AI Report

We are highly confident this text is

AI Generated

AI Probability

99%

This number is the probability that the document is AI generated, not a percentage of AI text in the document.

Plagiarism

?

The plagiarism scan was not run for this document. Go to gptzero.me to check for plagiarism.

Chapter 2: Background - 7/31/2025

Jack Whelan

Chapter 2: Background

This chapter covers why I decided to build "Coding Codec" and what motivated me to create a gamified C# learning platform.
This section also talks about how I think the platform will help its users and explains why I made certain design choices based on both research and my own experiences.

2.1 Educational Context and Digital Learning Trends

In the modern digital era, the demand for accessible and engaging programming education has surged.
As software development continues to grow in scope and influence, so too does the necessity for new programmers to enter the workforce equipped with both conceptual understanding and practical skills.
Sites like Udemy, Coursera, and FreeCodeCamp have made online learning popular, but most of them rely heavily on videos with not much hands-on practice.
Others expect you to already know some programming, which can put off complete beginners.

From what I've seen, most learning platforms are pretty passive Learners consume videos or articles but have limited opportunities for hands-on practice or reinforcement.
Feedback, when available, tends to be generic or delayed.
This disconnection between theory and practice can cause frustration, reduce motivation, and lead to premature disengagement.
Based on both my observations and discussions with peers, I realized there was a clear opportunity to bridge this gap by building an active learning environment eu one where users engage with the material directly, receive instant feedback, and feel rewarded for their progress.

The move toward gamified education represents a shift in pedagogical approaches.
Studies have shown that gamification eu the application of game-design elements in non-game contexts eu improves motivation, encourages consistent practice, and fosters deeper engagement.
This works really well in apps like Duolingo for languages, Fitbit for fitness, and Habitica for productivity.
It's also worked well in education, especially in STEM subjects where students often have trouble with abstract ideas.

I envisioned "Coding Codec" as a response to these trends: an interactive, gamified, and user-friendly platform that could make C# programming approachable and engaging for newcomers.

2.2 The Motivation Behind Learning C#

C# was selected as the target language for several reasons.
First, it is a widely used language in professional environments eu particularly for desktop applications, backend systems, and game development (via Unity).
Secondly, the.NET ecosystem has matured significantly and offers robust tools, libraries, and frameworks that work well in professional development.
Finally, C# has a relatively consistent syntax and supports both object-oriented and functional paradigms, making it suitable for teaching core programming principles.

In educational contexts, C# also represents a sweet spot between beginner-friendly structure and real-world applicability.
Unlike scripting languages such as JavaScript or Python, C# introduces concepts like strong typing, class-based inheritance, and memory management more explicitly.
These concepts are foundational for learners looking to move into more advanced software development roles later in their careers.

Despite its professional strengths, C# is often overlooked in entry-level learning platforms in favour of lighter scripting languages.
This creates an opportunity to design a platform that demystifies C# and gives learners a solid starting point in a language that is both academically rich and professionally useful.

2.3 My Personal Experience and Inspiration

Before beginning this project, I had previously interned at a company where I contributed to the development of internal web applications and participated in backend development projects.
During that internship, I became increasingly comfortable with database management, data validation, and full-stack workflows.
However, I also witnessed how new hires often struggled with learning C#, especially those coming from non-technical backgrounds.

This experience eu combined with my own journey through web development eu helped shape my understanding of how learning platforms succeed or fail.
I observed that structured systems with interactive components, clear milestones, and regular encouragement outperformed static or lecture-driven content by a large margin.

Users want to feel that they are progressing, that their actions are meaningful, and that they are part of a journey eu not just reading a tutorial page.

These insights motivated me to replicate that sense of progress and reward through a carefully designed gamification system.
My goal was not just to build an educational tool, but to create an environment where learning feels like achieving eu where every completed challenge is an accomplishment and every badge unlock is a reason to continue.

2.4 Anticipated Impact and User Benefits

The goal of "Coding Codec" is to help new programmers build confidence and skill in a low-pressure, encouraging environment.
By building a platform that rewards practice and celebrates progress, I want to make programming less intimidating.
Users start with simple challenges, earn points and badges, and gradually build confidence eu kind of like leveling up in a game.

This gives learners several benefits:

* Lowered learning anxiety: The platform avoids penalizing incorrect answers harshly.
Instead, it encourages people to keep improving their code, just like in real development work.

* Increased engagement: Immediate feedback, colourful visuals, and score-based rewards contribute to a sense of satisfaction and progress.

* Autonomy and pace control: Users can choose which challenges to complete, in which order, and how fast they progress.

* Peer motivation: Leaderboards and public achievement displays foster healthy competition and encourage regular use.

Instructors and educators may also benefit by using the platform to supplement their own curricula.
Since the system tracks progress and completion, teachers could potentially monitor student engagement or assign challenges as homework tasks.

2.5 Technological Foundations

Choosing the right tools was essential to the success of this project.
I opted to build the platform using the ASP.NET Core MVC framework, a powerful and modern web development platform offered by Microsoft.
This choice was guided by several factors:

* Familiarity and relevance: I had prior experience using C# and.NET technologies, which allowed me to hit the ground running.

