INTERACTIVE WEB: LEVERAGING AI-DRIVEN CODE GENERATION TO SIMPLIFY WEB DEVELOPMENT ALGORITHMS FOR NOVICE PROGRAMMERS

Kai Zhang [1], Vyom Kumar [1], Max Zhang [2]

[1] Moreau Catholic High School, Hayward, California
[2] Irvington High School, Fremont, California

ABSTRACT

In the history of computer science, the development of algorithms has long been pivotal to building the many programs used every day by civilians, such as programs for texting, entertainment, research, and anything on the internet. However, this programming remains time-consuming, and expensive, and requires years of expertise to master. Even if completing a singular project is not particularly difficult, the knowledge required to create that project might take years to learn. Since the development of AI, the process of fabricating algorithms has been shortened greatly. Anybody with internet access can make software using AI, but a novice in programming would still struggle with creating new code from a Pseudocode created by AI. By utilizing the OpenAI API, we created InteractiveWeb, a program that takes input from a user, sends it to ChatGPT, and, after creating an initial model of that program, repeatedly develops a new and improved version based on previous code. With this program, the AI could accurately develop instructed projects, allowing it to create the fully working games of Minesweeper, Connect 4, and even Chess by repeatedly refining code based on user feedback and self-debugging. InteractiveWeb aims to lower barriers to entry for novice programmers, foster creativity, and reduce burnout often associated with manual code development. Key results demonstrate improved coding efficiency, accuracy of results, and user satisfaction, highlighting the system's potential as a transformative educational and development tool.

KEYWORDS

AI, OpenAI, ChatGPT, Code Generation, Software Engineering using AI, AI-created Software

1. INTRODUCTION

Creating any program takes time and dedication. Initially, a creator must familiarize themselves with the tools necessary for their projects; Ex: learning a programming language. This is followed by pouring hours into coding and debugging the project they are trying to build. Such prolonged engagement can negatively impact programmers, leading to disrupted sleep, depression, stress, sensory overload, reduced social interactions, and even suicidal thoughts [1]. Many programmers can relate to the cycle of sleeplessness and intense focus from fixing bugs or implementing features, as well as the frustration of spending extensive time on mundane tasks instead of pursuing creative ideas. Studies indicate that 83% of programmers experience burnout due to unclear goals, inefficient processes, and high workloads [2]. Our project seeks to address issues

David C. Wyld et al. (Eds): BDAB, SVC, AIBD, ACSTY, NATP – 2025
pp. 81-91, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2024.150107

stemming from this burnout, breaking the cycle of sleepless nights and shifting the focus from merely fixing bugs to fostering creativity and generating fresh insights, allowing coders to start in the middle with a personalized template, skipping the useless time-wasting steps or debugging. This ultimately creates a healthier environment for programmers.

A common aspect among code generation AIs is their usage of AI to assist in game development by generating ideas, assets, and storylines; however, most artificial intelligence models like ChatGPT focus primarily on the early stages of development, such as ideation and storytelling, without addressing the full coding and development process. These tools often lack features for creating fully interactive games, comprehensive code generation, or self-debugging capabilities. InteractiveWeb stands out by addressing the entire development cycle, focusing on reducing high workload and inefficiencies, offering real recursive code generation and automatic debugging, as well as providing detailed feedback to improve coding outcomes, making our project more effective for developers than existing AI implemented solutions for creating advanced web apps or games.

InteractiveWeb allows creators to dedicate more energy to the enjoyable aspects of their projects—ideation, experimentation, and design. By shifting away from the repetitive environment, the mental fatigue and stress of coding can be eliminated [3]. This kind of interface not only enhances productivity but also revitalizes the joy of creation, empowering developers to focus on what they love most—bringing their imaginative ideas to life. Additionally, this system offers enhanced functionality compared to directly copying from ChatGPT by flagging potential errors automatically, providing an integrated game window for interactive outputs, and creating more advanced games beyond the basic features. Overall, InteractiveWeb allows novice coders to more efficiently and code better.

2. CHALLENGES

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

2.1. Accuracy

Sample Skeptical Question:

"How can an AI like ChatGPT generate accurate, functioning code without hallucinations or deviations from the original prompt?"

