Enhancing Programming Education with AI-Powered Study Support Tool

Transforming how students learn coding through adventure, personalized feedback, and study materials

Submitted by:
Solomon Mengesha Kebede
8336583
Máté Csizmadia
8522746

Under the guidance of:
Dr. Daniele Di Mitri

Goethe University
February 18, 2025

Abstract

Programming education for young learners presents unique challenges, including maintaining engagement, providing personalized feedback, and fostering a structured learning path. This paper explores the development of an AI-powered study support tool designed to transform how students aged 8 to 18 learn coding. By integrating engaging visual elements, dynamic animations, and an intuitive progress map, the tool offers an adventure-based learning experience. The system employs AI classifiers, including binary and multiclass classifiers, to assess coding submissions and categorize errors while neural networks personalize learning challenges based on user progress. A key feature of the tool is its structured feedback system, which provides instant, meaningful guidance to learners, helping them understand and correct their mistakes. Furthermore, the tool adopts a game-like learning approach, incorporating levels, challenges, and rewards to sustain motivation and encourage continuous learning. Ethical considerations, including privacy protection and inclusive design, are embedded within the system to ensure a safe learning environment. This paper discusses the pedagogical foundation, technical implementation, and ethical implications of AI-driven programming education, demonstrating how such tools can enhance accessibility and effectiveness in coding instruction.

Table of Contents

1 Introduction

The increasing importance of programming skills in the digital age has led to a growing demand for innovative and effective educational tools, especially for young learners. Traditional approaches to programming education often rely on static materials and one-size-fits-all curricula, which can hinder engagement and personalized learning. To address these challenges, we propose an AI-powered study support tool that enhances the learning experience through interactive adventure-based education, real-time feedback, and adaptive learning paths.

Our tool is designed for children and teenagers (ages 8 to 18) with little or no prior coding experience. It leverages AI-driven classifiers to evaluate code submissions, distinguish between syntax and logic errors, and provide targeted feedback. Additionally, neural networks are employed to personalize learning challenges based on individual progress, ensuring that students remain motivated and adequately challenged.

A defining feature of our approach is the integration of a game-like structure that transforms learning into an engaging journey. Learners progress through coding challenges represented as levels, overcoming obstacles, earning rewards, and unlocking new learning paths. The interactive progress map visualizes achievements and motivates users to advance. Furthermore, a structured feedback system is embedded to provide immediate and meaningful guidance. By highlighting errors, explaining concepts, and offering hints, this system helps students refine their coding skills in a constructive and supportive manner.

The platform's design principles prioritize visual engagement, accessibility, and ethical considerations, such as privacy protection and inclusivity. By combining AI-driven learning with an interactive, game-inspired approach, we aim to make coding education more accessible, engaging, and effective for young learners.

This paper details the development of this AI-powered educational tool, covering its pedagogical foundation, technical architecture, feedback mechanisms, and ethical framework. By integrating AI into programming education, we demonstrate how adaptive learning environments can enhance student engagement and improve coding proficiency.

2 Features and Implementation

Our AI-powered study support tool combines intelligent feedback mechanisms with an adventure-based learning environment to enhance programming education for young learners. This section outlines the core classifiers used to evaluate coding submissions as well as students according to theri lever of competency. Moreover, the game-like design features that make learning engaging and motivating.

2.1 Classifiers

The tool employs machine learning classifiers to evaluate coding submissions, detect errors, and personalize the learning experience. The classification system consists of three key components:

1. Binary Classifier – Determines if the submission is correct or incorrect.
2. Multiclass Classifier – Identifies and categorizes the error type.
3. Neural Networks – Personalizes learning challenges based on student progress.

2.1.1 Binary Classifiers

Binary classifiers are machine learning models used to classify data into one of two categories, such as $\{0, 1\}$ or $\{\text{correct}, \text{incorrect}\}$. In our project, this concept will be applied to tasks like detecting whether a student's code submission is correct or incorrect, or determining if a user is ready for advanced challenges.

Mathematical Representation

A binary classifier typically outputs a probability $p \in [0, 1]$, which represents the likelihood of the input belonging to class 1. The decision boundary is defined by a threshold t, such that:

$$\hat{y} = \begin{cases} 1 & \text{if } p \geq t \\ 0 & \text{if } p < t \end{cases}$$

The output $\hat{y}$ represents the predicted class.

Loss Function

For binary classification, the most commonly used loss function is **Binary Cross-Entropy (BCE)**:

$$L = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

Where:

- y_i: True label (0 or 1).
- p_i: Predicted probability for class 1.
- N: Number of samples.

This loss penalizes incorrect predictions more heavily as probabilities deviate from the true labels. [1]

Example: Error Detection in Block-Based Coding

Scenario: A student builds a block-based program with potential syntax or logical errors. A binary classifier determines whether the submission is correct (1) or contains errors (0).

Implementation: Using a logistic regression model:

```
import numpy as np
from sklearn.linear_model import LogisticRegression

# Example training data (features: [number_of_blocks, loops_used], labels: [correct/inc
X_train = np.array([[10, 1], [15, 0], [12, 2], [20, 3]])
y_train = np.array([1, 0, 1, 0])

# Train the model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict on new data
X_test = np.array([[14, 2]])  # New submission
prediction = model.predict(X_test)
print("Prediction:", prediction)
```

The model predicts whether the submission is correct ($y = 1$) or contains errors ($y = 0$).

Evaluation Metrics

To evaluate the performance of binary classifiers :

- **Precision:** Proportion of correctly identified errors among all predicted errors.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- **Recall:** Proportion of actual errors correctly identified.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- **F_1-Score:** Harmonic mean of precision and recall.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

These metrics ensure that the classifier minimizes false positives (flagging correct submissions as errors) and false negatives (missing actual errors).[2]

Use Cases in our Project

Here are some ways binary classifiers can be integrated into our AI-powered coding study tool:

1. **Real-Time Feedback:** Detect syntax or logic errors in block-based coding submissions.
   - Input: Features like number of blocks, loops used, and conditions present.
   - Output: Binary classification ($0 = error$, $1 = correct$).
2. **Adaptive Learning Paths:** Classify users into skill levels.
   - Input: Performance metrics (e.g., time taken per task, error rate).
   - Output: Binary classification ($0 = beginner$, $1 = advanced$).
3. **Ethical Safeguards:** Flag inappropriate content.
   - Input: Text from user-generated content.
   - Output: Binary classification ($0 = safe$, $1 = inappropriatecontentdetected$).

