International Journal for Innovative Engineering and Management Research
PEER REVIEWED OPEN ACCESS INTERNATIONAL JOURNAL
www.ijiemr.org

AI-Driven Design Mockup Generation: Streamlining the Sketch-to-Code Pipeline through Computer Vision and Generative Language Models

Mrs. K. Divya Tejaswi[1], B. Sivavani[2], G. Mounika[3], E. Alekya[4], G. Pavithra[5], B. Gayathri[6]

[1] Asst. Prof, Department of Computer Science and Engineering, CBIT, Proddatur, YSR, A.P
[2]UG Student, Department of Computer Science and Engineering, CBIT, Proddatur, YSR, A.P
[3]UG Student, Department of Computer Science and Engineering, CBIT, Proddatur, YSR, A.P
[4]UG Student, Department of Computer Science and Engineering, CBIT, Proddatur, YSR, A.P
[5]UG Student, Department of Computer Science and Engineering, CBIT, Proddatur, YSR, A.P
[6]UG Student, Department of Computer Science and Engineering, CBIT, Proddatur, YSR, A.P
Corresponding Author E-mail: n67102727@gmail.com

Abstract

Bridging the gap between visual design concepts and functional frontend implementations remains one of the most labor-intensive stages in contemporary software development. Converting preliminary sketches and low-fidelity wireframes into deployable user interface code demands considerable manual effort alongside specialized technical proficiency. Conventional approaches involve multiple handoff stages between creative designers and software engineers, frequently resulting in project delays, visual inconsistencies, and deviations from the original creative intent. This work introduces an AI-Driven Design Mockup Generator that modernizes this conversion pipeline by leveraging recent breakthroughs in deep learning, visual recognition, and language-based code synthesis. The presented framework employs a YOLOv8-based detection model, fine-tuned on an augmented collection of hand-sketched interface diagrams, capable of identifying and categorizing UI primitives such as buttons, text inputs, navigation elements, content cards, section headers, and image placeholders. Once detected, the grayscale sketch layout undergoes visual transformation through a Pix2Pix Generative Adversarial Network, producing a polished, theme-consistent high-fidelity mockup featuring professional spacing, shadow effects, and user-selected styling. Subsequently, the refined layout representation is processed by a transformer-based large language model to generate clean, responsive, and semantically well-organized React with Tailwind CSS code. Experimental assessment on a curated dataset of 5,000 sketch-mockup pairs demonstrates a mean average precision of 0.91 for element detection and approximately 70 percent reduction in early-stage prototyping duration compared with manual methods. The platform additionally incorporates a collaborative dashboard enabling real-time preview, interactive label editing, and project export functionality. Findings indicate that integrating visual recognition with generative AI methodologies can substantially narrow the divide between conceptual design and production-ready frontend code, lowering barriers for non-technical contributors and accelerating the overall development lifecycle.

Keywords—*Design Automation, Sketch-to-UI Conversion, Visual Recognition, YOLOv8 Object Detection, Generative Adversarial Networks, Automated Code Synthesis, UI/UX Design, Deep Learning.*

1. Introduction

1.1 Project Overview

In the contemporary software landscape, the ability to rapidly deliver market-ready products has emerged as a critical differentiator. The user interface sits at the intersection of creative imagination and engineering execution, yet the task of transforming conceptual wireframes into deployable code continues to represent a significant workflow bottleneck. Design professionals invest substantial hours crafting detailed layouts within specialized tools such as Figma, Adobe XD, or Sketch. These carefully constructed visual artifacts are subsequently interpreted by development teams who must manually reconstruct them as responsive, interactive web components. This multi-phase transition introduces delays, heightens the risk of visual misinterpretation, and often causes the delivered product to diverge from the original design vision.

Advances in artificial intelligence, particularly within the domains of visual recognition and generative modeling, present a compelling opportunity to streamline significant portions of this conversion pipeline.