Sample Response:

“The latest GPT-4 model, particularly when fine-tuned for code generation, has significantly improved in understanding and maintaining alignment with complex programming prompts [4]. This model can process context more effectively and produce coherent, logically consistent code. Additionally, we integrated mechanisms within InteractiveWeb that verify the code in stages, catching any discrepancies early in the development process. This layered approach allows for high accuracy in generating code that closely adheres to user instructions, minimizing errors and ‘hallucinations’ common in earlier versions."

2.2. Importance

Sample Skeptical Question:

“There are so many people who can program in this world. Why do we need this program?”

Sample Response:

"Programming inherently takes a lot of time to learn, with many students studying for many years to create basic programs. This program is a precursor for many other programs which will allow for people with zero programming experience to code something using just English. As Jensen states, coding is dead, and AI is the future."

2.3. Difference

Sample Skeptical Question:

"Why can’t we just copy paste from ChatGPT?"

Sample Response:

"Our program is similar to copy-pasting from ChatGPT, however it automatically flags errors, and provides a game window already built in. Coders can quickly visualize a result, and results are more accurate due too sending more information in our prompts like bugs, user feedback, and more. Overall, our program is completely faster, and doesn’t require the user to actually do anything. In addition, we have engineered a prompt perfect for generating large amounts of code with a lot of characters that is readable and can work."

3. SOLUTION

Upon the first time the program is run, the application starts on a blank window with 3 textboxes, one for HTML code, another for CSS code, and the last for Javascript code. Because the JSON file length is 0, the game creates a small Game Window for the project. This includes basic features that ChatGPT thinks most games will require. If the JSON Data exists when the project is started, but it doesn’t have any code categorized under a game name, the player inputs the game name such as “Chess”, “Tower Defense Game”, or something new like “3D web game”, and it will code a starting template for the inputted game name [5]. If the player wants any custom features to be implemented, It can put it in the Feedback Box, and click the “Submit Comments” button. These suggestions, as well as errors, are sent to errors.json, which is given to the GPT for fixing. To call ChatGPT, the user presses the “Update” button, upon which the existing HTML, CSS, Javascript code, and suggestions are all sent to ChatGPT, after ChatGPT finishes, the code goes to the JSON file, and the page automatically reloads to display the latest version. If this version has too many bugs or doesn’t work, there is a revert button that resets to the previous version. A user can also change projects simply by typing a new game or an already existing game in the box that contains the Game Name. This fetches the most recent version of that game from the JSON, and updates from the provided code. Using GPT-4o, this process costs about 1 cent per comment for every update.

