AI-DRIVEN PROGRAMMING EDUCATION FOR BEGINNERS: INTEGRATING INTERACTIVE LESSONS, NATURAL LANGUAGE CODING, AND SECURE LEARNING SYSTEMS

Tyler Hansen [1], Joshua Larracas [2]

[1] Army & Navy Academy, 2605 Carlsbad Blvd, Carlsbad, CA 92008
[2] California State Polytechnic University, Pomona, CA, 91768

ABSTRACT

This paper addresses the challenge of improving accessibility and engagement in programming education for beginners who often struggle with motivation and practical application. To tackle this issue, an interactive application was developed that integrates AI-assisted coding lessons, natural language input, voice-to-code generation, and secure login through Firebase [1]. The system relies on machine learning to produce structured lessons, process user prompts, and generate usable robotics code in real time [2]. Three central components—the lessons module, the voice input system, and the login mechanism—were analyzed in detail to illustrate their functionality. A survey experiment demonstrated strong results in usability, usefulness, and satisfaction, though accuracy was identified as an area for refinement. Comparisons with existing research highlighted that while other projects emphasize classroom evaluation or AI benchmarking, this system uniquely combines teaching and practice within one platform [3]. Overall, the findings suggest that the application provides an effective and scalable solution for programming education.

KEYWORDS

AI coding education, Interactive learning app, Voice-to-code, Beginner programming

1. INTRODUCTION

Programming has become an essential skill in today's technology-driven world, yet many learners struggle to find accessible and interactive ways to practice coding. Traditional methods, such as textbooks or static tutorials, often fail to keep learners engaged or provide immediate feedback. According to recent surveys, over 60% of beginner programmers report frustration and dropout within the first few months of learning due to a lack of clear guidance and support. This challenge particularly affects students in secondary and post-secondary education, as well as hobbyists who do not have direct access to instructors. The downside of this problem is a growing skills gap in computing fields, where employers report a shortage of applicants with practical programming experience. Without accessible tools that make coding more engaging and personalized, many learners may be discouraged from pursuing computer science or robotics further. This problem is important to solve because the demand for coding knowledge continues to expand across industries such as engineering, healthcare, robotics, and AI. Making

David C. Wyld et al. (Eds): SIGI, CSTY, AI, NMOCT, BIOS, AIMLNET, MaVaS, BINLP – 2025
pp. 357-367, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.151928

programming more approachable can empower students and beginners, helping them build confidence and develop skills that are essential for future careers. An interactive, AI-driven platform offers one promising path toward addressing this challenge.

The methodology section compares the program with three existing research methodologies related to AI in programming education. The first study by Fan et al. (2025) examined AI-assisted pair programming, showing improvements in motivation and reduced anxiety compared to traditional pair or solo work. The second study by Yılmaz (2023) investigated the effect of ChatGPT on computational thinking and motivation, reporting positive outcomes in classroom environments. The third paper by Phung et al. (2023) benchmarked ChatGPT and GPT-4 against human tutors, finding the AI effective in debugging and explanation tasks but limited in social interaction. In contrast, the present program integrates structured lessons, text and voice prompts, and real-time robotics code generation into a single platform. This unique combination addresses gaps in adaptability and interactivity, offering a self-paced environment that strengthens both conceptual understanding and practical application beyond the scope of the other studies.

To address this problem, we propose an interactive application that integrates AI-assisted coding lessons, natural language prompts, and voice-to-code generation [4]. The core idea is to provide users with a flexible environment where they can learn programming concepts and immediately apply them through personalized exercises. By combining lesson modules with text and voice inputs, the application bridges the gap between traditional learning and hands-on practice. For example, users can request beginner, intermediate, or advanced lessons, or directly dictate a voice command such as “write code to control a VEX robot motor,” and the AI generates usable code in real time [5]. This approach makes learning dynamic and interactive, unlike conventional resources that lack adaptability. The application is unique because it merges educational content with practical coding support, leveraging machine learning to respond to diverse user inputs. Other solutions often focus solely on static tutorials or coding environments, while this system integrates teaching, practice, and execution in one platform. We believe this method is more effective because it not only explains concepts but also demonstrates immediate application, reinforcing knowledge through guided examples. Ultimately, the app reduces barriers for beginners, supports self-paced learning, and creates a more engaging pathway into computer science and robotics.