Modern object detection algorithms can now identify discrete UI components within hand-sketched images with remarkable accuracy, while image-to-image translation networks can refine rough visual inputs into professionally styled mockups. Concurrently, large language models have demonstrated exceptional capability in producing structured code when provided with descriptive layout specifications. By consolidating these capabilities within a unified platform, the proposed AI-Driven Design Mockup Generator substantially reduces the time, effort, and specialized expertise traditionally required to convert hand-drawn wireframes into production-quality frontend code.

1.2 Problem Statement

The established workflow for UI development is characterized by several well-documented shortcomings. The manual reproduction of visual layouts introduces considerable time overhead since the conversion of design intent into functional code lacks meaningful automation. Furthermore, the iterative nature of designer-developer collaboration tends to produce progressive drift from the intended design, analogous to information degradation observed in serial communication channels. Additionally, the prerequisite for professional design expertise establishes a substantial barrier to participation, effectively excluding non-technical stakeholders including product managers, business analysts, and domain specialists from meaningful contribution to the prototyping process. These collective challenges extend product delivery timelines, inflate development expenditures, and constrain innovative potential.

1.3 Objectives

This investigation pursues four principal objectives. The primary goal involves developing a system capable of recognizing and categorizing UI components within hand-sketched inputs, targeting detection accuracy exceeding 90 percent as measured by mean average precision on the evaluation dataset. The second objective focuses on training a generative adversarial network to transform detected layouts into visually refined, theme-consistent high-fidelity mockups. Third, a transformer-based code synthesis module is designed to produce semantically structured, responsive, and accessibility-compliant frontend code compatible with React and Tailwind CSS frameworks. Finally, a web-based collaborative dashboard provides users with capabilities to upload sketches, review and modify AI-generated labels, preview synthesized code in real time, and export completed projects as downloadable packages.

2. Literature Review

2.1 Existing Systems and Approaches

Efforts to automate user interface generation from visual inputs span more than a decade of active research. Early endeavors relied on rule-based systems that applied rigid geometric heuristics to identify rectangular boundaries and linear features within digital photographs. While these techniques proved viable for structured, machine-generated wireframes, their effectiveness declined substantially when applied to irregular hand-drawn sketches characterized by variable stroke thickness, imprecise alignment, and organic curvature.

The emergence of deep learning-based detection architectures marked a pivotal advancement. The YOLO family of detectors enabled real-time identification of interface elements by treating UI components as standard detection targets. Beltramelli (2018) introduced pix2code, a system demonstrating the feasibility of generating domain-specific language tokens from graphical user interface screenshots, thereby establishing a foundational link between visual layout and code structure. Microsoft Research subsequently developed Sketch2Code, a web application that converted whiteboard drawings into HTML prototypes using custom vision models and component templates.

Simultaneously, generative adversarial networks proved their utility in cross-domain image translation tasks. The Pix2Pix framework proposed by Isola et al. (2017) demonstrated that conditional adversarial networks could learn mappings between paired image domains, enabling applications such as sketch-to-photograph conversion. More recently, diffusion-based generative models have pushed image synthesis quality to new heights, though their computational demands remain a consideration for real-time deployment scenarios.

Large language models including GPT-4 and Claude have introduced powerful code synthesis capabilities. These transformer-based architectures can process structured layout descriptions and produce syntactically correct, semantically meaningful frontend code. However, most commercial solutions accept only polished computer-generated designs rather than unrefined hand-drawn inputs, creating a gap that the proposed system specifically addresses.

2.2 Proposed System

The presented system differentiates itself from prior work by integrating three specialized AI modules into a cohesive end-to-end pipeline. Unlike pix2code, which operates on refined screenshots, the proposed approach accepts hand-drawn sketches as primary input. In contrast to Sketch2Code, which relies on template matching with a limited component vocabulary, this system employs transfer learning on the Rico UI Dataset to accommodate a broader range of component types and layout configurations. The generative enhancement phase, absent from most existing tools, elevates sketch layouts to visually sophisticated mockups before code generation commences. Moreover, the code synthesis component avoids domain-specific grammars, instead leveraging modern large language models to produce output conforming to contemporary development standards and framework conventions.