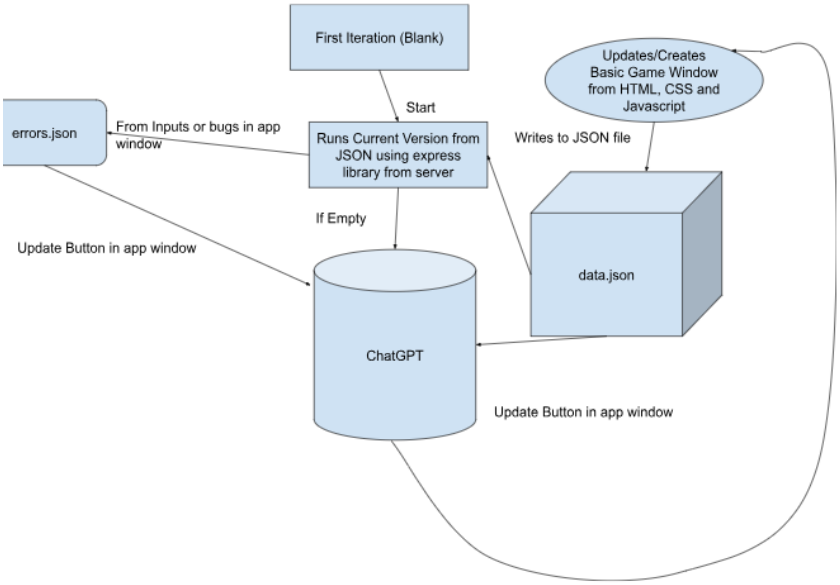

Figure 1. Overview of the solution

Using the "gpt-4o" model through the OpenAI API, users can achieve a complete set of HTML, CSS, and JavaScript code within approximately two minutes [6]. This model efficiently generates the core components required for a functional program, serving as the foundational structure that supports the entire application. The model leverages prior versions of code as well as user input to progressively refine and adapt the generated output, ensuring that each iteration aligns more closely with user expectations. By incorporating feedback or specific suggestions from users, the model not only generates initial code but also iteratively improves it, making the entire development process faster and more responsive.

```
  {
    "title": "",
    "html": "<div id=\"gameWindow\">\n  <div
id=\"scoreBoard\">Score: <span id=\"score\">0</span></div>\n
<button id=\"startButton\">Start Game</button>\n  <div
id=\"gameArea\"></div>\n</div>",
    "css": "#gameWindow {\n  width: 800px;\n  height: 600px;\n
 border: 2px solid black;\n  position: relative;\n  margin:
auto;\n}\n\n#scoreBoard {\n  position: absolute;\n  top:
10px;\n  left: 10px;\n  font-size: 20px;\n}\n\n#startButton
{\n  position: absolute;\n  top: 50px;\n  left:
10px;\n}\n\n#gameArea {\n  width: 100%;\n  height: 100%;\n
background-color: lightgrey;\n}",
    "javascript":
"document.getElementById('startButton').addEventListener('click
', startGame);\n\nfunction startGame() {\n
document.getElementById('score').innerText = 0;\n  // Add game
logic here\n}"
  },
```

Figure 2. Screenshot of the model

```
const GPT4Message = [
  {
    role: "system",
    content: "Generate the complete code for HTML, CSS, and Javascript without omitting any parts at all. Always include HTML, CSS, and JavaScript code as needed, in that order.
Fix any bugs that have a provided line number (in the Javascript code) and column number. If the previous code doesn't need to be changed, put it in anyway. Use ```html' for HTML,
```css' for CSS, and ```javascript' for JavaScript, ending each section with ````. For the HTML section, include only the code within the <body> tags.Do not include
explanations, titles, comments, or anything that could cause errors. Never use images, instead use colored divs. Analyze existing code to identify non-repetitive features and make
creative updates that enhance gameplay and mechanics. Ensure that your code is error-free, defines variables accurately, and does not shorten the original code. Include an update
menu that displays updates, pauses the game when opened, and provides a simple way to close it.",
  },
  {
    role: "user",
    content:
      ` Improve and debug the ${updatetype} using this code: ` +
      "\n```\n" +
      `HTML: ` +
      "\n```\n" +
      htmlupdate +
      "\n```\n" +
      `Javascript: ` +
      "\n```\n" +
      javascriptupdate +
      "\n```\n" +
      "CSS: " +
      "\n```\n" +
      cssupdate +
      "\n```" +
      "Bugs/Suggestions: " +
      "\n'''\n" +
      allsuggestions,
  },
];

let GPT4 = async (message) => {
  const response = await openai.chat.completions.create({
    model: "gpt-4o",
    temperature: 0.4,
    messages: message,
  });
```

Figure 3. Screenshot of code 1

The game starts with auto-generated code using a prompt that calls ChatGPT through the OpenAI API, allowing players to update game types and features seamlessly. Unlike requiring manual IDE use, our Express res.send() frontend displays code changes instantly, offering a user-friendly experience that adds real value by eliminating the need for direct backend interactions with ChatGPT.

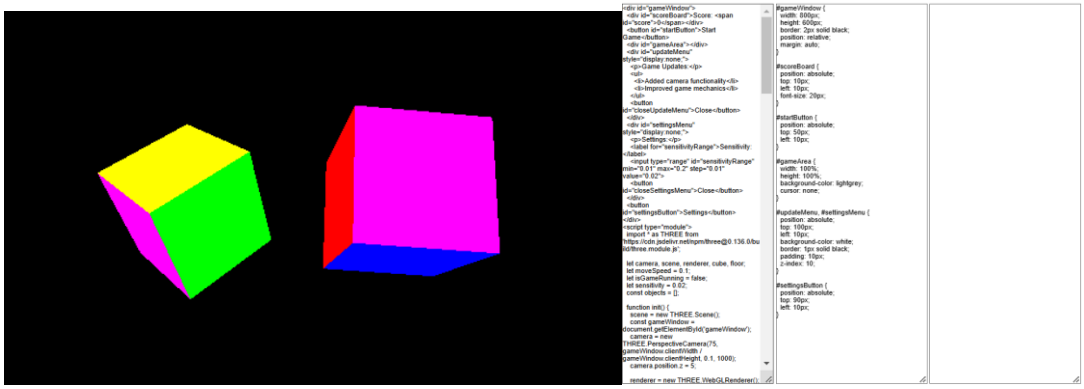

Figure 4. Screenshot of a game (Rotating 3D WebGL)
```

