Beyond Code: A Comprehensive Study on Website Builders, Their Limitations, and Opportunities for Innovation

CCDS24-0616

Submitted in Partial Fulfilment of the Requirements

for the Degree of Bachelor of Computing in Computer Science

of the Nanyang Technological University

Lim Boon Hian

[redacted]

Project Supervisor: Dr Li Fang

Examiner: Prof Ong Yew Soon

College of Computing and Data Science

Academic Year 2024/2025

Abstract

The rise of website builders has simplified online presence creation for individuals and businesses. However, existing platforms face limitations which hinder broader adoption. This study investigates the current landscape of website builders through case studies and user surveys, uncovering common limitations such as customisability, inaccurate documentation and lack of versioning tools. The study further explores innovative opportunities to tackle them from these insights, such as a Large Language Model (LLM) powered frontend UI generation functionality, a Plugins-of-Plugins system, an HTML importation system, and an automated documentation generation feature. This study then successfully proves these features' technical feasibility through experiments and implementation by building a proof-of-concept website builder. Additionally, through experiments, it is found that passing the partial design or sketch is a valid approach to make minor modifications to the current HTML codebase, and commercial models outperform open-source models in this task.

Part of the work presented in this thesis has also been accepted at AI4X 2025 (Poster Track), recognising its relevance in website builders and frontend UI generation.

The codebase and datasets are all generally available at **https://boonhianlim.github.io/beyond-code/**,

Acknowledgements

I express my gratitude to Dr Li Fang for her invaluable guidance and support throughout the entire project. This project would not have been possible without her support and understanding. I appreciate all the bits of advice given by her, especially when the project hit a significant roadblock. I would also like to thank Andy and Jasper for the foundational work they did in their studies, which significantly shaped this paper's direction. I am also grateful to the Fun of Growing Group (FOGG) for providing access to their Wix-built web pages for the case study. As always, I would like to thank my family and friends for their emotional support along the journey.

Contents

List of Figures

List of Listings

List of Tables

1. Introduction

1.1. Background

Websites have become a part of our lives. Businesses can create a well-designed front page that provides a suitable identity for the company, helps deliver first-hand information, reflects corporate culture, creates a positive image, and captures customers' attention [1]. Individuals, on the other hand, can build their portfolio with some web pages for exposure. However, building a custom website from scratch can be complicated and requires technical expertise.

To address this challenge, website builders have emerged as a convenient alternative. Website builders provide a platform allowing users to easily create and maintain websites with easy-to-use features such as drag-and-drop editors, templates and various plug-and-play plugins. These functionalities enable users to focus on their business needs instead of worrying about technical complexities [2].

Despite the easy-to-use features of website builders and the large population adopting website builders, the user experience of building a website on such platforms is not necessarily good, such as editor being slow and unstable [3], having high abstractions that restrict customizability [2] and a lack of version control [4]. These all suggest that the development experience with website builders in the existing market could be further improved to allow more parties to consider website builders in their workflow.

1.2. Objective of the Study

This study investigates current website builders' limitations and proposes solutions to these restrictions. We aim to ascertain the limitations through multiple methods, such as case studies and user surveys. We pick a few from these identified limitations as our focus area and propose improvements and new features to address them. We then perform a metric-based evaluation and build these features in a prototype to prove their technical feasibility. At the end of this project, we release the codebase and datasets to foster an open ecosystem that allows interested researchers, developers, or individuals to test the prototype and modify or improve it based on their needs.

1.3. Scope

According to Allison, editor-in-chief of the Wix Blog, a website builder is a “solution that enables you to set up, design, personalise, publish and manage a website without having to use code” [5]. While this means WordPress, a general-purpose Content Management System (CMS), can also be defined as a website builder, it would also be inclusive of various domain-specific CMSs, such as

1. Shopify [6], an e-commerce CMS,
2. Medium [7], a blog-focused CMS and
3. Blackboard by Anthology [8], a Learning Management System (LMS).

As mentioned above, these domain-specific CMSs also fall under the definition. However, these systems do not support creating custom websites, and they only create certain kinds of websites for specific uses. In this project, we would focus on website builders that do not restrict themselves to a specific type of content.

1.4. Key Contributions

The contributions of this paper are as follows:

- **Multi-method Evaluation Framework**. Similar to existing research, we apply multiple approaches to evaluate the website builders in the market, including case studies and user surveys through questionnaires [2], [4], [9]. However, existing works rarely provide validation for their proposed features. We address this gap with experiments and implementation. Specifically, we validate sketch-to-code functionality through experiments with rigorous metric evaluation systems, which we discuss below. We also build a prototype to reveal technical difficulties in building our proposed features in website builders.
- **Partial Design-to-Code/Sketch-to-Code**. To the best of our knowledge, this is the first attempt to explore the feasibility of generating code from users’ minor modifications to the current website through designs or sketches. We argue that this is a better-suited use case for non-technical users. We propose different strategies to tackle such tasks, such as

merging the sketches with the current designs or splitting them so that LLM can evaluate them separately based on established metrics.

- **Dataset Creation and Evaluation**. Building upon the work in WebCode2M [10], we modify their dataset for the partial design-to-code and sketch-to-code task evaluation. We also adopt evaluation metrics proposed in Design2Code [11] and further extend them by introducing the generation success rate to assess the accuracy of LLM-based UI generation. Additionally, we evaluate responsiveness across different viewports by evaluating the generated webpages in the top 50% of most popular viewports for three primary device types: desktop, tablet and mobile.

1.5. Project Background

This project is inherited from Mr Ng Heng Sheng Andy and Mr Yeo Kai Liang Jasper, who have previously built a website in Wix for the external stakeholder, FOGG (Fun of Growing Group). As a continued effort to support FOGG, a part of this project is to implement and maintain FOGG's current website, FOGG.com.sg, a camp booking web application.

As such, this project is split into two parts.

The first part of the project is to explore the current limitations of website builders. As the website is continuously developing, the experience of working with the aforementioned camp booking website is used as a case study of the limitations of the current platform. As the website is built on Wix, we perform market analysis on other website builder platforms to obtain a bigger view. Furthermore, we performed a user survey to obtain their opinions on the pros and cons of website builders. From there on, we propose improvements and features to tackle such restrictions.

The project's second part focuses on evaluating the solutions' technical feasibility. We refer to the metrics proposed in the existing works to evaluate the feature, particularly partial design-to-code and sketch-to-code. We also built a proof-of-concept website builder that included our proposed solutions. Due to time constraints, some solutions, such as the Git-powered version control feature, may be left for future work and remain in the planning stage.

1.6. Project Schedule

This project started in July 2024 and finished in March 2025. This project lasts 10 months with the following schedule:

Starting from July 2024:

- Perform technical analysis on website builders with existing FOGG websites.
    - Take over from the previous FYP students.
        - Understand the current progress of the project.
    - Help FOGG build their current website on Wix.
        - This task is later referred to as our case study.
        - Identify gaps that could be improved within the timeframe of the project.

August 2024 – October 2024:

- Perform an additional literature review of existing website builders' solutions.
    - We study the existing research on website builders and those in the market.
    - Identify gaps that could be improved within the timeframe of the project.

November 2024:

- Propose solutions and draft out implementation details.
- Sketch out the timeline for implementation.

December 2024 – March 2025:

- Prototype development and experiments.

February 2025 – March 2025:

- Compilation of results and final report writing.

In the middle of the project, if the external stakeholder, FOGG, raises any issues or new requirements, additional time would be spent to resolve them. As the required time to resolve the said issues is uncertain, this is not indicated in this timeline.

A Gantt chart is provided below:

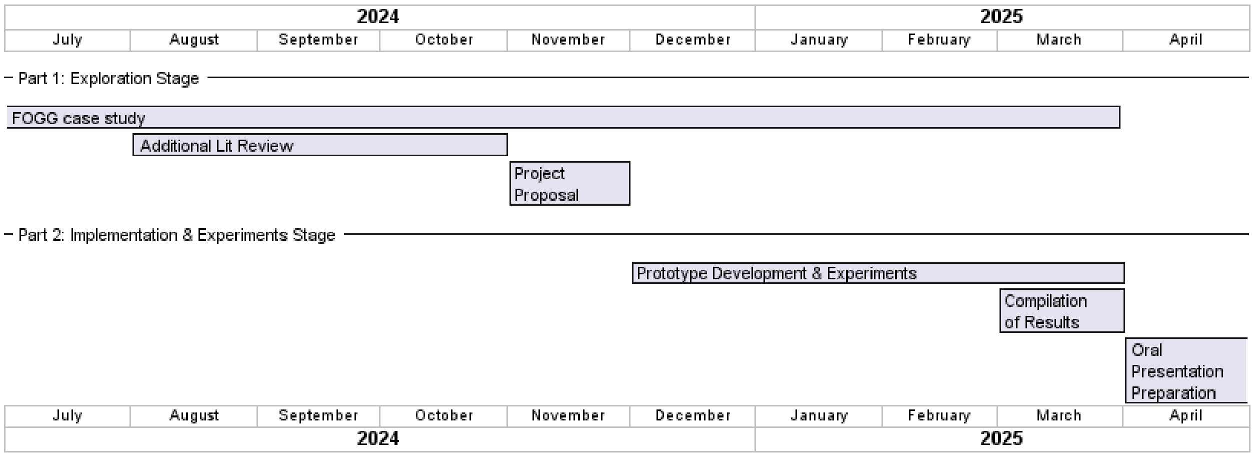


Figure 1: Gantt chart of the project schedule.

2. Literature Review

In this section, we aim to provide an overview of website builders and the foundational technologies relevant to our study. We begin with a market analysis to understand the current landscape of website builders, identifying common features, limitations, and trends. Following that, we review prior studies highlighting gaps in the existing website builders and their relevant technologies, such as static and dynamic websites, adaptive versus responsive web design and frontend UI generation from designs and sketches. By combining insights from both industry and academia, we aim to establish a strong contextual foundation for our proposed work.

2.1. Market Analysis

How does one create a web page? While one may argue that it can be done by just writing the Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS) code, it is often more than that. Jennifer identifies the process of building a successful custom website, which includes designing a well-organised website, performing Search Engine Optimisation (SEO), registering for domain names and handling website hosting. This does not yet include the non-technical soft skills such as using the right messaging and creating easy-to-digest content [12]. Using website builders helps simplify some of this development process, such as setting up payments and helping with hosting, although with some limitations.

In the following paragraphs, we provide an overview of the current website builders in the current market, specifically by analysing their widespread adoption, standard features and common limitations by examining commercially available platforms: WordPress and 10Web AI Builder.

2.1.1. Market Share

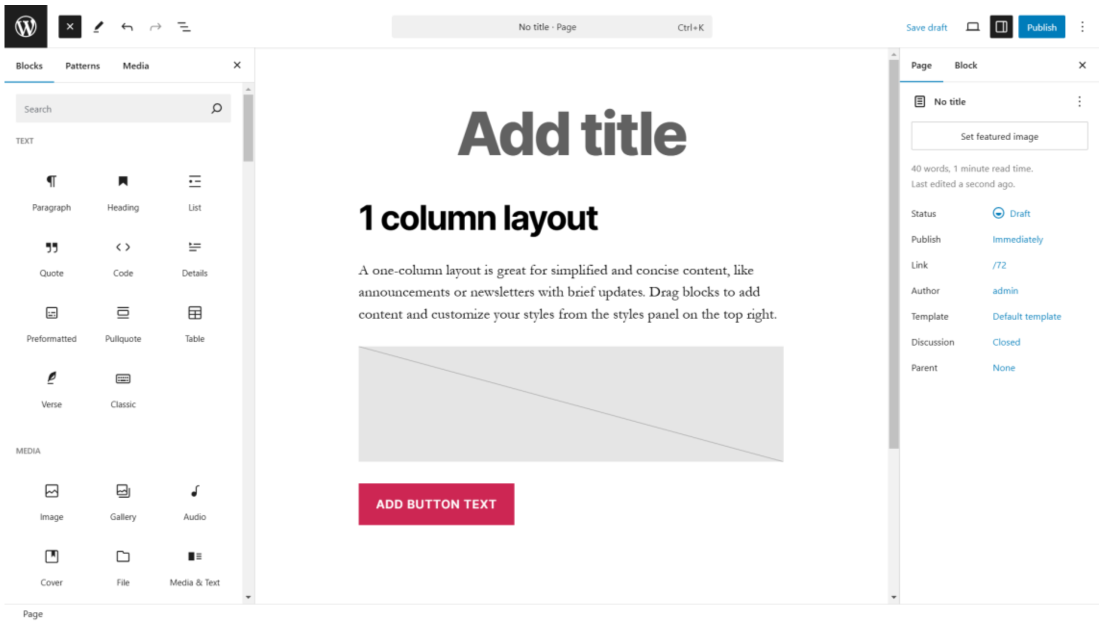
In today's internet, many web pages are built by website builders. W3Tech claims that 40% of the web uses WordPress in one or more subdomains after analysing the top 10 million websites by traffic collated by ceased web traffic analysis company, Alexa Internet [13]. This data does not yet include other website builders. Although so many web pages use website builders and more and more website builders are available in the market now, some specific platforms are still more popular than others. In 2023, Jacqueline Renouard discovered that all the websites built with website builders are mainly created with WordPress, Shopify or Wix. This trend also holds among the top 100,000 websites by traffic [14]. Over the 673 million websites that BuiltWith researched with [15], 33,321,761 (around 4.95%) websites are built with WordPress and 8,359,629 (around 1.24%) websites are built with Wix [16]. Of the top 3 popular website builders, WordPress, Shopify and Wix, WordPress is an open-source CMS in which source code can be obtained from WordPress subversion (SVN) repository or Github [17], while Wix and Shopify are both close-source and can be accessed on their respective official website as a Software-as-a-Service (SaaS) [6], [18].

