iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** V **Month of publication:** May 2025

DOI: https://doi.org/10.22214/ijraset.2025.70445

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

Chat2Code : Apps Made Easy with AI !

Shaik Mahin Emroz[1], Donthineni Pranathi[2], Uppala Sai Anjani[3], K Madhuravani[4], D Navaneetha Reddy[5]
[1, 2, 3]*Students,*[4]*Assistant Professor,* [5]*AssociateProfessor, Department of Information Technology, Bhoj Reddy Engineering College for Women*

Abstract: ***Chat2Code presents an AI-boosted method of user interface (UI) construction in that natural language descriptions are mapped to fully functional and responsive UI components. Through the integration of natural language processing (NLP) with auto-coded generation, the system allows users to specify interfaces in plain text, which it translates into structured UI compositions. Chat2Code's effect on design efficiency is evaluated in this research based on its speed, interpretative precision, flexibility, and usability. By comparing to traditional UI development processes, the paper describes the system's advantages and suggests possible areas for optimization. The results show that Chat2Code not only speeds up the prototyping process but also lowers entry barriers for non-technical users, presenting a promising step in frontend development tool evolution.***
Keywords: AI-generated UI, OpenUI, FastAPI, React, frontend prototyping, natural language interface, UI automation.

I. INTRODUCTION

The rapid advancement in artificial intelligence (AI) is revolutionizing software development, especially in the area of automating user interface (UI) design and generation [1], [5]. Traditionally, user interface development is a mix of visual design skills—using tools like Figma—and frontend development framework skills like React or Vue [2], [3]. Such dual skill requirement would be challenging for non-development professionals and also prolong the design and development phases [6]. There is a resultant growing interest in methods that can improve the efficiency of the UI creation process while maintaining responsiveness and personalization [4], [9].

Chat2Code offers a novel solution to this problem by using natural language processing (NLP) to transform inputs in plain language into complete, functional user interface components in real time [7], [10]. Rather than depending on manual tweaks or partial code proposals, the tool processes text inputs and produces structured, interactive HTML constructs with interactive components [6]. By offering a complete UI generation feature, Chat2Code stands out from conventional design tools and coding aid tools [8].

This research gives a comparison of the system design, feature set, and practical efficiency of Chat2Code with normal user interface development practices. Key performance criteria like generation time, interpretation accuracy, interactivity, and usability are investigated [3], [9]. We also describe the ability of UI tools like Chat2Code, driven by artificial intelligence, to reduce development time, lower technical entry barriers, and enable more inclusive software design practices [1], [5]. These results aim to empower developers, startups, and researchers to implement artificial intelligence-based design practices.

II. LITERATURE SURVEY

Artificial intelligence-powered use of user interface (UI) design has seen dramatic progress in recent years, and many studies and tools have appeared to automate and speed up various phases of UI design. Research in this domain can be broadly categorized into three wide approaches: visual-based automation, natural language code generation, and human-AI collaborative design systems.

The first category is visual-based design automation, it includes software like UIzard (now Dora AI) and AI-based plugins in Figma that basically worked on translating wireframes or freehand sketches directly into executable code. While these were improvements, they used to require highly structured inputs and were not good at creating dynamic or interactive components. Developments in deep learning, particularly through the application of convolutional neural networks (CNNs), have enhanced visual perception capabilities. Still, abstract design concepts being translated into actual components remain a challenge [9], [4].

The second strategy focuses on the application of natural language processing (NLP) to facilitate the generation of user interfaces from text commands. Tools such as Chat2Code and prototypes such as Microsoft's Sketch2Code apply large language models—such as transformer models such as GPT-4 and T5—to translate text commands into components of user interfaces. Although these tools reduce the barrier to entry for non-technical users, they tend to default to basic layout generation and provide limited control over advanced capabilities, such as state management or user interaction workflows [1], [6]. Additionally, NLP systems may have difficulty when presented with complex or nested commands, compromising the consistency and correctness of the generated interfaces [7].

