# SNAP2SITE: AUTOMATED WEB PAGE TEMPLATE GENERATION WITH INTEGRATED DIGITAL BRANDING

**Bato, Leovide Daniel**
*Saint Louis University*
Bakakeng Sur, Baguio City
2223451@slu.edu.ph

**Del Rosario, Remo**
*Saint Louis University*
Bakakeng Sur, Baguio City
2227393@slu.edu.ph

**Gura, Kaizer Cyn**
*Saint Louis University*
Ambiong Road, Baguio City
2221097@slu.edu.ph

**Narag, Ava**
*Saint Louis University*
Bakakeng Sur, Baguio City
2224512@slu.edu.ph

**Oman, Kaizer**
*Saint Louis University*
Bakakeng Sur, Baguio City
2222613@slu.edu.ph

## ABSTRACT

The increasing complexity of website design challenges developers in effectively organizing web application components. AI-driven automation has revolutionized website creation, enhancing efficiency and adherence to web design standards. The dynamic business environment also introduces the challenge of "digital branding," which defines a website's impact on the organization it represents. This study highlights the creation of an autonomous model using computer vision, leveraging object detection models, such as Roboflow and YOLOv8, to automate the conversion of webpage screenshots to HTML code. Thus, it aims to develop a model that adheres to the World Wide Web Consortium standards and can integrate digital branding principles, making it a unique contribution to the field of web development. Comparing the outputs of the Roboflow and YOLOv8 models on simple and complex web pages reveals significant differences in their detection capabilities. While the Roboflow model performed adequately on the simple webpage, detecting an adequate number of classes, it overlooked certain elements and misclassified others. In contrast, the YOLOv8 model demonstrated superior performance, accurately detecting various components. For future studies, the proponents recommend more techniques for digital brand incorporation for further research to enhance users' integration of their digital brand. Recommendations for future research include continuously enhancing the object detection models to accurately identify and classify a broader range of web page components, particularly within intricate web layouts.

***Keywords:*** *YOLOv8, Roboflow, Web Components, Website Screenshot, Digital Branding, Object Detection*

## I. INTRODUCTION

Nowadays, the ever-growing complexity of website design has become devious to developers as it poses challenges to successfully structuring and organizing the components [1]. One of the web application standards that has been developed by the World Wide Web Consortium (W3C) is called the “web components” [2]. Web components are defined as elements in a web page that are encapsulated from the rest of the code; thus, reusing and customizing them would not necessarily affect the web page as they are styled to prevent code collision [3]. Component-based web designs have proven effective in designing consistent user interfaces and experiences[4]. While they are divided into pre-made templates, they can also be advantageous in promoting and designing adaptive websites because a developer can manipulate them easily. Luckily, AI-driven automation has significantly reformed how developers solve problems–such as creating sites, serving developers' efficiency, and adhering to web standard designs, including human factors and ergonomics (having end users in mind at all times). The integration of machine learning and computer vision aims to bring human capabilities to data sensing, data understanding, and action-taking based on past and present outcomes into computers. This integration is revolutionizing the way websites are designed and developed. Computer vision, in particular, is an essential part of the Internet of Things, the Industrial Internet of Things, and brain-human interfaces, unlocking new possibilities for intelligent and adaptive website experiences [7]. By leveraging the power of machine learning and computer vision, developers can automate repetitive

tasks, make data-driven design decisions, personalize website experiences, and improve overall website responsiveness and performance. This AI-driven approach transforms the website design and development landscape, enabling developers to create more efficient, user-centric, innovative web solutions.

Latterly, with AI novelties, there are many well-established methods for prediction and analysis, such as supervised learning, unsupervised learning, and semi-supervised learning, which harness the abilities of object detection and image classification models. Some existing models include the Roboflow and YOLO models. Roboflow is a comprehensive computer vision platform that offers state-of-the-art object detection and image segmentation models, providing solutions for a wide range of tasks. Developed by Ultralytics, YOLOv8 is a cutting-edge object detection and image segmentation model that builds upon the success of its predecessor, YOLOv5. Launched in 2023, YOLOv8 demonstrated superior performance, achieving an impressive 80.2% mAP score on the Roboflow 100 benchmark, showcasing its effectiveness in domain-specific tasks. The proponents of this study will use these to develop and train the model.

Bakaev et al. [8] present a framework for automatically extracting and integrating metrics from web user interfaces. The researchers leveraged object detection techniques to identify and categorize UI elements within website screenshots. By recognizing and highlighting grids, images, text, and icons, the framework enabled a more comprehensive understanding of the layout and structure of web content. In addition, Goyal, et al. [9] focused on developing a method for the efficient layout analysis of screenshots on mobile devices. The researchers employed object detection to identify and segment different UI components within the screenshots, facilitating tasks such as smart editing, sharing, and saving visual information. The incorporation of object detection in the annotation process of website screenshots for the model being developed is crucial. By automatically recognizing and highlighting UI elements, the object detection techniques discussed in these studies contribute to the accuracy and efficiency of data processing, ultimately enhancing the overall understanding and analysis of web interfaces for the model. Finally, Anunphop, et al. [10] model can recognize the object of interest in the web screenshot, and his generator module can also transform the predicted result into a web components template using the RetinaNET object detection model. Furthermore, vision-language models (VLMs) for web development present a promising strategy to increase efficiency and unblock no-code solutions. In order to automate the conversion of webpage screenshots to HTML code, Laurençon et al. work [6] introduced two tools: Sightseer, a vision and language model with OCR ability fine-tuned on WebSight, and WebSight, a large synthetic dataset of 2 million pairs of HTML codes and corresponding renderings.