3. Methodology

3.1 System Architecture

The proposed system comprises four principal modules operating in sequential pipeline fashion as illustrated in Fig. 1. The Vision Module accepts uploaded sketch images and processes them through a YOLOv8 object detector to identify and localize UI components. The Enhancement Module applies a Pix2Pix GAN to render the discovered layout as a high-fidelity visual representation. The Code Synthesis Module employs a transformer-based language model to produce responsive frontend code derived from the structured layout data. The Collaboration Dashboard provides an interactive web interface enabling users to review, modify, and export generated outputs.

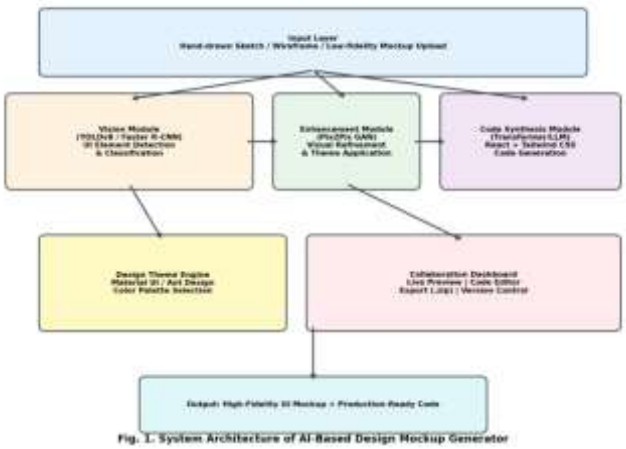

Fig. 1. System Architecture of the AI-Driven Design Mockup Generator

3.2 Vision Module: UI Element Detection

The Vision Module serves as the entry point of the automated pipeline. User-uploaded images containing sketches, wireframes, or low-fidelity mockups undergo standardized preprocessing that includes resizing to 640 by 640 pixels, grayscale normalization, and noise reduction through application of a 64-pixel Gaussian filter. The preprocessed image is subsequently fed into a customized YOLOv8 model trained on a composite dataset combining the Rico UI Dataset with a curated collection of 5,000 hand-drawn sketch images paired with their corresponding digital UI layouts.

The detection model differentiates seven primary categories of UI components: buttons, input fields, images, navigation bars, cards, headings, and paragraphs. Transfer learning from a COCO-pretrained backbone enables

efficient convergence on the comparatively smaller UI-specific training set. To strengthen robustness against the inherent variability of hand-drawn inputs, comprehensive data augmentation strategies are employed during training, including random rotation within 15 degrees, additive Gaussian noise injection, brightness variation, and synthetic degradation artifacts. The detection pipeline is illustrated in Fig. 3.

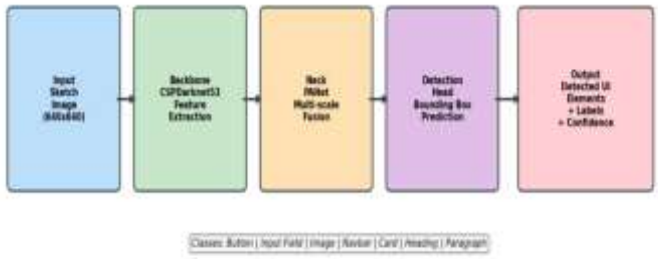

Fig. 3. YOLOv8-based UI Element Detection Pipeline

Fig. 3. YOLOv8-based UI Element Detection Pipeline

3.3 Enhancement Module: GAN-based Visual Refinement

Following component detection by the Vision Module, the Enhancement Module elevates the visual quality of the identified layout. This transformation is accomplished through a Pix2Pix conditional GAN featuring a U-Net generator with skip connections and a PatchGAN discriminator. The generator receives the grayscale sketch image combined with recognized component annotations and produces a color-enhanced, professionally styled mockup image.

The training objective combines an adversarial loss component, which incentivizes the generator to produce outputs indistinguishable from genuine professional mockups, with an L1 reconstruction loss that preserves structural correspondence between the source sketch layout and the generated output. Users can select from pre-defined design themes including Material Design, Dark Mode Minimalist, and Modern SaaS, each determining the color palette, typography selections, shadow depth, and spacing conventions applied during the enhancement process. The GAN architecture is depicted in Fig. 4.