Ethical Considerations

When implementing binary classifiers, it is important to consider:

- Training on diverse datasets to avoid bias in predictions.
- Anonymizing user data to maintain privacy and comply with regulations like GDPR.(we will discuss more in the coming chapters)

2.1.2 Multiclass Classifier: Categorizing Errors and levels

Multiclass classifiers are machine learning models designed to categorize data into one of three or more classes. Unlike binary classifiers, which deal with two classes (e.g., {0, 1}), multiclass classifiers handle problems where the target variable can take on multiple discrete values. In our project, they can be used for tasks such as classifying students into skill levels (e.g., *Beginner*, *Intermediate*, *Advanced*) or categorizing errors in code submissions (e.g., *Syntax Error*, *Logic Error*, *Runtime Error*).

Mathematical Representation

For a dataset with k classes, the goal of a multiclass classifier is to assign an input $\mathbf{x}$ to one of the classes $\{C_1, C_2, ..., C_k\}$. The model outputs a probability distribution over all k classes:

$$P(C_i|\mathbf{x}) = \frac{\exp(z_i)}{\sum_{j=1}^{k} \exp(z_j)}, \quad i = 1, 2, ..., k$$

Here:

- z_i: The raw score (logit) for class C_i.
- $\exp(z_i)$: The exponential function applied to the logit.
- $\sum_{j=1}^{k} \exp(z_j)$: Normalization term ensuring probabilities sum to 1.

The predicted class $\hat{y}$ is determined by:

$$\hat{y} = \arg\max_i P(C_i|\mathbf{x})$$

Approaches to Multiclass Classification

There are two common strategies for implementing multiclass classification:

1. One-vs-All (OvA)

In this approach, k binary classifiers are trained, where each classifier distinguishes one class from all others. For example:

- Classifier 1: Distinguishes C_1 vs. $\{C_2, C_3, ..., C_k\}$.
- Classifier 2: Distinguishes C_2 vs. $\{C_1, C_3, ..., C_k\}$.
- And so on for all k classes.

The final prediction is made by selecting the class with the highest confidence score.

2. One-vs-One (OvO)

In this approach, $\binom{k}{2}$ binary classifiers are trained for every pair of classes. For example:

- Classifier 1: Distinguishes C_1 vs. C_2.
- Classifier 2: Distinguishes C_1 vs. C_3.
- And so on for all pairs of classes.

The final prediction is made by applying a voting mechanism across all classifiers.
—

Loss Function

[3]
For multiclass classification, the most commonly used loss function is the **Categorical Cross-Entropy**:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{k} y_{ij}\log(P(C_j|\mathbf{x}_i))$$

Where:

- y_{ij}: Binary indicator (0 or 1) if sample i belongs to class j.
- $P(C_j|\mathbf{x}_i)$: Predicted probability for class j.
- N: Total number of samples.
- k: Number of classes.

Example: Classifying Errors in Code Submissions

Let's consider a scenario where a student submits code that may contain different types of errors (*Syntax Error*, *Logic Error*, or no error). A multiclass classifier can predict the type of error based on features extracted from the submission.

Implementation [4] Using a neural network with softmax activation:

```
import numpy as np
from sklearn.neural_network import MLPClassifier

# Example training data (features: lines_of_code, loops_used, labels: error_type)
X_train = np.array([[10, 2], [15, 1], [8, 3]])
y_train = np.array([0, 1, 2])  # Labels: 0=Syntax Error, 1=Logic Error, 2=No Error

# Train the model
model = MLPClassifier(activation="softmax", max_iter=500)
model.fit(X_train, y_train)

# Predict on new data
X_test = np.array([[12, 2]])  # New submission
prediction = model.predict(X_test)
print("Prediction:", prediction)
```

[5]

The model predicts the type of error as one of the three classes.

Evaluation Metrics

[6]

To evaluate multiclass classifiers:

- **Precision** (per class):

$$\text{Precision}_i = \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Positives}_i}$$

- **Recall**(per class):

$$\text{Recall}_i = \frac{\text{True Positives}_i}{\text{True Positives}_i + \text{False Negatives}_i}$$

- **F$_1$-Score** (per class):

$$F_{1,i} = 2 \cdot \frac{\text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}$$

These metrics can be averaged using:

1. **Macro Average:** Treats all classes equally.
2. **Micro Average:** Weighs metrics by the number of samples per class.
3. **Weighted Average:**Weighs metrics by the proportion of each class in the dataset.

—

Use Cases in our Project

Multiclass classifiers will be integrated into our AI-powered coding study tool as follow:

1. Error Categorization: - Input: Features from code submissions. - Output: Predicted error type (*Syntax Error*, *Logic Error*, etc.).
2. Skill Level Classification: - Input: Performance metrics (e.g., time taken per task). - Output: Predicted skill level (*Beginner*, *Intermediate*, or Advanced).
3. Challenge Recommendations: - Input: User progress and preferences. - Output: Recommended challenge category (*Loops*, Functions).

3 Functionality of Study Support Tool(SST)

In this Chapter we will introduce two mockup user stories that has been created in a guided assignment during the prepartaion of this project. Afterwards we will discuss the adventures theme and design of our app that could motivate childrens to learn coding in a fun and interesting way. We will elaborate our design concept using different sample designs that includes block coding for children at early ages.

3.1 Mockup User Story 1: Tommy's Search for the Right Platform

Fourteen-year-old Tommy had always been fascinated by technology. He spent hours watching videos about robots, video games, and how apps were made. The idea that someone could write lines of code and bring a game to life amazed him.

But when he tried to learn coding himself, it wasn't what he expected.

He started with YouTube tutorials, but the explanations felt too fast or too technical. He downloaded a coding textbook, but the pages were filled with complex terms and long blocks of text—it felt nothing like the exciting, hands-on experience he had imagined.

