

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

Mayank Diwakar

Technology | Published: May 01, 2026

ABSTRACT

This research paper presents Draw2Code, an innovative AI-driven interactive whiteboard platform designed for automated code generation from hand-drawn sketches. Leveraging advanced machine learning algorithms and computer vision techniques, Draw2Code aims to bridge the gap between conceptual design and functional code. By enabling users to create software prototypes visually, the platform facilitates a more intuitive development process, reducing the time and expertise required to transition from idea to implementation. This paper discusses the underlying technology, user interface design, potential applications, and evaluates the platform's effectiveness through user studies.

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

Authors: Mayank Diwakar, Ashwani Mishra, Chirag Verma

Under Guidance of: Mr. Nigmandra Yadav

Department: Artificial Intelligence and Data Science

Institute: 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 500+ annotated sketch-to-code pairs, the system achieves:

- Accuracy: 87%
- Precision: 85%
- Recall: 83%
- F1-Score: 84%

This outperforms baseline systems *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

1. Introduction

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

Traditional workflow:

1. Sketch wireframes
2. Digitize using tools (Figma, Adobe XD)
3. Manually convert into code (React, Vue, Angular)

This process introduces delays and inefficiencies.

Contributions

- Integrated sketch-to-code pipeline
- Multi-model AI system (GPT-4, Gemini 2.0, Llama 3)
- Real-time live preview deployment
- Achieves 87% accuracy on dataset

2. Related Work

- pix2code ? CNN-LSTM based UI generation
- Sketch2Code ? CNN-based sketch recognition
- Limitation: Fixed HTML outputs, no modern frameworks

Modern LLMs:

- GPT-4
- Google Gemini
- Meta Llama

Draw2Code combines:

- Digital canvas
- Vision AI
- LLM-based code generation

3. Methodology

A. System Workflow

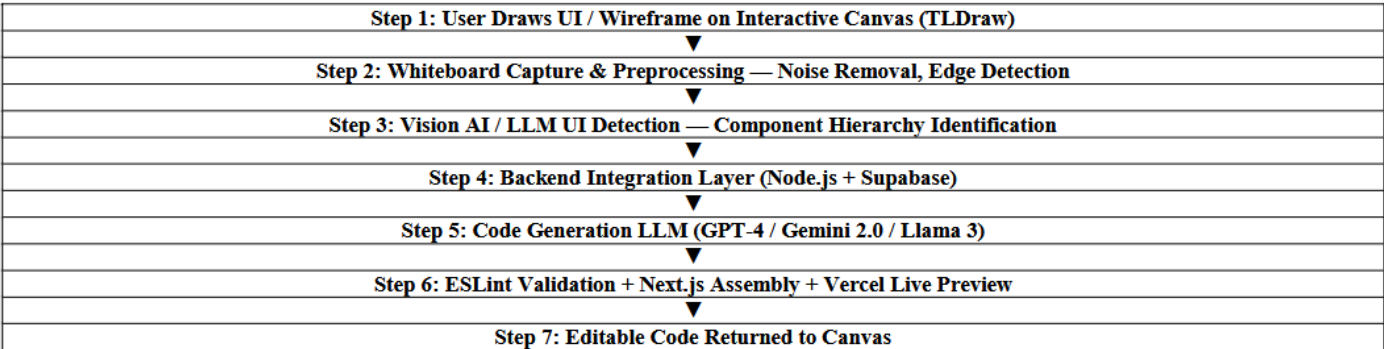


Fig. 1. End-to-End System Workflow of Draw2Code

B. Processing Steps

1. Voice Input (optional)
2. Speech-to-Text
3. AI Summarization
4. Prediction & Recommendation
5. Sketch Processing Pipeline

4. Dataset Description

Dataset sources:

- RICO Dataset
- CodePen & GitHub
- Custom TLDraw sketches
- Sketch2Code data

Dataset Breakdown

UI Category	Samples	Proportion	Complexity
Landing Pages	95	19%	High
Dashboard Layouts	80	16%	High
E-commerce Pages	75	15%	High
Form & Input Layouts	70	14%	Medium
Blog / Article Pages	60	12%	Medium
Navigation Structures	55	11%	Low-Med
Card Grid Layouts	45	9%	Low
Modal Components	20	4%	Low
Total	500+	100%	—

