Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 3, Issue 11, 2025

INTELLISITE AI: A UNIFIED AI-AUGMENTED WEB DEVELOPMENT ENVIRONMENT WITH REAL-TIME CODE PREVIEW AND DESIGN AUTOMATION

Muskan[*1], **Areej Fatemah Meghji**[2], **Abdul Aziz Khoso**[3]

[*1,3]*Department of Software Engineering, Mehran University of Engineering and Technology, Jamshoro, Pakistan*

DOI: https://doi.org/10.5281/zenodo.17519953

Keywords
Intelligent Web Designer, Live Code Modifications, Web Development, AI Integration, Large Language Model (LLM), Graphical User Interface (GUI)

Article History
Received: 13 September 2025
Accepted: 23 October 2025
Published: 04 November 2025

Corresponding Author: * Muskan

Abstract
Web development and Artificial Intelligence (AI) are rapidly evolving fields poised to transform web interactions. Web development involves website and web application creation and maintenance, while AI focuses on intelligent systems capable of learning and adaptation. Although these areas have gained momentum, the front-end web development integration remains fragmented and disjointed. Developers use diverse applications and libraries, which is time-consuming and error-prone, including browser tabs for web-based documentations, Figma, code editors, Adobe XD, Visual Studio Code, and preview windows. This juggling results in inefficiencies and a higher error rate, coupled with context switching. There is a need for an application that reflects live viewing of changes to the code and designs to minimize the browser tabs required. To resolve these issues, this research suggests the development of an Intelligent Web Designer to provide a unified platform that enables rapid and smooth design-to-code workflows by way of natural language interaction. We integrate a natural language chatbot, a live code editor, with a drag-and-drop graphical user interface for visual layout, all within a single web application. Pages are visually designed by developers as they issue commands, observing code changes immediately reflecting the live preview, thus eliminating recompilation delays and reducing open tools. The design workflow process is streamlined, and errors are minimized by way of this tight coupling of design, code, and content generation. The performance of IntelliSite AI is also assessed to provide the best usability, accuracy, responsiveness, and user experience. On the whole, the suggested solution provides an approach that is effective and novel for streamlining the design workflow process with AI-augmented front-end development and effective evaluation.

INTRODUCTION

Front-end web developers deal with multiple challenges while designing and building websites. They must use diverse applications and libraries, which is time-consuming and error-prone. There is a need for an application that reflects live viewing of changes to the code and designs, and also minimizes the multiple applications and browser tabs required [1]. However, the complexity of controlling multiple applications, libraries, and browser tabs has become a significant challenge. Developers often need to switch between distinct environments due to disorganization and increased bug potential. The purposeful research project is to develop IntelliSite AI, which will be visualized as an innovative platform to address these provocations by creating intelligent refactoring, real-time code preview, and smooth design development. The Generative Pre-training Transformer (GPT) model is meant to be integrated to enable ease in the

web development process and help the developers to have an intuitive and structured platform that will require less app integration. Additionally, the performance model would be tested in this research project so as to deliver the smooth front-end development activities. Such an evaluation will lead to uncovering the strengths and weaknesses and eventually give the appropriate choice of AI model, especially to support the web development process. Large Language Models (LLMs) have demonstrated strong potential across software engineering tasks, including code generation, completion, summarization, and bug detection, highlighting a shift toward AI-assisted development workflows [2]. Although the AI model succeeds, it is very restricted in massive descriptions. Recently, the LLM's role in design-to-code tasks, translating mockups or natural-language specifications into functional UI has gained traction. For instance, Dong et al. introduced a self-collaboration framework that orchestrates multiple LLM agents (analyst, coder, tester) to enhance code quality [2]. In another research, the AgentCoder framework demonstrated that multi-agent LLM collaboration, which integrates programming, test design, and execution that improves code robustness on benchmarks like HumanEval and MBPP [3]. MetaGPT employed standard operating procedures in multi-agent collaboration to produce more coherent software engineering solutions [4]. However, concerns remain regarding the robustness and security of LLM-generated code. Tóth et al. analyzed GPT-4 generated PHP web apps and reported vulnerabilities in over 26% of samples, including SQL injection and XSS flaws [5]. Community reports highlight GPT-4's capability to autonomously discover such vulnerabilities [6]. Further, while frameworks like AgentCoder and MetaGPT advance multi-agent orchestration, they typically focus on backend or algorithmic tasks, not on front-end UI design or real-time code-preview workflows. Although multi-agent LLMs show promise in collaborative programming, few integrate conversational, drag-and-drop UI design with live preview capabilities. Despite these promising innovations, there are still several notable gaps that need improvement.

Existing tools typically focus on a singular modality, text, visual mockups, or code, forcing developers to switch contexts across applications, Integrated Development Environment (IDE), design platforms, and documentation [7]. While some systems offer dynamic updates, few provide true live coding previews tightly coupled with conversational control in one integrated workspace. Even natural-language tools rarely support iterative, dialogue-based design adjustments, often relying on static prompts, without maintaining conversational context across multiple refinement steps [8].

Managing the diverse development platforms and environments became a herculean task that is usually faced by front-end web developers. The utilization of diverse libraries, browser tabs, platforms, integrated development environments, and applications made developers' efforts challenging and delayed the development phase. Although there is accessibility to several tools, there is no single platform to provide integrated design development, live code preview, and advanced refractory. In accordance with the study [8], there is no availability of an Intelligent Web Application (IWA) that incorporates both web development and AI. There is an emerging need for a single-focused development platform or environment that can integrate design and coding manually while reducing the multiple browser tabs, applications, and libraries needed, and also use LLMs to improve the web development process experience automatically.

In this research, AI will be integrated into website development in the form of an Intelligent web designer that will allow developers to design web pages more efficiently by using a chatbot to interact in natural language and a graphical user interface to drag and drop components. IntelliSite AI is designed to directly address the highlighted limitations by presenting a unified development environment that integrates drag-and-drop UI design, live code editing, and instantaneous visual feedback, eliminating context switching. The embedding of an LLM-powered chatbot will enable natural-language control with memory of design intent and conversational context, supporting iterative refinements.

This manuscript is organized as follows: Section 2 focuses on the research methodology, IntelliSite AI functionality, with details of the process used to develop the platform. We also discuss the system implementation and its working. Section 3 provides an evaluation of IntelliSite AI, discussing the parameters, findings of the evaluation, and

limitations. Section 4 concludes the research with a discussion of future directions.

2. IntelliSite AI

The IntelliSite AI workflow describes how the system works as a hybrid in the generation of intelligent web designs using the synergy of NLP, ML, and Interactive UI Components. The purpose of this workflow is to bridge the communication divide between human creativity and automated web-development. The system is committed at the base to transforming user intent into executable and editable code, thus providing power to both non-technical users (who prefer to use conversational inputs) and technical designers (who prefer to manipulate design elements directly), as shown in Figure 1.

