

Development of an AI-Assisted Coding Tool for Python Programming Course

Avilyne S. Malonzo^{1*}, Jeraldine M. Aquino², Hasly O. Duque³, Kristine B. Garcia⁴, Rhea D. Reyes⁵

Bachelor of Science in Information Technology – Mobile Development, College of computing and Information Sciences, Systems Plus College Foundation Angeles City, Pampanga, Philippines

Corresponding Author Email: avi.malonzo@gmail.com

ABSTRACT

The process of learning Python programming poses challenges for many students when it comes to program logic, error detection, debugging, and problem-solving. Even though there exist some AI-assisted coding tools like ChatGPT and GitHub Copilot, which can help students write code, they do not pay enough attention to the process of learning and developing skills for independent problem-solving. Thus, the purpose of this study is to create an AI-assisted coding tool for students enrolled in the Python Programming course. This tool will be able to provide code suggestions, detect programming errors, debug code and give explanations and personalized feedback according to the needs of the course. The research methodology used in the study will include a descriptive-developmental research design with a quantitative approach. The development of the tool will use Agile methodology based on requirements obtained from the course syllabus, common mistakes of novice programmers and consultation with the course instructor. The tool will be tested by unit, integration and user acceptance tests using prepared test cases and expert evaluation based on ISO/IEC 25010. It is expected that the tool will be able to provide reliable assistance with coding, detect and explain programming errors and help students learn programming concepts and debug code. The tool could also become an additional technological tool to teach Python programming for instructors.

Keywords: *Artificial Intelligence, AI-Assisted Coding, Python Programming, Programming Education, Coding Tool, Debugging Assistance.*

SECTION I — INTRODUCTION (THE PROBLEM AND ITS SETTING)

1.1 Introduction

The rapid development of Artificial Intelligence (AI), education has been highly impacted, especially programming and computer science areas. AI-assisted coding tools such as ChatGPT and GitHub Copilot became common due to the fact that they could provide suggestions, debugging and explanation of programming concepts. Large language models have also created new possibilities for personalized learning by offering immediate feedback and interactive assistance (Kasneci et al., 2023). Furthermore, AI-driven code generation tools have introduced both opportunities and challenges in programming education, particularly in introductory programming courses (Denny et al., 2024). As a result, the adoption of AI-assisted coding tools in programming teaching and learning continues to increase.

Despite these advancements, novice programmers often experience difficulties in understanding program logic, identifying errors, and developing problem-solving skills. While AI code generators can assist learners in completing programming tasks, the way students use and understand AI-generated code remains important to their learning process. Kazemitabaar et al. (2023) examined how novice learners use AI code generators when learning programming and identifies both opportunities and challenges associated with their use. These findings highlight the importance of developing AI-assisted tools that support learning rather than simply providing complete programming solutions.

Therefore, there is a need for an AI-assisted coding tool that focuses not only on providing coding assistance but also on helping students understand programming concepts and improve their problem-solving abilities. This study aims to develop an AI-assisted coding tool for Python Programming that provides guided coding assistance, understandable explanations, and debugging support to help students learn and comprehend Python programming concepts more effectively.

1.2 Background of the Study

Python is the primary programming language used in the Python course of the Bachelor of Science in Information Technology (BSIT), and the course requires students to master program logic, syntax, debugging, and problem-solving. Recent evidence shows that AI code generators can meaningfully assist novices: in a controlled study with 69 novice learners, students with access to an AI code generator completed significantly more code-authoring tasks, with a 1.15x increase in completion rate and 1.8x higher scores, without any decrease in their ability to modify code manually (Kazemitabaar et al., 2023). However, usability research on Copilot with novice programmers found that students struggle to understand and use such tools and face cognitive and metacognitive difficulties, which has direct design implications for how AI support should be scaffolded for beginners (Prather et al., 2023). Other studies also warn of overreliance, as students reported difficulty transferring knowledge to unaided tasks after using AI assistants, revealing gaps in conceptual understanding (Rojas-Galeano et al., 2025).

The gap of this study addresses is that most existing AI coding assistants are designed to produce complete code rather than to teach, and AI-generated hints often fail to target the learner's specific point of confusion or consider their previous help requests (Phung et al., 2025; Amiri & Islam, 2025). To address this gap, this study develops and evaluates an AI-assisted coding tool for the Python Programming course, one that combines coding assistance with understandable explanations, debugging support, and personalized feedback aligned with the learning requirements of the course. The tool is then evaluated in terms of functionality, usability, and effectiveness to determine how it can improve students' learning experience and confidence in Python programming.