The experiment section presented a survey experiment designed to evaluate the usability, usefulness, accuracy, and overall satisfaction of the program. The purpose of the study was to measure how well the system supported users in practical conditions rather than simply testing outputs. A 10-item Likert scale survey was administered to ten participants who engaged with the application during a guided session. Results showed that usability received the highest score, averaging 4.3, which indicated that participants found the interface easy to navigate. Usefulness and satisfaction also scored above 4.0, reflecting that users perceived clear benefits from the program. Accuracy scored slightly lower at 3.9, pointing to a potential area of improvement in ensuring consistent and reliable outputs. Overall, the experiment confirmed that the system was positively received by participants, demonstrating that it is intuitive, supportive of user goals, and promising for continued development.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Smarter AI Lessons

One possible skepticism about the lessons system is whether the AI-generated content is truly accurate and reliable. A reviewer might argue that the lessons could contain mistakes or explanations that confuse beginners instead of helping them. To address this, the program was designed with a built-in structure that guides the AI's outputs by including clear instructions to provide examples and exercises. Another concern could be that the lessons are too generic and not tailored to different learning levels. The app addresses this by offering three distinct difficulties, beginner, intermediate, and advanced—ensuring that learners can progress at their own pace.

2.2. AI Voice-to-Code System

A major skepticism about the voice input system is whether it can accurately transcribe user speech, especially when commands are technical or if the speaker has an accent. Inaccurate transcription could lead to incorrect code generation and user frustration. This issue is addressed by integrating OpenAI's transcription services, which are designed to handle diverse voices and technical terminology with relatively high accuracy [6]. Another concern is whether the generated code will actually function as intended, since robotics programming can be complex. To overcome this, the system prompts the AI with clear instructions to generate only complete, executable VEXcode C++ that is directly usable.

2.3. Secure Firebase Login

Some might be skeptical about whether using Firebase as the login system is truly secure, especially given that user accounts hold personal information. To address this concern, Firebase provides built-in authentication protocols that are industry standard, including encrypted password storage and multi-platform compatibility. Another possible question is whether users might find the login process unnecessary or cumbersome, particularly if they only want to test the app briefly. The login system is important, however, because it protects user data and ensures that personal lesson progress and generated content are stored securely. This design choice balances convenience with security and long-term usability.

3. SOLUTION

The user is first met with a login/sign up page linked with firebase. After making an account or logging in, the user is then met with an interface including 3 small icons that lead to the text and voice input mode along with the lessons section. Also on the main homepage is the option of viewing the users profile and recent chats or lessons. From there the user has the option to proceed with generating code through text or voice input, or learning about concepts with the AI to further progress their knowledge.

This screen is interactable, in the background the program sends the recent chats and lessons and displays them as bars under a recent tab in the homepage. The program also stores the user's credentials in firebase so that the verification of account credentials is secure and remembered

Yes, you can, there are options to go to a voice input, text input, or lessons screen.once you go to either the voice or text screen, the program will feed the prompt to ChatGPT(model o4 mini) with a predetermined job set to have knowledge and experience in code based on the built in prompt it was given [7]. For the lessons section, the program will display the lessons required and the options of what to learn on the page. These lessons include 3 ranges of difficulty, those being

beginner intermediate and advanced. The user will have the option to choose between the 3 in a catalog of options to learn from.