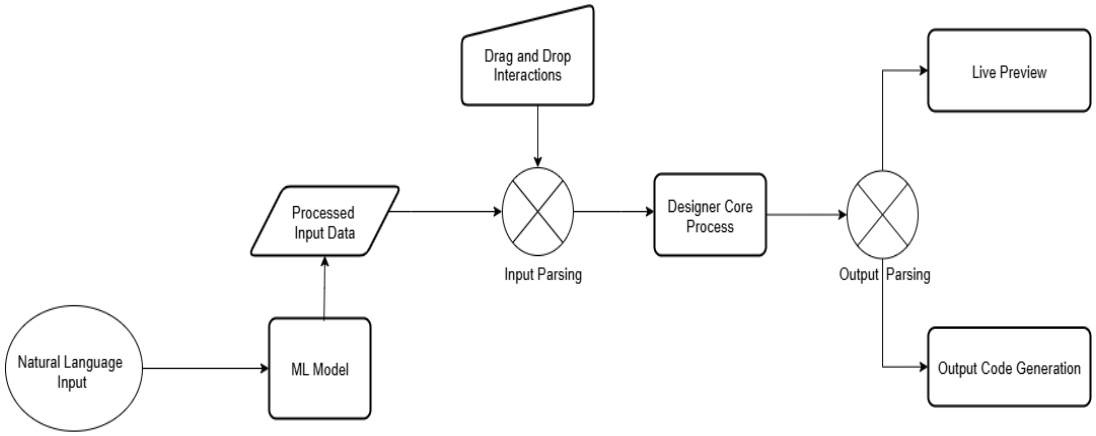

Figure 1: IntelliSite AI Workflow Structure

IntelliSite AI has the ability to empower both developers and non-developers to design modern web interfaces quickly, precisely, and creatively by combining NLP-based automation with real-time rendering. This AI-enriched architecture simplifies the design process while also representing a significant advancement in intelligent design systems that revolutionize the way of conception and implementation of web development.

As shown in Figure 1, the process starts with the user interacting with IntelliSite AI through natural language input, where the user can give commands like "Create a navigation bar with a logo on the left and links on the right" or "add a testimonial section with three cards". The first step in the system is intent recognition and semantic parsing of the command using the NLP Layer. The module looks at the grammatical and contextual structure of the sentence to isolate actionable objects (components like navigation bar, testimonial section), attributes (logo on the left), and the relationship between them. After being parsed, the system converts these interpretations to structured Processed Input Data, a standard format for the next step. Such pre-processed data will enable the smooth interaction between the NLP model and the ML Model and will guarantee uniformity in input translation and model flexibility in future training and optimization.

The Input Parser operates in two parallel but independent modes at this stage, allowing flexibility in how a user can behave:

i. Manual Code Creation (Canvas Interaction): Suited for visually-minded users, they have the option to create their own designs based on visual-only elements, where they can drag and drop design elements to their visualization. This is captured by the Input Parser, which then converts these interactions to code-ready formats, making them compatible with the Designer Core Processor.

ii. Automatic Component Creation (AI Model): In this case, through the use of natural language inputs, the NLP model of the system will automatically convert their commands into blueprints of components. Input Parser is used to check these AI-generated structures, and they are made to comply with proper hierarchies of design and the responsive design concept.

2.1 Working of IntelliSite AI

This section highlights the web development and AI Integration methods, processing, and working of IntelliSite AI.

2.1.1 Frontend Web Development

For the web development part, Agile methodology is appropriate when it comes to web-based application development, as one can receive feedback constantly and adapt to the development process. It is a gradual method of software development that focuses on flexibility and cooperation. IntelliSite AI frontend web UI methodology is built based on Redux design, as illustrated in Figure 2. It provides state management consistency across various interactive elements. This strategy was selected to facilitate the dual interaction framework of the system, drag-and-drop design, and chatbot-driven intelligent assistance while retaining real-time synchronization and responsiveness in the user interface. The UI structure of the site features four main interactive modules Canvas, Left Sidebar, Right Sidebar, and Chatbot.

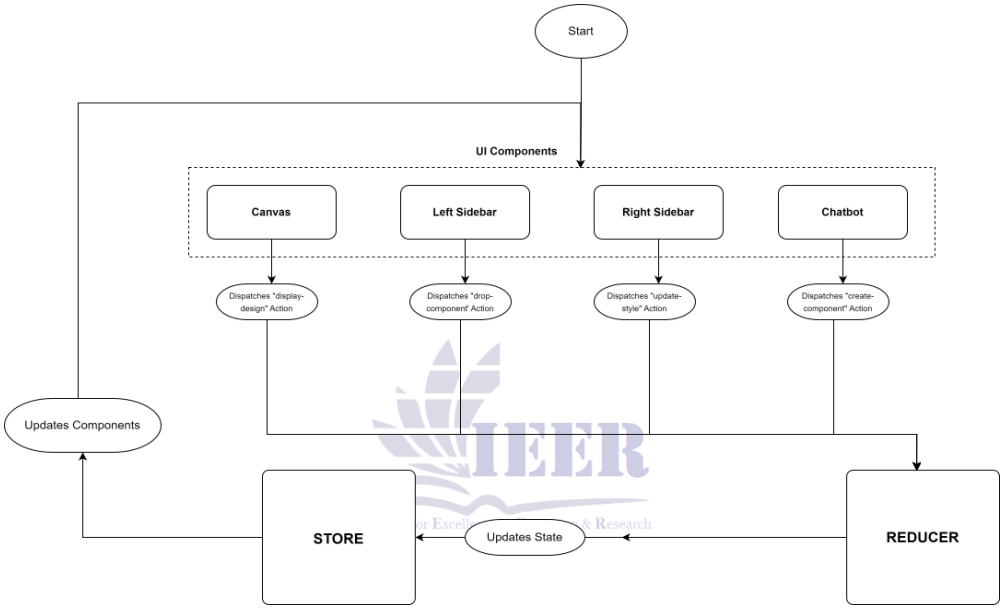

Figure 2: UI Design Methodology

Each of the modules has the task of responding to certain user interactions and sending corresponding actions to the Redux Store. Canvas is the central interactive interface on which the design of the IntelliSite AI tasks is visually created and displayed. Any changes to the design are controlled under a display-design action, which ensures that changes in the layout or structure are updated immediately. The left sidebar of IntelliSite AI provides users with drag and drop functionality with the pre-defined UI components as the component library. This type of interaction calls a drop-component action, which is further managed by the Store to dynamically amend the project design and structure. The right sidebar has various styling and customization options for selected components. All the styling attributes (color, font, size) are updated, and the update-style action is being invoked. This provides real-time visual properties changes throughout the design consistently. The natural language interface in the form of a chatbot allows the user to create or update components by using the NLP conversational prompts. The chatbot invokes an action of creating a component, which is seamlessly integrated with the Store, so a combination of AI-assisted design and manual processes is established. Any action that is dispatched becomes a part of the Store, and it is the only reference to how the application is. The Store continuously interacts with the Reducer to smoothly manage the process to update the design state based on the nature of the action it gained. The changed state is then transmitted back to the UI, and ensures that every element of Canvas, Sidebars, and Chatbot is automatically updated and kept in the same step. This strategy will

ensure that the IntelliSite AI offers a very engaging and smart frontend design experience.

2.1.2 AI Integration

The NLP algorithm that involves a backend server with customized chat completion functions that are enabled by the LLM to power the chatbot is applied in the implementation of AI. This is accomplished by the use of LLM, which understands the provided prompt and then asks the frontend chatbot to perform the required actions. From the AI aspect, our NLP solution applies custom chat completion functions using OpenAI to the chatbot. The use of LLM was performed in response to get the intended task done and comprehend the prompt, response command to the chatbot on the frontend to perform the necessary functions. The integration methodology of the IntelliSite AI that reveals the entire methodology of how the system will interconnect the user interface to the in-built AI model in Figure 3 to provide smart and contextually oriented web design advice.