1.3 Objectives of the Study

General Objective: develop and evaluate an AI-assisted coding tool for Python Programming course that provides intelligent coding assistance, debugging support, understandable explanations, and personalized feedback to support students in learning Python programming.

Specific Objectives:

1. Develop an AI-assisted coding tool that supports Basic, Intermediate, and Advanced Python programming concepts based on the learning requirements of the Python programming course.
2. Integrate features that provide code suggestions, identify programming errors, and assist students during coding activities.

3. Provide understandable explanations and personalized feedback to help students analyze, understand, and correct their Python code.
4. Integrate a built-in compiler that allows students to write, run, and test Python code within the application.
5. Evaluate the developed AI-assisted coding tool in terms of functionality, usability, and technical performance.

1.4 Scope and Limitation of the Study

Scope: This study focuses on the development of an AI-assisted coding tool designed to support students taking a Python Programming course. The proposed application will cover basic, intermediate, and advanced Python Programming concepts based on the learning requirements of the course. It will provide AI-assisted coding features, including code suggestions, programming error identification, debugging assistance, understandable explanations, and personalized feedback to support students during coding activities.

The application will also include a built-in compiler or code execution feature that will allow users to write, run, and test Python programs within the system. This feature is intended to help students immediately observe their program output and identify possible errors while practicing coding activities. The study will focus on the development and technical evaluation of these features to determine whether the proposed application functions according to its intended requirements.

Limitation: This study will be limited to Python programming and will not include other programming languages. Although the application will cover basic, intermediate, and advanced Python concepts, it may not include every topic, library, or framework related to Python programming. The built-in compiler or code execution feature may also have limitations when executing programs that require external libraries, advanced configurations, or additional system resources. The AI-generated suggestions, explanations, and feedback may also be affected by the quality of the user's input, available AI services, and internet connection.

The study will focus on the initial development and technical evaluation of the proposed application. It will not measure the long-term improvement of students' academic performance, programming confidence, or overall learning experience through surveys, long-term observation, or other student-outcome measurements.

1.5 Significance of the Study

Students. The developed AI-assisted coding tool may help students learn basic, intermediate, and advanced Python programming concepts by providing coding assistance, error identification, debugging support, and understandable explanations. It may also provide explanations that can support students in analyzing and correcting their code.

Instructor. The application may serve as an additional educational tool that instructors can use to support Python programming activities and provide students with supplementary coding assistance outside regular classroom discussions.

Educational Institutions. This study may support the integration of Artificial Intelligence into programming education by presenting an educational technology solution designed specifically for Python Programming students. The proposed tool may also serve as a basis for exploring the use of AI-assisted technologies in programming education.

Future Researchers. The study may serve as a reference for future researchers who plan to develop, improve, or evaluate AI-assisted coding tools for programming education. The proposed system and its technical evaluation may provide a basis for future studies involving additional programming concepts, features, or evaluation methods.

1.6 Definition of Terms

Artificial Intelligence (AI). A technology that enables computer systems to perform tasks that normally require human intelligence. In this study, it is used to provide coding assistance, explanations, debugging support, and feedback

AI-Assisted Coding Tool. A software application that uses AI to assist users in writing and understanding code. In this study, it refers to the proposed tool for learning Python programming.

Python Programming. The process of creating programs using the Python programming language. In this study, it covers the basic, intermediate, and advanced Python concepts included in the proposed tool.

Code Suggestion. A recommended piece of code provided to help users write or complete a program. In this study, it refers to AI-generated suggestions for Python code.

Debugging. The process of identifying and correcting errors in a program. In this study, it refers to the tool's assistance in identifying and correcting errors in Python code.

Built-in Compiler/Code Execution. A feature that allows programming code to be processed and executed. In this study, it allows students to write, run, and test Python programs within the application.

Personalized Feedback. Guidance adapted to the needs or responses of an individual learner. In this study, it refers to AI-generated feedback based on the student's Python code, errors, questions, or interactions.

Programming Error. A mistake in code that may prevent a program from working correctly. In this study, it refers to errors identified while students write or execute Python programs using the proposed tool.

