lessons and code challenges simultaneously. The graph indicates that the system consistently supported the full user load without degradation or forced scaling.

Figure 47 Response time (Successful responses)



This means 90% of successful requests were served within the above times. While most endpoints perform under 5 seconds, the Login endpoint shows room for optimization under high concurrency.

Figure 48 Requests per second (Avg)



The platform maintained an average throughput of 60.22 requests per second, evenly distributed across endpoints.

Figure 49 Errors

While overall performance remained stable, a total of 21 errors (0.05%) were recorded. The Login endpoint contributes most to the total 21 errors, which corresponds with its higher latency. This identifies the Login functionality as a bottleneck requiring backend optimization or infrastructure scaling.

Table 3 Load Test 1 - Performance Summary

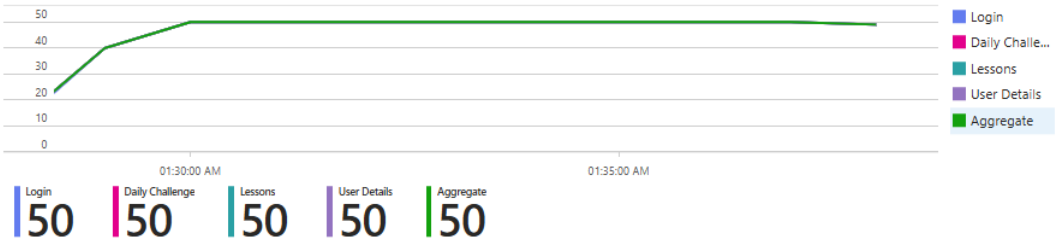
Sampler Name	Requests	Errors Count	Error %	Response Time (ms)	Throughput[3] (req/s)
Daily Challenge	9,928	1	0.01%	870	16.36
Lessons	9,917	0	0.00%	496	16.34
Login	9,958	19	0.19%	1160	16.41
User Details	9,943	1	0.01%	860	16.38
Total	**39,746**	**21**	**0.05%**	**940**	**65.48**

2. Load Test 2

Following this, several optimizations were applied to the backend system, including improved API response handling and error mitigation strategies. The post-optimization load test (Load Test 2) was run under the same conditions: with 50 virtual users targeting the four key API endpoints over a period of 10 mins, 8 secs. A total of 44,695 requests were successfully sent, with the error rate dropped to 0% across all endpoints, indicating the enhanced stability and reliability.

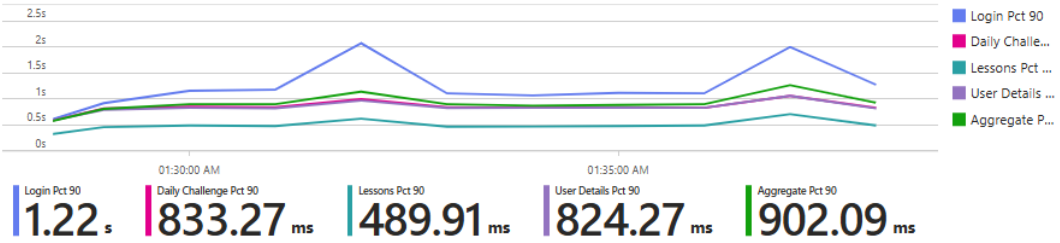
[3] A minor difference in throughput observed between the graph and the table summary is due to differences in sampling intervals vs. overall average calculation. This is normal in load testing environments and not considered a discrepancy.

Figure 50 Virtual users (max)



The test also reached a peak of 50 concurrent virtual users across all endpoints.

Figure 51 Response time (Successful responses)

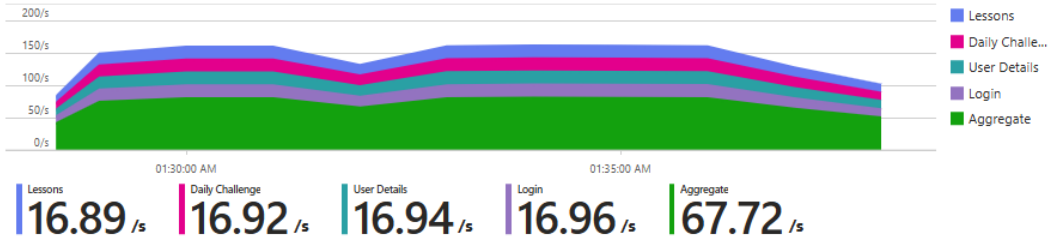


The 90th percentile response time across all endpoints was 920 ms, meaning 90% of requests completed in under 1 second, which aligns well with typical web application performance benchmarks.

- All response times remained under 1.5 seconds, which is well within acceptable UX limits.
- The Login endpoint exhibited the highest latency (1.22 s), likely due to authentication and token validation.
- The Lessons endpoint responded fastest, with sub-500 ms performance.

These results demonstrate that the backend logic is optimized for content delivery and code-related operations, with acceptable delay in authentication.

Figure 52 Requests per second (Avg)



The platform maintained an average throughput of 67.72 requests per second, evenly distributed across endpoints.

The stability and consistency in request rates imply that the backend handles load predictably, without spikes or bottlenecks.

Response time aggregation 90th percentile

Table 4 Load Test 2 – Performance Summary

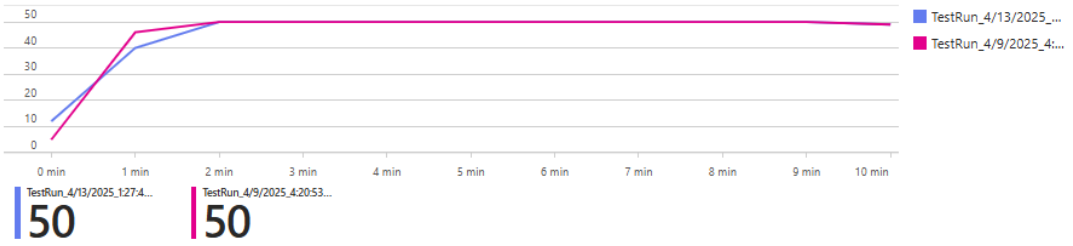
Sampler Name	Requests	Errors Count	Error Percentage	Response Time (ms)	Throughput[4] (req/sec)
Login	11,194	0	0%	1,198	18.41
User Details	11,183	0	0%	841	18.39
Daily Challenge	11,168	0	0%	855	18.37
Lessons	11,150	0	0%	499	18.34
Total	**44,695**	**0**	**0%**	**920**	**73.51**

The overall throughput was 73.51 requests per second, evenly distributed across endpoints (approx. 18 req/sec each), showing consistent server-side handling and load balancing.