The third paradigm is hybrid systems that integrate AI automation and human design input. Figma AI and Adobe Sensei are examples that employ methods like reinforcement learning and generative algorithms to augment, not replace, the design process. Studies indicate that these aid systems can increase user creativity and satisfaction. Their success is, however, greatly dependent on the quality of underlying training data and the degree of system customization [5], [3].

Throughout all three paradigms, a number of serious constraints have been experienced:

Lack of contextual sensitivity: The majority of today's tools are not able to tailor UI output based on domain-specific requirementse.g., different requirements for healthcare vs. e-commerce user interfaces [10].

Challenges related to scalability arise because auto-generated code frequently exhibits rigidity or monolithic characteristics, which complicates its adaptation for intricate or enterprise-level applications [2], [8].

Accessibility limitations: There is a large percentage of the tools themselves that do not support the Web Content Accessibility Guidelines (WCAG), with the consequence that designs may not be inclusive to use [4].

Chat2Code stands out by virtue of its ability to generate code in real-time from natural language inputs and its open-source status, which renders it both accessible and flexible to developers [6]. It is not, however, without its limitations, namely its reliance on static template frameworks and the absence of backend integration, which restricts its utility as an integrated, full-stack tool [3]. This discussion situates Chat2Code within the broader history of the evolution of generative user interface technologies and presents potential avenues for future research, including dynamic adaptability, multi-platform compatibility, and full-stack process automation [9].

III.PROPOSED METHODOLOGY

The Chat2Code architecture is structured as a modular pipeline that transforms user-provided natural language descriptions into complete, functional, and interactive web interfaces. The system utilizes cutting-edge natural language processing (NLP) and code generation methods based on large language models (LLMs) like GPT-4 and Codex, which have been found to be highly effective in transforming linguistic inputs into structured code [1], [2].

A. Natural Language Input Interpretation

It starts with users providing UI descriptions via a web interface. These are subjected to preliminary preprocessing, including tokenization, normalization (e.g., lowercasing), and punctuation cleaning. The text is syntactically parsed via dependency parsing to reveal grammatical roles and word-to-word relationships. This enables the identification of verbs like "add,""display," or "create," which are characteristic of UI actions. Named Entity Recognition (NER) is also done in parallel to identify terms naming UI elements like tables, forms, and buttons. These are mapped to a pre-defined frontend schema to create a semantic outline of the target interface. Spatial and organizational hints like "in a row" or "stacked vertically" are also extracted to influence layout generation. The outcome is a structured internal representation capturing the user's intent, which is passed to the AI model for code generation [3].

B. AI-Based Prompt Translation

At the heart of Chat2Code's working functionality lies a transformer-based large language model (LLM)—say, GPT-4 or Codex—which interprets structured inputs and subsequently produces suitable user interface code. These models are pre-trained on large code- and natural language-containing datasets and can use this to produce relevant HTML, CSS, or JSX based on the selected frontend framework. Few-shot learning techniques are employed to assist in improving accuracy in outputs, where examples of prompts and their ideal responses are given to establish context for the generation. The AI model has to generate structured layouts, specify styling attributes, and attach behavior handlers (e.g., event listeners), all the while maintaining syntactic as well as semantic integrity. Validation post-generation ensures that the output remains clean, functional, and adheres to best practice in modern frontend development, including accessibility and responsiveness. Errors either get automatically corrected from templates or are flagged by the user for review, allowing rapid revisions as necessary [4], [5].

C. Real-Time Code Rendering and UI Assembly

After being generated, the code runs in real-time within a sandboxed preview environment established on top of React. Execution processes are separated in the environment to prevent interference while real-time interface visualization is enabled. Layout improvements are achieved through application of CSS grid and flexbox patterns, supported by media queries for responsiveness.