After a few frustrating attempts, he wondered:

- **"Is learning to code supposed to be this boring?"**

One evening, Tommy searched online:

- `\Fun way to learn coding for beginners"`

Among the results, something caught his eye—a platform that turned coding into an adventure. It wasn't just another course; it promised a journey where he could solve challenges, earn rewards, and level up like in a video game.

With exitment, he clicked on it.

- **"Could this finally be the fun way to learn coding I've been looking for?"** he thought, as he signed up.

3.2 Mockup User Story 2: A Parent's Search for Answers

David had always encouraged his eight-year-old daughter, Lily, to be curious. And Lily had a lot of questions.

- "Daddy, how does my tablet know I'm touching it?"

- ”Who made my favorite game? Can I make one too?”
- ”What is a loop?”

At first, David loved answering her. He explained how touchscreens work and how video games are made using code. But as Lily’s questions became more complex, David realized something:

- He didn’t know how to explain programming in a way an eight-year-old would understand.

He tried looking for beginner coding lessons, but everything he found was either:

- **Too advanced** – filled with programming jargon she wouldn’t understand.
- **Too dry** – long, text-heavy lessons that didn’t match her energy and excitement.

He didn’t want her to lose interest, so he searched:

- ”Coding for kids – fun and easy”

That’s when he found something different—an app designed specifically for young learners, promising a fun, interactive introduction to coding.

- ”This might be what I need,” David thought. ”Maybe Lily can finally start learning in a way that makes sense to her.”

Excited, he downloaded the app and called Lily over.

3.3 Design Features

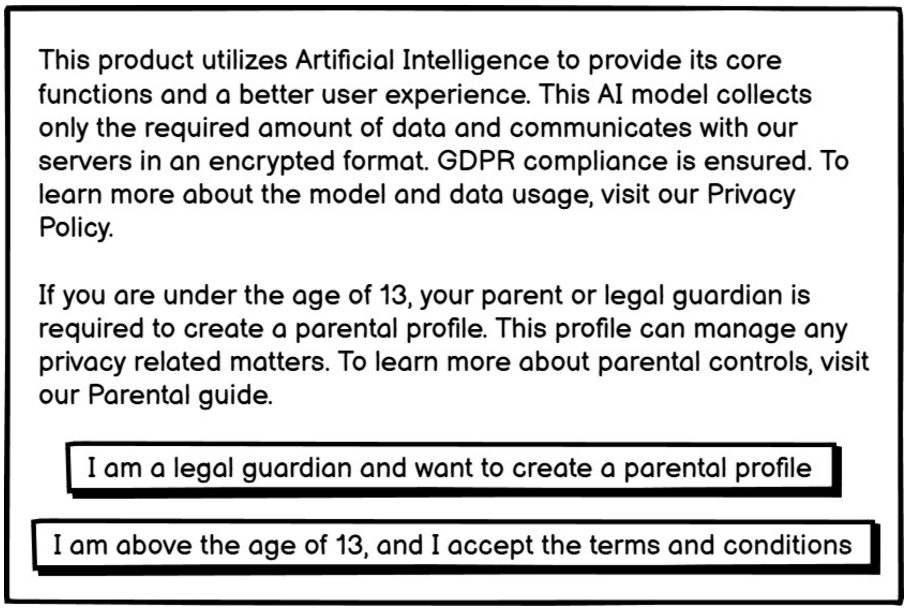
To create an engaging and effective learning experience, the platform incorporates interactive, game-inspired elements that motivate students. We have designed both web and mobile app versions of mockup UIs, allowing learners to access and practice coding anytime, anywhere. This makes our application accessible without limitations. The platform offers two types of UIs tailored for different age groups:

Block-Based Coding mockup UI:
Designed for beginners and children entering the world of coding without prior knowledge, this UI uses block-based coding to simplify the learning process. It helps young learners easily grasp programming concepts.

Text-Based Coding mockup UI:
Building on the foundational skills acquired through the block-based UI, this interface allows students to practice coding by writing and executing code directly. They can see the results in the console, providing a seamless transition to more advanced coding practices.

3.3.1 Starting Page

Either in Webapp or mobileapp , when new users open the app they will get the first window which exmplains the data processing agrement and if they are belw 13 years old parents should read and agree the policy , which is writen based of GDPR. We will discuss more about privacy protection and ethics in the chapter 5.


This product utilizes Artificial Intelligence to provide its core functions and a better user experience. This AI model collects only the required amount of data and communicates with our servers in an encrypted format. GDPR compliance is ensured. To learn more about the model and data usage, visit our Privacy Policy.

If you are under the age of 13, your parent or legal guardian is required to create a parental profile. This profile can manage any privacy related matters. To learn more about parental controls, visit our Parental guide.

I am a legal guardian and want to create a parental profile
I am above the age of 13, and I accept the terms and conditions

Figure 1: Data collection and privacy agremment

3.3.2 To Begin

After the agreement, users will be asked to provide information about their skill level by answering a few questions based on their preference. There are three options available: users can indicate that they are beginners, have some knowledge, or if they are unsure, they can take a short quiz to assess their level. Following this assessment, our AI (Multi-Class classifier) will categorize learners into three levels: beginner, intermediate, or advanced.

In our app, we offer a diverse selection of avatars that serve as mentors for children throughout their learning journey. These avatars are designed to guide, support, and motivate young learners as they progress through various coding challenges and levels.

