Integrating Large Language Models into Web Design Study: AI-Assisted Code Optimization in Higher Education

Olena KOPISHYNSKA
Educational Research Institute of Economics, Management, Law and Information Technologies, Poltava State Agrarian University
Poltava, 36003, Ukraine

Yurii UTKIN
Educational Research Institute of Economics, Management, Law and Information Technologies, Poltava State Agrarian University
Poltava, 36003, Ukraine

Ihor SLIUSAR
Educational Research Institute of Economics, Management, Law and Information Technologies, Poltava State Agrarian University
Poltava, 36003, Ukraine

Volodymyr PYSARENKO
Educational Research Institute of Economics, Management, Law and Information Technologies, Poltava State Agrarian University
Poltava, 36003, Ukraine

Oleksandr GALYCH
Educational Research Institute of Economics, Management, Law and Information Technologies, Poltava State Agrarian University
Poltava, 36003, Ukraine

Leonid FLEHANTOV
Educational Research Institute of Economics, Management, Law and Information Technologies, Poltava State Agrarian University
Poltava, 36003, Ukraine

Iryna ZAHREBELNA
Educational Research Institute of Economics, Management, Law and Information Technologies, Poltava State Agrarian University
Poltava, 36003, Ukraine

Svitlana PYSARENKO
Educational Research Institute of Economics, Management, Law and Information Technologies, Poltava State Agrarian University
Poltava, 36003, Ukraine

ABSTRACT

This study explores the potential and effectiveness of using large language models (LLM) of artificial intelligence (AI) based on GPT for enhancing, optimizing, and extending website code in during the development and technical auditing stages. The research is grounded in experiments conducted with commercial-purpose websites developed by university students. General characteristics and a list of essential elements and functions of the websites are provided; these sites were built using HTML, CSS, and JavaScript technologies, based on original projects and designs. Both ChatGPT and GitHub Copilot demonstrated high performance in completing the assigned tasks, with ChatGPT showing broader capabilities, especially in the role of a tutor. AI proved useful in analyzing website structure and design, offering suggestions for technical and SEO optimization, and proposing innovative solutions that significantly improved functionality. The findings support the integration of AI into the educational process for training specialists in programming and web design in higher education, positioning it as an additional tool for code development, optimization, and refactoring. Particular emphasis is placed on the importance of a critical approach to AI-generated recommendations and the ability to conduct productive dialogue.

Keywords: LLM, Artificial Intelligence, AI, ChatGPT, Copilot, JavaScript, code, website, SEO optimization

1. INTRODUCTION

One of the most widely debated topics in higher education today is determining the role of artificial intelligence (AI) in studying various academic disciplines and how it can be integrated with traditional teaching methods. This also concerns its impact on the content of future jobs and required competencies. The role of educators and approaches to mastering the essential knowledge in each professional field have undergone significant transformations.

Despite numerous publications reasonably highlighting the threats posed by AI – such as diminished critical thinking and reduced development of students' intellectual abilities when they rely entirely on AI for solving complex problems – a significant number of researchers emphasize the importance of incorporating AI and its vast knowledge base into the educational process, especially in higher education. This is a reality that must be acknowledged, with an urgent need to identify effective ways to integrate AI into training for future professions. The challenges and opportunities associated with AI were discussed in Mustafa Suleyman's recent book [1].

In a world where artificial intelligence is rapidly integrating into all spheres of life, the issue of the future of work is becoming increasingly relevant. A study conducted by the Upwork Research Institute [2] examined the causal impact of generative AI on freelancers' employment and income opportunities. The

ISBN: 978-1-950492-85-5
ISSN: 2771-0947

https://doi.org/10.54808/WMSCI2025.01.497

data analysis showed how generative AI is transforming work. A review of past technological suggests that AI will have a dual, contrasting impact on employment.

Firstly, there is the widely discussed substitution effect, where certain jobs are disrupted as specific tasks become automated. However, equally important is the less-discussed restoration effect – where newly created work opportunities eventually increase income, as new technologies generate new tasks in which human labor has a comparative advantage. Typically, the substitution effect is more noticeable in the short term, but as new jobs and roles emerge in the medium and long term, the restoration effect is expected to outweigh it. Over time, the impact of new technologies often balances itself out, giving way to many new skills and opportunities that were previously impossible or non-existent.

Generative AI has increased both the total number of job opportunities and the income of freelancers per newly created contract. This indicates a growing demand from clients for work performed by independent professionals. These new projects generally promote skilled, knowledge-based work, offering greater task complexity and diversity for the workforce.

Historically, technology has created more employment opportunities than it has eliminated, as society evolves, innovates, and discovers new ways of working.

