International Journal of Engineering Research & Science (IJOER) ISSN: [2395-6992] [Vol-12, Issue-3, March 2026]

Sketch2Code: AI-Powered Design to Interactive UI

Rushikesh Patil[1*]; Harsh Tambe[2]; Om Wagh[3]; Prof. Akshata S. Raut[4]

[1-3]Department of Computer Engineering, VIVA Institute of Technology, India
[4]Assistant Professor, Department of Computer Engineering, VIVA Institute of Technology, India
*Corresponding Author

Abstract— *Sketch2Code is an intelligent AI-driven platform designed to bridge the gap between user interface design and software development by converting hand-drawn UI sketches into fully functional application code. The system leverages advanced computer vision techniques to accurately detect and classify interface components, while large language models are employed to generate structured, human-readable, and production-ready code. In addition to code generation, the platform provides a live preview feature that enables immediate validation of the generated output. Unlike conventional prototyping tools that produce illustrative designs, Sketch2Code focuses on delivering deployable code suitable for real-world applications. Users can upload sketches, select the desired output format, and instantly obtain code with functionalities such as copy-to-clipboard, code download, and real-time preview. By automating a traditionally manual and time-consuming process, the platform significantly reduces development effort, lowers technical barriers for non-technical users, supports rapid prototyping for startups and enterprises, and enhances learning in UI/UX and web development. Overall, Sketch2Code accelerates the transition from conceptual ideas to working interfaces, making application development faster, more efficient, and more accessible.*

Keywords— Sketch2Code, artificial intelligence, computer vision, large language models, sketch-based UI, source code generation.

I. INTRODUCTION

The Sketch2Code project aims to transform hand-drawn user interface (UI) sketches into functional application code, addressing the growing demand for rapid prototyping and efficient design-to-development workflows. This approach is particularly useful in domains such as UI/UX design, software education, hackathons, and early-stage product development, where ideas must be quickly converted into working interfaces. Traditional methods require manual translation of sketches into code, which is time-consuming and demands technical expertise. Sketch2Code leverages advancements in computer vision and artificial intelligence to automate this process [1], [5].

The system utilizes OpenCV for image preprocessing and optical character recognition (OCR) techniques to extract textual labels from sketches [2]. YOLO-based object detection models are employed to identify and classify interface components such as buttons, text fields, and images [3], [9]. These detected elements are then provided as structured input to large language models, which generate clean, human-readable React.js or HTML/CSS code while preserving the original design intent [4], [13].

To ensure usability and accessibility, a web-based platform developed using React, Tailwind CSS, and Node.js allows users to upload sketches, preview the generated layout in real time, and download production-ready code. By integrating OCR, deep learning, object detection, and modern web frameworks, Sketch2Code significantly reduces manual coding effort, accelerates development cycles, and effectively bridges the gap between creative sketches and deployable applications.

II. MATERIAL AND METHODS

For the purpose of this research, a systematic and experimental methodology was adopted to design and develop the proposed Sketch2Code system, which aims to automatically convert hand-drawn user interface sketches or mockup images into functional front-end code. The methodology focuses on integrating image processing techniques, deep learning–based object detection, structured data representation, and large language model–based code generation to achieve an efficient end-to-end UI prototyping pipeline [1], [5], [10].

The overall process was implemented using a combination of software tools and frameworks including Python programming language, OpenCV library for image preprocessing, the YOLOv5 deep learning model for UI component detection, and large language models for intelligent HTML and CSS code generation. A live preview environment was also developed to validate and customize the generated user interface code [2], [8].

2.1 Proposed Detailed Architecture

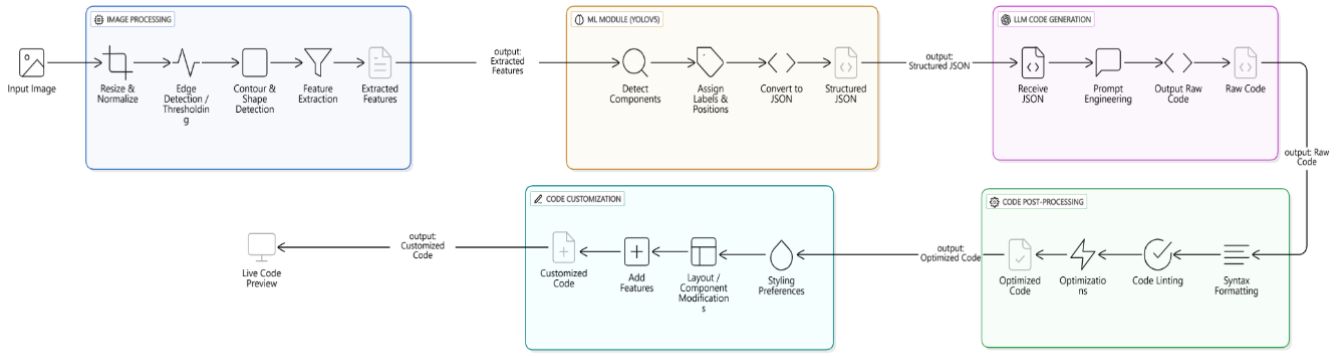

FIGURE 1: Block Diagram / Working of Sketch2Code

The methodology is divided into several sequential stages to ensure systematic processing and accurate transformation of visual design input into deployable front-end output.

Stage 1: Image Acquisition and Preprocessing — The input UI sketch or mockup image was resized and normalized to maintain uniform dimensions and improve computational efficiency [7]. Image enhancement techniques such as edge detection, thresholding, and contour analysis were applied to highlight component boundaries and identify distinct layout regions. Feature extraction methods were then used to obtain meaningful visual descriptors that support precise component recognition in subsequent stages [2].

Stage 2: UI Component Detection and Classification — Using the YOLOv5 object detection model, the preprocessed image features were provided as input to the trained detection network, which identified common interface elements such as buttons, text fields, images, icons, and text areas. For each detected component, the system generated class labels and bounding box coordinates representing the spatial position and size of the element within the layout. These detection results were further organized into a structured JSON format to preserve component attributes, hierarchical relationships, alignment patterns, and layout grouping information [3].

Stage 3: Structured Layout Interpretation and Code Generation — Carefully designed prompts were utilized to guide a large language model in interpreting the JSON-based component representation. Based on spatial constraints, hierarchy, and semantic attributes, the model generated corresponding HTML tags and CSS styling rules. Layout mechanisms such as Flexbox and Grid systems were automatically inferred to maintain design consistency, proportional spacing, and alignment similar to the original sketch [4].

Stage 4: Code Post-Processing and Optimization — Formatting and linting procedures were applied to correct syntax inconsistencies and enforce standardized coding practices. Redundant containers, duplicate styling rules, and inefficient layout structures were eliminated to produce clean and maintainable code. Additional optimization strategies were implemented to ensure responsive behavior across different screen sizes and devices [5].

Stage 5: User-Driven Customization and Live Preview — The optimized code was rendered in an interactive preview environment, allowing users to visualize the generated interface in real time. Designers and developers were able to modify visual attributes such as colors, typography, spacing, and component positioning without altering the underlying structural logic. This stage ensured that the final interface output met both functional and aesthetic requirements [6].

Thus, the proposed methodology establishes a comprehensive pipeline that transforms a hand-drawn sketch or UI mockup into structured, optimized, and deployable front-end code. Each stage contributes to preserving visual design intent while reducing manual development effort and accelerating the rapid prototyping process [1].

III. RESULTS AND DISCUSSION

The implementation of the proposed Sketch2Code system produced significant results in terms of automated user interface understanding, structured layout generation, and functional front-end code synthesis. The developed system was tested using

multiple hand-drawn and digitally created UI sketches to evaluate its ability to accurately detect interface components, preserve layout hierarchy, and generate visually consistent front-end outputs.

3.1 Sketch Processing

Initially, the system successfully processed input sketches through the upload and preprocessing module. The sketches were normalized and enhanced using image processing techniques, enabling clear boundary extraction of UI elements. The preprocessing stage improved the visibility of structural components such as headers, content sections, cards, and navigation areas, thereby increasing detection accuracy in subsequent stages.