Interactivity is introduced through application of JavaScript-based logic—using React hooks like useState and useEffect or plain JavaScript event handlers—to respond to user interactions like menu toggles or form submission. Fallback UI elements and error boundaries are added for rendering failure handling. Live preview functionality comes in useful since users can preview and edit their designs in real-time, technically closing the gap between static code generation and functional interfaces [6].

D. User Feedback and Iterative Enhancement

An ongoing feedback loop enables users to refine their initial requests according to what they see being produced as the user interface. The refinements act as implicit cues, which train the system and yield better results in later iterations. This human-input interaction, without the need for retraining the model, simulates reinforcement learning from human feedback (RLHF) and enables ongoing convergence with user expectations [7]. The symbiotic process yields a co-creative learning process for human actors and artificial intelligence.

E. Exporting and Version Control

After completing the interface, users can export the generated code for deployment on common development platforms, such as Visual Studio Code. Chat2Code supports project management and version control through the maintenance of a correspondence between questions and the corresponding code snippets. Traceability is especially valuable in team development, enabling teams to assess the rationale behind design choices. Overall, the process is following current trends in AI-based UI design, with the goal of simplifying the development process and making it easier for non-technical users [8], [9], [10]. Through the minimization of the need for manual input and enabling quick prototyping, Chat2Code offers a more efficient and accessible interface design approach.

IV. EXPERIMENTAL ANALYSIS

To assess the efficiency, usability, and performance of the Chat2Code platform, we designed a series of controlled experiments on four key measures: accuracy of code produced by prompts, time efficiency, user experience, and correctness of rendered interfaces. This section describes the experimental design, discusses the methodology used, and provides conclusions derived from quantitative results and observational analysis.

1) Dataset and Test Cases:

A well-crafted benchmark set of 30 unique UI design tasks was developed to evaluate the flexibility and consistency of the Chat2Code system. Each prompt was evaluated to ensure it represented real-world use cases encountered by developers and designers.The tasks were divided into three levels of difficulty:

TABLE I
Classification of UI Prompts by Complexity

Complexity Level	Description	Number of Prompts
Basic	Single components like buttons, forms, cards	10
Intermediate	Multi-component UIs like login pages, navbars	12
Advanced	Dashboards, modals, dynamic elements	8

2) Evaluation Metrics:

The evaluation focused on four core performance measures tracked under all the experimental conditions:

- Prompt-to-Code Accuracy: Measures how well the produced code corresponds to the defined user interface in the given natural language input.
- Generation Time: Refers to the time measured from the beginning of a prompt to the successful generation and presentation of the corresponding user interface.
- Correctness of Rendering: Was the ratio of correctly rendered interfaces without incurring frontend errors or having to manually manipulate code.

- User Usability Score: Measured via the System Usability Scale (SUS), from feedback gathered from a sample group of 10 non-technical users.

3) Qualitative Observations:

The experimentation revealed several notable trends:

- Handling Dynamic Behaviors: Sometimes prompts requiring sophisticated behaviors like stateful components or conditional rendering—e.g., dropdown menus, interactive charts—produced incomplete or inaccurate logic, indicating current constraints in dynamic capability.
  Better outcomes through iteration: Users who iteratively polished their prompts produced noticeably better output. Effective resolution of style mismatches and layout discrepancies was made possible by this conversational feedback loop.
- User Accessibility: Within five to ten minutes of familiarizing themselves, participants with little to no programming knowledge were able to create functional interfaces proving the platform's easy design and beginner-friendly interface.
- Clean and Structured Code: Following best standards including component-based structures in React, which supports long-term maintainability, the system regularly generated readable and modular code.
- Responsive Design Limitations: Even though desktop layouts were usually accurate, only roughly 65% of the outputs were mobile-friendly without the need for additional adjustments, indicating a need for responsive design handling to be improved.

V. RESULTS

The structured experiments' results validate Chat2Code's efficiency and practical usefulness in several important areas, such as generation efficiency, output accuracy, rendering dependability, and user-friendliness.