However, there are research gaps that warrant further exploration in these studies. [9] proposes a novel mechanism for Hierarchical Layout Analysis of Mobile Screenshots, focusing on entity segmentation and image processing techniques to enable real-time, on-device layout understanding. A potential research gap lies in investigating the scalability and adaptability of the proposed framework across a wider range of mobile devices and screen layouts. Additionally, there is an opportunity to explore the robustness and accuracy of the entity segmentation and image processing techniques under varying conditions and complexities of mobile application screenshots. [8], while the effectiveness of object detection for automatic extraction and integration of metrics is demonstrated, further research is needed to explore the scalability, accuracy, and robustness of the framework in handling diverse and complex web content layouts. Additionally, enhancing the framework's adaptability to evolving digital content and app layouts, especially in the context of mobile application screenshots, where the structure and constituent elements are highly diverse and complex, presents a promising avenue for future research. Lastly, [6] identified limitations in the model's ability, Sightseer, to generalize to websites that differ from those in the training dataset. Additionally, the validation loss does not indicate the trained model's ability to generate high-quality codes in real-world cases. Despite the validation loss decreasing significantly over several epochs, it did not translate into an increased ability to generalize to different websites.

Assimilating machine learning and computer vision can remold how websites are designed and developed, allowing developers to automate repetitive tasks, make data-driven design decisions, personalize website experiences, and improve overall website responsiveness and performance. This AI-driven approach is transforming the website design and development landscape; however, the fast-paced business environment has introduced another challenge, known as "digital branding," which defines the impact of a website on the organization it belongs to. Digital branding establishes the impact of a website on the organization it belongs to; it is a mark that differentiates a company from the rest [5]. This leads to the research question: "How can we leverage the power of AI (Artificial Intelligence) to develop a model that can revolutionize front-end development among businesses?"

This study highlights the creation of an autonomous model using computer vision. Therefore, the proponents propose to leverage the power of object detection models, such as Roboflow and YOLOv8, to develop a model that can automate the conversion of webpage screenshots to HTML code. Incorporating object detection techniques is crucial in understanding and processing website interfaces. Thus, it aims to build upon the findings and develop a model that not only adheres to the World Wide Web Consortium (W3C) standards but also integrates digital branding principles, making it a unique contribution to the field of web development.

## II. METHODOLOGY

This section presents all the steps taken in the study, detailing the methods employed, data collection procedures, tools involved, and various AI models utilized, as well as providing the logical and systematic approach to achieve the research objectives and ensure the validity and reliability of the study findings.

### *II.1. Data collection*

The available datasets sourced online were found inadequate for the research objectives due to two primary reasons. Firstly, many of these datasets lacked full screenshots of web pages, which is essential for the analysis. Secondly, even among the datasets that did provide screenshots, the annotation was limited to only a small number of classes. Given the nature of this study, it is imperative to have full screenshots of web pages along with a sufficient number of classes to classify the various components present on these web pages.

For this reason, the proponents created a custom dataset manually. The process involved three main steps: first, capturing a full screenshot of a webpage; second, annotating the screenshots with the necessary class components; and lastly, uploading these annotations to a valid format for model training. To facilitate these procedures, the Google Chrome extension 'Fireshot' [11] (Figure 1) was utilized for capturing the full webpage screenshots, and the Roboflow website [12] was employed for annotations and conversion into a valid dataset format, as illustrated in Figure 2. Finally, the dataset was uploaded in MS COCO format, suitable for both YOLOv8 and Roboflow object detection models.

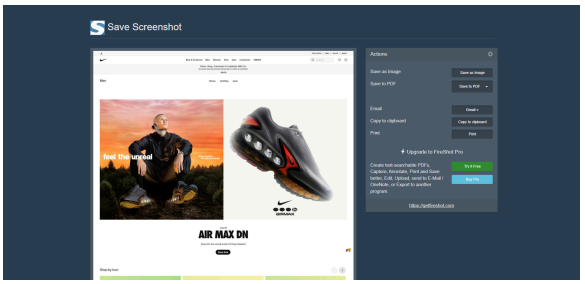


***Figure 1:*** *Capturing a screenshot with Fireshot*

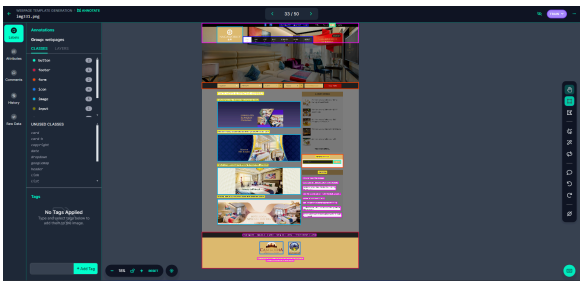

***Figure 2:*** *Sample webpage screenshot annotation using the Roboflow website*

The dataset contains 203 full webpage screenshots from various business websites with varying sizes to increase the model's flexibility. This includes main web pages for coffee shops, fashion and apparel, news, art galleries, restaurants, and hotels. Subsequently, annotation was performed on these screenshots, categorizing them into a total of 22 classes, which are further divided into two categories: the HTML tags and Custom tags.