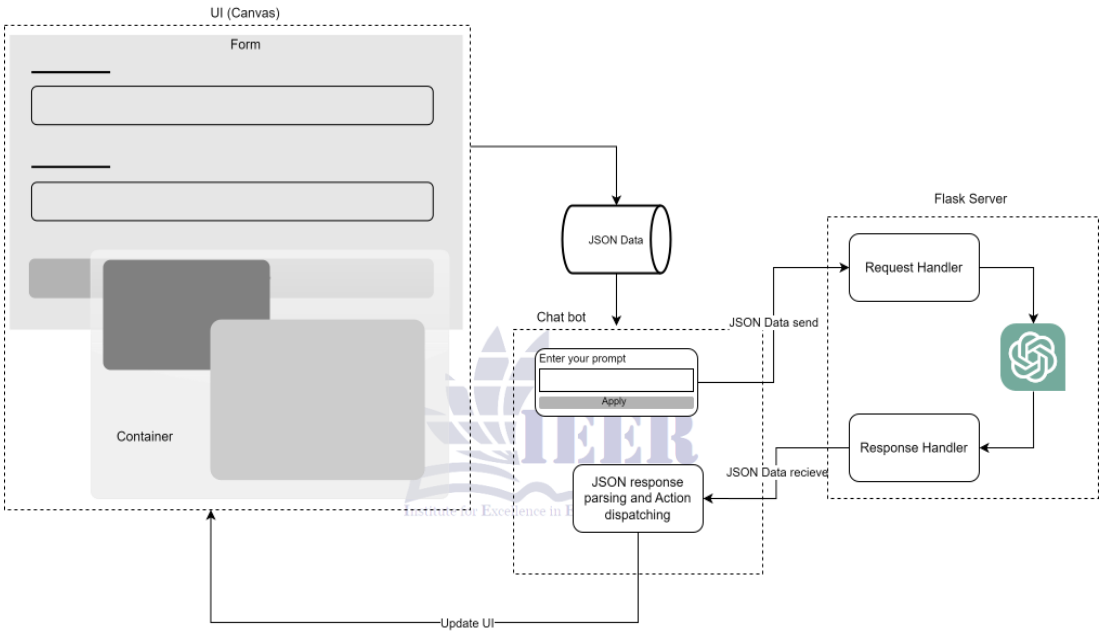

Figure 3: IntelliSite AI Integration

Once the JSON data is ready, it is then transmitted to the Request Handler where it is the responsibility of the Request Handler to send the request to the integrated OpenAI API. Request Handler takes care of the layer of communication so that the information is properly presented, sent safely, and placed in a contextually accurate position to the input demands of the AI model. After being fed the data, the AI model processes the prompt and comes up with a response, which contains intelligently structured output. This reply typically includes design advice, code segments, layout setups, or User Interface (UI) component system structures that reflect the query of the user. The AI model is actually the heart of the reasoning of the system, as it suggests logic-based ideas that appear as a design assistant in real time. The response generated by the AI is then sent back to the system in JSON format, and it is received by the Response Handler. Response Handler will take the responsibility of interpreting the data and deriving relevant information, and converting it into action commands that can be used by the front-end logic in the application. At this stage, the response in JSON and dispatching of actions occur, whereby the system decides what needs to be updated or created in the user interface. The system will then execute predefined Redux actions according to the content of the response of the AI, following the parsing process, and they include create-component, update-style, or modify-layout. This is transferred to the Store and then displayed in real time in the user interface.

2.2 IntelliSite AI Working

This section expounds on the user interface design, key design modules, real-time code visualization, and how the system dynamically combines AI and user interactivity.

2.2.1 User Interface

The IntelliSite AI UI has been designed such that it gives a smooth, user-friendly interface that allows collaboration between developers and designers to create and improve web pages. It has a simple and hierarchical workstation that consists of a main design canvas, interactive sidebars, and a built-in chatbot side panel. The layouts and web components can be visually manipulated by dragging and dropping elements without manually coding them, as presented in Figure 4. At the same time, Intelligent Chatbot, the embedded application, will support natural language instructions to automatically generate code and elements. As an example, a client only needs to type something such as, Add a navigation bar with logo and menu links, and the chat robot will automatically generate the design and code as well.

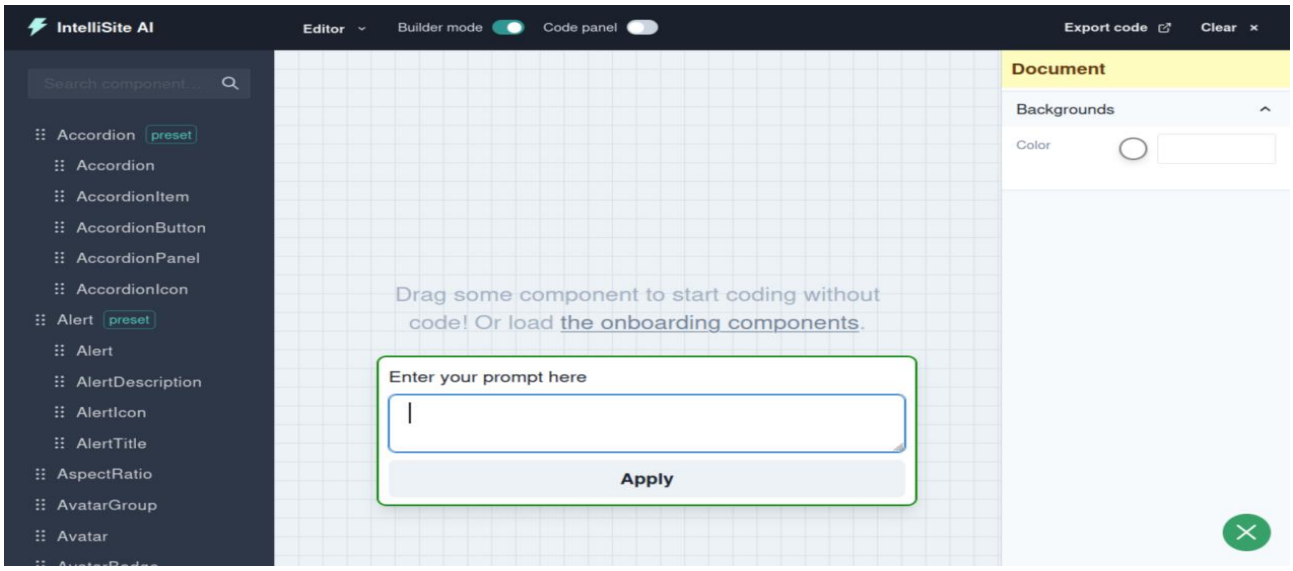

Figure 4. IntelliSite AI User Interface

The interface has been built mainly with ReactJS, which ensures that it has a modular and responsive framework. Next.js is the layer of the framework that makes use of React components to add greater rendering, routing, and server-side support. ReactJS and Next.js enjoy a certain synergy that enhances scalability, meaning that state management and high performance will be achieved in an environment of real-time design and preview sessions.

2.2.2 Drag-and-Drop Components

IntelliSite AI features a Drag-and-Drop Component System to improve accessibility and ease of development. The feature allows non-technical users like UI/UX creators, content creators, or product managers to create interfaces with their fingers without writing a single line of code. As each UI element is dragged onto the design canvas, it will automatically conform itself to grid alignments and hierarchical rules of nesting. As shown in Figure 5, the system cleverly understands the positioning of contexts, such as when a button is dropped within a navigation bar or container, and manipulates the related layout structure.

