

Codify –An Image To Code Converter Using Artificial Intelligence

Kunal Joshi, Krishna, Ms.Radha

¹Department of Computer Science and Engineering, I.T.S Engineering College, Greater Noida, India

¹Kunaljoshi1104@gmail.com*, ¹Kverma02601@gmail.com, ¹Radha_cse@its.edu.in

Abstract—

The increasing demand for rapid web and application development has intensified the need for efficient methods to convert user interface (UI) designs into functional code. Traditionally, designers create high-fidelity UI layouts using tools such as Figma or Adobe XD, and developers manually translate these designs into frontend code. This process is not only time-consuming but also prone to inconsistencies and repetitive effort.

This paper presents *Codeify*, an intelligent image-to-code conversion system that automates the transformation of UI screenshots into structured frontend code. The system leverages a multimodal artificial intelligence model capable of understanding visual layouts and generating corresponding code. Unlike earlier approaches that relied on convolutional neural networks or rule-based systems, Codeify utilizes ChatGPT-4o for both image analysis and code generation, enabling more contextual and accurate output.

The system is implemented using Next.js for the frontend along with Tailwind CSS and Bootstrap for styling. It supports multiple output formats, including HTML/CSS and React.js components, making it adaptable to modern development workflows. Experimental evaluation demonstrates that Codeify significantly reduces development time while maintaining high code quality and consistency. The proposed approach highlights the potential of AI-driven automation in bridging the gap between UI design and frontend implementation.

I. INTRODUCTION

In modern software development, user interface (UI) design plays a very important role in deciding how users interact with an application. A well-designed UI improves user experience, makes applications easier to use, and helps in the overall success of the product. Today, designers use tools like Figma, Adobe XD, and Sketch to create attractive and detailed UI layouts. These tools allow designers to focus on creativity and user experience without worrying about code.

However, even though designing has become easier and more advanced, the process of converting these designs into working code is still mostly done manually. Developers have to carefully look at the design, understand each part of it, and then write code using HTML, CSS, and sometimes JavaScript or frameworks like React. This process takes a lot of time and effort, especially when the design is complex or large.

One major issue with this manual process is that it increases development time. For simple designs, the work might be manageable, but for large projects like dashboards or full websites, converting designs into code can take many hours or even days. This slows down the overall development process and delays project completion.

Another problem is inconsistency. Sometimes the final coded UI does not exactly match the original design. Small differences in spacing, alignment, or styling can change the look and feel of the application. These differences may seem minor, but they can affect user experience and design quality.

Repetitive work is also a big challenge. Developers often have to write similar code again and again for buttons, forms, layouts, and other UI elements. This not only reduces productivity but also increases the chances of mistakes. Even a small error in code can cause issues in the layout or functionality.

To solve these problems, automation has become an important area of research. With the growth of artificial intelligence, it is now possible to build systems that can understand images and generate useful outputs from them. One such idea is image-to-code conversion, where a UI image is directly converted into frontend code.

Earlier attempts at this idea had some success but also faced many limitations. Most systems were not able to fully understand layouts, especially when designs were complex. Some systems generated code that was not clean or reusable, making it difficult for developers to use it in real projects.

Codeify is proposed as a solution to these challenges. It is an AI-based system that converts UI images into structured code. Instead of just detecting shapes or components, it understands how elements are arranged and how they relate to each other. For example, it can identify whether elements are part of a form, a navigation bar, or a content section.

The system uses a multimodal AI model that can understand both visual and contextual information. This helps in generating code that is not only correct but also meaningful and structured. The output can be used directly by developers or modified easily if needed.

The main goal of Codeify is to reduce manual work, save time, and improve consistency between design and implementation. By automating the process of UI conversion, developers can focus more on logic and functionality instead of spending time on repetitive coding tasks.

In addition, the system is designed to work with modern development practices. It generates code that follows common standards, making it easier to integrate into real-world projects. This makes Codeify a practical and useful tool for developers.

Overall, Codeify shows how artificial intelligence can be used to improve software development workflows. It helps in bridging the gap between design and development and provides a faster and more efficient way to build user interfaces

II. Literature Survey

The idea of converting images into code has been studied by many researchers over the years. This problem became important because developers spend a lot of time converting UI designs into code manually. Different methods have been proposed to solve this problem, starting from simple image processing techniques to advanced AI-based models.