Microsoft co-founder Bill Gates has shared his thoughts on the professions he believes will remain beyond the reach of AI [3]. He argues that while AI is capable of automating many tasks in manufacturing, logistics, and agriculture, there are still professions requiring human intuition, creativity, and critical thinking – capabilities that machines have yet to replicate. Among these professions are programmers and web designers. Despite AI's ability to generate code, programmers will continue to be essential for understanding, debugging, and optimizing AI algorithms. They will play a key role in ensuring the proper functioning and ongoing development of AI systems.

The primary objective of this study was to determine to what extent, and for which specific tasks, the use of language models based on artificial intelligence can effectively supplement the learning process – enabling time savings in website development, editing, or code optimization. It is essential to understand the logic behind AI's operation to assess whether it performs tasks correctly or sometimes "misses the point." The experiment used several similar website templates created by students using fundamental web technologies as part of their portfolios, suitable for use by various business companies.

2. TRADITIONAL LEARNING METHODS AND THE SELECTION OF AI MODELS FOR WORKING WITH CUSTOM CODE IN WEB DESIGN AND WEB TECHNOLOGIES

Traditional web technology education involves the development of fundamental skills in HTML and CSS for building web application structure and UI/UX design, as well as the use of the JavaScript programming language to add dynamic and interactive functionality to websites. These tools hold leading positions in web development, as confirmed by regular reports from analytics companies (e.g., [4]).

A well-designed layout, content, and functionality are crucial to a website's commercial success and its ability to meet defined objectives. Even at the beginner level, students must master these basic technologies, learn how to debug their code, and continuously improve their skills. Professional developers are also expected to work with JavaScript libraries and frameworks, which vary in purpose and structure [5]. Backend development represents a separate domain. A web developer's specialization may focus on frontend, backend, or graphic design. An essential aspect of the development process is performing technical optimization of the website before deployment.

Given the vast range of available web technologies and their specific toolsets, university-level web development education must include substantial time investment (no less than 10 ECTS credits), along with independent exercises and student-led learning.

The initial learning stage involves understanding the basic HTML structure of a webpage and the purpose of various tags, replicating ready-made examples, and creating layout drafts based on given properties. This phase emphasizes learning, comprehension, and retention. The next step is the application of Cascading Style Sheets (CSS) for more precise, compact, and systematic descriptions of each webpage element's properties. At this point, students face challenges such as writing clean and reusable code, unifying styles across similar elements, selecting appropriate images, and setting goals for future design improvement.

Introducing JavaScript elevates the learning process to a new level by enabling event handling and improving UI/UX design. We explore the capabilities of the language's functions and methods to create dynamic effects and interactive interactions. Instructor-led classes should be supplemented with online resources, such as *W3Schools Online Web Tutorials* [6]. Ultimately, students acquire the skills to independently develop websites on specific topics with defined properties. This represents the analytical and decision-making phase. At this stage, significant time is also spent on verifying code correctness, browser compatibility, and optimization. To reduce routine work – particularly in code editing and improvement (refactoring) – the use of artificial intelligence may be considered.

To choose suitable AI models, we analyzed the capabilities of various platforms. Numerous online sources provide reviews of the top 10 tools offering AI-integrated website builders for fast development of commercial websites of varying complexity [7]. Among the leaders in this field are platforms such as Squarespace and Shopify [8–11]. These tools are suitable for users who are not professional developers but wish to create a product to support their businesses using ready-made templates and AI-generated solutions within a selected platform.

After analyzing the strengths and weaknesses of each tool, users can choose between free versions with limited functionality and accompanying ads, or paid plans with custom domains, no advertising, and more personalized solutions and designs. When teaching web technologies, we introduce students to these tools; however, we ultimately prioritize professional web application development involving custom code and immersion in the world of professional web design.

In [12], the authors present results on the deep integration of AI models for code optimization and refactoring in software engineering. They compare AI-driven tools and frameworks such as BERT (for code comprehension using NLP), LLVM (for redundancy detection), and Transformer models (for performance enhancement and real-time code suggestions). The article examines the features of these models and concludes on the effectiveness of supervised, semi-supervised, and unsupervised learning approaches. These tools have proven effective in solving complex computational programming tasks.

More relevant to our research are GPT-based models applied to code improvement. Initially, GPT models were used for natural language processing, but later found use in code-related tasks, including translating code between languages, generating

comments, and refactoring. These models process code in the same way they handle human language, enabling them to understand how different programming languages work.