2.1.2. WordPress

WordPress is an open-source content management system (CMS), released under the GNU General Public License, that enables users to create websites with minimal setup but powerful features. It was created by Mike Little and Matt Mullenweg in 2003. WordPress was originally a blog CMS but has evolved into a general-use CMS. WordPress uses PHP for its application logic and MySQL for its database [19].

WordPress supports many easy-to-use features, allowing users to create their websites easily. A drag-and-drop editor, Gutenberg, has been shipped with WordPress since version 5.0 [20]. Remarkably, WordPress's drag-and-drop editor is a block editor that works like Lego blocks. Each component is a block by itself. Users can only place the components by stacking them on top of or next to each other in the drag-and-drop editor.

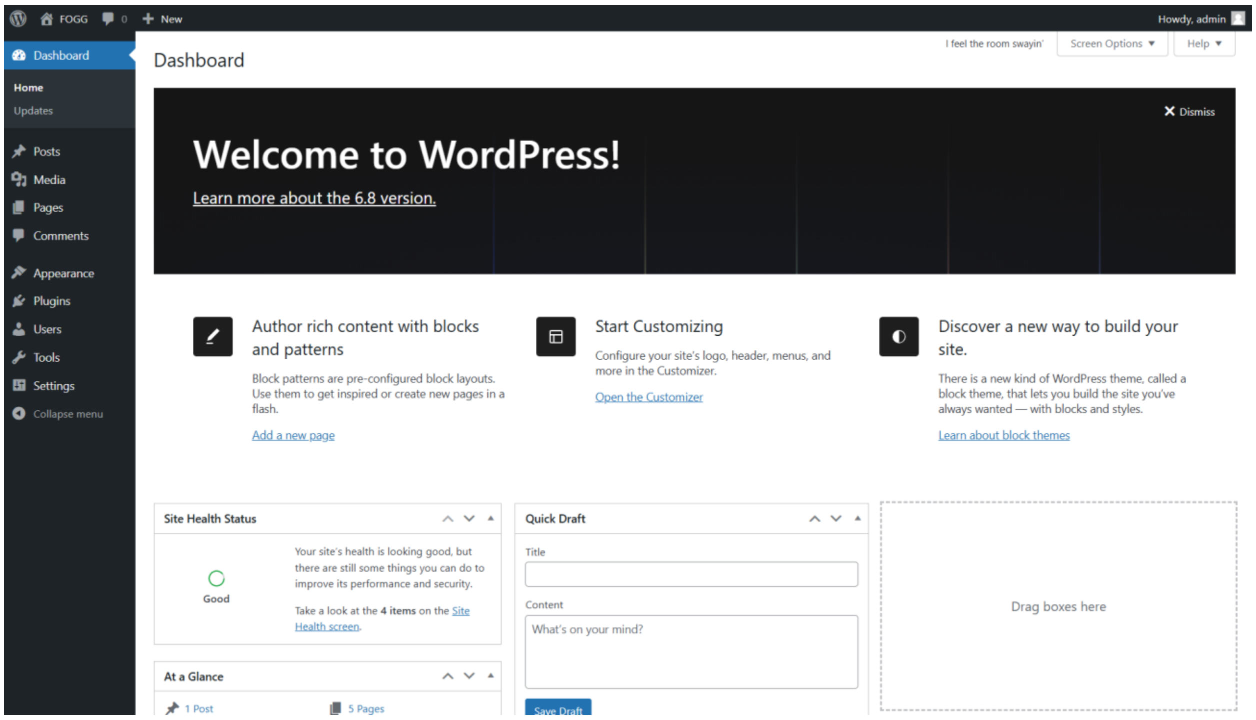
In contrast, an absolute-position editor, like the classic WYSIWYG (What You See Is What You Get) editor, lets us drag and drop the elements anywhere on the page like a drawing tool. The drawback is that the components can get messy and may not follow responsive web design, especially if the component has a fixed length or uses absolute positioning—more on responsive web design in Chapter 2.2.3.



A one-column layout is great for simplified and concise content, like announcements or newsletters with brief updates. Drag blocks to add content and customize your styles from the styles panel on the top right.

40 words, 1 minute read time.
Last edited a second ago.

Figure 2: WordPress drag-and-drop editor (Gutenberg) interface.



Block patterns are pre-configured block layouts. Use them to get inspired or create new pages in a flash.

Configure your site's logo, header, menus, and more in the Customizer.

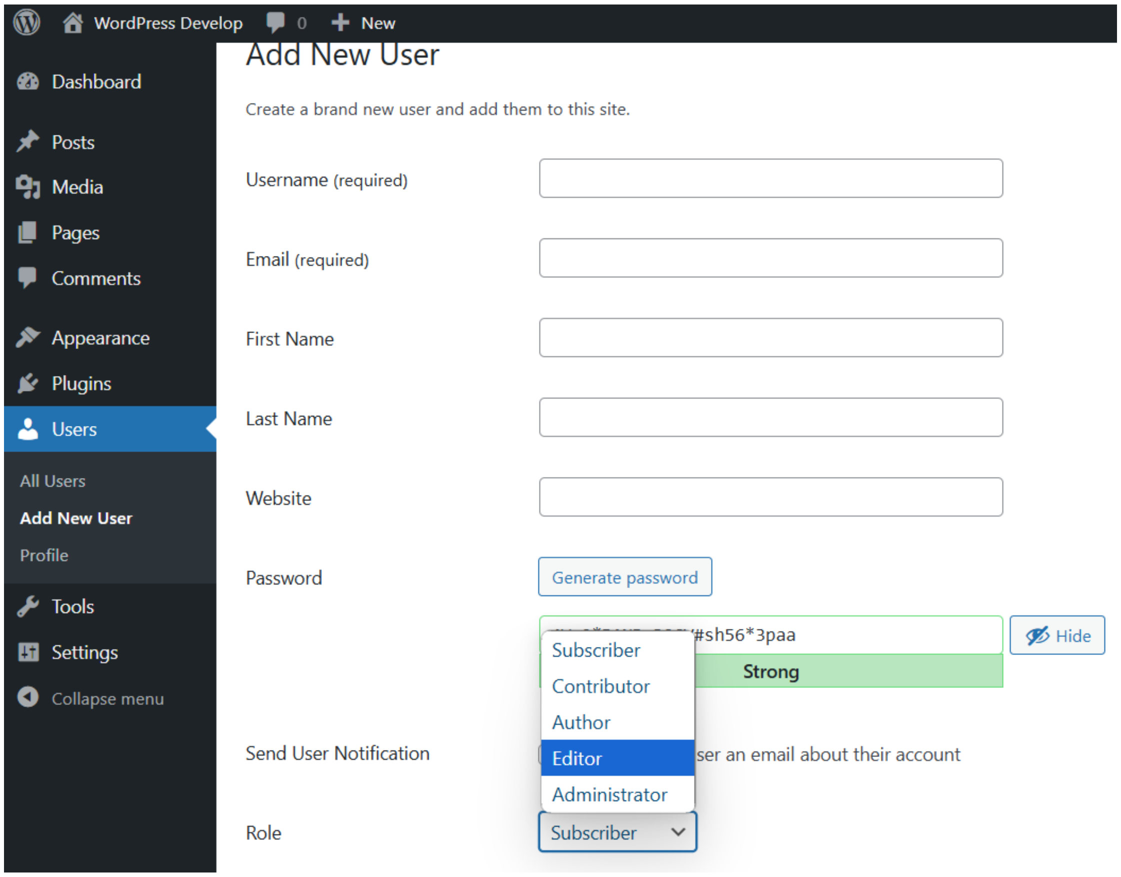
There is a new kind of WordPress theme, called a block theme, that lets you build the site you've always wanted — with blocks and styles.

Your site's health is looking good, but there are still some things you can do to improve its performance and security.

Take a look at the **4 items** on the Site Health screen.

Figure 3: WordPress dashboard interface.

Moreover, WordPress has an admin dashboard consisting of several menus for different functionalities. For example, the media menu allows users to quickly modify the assets, such as images and videos available on the web pages, and the page menu allows users to add new pages easily. It is also possible to add new users or admins through the user menu, where access and permissions are given to different users at a coarse-grained level [21].



Add New User

Create a brand new user and add them to this site.

Username (required)

Email (required)

First Name

Last Name

Website

Password: #sh56*3paa
Strong

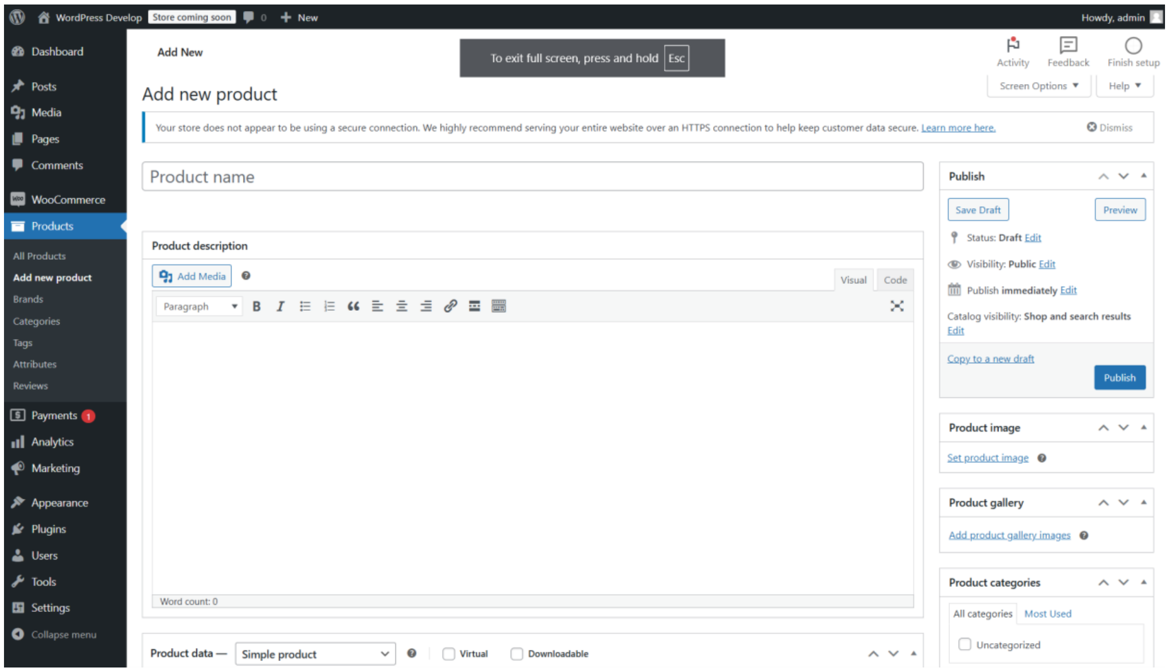
Send User Notification: ser an email about their account

Role: Subscriber

Figure 4: WordPress user role dashboard, where admins can add new users and assign different roles to them.

While users can set up WordPress websites from the source code and host them themselves, this could be challenging for those without hosting experience. Luckily, due to its open-source nature, many hosting services such as WordPress.com by Automattic [22], WP Engine [23], Hostinger [24] and Amazon LightSail [25] support deploying WordPress websites with their services. With these services, users without experience deploying websites can host WordPress websites easily. Data shows that just the WP Engine itself has served around 1 million websites [26],[27].

Another advantage of WordPress is that it allows customisation through plugins. The motivation for having plugins is to allow modification of WordPress without touching the core WordPress files, which will be overwritten during each update [28]. Thanks to its open-source nature, it has a vast and active plugin ecosystem that allows users to implement many features without manual programming, instead allowing users to plug and play. For instance, WooCommerce, a plugin made by Automattic, can transform WordPress into an e-commerce website. It allows users to update their products through the dashboard and integrates with 100+ payment gateways to accept various payment methods.