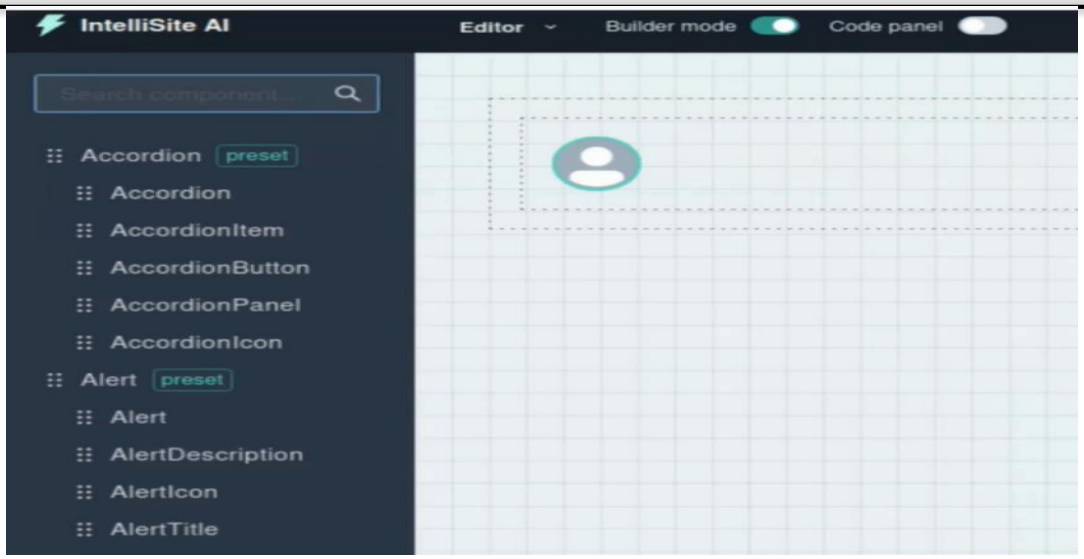

Figure 5: Drag-and-Drop Feature

The drag-and-drop engine is paired with an AI powered visual mapping, which ensures the user interactions are easily converted to code-compatible component hierarchies. This mechanism helps to bridge the gap between design intent and functional implementation by ensuring structural accuracy, both when doing visual composition and during generating code. In general, the feature saves design time, limits reliance on developers to make changes to the UI, and improves creative autonomy in multidisciplinary teams.

2.2.3 Real-time View of Changes

The ability to render in real-time is one of the key characteristics of IntelliSite AI; in this way, users could observe the immediate visual feedback of all the design actions they take. The Live Preview Engine dynamically displays changes to the interface as parameters are changed, elements are moved, or properties are modified by the user, without having to compile or refresh pages. This live synchronization of design view and codebase creates immediate feedback, resulting in quicker iteration of the design versus codebase, as shown in Figure 6.

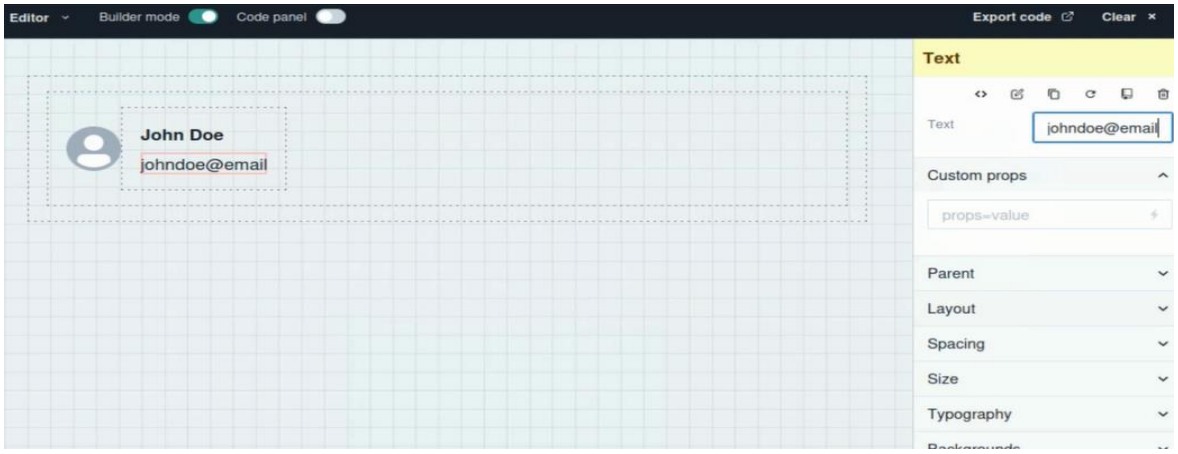

Figure 6: Real-time View of Code Modifications

This allows developers to visually verify their design changes and also peruse through the auto-generated React.js code to verify its correctness and maintainability. This interactivity is also real-time, meaning that inline editing can be done; therefore, the user can just click on the text or images in the preview box and edit them in place.

2.2.4 Code Generation

The design can be built through drag-and-drop actions or through natural language commands as understood by the AI chatbot, but the code that generates it is based on logic to ensure that the code is generated immediately and accurately. It is a real time translation where there is no compilation delay, implying that the user is able to view his/her code and verify its formatting as soon as it is modified. The code is created in a manner that will conform to industry standards (clean syntax, proper indentation, component modularization) as illustrated in as shown in Figure 7.

Figure 7: Code Generation

This is complemented by an export option, which enables the prompt access of the entire project files to be deployed and updated. Automating this traditionally complex process, IntelliSite AI saves a significant amount of labor-intensive workforce, removes duplication of effort, and provides a predictable correlation between the design and code structure.

2.2.5 Integrated AI Chatbot

The IntelliSite AI consists of an intelligent chatbot. It is an LLM-based interface that reads user instructions, performs tasks, and answers contextual questions during the design phase, as shown in Figure 8. The user can make natural language queries, like add a responsive footer that has contact information or change the background of the header to blue. These instructions are then processed by a chatbot with NLP and ML algorithms, which can identify the intent of the user and convert it into actions that can be implemented into a design.

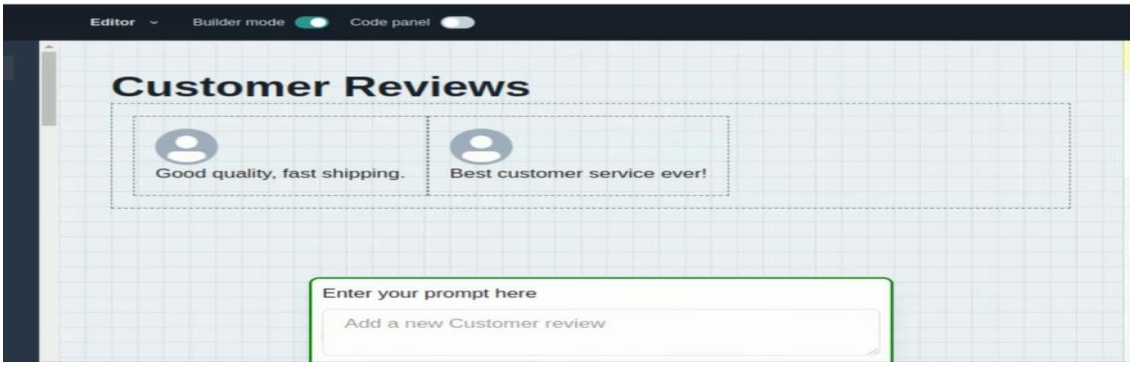

Figure 8: NLP Chatbot

In addition to the easy creation of components, the chatbot will be capable of managing multiple-step instructions, like nesting, styles, or responsive layouts. It communicates directly with the Designer Core

Processor to apply the requested changes in real time, and to show the changed appearance and the actual code.