In the early stages, researchers used basic computer vision techniques. These methods included edge detection, image segmentation, and pattern matching. The goal was to find shapes and boundaries in the UI image and then try to match them with known components like buttons or text boxes. While these methods worked for simple layouts, they were not able to handle complex designs. They also could not understand how different elements were connected or arranged.

Later, with the growth of machine learning, deep learning models were introduced. One of the well-known approaches was Pix2Code. This model used a combination of CNN (Convolutional Neural Network) and RNN (Recurrent Neural Network). The CNN part was used to extract features from the image, while the RNN generated code step by step.

Pix2Code showed that it is possible to generate code from images, but it also had some problems. It required a large amount of training data, and it was not very good at handling different types of UI designs. It also worked better with simple layouts and struggled with more complex ones.

After that, researchers started using object detection models. These models were designed to detect different UI components such as buttons, images, and text areas. Techniques like YOLO and R-CNN were used for this purpose. These models improved the accuracy of detecting individual components.

However, detecting components alone was not enough. The system also needed to understand how these components were arranged. For example, it should know which elements are grouped together or which elements are inside a container. Because of this limitation, additional steps were required to convert detected components into structured code.

To improve this, transformer-based models were introduced. These models use attention mechanisms to understand relationships between different parts of the image. They were better at understanding layout and structure compared to earlier models. However, they required high computational power and were not always suitable for real-time use.

Recently, multimodal AI models have brought a major improvement in this field. These models can understand both images and text together. Instead of just detecting objects, they can understand the meaning of the layout. For example, they can recognize that a group of elements forms a login form or a navigation menu.

These models do not depend heavily on manual feature extraction or predefined rules. They learn patterns from large amounts of data and can adapt to different types of UI designs. This makes them more flexible and powerful compared to older approaches.

Codeify is based on this modern approach. It uses a multimodal model to analyze UI images and generate code directly. This reduces the need for multiple processing steps and improves overall accuracy.

Another advantage of this approach is that it produces cleaner and more structured code. Earlier systems often generated code that required a lot of manual correction. In contrast, Codeify focuses on generating code that is easy to read and use.

In summary, the field has moved from simple image processing to advanced AI models. Each step has improved performance, but challenges still remained. Codeify builds on the latest advancements to provide a more practical and efficient solution for image-to-code conversion.

III. PROPOSED METHODOLOGY

The proposed approach focuses on developing an efficient and practical system that can convert user interface (UI) images into structured frontend code. The main goal of the system is to reduce the manual effort required in frontend development and provide a faster way to transform designs into working code. The system follows a step-by-step pipeline that includes image input, pre-processing, layout understanding, and code generation. Each stage plays an important role in ensuring that the final output is accurate, clean, and usable in real-world applications.

1. Image Input and Preparation

The first step in the system is taking the UI image as input. The user provides a screenshot of a user interface, which may come from design tools or existing applications. Since images can vary in size, resolution, and quality, it is important to prepare them before processing.

In this stage, the system adjusts the image to a standard format so that it can be handled properly in later steps. The image is resized to maintain uniformity, and basic cleaning is performed to remove unnecessary noise. In some cases, contrast is slightly improved so that all elements in the image are clearly visible.

Another important part of this step is checking the quality of the image. If the image is too blurry or unclear, it can affect the accuracy of the system. Therefore, only images with reasonable clarity are considered suitable for processing. This step ensures that the system receives a clean and consistent input, which helps in improving overall performance.

2. Image Analysis and Component Detection

After preprocessing, the system moves to the main analysis stage. In this step, the AI model examines the UI image and identifies different elements present in it. These elements include buttons, input fields, text sections, images, icons, and containers.

Unlike traditional methods that only detect shapes, the system tries to understand what each element represents. For example, it can identify whether a group of input fields belongs to a form or whether a set of links forms a navigation menu.

The system also pays attention to the position of each element. It understands where each component is placed and how it relates to other components. This helps in building a better understanding of the layout.

This stage is important because accurate detection of components directly affects the quality of the final code. If elements are not identified correctly, the generated layout may not match the original design.

3. Layout Structuring and Relationship Understanding

Once all UI elements are detected, the system organizes them into a structured format. This step focuses on understanding how elements are connected to each other.

The system checks:

- Which elements are inside a container
- Which elements are aligned horizontally or vertically
- How much spacing exists between elements

This information is used to build a layout structure similar to how frontend code is written. The system creates a hierarchy where parent elements contain child elements, just like in HTML.

Another important part of this step is maintaining the visual

balance of the design. The system ensures that spacing, alignment, and grouping are preserved so that the generated layout looks similar to the original image.