Your store does not appear to be using a secure connection. We highly recommend serving your entire website over an HTTPS connection to help keep customer data secure. Learn more here.

Figure 5: WordPress WooCommerce products dashboard.

Thanks to the active plugin ecosystems, it is also easy to integrate cutting-edge technologies into the websites. Media File Renamer claims to optimise SEOs by providing titles, alt texts and descriptions to the media files with LLM [29], while AI Engine provides chatbot and AI translation features with a broad range of numerous different LLMs such as ChatGPT-4 and Gemini [30].

Since WordPress is released under GNU, plugins can be considered derivative work from WordPress and hence should follow the license and be released under GNU. However, plugins can still have split licences and ask for charges for some of their features by splitting the base and premium features to be viewed as separate programs, though not explicitly allowed by WordPress. For example, while WooCommerce's basic features are free, users must pay a subscription to unlock some of its premium features, such as appointment booking [31] and monthly subscriptions

[32]. Hence, building a WordPress website may not be as cheap as it seems, as website owners may have to pay for multiple plugins for their business use case.

Moreover, since WordPress was originally a blog-focused CMS, its basic functionality is heavily blog-focused. For example, WordPress web pages are coupled with WordPress themes, so working with them requires some knowledge of WordPress themes. WordPress websites are also dynamic by default since a lot of its native features, such as users, blogs and comments, assume there exists a connection to the database. Hence, it is not straightforward for users to shift their WordPress websites to another website builder or the raw HTML code, but thanks to its open-source nature, there are existing solutions in the market that claim to help with this process [33], [34].

Needless to say, with customisation comes complexity. Hence, it is no wonder WordPress has a higher learning curve than other website builders like Wix. The first step for a non-technical person to start developing on WordPress is most likely choosing a hosting service, since they have no idea of self-hosting, which can be confusing, considering there are many hosting services in the market. Moreover, they would also have to handle the installation of plugins to enable extra functionalities on WordPress, and each hosting service may have slight differences in how they handle it, such as whether importing external plugins is allowed. A paywall may also exist to enable certain WordPress features. On the other hand, some other website builder solutions are built with hosting functionalities and are well-prepared with customisation tools such as dashboards and plugins without the need for setup or customisation, notably Wix, which we explore in this paper later.

2.1.3. 10Web AI Builder

WordPress is a versatile open-source website builder solution. Since the source code is generally available, building on top of it to create some other different website builders is possible. 10Web AI Builder is a website builder that extends on WordPress and allows the generation of WordPress websites with AI through user input.

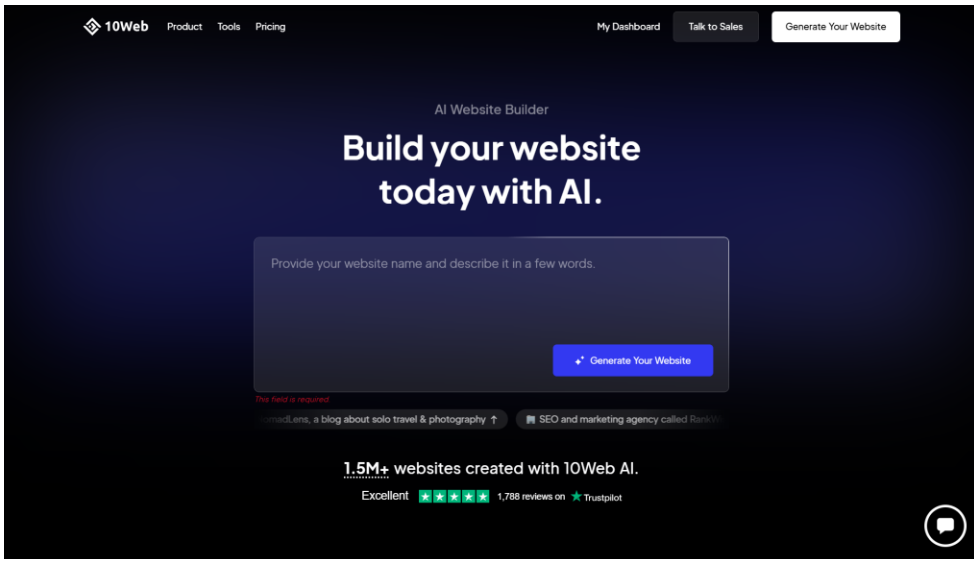
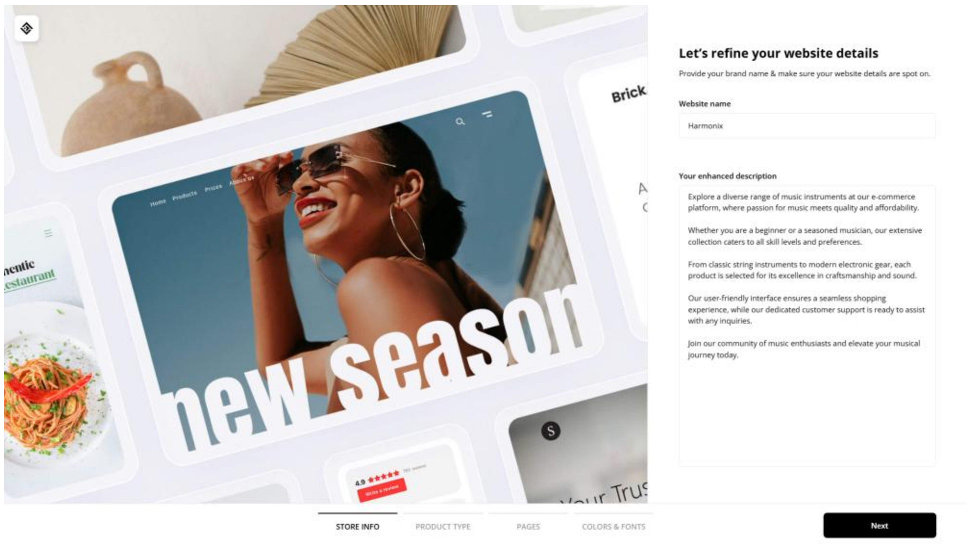

10Web
Product
Tools
Pricing
My Dashboard
Talk to Sales
Generate Your Website
AI Website Builder
Build your website today with AI.
Provide your website name and describe it in a few words.
Generate Your Website
This field is required
NomadLens, a blog about solo travel & photography
SEO and marketing agency called RankW
1.5M+ websites created with 10Web AI.
Excellent 1,788 reviews on Trustpilot

Figure 6: A screenshot of 10Web AI Website Builder.

First, users are prompted to describe the websites they want to create and insert their requirements about their websites. Then, the description provided by the user is enhanced with AI.


Let's refine your website details

Provide your brand name & make sure your website details are spot on.

Website name: Harmonix

Your enhanced description:

Explore a diverse range of music instruments at our e-commerce platform, where passion for music meets quality and affordability.

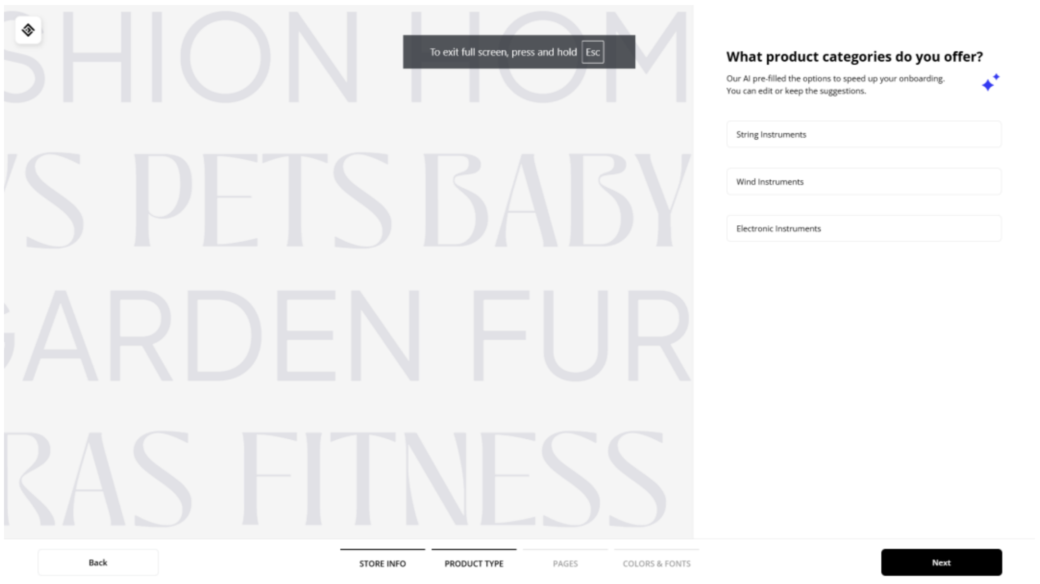
Whether you are a beginner or a seasoned musician, our extensive collection caters to all skill levels and preferences.

From classic string instruments to modern electronic gear, each product is selected for its excellence in craftsmanship and sound.

Our user-friendly interface ensures a seamless shopping experience, while our dedicated customer support is ready to assist with any inquiries.

Join our community of music enthusiasts and elevate your musical journey today.

Figure 7: Refined user prompt on 10Web AI Builder.



What product categories do you offer?

Our AI pre-filled the options to speed up your onboarding. You can edit or keep the suggestions.

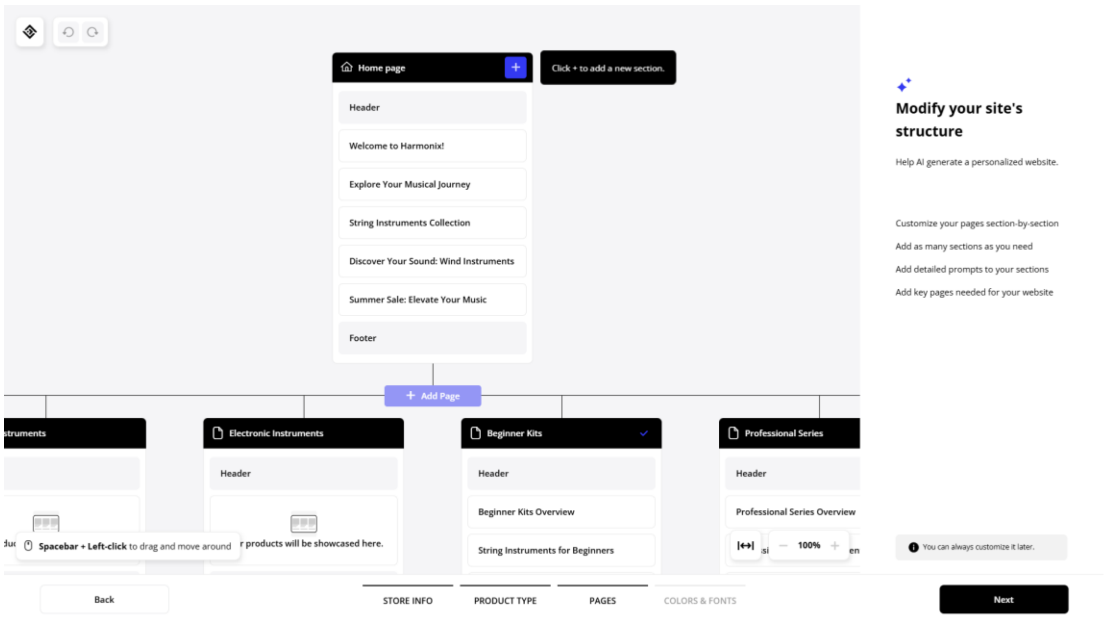
String Instruments

Wind Instruments

Electronic Instruments

Figure 8: User input product categories on 10Web AI Builder.

Users are then prompted to input the product categories. Then, users can see how the websites are generated and modify the structure.


Home page
Click + to add a new section.
Header
Welcome to Harmonix!
Explore Your Musical Journey
String Instruments Collection
Discover Your Sound: Wind Instruments
Summer Sale: Elevate Your Music
Footer
Add Page
Electronic Instruments
Beginner Kits
Professional Series
Header
Beginner Kits Overview
String Instruments for Beginners
Professional Series Overview
Spacebar + Left-click to drag and move around
100%
You can always customize it later.
Back
STORE INFO
PRODUCT TYPE
PAGES
COLORS & FONTS
Next

Modify your site's structure

Help AI generate a personalized website.