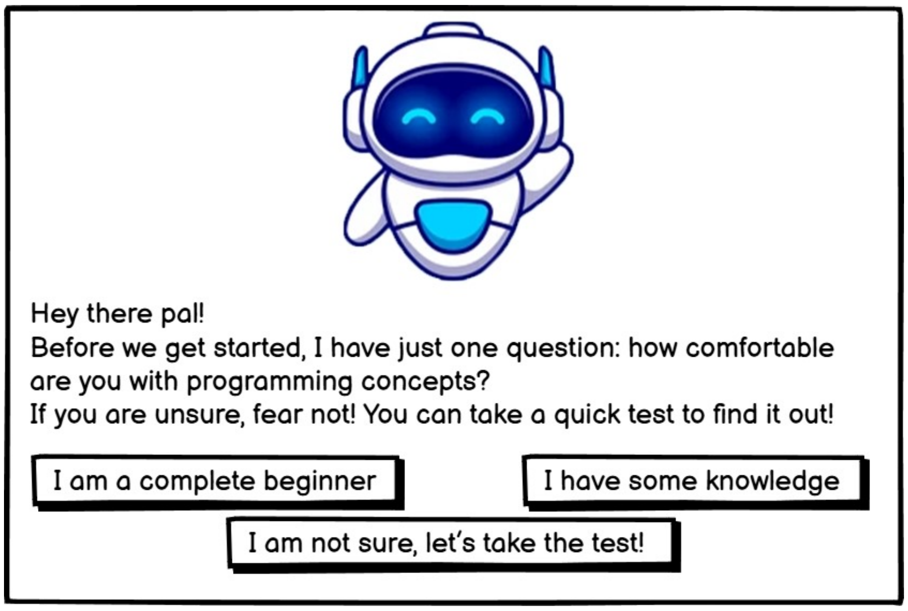

Hey there pal!
Before we get started, I have just one question: how comfortable are you with programming concepts?
If you are unsure, fear not! You can take a quick test to find it out!

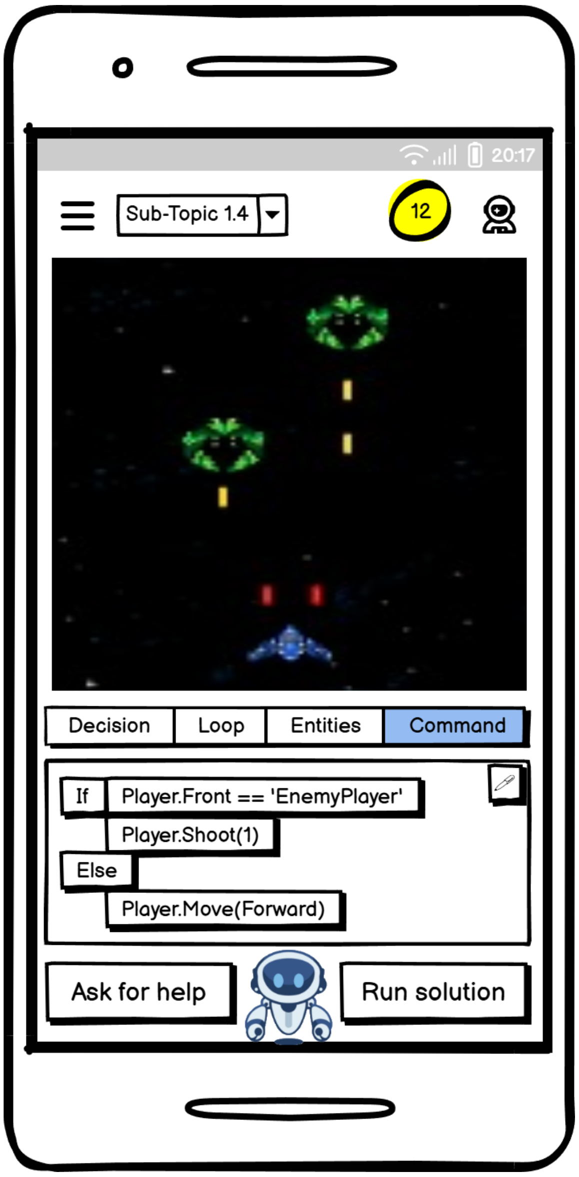
Figure 2: provide information about the level of skill

3.3.3 Block Coding

Block coding, also known as block-based programming or visual programming, is an approach to coding that uses graphical blocks to represent code concepts. Instead of writing traditional text-based code, users create programs by snapping together blocks, which fit together like puzzle pieces.

Benefits of Block Coding:

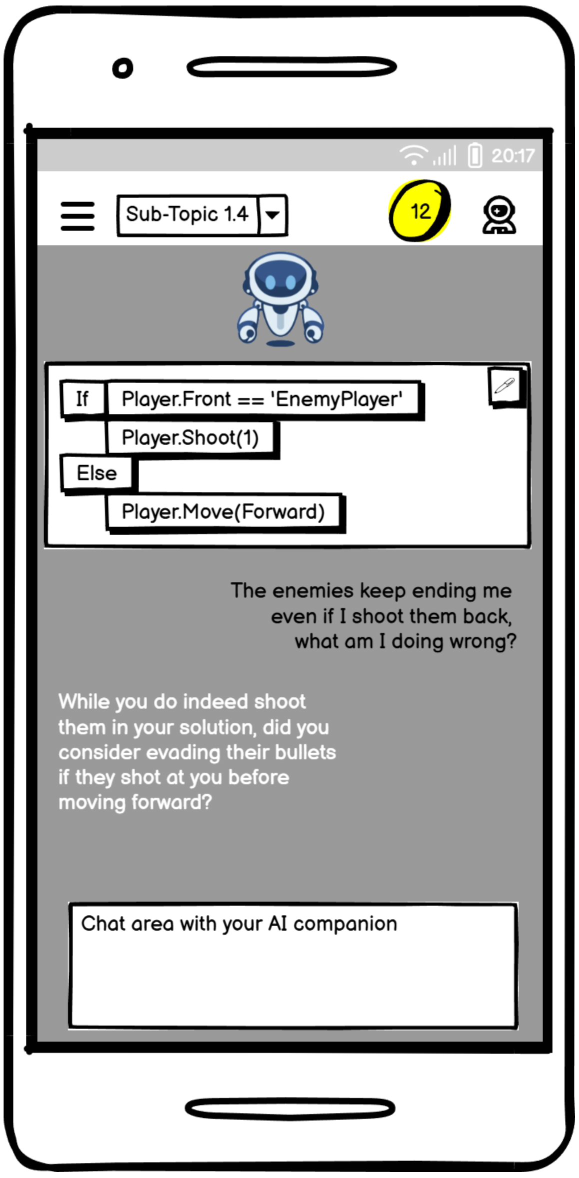
- **Engaging and Fun:** The visual and interactive nature of block coding makes it engaging and fun for learners.
- **Builds Confidence:** As learners successfully create programs, they gain confidence and are more likely to explore further coding.
- **Foundation for Text-Based Coding:** Once learners are comfortable with block coding, they can transition to text-based coding more easily, as they already understand the underlying logic. Block coding is a great way to introduce the world of programming to beginners and young learners, making the learning process accessible and enjoyable.



```
If   Player.Front == 'EnemyPlayer'
     Player.Shoot(1)
Else
     Player.Move(Forward)
```