5.2.3 Comparison

To evaluate the system two load tests were conducted. Following the 1st load test, a comparative load test analysis was conducted using Azure Load Testing to assess the performance optimizations made to the backend system. The graph presented in Figure 51-53 illustrates four key metrics i.e. Virtual Users, Response Time, Average Requests per Second and Error Count between the pre-optimization test (Load Test 1) conducted on April 9, 2025 and the post-optimization test conducted (Load Test 2) on April 13, 2025.

Figure 53 Comparison - Virtual users (max)

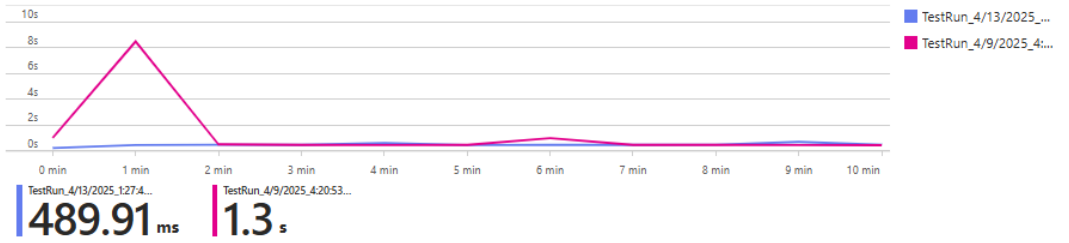


In both tests, the system handled 50 concurrent virtual users, showing consistent scalability in terms of user concurrency. However, noticeable improvements were observed in system responsiveness and throughput post-optimization. The 90th

[4] The throughput values visualized in the graphs represent averaged intervals (sampled over time), which may appear slightly lower than the overall calculated throughput shown in the summary table due to initial ramp-up and sampling granularity.

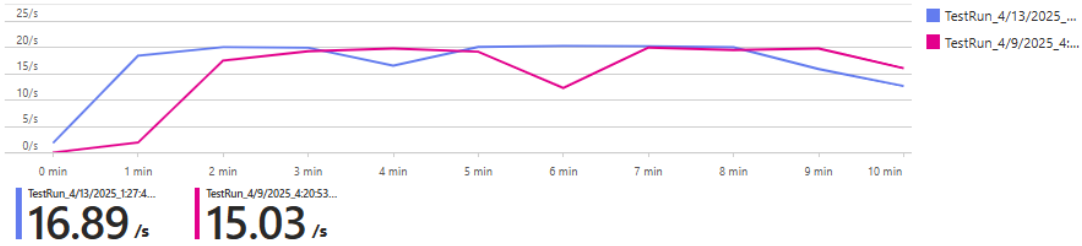
percentile response time decreased significantly from 1.3 seconds (1300 ms) to approximately 490 ms, demonstrating a substantial boost in how quickly the system responded under stress.

Figure 54 Comparison - Response time (Successful responses)



Moreover, the average requests per second increased from 15.03/s to 16.89/s, indicating that the optimized system could serve more requests efficiently within the same time frame. Importantly, no errors were recorded in either test, reinforcing the platform's backend stability and correctness even at full load.

Figure 55 Requests per second (Avg)



Overall, the optimization measures, such as improved endpoint handling and resource tuning, directly contributed to improved performance and user experience. These results confirm the readiness of the system to handle moderate-scale concurrent educational interactions without degradation in service quality.

5.2.4 Resource Utilization During Load Testing

Figure 56 CPU Percentage (Avg)

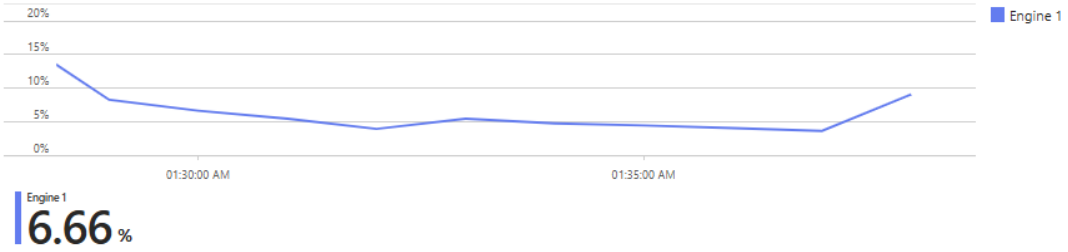


Figure 57 Memory Percentage (Avg)

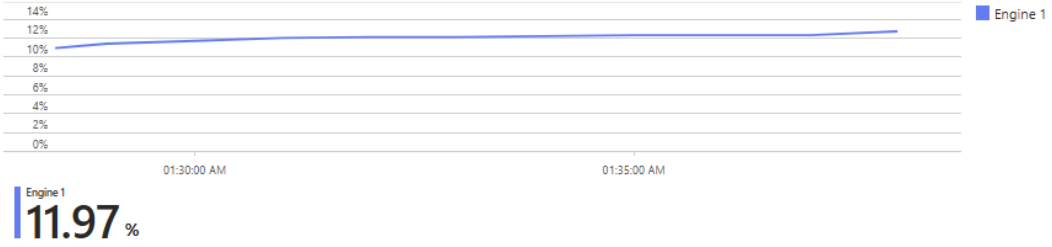
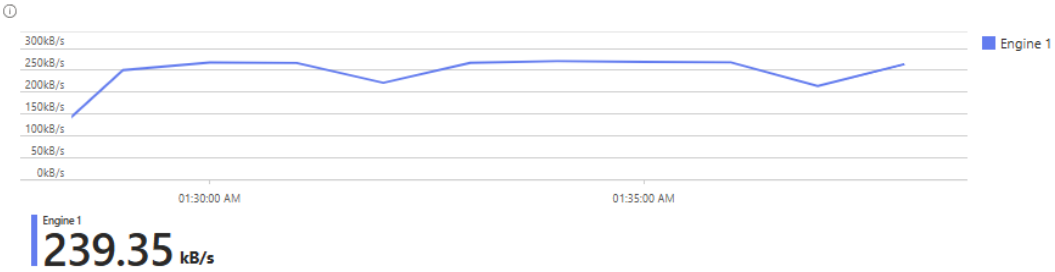


Figure 58 Network bytes per Seconds



The figures 53-55 illustrates average CPU usage at 6.66% and memory consumption at 11.97%, indicating efficient server-side resource utilization. The engine handled 239.35 kB/s network traffic and sustained 50 virtual users without performance degradation. These metrics reinforce the backend system's scalability and readiness for real-world deployment.