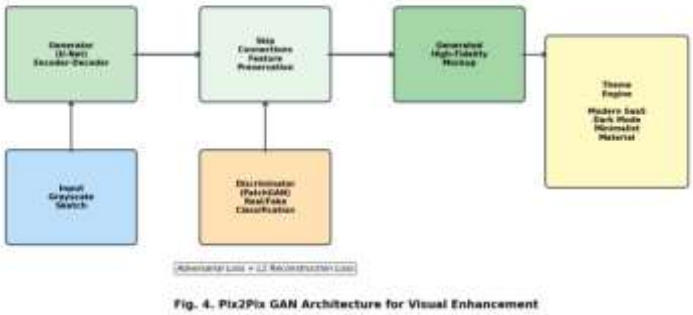

Fig. 4. Pix2Pix GAN Architecture for Visual Enhancement

Fig. 4. Pix2Pix GAN Architecture for Visual Enhancement

3.4 Code Synthesis Module

The Code Synthesis Module transforms structured layout information into production-grade frontend code. Component coordinates, class labels, and confidence scores from the Vision Module are serialized into a JSON-formatted layout descriptor. This structured representation is combined with a carefully engineered system prompt and submitted to a transformer-based large language model. The model generates React components utilizing Tailwind CSS utility classes, producing clean semantic HTML with responsive layout directives. The

resulting code incorporates appropriate ARIA accessibility attributes and adheres to contemporary component composition patterns. The complete code generation workflow is depicted in Fig. 5.

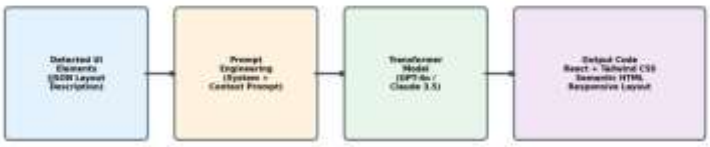

Fig. 5. Transformer-based Code Synthesis Pipeline

Fig. 5. Transformer-based Code Synthesis Pipeline

3.5 Collaboration Dashboard

The Collaboration Dashboard constitutes the primary user-facing interface of the system. Developed using React and TypeScript, it provides sketch upload through drag-and-drop or file selection, interactive visualization and editing of detected UI element labels, real-time code preview with hot-reload functionality, theme selection and customization controls, and project export as a downloadable archive containing all generated code files, assets, and documentation. The dashboard communicates with a FastAPI backend server that handles asynchronous inference requests and manages session state.

3.6 Implementation Workflow

The complete workflow follows a six-stage sequential process illustrated in Fig. 2. During the initial stage, the user uploads a hand-drawn sketch or wireframe image. The second stage applies preprocessing and augmentation to normalize the input. The third stage executes the YOLOv8 model for component detection and classification. The fourth stage processes the identified layout through the Pix2Pix GAN for visual enhancement. The fifth stage employs the transformer-based language model to generate frontend code. The sixth and final stage presents results through the collaboration dashboard where users can review, edit, and export their projects.