In more detail, conversational context is preserved by the chatbot, so users do not need to repeat the whole instruction when giving a follow-up command. To give an example, once a button has been added, a user can just say the command to make something bigger or move it to the right, and the artificial intelligence will know which element is being discussed. This natural and human-like interaction streamlines the development process, reduces the learning curve, and professional web design is available to users with differing degrees of technical skill. The chatbot can therefore be seen as a creative partner as well as an automation enabler, fusing human intuition and AI accuracy. The IntelliSite AI interface combines all the intelligent features, such as drag-and-drop composition, real-time visualization, automatic code generation, and AI-assisted conversational design, into a single, unified application.

3. Evaluation and Discussion

The overall project performance evaluation methods, parameters, results, and limitations will be provided in this section.

3.1 Evaluation

To discover the efficiency and usability of IntelliSite AI in the context of front-end development, which is built on React JS and Chakra UI, it was thoroughly analyzed. The evaluation was made to determine how well the system supports users in converting the natural language instructions into practical and visually accurate UI elements. The evaluation was obtained from university students, learners, professional web developers, as well as UI/UX designers, to ensure that we had a well-balanced skill and experience set. The respondents were asked to rank the system on different performance and usability attributes such as accuracy of the system, system adaptability, system functionality, and system usability. Parameters of evaluation were the following:

i. Accuracy of Intent Recognition: Determines how well IntelliSite AI can interpret user prompts and convert them into crucial elements to the UI design and development process.

ii. Function Call Execution: This is the extent of responsiveness, speed, and accuracy with which the functions of the system can be performed.

iii. Usability Testing: Places focus on usability, design direction, and the sufficiency of interaction between beginners and professionals.

iv. Learning and Adaptability: Measures the ability of the AI to learn as the user continues using the interface, react to the disparate phrasing syntaxes, and respond to the likes of that particular user.

v. Error Handling and Recovery: How the system can deal with user errors and give appropriate corrective advice, as well as have a recovery process that will not affect the workflow.

3.1.1 Accuracy of Intent Recognition

Experimenting with various natural language instructions, including the varying phrasing styles, synonyms, and even a set of instructions with typing errors, to simulate real-world interactions. This was done to test the intelligence and the flexibility of IntelliSite AI in anticipating developer intent; 56 respondents submitted their feedback. They were able to design layout components, modify styles, and arrange elements using IntelliSite AI.

a. Proper Recognition of Items: The subjects assessed the ability of the chatbot to identify and produce the asked parts. 33.9% users reported a 100 percent accuracy in identifying the items, as shown in Figure 9, meaning that the chatbot was perfect in understanding the instructions and performing corresponding actions. 28.6% users reported 90 percent accuracy with occasional, but minor misinterpretations. The low accuracy scores (50-70) were not prevalent, which means that the IntelliSite AI was seen to be consistent in most cases.

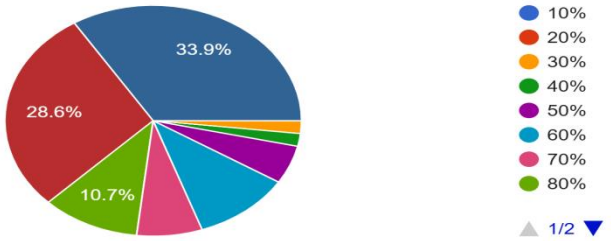

Figure 9: Proper Recognition of Items

b. Location and Positioning of Elements: This sub-parameter was used to gauge the level at which the AI put the components of the interface in the correct spot, as instructed by the user. IntelliSite AI was rated 10/10 on proper element location by 41.1% of the participants, as shown in Figure 10. This feature was rated 8 or higher by a combined 75% of respondents, suggesting that it is highly positional and has few layout-related errors. These results indicate that the layout understanding system of IntelliSite AI is powerful and can obtain absolute and relative positioning instructions effectively.

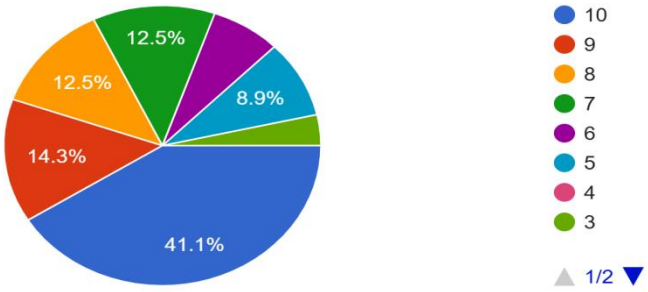

Figure 10: Position of Elements

c. **Attribute Accuracy (Color, Border, Size, Color):** This metric was how accurately the chatbot utilized design-specific attributes as per the user. The chatbot was rated 10/10 by 44.6% of users who tested the chatbot on the extent to which the chatbot used visual attributes, such as font size, color schemes, borders, and padding, as shown in Figure 11.

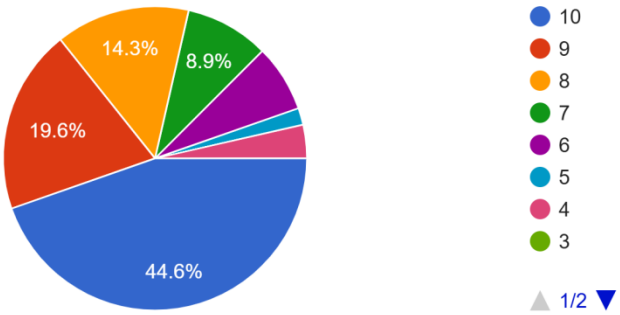

Figure 11: Attribute Accuracy

In addition, the fact that the system is capable of functionality with various versions of the natural language, including the ones that are characterized by synonyms and minor errors, depicts its flexibility and relevance in the real world. The high ranking in the context of routes of identification, placement, and attribute accuracy metrics is also justified by the high ranking of the program in the context of the natural language understanding and contextual mapping processes. The Accuracy of Intent Recognition test demonstrates that the IntelliSite AI is capable of addressing the emergent intention and delivering coherent and high-quality results. The overall result summary is presented in the Table1.

Table 1: Summary of Accuracy of Intent Recognition Findings

Parameter	Findings	Interpretation
Intent Accuracy	62.5% users reported 90-100% accuracy in identifying items.	Shows a strong understanding of user commands and reliable execution.
Position & Placement Accuracy	75% users rated element placement 8 or above.	Indicates accurate translation of layout and positioning instructions.
Attributes Accuracy	82% users rated attribute handling 8 or above.	Demonstrates high precision in applying visual and styling details.
Overall Intent Recognition	Around 70% users found the AI accurate in understanding prompts.	Confirms IntelliSite AI's consistent and dependable intent recognition performance.