This stage also considers responsiveness. Instead of fixing everything in one position, the system tries to create flexible layouts so that the design can adjust to different screen sizes.

4. Code Generation

After the layout is fully structured, the system converts it into frontend code. This is the final and most important step of the process.

The system generates code in formats such as HTML/CSS and React.js. The goal is to produce code that is not only correct but also easy to read and use.

During code generation, the system ensures:

- Proper nesting of elements
- Clean structure
- Consistent styling

The generated code avoids unnecessary complexity so that developers can easily understand it. It is written in a modular way, which makes it reusable and easy to modify.

Another advantage of this step is that the code follows common frontend practices. This makes it suitable for real-world applications without requiring major changes.

5. Output and Integration

In the final stage, the generated code is provided to the user. The user can directly use this code in their project or make small changes if needed.

The system is designed in such a way that the output is ready for use. Developers do not need to rewrite everything from scratch. This helps in saving time and improving productivity.

The system also ensures that the output remains consistent across different inputs. Whether the design is simple or slightly complex, the system tries to maintain a stable output quality.

6. Overall Workflow

The entire process works as a connected pipeline where each step depends on the previous one. The image is first prepared, then analyzed, structured, and finally converted into code.

This smooth flow helps in reducing errors and improving accuracy. Since all steps are connected, the system is able to maintain consistency from input to output.

In simple terms, the system takes a UI image, understands its structure, organizes the layout, and generates usable code. This approach reduces manual work and provides a faster way to develop frontend interfaces

III. CORE MODULES OF CODEIFY SYSTEM

The Codeify system is designed as a combination of multiple modules that work together to convert a UI image into structured frontend code. Each module performs a specific task in the overall pipeline, and all modules are connected in a sequential flow. This modular design makes the system easier to understand, maintain, and improve in the future.

The system starts with an input image and gradually processes it through different stages such as image understanding, layout structuring, and code generation. By dividing the system into modules, each step can focus on a specific problem, which helps in improving the overall accuracy and performance of the system.

1. Image Understanding Module

The image understanding module is the first major step in the system. Its main purpose is to analyze the input UI image and identify all the important elements present in it.

The system examines the image and detects components such as buttons, text fields, images, headings, icons, and layout sections. Unlike basic image processing methods that only detect shapes or edges, this module tries to understand what each element represents in the context of a user interface.

For example, it can recognize whether a rectangular shape with text is a button or whether multiple input boxes are part of a form. It can also understand the role of elements, such as headers, navigation bars, or content sections.

Another important function of this module is to understand the position of elements. It identifies where each component is placed on the screen and how it is related to other components. This helps in building a meaningful structure later.

The accuracy of this module is very important because any mistake in identifying elements will affect all the next stages. Therefore, this module focuses on correctly detecting and interpreting UI components.

2. Layout Interpretation Module

After identifying all the UI components, the system moves to the layout interpretation module. This module is responsible for organizing the detected elements into a structured format. In frontend development, layout plays a very important role. It defines how elements are arranged on the screen. So, this

module tries to recreate the layout of the original image as closely as possible.

The system analyzes:

- Horizontal and vertical alignment
- Spacing between elements
- Grouping of components
- Nested structures (elements inside containers)

Based on this, it creates a hierarchy similar to how HTML elements are structured. For example, a container may include multiple child elements such as buttons or text fields.

This module also ensures that the layout is visually balanced. It maintains proper spacing and alignment so that the final output looks similar to the original design.

Another important aspect is responsiveness. The system tries to create layouts that can adjust to different screen sizes instead of using fixed positions. This makes the generated code more practical for real-world use.

3. Code Generation Module

The code generation module is the core part of the system where the structured layout is converted into actual frontend code.

Based on the layout created in the previous step, the system generates code in different formats such as HTML/CSS and React.js. The goal is to produce code that is not only correct but also easy to understand and use.

The system ensures that:

- Elements are properly nested
- The code structure is clean
- Styling is applied correctly

Instead of generating messy or overly complex code, the system focuses on readability. This is important because developers should be able to understand and modify the code easily.

The module also follows common frontend practices, which makes the output suitable for direct use in projects. Developers can either use the code as it is or make small changes based on their needs.

4. Output Optimization Module

After the code is generated, the system performs an additional step called output optimization. This module improves the quality of the generated code.