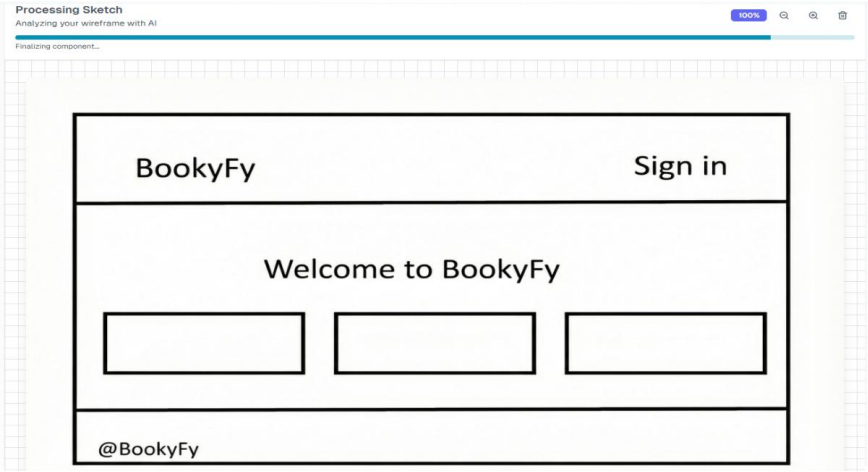

FIGURE 2: Sketch Processing Interface

3.2 Component Detection

During the component detection stage, the YOLOv5-based model demonstrated effective identification and localization of UI elements. The system was able to detect multiple components such as headers, text blocks, content cards, and layout containers with high confidence scores. The detected elements were visually highlighted with bounding boxes and categorized into semantic component types. This stage enabled the transformation of raw visual data into structured layout information.

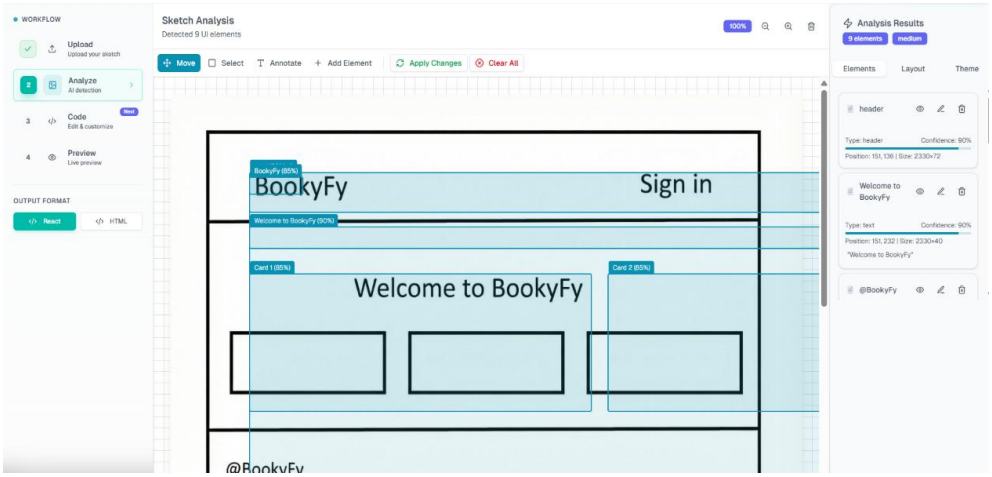

FIGURE 3: Detected UI Components with Bounding Boxes

3.3 Structured Layout Representation

Following detection, the structured layout representation was converted into JSON format containing positional attributes, hierarchical grouping, and spatial alignment constraints. This structured representation served as an intermediate bridge between visual understanding and code synthesis. The organized component metadata allowed the system to preserve the design intent and maintain proportional spacing among UI elements.

3.4 Code Generation

In the code generation phase, the large language model successfully translated structured layout data into functional HTML and CSS code. The generated code incorporated semantic tags, responsive layout techniques such as Flexbox and Grid systems, and styling attributes inferred from component positioning. The integrated code editor environment enabled users to review and refine the generated code efficiently.