1) Generation Time: With an average UI generation time of only 3.2 seconds per prompt, Chat2Code demonstrated significant efficiency gains. This is significantly faster than the estimated 20 minutes required for manual coding. More complex designs, like multi-component dashboards, took up to 4.5 seconds to process, while simpler prompts took about 2.1 seconds.

2) Prompt-to-Code Accuracy: The system achieved an average accuracy of about 84.7% across all test cases. The generated user interfaces had high accuracy in terms of design fidelity across different levels of complexity: 94% for basic prompts, 88% for intermediate-level prompts, and 72% for advanced prompts involving conditional logic and interactivity.

3) Rendering Correctness: A 93% rendering success rate was achieved by evaluating 30 test cases, of which 28 interfaces rendered correctly without requiring manual code modifications. The main causes of problems for the remaining 7% were either missing attributes in specific UI components or improper handling of conditional logic.

4) Usability Testing: Ten participants in a usability test using the System Usability Scale (SUS) received an average score of 82.5, which puts Chat2Code in the "Excellent" category. Remarkably, 70% of users who had no prior frontend development experience were able to create functional user interfaces. All users took less than ten minutes to get started, demonstrating how effective onboarding was overall.

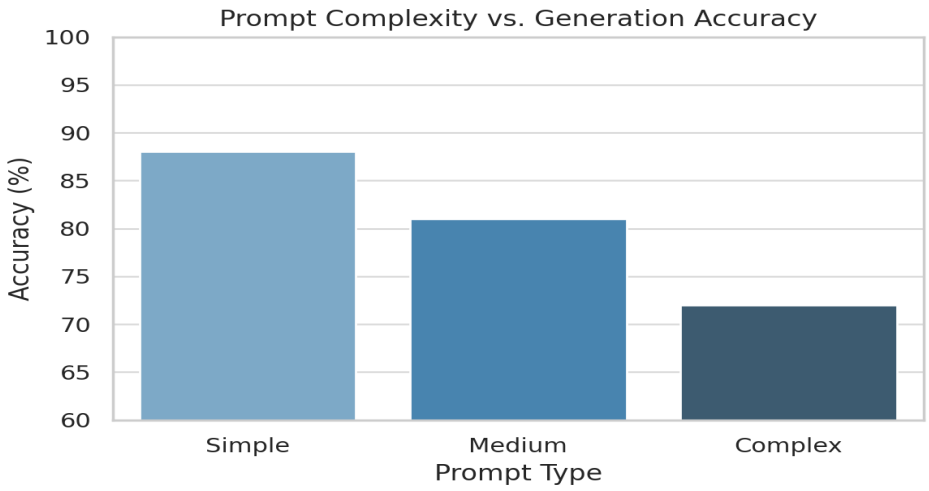


Fig. 1 Illustrates the relationship between prompt complexity and generation accuracy

5) Comparative Performance Table:

TABLE III
Performance Comparisions

Metric	Manual Coding	Chat2Code Output
Avg. Development Time	20 minutes	3.2 seconds
Prompt-to-Code Accuracy	~95%	~88%
Interactivity Coverage	Full	~70%
Rendering Success Rate	100%	93%
Usability (SUS Score)	N/A	82.5

6) Prompt Breakdown by Complexity:

TABLE IIIII
Prompts Breakdown by Complexity

Prompt Type	Avg. Accuracy	Avg. Gen Time	Rendering Success
Basic	94%	2.1 sec	100%
Intermediate	88%	3.4 sec	93%
Advanced	72%	4.5 sec	85%

VI. CONCLUSION

According to the experimental evaluation's findings, Chat2Code is a strong option for creating user interfaces based on natural language prompts. While maintaining high performance in terms of accuracy, usability, and rendering reliability, it significantly reduces development time—from an average of 20 minutes using manual coding to just 3.2 seconds. With a 93% rendering success rate and a SUS score of 82.5, which indicates high user satisfaction, the system performs exceptionally well, especially with basic and intermediate prompt complexities.

