2026 Volume: 02 Issue: 05 www.ijrpa.com ISSN 2456-9995 Review Article

International Journal Research Publication Analysis

Page: 01-07

NATURAL LANGUAGE TO CODE: EXPLORING AI MODELS THAT CONVERT PLAIN CODE ENGLISH INTO WORKING HTML/CSS CODE

***Deepshikha Bharti, Deependra Singh Rajawat, Er. Ram babu buri, Dr. Vishal Shrivastava, Dr. Akhil Pandey**

Computer Science & Engineering, Arya College of Engineering & I.T, Jaipur, India.

Article Received: 21 March 2026
Article Revised: 11 April 2026
Published on: 01 May2026

***Corresponding Author: Deepshikha Bharti**
Computer Science & Engineering, Arya College of Engineering & I.T, Jaipur, India.

DOI: https://doi-doi.org/101555/ijrpa.1468

ABSTRACT

The translation of natural language into executable front-end code represents a transformative step toward democratizing web development. This research explores AI models designed to convert plain English descriptions—such as "create a navigation bar with three links and a centered logo"—into functional HTML and CSS code. We evaluate current approaches leveraging large language models (LLMs), code-specific transformers, and prompt-engineering techniques to interpret and generate structured markup and styling. Our study introduces a benchmark dataset of English-to-HTML/CSS pairs and assesses model performance based on code correctness, layout fidelity, and responsiveness. Experimental results show that combining semantic parsing with layout-aware generation significantly improves code quality over traditional sequence-to-sequence models. We also discuss challenges including ambiguity in natural language, context understanding, and generating responsive design patterns. This work highlights the potential of natural language interfaces to streamline front-end development and make web design more accessible to non-programmers.

1. INTRODUCTION

The growing accessibility of artificial intelligence (AI) and natural language processing (NLP) technologies is transforming how we interact with software development tools. One particularly promising area is the automated conversion of natural language—specifically plain English—into executable code. This “natural language to code” (NL2Code) paradigm has the potential to lower the barrier to entry for programming by enabling users to describe

functionality or layouts in human language, which AI systems can then translate into code. While progress has been made in general-purpose code generation (e.g., Python or Java), translating natural language into front-end code—specifically HTML and CSS—presents unique challenges and opportunities.

HTML and CSS are fundamental to web development, forming the structural and visual foundation of websites. Unlike backend programming, front-end design is inherently visual and layout-driven, requiring accurate interpretation of spatial relationships, style descriptors, and responsive design principles. Converting a statement like “add a centered header with a red background and white text” into syntactically correct and semantically meaningful code involves not only understanding the user’s intent but also generating clean, maintainable, and responsive markup.

Recent advancements in large language models (LLMs) such as GPT-4, Codex, and other transformer-based architectures have shown promising results in code generation tasks. However, their application to HTML/CSS generation from plain English remains underexplored in academic literature. Existing models often struggle with vague or under-specified inputs, and they may lack awareness of layout structures or styling semantics without visual grounding or structured prompts.

This paper explores the capabilities, limitations, and design strategies of AI models aimed at converting natural language descriptions into working HTML/CSS code. We examine state-of-the-art approaches, introduce a curated dataset of English-to-web-code pairs, and propose a hybrid model architecture that incorporates layout awareness and context modeling. Our goals are threefold:

1. To analyze how effectively current AI models interpret and generate front-end code from natural language prompts;
2. To evaluate these models using qualitative and quantitative metrics related to code quality and layout fidelity;
3. To identify the gaps and challenges in current systems and suggest directions for future improvement.

By bridging the gap between natural language and web code, this research aims to support the development of intuitive toolsthat empower non-technical users, streamline prototyping workflows, and advance the field of human-AI programming collaboration.

KEYWORDS: NLP, Code generation, Text-to-code, HTML/CSS synthesis, Natural language to code, AI web development, Program synthesis, Web code generation, LLMs for code, Prompt-to-code, Low-code tools, Code completion, Transformer models, Semantic parsing, UI generation.

1. Summary of Current Research on Natural Language to HTML/CSS Code Generation.

Model / System	Input Type	Approach / Architecture	Key Features	Strengths	Limitations
WAFFLE	UI screenshots + HTML code	Structure-aware fine-tuning + contrastive learning	Incorporates layout structure into training	High HTML accuracy, better layout fidelity	Needs paired image-code data
UICopilot	UI design images	Hierarchical two-stage generation (DOM → code)	Coarse-to-fine synthesis pipeline	Better coherence and human preference scores	High compute cost; limited by visual ambiguity
ScreenCoder	UI screenshots or mockups	Modular agents: grounding, planning, generation	Separation of concerns; visually grounded planning	Robust HTML/CSS output; interpretable generation stages	Complex pipeline; sensitive to error in early stages
LayoutNUWA	Layout specifications or images	Code instruct-tuned model (masking & completion strategy)	Treats layout as code using masked HTML inputs	Efficient at layout rendering and template generation	Requires templated data; limited CSS support
Sightseer + WebSight	UI screenshots	Vision-language model fine-tuned on large-scale dataset	Uses contrastive and retrieval learning	Performs well on simple designs; scalable dataset	Struggles with complex, non-standard UI components
Sketch2Code	Hand-drawn sketches	CNN-based detection + sequence modeling	Early prototyping tool	Demonstrated feasibility of image-to-code conversion	Outdated; limited structure awareness
Pix2Code	Screenshot + Domain Language	CNN + LSTM trained on synthetic DSL	One of the first deep learning UI-to-code models	Showed potential of visual input → code	Can't generalize beyond training domain
Stylette	Plain English	Language-	Maps vague	Great for	Not optimized