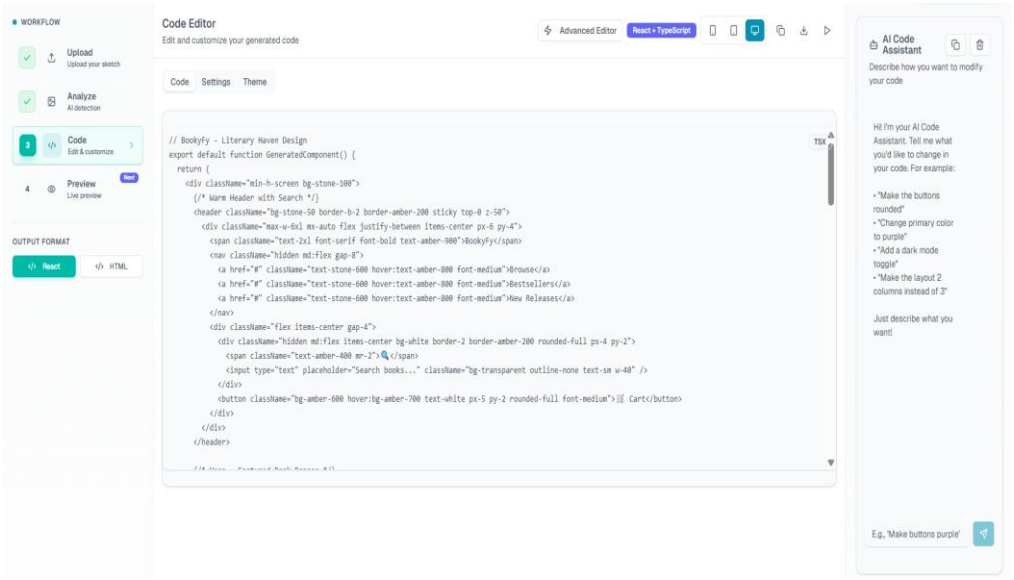

FIGURE 4: Generated Front-end Code in Editor Environment

3.5 Live Preview and Final Output

Further, the system provided a real-time live preview module that allowed users to visualize the rendered interface instantly after code generation. This interactive preview ensured validation of layout correctness, alignment consistency, and overall usability of the generated UI design. Users were able to make design adjustments and observe immediate visual feedback, thereby enhancing rapid prototyping capability.

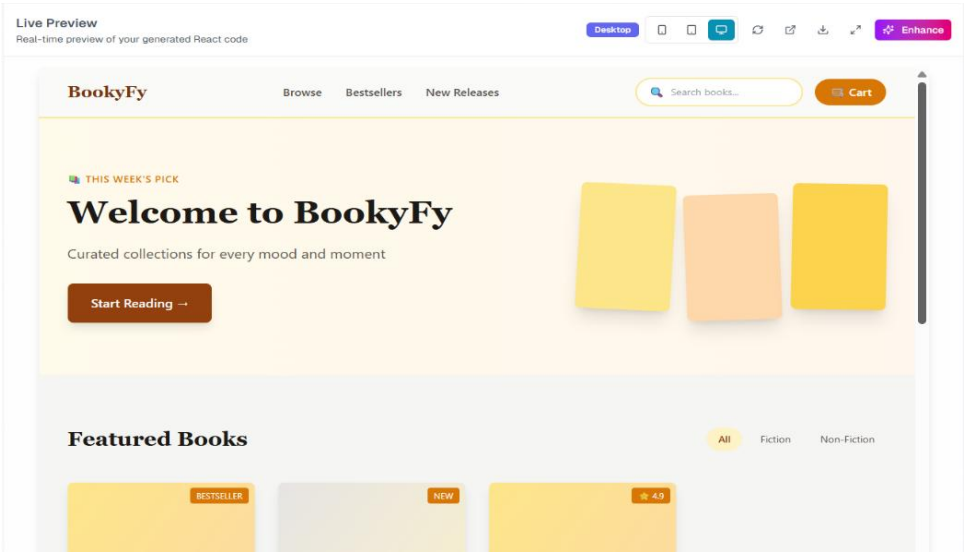

FIGURE 5: Final Rendered Interface Output (Live Preview)

3.6 Discussion

Overall, the experimental results demonstrate that the proposed system effectively reduces manual effort involved in UI development while maintaining structural accuracy and visual consistency. The integration of deep learning–based

component detection with intelligent code generation enables a seamless transformation from sketch input to deployable front-end interface. The obtained results confirm that the proposed approach provides a practical and efficient solution for accelerating the UI design workflow in modern web development environments.