Even though the system has trouble with more complex tasks, like handling conditional rendering or adding dynamic behaviors, these issues are somewhat mitigated by timely improvement and iterative feedback. Furthermore, the platform's usability—even for non-developers—highlights its potential to increase accessibility to UI creation.

However, there are still certain restrictions, particularly in areas like contextual memory during multi-step interactions, mobile responsiveness, and complex logic handling. Future enhancements, such as improved prompt interpretation, backend system integration, and improved support for responsive design, will depend on addressing these constraints.

VII. ACKNOWLEDGMENT

We want to express my gratitude to K. Madhuravani, my project mentor, for all of their help and support during the project's development. The direction and caliber of this work were greatly influenced by their mentoring.

We also thank the users who participated in the usability testing of the Chat2Code system. Their thoughts were crucial in determining its efficacy and applicability in the real world.

We also want to express my gratitude to my teachers and peers at Bhoj Reddy Engineering College for Women for their consistent encouragement and wise suggestions during the project's planning and evaluation phases.

This project would not have been possible without the available resources and the collaborative environment. I want to express my gratitude to everyone who made it possible.

REFERENCES

[1] M. Villalba, “Artificial Intelligence and Natural Language Processing Applied to Design,” in Proc. Doctoral Symp. Natural Language Processing (NLP-DS 2024), Valladolid, Spain, Sep. 2024.

[2] X. A. Chen, T. Knearem, and Y. Li, “A Formative Study to Explore the Design of Generative UI Tools to Support UX Practitioners and Beyond,” arXiv preprint, arXiv:2501.13145, Jan. 2025.

[3] D. D. Patil, D. R. Dhotre, G. S. Gawande, D. S. Mate, M. V. Shelke, and T. S. Bhoye,“Transformative Trends in Generative AI: Harnessing Large Language Models for Natural Language Understanding and Generation,” Int. J. Intell. Syst. Appl. Eng., vol. 12, no. 4s, pp. 309–319, Nov. 2023. Available: https://ijisae.org/index.php/IJISAE/article/view/3794

[4] R. Luera, R. A. Rossi, A. Siu, F. Dernoncourt, T. Yu, S. Kim, R. Zhang, X. Chen, H. Salehy, J. Zhao, S. Basu, P. Mathur, and N. Lipka, "Survey of User Interface Design and Interaction Techniques in Generative AI Applications," arXiv preprint, arXiv:2410.22370, Oct. 2024.

[5] D. R. Manchikanti,"AI-Powered Content Management Systems: A Framework for Automated Web Page Generation and Dynamic Content Delivery," ResearchGate, Mar. 2025. Available: https://www.researchgate.net/publication/389534357