Figure 3: Mobile Version of Block coding

Whenever learners face a challenging task and cannot pass a level, they can ask a robot mentor for assistance. If they successfully rearrange the blocks and create a correct logic, the animation above will activate, moving the robot to attack enemies. This is followed by a celebration and reward. Whenever mistakes are made, the robot will never give discouraging comments; instead, it will provide encouraging feedback. (we will discuss feedback in the next chapter)



```
If    Player.Front == 'EnemyPlayer'
      Player.Shoot(1)
Else
      Player.Move(Forward)
```

The enemies keep ending me even if I shoot them back, what am I doing wrong?

While you do indeed shoot them in your solution, did you consider evading their bullets if they shot at you before moving forward?

Figure 4: Asking For Help

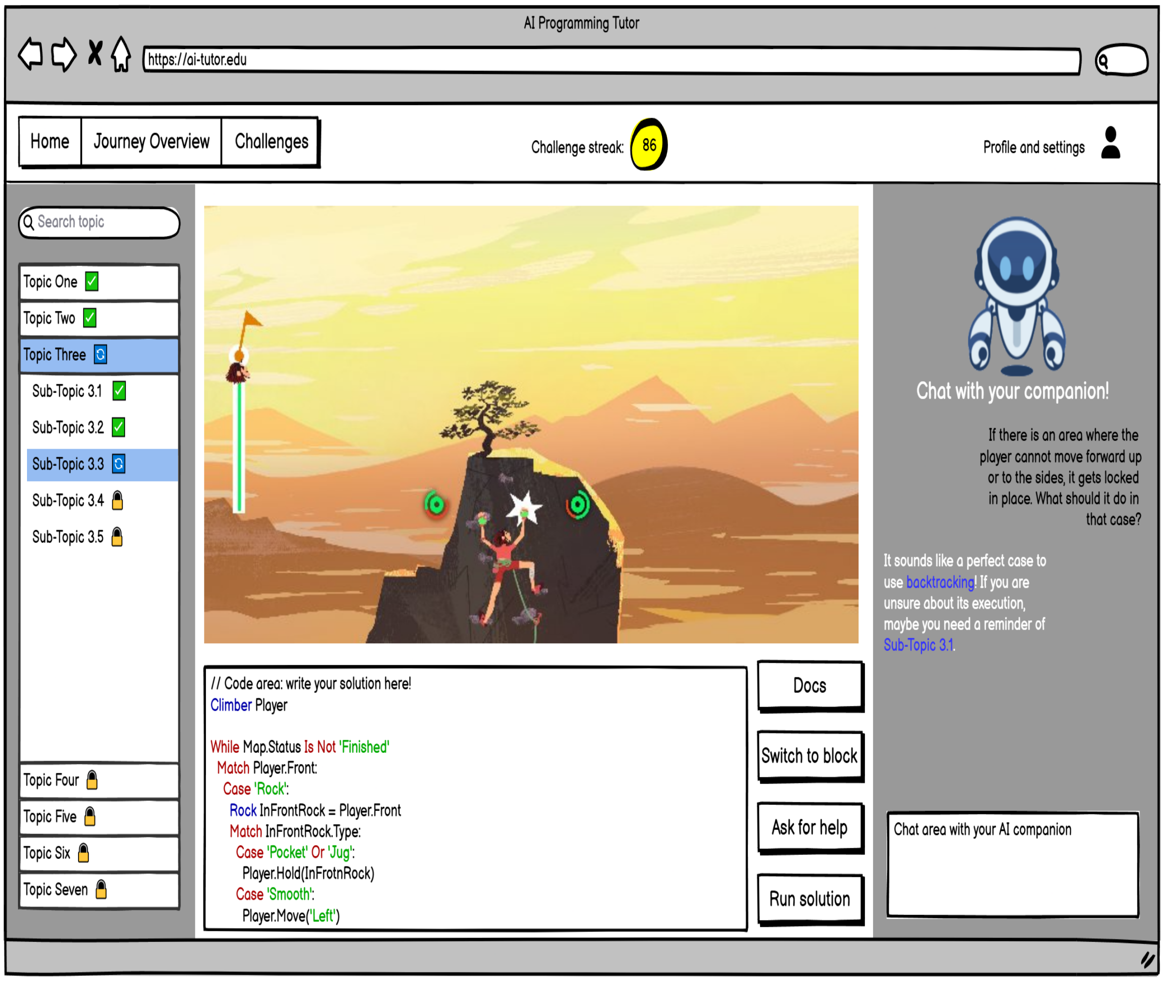
3.3.4 Text based Coding: Web app

This section of the application is designed for intermediate-level learners, providing them with the tools to write their own code and practice what they have been learning so far. The web app features an interactive and adventure-themed interface, drawing inspiration from concepts like climbing a mountain.

Key Features of the Web App:

- **Interactive Coding Environment:** Learners can write, edit, and run their own code directly within the app. Real-time feedback is provided, allowing users to see the results of their code immediately in the console.