Studies [13–14] have examined the democratizing impact of large language models (LLMs) on software development. Researchers noted that LLMs and GPT-based systems significantly enhance programmer productivity by automating routine tasks and offering real-time code suggestions. This contributes to a deeper understanding of software engineering and offers insight into the future trajectory of the field.

Research [15] analyzes the use of AI tools (GitHub Copilot, Azure AI) in .NET web application development. It describes code generation scenarios, bug detection, refactoring, and performance aspects, highlighting benefits such as productivity gains and reduced technical debt, as well as potential risks related to reliability and security.

In [16], the authors study the efficiency and maintainability of code generated by AI tools (Copilot, GPT-4, CodeWhisperer, ChatGPT) when solving *LeetCode* tasks. Copilot achieved the highest success rate (50% of solutions), while ChatGPT was the only tool to solve a complex task. Some incorrect outputs required only minor corrections, which significantly reduced development time.

Based on this analysis, we selected accessible GPT-based LLMs, specifically ChatGPT and GitHub Copilot, to conduct our own research using code samples developed during university-level web development coursework.

3. CHARACTERISTICS OF THE WEBSITES SELECTED FOR AI-BASED ANALYSIS AND OPTIMIZATION

To explore the capabilities and effectiveness of artificial intelligence in code optimization and editing of websites, several similar webpages developed by students at the end of an introductory web technologies course were selected for analysis. When setting the tasks, the functional and design quality requirements of commercial products – as typically defined by real-world IT companies specializing in web development and cooperating with universities – were taken into account [17].

The technical specifications, design planning, and formulation of plausible content and functionality requirements for these websites were guided by a case-based learning approach [18], which helped students understand the real-world contexts and business goals that websites are intended to serve.

In their final form, all websites were created as landing pages — commercial promo site templates for different business domains such as a coffee shop, a language school, energy efficiency services, and others. Each site consisted of 4 to 6 themed sections, including a main banner, navigation menu, slider, company reviews/testimonials, a contact form, and other standard components. Development was carried out in the *VS Code* environment. A part of the workspace is shown in Figure 1. The visual appearance of the top section of one of the websites is presented in Figure 2.

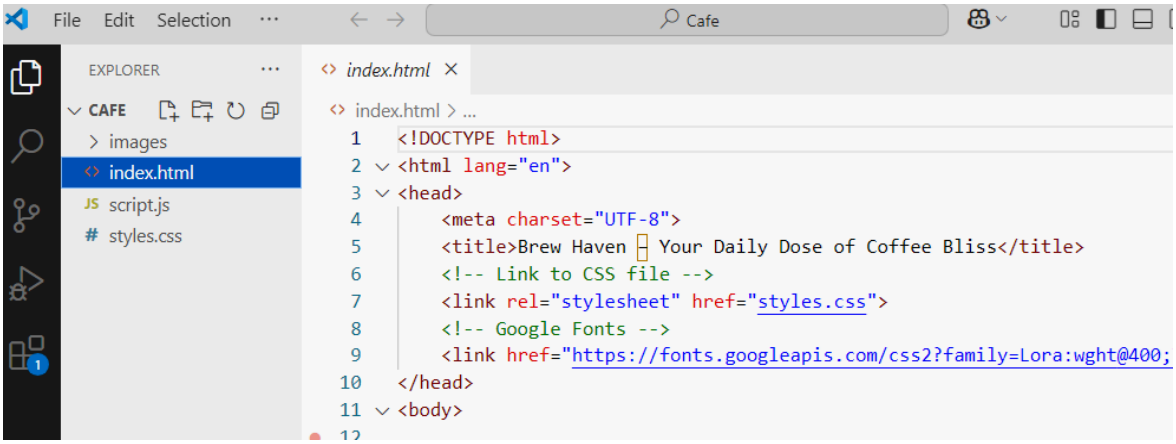

Figure 1. HTML file structure in VS Code and project folder for the coffee shop website