In addition, these technologies are in demand in enterprise environments, adding professional relevance to the project.

* Security and scalability: ASP.NET Core offers built-in support for authentication, routing, Managing user sessions, and security eu all essential for a web application dealing with user data.

* Separation of concerns: The Model-View-Controller (MVC) architecture provided a clear and scalable structure that separates the frontend (View), backend logic (Controller), and data access (Model).

For the database, I used SQL Server in conjunction with Entity Framework Core eu an Object-Relational Mapper (ORM) that allowed me to define database schemas through C# models.
This approach enabled a seamless link between user actions and stored data, ensuring that each badge earned or challenge completed was tracked reliably.

The frontend was styled with Bootstrap, which provided responsiveness and UI components that work across desktop and mobile devices.
Razor Pages were used for dynamic content rendering.

This stack provided me with everything I needed to build a full-stack, responsive, and secure platform.

2.6 Key Use Cases

To illustrate the kinds of experiences the platform was designed to support, here are a few sample user journeys:

Use Case 1: A Beginner Learner

A first-time coder logs into the platform and starts with the tutorial titled "Variables and Data Types."
After reading an explanation and seeing an example, they attempt a related challenge.
Upon submitting correct syntax, they are congratulated and awarded 10 points and a "First Challenge" badge.
This motivates them to move on to the next topic.

Use Case 2: A Competitive User

A student with prior experience in Java wants to learn C#.
They complete a series of challenges over a few days and climb into the top 5 on the leaderboard.
Seeing their name among the best performers, they feel motivated to continue improving their score and completing more complex challenges.

Use Case 3: A Teacher

An instructor assigns the platform to their students.
They check the leaderboard weekly to monitor who is engaging with the content.
Because the challenges are categorized by topic and difficulty, the teacher can align lessons with platform content for blended learning.

Each of these journeys reflects the project's goal: to provide structure, motivation, and feedback to learners at different skill levels and with different goals.

2.7 Conclusion

The background of "Coding Codec" is rooted in educational theory, technological strategy, and personal insight. From the very beginning, my focus has been on bridging the gap between passive learning and active engagement.
By creating a platform that makes programming more accessible, more interactive, and more rewarding, I aim to lower the barrier to entry and empower users to build confidence through code.

The next chapter will delve deeper into the academic and technical literature that underpins these ideas, reviewing the existing body of knowledge on gamified learning, user motivation, platform design, and coding education.

FAQs

What is GPTZero?

GPTZero is the leading AI detector for checking whether a document was written by a large language model such as ChatGPT. GPTZero detects AI on sentence, paragraph, and document level. Our model was trained on a large, diverse corpus of human-written and AI-generated text, with a focus on English prose. To date, GPTZero has served over 10 million users around the world, and works with over 100 organizations in education, hiring, publishing, legal, and more.

When should I use GPTZero?

Our users have seen the use of AI-generated text proliferate into education, certification, hiring and recruitment, social writing platforms, disinformation, and beyond. We've created GPTZero as a tool to highlight the possible use of AI in writing text. In particular, we focus on classifying AI use in prose. Overall, our classifier is intended to be used to flag situations in which a conversation can be started (for example, between educators and students) to drive further inquiry and spread awareness of the risks of using AI in written work.

Does GPTZero only detect ChatGPT outputs?

No, GPTZero works robustly across a range of AI language models, including but not limited to ChatGPT, GPT-4, GPT-3, GPT-2, LLaMA, and AI services based on those models.

What are the limitations of the classifier?

The nature of AI-generated content is changing constantly. As such, these results should not be used to punish students. We recommend educators to use our behind-the-scene Writing Reports as part of a holistic assessment of student work. There always exist edge cases with both instances where AI is classified as human, and human is classified as AI. Instead, we recommend educators take approaches that give students the opportunity to demonstrate their understanding in a controlled environment and craft assignments that cannot be solved with AI. Our classifier is not trained to identify AI-generated text after it has been heavily modified after generation (although we estimate this is a minority of the uses for AI-generation at the moment). Currently, our classifier can sometimes flag other machine-generated or highly procedural text as AI-generated, and as such, should be used on more descriptive portions of text.

I'm an educator who has found AI-generated text by my students. What do I do?

Firstly, at GPTZero, we don't believe that any AI detector is perfect. There always exist edge cases with both instances where AI is classified as human, and human is classified as AI. Nonetheless, we recommend that educators can do the following when they get a positive detection: Ask students to demonstrate their understanding in a controlled environment, whether that is through an in-person assessment, or through an editor that can track their edit history (for instance, using our Writing Reports through Google Docs). Check out our list of several recommendations on types of assignments that are difficult to solve with AI.

Ask the student if they can produce artifacts of their writing process, whether it is drafts, revision histories, or brainstorming notes. For example, if the editor they used to write the text has an edit history (such as Google Docs), and it was typed out with several edits over a reasonable period of time, it is likely the student work is authentic. You can use GPTZero's Writing Reports to replay the student's writing process, and view signals that indicate the authenticity of the work.

See if there is a history of AI-generated text in the student's work. We recommend looking for a long-term pattern of AI use, as opposed to a single instance, in order to determine whether the student is using AI.