Figure 5. Screenshot of code 2

The Express-based front end fetches HTML, CSS, and JavaScript code from a JSON file. The res.send function places the HTML (generated without body tags) inside the body, the CSS within the <head>, and the JavaScript at the end of the HTML body wrapped in <script> tags [7]. This setup allows the code to function like a standard web page. Additionally, the application incorporates built-in features that enhance the game's visual appeal, alongside a button that enables full screen gameplay. At the bottom of the page, text areas display the content generated by GPT, offering users insights and information about the game, as well as a way to find a potential problem. This dynamic integration of HTML, CSS, and JavaScript creates a seamless user experience, allowing for easy interaction with the game and real-time visibility of the AI's contributions. Overall, it effectively combines interactivity, aesthetics, and functionality into a cohesive experience for the user.

On the front-end, errors are automatically sent to the file errors.json, and a player can send messages to the AI to give suggestions for the next update [8]. This process ensures that every update works and that the code that ChatGPT generates doesn't have any bugs it cannot fix. This code solves the main problem of ChatGPT not being good at debugging. In addition, the revert and update buttons provide a good way to revert potentially bad updates directly from the frontend, and to update from the frontend.

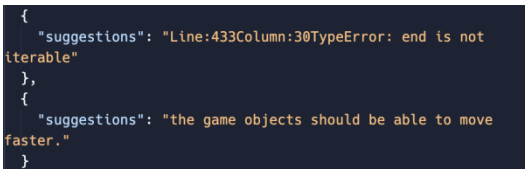

Figure 6. Screenshot of suggestions code

Figure 7. Screenshot of code 3

The front end, developed using the Express library's res.send() method, serves various dynamic pages directly to users. These pages feature buttons that enable a range of interactive functions, enhancing the overall user experience by making backend actions accessible with ease. When a user clicks any of these buttons, JQuery's AJAX functions send out asynchronous requests to the backend, allowing the app to perform actions like reverting changes, updating records, or submitting user-generated comments—all without reloading the page. This interaction provides an effortless and highly responsive experience. Additionally, this setup empowers ordinary users by facilitating direct access to JSON files and ChatGPT interactions from the front end, making it possible to create a fully functional program without intensive backend navigation. The integration of ChatGPT and JSON capabilities allows users to work with data dynamically, transforming the platform into a powerful tool for creating, updating, and managing data-driven applications.

4. EXPERIMENT

4.1. Experiment 1

Although ChatGPT typically performs extremely well while on a browser, its unique prompt might cause issues in runtime, or the quality of the update. In this experiment, we will test the runtime of prompts from people to add anything they want to the game. From this, we will find the time taken in seconds, as well as if the prompt was achieved.

Prompts from different people (3 total)

Each person will run a separate update on a version of a Tower Defense Game, and see if the features were added, as well as the cost of the ChatGPT.

Amount of Comments in Update	Time Taken(seconds)	Cost(cents)	Features Working
1	16.5	1	1
1	18.5	1	0
1	12.5	1	0
1	17	1	1
1	13	1	1
2	15	2	2
2	19	2	2
2	31	2	2
2	27	2	1
2	26	3	0
3	33	2	1
3	31	3	2
3	25	3	3
3	27	3	1
3	40	2	3
4	36	3	1
4	25.5	3	2
4	31.5	3	0
4	30	3	0
4	29	3	4