SECTION II — LITERATURE REVIEW

2.1 Introduction to the Review

The literature is organized into five themes: **AI-assisted coding in programming education; AI-Assisted programming and learning outcomes across different skill levels; AI-assisted debugging, error detection, and personalized feedback; AI-Assisted Programming Tools and Learning-Oriented Design; and AI adoption in Philippine education.**

2.2 Review of Related Literature and Studies

Liang et al. (2023) discussed the growing use of Generative AI in software development and programming activities. Their work highlighted how tools such as ChatGPT and GitHub Copilot can assist users through code generation, completion, explanation, and other programming-related tasks. These capabilities allow users to obtain immediate assistance while working on coding problems. The findings demonstrate the potential of AI-assisted coding to support programming activities and reduce some of the difficulties encountered during software development.

Peng et al. (2023) evaluated the effects of GitHub Copilot on programming productivity and reported that AI assistance improved the speed at which developers completed programming tasks. The study demonstrates how AI coding assistants can reduce the effort required for certain programming activities. However, productivity improvements primarily describe task performance and do not necessarily indicate that users have developed a deeper understanding of programming concepts. This distinction is important when applying AI coding assistance to educational settings.

Liang et al. (2024) investigated the limitations of AI coding assistants when dealing with software requirements and found that generated code may not always satisfy specific functional and non-functional requirements.

Atemkeng et al. (2024) likewise emphasized that Generative AI should complement rather than replace traditional programming skills.

These findings indicate that AI-generated code still requires human evaluation and understanding. For programming education, this supports the use of AI tools that provide explanations and guidance alongside code suggestions rather than simply delivering complete solutions.

AI-Assisted Programming and Learning Outcomes Across Different Skill Levels

Kazemitabaar et al. (2023) studied AI code generation among novice programming learners and found that students with access to an AI code generator completed more code-authoring tasks and achieved higher task scores. The study demonstrates that AI assistance can provide useful support to students who are still developing their programming skills. However, the results also indicate the importance of considering how students interact with generated code because successful task completion does not necessarily guarantee independent understanding.

Prather et al. (2024) investigated how programming students with different metacognitive abilities interacted with Generative AI. Their findings indicated that students with stronger metacognitive abilities benefited more from AI assistance, while students experiencing metacognitive difficulties could experience negative effects. The study highlights the importance of students' ability to evaluate and regulate their use of AI. These findings support the need for programming tools that guide students rather than completely taking over the problem-solving process.

Llerena-Izquierdo et al. (2024) evaluated the use of a Gemini-based AI assistant for educational programming activities and found that most participants considered the AI assistance useful and reported that it exceeded their expectations. Tanay et al. (2024) also reported that learners commonly used Generative AI to support programming and academic activities.

These findings demonstrate the usefulness and acceptance of AI-assisted programming among learners. However, they also support the need for structured AI assistance that encourages students to understand and evaluate the generated information.

AI-Assisted Debugging, Error Detection, and Personalized Feedback

Pankiewicz and Baker (2023) developed a system that used GPT-3.5 to generate personalized hints for students working on programming assignments. Students generally rated the AI-generated hints positively, and the group receiving the hints demonstrated higher success across consecutive submissions when the assistance was available. However, the researchers also observed indications of possible over-reliance when the AI feedback was removed. These findings support the use of AI-generated hints while emphasizing the importance of designing feedback that encourages students to develop independent problem-solving skills.

Nguyen and Allan (2024) implemented GPT-4 to provide individualized feedback on students' programming code and pseudocode. The system was instructed to provide feedback on conceptual understanding, syntax, and time complexity while also suggesting follow-up actions or guiding questions. Human evaluators generally considered the generated feedback useful for guiding students' next steps. The study demonstrates how AI can provide feedback that goes beyond identifying whether code is correct by helping students understand what they should consider next.

Phung et al. (2024) developed a human tutor-style approach that used GPT-4 to generate programming hints and GPT-3.5 to validate the usefulness of those hints. The approach was evaluated using Python programming datasets covering concepts such as basic algorithms, regular expressions, and data analysis with pandas. The study demonstrates the potential of AI to generate individualized programming guidance, while also recognizing that AI-generated feedback still requires careful validation because its quality can differ from that of human tutors.

AI-Assisted Programming Tools and Learning-Oriented Design

CodeAid, developed by Kazemitabaar et al. (2024), was designed as an LLM-based programming assistant that provides conceptual explanations, pseudocode, line-by-line explanations, and suggestions for correcting incorrect code without directly revealing complete solutions. The system was deployed in a programming class involving 700 students and was evaluated through thousands of interactions, surveys, and interviews. The study

demonstrates an approach in which AI assistance can provide students with programming support while reducing the emphasis on simply giving complete answers.

Xiao et al. (2024) explored multiple levels of GPT-generated programming hints with novice learners. Their system provided four levels of assistance ranging from general natural-language guidance to more concrete code assistance. The study found that high-level hints alone could sometimes be insufficient or misleading, while more specific assistance, such as code examples with inline comments, could provide better support. These findings suggest that AI feedback should be adjustable according to the student's specific difficulty and the level of assistance required.

Lohr et al. (2025) investigated whether LLMs could generate different types of hints for introductory programming tasks. The study reported generally positive feedback from students and experts regarding the generated hints but also identified instances of misleading information and insufficient detail. The findings demonstrate that AI-generated programming assistance has potential but should be carefully designed and evaluated.

These findings support the proposed tool's focus on understandable and targeted assistance rather than unrestricted AI-generated answers.

AI Adoption in Philippine Education

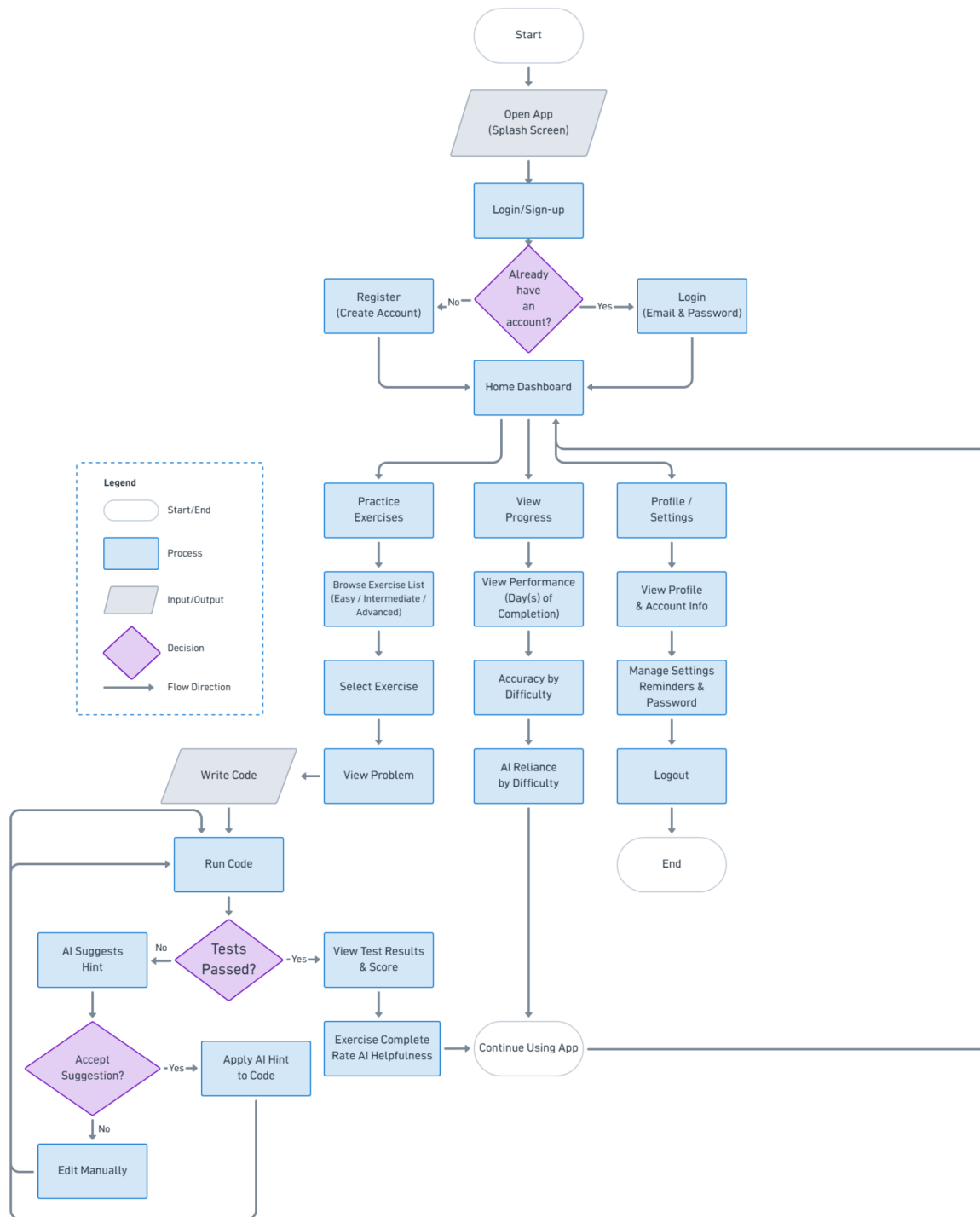
Estrellado and Miranda (2023) discussed the increasing integration of Artificial Intelligence into the Philippine educational system and identified opportunities for enhanced learning experiences and data-driven decision-making. At the same time, the authors highlighted concerns involving technological infrastructure, data privacy, the digital divide, and the need for faculty training. These findings demonstrate that AI adoption in Philippine education requires both technological preparation and responsible implementation.

Asirit and Hua (2023) assessed AI readiness and utilization in Philippine higher education and highlighted the importance of understanding how institutions and students respond to the increasing availability of AI technologies. The study is relevant to the proposed system because student readiness and familiarity can influence how effectively AI-based educational tools are used. This supports the need for AI applications that are accessible, understandable, and appropriately aligned with students' educational activities.

Co (2025) provided a narrative review of AI implementation in Philippine education based on 43 studies. The review identified differences in AI implementation across educational institutions and discussed challenges involving infrastructure, readiness, professional development, and responsible AI integration. The study recommends stronger AI education policies, improved digital infrastructure, and professional development.

These findings support the development of educational AI tools that are designed around actual student needs while remaining aligned with responsible AI practices in the Philippine educational context.

2.3 Conceptual Framework



2.4 Synthesis

Existing study demonstrates the growing impact and transformative potential of Artificial Intelligence in computer science education. Cambaz & Zhang (2024) found that AI-driven tools offer valuable assistance in code generation, error detection, interactive debugging, and immediate feedback, supporting self-regulated and personalized learning. Furthermore, Peng et al. (2023) highlight how rapidly AI integration is advancing within modern programming workflows. However, Denny et al. (2024) and UNESCO (2023) emphasize concerns regarding the use of generative AI in education, particularly when these tools complete tasks for students

instead of supporting their conceptual understanding and independent learning.

Despite these developments, existing research and popular commercial platforms focus primarily on general AI code generation, developer productivity tools, or static programming hints. As noted by Amiri & Islam (2025) and Phung et al. (2025), tools like GitHub Copilot do not distinguish between novice learners and experienced developers. These tools often provide direct code solutions instead of pedagogical explanations or failing to address a student's specific point of confusion.

Therefore, this study addresses a practical/knowledge gap by developing an AI-assisted coding tool specifically designed to support the learning requirements of a Python Programming course. The proposed application will combine real-time coding assistance with understandable explanations, error identification, personalized feedback, and a built-in compiler feature. Through the development and technical evaluation of these features, the study aims to provide an educational coding tool that supports students in understanding and practicing Python programming while addressing limitations observed in existing AI-assisted coding solutions.

SECTION III — METHODOLOGY

3.1 Research Design

This study will adopt a Developmental Research Design to guide the creation of an AI-assisted mobile application coding tool for students taking a Python Programming course. The research will begin by identifying common difficulties encountered in Python programming and determining the type of assistance that can be incorporated into the proposed tool. These identified needs will serve as the basis for defining the system requirements and selecting the appropriate AI-assisted functions for the application.

The development of the tool will proceed through several stages, including system planning, design, implementation, integration of the AI components, and iterative testing. The proposed tool will incorporate features for generating code suggestions, identifying programming errors, assisting with debugging, providing understandable explanations, and delivering feedback based on the student's coding activity. Each feature will be tested individually and as part of the complete system to determine whether it functions correctly and works together with the other components.

The researchers will use the results of system testing to identify errors, usability concerns, and areas requiring improvement. Necessary modifications will then be incorporated into subsequent versions of the tool until the intended functions are achieved. The study will primarily evaluate the development and functionality of the proposed system rather than measure long-term changes in students' academic performance.

A Developmental Research Design is appropriate because the primary output of the study is a functional AI-assisted coding tool designed to address specific difficulties in Python programming. It provides a structured approach for transforming identified learner needs into system features and for refining the resulting tool through testing and improvement. This approach allows the researchers to develop a practical technological solution while ensuring that its functions remain aligned with the objectives of the study.

3.2 System Development Methodology

The study will use a developmental research approach guided by the Agile model as its software development methodology. The Agile Model follows an iterative and incremental process where system features are developed, tested, and refined through repeated development cycles.

Requirements Analysis

The researchers will identify the functional and non-functional requirements of the proposed application through a review of related literature, existing AI coding tools, and analysis of common challenges experienced by novice Python programmers. The required Python code execution features, coverage of basic, intermediate, and advanced Python programming concepts, and learning oriented AI assistance will be defined during this stage.

System Design

The researchers will design the user interface, navigation flows, database structure, system architecture, and overall application layout. Detailed wireframes for the coding workspace, practice exercise, code execution interface, and feedback features will be created to serve as the guides during application development.

Development

The researchers will develop the application based on the approved requirements and system design. Core elements such as user interface, database, coding workspace, and learning support features, will be developed incrementally through Agile development cycles. Each completed feature will be reviewed and tested before proceeding to further development.

AI Scaffolding & Integration

The AI assisted features will be integrated using large language model (LLM) APIs and structured prompting techniques. Instead of focusing only on generating complete solutions, the AI features will be designed to provide code suggestions, identify programming errors, offer debugging guidance, and deliver understandable explanations appropriate to the user's programming level. These features will support students in understanding and improving their Python code.

Testing

System testing will be conducted throughout the development process and after the completion of the application. Functional testing will be used to verify whether each feature operates according to its requirements. Integration testing will assess whether the different system components work together correctly. Additional technical testing may be conducted to identify errors, execution issues, and performance limitations. Necessary corrections and refinement will be applied based on the testing results.

Implementation

After comprehensive testing and necessary refinements, the finalized AI-assisted coding tool will be prepared for deployment and technical evaluation. The application will be made available in its intended operating environment to demonstrate its functionality and assess whether its implemented features perform according to the specified requirements.

Maintenance

Continuous system monitoring, bug fixes, performance optimizations, and content updates will be conducted to address technical issues identified during actual deployment.

3.3 Requirement Specification and Analysis

The system requirements will be determined by reviewing relevant literature, existing AI-assisted programming learning platforms, coding practice applications, and the study's objectives. The proposed AI-assisted coding tool will be designed to support students enrolled in a Python Programming course by providing learning resources, coding practice, and AI-assisted programming guidance.

The proposed application will allow students to register and manage an account, use Python learning modules, work through coding exercises at various difficulty levels, write and submit Python code through an integrated code editor, view execution results, and receive automated evaluations using test cases. The system will also include learning progress tracking and performance reports to help students monitor their coding activities and completed exercises.

The AI component will offer:

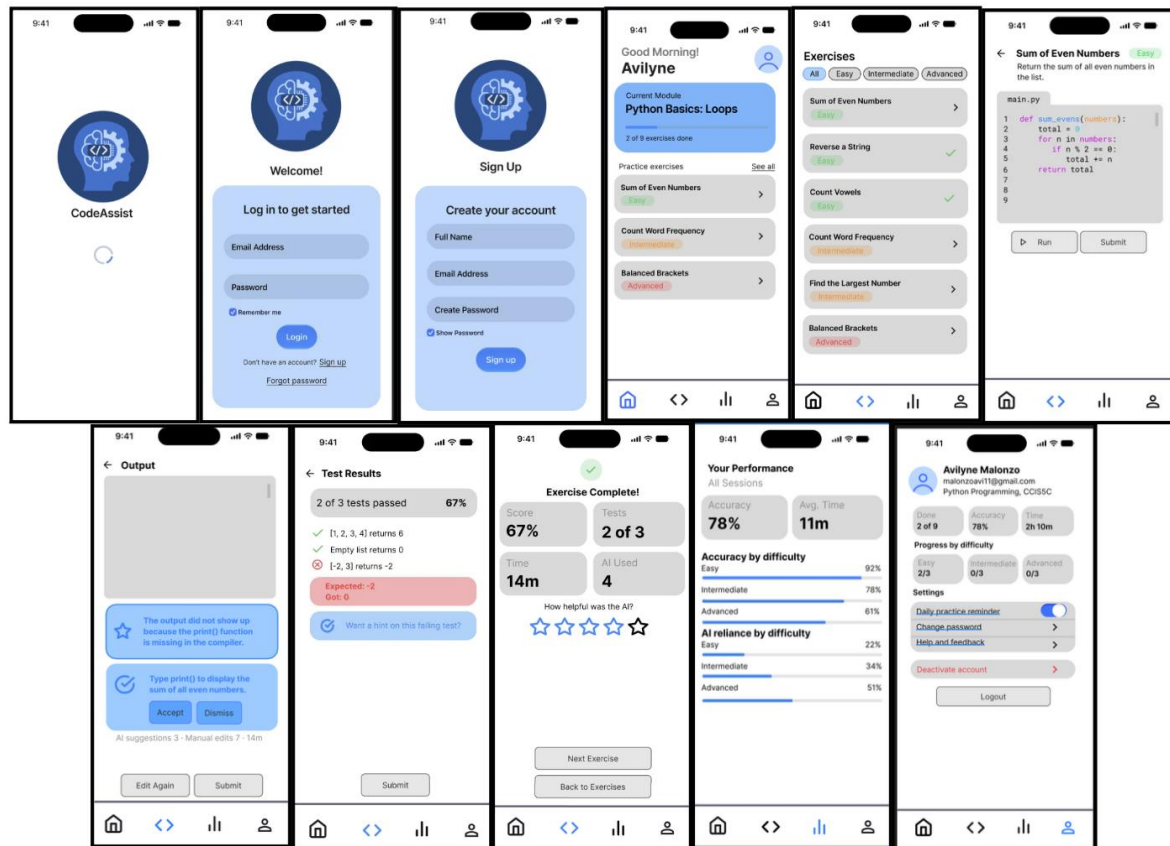
AI Code Analysis - examines the Python code submitted by the student to detect syntax errors, logical errors, and other typical programming mistakes.

AI Debugging and Feedback - offers explanations of the errors found, debugging suggestions, and personalized feedback that helps students understand and correct their code.

Student Control - lets students examine the AI-generated feedback and suggestions, choose whether to follow the recommendations, modify their code, and resubmit their solutions.

System requirements will be depicted through suitable diagrams, including use-case diagrams, flowcharts, Data Flow Diagrams (DFDs), and Entity-Relationship Diagrams (ERDs). These modeling tools will demonstrate the functional requirements, system processes, data flow, and database structure, offering a clear framework for designing and developing the proposed AI-assisted coding tool.

3.4 System Design Specification



3.5 Implementation

During the implementation phase, the core user interface, database structures, coding workspace, execution compiler, and AI-assisted feedback features will be combined into a single unified system.

A combination of an embedded local database (SQLite/Isar) and cloud storage (Firebase Firestore) will be used to store structured application data such as user accounts, exercise modules, test cases, progress metrics, and submission histories. Firebase Cloud Storage will be used for profile assets and static app resources.

The AI component will process relevant user code, error logs, and course context to return pedagogical hints,

line-by-line explanations, and debugging feedback that students can review, accept, or edit before applying to their code.

This section satisfies the proposal guide's requirements by detailing how the system organizes its code and configures the database. It also illustrates how each system component connects to build a fully operational application.

3.6 Testing and Evaluation

The Developed AI-Assisted Coding Tool for Python Programming Course will undergo a set of software testing procedures to confirm that the application works properly, successfully integrates all components, and fulfills the intended requirements of its users. Testing will be carried out during the development process to detect and fix errors prior to deployment.

The testing will focus on the following areas:

Unit Testing - Each individual system component, including user authentication, learning modules, coding exercise management, AI feedback generation, automated test case evaluation, and performance tracking, will be tested independently to confirm that each module operates according to its defined requirements.

Integration Testing - Once unit testing has been completed successfully, the separate modules will be combined and tested as a group to confirm that data moves correctly between the application's components. This covers testing how the code editor, AI feedback module, automated test case evaluator, user progress tracker, and database interact with one another.

Functional Testing - The application will be tested to determine whether its major features perform their intended functions. This will include testing user registration, Python code execution, AI-assisted code analysis, debugging feedback, exercise submission, and progress recording.

Technical Performance Evaluation - The system will be evaluated based on its response time, execution reliability, error handling, and ability to process coding-related requests. The results will be used to identify technical limitations and determine whether the application meets its intended performance requirements.

Testing results will be documented and analyzed to identify defects, integration issues, and areas requiring improvement. Necessary refinements will be implemented before the application is finalized.

