Draw2Code: An AI-Driven Interactive Whiteboard Platform for Automated Code Generation from Sketches

Mayank Diwakar

Technology | Published: May 01, 2026

ABSTRACT

This paper presents Draw2Code, an innovative AI-driven interactive whiteboard platform designed to enhance the coding process by allowing users to sketch application interfaces and automatically generate corresponding code. The research explores the use of machine learning algorithms and computer vision techniques to interpret user sketches and translate them into functional code snippets. By integrating this technology into a collaborative whiteboard environment, Draw2Code aims to simplify the development workflow, making programming more accessible to non-technical users while boosting productivity for experienced developers. The paper discusses the technical architecture, implementation challenges, real-world applications, user feedback, and future directions for development in this evolving field.

Draw2Code: An AI-Driven Interactive Whiteboard Platform for Automated Code Generation from Sketches

Mayank Diwakar, Ashwani Mishra, Chirag Verma

Department of Artificial Intelligence and Data Science

IIMT College of Engineering

Abstract

The integration of Artificial Intelligence in software engineering has significantly improved development workflows and prototyping processes. This paper presents *Draw2Code*, an AI-driven interactive whiteboard platform for automated code generation from freehand sketches. The system captures canvas drawings, applies computer vision preprocessing, and forwards the data through a multi-model AI pipeline using GPT-4, Google Gemini 2.0, and Meta Llama 3 to generate production-grade React components with Tailwind CSS styling.

Evaluated on more than 500 annotated sketch-to-code pairs, the system achieves an accuracy of 87%, precision of 85%, recall of 83%, and an F1-score of 84%, outperforming baseline systems such as pix2code and Sketch2Code by 11–15 percentage points.

Keywords

Sketch-to-Code, Artificial Intelligence, Large Language Models, Computer Vision, Interactive Whiteboard, UI Code Generation, GPT-4, Gemini, React, Next.js, TLDraw

I. Introduction

The software development industry has undergone transformative changes driven by cloud computing, mobile platforms, and Agile methodologies. Despite these advances, the transformation of design sketches into functional code remains time-consuming and manually intensive.

Traditional workflows require designers to sketch wireframes, digitize them using tools such as Figma or Adobe XD, and manually convert them into executable code using frameworks like React or Angular. This multi-step pipeline introduces inefficiencies and delays.

This paper introduces Draw2Code, an AI-driven system that directly converts freehand sketches into production-ready React code. The primary contributions include:

- A unified sketch-to-code pipeline integrating vision models and large language models
- A multi-model architecture utilizing GPT-4, Gemini 2.0, and Llama 3
- A real-time preview system for rapid feedback
- Improved performance with 87% accuracy on benchmark datasets

II. Related Work

Existing digital whiteboard platforms such as Microsoft Whiteboard, Miro, and TLDraw enable collaborative design but lack automated code generation capabilities.

Previous research includes pix2code, which uses CNN-LSTM architectures for GUI-to-code translation, and Sketch2Code, which applies convolutional models for hand-drawn UI recognition. However, these approaches are limited by restricted output formats and lack support for modern frameworks.

Recent advancements in large language models, including GPT-4, Gemini, and Llama, have enabled multimodal processing, allowing integration of visual inputs with code generation. Draw2Code combines these advancements into a unified system.

III. Methodology

A. System Workflow

The Draw2Code system operates through a structured pipeline:

1. User draws UI on an interactive canvas
2. Image preprocessing (noise removal and edge detection)
3. Vision-based UI component detection
4. Backend processing and data structuring
5. Code generation using large language models
6. Code validation and assembly
7. Deployment and live preview generation

B. Processing Pipeline

The system captures canvas data in image and JSON formats, applies preprocessing techniques such as Gaussian filtering and adaptive thresholding, and detects UI components. The detected structure is then passed to a code generation model that produces React and Tailwind CSS code, which is validated and deployed for preview.

IV. Dataset Description

The dataset used in this study consists of more than 500 annotated sketch-to-code pairs collected from multiple sources, including public datasets and custom-designed sketches.

CategorySamplesProportionComplexityLanding Pages9519%HighDashboards8016%HighE-commerce7515%HighForms7014%MediumBlogs6012%MediumNavigation5511%Low-MediumCard Layouts459%LowModals204%Low

All images were standardized and augmented to improve model performance.

V. Algorithm

The system follows a multi-stage algorithm:

1. Capture canvas input
2. Apply preprocessing (Gaussian blur, thresholding)
3. Extract contours and detect components
4. Generate component hierarchy
5. Select appropriate language model
6. Generate code with controlled parameters
7. Validate output using ESLint
8. Deploy to cloud environment

The overall system maintains low preprocessing complexity while optimizing model inference.

VI. Implementation

The system is implemented using modern web and AI technologies:

LayerTechnologyFrontendNext.js, React, TypeScriptCanvasTLDrawBackendNode.js, ExpressDatabasePostgreSQL, Prisma ORMAI ModelsGPT-4, Gemini 2.0, Llama 3DeploymentVercelQueue SystemBull.js with RedisVII. Results and Analysis

A. Performance Metrics

MetricDraw2Codepix2codeSketch2CodeAccuracy87%72%76%Precision85%69%74%Recall83%71%72%F1 Score84%70%73%

B. Observations

The system demonstrates superior performance compared to baseline models. Training and validation loss trends indicate strong convergence without overfitting.

VIII. Advantages

- Significant reduction in prototyping time
- Integration of design and development workflows
- Support for multiple AI models
- Real-time preview capabilities
- Accessibility for non-technical users
- Scalable architecture

IX. Limitations

- Reduced performance for abstract sketches
- Limited support for multi-page applications
- Latency in processing
- Lack of backend logic generation
- Dependency on external AI services

X. Future Work

Future improvements include multi-user collaboration, support for additional frameworks, model fine-tuning, voice interaction, and integration with CI/CD pipelines.

XI. Applications

The system can be applied in education, startup development, enterprise product design, hackathons, and non-technical innovation workflows.

XII. Conclusion

Draw2Code presents a novel approach to bridging the gap between design and development by leveraging multimodal AI. The system demonstrates high accuracy and efficiency, validating the feasibility of automated sketch-to-code generation and its potential for widespread adoption.

References

[1] T. Beltramelli, “pix2code: Generating Code from a GUI Screenshot,” ACM SIGCHI, 2018.

[2] A. Jain et al., “Sketch2Code,” arXiv, 2019.

[3] OpenAI, “GPT-4 Technical Report,” 2023.

[4] Google DeepMind, “Gemini Models,” 2023.

[5] Meta AI, “Llama Models,” 2023.

[6] TLDraw Documentation, 2023.

[7] REMAUI, ASE Conference.

[8] RICO Dataset, UIST.

[9] React Documentation.

[10] Next.js Documentation.