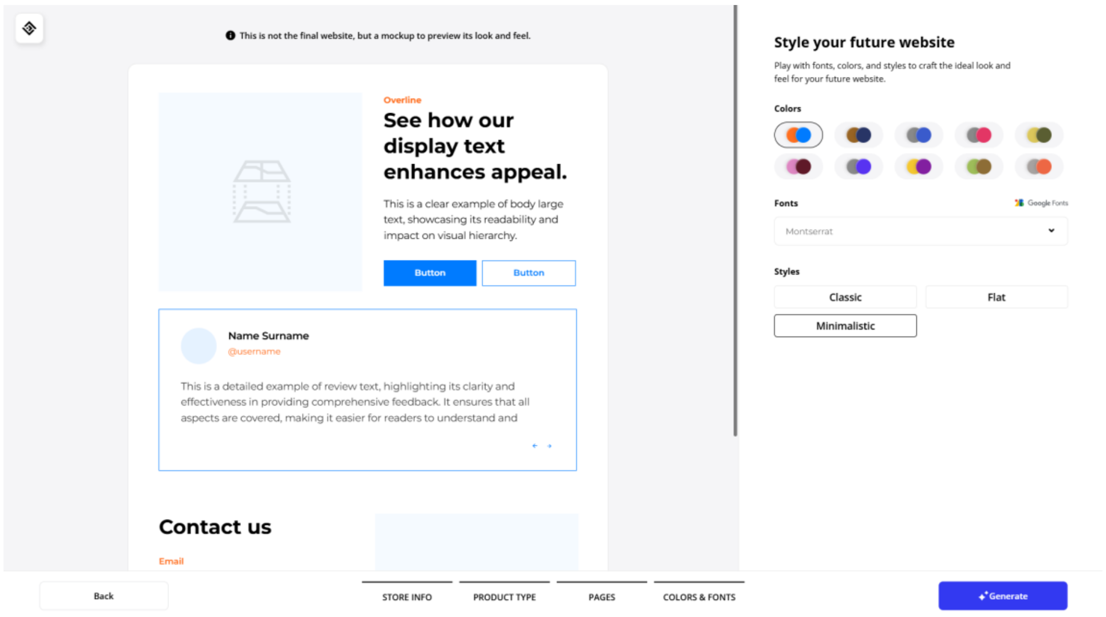
Customize your pages section-by-section

Add as many sections as you need

Add detailed prompts to your sections

Add key pages needed for your website

Figure 9: Generating website structure on 10Web AI Builder.


This is not the final website, but a mockup to preview its look and feel.

Overline

See how our display text enhances appeal.

This is a clear example of body large text, showcasing its readability and impact on visual hierarchy.

Name Surname

@username

This is a detailed example of review text, highlighting its clarity and effectiveness in providing comprehensive feedback. It ensures that all aspects are covered, making it easier for readers to understand and

Contact us

Email

Style your future website

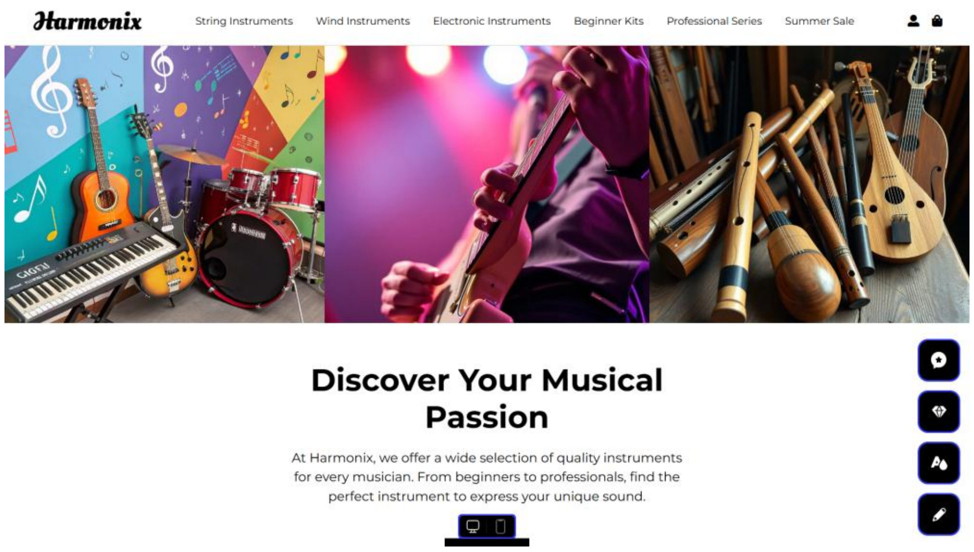
Play with fonts, colors, and styles to craft the ideal look and feel for your future website.

Fonts: Montserrat

Styles: Minimalistic

Figure 10: Apply styling on 10Web AI Builder.

After fixing the web page structure, the user can apply the desired styling. The web pages will then be generated according to the structure and styling. The user can click the bottom-right button to enter the drag-and-drop editor to modify the generated web pages.


Discover Your Musical Passion

At Harmonix, we offer a wide selection of quality instruments for every musician. From beginners to professionals, find the perfect instrument to express your unique sound.

Figure 11: Generated web page on 10Web AI Builder.

Due to its proprietary nature, we can only analyse its functionality from its appearance. We suspect 10WebAI sends the user request to some LLM, which matches the user request to some pre-defined templates. Based on the selected templates, LLM then replaces the placeholder text and images with suitable text and images based on the context. Afterwards, the user can choose which WordPress style to apply to the generated web pages, as shown in the last figure. We assume these features are implemented through plugins with the split licenses technique discussed above in the WordPress section, since the 10Web AI Builder’s source code is not generally available to the public.

By generating the website following the user's natural language input, it is undoubtedly much easier for the users to create websites without any technical expertise, as they only need to describe their requirements well enough. Even if not, they can adjust the prompt in the middle. In the end, the user can also preview the websites and use a drag-and-drop editor, so it would be fine even if the generated web pages go slightly off the users’ expectations as they can always fix it themselves.

It has similar limitations as WordPress, as the underlying website builder is still WordPress. On top of that, some of the features are hidden behind a paywall, such as modifying the elements in drag-and-drop editors, completing e-commerce integrations or adding additional plugins.

2.2. Research Findings

2.2.1. Website Builder

Existing literature justifies the usability of website builders through two methods: directly evaluating the website builder as a tool or assessing the website created by it. For instance, Forbes evaluated website builders and argued that they speed up the development, especially those with AI integration, and relatively low cost as users can Do-It-Yourself (DIY) the webpage instead of hiring experienced developers [9]. Halim et al. involved 41 WordPress developers and identified the features and plugins they rely on to create websites [35], showing how developers use website builders for E-commerce development. On the other hand, Kusumo examined the websites built with three website builders, Wix, Weebly.com and Ecwid.com, with five usability metrics: accuracy of task completed, time to complete task, user satisfaction, ease of use and user confusion. The study supports the usability of websites built with website builders, but such websites should adhere to design principles such as avoiding complex design and ensuring user-friendliness, such as readability [36].

Prior studies have identified limitations of website builders through user studies or hands-on development. Oswal and Oswal identified the lack of accessibility in three website builders, Dorik, Relume and Wix [37]. Palmer and Oswal further argued that the websites generated by website builders lack considerations of disability as well [38]. Andy argued that the abstraction of website builders hinders customizability and suggested building simple features on website builders and having complex logic on custom-coded websites [2]. Jasper supported Andy's findings after his research on Wix, highlighting a lack of collaborative support in the drag-and-drop editor, poor editor performance when more web pages are created, and the limited version control offered by the website builder. He further identified vendor lock-in issues, requiring complicated refactoring to leave the website builders' platform. One example is that it would be challenging to shift from Wix to Weebly, a proprietary website builder, to another proprietary website builder [4].

Improvements have been discussed with some proposed new general or domain-specific website builders to improve these limitations. Domain-specific website builders often exploit the fact that the websites generated will have some fixed layouts or styles and generate them through some pre-defined formatted template. For instance, Jiang et al. presented using XHTML as an intermediate format to allow quick generation, modification, and management of e-commerce websites [39]. Meanwhile, Sutchenkov and Tikhonov developed a static site generator for online courses, using Excel spreadsheets to define the content structure and automatically assemble the learning materials such as videos and PDFs. Notably, they also introduced server-side rendering to improve SEO [40]. As for general-purpose website builders, Mamatha SK et al. suggested creating mobile-friendly editors and integrating an AI-powered recommendation engine to enhance user experience. However, some features proposed in the study, such as having a dynamic CMS and supporting concurrent editing by multiple users, are present in mainstream builders [41].

2.2.2. Static and Dynamic Websites

In Petersen's study, he pointed out two types of websites: static and dynamic. In his terms, static websites consist of pages loaded from persistent storage (such as a hard disk) without server interaction. Though not explicitly stated, client-side interaction is also allowed on static web pages, such as using JavaScript to build a calculator which runs on clients' browsers. In contrast, dynamic websites communicate with servers to perform some actions based on client requests. Auto-generated web pages from statically stored information are also dynamic websites [42].

In the report, Peterson suggested that plenty of sites provide little to no dynamic functionality yet utilise a dynamic website builder such as WordPress instead of a static website builder, making some supposedly static web pages unnecessarily dynamic. The author argued that website owners may choose dynamic website builders over static ones because they have dynamic user-input features and a richer toolset. However, some of these features, such as templates and admin interface, are also possible with static website builders. Moreover, static web pages can provide extra benefits to the website owner, require less maintenance, are easy to scale, have a lower page load time, and have high availability and low security risk. Hence, the author suggests that the website stay static and add dynamic features as needed [42].

Peterson also described an ideal developer-friendly website builder that should appeal to developers and users with limited technical expertise. The author also suggested that a static website builder can refer to the features of their dynamic counterparts and improve upon them [42].

2.2.3. Adaptive Web Design and Responsive Web Design

In today's world, people access web pages from different devices. Hence, to adapt to the different screen sizes of different devices, web pages need the capability to resize and reorder their elements to fit content within the screen. There are two main approaches: Adaptive Web Design (AWD) and Responsive Web Design (RWD). Their definitions are ambiguous and used interchangeably sometimes. Rathore and Singhal define AWD as "uses different website layouts and deploys the most suitable option based upon the device dimensions" [43, p.3] and RWD as a "dynamic approach … where user interfaces change with the screen resolution" [43, p.2] On the other hand, Yousaf et al. defined AWD as a "uniform representation of the site over any intended medium" and RWD as "advanced navigation and easy reading with a minimum of panning, resizing and scrolling and interactive over a multitude of different devices to enhance accessibility, ease of use and general adaptation of crossing limitations to providing better interface" [44, p.1]. In this report, we would follow Rathore and Singhal definition where AWD means having different version of websites for different dimensions and RWD means having one fixed version of website which can adapt to different dimensions.

AWD and RWD each have their advantages and disadvantages. For example, AWD provides faster load times as developers can choose to deliver specific assets for users with specific screen sizes, such as displaying the banner only when the user is on desktop and a button when the user is on mobile. However, it also increases the maintenance required since there are multiple versions of websites to be maintained, and the SEO will be worse if the layouts among the versions are inconsistent. On the other hand, RWD would be easier to maintain and have a better SEO, but it would add complexities to the development process as developers now have to make sure the fixed design looks good regardless of the screen size [43].

Both AWD and RWD are not foreign concepts used by website builders. Bekmanova et al. proposed a website builder that allows users with limited technical expertise to build adaptive websites which render user-created content dynamically upon request. They also utilise server-side rendering to achieve better SEO in adaptive websites [45]. On the other hand, Wix supports RWD by allowing elements such as columns, full-width slideshows, galleries and horizontal menus to dynamically adjust elements to adhere to the screen size [46].

2.2.4. Frontend User Interface (UI) Generation

Frontend UI generation is a heavily studied topic. The first few pioneer research applied computer vision techniques for frontend UI generation, as demonstrated in Landay and Myers' SILK [47] and Seifert et al.'s Mobidev [48]. However, as discussed in Robinson's paper, such techniques require programmer judgement when adding new elements into the systems and are prone to user error as the sketch may not precisely reflect the user requirements [49].

More recently, due to advancements in machine learning, researchers have attempted to apply it to UI generation. There are two main research directions: low-fidelity sketches and high-fidelity designs. Robinson tackled low-fidelity sketches using Artificial Neural Networks (ANN) to translate sketches into normalised images before translating them into code [49]. Jain et al. first detected the components with RetinaNet, resolved overlapping components and generated code from the detected components [50]. On the other hand, Soselia et al. used a vision-code transformer to encode images, which GPT-2 consumes to generate code. Due to the similarity of the sketch-to-code and design-to-code, similar techniques or models can be applied for both tasks. For example, Pix2Struct [51], a vision model pre-trained to convert webpages' screenshots into HTML, can be fine-tuned to convert sketches to code, as demonstrated in WebCode2M [10].

The latest studies often exploit the vision Large Language Models (VLLMs) vision capability to generate HTML directly instead. Design2Code by Si et al. focused on generating HTML and CSS code from screenshots [11], while Li et al. extended it with low-fidelity sketches [52]. Design2Code proposed that VLLMs can reliably generate frontend UI on simple websites, but they may struggle in complex websites. Sketch2Code found that UI generation from low-fidelity sketches is challenging to VLLMs, but the quality of the generated websites can be improved with multi-turn prompting.

3. Methodology

3.1. Project Methodology

Through literature reviews, we notice website builders' limitations and potential improvements. However, considering the lack of study on website builders, more limitations might hinder its adaptation that has not yet been found in previous research. Previous studies rarely attempt to implement their proposed solutions to verify their technical feasibility or usability.

In this thesis, we propose to identify the limitations of website builders and provide solutions. While previous studies have found some limitations regarding website builders, it is not reasonable to address all limitations and verify improvements for all of them due to resource constraints. Hence, we first identified the most urgent areas for improvement and proposed solutions. We achieve this through case study, user survey, metric evaluation and implementation.

As shown in studies done by Andy and Jasper [2], [4], a user study is proven to be a valid approach to identifying limitations in website builders. We take over the websites built in Wix by Andy and Jasper for Fun Of Growing Group (FOGG), a company that designs learning programmes for children aged 5 to 16 and above [53]. As a small and medium enterprise (SME), FOGG relies on its websites to allow users to book their programmes and make payments online. The user study is conducted by implementing proposed features from FOGG, all of which are real business use cases in their daily operations. Through this user study, we plan to reflect on users' struggles using website builders.

As observed by Almazroi, questionnaires, or user surveys, are commonly used to evaluate software usability [54]. Similarly, we perform a user survey to obtain a general view of why users may use website builders and if they ever face limitations when using website builders.

We pick a few that require immediate attention from these identified limitations from the case study and user survey and propose some solutions to tackle them. These features may be immature, experimental or not commonly used; hence, we attempt to prove their usability and feasibility through two methods. For the experimental feature, which is sketch-to-code in this thesis, we prove its usability through metric evaluation to ascertain that it is ready for integration in website builders.

For the uncommon features, we implement a proof-of-concept (POC) website builder containing some of these features to prove that they are technically possible. Moreover, further user study can be performed later with this POC website builder to identify if these features successfully tackle the limitations.

At the end of this project, we release all datasets, the experiment codebase, and the POC website builder to allow researchers, developers, or interested individuals to develop further on top of these ideas and features.

4. Problem Discovery

4.1. Case Study: FOGG on Wix

As a learning service provider, FOGG built its website with Wix to allow interested students to join its programmes through online booking.

4.1.1. Switching Domain

In Wix, one website corresponds to one project. One website can have multiple pages controlled by the same dashboard. FOGG had two websites, one for development by contractors and one for official use. Since the contractor had implemented some new features, FOGG wanted to change their domain from the current official website to the contractor's one instead.

All Wix websites receive a free domain, but users can connect their websites to a custom domain if they subscribe to the Wix Premium Plan. Moreover, users can also transfer their domains between the websites they have created. Using this native feature, we effortlessly switched the domain from the previous official website to the website developed by the contractor.

4.1.2. Changing Default Language

FOGG used Wix Multilingual [55] to support displaying different languages on its web pages. FOGG realised the default language is English, while FOGG preferred Chinese as the default language. However, we found that the default language is unmodifiable after initialisation. The only way to modify it was to remove the plugin and reinstall it again, which caused all the existing localisation to be lost [56]. The only remedy was duplicating a current site as a reference, then removing the plugin and manually copying the localisation one by one for every component on the web pages from the duplicated site. Luckily, most current web pages were not localised yet, but we could foresee the pain of copy-pasting the translation in established Wix websites.

4.1.3. Localisation of Program Information

FOGG relied on the Wix Booking Plugin [57] to manage all programmes. With this plugin, FOGG could easily modify the programme details such as name, tagline, description, locations, and photos in the dashboard. FOGG could also easily adjust the pricing and payment method through the dashboard. However, as of March 2025, the plugin does not support localisation natively. Users are only given one textbox to fill in the program details. Hence, if users want to provide details in multiple languages, they must input them inside the exact textbox and display all of them on the web pages.

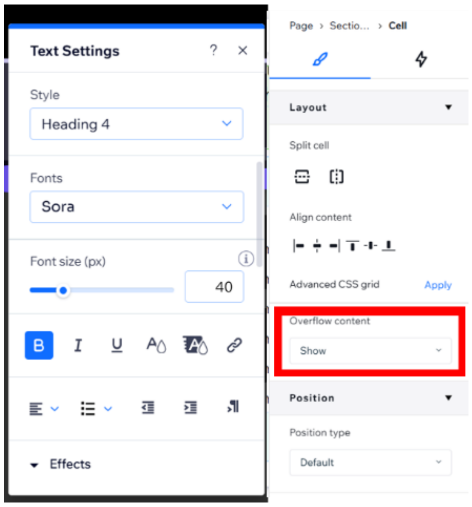
We remedied this by having custom logic on the web pages that parsed the details and only displayed the corresponding details based on the selected language. This implementation was possible thanks to Wix Velo, a full-stack development platform that allowed developers to build both frontend and backend JavaScript code, which was injected and executed during web page load time [58],[59].

However, this had a drawback: users must input the details in a specific format for the parser to work correctly. Moreover, the textbox provided in the dashboard is too small and limited to filling in all the details. Hence, we let the users key in the programme details in multiple languages in Excel. The user could then execute a VBA script to convert the details into a valid JSON object that the custom parser understood. This modification provided an easy-to-use interface for non-technical users to modify the program details without hassle.

4.1.4. Revamp of the Home Page

FOGG wanted to include a programme recommendation section, current user feedback and more FOGG-related information, such as a general description and years of experience in the industry on the home page. We designed the new user interface on Figma, a commonly used tool for UI design, and implemented it on the websites with the built-in drag-and-drop editor.

As of March 2025, Wix supports two drag-and-drop editors, Wix Editor and Wix Studio, where Wix Editor targets hobbyists and small businesses, and Wix Studio targets professionals and enterprises. Wix Editor is more “beginner-friendly” as it hides some complexity and only displays some basic properties of the HTML elements. Instead, Wix Studio provides more flexibility to developers by allowing custom CSS and supporting responsive websites [60]. For example, a text element in Wix Editor only shows basic settings such as text size and style, but in Wix Studio, we can adjust how the text behaves when it overflows. However, switching from Wix Editor to Wix Studio is impossible, and the only way to migrate is to manually recreate the websites in Wix Studio [62].


Text Settings
Style: Heading 4
Fonts: Sora
Font size (px): 40

Page › Sectio... › Cell
Layout
Split cell
Align content
Advanced CSS grid: Apply
Overflow content: Show
Position
Position type: Default

Figure 12: Wix Editor text settings (left) and Wix Studio text settings (right). Overflow content only exists as an option in Wix Studio.

Unfortunately, some designs could not be implemented since the website was initially built with Wix Editor. For instance, we planned to introduce a repeater with a horizontal scroll bar for the recommended programs. However, such a scroll bar was only available as a drag-and-drop component in Wix Studio, not Wix Editor. Hence, we replaced the scroll bar with buttons in our new design. Ultimately, FOGG stepped back by removing the buttons and only keeping three recommendations on their main page.

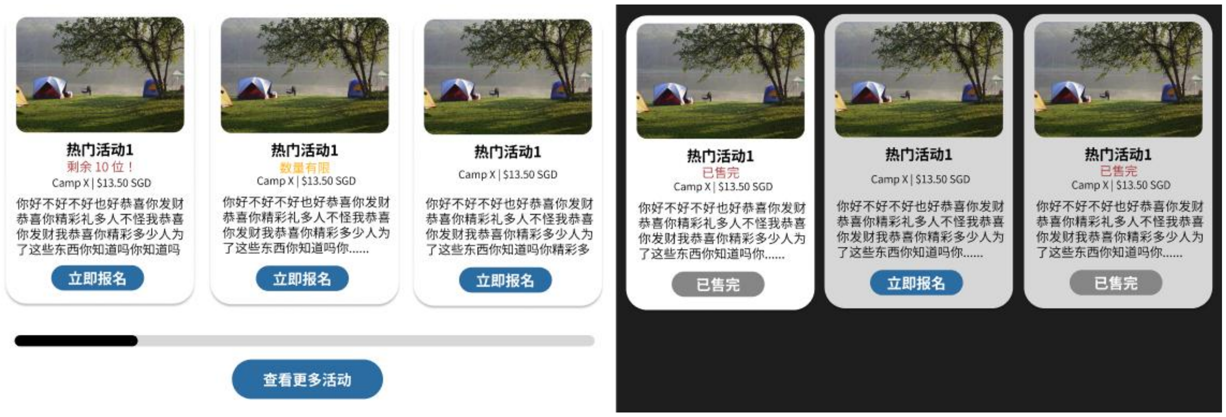


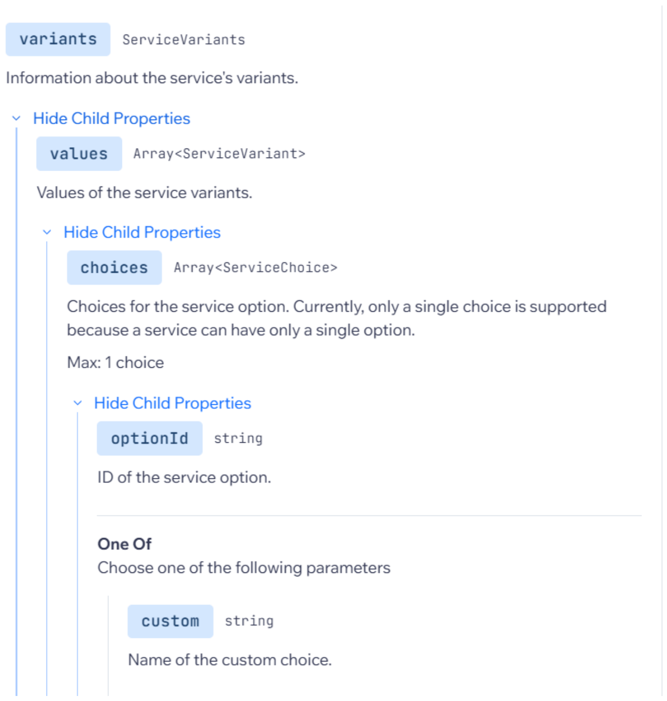
Figure 13: The original intended design for the recommendation repeater in FOGG’s Main Page.

To include programme recommendations, we applied Wix Bookings Velo API to search for any programme with a name ending with (推荐), meaning recommendation in Chinese, and displayed their services on the column repeater. FOGG could then change the recommendations displayed on the main page by modifying the programme name from the Wix Booking Dashboard. We left the user feedback hardcoded on the main page, intending to link it up with comments on Google Maps or some other CMS in the future. We put the revamped web page design in Appendix A for reference.

4.1.5. Revamp of Program Details Page

Previously, FOGG only displayed the program name, a banner photo, the pricing and a general description on the program details page. They requested a new design that could include more information, such as teachers' information, program timetables, a photo gallery and the address where FOGG hosted the programme. Moreover, they would like to modify the booking form components linked to the program details page by having a calendar to display the programme date and time and adding the option to buy T-shirts for outdoor programmes.

We added localisation to the program details page through the custom description parser discussed in section 4.1.3. We also extended the parser functionality to parse the correct teacher information, program timetables and addresses from the program description. Then, we implemented the photo gallery features using the experimental Wix Bookings V2 API [61]. The problem with such an experimental API is that it is prone to breaking changes without direct notifications, and the documentation is erroneous and may not match its implementation. For example, the getServiceOptionsAndVariantsByServiceId function in Wix Bookings V2 API is supposed to return an array of strings as shown in their documentation, but instead, the field merely consists of a string called "[Object]". Considering this is an experimental API, this may be an implementation error or an upcoming unfinished feature. Regardless, we cannot rely on such APIs for implementation as there are no apparent signs of when the API will be fixed, become stable and become generally available other than a notice mentioning that an API will stay experimental for a maximum of 6 months [62]. However, since this experimental API was the only available approach we identified to fetch the photos set in the Wix Booking Plugin Dashboard, we had to apply error-catching, where we would not display photos if the API response was erroneous.


variants ServiceVariants

Information about the service's variants.

Hide Child Properties

values Array<ServiceVariant>

Values of the service variants.

Hide Child Properties

choices Array<ServiceChoice>

Choices for the service option. Currently, only a single choice is supported because a service can have only a single option.

Max: 1 choice

Hide Child Properties

optionId string

ID of the service option.

One Of

Choose one of the following parameters

custom string

Name of the custom choice.

Figure 14: A screenshot of the documentation for the getServiceOptionsAndVariantsByServiceId function in Wix Booking V2 API regarding the data type of the "choices" attribute.