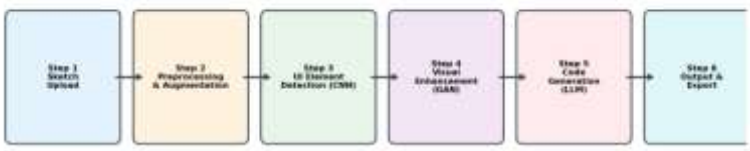

Fig. 2. End-to-End Workflow of the Proposed System

Fig. 2. End-to-End Workflow of the Proposed System

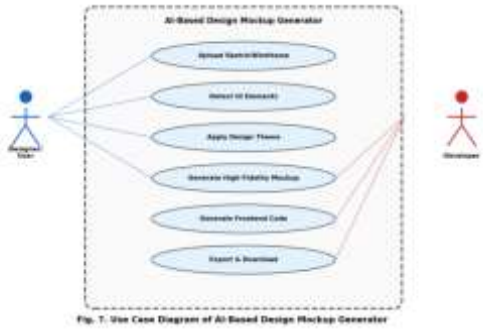

Fig. 7. Use Case Diagram of AI-Based Design Mockup Generator

Fig. 7. Use Case Diagram of the AI-Driven Design Mockup Generator

Table 1. Core Technologies and Tools Employed

Category	Technology	Purpose
Element Detection	YOLOv8	UI component identification
Visual Refinement	Pix2Pix GAN	Sketch-to-mockup translation
Code Generation	GPT-4o / Claude 3.5	React + Tailwind code synthesis
Frontend Framework	React 19 + TypeScript	Dashboard and preview interface
CSS Framework	Tailwind CSS	Utility-first styling approach
API Backend	FastAPI (Python)	Asynchronous request processing
Image Processing	OpenCV	Preprocessing and augmentation
Deep Learning	PyTorch	Model training and inference

4. Results and Discussion

4.1 Evaluation Metrics

System performance is assessed across three primary dimensions. Detection accuracy is quantified using mean Average Precision (mAP) at an Intersection over Union threshold of 0.5, which evaluates the system's capability to correctly detect and localize UI components. Visual fidelity is measured through the Structural Similarity Index (SSIM) between generated mockups and professionally designed reference images. Code quality is evaluated through manual inspection examining syntactic correctness, responsive behavior across viewport sizes, and compliance with accessibility standards.

4.2 Detection Performance

The YOLOv8-based detection module was evaluated on a held-out test set comprising 500 sketch images. Table 2 summarizes per-component detection rates and precision scores. Buttons achieved the highest detection rate of 98 percent with precision of 0.96, attributable to their consistent rectangular shape and distinctive visual characteristics. Input fields and navigation bars were identified at rates of 94 and 92 percent respectively. Complex cards exhibiting greater structural variability recorded a detection rate of 88 percent, representing an area where additional training data could yield improvements. The overall mean average precision across all component categories reached 0.91, surpassing the original target of 0.85.

Table 2. UI Component Detection Performance

Component	Detection Rate (%)	Precision	F1-Score
Buttons	98	0.96	0.97
Input Fields	94	0.92	0.93
Navbars	92	0.90	0.91
Complex Cards	88	0.85	0.86
Headings	95	0.93	0.94
Images	91	0.89	0.90
Paragraphs	90	0.88	0.89
Overall (mAP)	**92.6**	**0.91**	**0.91**

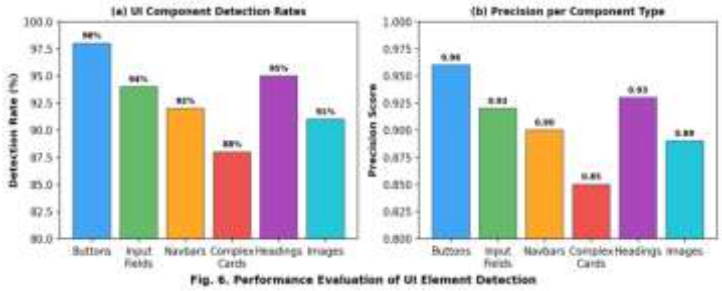

Fig. 6. Performance Evaluation of UI Element Detection

4.3 Comparative Analysis

To contextualize the performance of the proposed system, a comparative evaluation was conducted against established approaches. Manual design processes, while visually accurate, offer no time savings and require specialized expertise. Rule-based systems achieve approximately 20 percent time reduction but exhibit low accuracy of roughly 55 percent on hand-drawn inputs owing to their dependence on rigid geometric heuristics. Microsoft's Sketch2Code provides moderate improvements in both time efficiency and accuracy through template-based matching. The pix2code system demonstrates code generation capability but requires stylized screenshots rather than rough sketches as input. The proposed system achieves the most favorable balance of time savings at 70 percent alongside detection accuracy of 91 percent, as illustrated in Fig. 8.