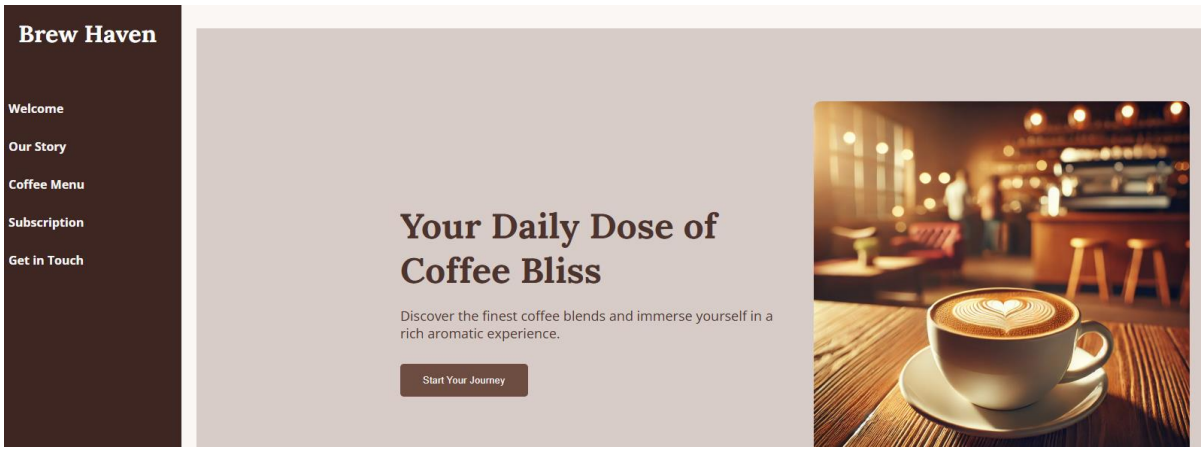

Figure 2. Visual layout of the top section and side navigation menu of the coffee shop website

The layouts were primarily designed using *Figma*. The HTML file of the page contained around 160 lines of code with markup tags. The block structure of each page was built with HTML and included semantic tags such as `<nav>`, `<footer>`, `<section>`, `<main>`, `<aside>`, in addition to the universal `<div>` and `<span>` tags. One original design solution was the side navigation menu block, which remains fixed on the page. The markup of this menu is shown in Figure 3.

```
<nav id="side-nav">
    <div class="logo">
        <h1>Brew Haven</h1>
    </div>
    <ul>
        <li><a href="#home">Welcome</a></li>
        <li><a href="#story">Our Story</a></li>
        <li><a href="#menu">Coffee Menu</a></li>
        <li><a href="#subscription">Subscription</a></li>
        <li><a href="#contact">Get in Touch</a></li>
    </ul>
```

Figure 3. Markup of the sidebar navigation menu block for the coffee shop website

CSS styles were used to define the properties of each block, using tag selectors, classes (.class), and custom #id selectors. Special pseudo-classes and transformations (e.g., `a: hover, scale`) were also applied. For responsive layout, the flex property and its functions were employed. For instance, Figure 4 shows the styling of the menu when hovered over with the mouse.

```
#side-nav ul li a {
    display: block;
    color: #faf7f5;
    padding: 15px 20px;
    text-decoration: none;
    font-weight: bold;
    transition: background-color 0.3s;
}

#side-nav ul li a:hover {
    background-color: #5d4037;
}
```

Figure 4. CSS properties for menu items on hover and active states

During development, students were tasked with code optimization, considering aspects such as CSS cascading and property inheritance. Figure 5 shows the "Coffee Menu" section of the coffee shop site, which features drink images.

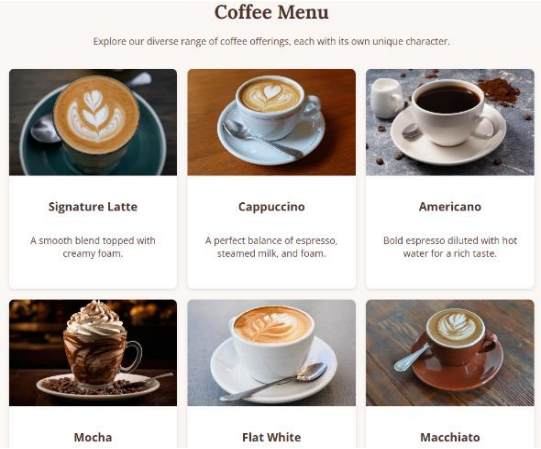

Figure 5. Mid-section design of the coffee shop promo website

All content blocks were uniformly styled using the `.card` class. Figure 6 displays the contents of the `styles.css` file with CSS properties for typical image and text elements belonging to this class.

```
.card {
    background-color: #fff;
    width: 300px;
    margin: 20px;
    border-radius: 10px;
    overflow: hidden;
    text-align: left;
    box-shadow: 0 2px 6px rgba(0,0,0,0.1);
}

.card img {
    width: 100%;
    height: 200px;
    object-fit: cover;
}

.card h3 {
    padding: 20px;
    font-size: 24px;
    color: #4e342e;
}

.card p {
    padding: 0 20px 20px;
    color: #5d4037;
```

Figure 6. CSS rules in styles.css for styling elements of the `.card` class

Thanks to CSS cascading, the styling code was concise and easy to manage. The main `.card` class defined the shape, size, background (`background-color: #fff;`), margins, rounded corners (`border-radius: 10px;`), shadow effects (`box-shadow: 0 2px 6px rgba(0,0,0,0.1);`), and other visual properties for each block.

JavaScript was used to implement interactive features such as pop-up message windows and carousels. Overall, the HTML and CSS components were developed to a high academic standard. Buttons, menus, forms, tables, graphs, maps, and other interactive elements were created using JavaScript. However, JavaScript at the introductory level poses difficulties for students due to the need to learn extensive syntax, variable usage, and function logic. As a result, students often rely on ready-made templates and examples from reference sites.

The purpose of engaging with artificial intelligence was to solve tasks such as conducting external quality assessments of the developed websites, identifying coding issues, and receiving suggestions for improvements and enhancements. All conversations were recorded for further evaluation regarding their accuracy, quality, sufficiency, and completeness. The interaction was conducted via ChatGPT.

4. RESULTS OF THE EXPERIMENT USING CHATGPT AS AN AI ASSISTANT FOR WEB CODE RECOGNITION AND OPTIMIZATION

The interaction was carried out using ChatGPT, a generative AI chatbot developed by OpenAI and released on November 30, 2022. It utilizes large language models (LLMs), such as GPT-4o,

to generate human-like responses in text, speech, and image formats.

The initial prompt (performed by different users for different websites) typically included a description of the software product and the task. For example: "I created a landing page on the topic of (general theme or type of business). The site is a SPA built using HTML, CSS, and JavaScript. It includes navigation, a slider, block text elements, and other components. Please analyze the site and suggest what can be improved."

AI Response 1: "Of course, I can help! Please upload the necessary code or all project files."

The users then uploaded the website files directly into the chat. The AI's response consisted of three message blocks:

Block 1: Identified strengths – This section included the achievements and strong points of the work. For the majority of the websites, ChatGPT highlighted the following positive features (the wording is retained from the original AI responses):

1) SPA structure as a universal and lightweight solution for browser-based use;
2) Stylish navigation — fixed at the top with smooth scrolling;
3) Consistent style and design (if it matched the thematic purpose);
4) Presence of animations (such as sliders or carousels, mentioned precisely);
5) Responsiveness and adaptability.

Block 2: Suggestions for improvement, grouped by criteria:

1) SEO and accessibility:

- Replace `<html lang="ua">` with "`uk`" — the official ISO code for Ukrainian;
- Add alt descriptions to all images, especially to banner backgrounds (using aria-label or other techniques) for screen reader compatibility;
- Include <meta name="description">.

2) Mobile responsiveness:

- In some blocks, adding `snap-x snap-mandatory overflow-x-auto` was suggested to enable swipe functionality for sliders on mobile devices;
- Footer layout had too many columns — potentially overwhelming on mobile screens. Suggestions included switching to two rows or using accordion elements.

3) Slider functionality (in several websites):

- Sliders were non-infinite — moving between slides 0, 1, 2 only, without cycling. ChatGPT provided code fragments to make sliders loop seamlessly;
- Smooth transitions between slides: "You already use `transition-transform`, but for a better UX, consider adding `easing`:"

```
#slider {
  transition: transform 0.5s ease-in-out;
}
```

Block 3: Optional enhancements ChatGPT also added ideas such as *"If you're interested, I can add…"* followed by a list of features.

Further communication required users to explicitly name the desired improvements. For instance, in one case, the user accepted the suggestion to create a responsive hamburger menu. The prompt was: *"Yes, please create a responsive hamburger menu."* ChatGPT then generated the necessary HTML fragments, CSS properties (Figure 7), and JavaScript code (Figure 8), which were integrated into the café website via copy-paste. It then offered to add an animation effect (transforming three lines into a cross), which was also easy to implement.

```
/* Hamburger Styles */
#hamburger {
    display: none;
    flex-direction: column;
    justify-content: center;
    gap: 5px;
    background: none;
    border: none;
    cursor: pointer;
    margin-bottom: 20px;
}

#hamburger .bar {
    width: 25px;
    height: 3px;
    background-color: #faf7f5;
    transition: all 0.3s;
}
```

Figure 7. Properties of the hamburger menu added to the #hamburger ID, developed using ChatGPT

```
/* Hamburger menu toggle */
const hamburger = document.getElementById('hamburger');
const navLinks = document.querySelector('#side-nav ul');

hamburger.addEventListener('click', () => {
    navLinks.classList.toggle('active');
});
```

Figure 8. JavaScript code for toggling the hamburger menu

Following the preliminary audit of the developed promotional website, the collapsed hamburger menu appeared as shown in Figure 9.

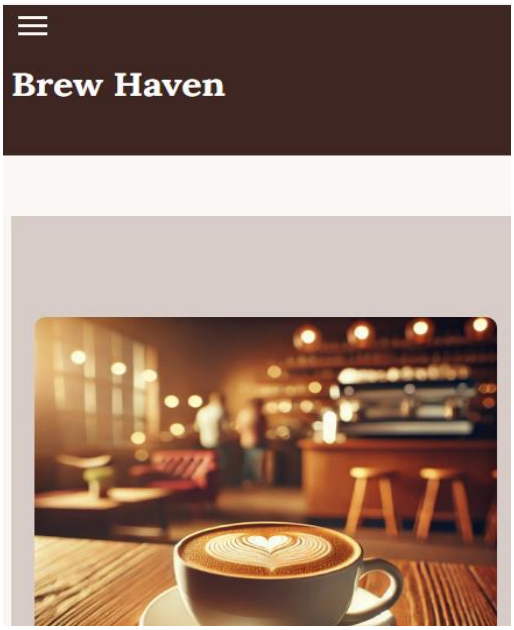

Figure 9. Appearance of the collapsed hamburger menu in the upper part of the website

One of the websites used in the experiment differed from the others in that it employed the Tailwind CSS framework instead of a separate stylesheet file. Its use inherently supports

optimization for landing pages. The framework helps avoid class duplication and provides a built-in system of breakpoints and default design tokens, including colors, spacing, and fonts.
Tailwind enables inline styling within HTML through utility classes, though this can lead to HTML clutter. For example, a button is described as follows:
`<button class="bg-blue-500 text-white px-4 py-2 rounded hover:bg-blue-600">.`
The full HTML code of the site (contained in a single file) was uploaded to ChatGPT with a request to analyze and split it into three parts, separating all style descriptions into a standalone file. The AI recognized these requirements and agreed to divide the code into structure, styles, and JavaScript.
However, despite multiple attempts, the task was not executed correctly. Even when generating a separate .css file, the source code still contained embedded styles. As the number of revised site versions grew, the AI began adding new elements, modifying content, and eventually making the code unreadable and overly abstract. Previously implemented tasks from the same user account (stored in history) began to influence subsequent answers. This suggests that the AI might adapt to user needs when encountering similar tasks.
When the user later requested to extract only the scripts into a separate file, ChatGPT acknowledged this as a step toward scalability and optimization, providing a brief justification and refactoring recommendations.

5. RESULTS OF THE USING COPILOT FOR WEB CODE ANALYSIS, OPTIMIZATION, AND REFACTORING

A comparable experiment was conducted using GitHub Copilot to assess its capabilities in web code optimization and support. Copilot is developed by GitHub (a Microsoft subsidiary) in collaboration with OpenAI. It is a generative AI assistant powered by the OpenAI Codex language model, an advanced derivative of GPT, designed specifically for code autocompletion. Copilot supports dozens of programming languages including Python, JavaScript, TypeScript, Go, C++, Java, HTML, CSS, and others. It is fully integrated into specialized IDEs such as Visual Studio Code, Visual Studio, Neovim, and various JetBrains IDEs.
All interactions with Copilot were conducted through Copilot Chat. In contrast to ChatGPT, Copilot Chat only accepts code snippets, not full file uploads, due to limitations in file handling support. As a result, code analysis for each component of a website was relatively general and descriptive, often lacking in detailed evaluation of the developer's achievements. Nevertheless, the improvement suggestions were frequently constructive and, in many cases, more detailed – complete with inline comments and ready-to-implement code fragments. Accessibility-focused enhancements proposed by Copilot:

1) Adding `aria-label` attributes to social media links and form fields (e.g., `aria-label="Your name"`);
2) Ensuring adequate text-background contrast for users with visual impairments (again referencing tools like WebAIM).

SEO improvement suggestions were consistent and typically included:
1) Adding meta tags, such as `<meta name="description">` for site descriptions and `<meta name="viewport">` for responsive scaling;
2) Ensuring adequate contrast between text and background (e.g., using tools like WebAIM **to** verify accessibility);

In general, Copilot provided more extensive suggestions across several website tests. These included justifications and generated code that helped users implement changes efficiently.

JavaScript improvements included:
1) Smooth scroll functionality that accounted for dynamic navigation bar height – Copilot recommended calculating the navigation bar height dynamically to accommodate layout changes on different devices.
2) Automatically clearing form messages after submission, accompanied by the corresponding script;
3) Adding form validation (e.g., checking for empty fields or validating email format) in cases where server-side integration was planned.