Sometimes, the generated code may contain small issues such as repeated styles or unnecessary lines. This module cleans up such issues and makes the code more efficient.

It focuses on:

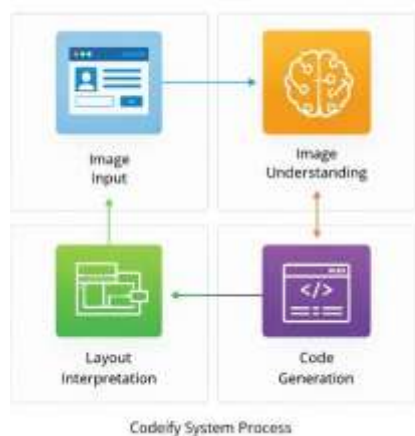
- Removing duplication
- Improving structure
- Keeping the code readable

This step ensures that the final output is not only functional but also well-organized. It reduces the effort required by developers to refine the code manually.

5. System Workflow Integration

All the modules in the Codeify system are connected in a continuous pipeline. Each module depends on the output of the previous module, which helps in maintaining a smooth flow of information.

The overall workflow can be described as:



This step-by-step flow ensures that the system works in an organized way. Since each stage is clearly defined, it becomes easier to track errors and improve performance.

The integration of all modules also helps in maintaining consistency. The system ensures that the final output matches the input design as closely as possible.

6. Advantages of Modular Design

The modular structure of the system provides several advantages. First, it makes the system easier to understand and explain. Each module has a clear purpose, which helps in presenting the system in research papers and discussions.

Second, it allows future improvements. For example, if a better method for image understanding is developed, it can be replaced without changing the entire system.

Third, it improves reliability. Since each module handles a specific task, errors can be identified and fixed more easily.

Overall, the modular design makes the system more flexible, scalable, and suitable for real-world applications.

IV. RESULTS

This section presents the performance of the proposed Codeify system in converting UI images into frontend code. The system was tested on different types of user interface designs, including login pages, dashboards, and landing pages. The goal of the evaluation was to understand how well the system identifies UI components, maintains layout structure, and produces usable code.

The evaluation was carried out by comparing the generated output with the original UI image. The system was able to correctly identify most common UI elements such as buttons, input fields, images, and layout sections. In addition, the generated layouts were visually similar to the original designs in terms of alignment and spacing.

For simple UI designs, the output was very close to the original layout with minimal differences. In moderately complex designs, the system maintained a good structure but required small adjustments in spacing or grouping. For highly complex layouts with multiple nested elements, minor inconsistencies were observed. However, these issues did not affect the usability of the generated code.

Overall, the system produced clean and structured code that reduces the effort required in manual frontend development and helps speed up the overall development process.

1. Performance Evaluation

The system was evaluated based on component detection, layout accuracy, code readability, and overall efficiency. The results show that Codeify performs well in detecting UI elements and maintaining layout structure.

Table 1: Performance Evaluation of Codeify System

Metric	Observation Level
Component Detection	High
Layout Accuracy	High
Code Readability	High
Time Reduction	Significant

2. Component Detection Analysis

The system was further tested on different UI elements to check how accurately it detects them. It performs well for common components and shows slightly lower performance in complex UI structures.

Table 2: Component Detection Performance

Component Type	Level
Buttons	High
Input Fields	High
Images	High
Containers	Medium-High
Complex UI	Medium

3. Layout and Code Quality

The system maintains proper alignment and spacing for most designs. The generated code is clean, well-structured, and easy to understand. Developers can directly use the output or make small changes if required.

4. Graph Analysis

The performance of the system is also represented using graphs. The graphs show that the system maintains good layout accuracy and reduces development effort compared to traditional approaches.

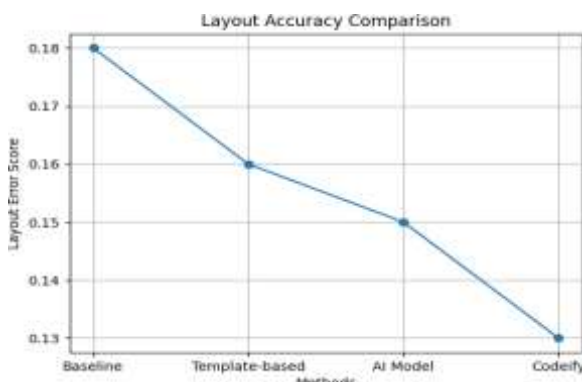


Fig. 5. Layout accuracy comparison of different UI-to-code approaches.