- **Adventure-Themed Progression:** The learning journey is visualized as an adventure, with learners climbing higher up a mountain each time they complete a challenge. Each level presents new coding challenges that must be solved to advance to the next stage. Visual and animated feedback celebrates their achievements, making the learning process engaging and rewarding.
- **Personalized Learning Paths:** The app adapts to the learner's skill level, offering challenges that are appropriate for their current abilities. Users can choose to revisit previous levels to improve their scores and reinforce their understanding.
- **Gamified Experience:** The app incorporates game elements such as points, badges, and rewards to motivate learners. Completing challenges and earning rewards unlocks new levels and content, maintaining a sense of excitement and progression.
- **Mentor Avatars(AI companion):** Learners are guided by mentor avatars that provide tips, encouragement, and support throughout their journey. These avatars offer personalized feedback, suggest the next topic by accessing their progress and help users stay motivated, even when facing difficult challenges.
- **Accessibility and Convenience:** The web app is accessible from any device with an internet connection, allowing learners to practice coding anytime, anywhere. The user-friendly interface ensures that learners can easily navigate the app and focus on their coding practice.



```
// Code area: write your solution here!
Climber Player

While Map.Status Is Not 'Finished'
 Match Player.Front:
  Case 'Rock':
   Rock InFrontRock = Player.Front
   Match InFrontRock.Type:
    Case 'Pocket' Or 'Jug':
     Player.Hold(InFrotnRock)
    Case 'Smooth':
     Player.Move('Left')
```

If there is an area where the player cannot move forward up or to the sides, it gets locked in place. What should it do in that case?

It sounds like a perfect case to use backtracking! If you are unsure about its execution, maybe you need a reminder of Sub-Topic 3.1.

Figure 5: Webapp version

3.3.5 Future Enhancement

Taking into account the rapidly changing technology landscape, our tool will be subject to continuous updates, incorporating new programming languages, more games, and additional challenges. We aim to keep our platform relevant and engaging by constantly enhancing and expanding its features.

Continuous Updates

- The platform will be regularly updated to include the latest programming languages and tools, ensuring that learners have access to cutting-edge technology and industry trends.
- New games and challenges will be added to keep the learning experience fresh and exciting, catering to various skill levels and interests.

Strategic Partnerships

- We plan to establish partnerships with key stakeholders, including AI tools such as ChatGPT and other technology providers. These collaborations will enhance the app's capabilities and ensure it remains up-to-date with the latest advancements.
- By collaborating with industry leaders, educational institutions, and content creators, we aim to offer a comprehensive and enriched learning experience for our users.

Enhanced Efficiency and Engagement

- Integrating AI-powered features, such as personalized feedback, adaptive learning paths, and intelligent tutoring systems, will make the app more efficient and responsive to individual learners' needs.
- Gamified elements, interactive tutorials, and real-time problem-solving activities will engage users, fostering a love for coding and continuous learning.

Global Accessibility

- The app will be designed to be accessible globally, supporting multiple languages and catering to diverse educational needs. This ensures that learners from different backgrounds can benefit from the platform.

By implementing these strategies, our platform will remain at the forefront of programming education, offering learners an effective, engaging, and up-to-date tool to enhance their coding skills.

4 Feedback

The effectiveness of AI-powered study support tools in programming education depends heavily on how feedback is exchanged between the system and the learner. A well-designed feedback mechanism ensures that students receive meaningful insights on their coding attempts while also enabling the AI to improve by learning from user interactions. This two-way feedback process is best understood through the concept of mental models, which shape how learners perceive, interpret, and internalize programming concepts. Mental models are simplified representations of complex systems that help individuals predict outcomes and solve problems. In programming education, mental models guide learners in understanding abstract concepts such as loops, conditionals, and algorithms. By aligning AI-driven feedback with learners' evolving mental models, we can create a more intuitive and effective learning experience.[7] [8]

4.1 AI-to-User Feedback: Enhancing Understanding

[12] The primary role of AI in programming education is to provide structured feedback that aids learners in debugging their code, refining their logic, and improving overall comprehension. The AI-to-user feedback in our system is categorized into three key components:

4.1.1 Binary Feedback: Correct vs. Incorrect Submissions

The simplest form of AI feedback is a binary classification that distinguishes between correct and incorrect code submissions. This provides immediate validation for learners, allowing them to recognize whether their solution aligns with the expected output. Example:

- A student submits a Python program to calculate the factorial of a number. The AI checks the output against test cases:
- If correct: "Great job! Your solution works perfectly!"
- If incorrect: "You are not there yet. Try revisiting your loop structure."

Binary feedback serves as the foundation for great forms of guidance.

4.1.2 Categorized Error Feedback: Syntax vs. Logic Issues , level of skills

To go beyond basic correctness, our system employs a multiclass classifier that categorizes mistakes into specific types:

- Syntax Errors: Issues like missing semicolons, incorrect indentation, or misspelled keywords.
- Logic Errors: Incorrect loops, misused conditionals, or flawed algorithm implementation.
- Runtime Errors: Problems such as division by zero or accessing undefined variables.
- The system to adapt to individual learning curves by analyzing past interactions and dynamically adjusting future tasks.

By breaking errors into categories, the AI provides more informative hints, guiding students toward appropriate corrections without directly giving away the solution. Example: A student writes a program to find prime numbers but forgets to handle edge cases (e.g., input = 1).

- The AI categorizes this as a logic error and offers targeted feedback: "There is something you should check, for edge cases like 1. Consider adding an additional condition to handle such inputs."
- Adaptive challenges ensure that students remain engaged without feeling overwhelmed or under-challenged.

4.2 User-to-AI Feedback: Refining AI Assistance

Effective learning tools should not be static; they must evolve based on user interactions. Our system integrates multiple ways for learners to provide feedback to AI, enabling continuous refinement of teaching strategies.

4.2.1 Correction Recognition

The AI tracks how learners correct their mistakes. If many students resolve an error in a specific way, the system can infer improved guidance strategies and refine its feedback accordingly. Example:
If most students fix a syntax error by adding a missing colon (':') after an 'if' statement, the AI could preemptively suggest this correction for future users encountering similar issues.

4.2.2 Explicit User Ratings and Feedback Forms

Students can rate the AI-generated hints as "helpful" or "not helpful," enabling the system to gauge the effectiveness of its explanations. When multiple users find a hint unclear, the AI can modify its response strategy by offering alternative explanations or interactive tutorials. Example:
If several students rate a hint about nested loops as unhelpful, the system might replace it with step-by-step guidance or visualizations showing how nested loops work.