5.2.5 Unit Testing

To evaluate the correctness of AI-generated coding solutions within the interactive learning platform, an accuracy test was performed using a suite of ten test cases. Each test case consisted of:

- A daily coding challenge question automatically generated based on completed modules

- The AI-generated Python code solution
- An expected output or behavior
- A test function to validate the correctness of the AI's code

The Python exec() function was used to dynamically run each code snippet in an isolated namespace. Then, custom validation logic was applied to compare the actual output against the expected output, verifying functional correctness.

Table 5 Test Design and Coverage

Test Case No.	Module	Concept Tested
1	Introduction to Python	Function declaration and invocation
2	Variables and Data Types	Variable definition and string formatting
3	Operators and Expressions	Arithmetic operations
4	Control Flow	Conditional logic (FizzBuzz)
5	Functions	Parameterized function and return values
6	Operators and Expressions	Rectangle area calculation
7	Functions and Math Module	Circle area with *math.pi*
8	Operators and Expressions	Compound arithmetic operators
9	Operators and Expressions	Trapezoid area formula
10	Data Structures	Word frequency counter using a dictionary

Table 6 Evaluation Results

Test Case	Result
Test 1: Greeting Function	Pass
Test 2: Variables & Data Types	Pass
Test 3: Area of Triangle	Pass
Test 4: FizzBuzz	Pass
Test 5: Average Calculator	Pass
Test 6: Area of Rectangle	Pass
Test 7: Area of Circle	Pass
Test 8: Arithmetic Operators	Pass
Test 9: Area of Trapezoid	Pass
Test 10: Word Frequency	Pass

All ten tests passed successfully, confirming that the system-generated Python code matched expected behavior across a range of fundamental programming concepts. This validates the accuracy and reliability of the AI model integrated into the learning platform.

This result reinforces the system's credibility in automatically generating instructionally aligned, syntactically valid and functionally correct Python code. It demonstrates that the AI backend (powered by Gemini) not only provides educational content but also produces executable code that meets learning outcomes. The ability to test and verify correctness programmatically further strengthens confidence in the platform's use in real learning environments.

CHAPTER 6

DISCUSSION AND FUTURE WORK

6.1 Discussion

This chapter presents the results regarding the research problem and objectives. It also discusses the implications of having a Generative Artificial Intelligence (GAI) based interactive learning environment in programming education. The discussion is also enriched with the insight gained from the preceding chapters, that is, the in-depth review of opportunities and challenges as described in sections 2.4 and 2.5. The chapter also connects the observed outcomes to the broader educational context and highlights how the developed system addressed key challenges in existing programming education models.

Implementing GAI within the education system provides promising possibilities yet introduces challenges identified within the research problem. Traditional programming education often fails to accommodate diverse learner profiles, lacks immediate feedback mechanisms and offers minimal personalization. Furthermore, challenges regarding the accuracy of AI-generated content and ethical issues such as academic misuse persist.

This study developed and tested an AI-powered learning environment based on the Gemini API for Python programming education. The results show that the system was able to minimize most of the following issues:

- **Accuracy and Reliability**

While overall concerns over GAI unreliability, a survey's feedback guaranteed the confidence in explanation, code tip and quiz feedbacks from the system. Every participant rated content as "Accurate" or "Very Accurate"

Real-Time and Adaptive Support

In contrast to static traditional systems, the platform provided just-in-time feedback, clarification options tailored based on the user and consistent with learner-centered educational theories.

- **Ethics and Practical Issues**

While the system was built to highlight reasonable use of AI and transparency—completely alerting users that some content is generated using AI—there remain broader ethical concerns that are prevalent in the educational use of Generative AI. These include the potential for academic dishonesty, dependency on automated systems and the risk of learners bypassing foundational understanding by relying on AI to complete tasks.

To counter academic exploitation, like plagiarism or contract cheating, the system was deliberately made to be a learning companion and not a solution provider. For

instance, rather than providing direct solutions, the AI describes underlying coding concepts, provides sample solutions after attempting and breaks down steps while promoting self-reflection and personal problem-solving. The quiz design also supports this by providing explanation-based feedback rather than just scores, reducing the incentive to game the system for correct answers.

Another issue is over-reliance on AI, particularly in contexts where students start replacing critical thinking with AI-created content. The system addresses this by including elements such as daily challenges, open-ended problem-solving exercises and performance targets that motivate learners to interact with content instead of passively consuming it. The feedback is also made informative and interactive, prompting learning through discovery and error correction instead of providing learners with the direct solution.

- **Lack of Personalization**

Traditional systems provide static content without considering diversity among learners. This system gathered onboarding information, enabling Gemini to provide adaptive content, clarification sessions, motivational quotes and progress tracking based on users.

- **Engagement and Motivation**

Subsections 2.4.1 and 2.5.2 explain the motivational impact of GAI in learning. Completing a daily challenge, tracking progress and motivational remarks by this system effectively resolved demotivation in learning programming.

- **Immediate Feedback and Support**

Real-time feedback was a standout feature that replaced the typical delay of human evaluation. The AI-powered feedback process provided justification and recommendations when needed, resulting in more effective learning.

- **Availability of Resources**

As explained in Section 2.4.3, access to affordable and inclusive programming education remains a barrier. This system enabled students to learn independently from anywhere and minimize reliance on paid platforms and fragmented external resources.

6.1.1 Achievement of Research Objectives

Objective 1: Identify the impact of Generative AI on the education system

This study aimed to assess how Generative AI (GAI) technologies are influencing education; specifically in terms of enhancing learning outcomes, improving accessibility and supporting teaching processes. To achieve this, a thorough review of the literature was conducted in Sections 2.4 and 2.5, followed by the practical development and evaluation of an interactive Python learning platform powered by GAI.

In the literature review, Section 2.4 explores the role of GAI in programming education. It highlights how models like Gemini and ChatGPT can support learners through automatic code generation, real-time feedback, diverse problem-solving examples, code explanations and language translation. These features offer meaningful benefits, especially for students struggling to keep up with traditional, one-size-fits-all programming content. The review also identifies how GAI enhances engagement by simulating conversational support, generating exercises and allowing students to interact with code more flexibly and intuitively.

However, the review also notes several challenges. Section 2.4.2 outlines issues such as inaccuracies in AI-generated content, lack of contextual awareness and potential misuse in academic settings. Moreover, Section 2.4.3 addresses the shortcomings of current programming tools, such as the lack of personalization, limited real-time feedback and poor interactivity problems that GAI tools are well-positioned to solve when thoughtfully integrated.