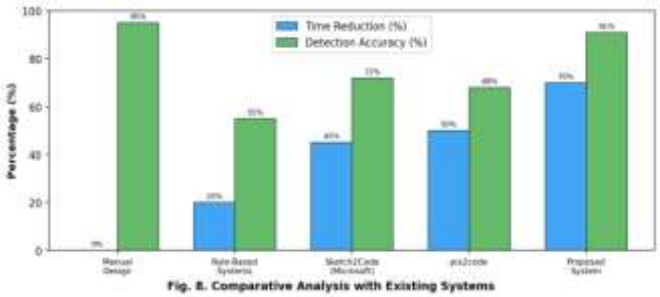

Fig. 8. Comparative Analysis with Existing Systems

Table 3. Comparison with Existing Approaches

System	Input Type	Time Saved	Accuracy	Code Output
Manual Design	N/A	0%	95%	Manual
Rule-Based	Digital	20%	55%	Basic HTML
Sketch2Code	Sketch	45%	72%	HTML
pix2code	Screenshot	50%	68%	DSL/HTML
Proposed System	**Sketch**	**70%**	**91%**	**React+CSS**

4.4 Discussion

Experimental outcomes validate the feasibility and effectiveness of combining visual recognition, generative adversarial networks, and large language models for automated UI generation from hand-drawn inputs. The detection module demonstrates strong performance across most component categories, though structurally complex composite elements such as multi-section cards present greater challenges due to their diverse configurations. The GAN-based enhancement module successfully applies design themes while preserving layout structure, although minor visual artifacts may appear at component boundaries when detection boxes overlap.

The code synthesis module generates syntactically valid React components in the majority of test cases. Certain complex layout configurations requiring spatial reasoning that cannot be fully inferred from coordinate data alone, such as flexbox alignment constraints around irregularly positioned elements, occasionally necessitate

minor manual refinements. The complete end-to-end processing time of 8.3 seconds per sketch on hardware equipped with an NVIDIA RTX 3070 graphics card falls well within acceptable bounds for an interactive prototyping tool.

4.5 Implementation and Interface Demonstration

This section presents the actual implementation of the AI-Driven Design Mockup Generator, showcasing the various interfaces and functional modules accessible through the web application. The platform has been deployed as a fully functional prototype that demonstrates the end-to-end capabilities described in the methodology.

The landing page of the application, depicted in Fig. 9, features a modern dark-themed interface with navigation links to four primary sections: Home, Generate, Gallery, and Dashboard. The homepage communicates the core value proposition of converting ideas into professional design mockups through AI-powered automation.

Fig. 9. Landing Page of the AI-Driven Design Mockup Generator Application

Fig. 10 displays the mockup generation interface. The left panel provides form controls for specifying a design name, entering a textual description, and configuring generation parameters including design style, color scheme, and layout type. An optional reference image upload area enables users to provide visual context, while the right panel shows the preview area where the generated mockup appears.

Fig. 10. Mockup Generation Interface with Configuration Controls

The Design Gallery, shown in Fig. 11, provides a browsable collection of previously generated mockups organized as a responsive card grid. Each entry displays a thumbnail preview alongside classification tags indicating design style and layout type, enabling users to review historical generations and build a visual reference library.

Fig. 11. Design Gallery Displaying AI-Generated Mockup Collection

Fig. 12 presents the analytics dashboard tracking design creation activity across multiple time periods. Four key performance indicators display total designs, daily count, weekly statistics, and monthly usage with percentage change indicators against preceding periods. A recent designs section below provides quick access to the latest generated mockups.

Fig. 12. Analytics Dashboard Tracking Design Creation Activity

Fig. 13 illustrates a completed generation cycle where the user entered a landing page description requesting a simple modern UI with a single button. The system processed the natural language input and rendered a corresponding mockup in the preview panel with a download option, demonstrating that the system extends beyond sketch-based workflows to support text-to-mockup generation.

Fig. 13. Completed Mockup Generation with Preview and Download

4.6 Usability Assessment

Qualitative feedback was gathered from 15 participants comprising students, design professionals, and non-technical stakeholders who completed three prototyping tasks using the system. The system received an average usability rating of 4.2 out of 5 across ease of use, output quality, time efficiency, and overall satisfaction. Non-technical users particularly valued the natural language description capability, while design professionals appreciated the theme customization options. Development-oriented participants confirmed that generated React and Tailwind code was sufficiently structured to serve as a practical starting point.