4.2.3 Error Resolution Tracking

The system monitors how quickly and effectively students resolve their mistakes. If repeated errors persist despite AI suggestions, this signals that a concept needs deeper reinforcement. Example:
If students frequently misuse 'break' statements within loops despite receiving hints, the system might introduce an interactive tutorial focusing on proper loop control mechanisms.

4.3 The Role of Mental Models in AI-Driven Feedback

Mental models are critical in programming education because they determine how students understand and predict code behavior. Our AI system ensures that learners build robust mental models through:

- **Clear Feedback:** Providing structured explanations that promote conceptual understanding.
- **Scaffolding Complexity:** Gradually introducing more challenging tasks to ensure students are neither overwhelmed nor under-challenged.
- **Iterative Debugging Opportunities:** Encouraging self-reflection and learning from mistakes by guiding students through debugging processes.

Example:
When teaching recursion, the AI might first present visualizations showing how function calls stack in memory (notional machine model). This helps students develop an accurate mental model of recursion before tackling complex problems. By incorporating two-way feedback loops between users and AI, the system dynamically adapts to individual learning styles and broader user trends, making programming education more effective and engaging.

4.4 Conclusion for Feedback

Integrating mental models into AI-driven feedback systems fosters a more interactive, adaptive, and learner-centric approach to programming education. By leveraging both AI-to-user and user-to-AI feedback mechanisms:

- Students receive tailored learning experiences that address their unique challenges.
- The system continuously evolves based on user interactions, improving its ability to teach effectively.

This bidirectional feedback model ensures that AI not only instructs but also learns from its users, refining its teaching methodologies to better support future programmers.

5 Ethics & Fairness

In today's world, ethical concerns related to diversity, fairness, and the potential failures of AI models are well-grounded. As our Study Support Tool heavily leverages AI models, we must address these challenges diligently. This chapter will explore specific problems that could arise and propose possible mitigations.

5.1 Main Ethical Concerns

Our user group's direct interaction with the AI model will be through AI companions guiding them on their coding journey. It is crucial to be forthright and transparent about the AI's capabilities and limitations. These companions are designed to provide factual information while adapting to the user's journey.

Key Ethical Concerns

Transparency and Clarity: Clearly communicate that the AI companion is a computer system, not a human. This will help mitigate the risk of users, particularly children aged 8-18, mistaking AI interactions for genuine human communication.

Emotional Attachment: There's a significant risk of emotional attachment to AI companions. While these companions adapt to users' stories and behaviors, they must never align with or support harmful or malicious intents.

Ethical Interaction: AI companions must adhere to ethical guidelines and provide constructive, non-harmful responses. They should never reinforce negative or inappropriate conduct.

5.2 Guidelines and Considerations

To address these ethical concerns, we draw on established guidelines and frameworks:

EU's Ethics Guidelines for Trustworthy AI: [9] Outlines seven key areas for robust, lawful, and ethical AI operations. Ensuring that our product adheres to these points is crucial both here and in Chapter 6.

UNICEF's Policy Guidance on AI for Children [10]: Emphasizes children's healthy development and well-being. It provides guidelines and practical implementation details to ensure AI systems are designed with children's best interests in mind.

EU's Ethical Guidelines on AI and Data Usage in Education [11]: These guidelines focus on the ethical use of AI and data in educational contexts.

5.3 Transparency and Directness

Given the wide age range of our target user group, it's important to mitigate the risk of children forming emotional bonds or developing addiction to AI companions.

Age-Appropriate Disclosure: Clearly communicate to users that their companion is an AI model, not a real human, in an age-appropriate manner.

Non-Human Visuals: AI companions should not represent humans. They can have humanoid features like hands and faces, but their likeness should clearly differ from that of a human. Using personas like robots or animals can create a clear disconnect from human nature.

Parental and Guardian Involvement: Parents or legal guardians should monitor their children's accounts periodically to ensure progression aligns with expectations. They should have the option to flag and report any mistaken or harmful interactions.

5.4 Fairness and Diversity

Given the personalized and adaptive nature of the user's coding journey, the underlying AI model should create diverse scenarios and personas.

Customizable Personas: Users should be able to ”tell their own story” throughout their journey. The AI companions should serve as guides, with customizable base personas like robots or animals.

5.4 Failure Considerations

To prevent misuse and ensure ethical operation, several measures should be in place:

Detecting Malicious Use: The system should detect and report attempts at ”jailbreaking” to parental users. This is the most direct method of addressing such behavior.

Soft Blocking and Censoring: Employ soft blocking or censoring to alter conversations and ask users to restate their questions, helping develop their intentions. A rule-based system can flag harmful or discriminatory words and phrases.

Expert-Designed Study Material: Technical study material should be designed and verified by experts. These materials should be accessible regardless of the companion's answers. The AI model should advocate searching these materials or directly referencing them, reducing the risk of ”hallucinations.”

User Warnings and Continuous Improvement: The chat interface should remind users that the AI model can make mistakes, encouraging them to research and verify topics further. Continuous feedback from users and automated detections must be included in development to improve performance and correct inaccuracies.

6 Privacy

By addressing these ethical concerns and implementing robust mitigation strategies, we can create a trustworthy, fair, and supportive learning environment for our users.

In this section, we will overview and detail privacy-related concerns and their solutions. This topic is of paramount importance, closely linked with our AI model's ethics. These areas overlap significantly, especially considering the age of our user group, which necessitates parental supervision, data storage, and processing.

6.1 General Data Protection Regulation (GDPR)

The General Data Protection Regulation (GDPR) [13] is the EU's primary data privacy law. It outlines several obligations that data processors must meet. Enforcing compliance with these regulations can be challenging, but resources such as CNIL's recommendations for GDPR compliance in AI system development [14] can be effectively integrated.

Core GDPR Guidelines Implemented in Good Faith

Data Minimization: GDPR Article 5 lists several principles of personal data processing. One principle is the collection of only adequate, relevant, and limited personal data necessary for our purposes. This guideline, often called the "Principle of Least Authority" in information security, means collecting only the minimally required information from users.

Pseudonymization: GDPR Articles 25 and 32 emphasize "Data protection by design" and users' right to secure data processing. An effective measure mentioned is pseudonymized data storage.