Section 2.5 builds on this by focusing specifically on the capabilities of Gemini. It shows that Gemini not only supports students with answers and concept explanations but also helps teachers by generating learning materials, quizzes and suggestions for classroom activities. Gemini's ability to personalize learning, operate across multiple languages and offer instant, around-the-clock support gives it strong potential to enhance both learning and teaching. At the same time, this section also reflects on ethical concerns, including academic integrity, data reliability and the importance of responsible usage.

To validate these insights practically, the study implemented Gemini into a real learning environment and evaluated it through user testing, as presented in Chapter 5. The results strongly supported the findings from the literature. For instance, 88% of users found the AI-generated feedback effective and 87.5% stated that they did not need to rely on any external resources while using the system. These outcomes suggest that GAI, when integrated properly, can indeed improve feedback quality, reduce learning barriers and centralize the learning experience. Users also appreciated features like AI-generated teaching tips, daily challenges and personalized quizzes, all of which contributed to motivation and a better understanding of content.

Overall, this study achieved Objective 1 by combining a detailed theoretical exploration with practical implementation and evaluation. The findings confirm that GAI has a positive and measurable impact on the education system which make learning more adaptive, accessible and engaging while also highlighting areas that require thoughtful design to avoid misuse and maintain educational quality.

Objective 2: Analyze and address the challenges in the current programming educational environments

This objective focused on identifying the main difficulties learners face in programming education today and designing a system to effectively overcome

them. Section 2.4.3 of the literature review provided a detailed summary of these challenges. It highlighted that many existing programming learning platforms suffer from a lack of personalization, limited real-time feedback, passive user interfaces, outdated content and poor support for debugging. Additionally, they are often costly and inaccessible to many students, with little consideration for diverse learning needs or inclusivity.

The review also pointed out that these limitations can make learning programming slow, frustrating and demotivating specially for beginners. Without personalized support, learners struggle to follow at their own pace. Without timely feedback, they are often left confused about their mistakes. And without engaging, interactive features, motivation drops quickly. These issues are further made worse by content that doesn't stay up to date with modern practices or programming trends.

Many traditional platforms present the same content to all users regardless of their background or learning goals, which can make the material either too basic or too advanced. Additionally, without timely and meaningful feedback, learners often struggle to identify and correct mistakes. Interfaces that are static and non-interactive further reduce engagement, while expensive subscriptions or limited availability create accessibility barriers. The lack of up-to-date content and practical guidance also limits the value of such tools, especially for educators or students preparing for real-world applications.

To address these concerns, the system developed in this study was designed around the very gaps identified in the literature. The Python learning environment was built with personalization, real-time AI assistance. From the start, learners go through an onboarding process where their skill level, learning goals and interests are assessed. This information is then used to generate a personalized lessons. Beginners receive simpler explanations and foundational level tips and quizes, while intermediate users are given more advanced practice-oriented material. Educators receive teaching tips and guidance that support lesson planning and student engagement.

The system also includes an interactive code editor that allows users to write and run code in real time. After each lesson or quiz, detailed explanations are provided by the Gemini API, helping users understand not only which answers are correct, but why. Instead of generic error messages or static hints, learners receive context-based support that adapts to their needs. These explanations also help close the feedback loop, which is often missing or delayed in traditional platforms.

To increase motivation and engagement, the platform includes daily coding challenges, progress tracking dashboards and motivational tips, automatically generated based on the learner's activities and performance. The interface is designed to be clean, accessible and device-independent, making it usable across various systems without technical barriers. This directly responds to the cost and accessibility issues identified in Section 2.4.3.

Chapter 5 confirms the effectiveness of this design through user testing and feedback. For instance, 88% of users found the real-time feedback helpful, while 87.5% said they didn't need to use any external resources while learning. Most users described the system as engaging and effective and all participants reported overall satisfaction and said they would recommend it to others. These results show that the platform not only responded to the literature-based challenges but did so in a way that improved the overall learning experience.

By combining thoughtful design, Generative AI capabilities and direct user testing, this study successfully analyzed and addressed the key problems in programming education. The final system provides a more personalized, interactive and accessible way to learn Python achieving Objective 2 in both theory and practice.

Objective 3: Develop an interactive and personalized learning environment for students to learn Python programming

The system was successfully implemented to deliver an individualized and dynamic learning experience. As shown in the results Chapter 5, users particularly appreciated the daily challenges, AI-generated teaching tips, on-demand clarifications and quiz explanations. These tools directly supported both conceptual and applied learning outcomes, confirming the system's ability to meet diverse learner needs.

Implementing Gemini API in the Python learning environment effectively solved the main research issue by removing gaps in engagement, feedback, personalization and accessibility. The system successfully implemented all three components of research and offered a scalable approach to AI-based learning. In the future, the framework could be extended to other areas for broader application of Generative AI in online learning environments.

The interactivity of the environment is not solely a result of the technology infrastructure but supplemented by the way the tools are pedagogically organized to facilitate active learning. According to the constructivist learning theory[5], where it is assumed that students construct knowledge as they explore, get feedback and reflect. The system integrates several strategies and features that reflect this philosophy. These include:

- A skills-level onboarding questionnaire to determine previous knowledge and usage intention, so the system can provide personalized lessons.
- Lesson content that adapts to user roles and skill levels, where beginners receive simplified explanations, intermediates are guided through applied

[5] Constructivist learning theory, as developed by Jean Piaget and further expanded by Lev Vygotsky, emphasizes that learners actively construct knowledge through experiences, exploration, feedback, and reflection. See: Piaget, J. The Psychology of the Child. Basic Books, (1950); and Vygotsky, L. S. Mind in Society: The Development of Higher Psychological Processes. Harvard University Press, (1978).

examples and educators are provided with AI-generated teaching tips to enhance instruction or curriculum planning

- A real-time interactive code editor for writing and executing code within the browser, complemented by AI-generated sample solutions and hints where learners write, run, and test sample code in a real-time environment.
- An on-demand AI explanation service where users may ask for explanations or alternative explanations of programming concepts, mimicking tutor-like interactions.
- Practice exercises that are tailored to the user's skill level and focus area, with detailed, step-by-step explanations provided after each question. Rather than simply marking answers as right or wrong, the system explains why each option is correct or incorrect, reinforcing conceptual understanding and closing feedback loops. These immediate explanations and practice exercises help learners reflect and improve.
- Gamified motivational features, such as daily coding challenges, progress dashboards and AI-generated motivational tips depending on the users' progress, that keep learners engaged and provide a sense of accomplishment. These daily challenges, quizzes, and code tasks promote discovery, experimentation and critical thinking.
- Personalized feedback on quizzes, where students are not only given scores but specific explanations based on their answers to enable them to identify misunderstandings and reinforce learning.