The original AI-generated code is shown in Figure 10. UX/UI enhancements addressed:
1) Close button visibility and behavior;
2) Addition of dark mode as a user toggle option;
3) Animated hamburger menu features — similar to those proposed by ChatGPT.

Broader professionalization recommendations provided by Copilot included:
1) Modularizing JavaScript code**:** extracting logic into separate files (e.g., `navigation.js`, `form.js`) to improve maintainability and scalability;
2) Using lightweight libraries (e.g., GSAP for animations or Validate.js for form validation) when more interactivity was needed;
3) Implementing error handling for edge cases, such as missing scroll targets or incomplete form submissions.

Overall, the process of improving, optimizing, and refactoring code with Copilot received high evaluations from all experiment participants. The tool proved particularly effective in verifying code correctness, correcting inaccuracies, and introducing professional enhancements that contributed to more polished web products. The assessment result - based on a 10-point scale - are presented in Table 1. The table compares the performance of both GPT-based AI models (ChatGPT and Copilot) across several evaluation criteria, using average scores derived from the subjective ratings of 24 participants.

```
const navHeight = document.getElementById('side-nav').offsetHeight;
window.scrollTo({
    top: targetSection.offsetTop - navHeight,
    behavior: 'smooth'
});
```

Figure 10. AI-generated by Copilot code for dynamic calculation of navigation bar height during scroll navigation,

Table 1. Comparative assessments of the convenience and completeness of individual results of GPT AI models when working with website codes on a 10-point scale

Criterion	ChatGPT	GitHub Copilot	Notes
Developer and purpose in programming	OpenAI, generation, explanation, debugging of code, as well as dialogue	GitHub + OpenAI (Microsoft-owned), mainly used for code autocompletion in IDEs	Reference information
Convenience and comfort of chatting	10 (main function of ChatGPT)	8 (additional but not primary function)	Includes chat menu, conversation saving, structured replies
Methods of uploading data for analysis	10 (listing, code parts, files of various types, archives)	8 (code listing, code snippet in programming language)	Based on supported code upload formats
Ability to understand third-party code and explain algorithms	10 (complete structured analysis)	8 (sufficient understanding, limited algorithm explanation)	Includes analysis of websites on various topics
Code generation based on code analysis	10 (fast and correct)	10 (fast, detailed, with comments)	Code generation occurred after developer confirmation
Optimization of third-party code	10 (all types of optimization)	10 (all types of optimization)	Evaluation based on understanding of optimization components and quality of suggestions
Effectiveness as a mentor	10 (acts like an experienced teacher)	9 (explains, breaks down its own code, but with limitations)	To learn effectively, one needs to master programming basics and web layout to interact with AI clearly and formulate tasks precisely

The results showed that GPT models exhibited a strong understanding of the structure and intent of the code, were capable of conducting syntactic analysis, effectively identified deficiencies in implementations, and provided meaningful suggestions for improvement.

6. CONCLUSIONS

Through multifaceted research on the use of large language models of artificial intelligence, exemplified by ChatGPT and Copilot, in the process of improving and optimizing website code, a number of positive results were obtained. The overall results of the study are presented in the following conclusions:

1) Both models involved in the experiment can be effective for programmers in writing, completing, explaining, and refactoring code. From a technical point of view, results similar to other studies were identified or confirmed: Copilot works more organically for code completion and suggestions within IDEs due to its integration into development environments. This is a powerful factor that helps avoid simple mistakes at the stages of code writing.
2) Both models are effective for analyzing existing code and generating suggestions for SEO and technical optimization. In the learning process, it is important to see the potential for improving one's work and advancing skill level. AI can greatly assist as an external auditor and assistant in this context.
3) All suggestions provided for upgrading or expanding element features and functionality through ready-to-use code fragments were easily integrated and enhanced the originality of the final web product.
4) A clear conclusion made by 95% of all experiment participants is that relying on AI to generate complete code without acquiring personal knowledge and practicing independent programming is a dead end. Instead, detailed analysis of suggestions and breakdown of code into parts promotes learning and reveals directions for further development. In this sense, ChatGPT is excellent as a mentor-assistant, analyst, and explainer. It is well-suited for learning, discussion, debugging, working with algorithms, and large code fragments. However, AI can also make mistakes or misunderstand you.
5) All experiment participants also unanimously concluded that using AI will soon become an essential part of programming education. It will be important to understand its capabilities and develop skills for effective dialogue (prompt engineering).
6) Everything stated in this study implies shaping an attitude toward AI as a useful professional tool that significantly increases programmer productivity, reduces time spent on routine tasks, and aids learning. However, it does not replace a full-fledged developer, sometimes makes contextual errors, and requires a critical approach to using its suggestions.