Figure 8. Figure of experiment 1

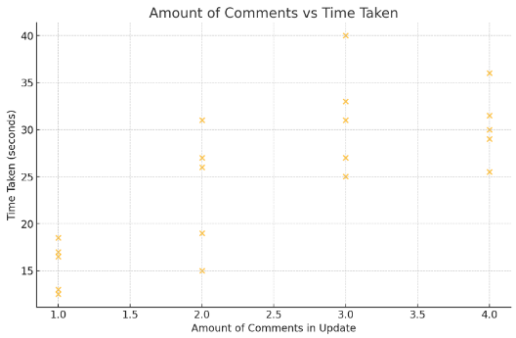

Figure 9. Amount of Comments vs Time Taken

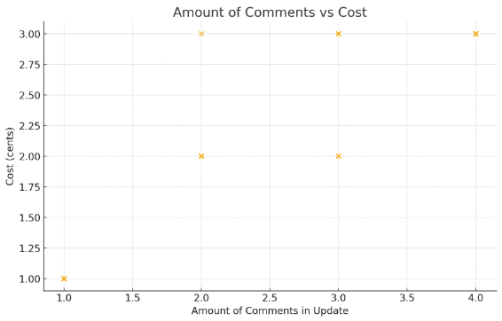

Figure 10. Amount of Comments vs Cost

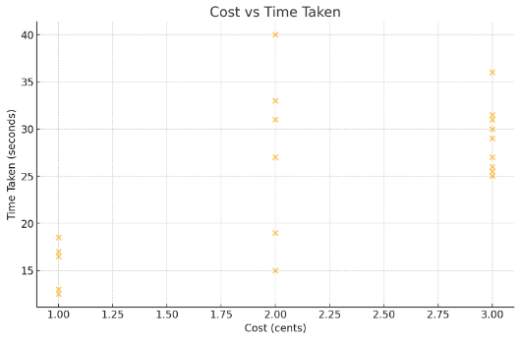

Figure 11. Cost vs Time Taken

The data offers intriguing insights into how a specific game affects introverted and extroverted individuals differently. As seen in Figure 8 and Figure 10, the Average Costs of the updates seem directly proportional to the resulting cost of the Update. From Figures 8 and 9, it can be seen that the average time taken tends to go up between updates. However, this may also be influenced by the amount of previous code that is in need of being updated. From Figure 11, we can also see that the Time Taken and Cost are directly correlated. Overall, all of these Figures suggest that these three variables follow a linear pattern that increases every update based on the amount of comments presented.

5. RELATED WORK

Ludo.ai is a platform that allows game developers to design and create games by helping them create ideas, track market trends, and provide them with researching tools [9]. It helps them to boost their ideation process, organize assets, and make it easier to create games that align with current trends. Moreover, Ludo.ai gives users tools to create new assets, new mechanics, and more to produce engaging and market ready games. It also allows for teams to work together and transfer ideas.

Ludo.ai, however, focuses more on the ideation process, which only accounts for about 29% (in combination of other factors) of burnout for developers under “Unclear Goals and Targets”. The rest, high workload and inefficient process, accounting for a total of 47% and 31% respectively, of the causes of burnout, is what our project hopes to subside. By fixing the bigger problem created by coding, our project is more effective than Ludo.ai by a substantial margin, because our project handles the much larger problem of high workload and inefficient processes, and lets the programmer find the ideas by themselves.

Giz.AI takes inspiration from premade prompts and uses them to generate games [10]. It uses AI to generate a storyline, sounds, and pictures to represent the time. The concept seems cool, but it doesn’t generate an actual game with real player interaction, only a text based rpg, which doesn’t generate code.

Unlike Giz.AI, our project generates a real playable game that is fun to create. The process of converting ideas into code creates a wider range of experiences, unlike Giz.AI’s purely storytelling perspective. The availability of an infinite possible amount of prompts from a user destroys a limited scale prompt that only results in AI generated images, sounds, and text.

Workik’s AI Code Generator seems to just be the same as ChatGPT [11]. It is helpful for people to work along with other IDEs, but it doesn’t contain its own. It has support for most languages in coding, but it is the same as asking ChatGPT to generate code as well.

Our program is much more refined as it provides a way for the program to debug itself. Our prompt engineering is much better and it typically yields a better result. Overall, Workik's AI Code Generator is the exact same as asking ChatGPT for some code. It has no way to automatically debug itself, and it ends up the same or slightly worse than asking ChatGPT for code, while our project fixes the main problem of ChatGPT's code causing bugs by providing the error stacktraces to ChatGPT for it to update, as well as providing a window to show what parts of the code work.