For example, during practice sessions, the system does not merely grade responses or indicate correctness. Instead, it delivers personalized feedback generated by the Gemini API, designed to help learners grasp the underlying logic of their errors and understand key concepts in greater depth. Each learner's answer is provided with individual explanations tailored to them so they learn by understanding and not by memorization. The exercises themselves are dynamically adjusted to fit the user's skill level and learning needs, supporting interactive learning that grows with the learner's proficiency.

This combination of detailed feedback, personalized content and adaptive learning paths closely simulates the experience of having a personal tutor. It helps guide learners step by step, offering support that adjusts to their individual progress. As a result, the system turns self-paced programming education into a more engaging, flexible and student-centered experience; addressing the key limitations often seen in both traditional classrooms and many current online learning tools.

6.1.2 Technical Contribution

The technical contribution of this research lies in the successful development and integration of a personalized programming learning platform powered by Generative AI. The system goes beyond using an AI chatbot. It combines a

carefully designed architecture, adaptive content delivery and real-time AI feedback to support effective learning.

The platform is built on a three-tier architecture consisting of a React-based front-end, a Node.js back-end server and a MongoDB database (see Section 4.6.2). The front-end offers users an interactive experience, including live code editing, lesson viewing, progress dashboards and a bot icon to request explanations. The back-end manages user data, lesson delivery and communication with the Gemini API. The database stores learner profiles, quiz results and progress, enabling the system to provide a personalized experience for each user.

One of the key technical features is the integration of Google Gemini, which is used to generate lesson contents, personalized explanations, feedback, code suggestions, practice exercises and motivational tips. This is not done through generic queries. Instead, the system uses custom-built prompts that include details such as the user's skill level, learning goals, user's progress etc. This ensures that the AI responses are relevant, accurate and adapted to each learner's needs (see Section 4.6.3).

The system also includes a dynamic lesson engine, which generates lessons based on the onboarding survey. As users progress, the platform adjusts the difficulty of lessons and provides extra explanations or practice exercises if needed. Educators also receive AI-generated teaching suggestions tailored to their role.

A key strength of the system is the real-time feedback. When a user submits code or answers a quiz, the platform sends the input to Gemini, which returns a detailed explanation. This feedback is shown immediately, helping users understand their mistakes and learn from them which is similar to having a personal tutor.

The platform includes motivational features such as daily challenges and motivational tips depending on the progress tracking. These are also generated by Gemini using user-specific data, making them more meaningful and engaging.

Finally, the system was deployed on Microsoft Azure using scalable cloud services (see Section 4.6.5). Performance testing showed the platform can handle up to 50 users at once with fast response times and no errors, making it suitable for real educational environments.

In summary, this research contributes a complete, technically sound learning platform that combines interactive user design, personalized content generation and scalable deployment. It demonstrates how Generative AI can be effectively integrated into programming education to offer real-time support, personalized lessons and engaging features which are all within a stable and user-friendly system.

6.2 Limitations and Future Work

Other than the promising outcomes and positive response of the users, the interactive learning system has some limitations and areas of improvement. The majority of these constraints are functionality-related, depth of content, accessibility and support systems—all of which are essential to maximizing learners' performance and introducing the system's applicability to larger educational domains.

Even though the system does possess an integrated editor with real-time execution, it doesn't currently possess dynamic user input and advanced error handling. The users complained that the restriction of running programs that needed input at run-time limited coding practices. Future work must focus on accepting inputs and interactive debugging in the code editor.

Another essential component is that availability of peer interaction. Having aspects of community, i.e., forums or a chatbot assistant, would really add much value to collaborative learning. A course-content and user-behavior-directed chatbot would be a 24/7 personal aid. Meanwhile, a peer-to-peer discussion forum would enable users to ask questions, exchange thoughts and develop a sense of community—responding to learning as well as motivational requirements.

Content-wise, the system is currently focused most on introductory programming principles. While this makes it highly beginner-oriented. It also restricts its utility to higher-level learners or those wishing to enhance their mastery. Adding more advanced programming subjects would make the site more generalizable. Also this can be expanded by including more programming languages. The inclusion of multimodal content, including animations, code visualizations and video explanations, would also accommodate multiple learning styles.

Also, this can be further enhanced with integration several other popular programming language modules. As a prototype python language were chosen and with the integration with other languages this system can be more effective.

In accessibility and localization contexts, the system only supports English at the moment and has not yet fully addressed the needs of individuals with disabilities. Future releases should include multilingual support to reach communities that are not English speakers as well as incorporating accessibility features like screen reader support, zooming font size and keyboard-only functionality. These additional features would not only increase global access to the system but also make it more inclusive for various types of learners.

Finally, the system does not have any educator administrative functions. Adding an educator dashboard would provide teachers with the capacity to track student progress, recognize areas students need additional support in and provide one-on-one assistance. Progress tracking analytics, assignable tasks with settings to customize and manual feedback controls would be key to implementing the system in typical classroom settings and blended learning settings.

In brief, the elimination of these limitations with focused improvements will take the system from being an efficient learning platform to a comprehensive and scalable learning solution. The enhancements of the future will endeavor to make the platform more accessible, adaptable and conducive to more diverse, richer and meaningful learning experiences in the area of programming education.

REFERENCES

[1] P. Tarkar, "Impact Of Covid-19 Pandemic On Education System," *International Journal of Advanced Science and Technology*, vol. 29, no. 9s, pp. 3812–3814, 2020.

[2] C. K. Lo, "What Is the Impact of ChatGPT on Education? A Rapid Review of the Literature," Apr. 01, 2023, *MDPI*. doi: 10.3390/educsci13040410.

[3] H. B. Essel, D. Vlachopoulos, A. Tachie-Menson, E. E. Johnson and P. K. Baah, "The impact of a virtual teaching assistant (chatbot) on students' learning in Ghanaian higher education," *International Journal of Educational Technology in Higher Education*, vol. 19, no. 1, Dec. 2022, doi: 10.1186/s41239-022-00362-6.

[4] S. D'Mello, A. Olney, C. Williams and P. Hays, "Gaze tutor: A gaze-reactive intelligent tutoring system," *International Journal of Human Computer Studies*, vol. 70, no. 5, pp. 377–398, May 2012, doi: 10.1016/j.ijhcs.2012.01.004.

[5] Mehmet Firat, "What ChatGPT means for universities: Perceptions of scholars and students," *Journal of Applied Learning & Teaching*, vol. 6, no. 1, Apr. 2023, doi: 10.37074/jalt.2023.6.1.22.

[6] M. Sullivan, A. Kelly and P. McLaughlan, "ChatGPT in higher education: Considerations for academic integrity and student learning," *Journal of Applied Learning & Teaching*, vol. 6, no. 1, Mar. 2023, doi: 10.37074/jalt.2023.6.1.17.

[7] S. Bubeck *et al.*, "Sparks of Artificial General Intelligence: Early experiments with GPT-4," Mar. 2023, [Online]. Available: http://arxiv.org/abs/2303.12712

[8] J. C. Young and M. Shishido, "Investigating OpenAI's ChatGPT Potentials in Generating Chatbot's Dialogue for English as a Foreign Language Learning," *IJACSA) International Journal of Advanced Computer Science and Applications*, vol. 14, no. 6, 2023, [Online]. Available: www.ijacsa.thesai.org

[9] K. Malinka, M. Perešíni, A. Firc, O. Hujňák and F. Januš, "On the Educational Impact of ChatGPT: Is Artificial Intelligence Ready to Obtain a University Degree?," Mar. 2023, [Online]. Available: http://arxiv.org/abs/2303.11146

[10] S. Russell and P. Norwig, *Artificial Intelligence: A Modern Approach*. 2010.

[11] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, vol. 22, no. 4. MIT Press: The Korean Society of Medical Informatics, 2016. doi: 10.4258/hir.2016.22.4.351.

[12] M. Chassignol, A. Khoroshavin, A. Klimova and A. Bilyatdinova, “Artificial Intelligence trends in education: A narrative overview,” in *Procedia Computer Science*, Elsevier B.V., 2018, pp. 16–24. doi: 10.1016/j.procs.2018.08.233.

[13] Gatner, “Gartner Places Generative AI on the Peak of Inflated Expectations on the 2023 Hype Cycle for Emerging Technologies,” https://www.gartner.com/en/newsroom/press-releases/2023-08-16-gartner-places-generative-ai-on-the-peak-of-inflated-expectations-on-the-2023-hype-cycle-for-emerging-technologies.

[14] Gatner, “Gartner Experts Answer the Top Generative AI Questions for Your Enterprise,” https://www.gartner.com/en/topics/generative-ai.

[15] OpenAI, “OpenAI,” https://www.openai.com.

[16] B. Elad, “OpenAI Statistics By Demographics, Products, Revenue and Growth,” https://www.enterpriseappstoday.com/stats/openai-statistics.html.

[17] F. Richter, “Which Sectors Are Working With OpenAI?,” https://www.statista.com/chart/29244/number-of-companies-using-open-ai-in-their-business-processes-worldwide/.

[18] Sentisight.ai, “Google Gemini: How has it been received by users so far?” Accessed: Feb. 22, 2025. [Online]. Available: https://www.sentisight.ai/google-gemini-how-has-it-been-received-by-users-so-far/

[19] S. Pichai and D. Hassabis, “Introducing Gemini: our largest and most capable AI model.” Accessed: Feb. 23, 2025. [Online]. Available: https://blog.google/technology/ai/google-gemini-ai/

[20] D. L. Bitzer and R. L. Johnson, “PLATO: A Computer-Based System Used in the Engineering of Education,” 1971.

[21] E. H. Shortliffe^ and B. G. Buchanan, “A Model of Inexact Reasoning in Medicine*,” *Math Biosci*, vol. 23, pp. 351–379, 1975.

[22] A. C. Graesser, K. VanLehn, C. P. Rose, P. W. Jordan and D. Harter, “Intelligent Tutoring Systems with Conversational Dialogue,” *AI Magazine*, pp. 39–51, 2001.

[23] Anderson and John R, “Automaticity and the ACT theory,” *Am J Psychol*, pp. 165–180, 1992.

[24] J. R. Anderson, A. T. Corbett, K. R. Koedinger and R. Pelletier, “Cognitive Tutors: Lessons Learned,” *The journal of the learning sciences*, vol. 4, no. 2, pp. 96–66, 1996.

[25] H. David, “https://blog.khanacademy.org/how-khan-academy-is-using-machine-learning-to/,” How Khan academy is using machine learning to assess student mastery.

[26] F. D. Davis, “Perceived usefulness, perceived ease of use and user acceptance of information technology,” *MIS Q*, vol. 13, no. 3, pp. 319–339, 1989, doi: 10.2307/249008.

[27] G. Liu and C. Ma, “Measuring EFL learners’ use of ChatGPT in informal digital learning of English based on the technology acceptance model,” *Innovation in Language Learning and Teaching*, 2023, doi: 10.1080/17501229.2023.2240316.

[28] Generative AI Solutions Hub, “Best Practices in Teaching and Learning.” Accessed: Feb. 20, 2025. [Online]. Available: https://genai.illinois.edu/best-practices-teaching-and-learning/

[29] S. Biswas, “Role of ChatGPT in Computer Programming,” *Mesopotamian Journal of Computer Science*, pp. 8–16, Jan. 2023, doi: 10.58496/mjcsc/2023/002.

[30] N. M. S. Surameery and M. Y. Shakor, “Use Chat GPT to Solve Programming Bugs,” *International Journal of Information technology and Computer Engineering*, no. 31, pp. 17–22, Jan. 2023, doi: 10.55529/ijitc.31.17.22.

[31] S. Lau and P. Guo, “From ‘Ban It Till We Understand It’ to ‘Resistance is Futile’: How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot,” in *ICER 2023 - Proceedings of the 2023 ACM Conference on International Computing Education Research V.1*, Association for Computing Machinery, Inc, Aug. 2023, pp. 106–121. doi: 10.1145/3568813.3600138.

[32] R. Yilmaz and F. G. Karaoglan Yilmaz, “Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning,” *Computers in Human Behavior: Artificial Humans*, vol. 1, no. 2, p. 100005, Aug. 2023, doi: 10.1016/j.chbah.2023.100005.

[33] Open AI, “Does ChatGPT tell the truth?,” https://help.openai.com/en/articles/8313428-does-chatgpt-tell-the-truth.

[34] H. Pearce, B. Ahmad, B. Tan, B. Dolan-Gavitt and R. Karri, “Asleep at the Keyboard? Assessing the Security of GitHub Copilot’s Code Contributions,” *IEEE Symposium on Security and Privacy (SP)*, pp. 754–768, Aug. 2022, [Online]. Available: http://arxiv.org/abs/2108.09293

[35] S. Pichai, “An important next step on our AI journey,” https://blog.google/technology/ai/bard-google-ai-search-updates/.