### *II.2. Object Detection Models*

In model training, two object detection model architectures were utilized: the Roboflow 3.0 and the YOLOV8.

First, the Roboflow model architecture. In an effort to explore the most optimal Roboflow object detection model for the dataset, two models were trained with distinct preprocessing techniques applied. The preprocessing done in the first model(R1) consists of auto-orienting and resizing the images to 1000x1000. While the second one(R2), consists of auto-orienting, resizing to 640x640, grayscaling, and tiling images to 20 by 20 for better detection of smaller components. Using a free research plan account in Roboflow, the only available model size that can be opted for is the 'Fast' model size, which is less accurate than the 'Accurate' option but is quicker to train and infer. After being able to train the two Roboflow models, to further improve their performance, the proponents trained another two similar models. The first one(R3) was pre-trained on the weights of the first model(R1), and the second one(R4) was pre-trained on the weights of the second model(R2). The same preprocessing was done on both pre-trained models, consisting of auto-orienting, resizing the images to 1000x1000, and splitting the dataset in a ratio of 88:7:5. Furthermore, all the procedures for training the Roboflow models were made using the Roboflow website.

Second, the implementation of the YOLOv8 model. Similar to what has been done on the Roboflow model. The same preprocessing techniques were used in the training methods. The first YOLOv8 model was trained on 300 epochs, and another one was pre-trained on the weights of the first model with 100 epochs to improve its performance further.

To compare the models' output, both were tested on 2 sample web pages, categorized as simple and complex, wherein one was identified as simple and the other was a complex webpage. Defined by the size of their dimensions and the number of components involved, simple ones have smaller dimensions and number of components than complex ones.

***II.3. Code Generation***

For HTML/CSS code generation of the classified objects from the webpage screenshot. The first step was to map the labeled components to a webpage component, as presented below (Table 1). Then, use Python to extract the positions x and y of the class in the webpage, their height and width, and their labels to convert them to a webpage component.

| No | Class | Category | Component |
|---|---|---|---|
| 1 | button | HTML | btn |
| 2 | card | Custom | card |
| 3 | card-h | Custom | card-h |
| 4 | copyright | HTML | copyright |
| 5 | date | Custom | date |
| 6 | dropdown | HTML | select |
| 7 | footer | HTML | footer |
| 8 | form | HTML | form |
| 9 | googlemap | Custom | googlemap |
| 10 | header | HTML | header |
| 11 | icon | Custom | icon |
| 12 | image | HTML | img |
| 13 | input | HTML | input |
| 14 | link | HTML | a |
| 15 | list | HTML | ul:li |
| 16 | navbar | HTML | nav |
| 17 | search | HTML | search |
| 18 | span | HTML | span |
| 19 | text | HTML | p |
| 20 | text-h | HTML | h3 |
| 21 | textcard | Custom | textcard |
| 22 | video | HTML | video |

***Table 1:*** *Component Mapping*

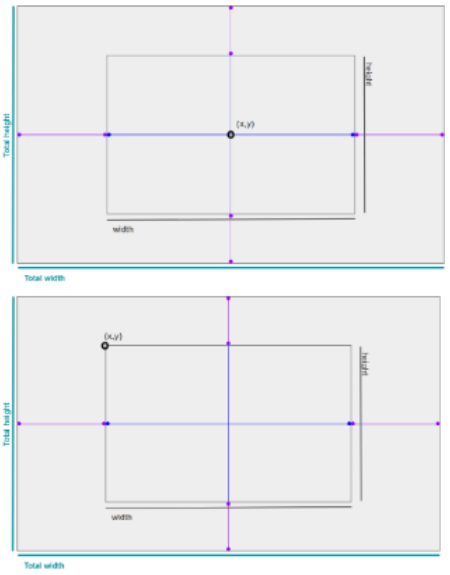

***Figure 3:*** *Position x and y output from Object Detection(Top) and expected position x and y in HTML(Bottom)*

As shown in Figure 3, the proponents have identified that there is a difference in the x and y positions of the component from an object detection output(Top photo) and the expected x and y positions of a component on an HTML webpage(Bottom Photo). With this information, a component's relocation for x and y to align with the HTML standard is needed. To achieve this, the proponents used the formula:

*x = x - (component width / 2)px*
*y = y - (component height / 2)px*

Having been able to relocate the x and y positions, the components were divided into three divisions according to where they are placed: header, main, and footer. Then the y positions of the components were relocated again relative to the 'division' they belong to using the formula:

*y = y - (total height - footer)px*

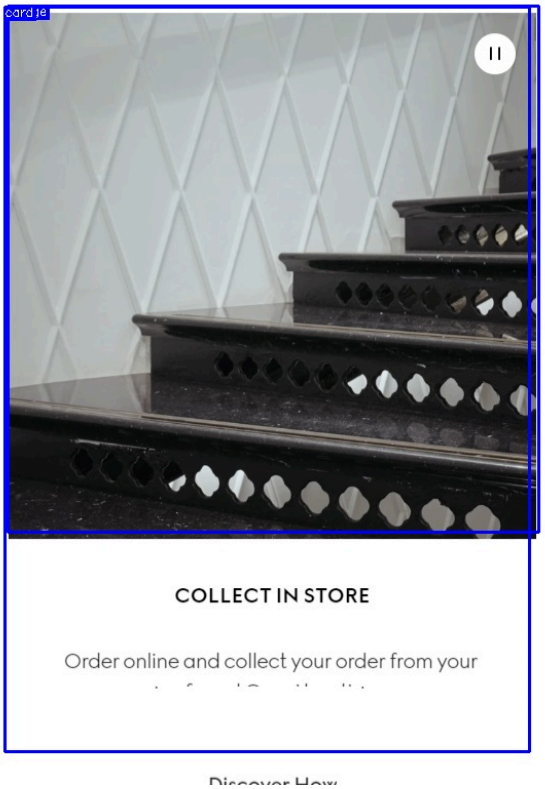


***Figure 4:*** *Unnecessary nested bounding box*

Furthermore, removing unnecessary predicted nested bounding boxes was performed to improve the template generation output. Figure 4 shows an example of an unnecessary nested box in which the image is nested on a card, which is a component designed to come with a photo and text. More examples include a text nested on a text card component that already comes with text inside or a link nested in a navbar.

### *II.4. Custom Tags Creation*

Aside from HTML tags, 6 custom tags were used to identify some of the annotated classes. This includes Card, Card-h, Date, Googlemap, Icon, and Text Card. The card tag is any element in the webpage that looks like a card, it contains an image on top and any text or button below it; an example is shown in Figure 5. Card-h is similar to the card, but the image is either on the right or left side, and the text is adjacent to the image; an example is shown in Figure 6. The date tag is any webpage element containing the current date. The Icon tag is any element in the web page containing a PNG photo, like logos, social media icons, and menu icons. Lastly is the text card which is also similar to cards but with no image inside the card, only text or button; an example is shown in Figure 7.


***Figure 5:*** *The left photo shows a sample card from a screenshot, and the right is its corresponding component.*


***Figure 6:*** *The top photo shows a sample card-h from a screenshot, and below it is its corresponding component.*

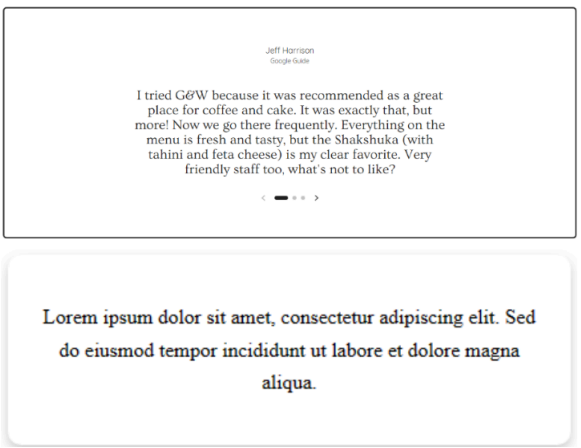


***Figure 7:*** *The top photo shows a sample text card from a screenshot, and below it is its corresponding component.*

### *II.5. Digital Branding*

The proponents designed a prototype of the website. This aims to provide users the ability to effectively customize the templates generated by the model to align with their digital branding preferences. This approach facilitates a more tailored and cohesive online presence for businesses or individuals, ensuring that their branding elements are seamlessly integrated into the website design. Initially, the customization techniques that will be employed for the first version of the model will be color palette customization, and fonts customization for each division of the webpage: header, main, and footer.

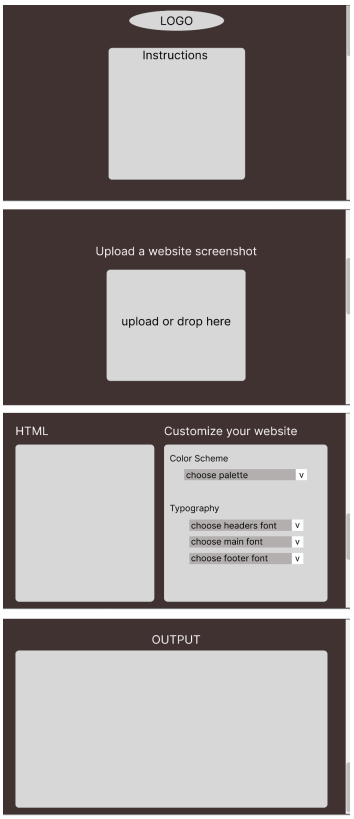


***Figure 8****: Website Prototype*

## III. RESULTS AND DISCUSSION

This section presents the results of the methods employed and experiments. It includes the presentation, metrics analysis, and interpretation of the model outturn.

### *III.1. Roboflow Model*

Four models were made using the Roboflow architecture: the R1, R2, R3, and R4. The first two models, R1 and R2 were trained first, and then the last two, R3 and R4, were pre-trained on the weights of the first two trained models(R1, R2).

Table 2 shows the overall performance metrics of the 4 Roboflow models, with the number of epochs in which they were trained. The result shows that the R3 achieved the highest mAP and precision among the models, indicating the best overall performance. Thus, since the pretrained R3 model has a better recall than the R4, the proponents will use the R3 model to compare with the YOLOv8 Model.

| M | Epochs | mAP | Precision | Recall |
|---|---|---|---|---|
| R1 | 299 | 26.4% | 41.1% | 24.3% |
| R2 | 47 | 29.1% | 33.9% | 34.4% |
| R3 | 146 | 34% | 59.2% | 30.9% |
| R4 | 240 | 29.6% | 53.8% | 29.9% |

***Table 2:*** *Performance Metrics of the Roboflow Models with the number of epochs in which they were trained*

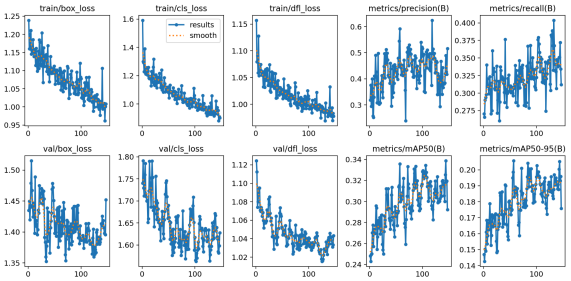


***Figure 9:*** *Pretrained R3 Roboflow Model Training Graphs*

Shown in Figure 9 is the training graph of the R3 Model, achieving a mAP of 34% at the end of 146 epochs, and the decreasing Box loss, Class loss, and Object loss indicate that the model is making progress in terms of accurately detecting and classifying objects in the training. However, fluctuations during the training indicated variations in its performance, suggesting that further fine-tuning and additional training epochs might be needed to stabilize its performance. Despite the fluctuations, the trend still moved upward, showing a small improvement.

### *III.2.* YOLOv8 Model

Similar to the approach employed with the Roboflow models, two models were made using the YOLOv8 architecture, the first one being the first trained model and the second one being the pre-trained model on the weights of the first model.

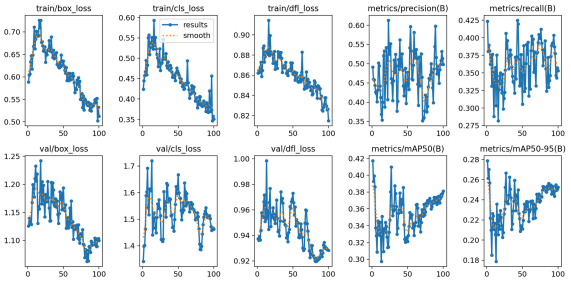


***Figure 10:*** *Pre Trained YOLOv8 Model Training Graphs*

Figure 10 shows the training graph of the pre-trained YOLOv8 model. Using the weights of the first model. From the graphs, the model significantly improved its initial performance across all metrics and losses, improving from pre-trained weights. Compared to the R3 Model, the YOLOv8 Model yielded better and accurate training and validation graphs with fewer fluctuations showing more consistent improvement for both precision and recall metrics in terms of stability. In the context of overall performance, the pretrained YOLOv8 model achieved higher mAP values making it the superior choice than the other model.

### *III.3. Model Comparison*

Figure 11 shows a sample simple webpage in which the proponents tested the two models as well as their result when their outputs were fed into the code generator shown in Figure 12 and 13.

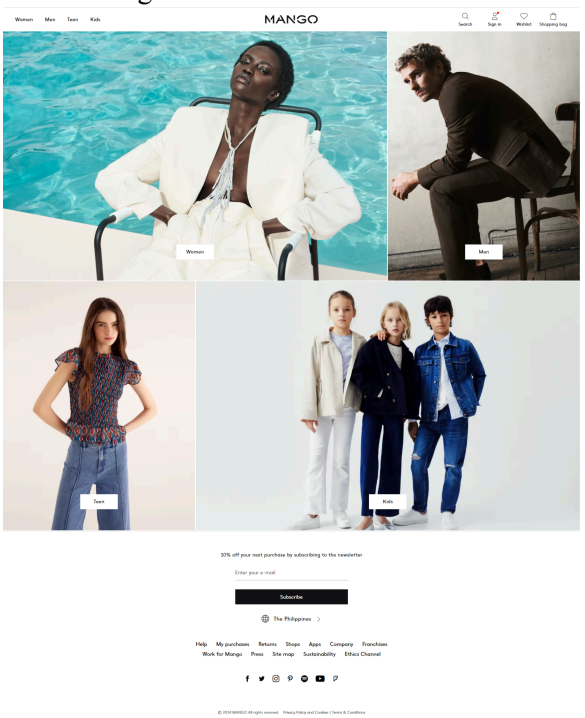


***Figure 11:*** *Sample simple web page*

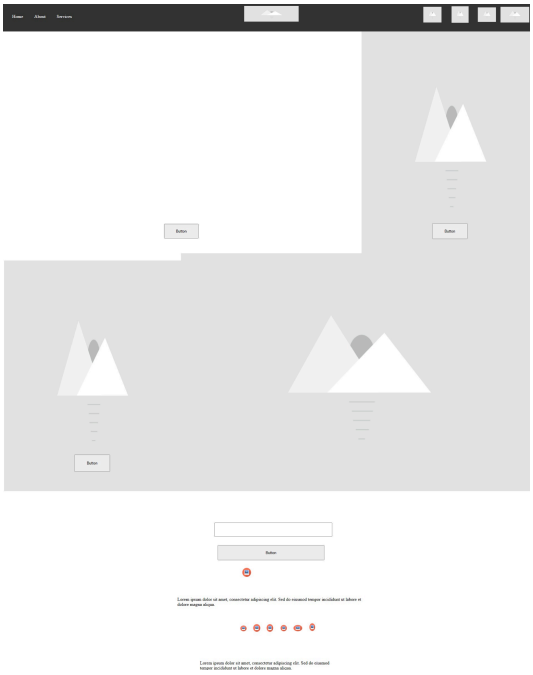

***Figure 12:*** *Model output with 30% minimum confidence using Roboflow on the sample simple web page*

***Figure 13:*** *Model output with 30% minimum confidence using YOLOv8 on a sample simple web page*

The output of the Roboflow model by uploading a simple webpage yielded the detection of an adequate amount of classes, as shown in Figure 12. It recognized the header, navbar, icons, images, buttons, and input field. However, it overlooked an image and text. Furthermore, misconceived a menu in the footer as normal text, which may be attributed to the model's reliance on visual features rather than semantic understanding. Forbye, the model's performance was likely influenced by the data imbalance inherent in the training dataset, where certain classes may have been overrepresented, leading to biased detection. Compared to the output of the code generator using Roboflow, the output of the code generator with the YOLOv8 detection model(Figure 14), yielded better precision on predicting the positions of the components as well as their classes.

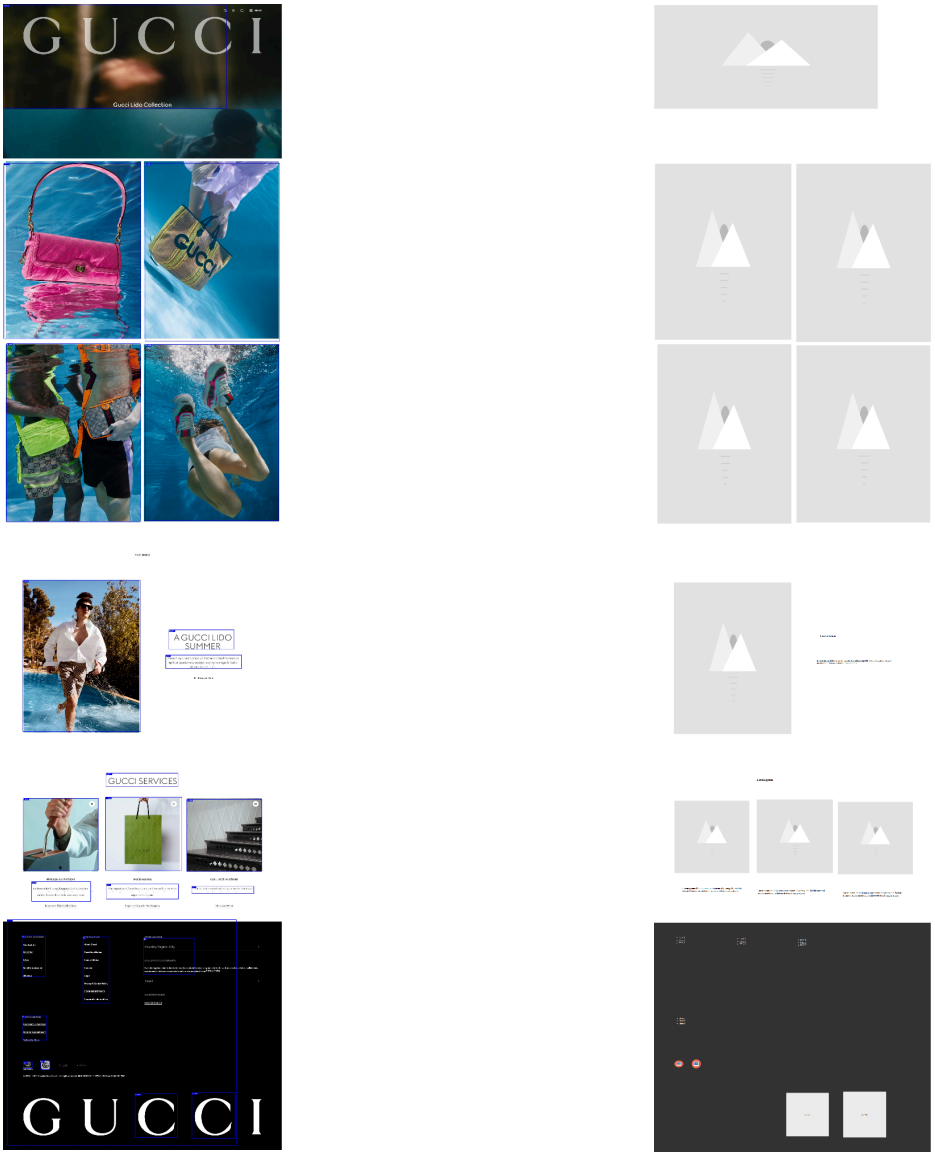


***Figure 14:*** *Model output with 30% minimum confidence using Roboflow on a sample complex webpage*

Illustrated in Figure 14, the Roboflow model output on a sample complex webpage successfully identified a majority of the classes from the uploaded input. However, it exhibited limitations in detecting specific elements, notably failing to recognize the icons in the header, the full image size, several text components, and both text and icons in the footer section. Furthermore, the model erroneously interpreted the text and spans in the footer as a list.

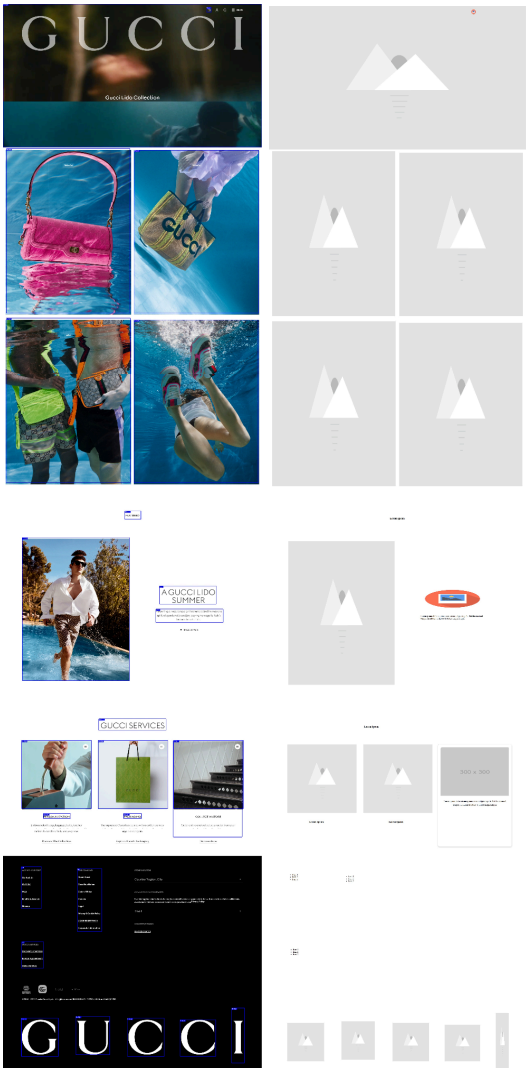


***Figure 15:*** *YOLOv8 Object detection output with a minimum 30% confidence threshold on a complex webpage*

As for the YOLOv8 model detecting website components with an uploaded complex webpage, it performed better than the Roboflow model. As shown in Figure 15, YOLOv8 detected more accurate classes, such as icons, text-cards, and texts. While the Roboflow model may have identified a larger quantity of classes, the YOLOv8 model's accuracy in detecting the correct classes is more crucial for effective website component identification.

Testing the Roboflow model by uploading simple and complex websites as input showed a contrast in the model's ability to accurately detect various elements, highlighting the impact of webpage complexity on detection accuracy. The discrepancies observed in the model's performance underscore the importance of considering the intricacies of webpage design and content. Object detection models like Roboflow primarily focus on identifying visual patterns and shapes to classify elements without considering the underlying meaning or purpose of the components. Thence, as the model needed to understand the complexity of its detecting website, it can misinterpret a fair few components, such as a video to a standard image. A possible cause is the user's uploaded still image (screenshot of the webpage) as it is hard to distinguish a moving picture for the model and still in the uploaded image. The model's performance may also be influenced by the data imbalance inherent in the training dataset, where certain classes may have been overrepresented, leading to biased detection.

Comparing the outputs of the Roboflow and YOLOv8 models on simple and complex web pages reveals significant differences in their detection capabilities. While the Roboflow model performed adequately on the simple webpage, detecting an adequate number of classes, it overlooked certain elements and misclassified others. In contrast, even on the complex webpage, the YOLOv8 model demonstrated superior performance, accurately detecting a more comprehensive range of components, including icons, text cards, and texts. The YOLOv8 model's ability to accurately detect website components, particularly on complex web pages, is crucial for practical website analysis and automation tasks. Accurate detection enables a better understanding of web page structure, content, and functionality, which is essential for applications such as web scraping, content extraction, and user experience optimization.

### *III.4. Incorporating Digital Branding*

From the implementation of the digital brand incorporation on the template output of the code generator. The proponents were able to customize the color palette of the webpage and typography, as shown in Figure 16.

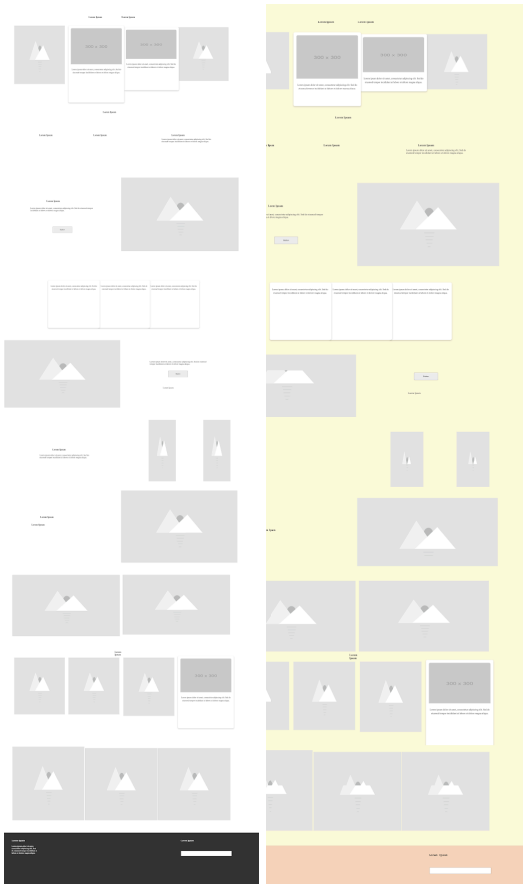

***Figure 16:*** *Digital Branding Incorporation*

## IV. CONCLUSION

In conclusion, the research paper focused on leveraging object detection models, specifically Roboflow and YOLOv8, to automate the conversion of webpage screenshots into HTML code. By adopting these models, the research aimed to improve website development processes and integrate digital branding principles, offering a significant impact to the field of web development.

The results revealed that the R3 pre-trained Roboflow model, had promising performance metrics, achieving the highest mAP, precision, and recall among the trained roboflow models, exhibiting its ability to effectively recognize and classify web page components. However, the YOLOv8 model outperformed the Roboflow model in detecting website components on both simple and complex web pages, emphasizing the importance of considering webpage complexity for optimal model performance. The problem in the models mainly revolves around class imbalance, lack of more classes to identify more components, inadequate dataset to represent various categories of webpages as well as the lack for further preprocessing and data augmentation to identify smaller components like the copyright class.