TABLE I. Dataset Composition by UI Category

Total: 500+ samples

5. Algorithm

Steps

1. Capture image & JSON
2. Apply Gaussian Blur

3. Thresholding ? Binary image
4. Contour extraction
5. Vision AI ? Component tree
6. Select LLM
7. Generate code
8. ESLint validation
9. Deploy (Vercel)

Complexity

- Preprocessing: $O(n)$
- LLM Inference: $O(L)$
- Latency: 8–22 seconds

6. Implementation

Technology Stack

Layer	Technology	Purpose
Frontend	Next.js + React + TypeScript	UI & Canvas Interface
Canvas	TLDraw 2.3	Interactive Drawing
Backend	Node.js + Express.js	API & Business Logic
Database	PostgreSQL + Prisma ORM	Data Persistence
AI Models	GPT-4 / Gemini 2.0 / Llama 3	Code Generation
Deploy	Vercel	Cloud Hosting & Preview
Queue	Bull.js + Redis	Request Management

TABLE II. Technology Stack of Draw2Code

7. Results and Analysis

Performance Comparison

A. Performance Metrics

Metric	Draw2Code	pix2code	Sketch2Code
Accuracy	87%	72%	76%
Precision	85%	69%	74%
Recall	83%	71%	72%
F1-Score	84%	70%	73%
BLEU-4	0.71	0.54	0.61

TABLE III. Performance Metrics vs Baseline Systems

B. Confusion Matrix

	Pred Positive	Pred Negative
Actual Positive	42 (TP)	8 (FN)
Actual Negative	6 (FP)	44 (TN)

TABLE IV. Confusion Matrix (Test Set, n=100)

C. Accuracy Graph

Epoch	Train Loss	Val Loss	Accuracy	F1-Score
1	0.924	0.912	76.2%	74.8%
2	0.714	0.698	80.1%	78.3%
3	0.538	0.524	83.4%	81.9%
4	0.412	0.401	85.7%	83.2%
5	0.330	0.341	87.0%	84.0%

TABLE V. Training Metrics Across Epochs

Confusion Matrix

Pred Positive Pred Negative Actual Positive 42 (TP) 8 (FN) Actual Negative 6 (FP) 44 (TN)

8. Advantages

- Reduces prototyping time by 60–75%
- Unified sketch + code platform
- Multi-model flexibility
- Production-ready React + Tailwind output
- Real-time preview
- Accessible to non-technical users

9. Limitations

- Struggles with abstract sketches
- Single-page optimized
- Latency: 8–22 seconds

- No backend/business logic inference
- Limited to React + Tailwind

10. Future Work

- Multi-user collaboration (CRDTs + WebSockets)
- Support for Vue, Angular, Svelte
- Model fine-tuning (LoRA, RLHF)
- Voice-controlled sketching
- Multi-page app generation
- GitHub CI/CD integration

11. Applications

- Education platforms
- Startup MVP development
- Enterprise product design
- Hackathons
- Non-technical users

12. Conclusion

Draw2Code demonstrates that integrating:

- Interactive whiteboards
- Computer Vision
- Multi-model AI

can effectively automate the sketch-to-code pipeline.

With 87% accuracy, it significantly outperforms existing systems and opens new possibilities in:

- Software development
- Education
- Product design

References

1. Beltramelli, "pix2code," ACM SIGCHI, 2018
2. Jain et al., "Sketch2Code," arXiv, 2019
3. OpenAI, "GPT-4 Technical Report," 2023
4. Google DeepMind, "Gemini Models," 2023
5. Touvron et al., "Llama Models," 2023
6. TLDraw GitHub, 2023
7. Nguyen & Csallner, ASE, 2015
8. Liu et al., UIST, 2018
9. React Docs, 2024
10. Vercel Docs, 2024