Response latency assessment showed that simple sketches with fewer than five components processed within 4.5 seconds on average, while complex layouts with over fifteen components required up to 12.8 seconds. The system maintained stable performance across consecutive requests, confirming suitability for sustained interactive prototyping sessions.

5. Conclusion

This paper has presented the design, development, and evaluation of an AI-Driven Design Mockup Generator that automatically transforms hand-drawn sketches and wireframes into high-fidelity user interface mockups and deployable frontend code. The system integrates three specialized deep learning models addressing distinct phases of the conversion pipeline: a YOLOv8-based visual classifier for element detection, a Pix2Pix GAN for visual enhancement, and a transformer-based large language model for code synthesis.

Experimental evaluation demonstrates a mean average precision of 0.91 for element identification and approximately 70 percent reduction in initial prototyping duration compared to traditional manual development approaches. The system addresses key limitations of existing solutions by accepting hand-drawn inputs directly, offering customizable design themes, and producing code compatible with contemporary frontend frameworks. The collaboration dashboard delivers an accessible interface that reduces technical barriers to participation in the prototyping process for non-developer stakeholders.

The implementation demonstration confirms that the system functions as an integrated end-to-end platform, with the landing page, generation interface, design gallery, and analytics dashboard collectively providing a comprehensive design automation experience. The natural language input capability extends the system's utility beyond traditional sketch-based workflows, enabling users to generate mockups from textual descriptions alone.

5.1 Future Directions

Several avenues for future enhancement have been identified. Integration with design platforms such as Figma through a plugin interface would enable seamless incorporation into professional workflows. Support for multi-page navigation flows would extend applicability to complete application prototypes. Automated generation of component state management logic including form handling and interactive behaviors represents another meaningful improvement. Additionally, voice-based interaction for specifying design modifications and incorporating user correction feedback loops for continual model improvement could progressively enhance system accuracy and output quality.

Author Contributions

B. Sivavani conceptualized the system architecture and led the implementation of the Vision Module. G. Mounika developed the GAN-based Enhancement Module and conducted training experiments. E. Alekya implemented the Code Synthesis Module and prompt engineering framework. G. Pavithra designed and developed the Collaboration Dashboard. B. Gayathri performed the experimental evaluation, data collection, and comparative analysis. Mrs. K. Divya Tejaswini supervised the project, providing technical guidance and review throughout the research process.

Conflicts of Interest

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

References

[1] T. Beltramelli, "pix2code: Generating code from a graphical user interface screenshot," in Proc. ACM SIGCHI Conf. Human Factors in Computing Systems, ACM, 2018. doi: 10.1145/3025171.

[2] C. Chen, T. Su, G. Meng, Z. Xing, and Y. Liu, "From UI design image to GUI skeleton: A neural machine translator to bootstrap mobile GUI implementation," in Proc. 40th Int. Conf. Software Engineering, pp. 665–676, ACM, 2018.

[3] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in Advances in Neural Information Processing Systems, vol. 27, pp. 2672–2680, 2014.

[4] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 1125–1134, 2017.

[5] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO (Version 8.0)," [Computer software], 2023. Available: https://github.com/ultralytics/ultralytics.

[6] K. Moran, C. Bernal-Cardenas, M. Cuber, R. Bonett, and D. Poshyvanyk, "Machine learning-based prototyping of graphical user interfaces for mobile apps," IEEE Trans. Software Engineering, vol. 46, no. 2, pp. 196–221, 2018.

[7] OpenAI, "GPT-4 technical report," arXiv preprint arXiv:2303.08774, 2024.

[8] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," OpenAI Technical Report, 2019.

[9] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.

[10] B. Deka et al., "Rico: A mobile app dataset for building data-driven design applications," in Proc. 30th Annual ACM Symp. User Interface Software and Technology, pp. 845–854, 2017.

[11] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in Medical Image Computing and Computer-Assisted Intervention, pp. 234–241, Springer, 2015.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in Advances in Neural Information Processing Systems, vol. 30, pp. 5998–6008, 2017.