Data Retention Policies: GDPR Articles 5 and 17 outline rules of storage limitations and the "Right to Erasure" (also known as the "Right to Be Forgotten"). Compliance with these articles ensures users' right to have their private data kept only as long as necessary and allows them to request the deletion of all their personal data.

Combined, these principles ensure that even in the unfortunate case of a data breach by a malicious actor, the data at hand is minimal and non-identifiable. Of course, every other GDPR article not mentioned here must also be followed in a consistent and lawful approach.

6.2 User Group-Specific Data Concerns

Our target user group consists mainly of children under the age of 18, posing specific challenges related to their rights and those of their parents or legal guardians. As a vulnerable user group, we must ensure their rights and privacy are safeguarded.

Parental Supervision and Data Handling

Parental Profiles: Parental profiles will have all the data necessary to make informed decisions for their associated children's profiles. Whenever a significant decision related to data privacy must be made, the parental profiles must be alerted to act accordingly.

Minimizing Personal Information Collection: Children's personal information not required for the app's functioning, such as race or gender, should not be collected. Age must be collected to specify a baseline for the model and ensure data privacy compliance, potentially necessitating parental profiles. Any required personal information, such as age, should not be provided to external third-party data processors.

6.3 Parental Profiles

Parental profiles ensure that children's profiles can be monitored and any legal challenges, such as privacy policies and data collection settings, can be addressed by their parents or legal guardians.

Key Features of Parental Profiles

Access to Children's Data: Parental profiles will have access to specific data of the associated children's profiles by default, including the AI companion's persona characteristics and study material progression. Parents can adjust privacy and interaction settings according to their and their children's comfort.

Incident Reporting System: A comprehensive incident reporting system must be in place to inform or warn parental profiles in case of serious problems. This includes detecting malicious or dangerous situations or when direct intervention is required (e.g., privacy policy confirmations). Parents should also be able to flag and report behaviors of the AI model that they find harmful, ensuring continuous improvement and integration of user feedback into the platform's development.

By implementing these privacy measures, we can ensure a secure, respectful, and user-friendly environment for our young learners.

7 Conclusion

In conclusion, our AI-Powered Study Support Tool represents a significant advancement in programming education. By incorporating interactive, game-inspired elements and personalized feedback, we have created an engaging and effective learning experience for students. The dual-UI approach caters to beginners with block-based coding and progresses to text-based coding for intermediate learners, ensuring a seamless transition as they develop their skills. Our platform is designed to adapt to the ever-changing technological landscape, with continuous updates that add new programming languages, games, and challenges. Strategic partnerships with stakeholders, such as ChatGPT and other technology providers, further enhance the app's capabilities and keep it up-to-date.

Ethics and fairness are at the core of our AI model, ensuring transparency, diversity, and adherence to privacy regulations. The implementation of parental profiles and the focus on data minimization, pseudonymization, and data retention policies align with GDPR guidelines, safeguarding the privacy and rights of our young learners. By leveraging the power of AI and gamification, our Study Support Tool not only makes learning to code accessible and enjoyable but also fosters a positive attitude towards continuous learning and problem-solving. As we move forward, we remain committed to refining and expanding our platform, ensuring it meets the highest standards of ethics, fairness, and educational excellence.

References

[1] Lucas Eduardo, ”How to Use Binary Categorical Crossentropy with Keras,” Github, 2023. [Online]. Available: https://github.com/christianversloot/machine-learning-articles/blob/main/how-to-use-binary-categorical-crossentropy-with-keras.md. [Accessed: Feb. 18, 2025].

[2] ML_equations, ”Classical ML Equations in LaTeX,” Github, 2022. [Online]. Available: https://github.com/blmoistawinde/ml_equations_latex. [Accessed: Feb. 20, 2025].

[3] Christos Thrampoulidis,Samet Oymak, Mahdi Soltanolkotabi ”Theoretical Insights Into Multiclass Classification: A High-dimensional Asymptotic View”, published paper, https://proceedings.nips.cc/paper_files/paper/2020/file/6547884cea64550284728eb26b0947ef-Paper.pdf

[4] Turing, ”Softmax: Multiclass Neural Networks,” Turing, https://www.turing.com/kb/softmax-multiclass-neural-networks

[5] Baeldung, ”Multiclass Classification Using Support Vector Machines,” Baeldung, 2025.[Online]. Available: https://www.baeldung.com/cs/svm-multiclass-classification. [Accessed: Feb. 20.2025]

[6] Julien Vitay, ”Linear Model: Multi-class classification”, julien-vitay, 2021 [Online]. Available: https://julien-vitay.net/lecturenotes-neurocomputing/2-linear/5-Multiclassification.html. [Accessed: Feb. 8. 2025]

[7] Naomi Ceder, ”Notes on Teaching Python – Mental Models — Learn Python,” Naomicedar Blog, 2023 https://www.naomiceder.tech/imports/notes-on-teaching-python-mental-models-learn-python/

[8] Scott H. Young, ”Ten Mental Models for Learning,” Scotthyoung, Sep. 2, 2022 https://www.scotthyoung.com/blog/2022/09/19/learning-mental-models/

[9] European Commission. ”Ethics Guidelines for Trustworthy Artificial Intelligence”. https://digital-strategy.ec.europa.eu/en/library/ethics-guidelines-trustworthy-ai.

[10] UNICEF. *Policy Guidance on Artificial Intelligence for Children.* Available at: https://www.unicef.org/globalinsight/reports/policy-guidance-artificial-intelligence-children.

[11] European Commission. *Ethical Guidelines on Artificial Intelligence (AI) and Data Usage in Teaching and Learning for Educators.* Available at: https://education.ec.europa.eu/focus-topics/digital/ethical-guidelines-on-ai-and-data-usage-in-education.

[12] Kim, ”Learning design to support student-AI collaboration: perspectives of leading teachers for AI in education”, https://link.springer.com/article/10.1007/s10639-021-10831-6#citeas

[13] European Union. *General Data Protection Regulation (GDPR).* Available at: https://gdpr-info.eu/.

[14] CNIL. *Recommendations for GDPR Compliance in AI System Development.* Available at: https://www.cnil.fr/en/recommendations-ai.