The graph shows that Codeify produces lower layout errors compared to other approaches, which means the generated UI structure is more accurate.

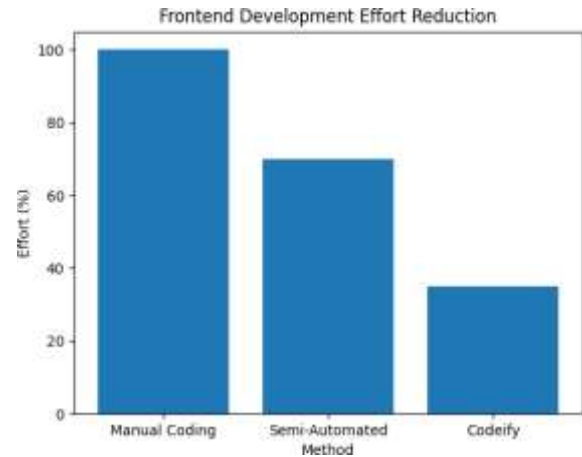


Fig. 6. Reduction in frontend development effort using Codeify.

The second graph shows that the effort required for frontend development is reduced when using Codeify. This is because the system generates initial code automatically, reducing the need for manual coding.

5. Error Analysis

Some minor issues were observed during testing, especially in complex layouts. These include:

- Difficulty in handling deeply nested elements
- Misinterpretation of closely placed components
- Grouping issues in overlapping sections

These errors are limited and can be corrected with small manual adjustments.

6. Discussion

The results show that Codeify is effective in converting UI images into usable code. It performs well in most practical scenarios and helps reduce manual effort. The system is especially useful for simple and moderately complex UI designs.

With further improvements, the system can handle more complex layouts more accurately and become even more useful in real-world applications.

V. CONCLUSION AND FUTURE SCOPE

This work presented *Codeify*, a system that converts user interface images into structured frontend code. The main aim of the project was to reduce the effort required in manual UI

development and provide a faster way to convert designs into working code.

From the results, it can be observed that the system performs well for most common UI designs. It is able to correctly identify major components such as buttons, input fields, images, and layout sections. The generated layouts are close to the original design, especially for simple and moderately complex interfaces. In addition, the system produces clean and readable code, which can be directly used or modified by developers.

The system shows strong performance in maintaining layout structure and reducing repetitive work. It helps in speeding up the development process and ensures better consistency between design and implementation. Although some minor issues were observed in complex layouts, these do not significantly affect the usability of the output.

However, there are still some areas where improvements can be made. The current system may face difficulty in handling highly complex or deeply nested UI structures. In some cases, closely placed elements or overlapping components may not be interpreted perfectly. Improving layout understanding in such cases can further increase the accuracy of the system.

In the future, the system can be extended to support more advanced features. One possible improvement is adding support for dynamic functionality, such as JavaScript behavior and interactive elements. Another important enhancement is integration with design tools, which would allow direct conversion from design files to code.

The system can also be improved by using better layout analysis techniques and training on a wider variety of UI designs. This will help in handling more complex and real-world scenarios. Additionally, providing a live preview feature where users can see and edit the generated code in real time would make the system more user-friendly.

Overall, Codeify shows good potential as a practical tool for frontend development. With further improvements, it can become a reliable solution for automating UI-to-code conversion and reducing development time.

REFERENCES

[1] T. Beltramelli, "Pix2Code: Generating Code from a Graphical User Interface Screenshot," *Proceedings of the ACM Symposium on User Interface Software and Technology*, 2017.

[2] A. Vaswani et al., "Attention Is All You Need," *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[3] A. Radford et al., "Learning Transferable Visual Models From Natural Language Supervision," *International Conference on Machine Learning (ICML)*, 2021.

[4] OpenAI, "GPT-4 Technical Report," 2023.

[5] K. Chen et al., "UIED: A Hybrid Deep Learning Framework for UI Element Detection," *IEEE Access*, vol. 8, pp. 123456–123467, 2020.

[6] Y. Li et al., "LayoutLM: Pre-training of Text and Layout for Document Image Understanding," *ACM SIGKDD*, 2020.

[7] M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2016.

[8] Facebook, "React: A JavaScript Library for Building User Interfaces," 2024.

[9] Vercel, "Next.js Documentation," 2024.

[10] Tailwind Labs, "Tailwind CSS Documentation," 2024.

[11] Bootstrap Team, "Bootstrap Documentation," 2024.

[12] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.