Key Performance Indicators:

Stage	Key Outcome
Preprocessing	Successful normalization and enhancement of input sketches
Component Detection	Accurate detection of buttons, text fields, headers, cards
Structured Representation	JSON conversion preserving hierarchy and alignment
Code Generation	Production-ready HTML/CSS with Flexbox/Grid
Live Preview	Real-time validation with customization options

IV. CONCLUSION

The proposed Sketch2Code framework provides an efficient approach for automatically converting hand-drawn user interface sketches into functional front-end code by integrating image processing, deep learning–based component detection, structured layout understanding, and large language model–driven code generation. The system improves UI development productivity by reducing manual coding effort while maintaining design consistency and layout accuracy. Experimental implementation demonstrates its capability to generate responsive and customizable interfaces with real-time preview support.

Future work will focus on:

- Enhancing detection performance for complex and overlapping sketches
- Expanding component datasets to cover more UI elements and patterns
- Integrating advanced multimodal models (e.g., Vision Transformers) for improved accuracy
- Supporting additional output frameworks (Vue.js, Angular, Svelte)
- Enabling iterative design feedback loops for enhanced practical adoption

ACKNOWLEDGMENT

The authors would like to acknowledge the open-source research and developer community for their valuable contributions to technologies and frameworks such as YOLOv5, OpenCV, React, and large language model platforms, which supported the development and implementation of the proposed Sketch2Code system.

CONFLICT OF INTEREST

The authors declare no conflict of interest regarding the publication of this paper.

REFERENCES

[1] Bouças, T., & Esteves, A. (2026). Converting web page mockups to HTML using ML. *[Journal Name], 1*(9), 7.

[2] Beltramelli, T. (2017). Pix2Code: Generating code from a graphical user interface screenshot. *arXiv*, arXiv:1705.07962.

[3] Calò, T., & De Russis, L. (2022). Style-aware sketch-to-code conversion for the web. In *Companion of the 2022 ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS '22 Companion)* (4 pages). ACM.

[4] Kumar, Y., Gordon, Z., Alabi, O., Li, J., Leonard, K., Ness, L., & Morreale, P. (2024). ChatGPT translation of program code for image sketch abstraction. *Applied Sciences, 14*(3), 992.

[5] Swathi, P., Prajwal, V., Abhishek, B. M., & Bhat, D. S. (2023). Automated HTML code generation from hand drawn images using machine learning methods. *International Journal of Progressive Research in Engineering Management and Science, 3*(5), 660–664.

[6] Asha, G., Chitralekha, S., Keerthi Basavaraj, N. M., & Latha, A. (2023). Automatic generation of HTML code from hand-drawn images using machine learning techniques. *International Journal of Engineering Research & Technology, 11*(8). (RTCSIT – 2023).

[7] D'Amorim, M., Abreu, R., & Mello, C. (2020). Visual sketching: From image sketches to code. In *[Proceedings Title]* (pp. 101–104).

[8] JR, A. S., Karthik, S., Kumar, A., & Vignesh, R. (2022). A deep learning approach for generating markup code from sketch images. *International Journal for Research in Applied Science & Engineering Technology, 10*(4), 235–238.

[9] Gour, A., Rajguru, S. S., Yadav, V. S., & Sharma, S. (2022). Coding sketch: Turn your sketch designs into code. *EasyChair Preprint*, 9410.

[10] Rashid, F., Darji Devam Prakash, Dalesh, N., Kushagradheer, S., & Geetha, S. (2022). Sketch2Code: HTML code generation from sketch. *International Journal for Research Trends and Innovation, 7*(7), 996–999.

[11] Vishal, M., & Nandhini, K. (2025). Integrated development environment for hand-drawn web sketch recognition and code generation. *International Journal of Science, Engineering and Technology, 13*(2).

[12] Zhu-Tian, C., Xiong, Z., Yao, X., & Glassman, E. (2024). Sketch then generate: Providing incremental user feedback and guiding LLM code generation through language-oriented code sketches. *arXiv*.

[13] Li, J., Li, Y., Li, G., Jin, Z., Hao, Y., & Hu, X. (2023). SKCoder: A sketch-based approach for automatic code generation. *arXiv*.

[14] Jadhav, G., Gaikwad, H., & Gawande, R. (2022). Generation source code from hand draw image – A machine learning approach. In *Proceedings of the 3rd International Conference on Contents, Computing & Communication (ICCCC-2022)*.

[15] Riaz, S., Arshad, A., Band, S. S., & Mosavi, A. (2022). Transforming hand drawn wireframes into front-end code with deep learning. *Computers, Materials & Continua, 73*(1), 1–15.