3.1.2 Function Call Execution

The Function Call Execution parameter is used to estimate the performance and dependability of the IntelliSite AI to translate user instructions into the correct system functionality and execute them properly. Such an aspect is essential in defining whether the developing AI will succeed in developing the AI to learn and performing the action in the development environment. It is analyzed in terms of the ability of the system to execute some functions, such as theme updating, the addition of new components, and alterations to elements of an interface, without mistakes, incompatibility, and time lag.

a. Function Match Rate: Respondents were prompted to rate the accuracy of the AI in mapping their written instructions to the desired functions. The results are depicted in Figure 13, 32.1% rated the function as matching at 7/10, or sometimes the function was not fitting or was misinterpreted. One-quarter rated it 8/10, with an overall high level of accuracy and some inconsistencies. An almost perfect fit of instructions to executed functions was observed with a 19.6% rating of 9/10. 33.9% gave a rating of 10/10, indicating that in a significant proportion of instances, IntelliSite AI was able to read and match functions with high accuracy. These findings mean that there was a minor discrepancy in certain cases, although generally the AI showed a great deal of awareness about the intentions of users.

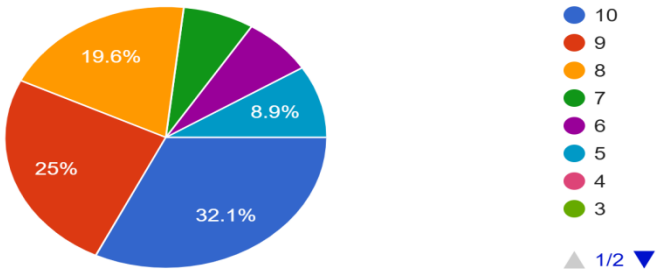

Figure 13: Function Match Rate

Execution Success Rate: This sub-parameter measured the probability of success of the functions that occurred after being matched and activated by the AI. The results were highly reliable in implementation, as shown in Figure 14: 44.6% of the respondents rated the AI 10/10 in the implementation of the functions, implying that the operations were carried out without any issues. The overall outcomes are summarized in Table 2.

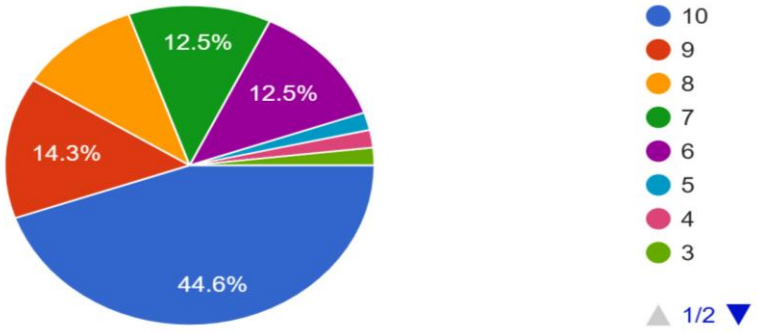

Figure 14: Functions Execution Success Rate

Table 2: Summary of Function Call Execution Findings

Parameter	Findings	Interpretation
Function Match Rate	34% users rated 10/10; majority rated between 8-9/10.	Shows strong accuracy in mapping user instructions to correct functions.
Execution Success Rate	45% users rated 10/10; 70% rated 8 or above.	Indicates reliable and smooth execution of commands without errors.
Overall Function Execution	Around 70% users rated the performance 8 or above.	Confirms IntelliSite AI's consistent and dependable handling of operations.

3.1.3 Usability Testing

The usability testing parameter can be described as an indicator of the efficiency and intuitiveness of the front-end development activities that are conducted using IntelliSite AI. The main idea was to find out whether users could fulfill the sought design tasks by chatting with the bot without facing any unnecessary complexity or misunderstanding. Three basic usability indications were evaluated:

a. Task Success Rate: Respondents were requested to measure the effectiveness of the chatbot in assisting them to meet the desired results. 28.6% rated success at 8/10, and 26.8% rated it 7/10. 19.6% rated it 9/10, and 14.3% gave a perfect 10/10 as shown in Figure 15. Overall, almost 9 out of 10 participants rated task success 7 and above, which confirms that IntelliSite AI helped users to achieve their desired objectives.

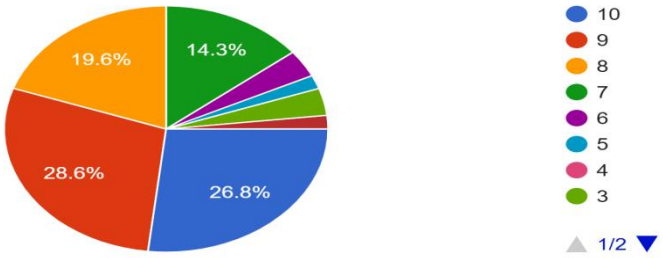

Figure 15: Task Success Rate

b. Task Completion Time: The measure assessed how fast tasks could be completed using the IntelliSite AI compared to the situation of manual performance. Completion time scored 2/10; 32.1% of the participants reported that there were activities that took longer than they were supposed to take. Only 12.5% rated it 10/10, while 16.1% rated it 7/10, as shown in Figure 16. The data point to the fact that despite the chatbot being a successful tool in meeting its tasks, it may not always deliver the same outcomes as quickly.

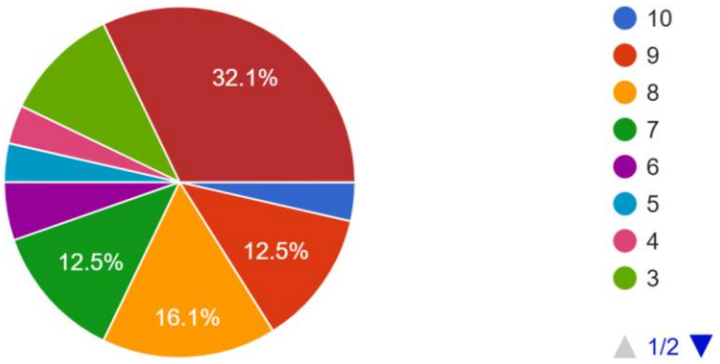

Figure 16: Task Completion Time

c. Number of Interactions: This sub-parameter was assessed by the efficiency of the task completion with regard to the number of interactions. 23.2 answered 9/10 and 17.9 answered 10/10, demonstrating that many users were able to complete tasks as depicted in Figure 17. However, 41 percent of respondents rated it as 2/10 or less, indicating that there can be some activities that require repetitive prompts or corrections in order to finish them. These findings indicate an opposing response, with reportedly no problem with the flow of interaction with simple tasks by experienced users, but a few novice users had a few over-communication problems.

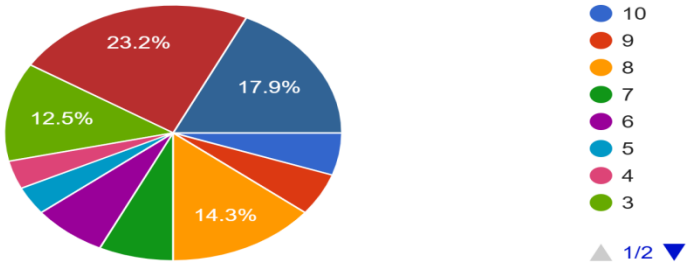

Figure 17: No of Interactions

The overall usability test results reveal that IntelliSite AI is performing well in the area of task success and average interaction efficiency, as depicted in Table 3, particularly among those users who were already experienced in the design and development field. However, the turnaround time could use a little bit more work, especially when dealing with more advanced tasks that require revision more than rewriting.

Table 3: Summary of Usability Testing Findings

Parameter	Findings	Interpretation
Task Success Rate	90% users rated success 7 or above.	Indicates strong reliability in achieving user goals accurately.
Task Completion Time	32% users rated 2/10, suggesting slower task execution.	Shows that while accurate, response speed needs improvement.
Number of Interactions	41% rated low (1–2/10), while 41% rated 9-10/10.	Reflects efficient performance for experts but inconsistency for beginners.
Overall Usability	High success but variable task speed and interaction steps.	Confirms usability strength with scope for optimization in execution time.