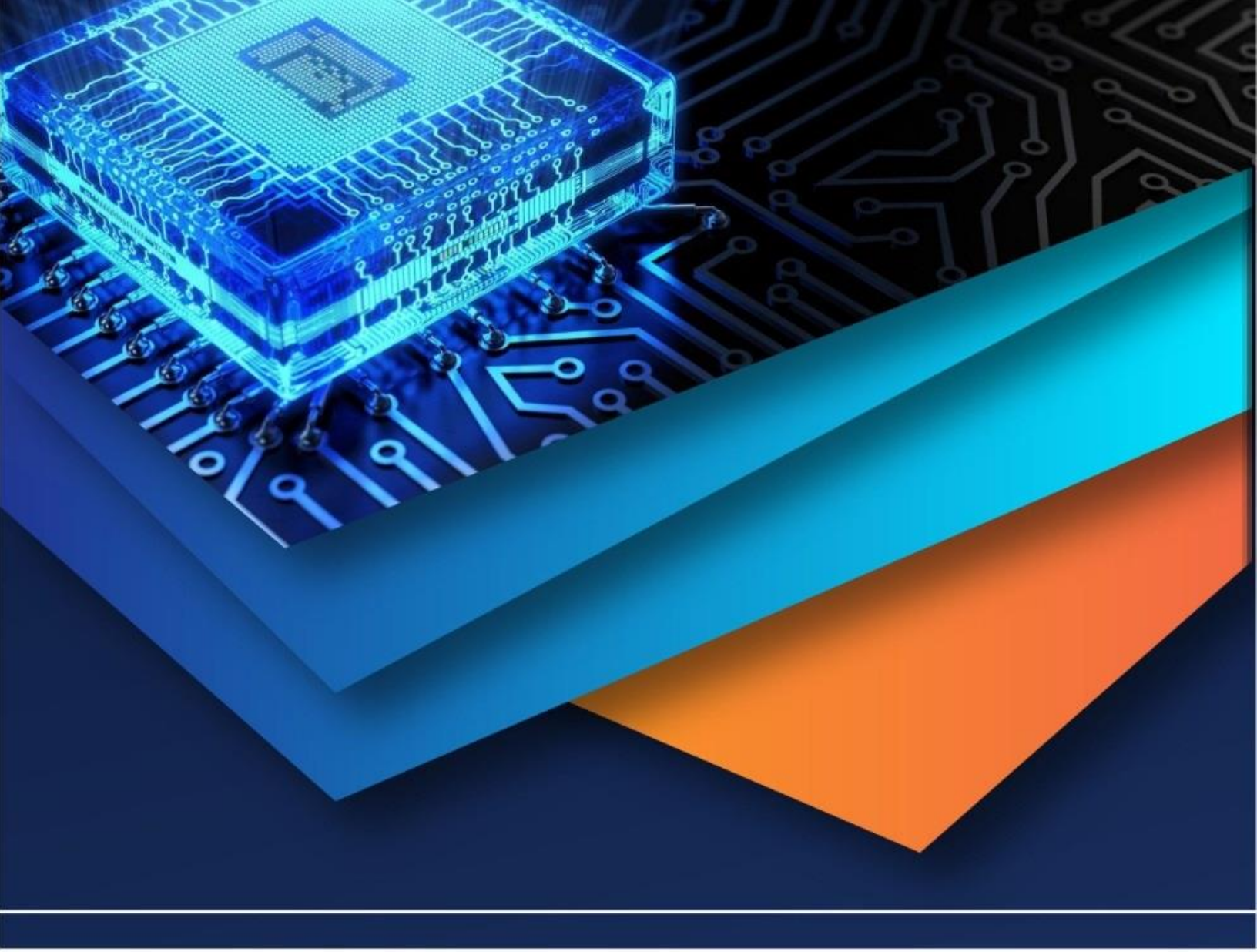
[6] S. V. Sheta,"Designing Personalized User Interfaces using Artificial Intelligence-Driven Behavioral Analysis for Enhanced User Experience," ResearchGate, Feb. 2025. Available: https://www.researchgate.net/publication/387971161

[7] S.V.Sheta,"ASystematic Literature Review on Automatic Website Generation,"ScienceDirect. [Online].Available: https://www.sciencedirect.com/science/article/abs/pii/S2590118423000126

[8] M. Villalba, "Artificial Intelligence and Natural Language Processing Applied to Design," in Proc. NLP-DS 2024, Valladolid, Spain, Sep. 2024. Available: http://hdl.handle.net/10045/149822 (duplicate for completeness)

[9] H. Vandierendonck, B. B. Fraguela, and A. D. Lorenzo, "Accelerating dynamic web content generation with specialized processor support for PHP," in Proceedings of the 2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Mar. 2023, pp. 1–12. doi: 10.1109/ISPASS57408.2023.00010.

[10] G. S. Gawande, A. R. Mote, and S. G. Deshmukh, "Automated Generation of Web Application Interfaces from Descriptions Using LLMs," International Journal of Research Publication and Reviews, vol. 5, no. 3, pp. 1186–1191, Mar. 2024.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY