

**A**

**Report on**

**“HTML Code Generation Using Mock-up Images Using ML Techniques”**

Submitted To

Shivaji University, Kolhapur

In Partial Fulfillment of the Requirement for the Degree

Of

Final Year of Engineering

(COMPUTER SCIENCE ENGINEERING)

Submitted by,

**Roll No Name of Student**

**33 Kaushik Walwadkar**

**34 Prathamesh Chavan**

**35 Akshata Kulkarni**

**59 Piyusha Patil**

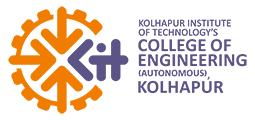
Under the guidance of

Dr. Mamta Kalas

Kolhapur Institute of Technology’s

College of Engineering, Kolhapur

Year2022-23



**CERTIFICATE**

This is to certify that **Mr. Kaushik Walwadkar, Mr. Prathamesh Chavan, Ms. Akshata Kulkarni, Ms. Piyusha Patil** have completed the Project on subject entitled “HTML code generation using mock-up images using ML techniques”, in the fulfilment of the requirement for the award of Final Year (Computer Science and Engineering) of KIT’s College of Engineering, Kolhapur in the academic year 2022-23.

Date:

Place:

**Dr. M. S. Kalas** **External Examiner** **Dr. Ajit S. Patil**

Project Guide Head CSE

**Prof. Dr. MOHAN. B. VANAROTTI**

Director

**ACKNOWLEDGMENT**

We would like to express our deep gratitude towards **Dr. Ajit S. Patil** Head of Department for their constant encouragement and guidance. Their support and cooperation throughout the project work has been of immense help to us.

We express our sincere thanks to all the teaching and non-teaching staff and all those who have directly or indirectly helped in making the project a success.

**Roll No Name of Student**

**33 Kaushik Walwadkar**

**34 Prathamesh Chavan**

**35 Akshata Kulkarni**

**59 Piyusha Patil**

**Contents**

1. **ABSTRACT**
2. **INTRODUCTION**
3. **PROBLEM STATEMENT**
4. **PRESENT THEORY**
5. **METHODOLOGY**
6. **PROGRAM IMPLEMENTATION**
7. **IMPLEMENTATION**
8. **ANALYSIS**
9. **CONCLUSION**
10. **REFERENCES**

**ABSTRACT**

The initial step in website design often involves creating mock-up images of the web pages, either by hand or using mock-up developer tools. These mock-ups serve as a visual representation of the desired layout and design for the web pages. The purpose is to assist developers in translating these mock-ups into actual code.

To streamline this process and make it more efficient, we propose a system that automates the generation of HTML code from mock-up images using machine learning techniques. This system utilizes two main approaches: computer vision and deep systematic analysis.

Computer vision techniques are employed to analyze the mock-up images and identify the different elements and components within them. This includes recognizing buttons, forms, menus, and other interface elements.

Deep systematic analysis is utilized to generate the corresponding HTML code based on the identified elements. By leveraging machine learning models and extensive training on HTML code examples, the system can intelligently generate structured and accurate HTML code that corresponds to the layout and design of the mock-up images.

The automation of this process significantly reduces the time and cost associated with manually translating mock-ups into HTML code. By providing an efficient and automated way to generate HTML code from mock-up images, the system aims to streamline the web development process and improve productivity.

**INTRODUCTION**

Transforming hand-drawn mock-ups into structured HTML code is often a slow and monotonous task. Developers need to spend time identifying the elements in the image and translating them into HTML manually.

To make this process faster and easier, a project is underway to automate the conversion of hand-drawn mock-ups into structured HTML code using advanced deep learning techniques. This project combines a special kind of neural network called a convolutional neural network (CNN) with a transformer-based language model.

The CNN plays the role of an intelligent observer, examining the hand-drawn mock-up image and extracting important information about its elements. It can recognize buttons, forms, menus, and other components by analyzing the image's visual patterns and characteristics.

Once the elements are identified, the transformer-based language model takes over. This model has been trained extensively on a wide range of HTML code examples, enabling it to understand the structure and syntax of HTML. It uses this knowledge to generate the corresponding HTML code based on the recognized elements from the hand-drawn mock-up.

By automating this process, developers are relieved of the burden of manually converting mock-ups into HTML code. This not only saves time but also reduces the chance of human errors. The goal of this project is to simplify the development workflow and boost productivity by seamlessly translating hand-drawn mock-ups into structured HTML code with the help of deep learning techniques.

**PROBLEM STATEMENT**

The project's objective is to automate the conversion of hand-drawn mock-up images into HTML code. By leveraging computer vision and deep learning techniques, the solution will analyze the mock-up images, identify the different elements present, and generate the corresponding HTML code. This automated process eliminates the need for manual identification and coding, saving time and effort for developers. It streamlines the web development workflow and enhances productivity.

By integrating advanced technologies, the system reduces the manual effort required to translate mock-up designs into functional HTML code. It utilizes computer vision to analyze the visual elements in hand-drawn mock-up images and employs deep learning to generate accurate HTML code. This approach not only saves time but also ensures the consistency and accuracy of the resulting HTML code. The automated process enables developers to focus more on the overall design and functionality of the website, enhancing efficiency and productivity in the web development process.

4 PRESENT THEORY

**4.1 Existing System**

Currently, the process of converting mock-up images into HTML code heavily relies on manual efforts by designers and developers. Designers typically analyze the visual elements of the mock-up images and manually write the corresponding HTML and CSS code to recreate the design in a web format. However, this manual conversion process is time-consuming and prone to errors, often leading to inconsistencies between the original design and the implemented web version.

While there are existing tools available that aim to streamline the process, they often provide only semi-automated features. These tools may utilize predefined templates or heuristics to extract design elements and generate the corresponding code. However, they often lack the flexibility to handle complex designs or variations in design styles effectively. Consequently, these tools may not produce accurate or optimal results, requiring significant manual intervention and customization.

Given these limitations, there is a clear need for an improved system that leverages machine-learning techniques to automate the conversion process more accurately and efficiently. By developing a novel solution, the proposed system aims to address the drawbacks of the existing approaches and provide a seamless and reliable way to generate HTML code from mock-up images.

By focusing on the shortcomings of the existing system, you can emphasize the necessity and importance of your proposed solution.

**4.2 Proposed System**

The proposed system aims to revolutionize the process of converting mock-up images into HTML code by leveraging machine learning techniques. By harnessing the power of artificial intelligence, the system aims to provide an accurate and efficient solution that overcomes the limitations of manual conversion and existing tools.

The key components of the proposed system include:

a. Machine Learning Model

The system will incorporate a machine learning model capable of understanding the visual elements present in mock-up images and generating corresponding HTML code. Various machine learning algorithms and architectures, such as convolutional neural networks (CNNs) or generative adversarial networks (GANs), will be explored and evaluated to determine the most effective approach. The model will be trained on a diverse dataset of mock-up images and their associated HTML code to learn the relationship between visual elements and the underlying code structure.

b. Data Preprocessing

To prepare the data for training the machine learning model, data preprocessing techniques will be applied. This may involve enhancing the quality and clarity of input images through image preprocessing methods. Additionally, the HTML code associated with the mock-up images will undergo preprocessing to extract relevant features and annotations, facilitating the training process.

c. User Interface

A user-friendly interface will be developed to enable users to input their mock-up images and obtain the generated HTML code. The interface will be intuitive and easy to navigate, allowing users to upload images, preview results, and download the generated code. The design of the interface will prioritize a seamless user experience and smooth interaction with the system.

d. Training and Optimization

The machine learning model will be trained using the collected dataset, employing appropriate training techniques and optimization algorithms. The model will learn to accurately recognize and interpret visual elements in the mock-up images, generating the corresponding HTML code. The training process will involve evaluating the model's performance using metrics such as accuracy, precision, recall, and F1 score, and iteratively optimizing the model to achieve the best results.

e. Integration and Deployment

Once the machine learning model has been trained and optimized, it will be integrated into an application or tool that generates HTML code from mock-up images. The application will offer a seamless workflow, accepting user inputs, utilizing the trained model for code generation, and presenting the output in a user-friendly format. The application will be deployed on a suitable platform or made accessible as a web-based tool, ensuring ease of use and accessibility.

f. Evaluation and Refinement

The proposed system will undergo comprehensive evaluation to assess its performance, accuracy, and efficiency. Comparative analysis with existing approaches or tools will be conducted to highlight the advantages of the proposed system. User testing and feedback will be collected to identify areas for improvement and refinement, ensuring that the system meets the requirements and expectations of end-users.

In summary, the proposed system aims to automate the process of HTML code generation from mock-up images using machine learning. By mitigating the limitations of manual conversion and existing tools, it strives to significantly reduce manual effort, improve accuracy, and enhance productivity in web development projects.

**4.3 Objective**

The objectives of this project are as follows:

a. Develop a Machine Learning Model

The primary objective of this project is to develop a machine learning model specifically designed to convert mock-up images into HTML code accurately. This model will be trained using a diverse dataset comprising mock-up images along with their corresponding HTML code. The aim is to enable the model to learn the relationships between visual elements in the images and the appropriate HTML code representation.

b. Create an Intuitive User Interface

Another objective is to design and implement a user-friendly interface that allows users to easily input their mock-up images and obtain the generated HTML code. The interface will be intuitive, providing a seamless user experience and facilitating smooth interaction with the system.

c. Evaluate Accuracy and Efficiency

The project aims to conduct a thorough evaluation of the proposed solution in terms of its accuracy and efficiency. This will involve comparing the HTML code generated by the system with manually created code for a set of mock-up images. Additionally, the efficiency of the system, such as the time required for code generation, will be measured and compared to existing manual methods.

d. Enhance Productivity and Reduce Manual Effort

An objective is to improve productivity in web development projects by automating the HTML code generation process. By minimizing the need for manual intervention and reducing errors, the system aims to streamline the conversion of mock-up images into HTML code, saving valuable time and effort for designers and developers.

e. Ensure Scalability and Adaptability

The project aims to develop a system that is scalable and adaptable to different design styles and complexities. It should be capable of handling a wide range of mock-up images and accurately generating HTML code that represents the visual elements of the design faithfully.

By achieving these objectives, the project aims to deliver an efficient and reliable solution for automating HTML code generation from mock-up images. This will significantly improve productivity, reduce manual effort, and enhance the overall web development process.

**4.4 Scope**

The scope of this project includes the following key elements:

a. Mock-up Image Conversion

The project focuses on developing a system that can convert different types of mock-up images, including web page designs and user interface layouts, into corresponding HTML code. The system will handle various design styles and complexities encountered in typical mock-up images.

b. HTML Code Generation

The project aims to create a machine-learning model that accurately generates HTML code from mock-up images. The generated code will encompass the necessary HTML structure, tags, attributes, and CSS styling instructions to faithfully represent the visual elements and layout of the mock-up.

c. Training Dataset

A diverse dataset of mock-up images along with their corresponding manually created HTML code will be collected and curated for training the machine learning model. The dataset will cover a wide range of design patterns, styles, and complexities to ensure the model's ability to generalize and handle different scenarios.

d. User Interface

The project includes the development of a user-friendly interface that allows users to input their mock-up images and obtain the generated HTML code. The interface will provide a seamless and intuitive user experience, allowing for easy interaction, previewing of results, and downloading of the generated code.

e. Evaluation Metrics

project will define evaluation metrics to assess the accuracy and efficiency of the generated HTML code. Comparative analysis with manually created code and existing tools will be conducted to evaluate the performance of the system. Metrics such as accuracy, precision, recall, and F1 score may be utilized to measure the effectiveness of the model.

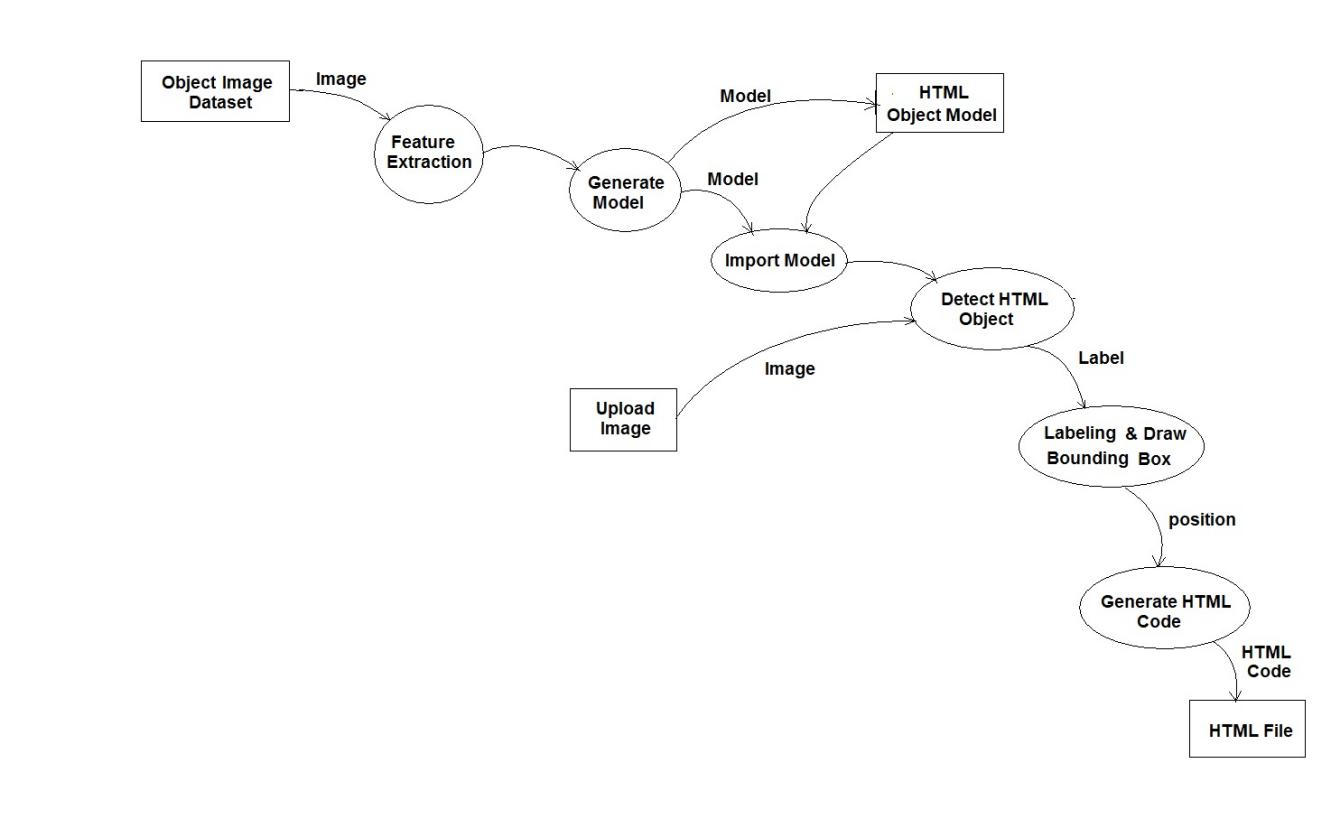
f. Limitations

It is important to recognize certain limitations of the project. Complex or ambiguous mock-up designs may pose challenges in accurately interpreting and generating HTML code. The system will primarily focus on addressing common design patterns and providing a robust solution for a wide range of typical mock-up images.

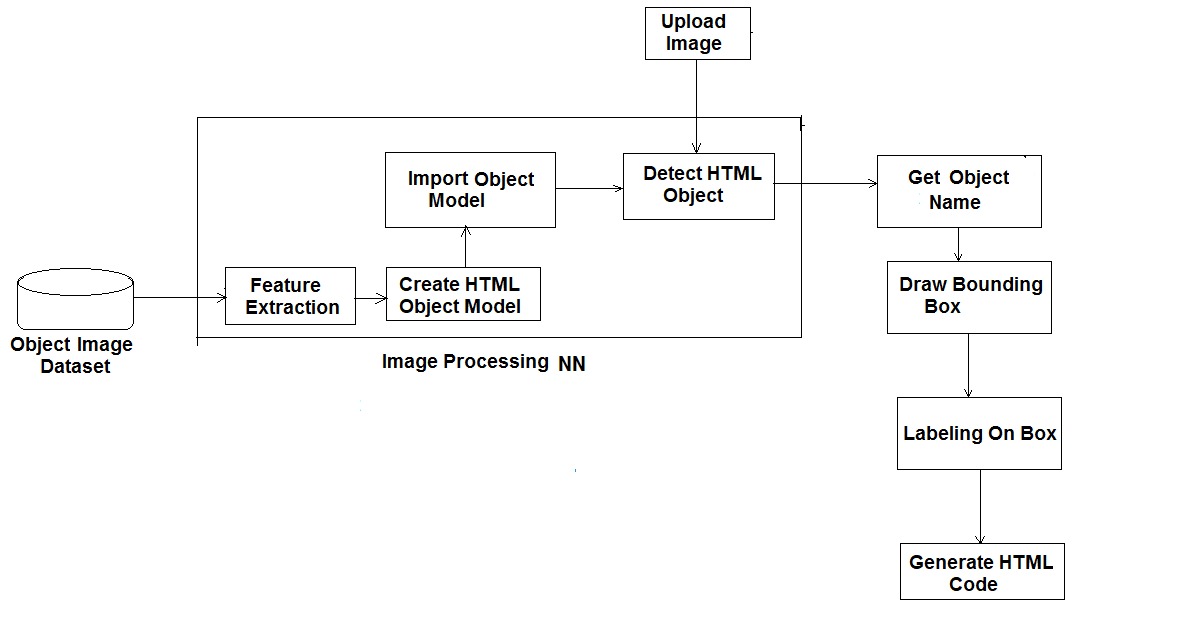
The scope of this project does not include the implementation of a full-fledged web development framework or the integration of dynamic functionality. The primary objective is to automate the generation of HTML code from mock-up images using machine learning techniques, thereby reducing manual effort and enhancing productivity in the early stages of web development.

5 METHODOLOGY

5.1 Block Diagrams



Data Flow Diagram (fig 1)



Architecture (fig 2)

5.2 Description

Mock-up Images

Image to NumPy Array

Training and Testing Model

Training and Testing split

Model Generation

The project will use a TensorFlow Lite model to detect objects in the mock-up image. The model will be trained on a dataset of mock-up images with corresponding HTML code. Once the model is trained, it can be used to detect objects in new mock-up images. The detected objects will then be used to generate HTML code.

The project will be implemented in Python using the TensorFlow Lite API. The project will also use the OpenCV library for image processing. The project will be divided into the following steps:

Data preparation: The first step is to prepare the data. This includes collecting a dataset of mock-up images with corresponding HTML code. The dataset can be collected by manually creating the images and code, or by scraping the web.

Model training: The next step is to train the TensorFlow Lite model. This can be done by using the TensorFlow Lite API. The model will be trained on the dataset of mock-up images with corresponding HTML code.

Model inference: Once the model is trained, it can be used to infer the HTML code for a new mock-up image. This can be done by using the TensorFlow Lite API. The model will be given the mock-up image as input, and it will output the corresponding HTML code.

HTML code generation: The final step is to generate the HTML code. This can be done by using the OpenCV library to extract the detected objects from the mock-up image. The detected objects will then be used to generate the HTML code.

5.3 Algorithms

5.3.1 - Image Processing Neural Network :

An image processing neural network is a type of artificial neural network that is used to process images. Image processing neural networks are typically used for tasks such as image classification, object detection, and image segmentation.

Image processing neural networks work by learning to identify patterns in images. The neural network is trained on a dataset of images that have been labeled with the correct labels. For example, a neural network that is being trained for image classification might be trained on a dataset of images that have been labeled with the names of different objects.

Once the neural network is trained, it can be used to process new images. The neural network will identify the patterns in the new image and then classify the image or identify the objects in the image.

Image processing neural networks are a powerful tool that can be used to automate many tasks that were previously done manually. Image processing neural networks are used in a variety of applications, including:

Image classification: Image classification is the task of identifying the objects in an image. Image processing neural networks can be used to classify images of objects such as cars, people, and animals.