[36] Y. Mehdi, “Reinventing search with a new AI-powered Microsoft Bing and Edge, your copilot for the web,” https://blogs.microsoft.com/blog/2023/02/07/reinventing-search-with-a-new-ai-powered-microsoft-bing-and-edgeyour-copilot-for-the-web/.

[37] J. D. Zamfirescu-Pereira, R. Y. Wong, B. Hartmann and Q. Yang, “Why Johnny Can’t Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts,” in *Conference on Human Factors in Computing Systems - Proceedings*, Association for Computing Machinery, Apr. 2023. doi: 10.1145/3544548.3581388.

[38] J. White *et al.*, “A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT,” Feb. 2023, doi: https://doi.org/10.48550/arXiv.2302.11382.

[39] P. P. Liang, C. Wu, L.-P. Morency and R. Salakhutdinov, “Towards Understanding and Mitigating Social Biases in Language Models,” in *Proceedings of the 38 th International Conference on Machine Learning,* 2021.

[40] E. M. Bender, T. Gebru, A. McMillan-Major and S. Shmitchell, “On the dangers of stochastic parrots: Can language models be too big?,” in *FAccT 2021 - Proceedings of the 2021 ACM Conference on Fairness, Accountability and Transparency*, Association for Computing Machinery, Inc, Mar. 2021, pp. 610–623. doi: 10.1145/3442188.3445922.

[41] M. Chen *et al.*, “Evaluating Large Language Models Trained on Code,” Jul. 2021, [Online]. Available: http://arxiv.org/abs/2107.03374

[42] B. A. Becker, P. Denny, J. Finnie-Ansley, A. Luxton-Reilly, J. Prather and E. A. Santos, “Programming Is Hard - or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation,” in *SIGCSE 2023 - Proceedings of the 54th ACM Technical Symposium on Computer Science Education*, Association for Computing Machinery, Inc, Mar. 2023, pp. 500–506. doi: 10.1145/3545945.3569759.

[43] S. Lau and P. Guo, “From ‘Ban It Till We Understand It’ to ‘Resistance is Futile’: How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot,” in *ICER 2023 - Proceedings of the 2023 ACM Conference on International Computing Education Research V.1*, Association for Computing Machinery, Inc, Aug. 2023, pp. 106–121. doi: 10.1145/3568813.3600138.

[44] E. Thompson, J. Whalley, R. Lister and B. Simon, “Code Classification as a Learning and Assessment Exercise for Novice Programmers,” *National Advisory Committee on Computing Qualifications*, 2006, [Online]. Available: www.naccq.ac.nz

[45] D. Baidoo-Anu, “Education in the Era of Generative Artificial Intelligence (AI): Understanding the Potential Benefits of ChatGPT in Promoting Teaching and Learning,” 2023. [Online]. Available: https://www.researchgate.net/publication/369385210

[46] O. A. Alasmari, J. Singer and M. Bikanga Ada, “Do Current Online Coding Tutorial Systems Address Novice Programmer Difficulties?,” in *ACM International Conference Proceeding Series*, Association for Computing Machinery, Sep. 2023, pp. 242–248. doi: 10.1145/3629296.3629333.

[47] S. Altares-López, J. M. Bengochea-Guevara, C. Ranz, H. Montes and A. Ribeiro, “Qualitative and quantitative analysis of student’s perceptions in the use of generative AI in educational environments,” May 2024, [Online]. Available: http://arxiv.org/abs/2405.13487

[48] J. Evanick, “From One-Size-Fits-All To Tailored Online Education: The Advantages Of Personalized Learning.” Accessed: Mar. 02, 2025. [Online]. Available: https://elearningindustry.com/from-one-size-fits-all-to-tailored-online-education-advantages-of-personalized-learning

[49] K. Cesur Aydin and H. G. Gunec, “Quality of information on YouTube about artificial intelligence in dental radiology,” *J Dent Educ*, vol. 84, no. 10, pp. 1166–1172, Oct. 2020, doi: 10.1002/jdd.12362.

[50] J. W. Gichoya *et al.*, “AI recognition of patient race in medical imaging: a modelling study,” *Lancet Digit Health*, vol. 4, no. 6, pp. e406–e414, Jun. 2022, doi: 10.1016/S2589-7500(22)00063-2.

[51] A. Tlili *et al.*, “What if the devil is my guardian angel: ChatGPT as a case study of using chatbots in education,” *Smart Learning Environments*, vol. 10, no. 1, Dec. 2023, doi: 10.1186/s40561-023-00237-x.

[52] A. Thurzo, M. Strunga, R. Urban, J. Surovková and K. I. Afrashtehfar, “Impact of Artificial Intelligence on Dental Education: A Review and Guide for Curriculum Update,” Feb. 01, 2023, *MDPI*. doi: 10.3390/educsci13020150.

[53] D. Herman, “The End of High-School English,” https://www.theatlantic.com/technology/archive/2022/12/openai-chatgpt-writing-high-school-english-essay/672412/.

[54] J. Sill, “Gemini for Education: A New Era of AI-Powered Learning in Schools.” Accessed: Feb. 23, 2025. [Online]. Available: https://deploylearning.com/gemini-for-education-a-new-era-of-ai-powered-learning-in-schools/

[55] Tutorialspoint, “Gemini for Educational Purposes.” Accessed: Feb. 23, 2025. [Online]. Available: https://www.tutorialspoint.com/gemini/gemini-for-educational-purposes.htm

[56] R. Davis, “Exploring the Impact of Gemini: Enhancing Technology Education in University Teaching.” Accessed: Feb. 23, 2025. [Online]. Available: https://devset.ai/blog/exploring-the-impact-of-gemini-enhancing-technology-education-in-university-teaching

[57] “Gemini.” Accessed: Feb. 23, 2025. [Online]. Available: https://deepmind.google/technologies/gemini/

[58] C. Kooli, “Chatbots in Education and Research: A Critical Examination of Ethical Implications and Solutions,” *Sustainability (Switzerland)*, vol. 15, no. 7, Apr. 2023, doi: 10.3390/su15075614.

[59] S. S. Gill *et al.*, “Transformative effects of ChatGPT on modern education: Emerging Era of AI Chatbots,” *Internet of Things and Cyber-Physical Systems*, vol. 4, pp. 19–23, Jan. 2024, doi: 10.1016/j.iotcps.2023.06.002.