Future research directions may include experiments with code involving more complex technologies, other programming languages, and the generation of visual content for website graphic design.

7. ACKNOWLEDGMENTS

We would like to thank Vadym Slyusar, Doctor of Science, Professor, Chief of R&D Group at Central Scientific Research Institute of Armaments and Military Equipment of Armed Forces of Ukraine, for their non-blind review this article.

8. REFERENCES

[1] Mustafa Suleyman, **The Coming Wave: Technology, Power, and the Twenty-First Century's Greatest Dilemma**, New York: Crown, 2023, 332 p.

[2] Monahan, K.: Studying Generative AI's Impact on Work. https://www.upwork.com/blog/generative-ai-impact-on-work (2024) [accessed 2025/07/16].

[3] Younker, S.: Bill Gates just predicted the death of every job thanks to AI — except for these three. Tom's Guide. https://www.tomsguide.com/ai/bill-gates-just-predicted-the-death-of-every-job-thanks-to-ai-except-for-these-three (2025) [accessed 2025/07/16].

[4] The Top Programming Languages. IEEE Spectrum. 2024. URL: https://spectrum.ieee.org/top-programming-languages-2024, [accessed 2025/07/16].

[5] Swacha, Jakub, and Artur Kulpa. "Evolution of Popularity and Multiaspectual Comparison of Widely Used Web Development Frameworks", **Electronics** 12, no. 17: 3563, 2023. https://doi.org/10.3390/electronics12173563.

[6] "W3Schools.com". W3Schools Online Web Tutorials. https://www.w3schools.com/.

[7] Top 10 Best AI Website Builders of 2025. https://www.top10.com/website-builders/AI-powered-websitebuilder [accessed 2025/07/16].

[8] Squarespace. https://www.squarespace.com/, [accessed 2025/07/16].

[9] Shopify. https://www.shopify.com/ [accessed 2025/07/16]

[10] Ionos https://www.ionos.com/ [accessed 2025/07/16]

[11] Networksolutions. https://www.networksolutions.com/ [accessed 2025/07/16].

[12] Sandeep Konakanchi, "Artificial Intelligence in Code Optimization and Refactoring", **International Journal of Scientific Research in Computer Science, Engineering and Information Technology.** 11 (2): 1197-1211. https://doi.org/10.32628/CSEIT25112463 [accessed 2025/07/16].

[13] Prakash Raj Ojha "Democratizing Code: How GPT and Large Language Models Are Reshaping the Landscape of Software Creation", **International Journal of Scientific Research in Computer Science, Engineering and Information Technology**, 10(5), 2024, pp. 503–512. doi:10.32628/CSEIT241051031.

[14] Levine S. AI-Augmented Software Engineering: Automated Code Generation and Optimization Using Large Language Models. IJETCSIT [Internet]. 2020 Dec. 2 ;1(4):21-9. Available from: https://ijetcsit.org/index.php/ijetcsit/article/view/48.

[15] Shethiya, A. S.. "AI-Assisted Code Generation and Optimization in .NET Web Development", **Annals of Applied Sciences,** 6(1), 2025, Pp. 1-7**.** Retrieved from https://annalsofappliedsciences.com/index.php/aas/article/view/15 [accessed 2025/07/16].

[16] Idrisov, Baskhad, and Tim Schlippe, "Program Code Generation with Generative AIs", **Algorithms**, 17, no. 2: 62, 2024, pp.1-19. https://doi.org/10.3390/a17020062.

[17] Olena Kopishynska, Yurii Utkin, Ihor Sliusar, Vadym Slyusar, Nadiia Protas, Olha Barabolia, "Professional-oriented training of specialists under implementation of cloud computing information systems in cooperation between universities and IT companies" *Proceedings of The 14th International Multi-Conference on Society, Cybernetics and Informatics (IMSCI 2020), September 13-16, 2020,* pp.17-22. URL: https://www.iiis.org/CDs2020/CD2020Summer/papers/EA790UO.pdf [accessed 2025/07/16].

[18] Kopishynska O., Utkin Y., Galych O., Makhmudov H., Svitlychna A., Lyashenko V., "Features of the Case Method Application in the Study of Disciplines Related to Information Technologies and IT Project Management", *Proceedings of the 25th World Multi-Conference on Systemics, Cybernetics and Informatics: WMSCI 2021*, July 18-21, 2021. Pp. 7-12. URL: https://www.iiis.org/CDs2021/CD2021Summer/PapersS2.htm#/ [accessed 2025/07/16].