Object detection: Object detection is the task of identifying the location and size of objects in an image. Image processing neural networks can be used to detect objects such as faces, cars, and traffic lights.

Image segmentation: Image segmentation is the task of dividing an image into different regions. Image processing neural networks can be used to segment images into regions such as foreground and background, or different objects in an image.

Image processing neural networks are a rapidly evolving field. New research is being conducted all the time to improve the accuracy and performance of image processing neural networks.

As image processing neural networks become more accurate and efficient, they will be used in even more applications.

Benefits of using image processing neural networks:

Accuracy: Image processing neural networks can be very accurate at identifying patterns in images. This is because they are trained on large datasets of images that have been labeled with the correct labels.

Speed: Image processing neural networks can be very fast at processing images. This is because they are able to learn to identify patterns in images very quickly.

Scalability: Image processing neural networks can be scaled to process large datasets of images. This is because they are able to learn to identify patterns in images very quickly.

5.3.2 - Object detection using TensorFlow :

TensorFlow Lite is a machine learning framework that makes it easy to deploy machine learning models to mobile and embedded devices. TensorFlow Lite models are smaller and faster than TensorFlow models, making them ideal for devices with limited resources. TensorFlow Lite also supports a variety of hardware platforms, including Android, iOS, and Raspberry Pi.

TensorFlow Lite can be used for object detection by using a pre-trained model or by training your own model. Pre-trained models are available for a variety of object detection tasks, such as detecting faces, cars, and animals. You can also train your own model by using a dataset of images that have been labeled with the objects that you want to detect.

Once you have a TensorFlow Lite model for object detection, you can use it in your mobile or embedded application. TensorFlow Lite provides a variety of APIs that make it easy to integrate object detection into your application.

5.3.3 Model Implementation:

1. Load the TFLite model.

The TFLite model is a small, lightweight model that can be used to perform object detection on mobile devices. The model is loaded using the tf.lite.Interpreter class.

1. Load the label map.

The label map is a file that contains the names of the objects that the model can detect. The label map is loaded using the open("labelmap.txt") function.

1. Define a list of colors for visualization.

A list of colors is defined for visualization. The colors are used to draw the bounding boxes around the detected objects.

1. Create a function to set the input tensor.

A function is created to set the input tensor. The function takes an image as input and sets the input tensor of the model to the image.

1. Create a function to open a new window to display the detected image.

A function is created to open a new window to display the detected image. The function takes the name of the image file as input and opens a new window with the image displayed.

1. Create a function to open a new window to display the HTML code.

A function is created to open a new window to display the HTML code. The function takes the HTML code as input and opens a new window with the code displayed.

1. When the user opens a file, load the image and preprocess it.

When the user opens a file, the image is loaded using the cv2.imread() function. The image is then preprocessed by resizing it to the input size of the model.

1. Run object detection on the image.

Object detection is run on the image using the interpreter.invoke() function. The function returns a list of detection results.

1. Plot the detection results on the image.

The detection results are plotted on the image using the cv2.rectangle() and cv2.putText() functions. Save the detection result image. The detection result image is saved using the cv2.imwrite() function.

1. Show the detection result image in a new window.

The detection result image is shown in a new window using the cv2.imshow() function.

Generate HTML code for the detection results.

HTML code is generated for the detection results using the open\_html\_code() function. The function takes the list of detection results as input and generates HTML code that displays the results.

Show the HTML code in a new window.

The HTML code is shown in a new window using the webbrowser.open() function.

The methodology is based on the following principles:

* Use a lightweight, efficient machine-learning framework.
* Train the model on a dataset of images that contain objects of interest.
* Define a list of colors for visualization.
* Create functions to set the input tensor, open a new window to display the detected image, and open a new window to display the HTML code.
* When the user opens a file, load the image and preprocess it.
* Run object detection on the image.
* Plot the detection results on the image.
* Save the detection result image.
* Show the detection result image in a new window.
* Generate HTML code for the detection results.
* Show the HTML code in a new window.