3.1.4 Learning & Adaptation

The parameter is the extent to which IntelliSite AI is learned in the process of the user interaction, and which is altered to adapt to changing demands as the design process advances. Personalization and Correction Handling were the adaptability indicators. This sub-parameter evaluated both the capacity of IntelliSite AI to remember user preferences in a sequence of prompts (remembering color preferences or layout settings) and the capacity to modify the outputs when the user changed or corrected the instructions.

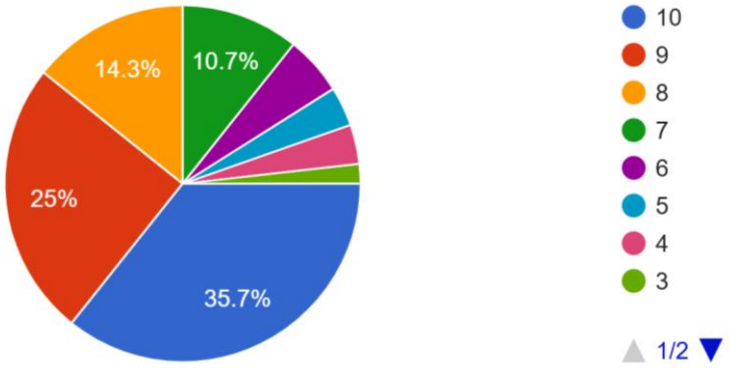

Figure 18: Personalization Adaptability

More precisely, 35.7% of the interviewees graded adaptability at 7/10 (moderate adaptation); 25 percent graded it at 8/10 and 14.3% at 9/10. Interestingly, the highest number of subjects (35.7) rated IntelliSite AI as a 10/10, as illustrated in Figure 18, suggesting that the system was able to memorize and apply preferences at various stages of interaction. Generally, the results of the survey show that IntelliSite AI performs well in terms of adaptability and learning. More than three-quarters of the respondents rated its flexibility at 8 or above, which is a successful personalization of user experiences and making corrections.

3.1.5 Error Occurrence Rate

This parameter is also related to the ambiguous, incomplete or invalid command coverage of the IntelliSite AI and its ability to recover the errors without interfering with the operation process. In order to check this, the participants were asked to provide intentionally ambiguous or erroneous inputs (e.g., misspelled instructions, incomplete prompts, or conflicting requirements). Then they talked of how many times the system had mishandled the command, or rebooted, or failed to cope with the command. Among the 56 survey responses, it was revealed that 41.1% of individuals who were assigned the responsibility of making decisions relating to errors were able to manage the decision using the 8/10 rate that proved to be effective in most cases. The remaining 12.5% rated it out of 7/10, and 10.7% rated it a perfect 10/10, as seen in Figure 19, indicating that it handled minor errors. Nonetheless, a considerable proportion of them, 23 (close to 41) of the participants, were satisfied with the rating of 1/10 and reported cases of total failure or crashes when implementing the system.

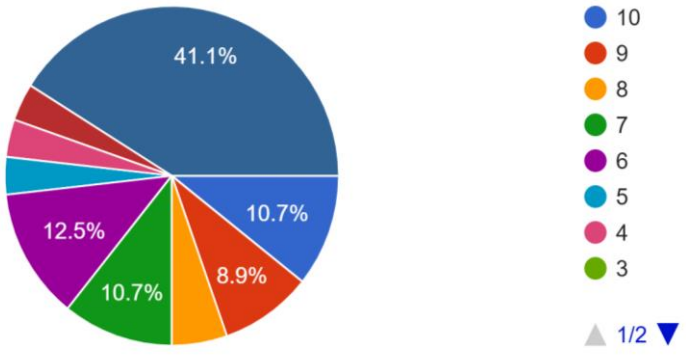

Figure 19: Crash Occurrence Rate

3.1.6 Overall User Satisfaction

This parameter shows how satisfied the users are after the execution of prescribed tasks with the help of IntelliSite AI. It is a comprehensive measure of the perceptions of the participants in the system with regard to its usability, accuracy, responsiveness, and efficiency in terms of the interaction. Post-task Feedback Score was the primary indicator of this parameter. Overall satisfaction was measured on a 1-10 scale with the ease of use, quality of the results, task completion rate, and the quality of interaction (need to be rated by themselves). 56 participants gave their comments. The findings showed quite a positive attitude to the performance of the system as depicted in figure 20, satisfaction was high with 58.9% of the participants rating at 9/10. One-fourth of respondents rated a perfect 10/10, which means the highest level of satisfaction and positive experience. 14 respondents (25 percent) scored the system 8/10, indicating good performance but with few areas of enhancement.

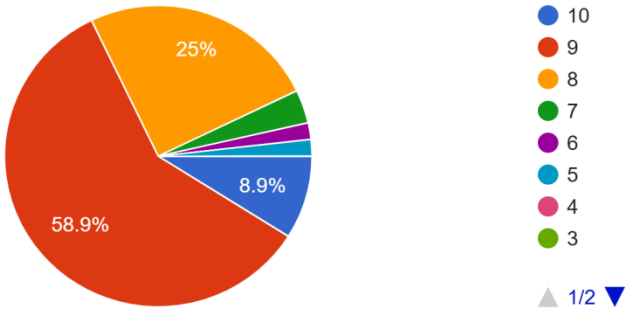

Figure 20: Overall Satisfaction Score

Overall, the survey results reflect extremely high levels of overall user satisfaction with the IntelliSite AI. About 92 percent of the respondents rated their experience 8 or more, which means that the system met or surpassed user expectations. Although minor issues, like delays in the response time, occurred periodically, the participants all emphasized the ease of interaction with IntelliSite AI, its ability to complete design-related tasks, and helpful guidance throughout project processes. These results verify that IntelliSite AI can offer significant practical value to front-end development, with smart automation and convenient interaction. Its remarkable appeal to beginner and professional developers implies a high chance of more widespread usage in AI-based web design setups.

3.2 LIMITATIONS

Despite these very positive results of IntelliSite AI on most of the parameters of the assessment, several flaws were identified that restrict the scalability of the solution. The elimination of these disadvantages will be an important process in further development.

3.2.1 Static UI

At the moment, IntelliSite AI creates static interfaces without dynamic logic and user interactivity. With the inability to simulate responsive layouts in the real world, users can visualize layouts effectively, but the lack of event handling or dynamic behavior limits the visualization. This shortcoming minimizes its applicability to interactive prototyping or in the production of designs.

3.2.2 Chakra UI Constraints

The Chakra UI integration of the system gives greater flexibility in development, but limits the components to the established Chakra ecosystem. The ability to import and combine third-party or custom-designed elements is not free, thus restricting the freedom of the creator and the expansion of designs.

3.2.3 Prompt Limitations

Complex or lengthy prompts that are past the token limit will give the IntelliSite AI a degraded performance. In these instances, the chatbot truncates

or produces partial answers, which decreases the quality of the generated code. This is especially noteworthy when using multi-layered design instructions or requests to generate detailed code.

3.2.4 Lack of Contextual Memory

The lack of contextual memory between interactions restricts the conversation between the chatbot and its user. With no ongoing memory, the IntelliSite AI will not remember past commands or preferences and will require the user to re-enter information in a sequential operation. This impacts on productivity and the flow of long-form design sessions.