6. CONCLUSIONS

Our program, which utilizes ChatGPT to generate code in the browser, and send responses in JSON, is highly innovative but has a few limitations and areas for improvement. The primary limitation lies in token limits imposed by the GPT model, which restricts the complexity and depth of code that can be generated in one response [12]. For larger projects, these limits require breaking down requests, which can be cumbersome and may lead to inconsistencies in code. An improvement that can be made here is having ChatGPT only generate new parts of the code when necessary, instead of generating the full code for each version, which takes money and the consistency goes down as the code gets bigger. Additionally, GPT models are constantly evolving, meaning current capabilities may lag behind the latest advancements in accuracy, adaptability, or language comprehension. Regular updates to align with newer model versions would keep the program competitive. Lastly, our prompt engineering and design can be improved to add support for other languages, and it was a limitation in our project. Vague prompts led to sub-par outcomes from ChatGPT, while a better prompt led to better results [13]. Overall, our program, although innovative, has a few limitations and improvements that can be made, mainly more efficient and accurate code generation and keeping up with GPT models.

REFERENCES

[1] K. L. DeWeese, "Screen time, how much is too much? The social and emotional costs of technology on the adolescent brain," 2014.

[2] A. Razaghpanah, et al., "Haystack: In situ mobile traffic analysis in user space," *arXiv preprint arXiv:1510.01419*, 2015, pp. 1-13.

[3] E. Neophytou, L. A. Manwell, and R. Eikelboom, "Effects of excessive screen time on neurodevelopment, learning, memory, mental health, and neurodegeneration: A scoping review," *International Journal of Mental Health and Addiction*, vol. 19, no. 3, pp. 724-744, 2021.

[4] J. A. Baktash and M. Dawodi, "GPT-4: A review on advancements and opportunities in natural language processing," *arXiv preprint arXiv:2305.03195*, 2023.

[5] P. Bourhis, J. L. Reutter, and D. Vrgoč, "JSON: Data model and query languages," *Information Systems*, vol. 89, Art. no. 101478, 2020.

[6] A. Y. Sun, et al., "Does fine-tuning GPT-3 with the OpenAI API leak personally-identifiable information?" *arXiv preprint arXiv:2307.16382*, 2023.

[7] A. Ranjan, A. Sinha, and R. Battewad, *JavaScript for modern web development: building a web application using HTML, CSS, and JavaScript*, BPB Publications, 2020.

[8] Z. H. İpek, et al., "Educational Applications of the ChatGPT AI System: A Systematic Review Research," *Educational Process: Int. J.*, vol. 12, no. 3, pp. 26-55, 2023.

[9] P. R. Singh, M. A. Elaziz, and S. Xiong, "Ludo game-based metaheuristics for global and engineering optimization," *Applied Soft Computing*, vol. 84, 2019, Art. no. 105723.

[10] F. Lu, et al., "Fighting game artificial intelligence competition platform," in *Proc. IEEE 2nd Global Conf. Consumer Electronics (GCCE)*, 2013.

[11] M. Kazemitabaar, et al., "Studying the effect of AI code generators on supporting novice learners in introductory programming," in *Proc. CHI Conf. Human Factors in Computing Systems*, 2023.

[12] J. He, et al., "Talk too much: Poisoning large language models under token limit," *arXiv preprint arXiv:2404.14795*, 2024.

[13] T. Wang, N. Zhou, and Z. Chen, "Enhancing computer programming education with LLMS: A study on effective prompt engineering for python code generation," *arXiv preprint arXiv:2407.05437*, 2024.

AUTHORS

Kai Zhang
Biography: Hi, my name is Kai Zhang. I am currently a sophomore at Moreau Catholic High School in Hayward. I enjoy martial arts, coding, and math. I made a majority of the program as well as most of the paper.

Max Zhang
Biography: My name is Max Zhang. I am from Irvington High School in Fremont and I am a sophomore currently. My hobbies are playing badminton, coding, and drawing. I helped create this project program mainly and contributed to the writing of the paper.

Vyom Kumar
Biography: Hi, I'm Vyom and I am a sophomore at Moreau Catholic High School in Hayward. I enjoy running, coding, chess, and math. I helped with testing the project as well as recording the data. I also contributed to the writing and formatting of this paper.