[60] K. Bala *et al.*, “Generative Artificial Intelligence for Education and Pedagogy,” 2023. Accessed: Feb. 24, 2025. [Online]. Available: https://teaching.cornell.edu/generative-artificial-intelligence/cu-committee-report-generative-artificial-intelligence-education

[61] M. Halaweh, “ChatGPT in education: Strategies for responsible implementation,” *Contemp Educ Technol*, vol. 15, no. 2, Apr. 2023, doi: 10.30935/cedtech/13036.

[62] S. Dhingra, M. Singh, V. SB, N. Malviya and S. S. Gill, “Mind meets machine: Unravelling GPT-4’s cognitive psychology,” Mar. 2023, [Online]. Available: http://arxiv.org/abs/2303.11436

[63] W. Geerling, G. Dirk Mateer, J. Wooten and N. Damodaran, “Is ChatGPT Smarter than a Student in Principles of Economics?,” 2023. [Online]. Available: https://www.chronicle.com/article/chatgpt-has-everyone-freaking-out-about-cheating-its-not-the-first-time

[64] A. Bozkurt *et al.*, “The Manifesto for Teaching and Learning in a Time of Generative AI: A Critical Collective Stance to Better Navigate the Future,” *Open Praxis*, vol. 16, no. 4, pp. 487–513, Nov. 2024, doi: 10.55982/openpraxis.16.4.777.

APPENDIX A

Survey Questionnaire: Evaluation of the Interactive Learning Environment

Survey questionnaire designed to evaluate the interactive and personalized learning environment for programming education. The questionnaire is divided into sections to gather feedback on usability, functionality, engagement and overall satisfaction.

Section 1: Demographic Information

1. What is your age group?
    - Under 18
    - 18–24
    - 25–34
    - 35–44
    - 45 and above
2. What is your gender?
    - Male
    - Female
    - Prefer not to say
3. What is your educational background?
    - High school
    - Undergraduate
    - Postgraduate
    - Other _____
4. You joined the system as a:
    - Beginner
    - Intermediate
    - Educator
5. What is your overall level of programming experience? *
    - Beginner – Just starting out or learning basics
    - Intermediate – Can write simple programs
    - Advanced – Confident with building full applications
    - Expert – Deep experience, possibly working professionally

Section 2: Usability

6. How easy was it to navigate the learning environment?
    - Very easy
    - Easy
    - Neutral

- Difficult
- Very difficult

7. How easy was it to understand and use the user interface (UI)?
    - Very easy to understand and use
    - Easy to understand and use
    - Neutral
    - A bit hard to understand
    - Very hard to understand

Section 3: Functionality

8. How effective was the personalized feedback provided by the system?
    - Very effective
    - Effective
    - Neutral
    - Ineffective
    - Very ineffective
9. How helpful were the code suggestions and hints generated by the system?
    - Very helpful
    - Helpful
    - Neutral
    - Unhelpful
    - Very unhelpful
10. How accurate were the explanations provided by the system?
    - Very accurate
    - Accurate
    - Neutral
    - Inaccurate
    - Very inaccurate
11. How well did the system adapt to your learning pace and style?
    - Very well
    - Well
    - Neutral
    - Poorly
    - Very poorly

Section 4: Engagement

12. How engaging was the learning experience?
    - Very engaging
    - Engaging
    - Neutral

- Boring
- Very boring

13. How motivated were you to complete the programming exercises?
    - Very motivated
    - Motivated
    - Neutral
    - Unmotivated
    - Very unmotivated
14. Did the system make learning programming more enjoyable?
    - Strongly agree
    - Agree
    - Neutral
    - Disagree
    - Strongly disagree

Section 5: Learning Resources & Comparison

15. What sources do you often refer to when learning programming?
    - YouTube tutorials
    - Online courses (e.g., Coursera, Udemy)
    - Documentation and official guides
    - Forums (e.g., Stack Overflow, Reddit)
    - Blogs or articles
    - Books
    - Friends or mentors
    - Other_____
16. Compared to the sources you usually refer to, how easy was it to understand our system?
    - Much easier
    - A bit easier
    - About the same
    - A bit harder
    - Much harder
17. Did you use any other external sources to understand any concept while using the system?
    - Yes, frequently
    - Yes, occasionally
    - No, I understood everything within the system
    - Not sure
18. If yes, what kind of external sources did you find most helpful?

Section 6: Overall Satisfaction

19. How satisfied are you with the overall learning experience?
    - Very satisfied
    - Satisfied
    - Neutral
    - Dissatisfied
    - Very dissatisfied
20. Would you recommend this system to others?
    - Definitely
    - Probably
    - Neutral
    - Probably not
    - Definitely not

Section 7: Open-Ended Feedback

21. What did you like most about the learning environment?
22. What did you like least about the learning environment?
23. What features or improvements would you suggest to enhance the system?

APPENDIX B

Accuracy Testing Python Codes

Test 1: Greeting Function

```
def greet(name):
    print(f"Hello, {name}!")

output = []
def capture_print(msg):
    output.append(msg)

import builtins
real_print = print
print = capture_print
greet("World")
print = real_print
```

Test 2: Variables & Data Types

```
age = 30
name = "Alice"
height = 5.8
formatted = f"{name} is {age} years old and is {height} feet tall."
```

Test 3: Area of Triangle

```
base = 10
height = 5
area = 0.5 * base * height
```

Test 4: FizzBuzz

```
output = []
for number in range(1, 21):
    if number % 3 == 0 and number % 5 == 0:
        output.append("FizzBuzz")
    elif number % 3 == 0:
        output.append("Fizz")
    elif number % 5 == 0:
        output.append("Buzz")
    else:
        output.append(str(number))
```

Test 5: Average Calculator

```
def calculate_average(*args):
    if not args:
        return 0
    total = sum(args)
    return total / len(args)
```

Test 6: Area of Rectangle

```
length = 10
width = 5
area = length * width
```

Test 7: Area of Circle

```
import math
def calculate_area(radius):
    return math.pi * radius**2
area = calculate_area(5)
```

Test 8: Arithmetic Operators

```
a = 15
b = 7
sum_ab = a + b
difference_ab = a - b
product_ab = a * b
quotient_ab = a // b
remainder_ab = a % b
power_ab = a ** b
```

Test 9: Area of Trapezoid

```
a = 5
b = 7
h = 4
area = (a + b) / 2 * h
```

Test 10: Word Frequency

```
def word_frequency(sentence):
    word_counts = {}
    words = sentence.lower().split()
    for word in words:
        word = word.strip(".,!?\"'")
        if word in word_counts:
            word_counts[word] += 1
        else:
            word_counts[word] = 1
    return word_counts

result = word_frequency("This is a test sentence. This sentence is a
test.")
```