In the integration of digital branding customization with the code generator output, only minor customization techniques were employed to incorporate users' digital brand into the template. For future studies, it is recommended to research more techniques for digital brand incorporation to further enhance users' integration of their digital brand.

The significance of this research extends to both societal and business realms, offering substantial practical implications. Through the automation of webpage template generation and integration of digital branding elements, developers stand to improve efficiency in web development, enhance user experience, and establish distinctive digital identities for organizations and individuals. This approach not only streamlines website creation but also ensures seamless branding integration, fostering a more engaging online environment. Furthermore, the integration of AI-driven automation in web development processes promotes cost savings, heightened productivity, and accelerated time-to-market for digital products and services. The study's findings also contribute to advancing AI technologies within web development, paving the way for more sophisticated and user-centric digital experiences. Moreover, the research outcomes hold potential to influence industry practices by advocating for the adoption of AI-powered tools in website design and development, thereby stimulating innovation and bolstering competitiveness in the digital landscape.

Despite the advancements made in automating web page template generation, certain limitations exist within the scope of this study. These limitations include the possible difficulty in effectively detecting all web page components, particularly in complex web designs, as well as the need for additional refining to improve model accuracy and robustness. Furthermore, the study's emphasis on object detection models may overlook other important aspects of web page analysis, such as semantic understanding and context interpretation, while the models' generalizability to a broader range of website types and design styles remains a potential limitation that requires further research.

Recommendations for future research include continuous enhancement of the object detection models to accurately identify and classify a broader range of web page components, particularly within intricate web layouts. This could involve integrating advanced AI techniques such as natural language processing to automate web page content extraction and analysis. Understanding the impact of AI-driven automation on website accessibility and inclusivity would provide valuable insights for improving digital experiences for diverse user groups. Additionally, incorporating user feedback mechanisms into the automated webpage generation process is recommended to ensure alignment with user preferences and usability standards. Further exploration of the scalability of the automated model across different web development frameworks and platforms would enhance its applicability in diverse web development environments.

## REFERENCES


[1] Vu, K. L., Proctor, R. W., & Hung, Y. (2021). WEBSITE DESIGN AND EVALUATION. Handbook of Human Factors and Ergonomics, 1016–1036. Retrieved from https://doi.org/10.1002/9781119636113.ch39

[2] Pattana Anunphop, & Prabhas Chongstitvatana. (2021). Web Components Template Generation from Web Screenshot. Retrieved from https://doi.org/10.1145/3468784.3468787

[3] Web Components - Web APIs | MDN. (2023, June 11). Developer.mozilla.org. Retrieved from https://developer.mozilla.org/en-US/docs/Web/API/Web_Components

[4] Shah, H. (2023, November 25). Harnessing customized built-in elements -- Empowering Component-Based Software Engineering and Design Systems with HTML5 Web Components. Retrieved from https://doi.org/10.5121/csit.2023.132219

[5] Rowles, D. (2022). Digital Branding: A Complete Step-by-Step Guide to Strategy, Tactics, Tools and Measurement. In Google Books. Kogan Page Publishers. Retrieved from https://books.google.com.ph/books?hl=en&lr=&id=BDpUEAAAQBAJ&oi=fnd&pg=PP1&dq=digital+branding+on+websites&ots=WeDuRDC1rQ&sig=DsW7p-VGW9SfDXI8In55zYjCrJM&redir_esc=y#v=onepage&q=digital%20branding%20on%20websites&f=false

[6] Laurençon, H., Tronchon, L., & Sanh, V. (n.d.). Unlocking the conversion of Web Screenshots into HTML Code with the WebSight Dataset. Retrieved from https://arxiv.org/pdf/2403.09029

[7] Khan, Asharul Islam, and Salim Al-Habsi. “Machine Learning in Computer Vision.” *Procedia Computer Science*, vol. 167, 2020, pp. 1444–1451. Retrieved from https://doi.org/10.1016/j.procs.2020.03.355.

[8] Bakaev M., Heil S., khvorostov V., & Gaedke M., (2019 March). Auto-Extraction and Integration of Metrics for Web User Interfaces. Retrieved from https://www.researchgate.net/figure/A-web-interface-screenshot-with-UI-elements-recognition-results-highlighted-in-the_fig3_331826158

[9] Goyal, M., Ai, O., Garg, D., Prasanna, D., Ondevice, M., Prasad, S., & Ondevice, T. (2021). ScreenSeg: On-Device Screenshot Layout Analysis. Retrieved from https://arxiv.org/pdf/2104.08052

[10] Anunphop, P., & Chongstivatana, P. (n.d.). Web Components Template Generation from Web Screenshot. Retrieved from https://www.cp.eng.chula.ac.th/~prabhas/paper/2021/web-component-iait2021.pdf

[11] Take Webpage Screenshots Entirely - FireShot. (n.d.). Chromewebstore.google.com. Retrieved from https://chromewebstore.google.com/detail/take-webpage-screenshots/mcbpblocgmgfnpjjppndjkmgjaogfceg

[12] Roboflow: Go from Raw Images to a Trained Computer Vision Model in Minutes. (n.d.). Roboflow.ai. Retrieved from https://roboflow.com/