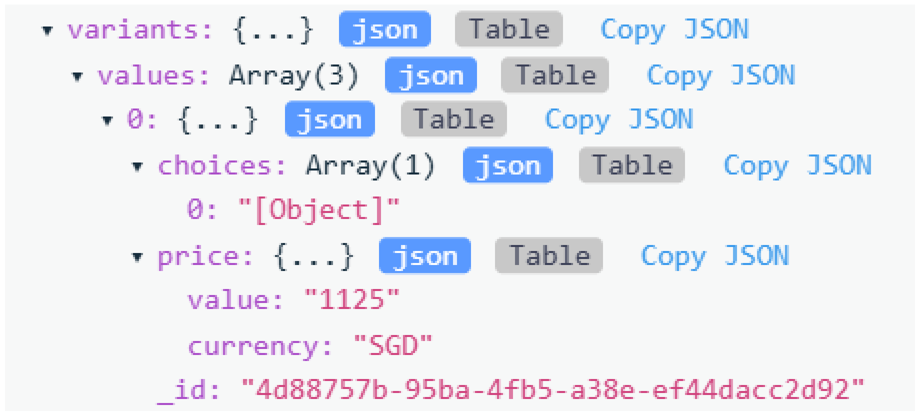

```
▾ variants: {...}  json  Table  Copy JSON
  ▾ values: Array(3)  json  Table  Copy JSON
    ▾ 0: {...}  json  Table  Copy JSON
      ▾ choices: Array(1)  json  Table  Copy JSON
          0: "[Object]"
      ▾ price: {...}  json  Table  Copy JSON
          value: "1125"
          currency: "SGD"
        _id: "4d88757b-95ba-4fb5-a38e-ef44dacc2d92"
```

Figure 15: A screenshot of the JSON response returned by the getServiceOptionsAndVariantsByServiceId function in Wix Booking V2 API.

We also had to use custom elements [63] to implement the calendar in the booking portion, as no existing calendar components in Wix suit our use case. The calendar should be in a monthly format, and the availability of the programme should be displayed on all days of the month in different colours. We achieved this by using a pre-existing calendar template in raw HTML and CSS, then modularising it as a custom element and extending its functionalities to display availabilities with the callbacks. Notably, we used connectedCallback() and attributeChangeCallback() [64], which allow us to render date and availability dynamically when users change the selected month and year or the primary language. One example is when the year or month changes, the calendar triggers a custom event, which a Wix Velo function listens to and fetches the correct availability for that selected month. We then send this availability info to the calendar by setting it as the new attribute and triggering the update through attributeChangeCallback() to refresh the calendar.

Regarding the booking form, we adjusted it to ask users whether they have a T-shirt before proceeding to payment. However, we found that adding this price to the final payment for users to checkout is impossible. The payment functionality is limited to a fixed price; the only modification allowed was whether users provided a promotion code, which reduced the price by a fixed amount or some percentage. These limitations also appeared later in the development, as FOGG proposed having varied pricing for some of their recurring programmes. However, despite the Wix Bookings Plugin containing such functionality to set varied pricing for a program, its API does not support varied pricing for online payment [61]. As we had to go through a custom booking flow and asked users to make payments through the API, we could only implement a fixed price for all programs instead until we created a custom plugin to handle online payment.

4.1.6. Other Findings

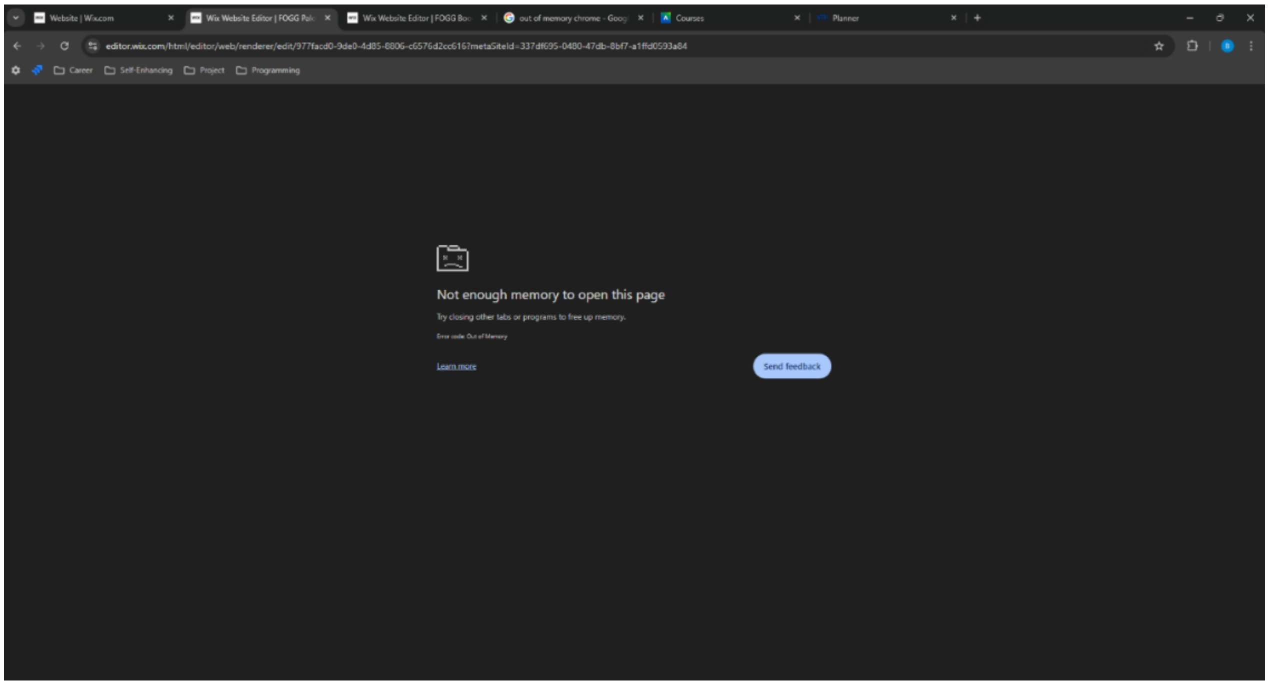
4.1.6.1. AI Adaptation

Wix has embraced AI as one of its tools. With Wix ADI, users can create a website by talking with a chatbot, which will analyse the user requirements based on the input [65]. The created websites are similar to a template but populated with mock details. Users can then adjust the website (photos or texts) according to their needs in the editor. Users can also further generate text or images with text input in the editor with Wix ADI.

4.1.6.2. Wix Editor Performance Issues

Wix Editor is sometimes slow and even crashes on its own sometimes, and it has led to some complaints from unhappy customers [3],[66],[67]. We also experienced this problem during implementation, as shown in Figure 16. In fact, in one of the articles in Wix's Help Centre, Editor

Performance: Best Practices, it is mentioned that editors in Wix would load all web pages on the website at once [68], which may be the culprit why the Wix Editor gets slower and slower as the website contains more pages and elements and eventually crashes on its own.


Website | Wix.com
Wix Website Editor | FOGG Pal
Wix Website Editor | FOGG Boo
out of memory chrome - Goog
Courses
Planner
editor.wix.com/html/editor/web/renderer/edit/977facd0-9de0-4d85-8806-c6576d2cc616?metaSiteId=337df695-0480-47db-8bf7-a1ffd0593a84
Career
Self-Enhancing
Project
Programming
Learn more
Send feedback

Not enough memory to open this page

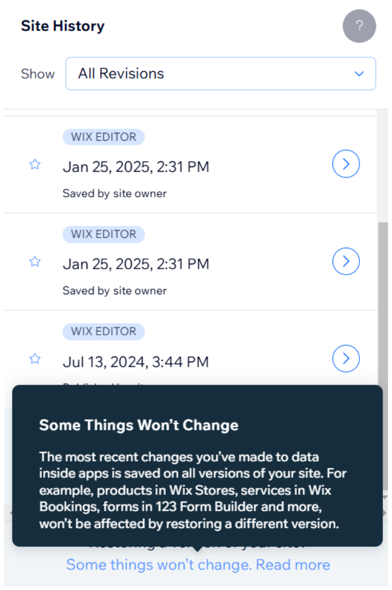
Try closing other tabs or programs to free up memory.

Error code: Out of Memory

Figure 16: Wix editor crashes due to not enough memory.

4.1.6.3. Poor Versioning Tools

Wix has a site history feature which records changes whenever users modify the web pages' UI. Users can manually save a copy or turn on the autosave features to save the changes periodically while they are in the editor. Users can also revert to a specific snapshot through the Site History dashboard. However, it can only keep a linear history, and reverting means erasing all changes after the recovered snapshot [69]. All editors of the web pages are also immediately affected by that revert since all editors share one dashboard and the same website layout. It is impossible for different editors to develop their website layout on their local editor and then “commit” them upstream to merge their changes. The site history is not page-specific but a snapshot of the entire website, so users cannot perform separate version control for each page. Moreover, this feature is not a snapshot of the entire website but merely a snapshot of the frontend design, meaning that all non-UI changes, such as program descriptions in Wix Booking or any CMS content, would not be tracked, as shown in Figure 17.


Site History

Show All Revisions

WIX EDITOR
Jan 25, 2025, 2:31 PM
Saved by site owner

WIX EDITOR
Jan 25, 2025, 2:31 PM
Saved by site owner

WIX EDITOR
Jul 13, 2024, 3:44 PM

Some Things Won't Change

The most recent changes you've made to data inside apps is saved on all versions of your site. For example, products in Wix Stores, services in Wix Bookings, forms in 123 Form Builder and more, won't be affected by restoring a different version.

Some things won't change. Read more

Figure 17: Wix editor site history feature, with the "Some Things Won't Change" warning.

4.1.6.4. Lack of TypeScript Support

While Wix Velo allows developers to customise their web pages, its in-built code editor is stuck with JavaScript. The said editor does not support TypeScript, which means no static type checks and error catching during transpile time.

4.1.7. Discussion

Overall, we notice the ease of use of website builders, especially when the builders have precisely the features we are looking for. For instance, we can quickly transfer the domain between websites. However, as soon as we have some non-general requirements and customisations that do not exist in its current system, we have to take alternative approaches to build the features on top of the Wix system, which is often restricted by what is allowed in Wix Velo and its relevant API. This limitation is why we must build a custom parser to localise camp program information. Moreover, building these custom features takes more effort and time than building them in some custom tech stacks other than website builders. For instance, the calendar in the program details page can be done much easier in a custom-coded web page such as React, Vue or even pure HTML + CSS, but we have to modularise it and wrap it with complex JavaScript logic for the calendar to work correctly.

We identify that Wix's close-source nature makes it even more challenging to build customisation on top of it. For example, modifying the Wix Booking plugin to allow varied pricing in its API is possible if the source code is released and made available openly. We notice similar issues may not happen in WordPress, considering its Wix Booking counterpart, Woocommerce, is open source. Even though the booking component in Woocommerce is a premium feature, it is still possible to modify its features as its source code is available, although such modifications are not encouraged as the modifications may break during plugin upgrades.

We also notice that even though Wix is quite well-established, there is still room for user experience improvements. For instance, its versioning tool only maintains a linear history, erases all changes made after that snapshot, and is not fine-grained enough to have different versions for each page.

4.2. User Survey

4.2.1. Purpose of Study

We survey both potential users and frequent users of website builders to identify the limitations they encountered. We aim to gain insights into potential new solutions from the limitations they encounter. Since we had already identified some limitations from the case study, we proposed some improvements: sketch-to-code, plugins-of-plugins systems, Git-powered version controls, and offline editor, and asked participants for their opinions about these features. We then examined the collected user input before making our next decision on the implementation. The survey questions we designed can be found in Appendix B.

4.2.2. Method

First, we created a series of questions through Microsoft Forms, including multiple-choice, multiple-response, Likert-scale and open-ended questions. We intended to measure perceived importance and frustration levels through Likert-scale questions and capture novel insights outside of our pre-defined answers from open-ended responses. We performed a pilot test with a small group of 5 participants to catch unclear questions, misleading answers, miswords, or technical issues. After ensuring the clarity and relevance of the survey questions, we recruited participants through online forums, social media groups and professional networks. At the start of the survey, we informed them about the purpose of the study and the right to withdraw at any time. We also mentioned that the response would be anonymised as no identifiable information was collected, and the response would only be shown in an aggregated manner. After the data collection, we filtered out responses in less than 3 minutes as the participants might have rushed through the surveys and did not read the questions carefully, hence not providing meaningful insights.

4.2.3. Participants

We collected 27 responses, and after filtering out those responses that finished in under 3 minutes, we found 21 valid responses. The majority of respondents (52%, 11) are students, followed by developers (24%, 5), hobbyists (10%, 2), product managers (10%, 2) and a business owner (4%, 1).

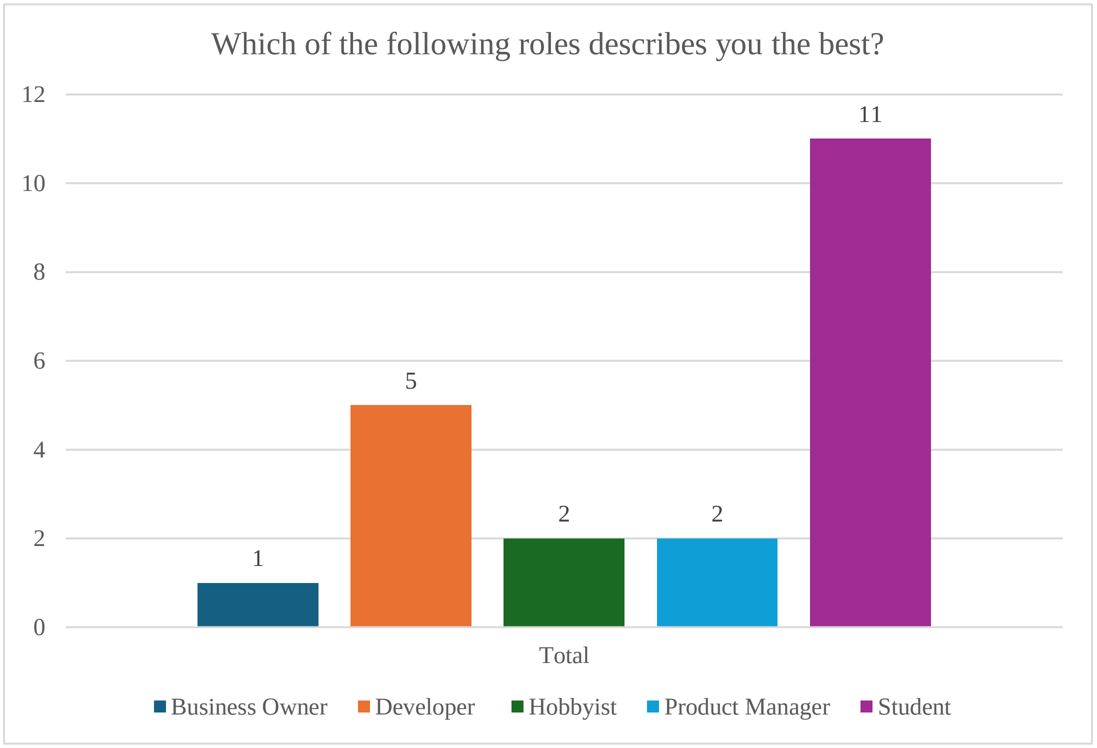


Figure 18: A bar chart that visualises the distribution of respondents based on their self-identified roles.

Among all the respondents, we noticed six respondents had zero website builder experience, while the rest mostly had one to two years of experience in website builders. In the discussion below, we would label those respondents with no experience as potential users while the rest as experienced users.

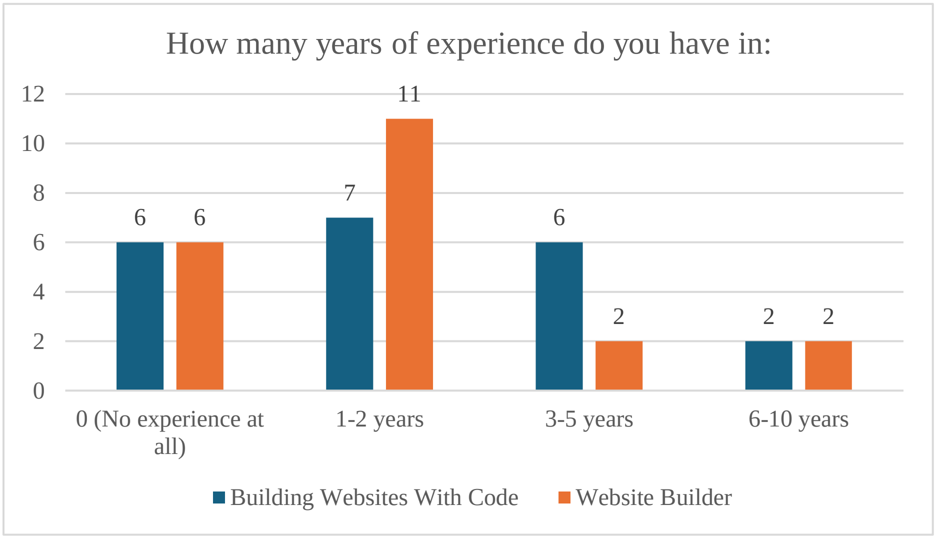


Figure 19: A bar chart showing respondents' years of experience building websites from scratch versus using website builders.

We further identified why these respondents had never used website builders before and found that most (67%, 4) have never needed to build a website, which we could then argue we only left two respondents who might have an interest in website builders. Considering the small sample size, making any generalisation based on their responses is unfair. Moreover, none of these respondents provide any open-ended responses for analysis. Hence, we skip them from our discussion.

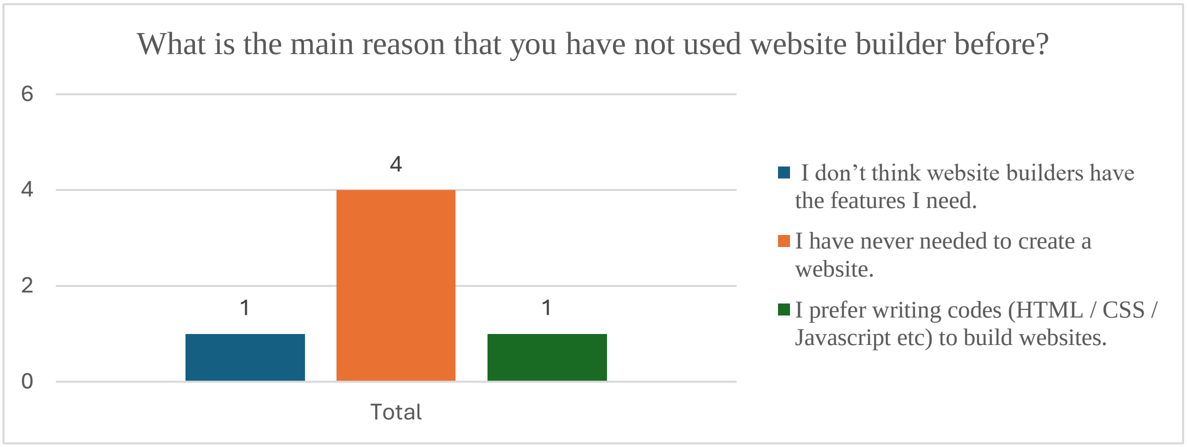


Figure 20: A bar chart showing respondents' primary reasons for not using a website builder.

4.2.4. Results and Discussion

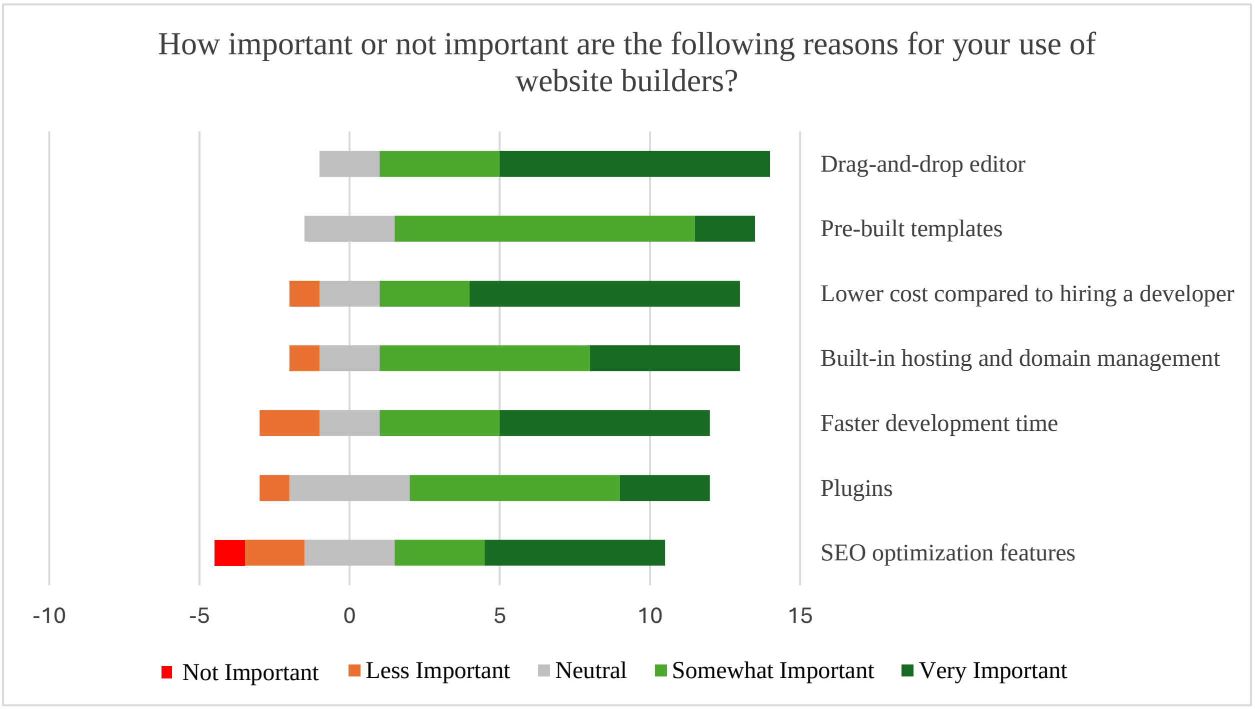


Figure 21: A bar chart that illustrates user opinions on various features of website builders, rated on a five-point scale ranging from “Not Important” (left) to “Very Important” (right).

Among the experienced users, we asked them to evaluate why they chose to use website builders, such as SEO optimisation, development time, plugins, lower cost compared to hiring a developer, hosting and domain management, pre-built templates and drag-and-drop editor. Most responders rated drag-and-drop editors and pre-built templates as the most important features, followed by hosting, lower cost and plugins. This result is not surprising, considering drag-and-drop editors and templates are the main selling point of website builders.

In the open-ended section, several respondents highlighted the ease of use of website builders, especially for non-technical people. One participant noted the “low learning curve”, while another appreciated that it was “easier to let non-technical staff edit the site without having to take up developer time. ” P18 concluded this in the best way, as in his words: “I don’t need to learn complex programming systems, yet I can design a simple and functional website that allows my users to navigate easily.”

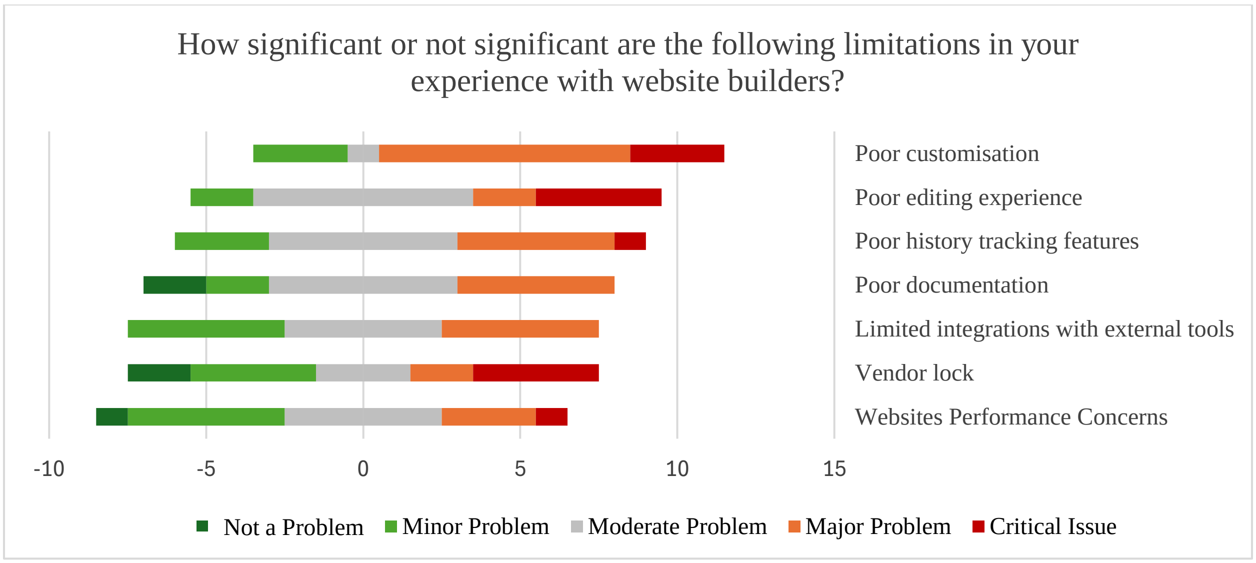


Figure 22: A bar chart that presents users' perceptions of limitations in website builders, rated on a five-point scale ranging from "Not a Problem" (left) to "Critical Issue" (right).