3.2.5 Evaluation-Specific Limitations

The evaluation revealed that the Web Builder AI was performing well regarding the implementation of functions and usability; it was rated 8 or more by most of the participants on these aspects. Similarly, time spent to complete tasks was cited as one of the key areas of enhancements, with a few users citing slower work compared to manual development.

4. Conclusion

The IntelliSite AI development and evaluation is a major and important step towards programming the front-end in an automated manner through the use of a conversational style interface and smart design tools. The integration of a chatbot interface with a visual design environment allows developers to write, make edits, and preview UI components in a natural language, without having to write manual code, speeding up the workflow. The outcome of the analysis shows that IntelliSite AI is characterized by high functional accuracy, usability, and high user satisfaction. Nearly 92 percent of those who participated in the experience rated it well, and they realized the level of reliability of the system, the ease of use, and its capability to perform design tasks correctly. Even though the system still has some limitations, such as less speed in completion of tasks, it has exhibited high potential for future development. With the help of this assessment, it is possible to verify the feasibility and viability of introducing the use of AI-driven conversational design to the process of web development. IntelliSite AI provides more productivity, easier interaction, and creativity to amateur as well as professional coders by reconnecting the relationship between technical coding and user-friendly interface. The enhanced version of IntelliSite AI will incorporate the logic of mobile UI, context-memory, and adaptive learning entities so that it is not only a friendly design tool but also an intelligent web co-designer.

The advancement of the IntelliSite AI can be continued with the following recommended changes being introduced in the future to make the IntelliSite AI more versatile and ensure stronger functionality. The dynamic UI enhancements can add the logical interactivity and user event handling support so that the UIs can be generated dynamically. This will enable IntelliSite AI to generate functional, reactive interfaces that closely mirror real-world applications, which will make the tool both prototyping- and deployment-friendly. The Chakra UI extensibility can generalize the integration layer to enable adding of custom or external elements outside the Chakra UI framework. With the increased flexibility of components, more project-specific, visually diverse, and complex user interfaces will be feasible to developers. The advance token limit processing can utilize the smart best practices to optimize prompts including automatic summarization, chunking and token management, to ensure prompt truncation does not occur in lengthy commands and to ensure coherence in long commands. The advances will facilitate easier communication and more thorough AI feedback when performing intricate designs.

References

[1] D. R. Ellis, "How AI is transforming website design," *HubSpot*, 2024. [Online]. Available: https://blog.hubspot.com/website/ai-website-design. Accessed: Apr. 23, 2024.

[2] Y. Dong, X. Jiang, Z. Jin, and G. Li, "Self-collaboration code generation via ChatGPT," *arXiv preprint* arXiv:2304.07590, 2023. [Online]. Available: https://arxiv.org/abs/2304.07590. Accessed: May 15, 2024.

[3] D. Huang, J. Feng, H. Zhang, M. Liu, Q. He, Y. Sun, S. Zhang, L. Li, T. Wang, and Y. Chen, "AgentCoder: Multi-agent-based code generation with iterative testing and optimisation," *arXiv preprint* arXiv:2312.13010, 2023. [Online]. Available:

https://arxiv.org/abs/2312.13010. Accessed: Oct. 13, 2024.

[4] S. Hong, X. Wang, R. Zhang, H. Zhang, L. Wang, B. Li, J. Li, S. Liu, H. Wen, L. Chen, and T. Zhang, "MetaGPT: Meta programming for a multi-agent collaborative framework," *arXiv preprint* arXiv:2308.00352, 2023. [Online]. Available: https://arxiv.org/abs/2308.00352. Accessed: Nov. 16, 2024.

[5] R. Tóth, T. Bisztray, and L. Erdődi, "LLMs in web development: Evaluating LLM-generated PHP code—unveiling vulnerabilities and limitations," *arXiv preprint* arXiv:2404.14459, 2024. [Online]. Available: https://arxiv.org/abs/2404.14459. Accessed: Nov. 13, 2024.

[6] R. Mortensen, "GPT-4 autonomously finds vulnerabilities in websites," *Reddit: r/webdev*, 2024. [Online]. Available: https://www.reddit.com/r/webdev/. Accessed: Oct. 23, 2024.

[7] "ACM Research Article," *ACM Digital Library*. [Online]. Available: https://dl.acm.org/doi/full/10.1145/3607868. Accessed: Oct. 20, 2024.

[8] K. Wakil and D. N. A. Jawawi, "Intelligent web applications as future generation of web applications," *Scientific Journal of Informatics*, vol. 6, no. 2, p. 213, Nov. 2019.

[9] G. Fitzmaurice, "'Context switching' is a major drain on developer productivityhere's how GitHub plans to solve that," *IT Pro*, May 23, 2024. [Online]. Available: https://www.itpro.com/software/development/context-switching-is-a-major-drain-on-developer-productivity-heres-how-github-plans-to-solve-that. Accessed: Oct. 15, 2024.

[10] C. Wood, "[Prototype] LLM Drag & Drop Website Builder (Spring 2024)," *Christopher Wood Portfolio*, Feb.–Mar. 2024. [Online]. Available: https://portfolio.christopherhwood.com/llm-drag-drop-website-builder. Accessed: May 15, 2025.

[11] M. Milanović, "Context-switching is the main productivity killer for developers," *Tech World With Milan Newsletter*, Feb. 6, 2025. [Online]. Available: https://newsletter.techworld-with-milan.com/p/context-switching-is-the-main-productivity. Accessed: Jul. 15, 2025.

[12] I. Tkanov, "Too many tabs open? Why real multitasking is hard," *IT Blog*, 2025. [Online]. Available: https://igortkanov.com/too-many-tabs-open-why-real-multitasking-is-hard/. Accessed: Jul. 15, 2025.

[13] Workona, "How to fix the problem of too many tabs," *Workona Blog*, 2025. [Online]. Available: https://workona.com/blog/how-to-fix-too-many-tabs-problem/. Accessed: Jun. 15, 2025.

[14] J. Lively, J. Hutson, and E. Melick, "Integrating AI-generative tools in web design education: Enhancing student aesthetic and creative copy capabilities using image and text-based AI generators," *DS Journal of Artificial Intelligence and Robotics*, vol. 1, no. 1, pp. 23–33, 2023, doi: 10.59232/AIR-V1I1P103.

[15] A. Ayyagiri, P. Goel, and A. Renuka, "Leveraging AI and machine learning for performance optimization in web applications," *International Journal of Scientific Research in Engineering and Management (IJSREM)*, vol. 8, no. 2, pp. 1–8, 2024, doi: 10.55041/IJSREM15294.

[16] Y. Loboda, O. Trofymenko, S. Manakov, and V. Hura, "Artificial intelligence in modern web development and web design: Multilevel classification and systematization," *Computer Modelling and Intelligent Systems (CMIS)*, vol. 2, no. 1, pp. 1–15, 2025, doi: 10.32782/CMIS.2025.2.1.1.

[17] Robotics and Automation News, "Artificial intelligence and the future of web design," Jul. 29, 2024. [Online]. Available: https://roboticsandautomationnews.com/2020/07/29/artificial-intelligenceand-the-future-of-web-design/34559/. Accessed: Jul. 29, 2024.

[18] Thinhdanggroup, "Function calling with OpenAI ChatGPT," GitHub Pages. https://thinhdanggroup.github.io/function-calling-openai/. Accessed: Aug. 02, 2024.