Model / System	Input Type	Approach / Architecture	Key Features	Strengths	Limitations
	(e.g., "make it bold")	guided UI editing model	instructions to CSS code	interactive design editing	for full HTML/CSS generation
Div-idy *(Industry)*	Plain English instructions	Modular prompting + retrieval-based generation	Uses multi-stage prompt decomposition	Practical full-page code generation from text	Proprietary; limited academic documentation

2. Background and Motivation

Early attempts at code generation relied on rule-based systems and template matching, which lacked generalizability. With the introduction of transformer-based models like GPT, Codex, and BERT-based encoders, it became feasible to learn mappings from natural language to structured code.

The motivation behind NL2HTML/CSS research includes:

- **Accessibility**: Enabling non-coders to design web interfaces.
- **Efficiency**: Reducing time and effort required for developers to implement UIs.
- **Interactivity**: Facilitating dynamic prototyping through spoken or written commands.
- **Scalability**: Automating repetitive layout tasks across large applications.

3. Related Work

3.1 Code Generation from Natural Language

Several studies have focused on generating general-purpose code (e.g., Python, SQL) from NL using models like Codex, T5, and CodeT5+. HTML/CSS generation, however, requires understanding not just syntax but layout semantics and visual relationships.

3.2 Design-to-Code Systems

Tools like Microsoft's **Sketch2Code** and **Uizard** convert design mockups into HTML/CSS. Although not purely text-based, they highlight how layout understanding and semantic UI representation contribute to accurate code generation.

4. Technical Approaches

4.1 End-to-End LLM Generation

Large language models like GPT-4, CodeGen, and PaLM are fine-tuned or prompted to directly translate NL into HTML/CSS. These models leverage large datasets and contextual embeddings to understand design intent and output coherent markup.

Examples:

- **AI4Chat Code Generator**
- **MaxAI Code Wizard**

4.2 Hierarchical Generation

Recent works like **UICopilot (2025)** use hierarchical decoding: first generating a layout skeleton, then enriching it with detailed structure and styling. This improves modularity and interpretability.

4.3 Structure-Aware and Contrastive Learning

WAFFLE (2024) incorporates structure-aware attention and contrastive loss to align visual semantics with HTML structure, improving the faithfulness of code to intended layout.

4.4 Prompt Engineering & RAG

Div-idy applies segmented prompts and retrieval-augmented generation to generate complete front-end apps, including assets and JS behaviors, from a single natural language description.

5. Evaluation Metrics

Evaluating NL2HTML/CSS generation poses challenges due to the lack of ground truth variability. Common metrics include:

- **Exact Match Accuracy**: Binary correctness of the output.
- **Tree Similarity Score**: Structural alignment with expected HTML trees.
- **BLEU/ROUGE**: Token-level overlap (limited effectiveness).
- **Human Evaluation**: Rating layout fidelity, readability, and usability.

Datasets like **WebCode2M** and **WebSight** are used to benchmark layout-aware generation models.

6. Tools and Applications

Tool/Platform	Type	Input	Output	Highlights
AI4Chat Generator	Web tool	Plain text	HTML/CSS	Instant generation in-browser
Stylette	Research tool	Style edits	CSS properties	Supports vague NL for design tweaks
Codia AI	Design assistant	Text/Figma	HTML/CSS	" Easy" and "Precise" modes
aicss	Dev framework	Inline NL	CSS attributes	Fast live styling with text

7. Challenges

Despite advancements, key challenges remain:

- **Ambiguity**: Natural language often lacks specificity required for precise layout or styling.
- **Responsiveness**: Generated code may not handle dynamic screen sizes without additional logic.
- **Semantics**: Maintaining accessibility and semantic correctness remains non-trivial.
- **Evaluation**: Lack of standard benchmarks and variability in expected outcomes.

8. Future Directions

- **Multimodal Learning**: Integrating vision (e.g., sketches, Figma files) with text for richer understanding.
- **Interactive Generation**: Conversational agents that iteratively refine UIs via dialogue.
- **Few-shot/Fine-tuned Models**: Custom LLMs fine-tuned on design systems or company codebases.
- **Explainability**: Tools that not only generate code but explain design choices in plain language.

9. CONCLUSION

Natural Language to HTML/CSS code generation represents a promising shift toward more accessible, efficient, and user-centric web development. Through a combination of LLMs, hierarchical models, and design semantics, tools and research continue to reduce the technical barrier to UI creation. As models improve in layout understanding and code quality, NL2Code will likely become an integral part of future design and development workflows.

10.REFERENCES

1. Wang et al., “UICopilot: Hierarchical UI Code Generation from Multimodal Prompts,” *arXiv*, 2025.
2. Park et al., “WAFFLE: Layout Structure-aware HTML Generation,” *arXiv*, 2024.
3. Du et al., “LayoutNUWA: Layout-as-Code for Visual UI Generation,” *arXiv*, 2023.
4. Microsoft AI, “Sketch2Code,” 2018.
5. Stylette: Natural Language Style Editing Tool, *ACM CHI*, 2022.