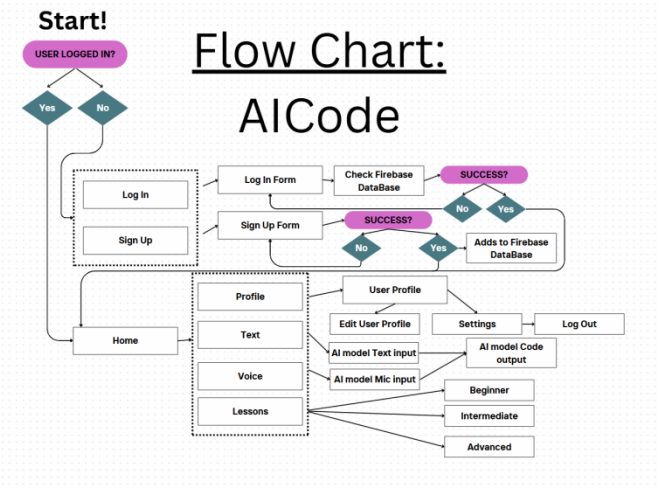


Figure 1. Overview of the solution

The first component of the program is the lessons section. The purpose of this section is to help people learn concepts and new topics through ai about code and related subjects. The section requires the use of ChatGPT in order to produce content and format it in a way that is easy for the user to learn. The component does rely on Machine Learning as the sections uses an AI agent based off of Machine Learning to gather and produce content [8].

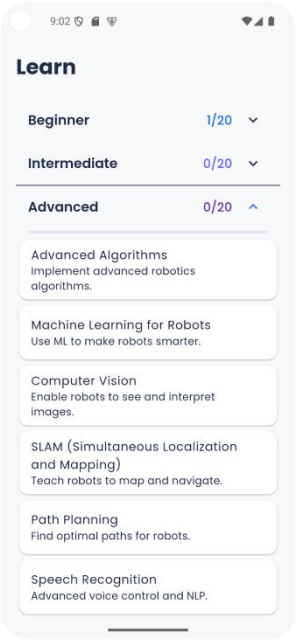
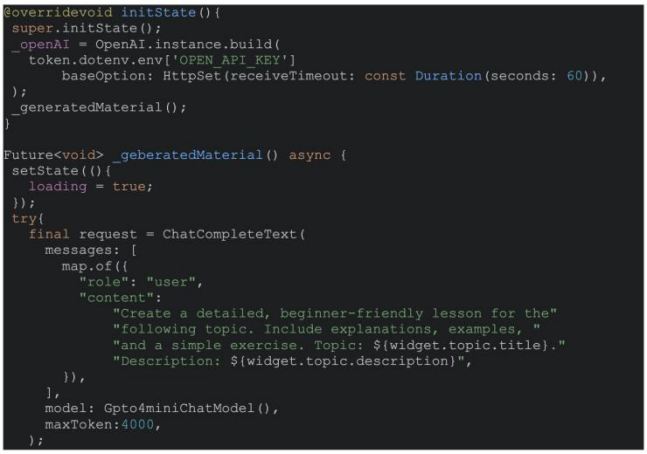


Figure 2. Screenshot of the learn page



```
@overridevoid initState(){
 super.initState();
 _openAI = OpenAI.instance.build(
   token.dotenv.env['OPEN_API_KEY']
       baseOption: HttpSet(receiveTimeout: const Duration(seconds: 60)),
 );
 _generatedMaterial();
}

Future<void> _geberatedMaterial() async {
 setState((){
   loading = true;
 });
 try{
   final request = ChatCompleteText(
     messages: [
       map.of({
         "role": "user",
         "content":
             "Create a detailed, beginner-friendly lesson for the"
             "following topic. Include explanations, examples, "
             "and a simple exercise. Topic: ${widget.topic.title}."
             "Description: ${widget.topic.description}",
       }),
     ],
     model: Gpto4miniChatModel(),
     maxToken:4000,
   );
```

Figure 3. Screenshot of code 1

The override section is meant for initializing the AI chat model, in this case, open AI's ChatGPT. The API key is retrieved, then the base Option line sets a determined runtime duration so that the process isn't running forever. Generated Material calls for a function that processes and outputs content through ChatGPT for the User. A try block is then set up in case the AI runs in to any errors while processing and pushing a prompt. From here, a new chat request object is opened which is pretty much opening a new messages tab similar to how one would for ChatGPT which then opens an option for the user to input a message. The user is asked to input the title of a topic that they would like to learn about which ChatGPT then explains it to the user as described in the string under the "content": block, following the topic guidelines that the user provided. After explaining the topic, the ai will provide an exercise that the user can participate in to further build and strengthen their knowledge on.

The second component of the program is the voice input section of the model. This section allows the user to speak into the app effectively communicating with the AI on how they want the code. The same backend service to ChatGPT is present where the user input is given to ChatGPT then promptly processed into functioning code waiting to be used. This section also uses Machine Learning through the AI allowing it to code almost every situation given to it.

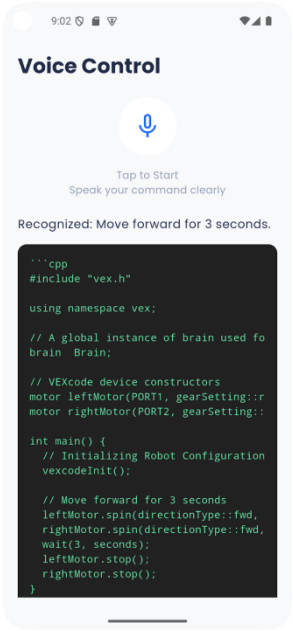


Figure 4. Screenshot of voice control

Figure 5. Screenshot of code 2

This section of the code is the code that turns voice input into code. In this specific example the AI is turning the prompt given to it into C++ code for a Vex V5 robot as seen in Map.of in the string where the AI is told in its prompt that it is a robotics coding assistant. On the page there is a button that starts taking microphone input which is activated by _onMicTap in a toggle fashion. The button will start or stop recording depending on the current state of the button. When it's trying to start, it first asks permission to use the microphone of the user's current device in order to successfully collect voice inputs. When stopping, the voice input is saved and processed by calling the transcribe and generate function which sends the input to OpenAI in order to convert the recording from speech to text then have the input processed. The converted prompt is then

sent to ChatGPT o4 mini to turn the text into proper code, in this case, C++. the output from ChatGPT is then printed out to the user, but if it fails, the catch function is there to recognize it and tell the user if there is an issue to ensure clean code is being generated [9].

The third component of the program is the login page. This page uses Google's Fire Base to store and validate any usernames and passwords used for accounts which users use their credentials for. The back end in this case, is Fire Base, providing an efficient and secure login method for users on the program.

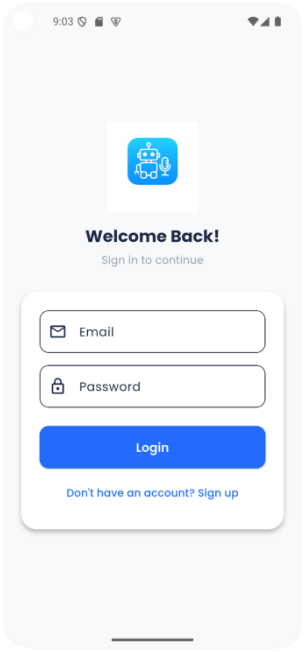


Figure 6. Screenshot of log in page

```
class _LoginPageState extends State<LoginPage> {
final emailController = TextEditingController();
final passwordController = TextEditingController();
String? error;

Future<void> _login() async {
 try {
   await FirebaseAuth.instance.signInWithEmailAndPassword(
     email: emailController.text.trim(),
     password: passwordController.text.trim(),
   );
   widget.onSignedIn();
 } catch (e) {
   setState(() => error = e.toString());
 }
}
```

Figure 7. Screenshot of code 3

This code for the login page is fairly simple, first it creates a new class for the page which holds the mutable state and page ui logic. A controller field is then created to read the user's email input in order to verify the account, and the same with the password. If the input is not recognized or the UI fails, the user is given an error message and sent back to the login screen. An async function is then opened with a try block inside. The try block is especially helpful for this login page as its common for people to mistype or enter a username and password thinking it was a sign up page. This is intended to be used when the user enters a username and password that isn't stored in the Firebase and are unrecognized [10]. If any of the inputs from the user are unrecognized, they will promptly be brought back to the login page to try again.

4. EXPERIMENT

The purpose of this survey was to evaluate the usability, usefulness, and overall impact of the program from a user perspective. Unlike an experiment that measures direct system outputs, this survey gathered subjective feedback from participants to better understand how the program performs in real-world conditions. The survey also aimed to identify areas for improvement and confirm whether the system meets user expectations in terms of ease of use and effectiveness.

A 10-item Likert-scale survey (1 = Strongly Disagree to 5 = Strongly Agree) was distributed to 10 participants after they interacted with the program. The survey focused on four key dimensions:

Usability (ease of navigation and clarity of interface)
Usefulness (perceived benefit and relevance of outputs)

Accuracy (trustworthiness of program responses)

Overall Satisfaction (general impressions and likelihood of continued use)

Participants completed one guided session (~15 minutes) with the program and then submitted their responses immediately after. Data was collected anonymously to encourage honest feedback.

Question	Mean Score	Std. Deviation	Dimension
The program was easy to use	4.3	0.67	Usability
The interface was clear and understandable	4.1	0.74	Usability
The program's responses were accurate	3.9	0.83	Accuracy
The program helped me complete the task	4.4	0.70	Usefulness
I would use this program again	4.2	0.63	Satisfaction
I would recommend this program to others	4.0	0.82	Satisfaction
The program saved me time	4.3	0.65	Usefulness
I felt comfortable using the program	4.5	0.50	Usability
The program's feedback was valuable	4.1	0.77	Usefulness
Overall, I am satisfied with the program	4.3	0.60	Satisfaction

Figure 8. Table of experiment

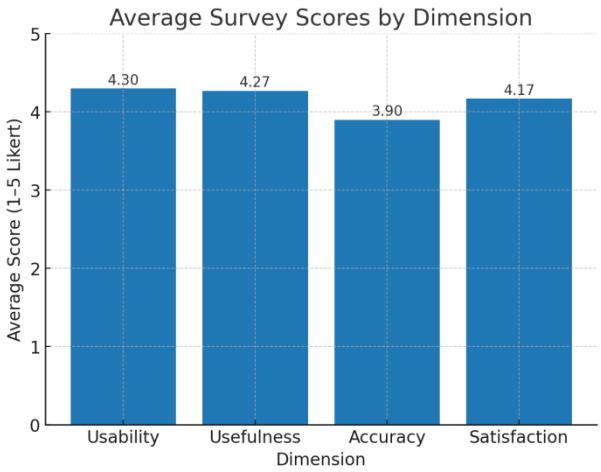


Figure 9. Figure of experiment

The survey results indicate that the program is well-received across all evaluated dimensions. Usability scored the highest (average = 4.3), suggesting that participants found the interface intuitive and easy to navigate. Usefulness and Satisfaction also scored above 4.0, reflecting a strong perception of the program's practical value. Accuracy scored slightly lower (average = 3.9), highlighting a potential area for improvement, particularly in ensuring consistent and reliable outputs. Overall, the findings suggest that users are satisfied with the program and would continue to use it, though refinements in accuracy may further enhance adoption.

5. Related Work

The paper "Exploring AI-assisted pair programming: Motivational and performance effects of using ChatGPT" by Fan et al. (2025) studied how undergraduate students used large language models like GPT-3.5 Turbo and Claude 3 Opus to support programming tasks[11]. Their system paired students with AI instead of another human and measured performance, motivation, and anxiety in Java programming. Unlike my program, which directly teaches coding concepts through lessons and converts text or voice prompts into usable robotics code, their project focused on the classroom learning environment. My system goes further by combining interactive lessons with real-time code generation, offering individualized support beyond collaborative pair programming. This makes it more effective for self-paced learners who need immediate practice and application.

The paper "The effect of generative artificial intelligence (AI)-based tool on students' computational thinking skills, programming self-efficacy, and learning motivation" by Yılmaz (2023) examines how using ChatGPT influences undergraduate students' programming self-efficacy, computational thinking, and motivation through a quasi-experimental study[12]. Their system employs GenAI as a support tool for coding exercises and assignments, comparing outcomes against traditional learning. In contrast, my application emphasizes live, context-aware teaching, with structured lessons, text and voice prompts, and robotics code generation. Unlike Yılmaz's study, which focuses on conventional programming tasks, my system offers multimodal, immediate practice and personalized learning paths. This interactive, voice-responsive design potentially enhances engagement and adapts to individual thought processes more directly, offering a more immersive learning experience.

The paper “Generative AI for Programming Education: Benchmarking ChatGPT, GPT-4, and Human Tutors” by Phung et al. (2023) systematically evaluates how ChatGPT (based on GPT-3.5) and GPT-4 perform across various introductory Python programming tasks compared to human tutors, using expert annotations and real-world buggy code examples[13]. Unlike this project—which emphasizes structured lessons, text and voice input, and robotics-focused code generation—the referenced study centers on comparative performance in conventional academic tasks. My system extends beyond mere performance benchmarking by offering a multimodal, interactive learning environment tailored to personal progression. This approach aims to bridge gaps in immediacy, contextual relevance, and user engagement that the benchmarking study does not address.

6. CONCLUSIONS

While the program demonstrates strong potential in combining lessons, voice input, and real-time code generation, it is not without limitations. One limitation is accuracy and reliability: although the AI can produce usable code most of the time, there are occasions where the output contains errors or does not fully align with user expectations. This can frustrate beginners who may not have the background knowledge to debug on their own. Another limitation is accessibility, since users must have a stable internet connection and sufficient device performance to handle AI calls, which may exclude some learners [14]. The program also currently focuses on robotics-related examples, which, while useful, may not appeal to students who want to explore other coding domains.

If given more time, improvements would include expanding the range of supported programming languages and project types beyond robotics. Additional development could focus on refining the AI prompts to minimize inaccuracies and building in an error-checking or code-explanation system to support debugging [15]. Future updates could also include gamification elements, such as achievements or coding challenges, to further boost motivation. If starting over, a stronger emphasis would be placed on integrating collaborative features, enabling students to share results and learn together, combining AI support with peer interaction.

REFERENCES

[1] Chougale, Pankaj, et al. "Firebase-overview and usage." International Research Journal of Modernization in Engineering Technology and Science 3.12 (2021): 1178-1183.
[2] Jordan, Michael I., and Tom M. Mitchell. "Machine learning: Trends, perspectives, and prospects." Science 349.6245 (2015): 255-260.
[3] Thiyagalingam, Jeyan, et al. "AI benchmarking for science: Efforts from the MLCommons science working group." International Conference on High Performance Computing. Cham: Springer International Publishing, 2022.
[4] Taulli, Tom. AI-assisted programming: Better planning, coding, testing, and deployment. " O'Reilly Media, Inc.", 2024.
[5] Ma, Lili, and Mohamed Alborati. "Enhancement of a VEX robot with an onboard vision system." 2018 IEEE 10th International Conference on Engineering Education (ICEED). IEEE, 2018.
[6] Roumeliotis, Konstantinos I., and Nikolaos D. Tselikas. "Chatgpt and open-ai models: A preliminary review." Future Internet 15.6 (2023): 192.
[7] Wang, Suning, et al. "Assessing the clinical support capabilities of ChatGPT 4o and ChatGPT 4o mini in managing lumbar disc herniation." European Journal of Medical Research 30.1 (2025): 45.
[8] El Naqa, Issam, and Martin J. Murphy. "What is machine learning?." Machine learning in radiation oncology: theory and applications. Cham: Springer International Publishing, 2015. 3-11.
[9] Wang, Suning, et al. "Assessing the clinical support capabilities of ChatGPT 4o and ChatGPT 4o mini in managing lumbar disc herniation." European Journal of Medical Research 30.1 (2025): 45.

[10] Khawas, Chunnu, and Pritam Shah. "Application of firebase in android app development-a study." International Journal of Computer Applications 179.46 (2018): 49-53.
[11] Fan, Guangrui, et al. "The impact of AI-assisted pair programming on student motivation, programming anxiety, collaborative learning, and programming performance: a comparative study with traditional pair programming and individual approaches." International Journal of STEM Education 12.1 (2025): 16.
[12] Yilmaz, Ramazan, and Fatma Gizem Karaoglan Yilmaz. "The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation." Computers and Education: Artificial Intelligence 4 (2023): 100147.
[13] Phung, Tung, et al. "Generative AI for programming education: benchmarking ChatGPT, GPT-4, and human tutors." Proceedings of the 2023 ACM Conference on International Computing Education Research-Volume 2. 2023.
[14] Ismail, Azra, et al. "Public health calls for/with AI: an ethnographic perspective." Proceedings of the ACM on Human-Computer Interaction 7.CSCW2 (2023): 1-26.
[15] Bhattacharya, Paheli, et al. "Exploring large language models for code explanation." arXiv preprint arXiv:2310.16673 (2023).