REFERENCES

Reference Guidelines

1. Amiri, S., & Islam, M. (2025). Enhancing Python programming education with an AI-powered code helper: Design, implementation, and impact. *Software Engineering*, 11(1), 1–17.
<https://doi.org/10.11648/j.se.20251101.11>
2. Asirit, L. B. L., & Hua, J. H. (2023). Converging perspectives: Assessing AI readiness and utilization in Philippine higher education. *Polaris Global Journal of Scholarly Research and Trends*, 2(3).
<https://doi.org/10.5281/zenodo.15694643>
3. Cambaz, D., & Zhang, X. (2024). Use of AI-driven code generation models in teaching and learning

- programming: A systematic literature review. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (pp. 172–178). ACM. <https://doi.org/10.1145/3626252.3630958>
4. Co, S. J. (2025). Artificial intelligence in Philippine education: A narrative review of applications, perceptions, and challenges. *International Research Journal of Science, Technology, Education and Management*, 5(2), 25–38. <https://doi.org/10.5281/zenodo.15902950>
 5. Denny, P., Prather, J., Becker, B. A., Finnie-Ansley, J., Hellas, A., Leinonen, J., Luxton-Reilly, A., Reeves, B. N., Santos, E. A., & Sarsa, S. (2024). Computing education in the era of generative AI. *Communications of the ACM*, 67(2), 56–67. <https://doi.org/10.1145/3624720>
 6. Estrellado, C. J. P., & Miranda, J. C. (2023). Artificial Intelligence in the Philippine Educational Context: Circumspection and Future Inquiries. *International Journal of Scientific and Research Publications*, 13(5). <http://dx.doi.org/10.29322/IJSRP.13.05.2023.p13704>
 7. Jošt, G., Taneski, V., & Karakatič, S. (2024). The impact of large language models on programming education and student learning outcomes. *Applied Sciences*, 14(10), Article 4115. <https://doi.org/10.3390/app14104115>
 8. Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., ... Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103, Article 102274. <https://doi.org/10.1016/j.lindif.2023.102274>
 9. Kazemitabaar, M., Chow, J., Carl, M., Ericson, B. J., Weintrop, D., & Grossman, T. (2023). Studying the effect of AI code generators on supporting novice learners in introductory programming. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. ACM. <https://doi.org/10.1145/3544548.3580919>
 10. Lohr, D., et al. (2025). You're (Not) My Type—Can LLMs Generate Feedback of Specific Types for Introductory Programming Tasks? *Journal of Computer Assisted Learning*. <https://doi.org/10.1111/jcal.13107>
 11. Molenaar, I. (2022). The concept of hybrid human-AI regulation: Exemplifying how to support young learners' self-regulated learning. *Computers and Education: Artificial Intelligence*, 3, Article 100070. <https://doi.org/10.1016/j.caeai.2022.100070>
 12. Nguyen, H., & Allan, V. (2024). Using GPT-4 to Provide Tiered, Formative Code Feedback. *Proceedings of the 55th ACM Technical Symposium on Computer Science Education*. <https://doi.org/10.1145/3626252.3630960>
 13. Pankiewicz, M., & Baker, R. S. (2023). Large Language Models (GPT) for automating feedback on programming assignments. *International Conference on Computers in Education*. <https://doi.org/10.58459/icce.2023.950>
 14. Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023). *The impact of AI on developer productivity: Evidence from GitHub Copilot*. arXiv. <https://arxiv.org/abs/2302.06590>
 15. Phung, T., Wu, M., Choi, H., Soares, G., Gulwani, S., Singla, A., & Brooks, C. (2025). Bridging gaps between student and expert evaluations of AI-generated programming hints. In *Proceedings of the Twelfth ACM Conference on Learning @ Scale* (pp. 241–245). ACM. <https://doi.org/10.1145/3698205.3733932>
 16. Rojas-Galeano, S., Tejada, J., & Marmolejo-Ramos, F. (2025). *Between tool and trouble: Student attitudes toward AI in programming education*. arXiv. <https://doi.org/10.48550/arXiv.2508.05999>
 17. UNESCO. (2023). *Guidance for generative AI in education and research*. <https://unesdoc.unesco.org/ark:/48223/pf0000386693>
 18. Wang, F., King, R. B., Chai, C. S., et al. (2023). University students' intentions to learn artificial intelligence: the roles of supportive environments and expectancy-value beliefs. *International Journal of Educational Technology in Higher Education*, 20, 51. <https://doi.org/10.1186/s41239-023-00417-2>
 19. Xiao, R., Hou, X., & Stamper, J. (2024). Exploring How Multiple Levels of GPT-Generated Programming Hints Support or Disappoint Novices. <https://doi.org/10.1145/3613905.3650937>