Most experienced users among the respondents felt that lack of customisation was the most significant limitation for website builders, with poor editing experience and history tracking features trailing behind. Notably, users were at the two extremes when asked about vendor lock. We assumed this was due to two reasons. When users choose website builders, some may have plans in mind that their use case will not require many extra features, and the website builder they choose has the exact features provided. However, those using website builders for prototyping or projects with evolving requirements may use one website builder in the initial stage of the project and find it unsuitable for future requirements.

The respondents primarily complained about website builders' lack of customisation in the open-ended section. P10 claimed that coding websites from scratch is "easier to manage mobile responsive" and "can have custom CSS", while P18 stated that website builders are frustrated as "some things which would be trivial to do in HTML/CSS that become almost impossible using a website builder". P14 pointed out the design limitations set by website builders and the limited available templates, which "often prevents me from fully expressing my creativity". Notably, P25 mentioned the accessibility issues similar to the research findings by Oswal and Oswal [37], "in particular for blind users … may not be able to utilize certain interactions well (e.g. drag and drop)."

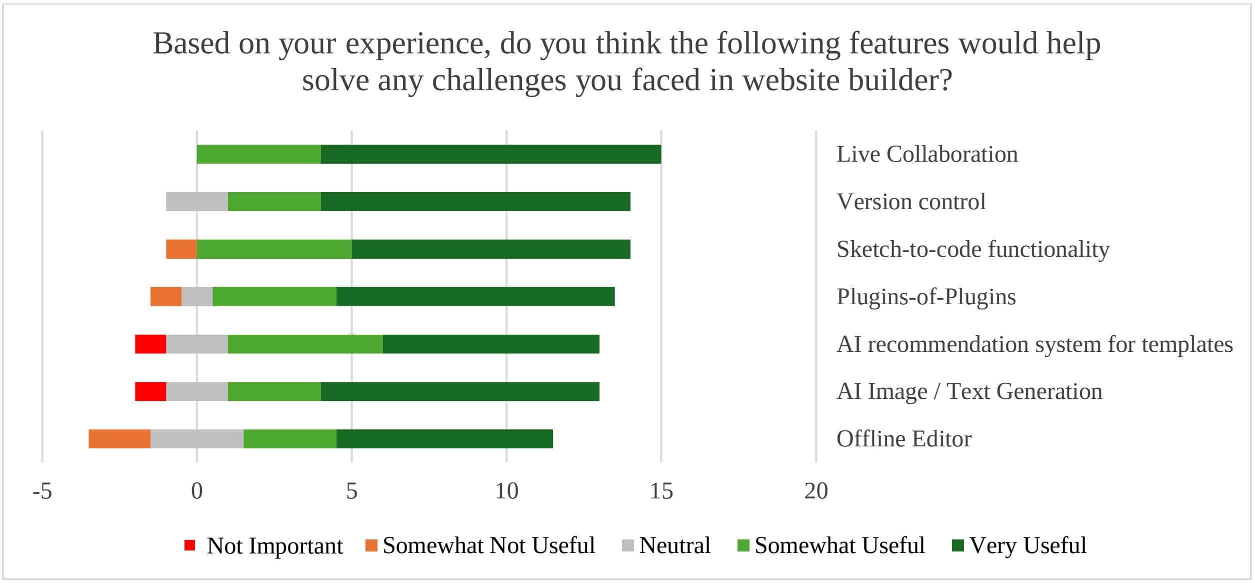


Figure 23: A bar chart that presents users' perceptions of how useful the proposed features would be, rated on a five-point scale ranging from "Not Important" (left) to "Very Useful" (right).

We posed a question for experienced users to rate our proposed features with the premise of solving some challenges they faced in website builders. Before the rating, we explained in detail what we meant by the terms to ensure respondents understood the concept. For instance, we introduced the Plugins-of-Plugins system as "having extensions that enhance the plugins or tools", which would allow custom payment methods, custom checkout flow, and custom media plugins.

We spotted that the top three rated features are live collaboration, version control, and sketch-to-code functionality. Surprisingly, the offline editor attracts much less interest from the respondents than other features. This lower rating might be that experienced users among our respondents are accustomed to the fact that most website builders are Software-as-a-Service (SaaS) and have less concern that network connection is mandatory for website builders. Respondents rated the poor functionality of website editors as one of the main limitations, but the offline editor functionality did not gain much interest. This response might hint that the respondents are complaining about other editor aspects, such as the poor user experience, lack of live collaboration or slow editor.

Respondents provided some interesting ideas in the open-ended section. P5 suggested allowing natural language input to control the website builders, as quoted: "If I feel that some of the parts (of the websites) are not optimal … I want to instruct LLM to do those edits for me". P10 suggested adding mobile responsive support, while P17 expressed concern about these features' pricing and scalability. P18 proposed direct CSS modifications, Figma integration, and importing the Lottie [70] animation file. However, CSS modification and Figma integration are available in some website builders, such as Wix [71] and Builder.io [72]. This shows that some experienced users do not yet recognise these technologies in website builders.

4.3. Identified Limitations and Gaps

We recognise that the central theme of discussion in both the case study and user survey was the customizability of website builders. In the case study, we had to give up the horizontal scroll bar design as it was impossible with the Wix Editor. Another example is the Wix Bookings Plugin, which did not support varied pricing for online payment, so the only solution was to create a custom plugin ourselves. Lack of customizability also ranks as the top limitation of website builders among our survey's expert website builder users. We notice that closed-source website builders generally make customisability much more challenging, as discussed previously, where varied pricing was not supported for the Wix Booking Plugin. Website builders being close-source also makes implementation much more challenging than the open-source alternatives, as developers have no idea what happened under the hood, and it will be even worse without proper documentation.

We also discover other limitations in website builders, such as poor documentation and poor editing experience. We encounter erroneous documentation in Wix where the responses do not precisely match what was shown in the documentation during our case study. We also see user complaints in our user survey for the same problem. We also encounter some problems with the current drag-and-drop editor, such as lag, crashes, or having no direct access to the CSS, as mentioned by the respondents in the user survey. Another limitation is the lack of proper versioning tools in website builders. Among the power users in our respondents, versioning tools are also the second most wanted feature.

4.4 Proposed Ideas

Since the lack of customizability has been the recurring theme during our discovery, we plan to tackle it as our main area of improvement. We believe that frontend UI generation through designs and sketches and the plugins-of-plugins systems can help mitigate this. We also found an HTML importation feature, which can further diminish the customisation limitation in website builders. We will discuss this importation feature, the frontend UI generation and the plugins-of-plugins system later in Chapter 6.

Regarding the documentation, we believe it is more of a task that requires human intervention, and the solution is to have developers provide them with excellent care. Regardless, we find some automated tools that allow the generation of documentation, which may help provide guidelines for the new developers to understand and develop the systems. Similarly, we elaborate on this in Chapter 6. Regarding the versioning issues in website builders, we proposed integrating versioning tools with Git. We believe this is technically possible, as WordPress has a plugin that achieves a similar functionality, namely VersionPress [73]. Due to the time limitations, we will not explore such functionality in our proof-of-concept website builders.

Our next step is to verify the technical feasibility of all our proposed solutions through experiments and implementation, namely frontend UI generation from designs and sketches, plugins-of-plugins systems, HTML importation and documentation generation features.

5. Experiment

5.1. Purpose of Study

We propose frontend UI generation from designs and sketches as potential solutions, as it allows users to focus on whatever they want and lets the website builders handle whatever customisation may be required. During our case study, we saw a pattern where the business owner modified the design of some small components from the layout the external contractors designed. We argue that partial frontend UI generation is valuable, especially for non-technical people, such as SME owners or product managers, to make minor changes like adding or modifying some components with sketches or designs.

However, while users express interest in sketch-to-code in user surveys, it is not generally available in mainstream website builders such as WordPress and Wix. For instance, we can only find frontend code generation from designs available in some less popular website builders like Builder.io [72] and App Builder [74], let alone generation from sketches. Since such a feature is rare among popular website builders, we are curious if there are any fundamental technical difficulties in the task and whether specific approaches or LLMs may provide better performance.

In this study, we explore partial generation using low-fidelity sketches (partial sketch-to-code) and high-fidelity designs (partial design-to-code). We propose four different approaches to tackle the task: (1) Provide partial HTML code and a screenshot where new user design/sketch is merged into the screenshot (Merging), (2) Provide partial HTML code with a hint of where the LLM should insert the new code, and a screenshot where new user design/sketch is merged into the screenshot (Merging with Hint), (3) Provide partial HTML with a hint of where the LLM should insert the new code and the new design in fixed width (Fixed-width with Hint), (4) Provide partial HTML with a hint of where the LLM should insert the new code and the new design/ sketch which has varied sizes (Varied-sized with Hint). As the sketch we prepare is varied in size by nature, we will skip (3) for the low-fidelity sketches. Considering the task required vision capabilities from LLMs, we conduct experiments using a total of 7 VLLMs, four of them are open-source models: Llama 3.2-vision:11b, Llama 3.2-vision:90b, Gemma2:27b, Gemma3:27b; and three of them are commercial models: Gemini 2.0-Flash, GPT-4o and Grok 2.0 vision. We then utilise established metrics from previous studies to evaluate their performance.

5.2. Method

5.2.1. Dataset

We created the test dataset by selectively choosing HTML pages from the WebCode2M training set [10] with fewer than 2048 tokens. A machine-learning-powered profanity check was then performed to filter harmful and explicit data. Among these web pages, we manually picked 25 responsive web pages for high-fidelity design and 15 responsive web pages for low-fidelity sketches.

Regarding the high-fidelity design dataset, we duplicated the 25 web pages into a dataset of 50 web pages. For each web page, we randomly chose one group of components we wished to remove afterwards, which must be next to each other. We took a full screenshot of the original web page and extracted the screenshot of those components. Afterwards, we manually replaced that one chosen group of components from the HTML code with a <missing></missing> tag and applied *uncss* [75], a popular JavaScript library for minifying CSS, to remove all the unused CSS from the codebase. We refer to this web page with the <missing> tag as “Partial HTML” in the later sections.

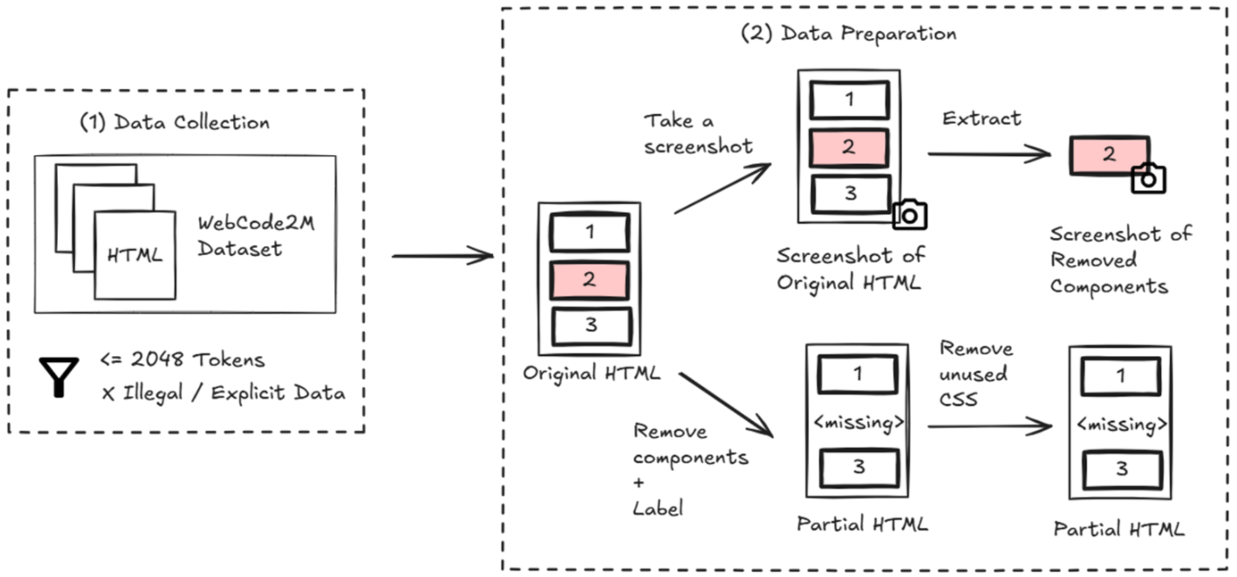


Figure 24: The Pipeline of Constructing the High-Fidelity Design Dataset.

For the low-fidelity sketch dataset, we removed the selected component from the HTML code, including all its relevant CSS, similar to the high-fidelity design dataset. We then manually sketched the removed components and saved the sketch separately. Additionally, we modified the web page screenshot by overlaying the sketch on top of the to-be-replaced components. Appendix C contains examples of our high-fidelity design and low-fidelity sketch dataset.

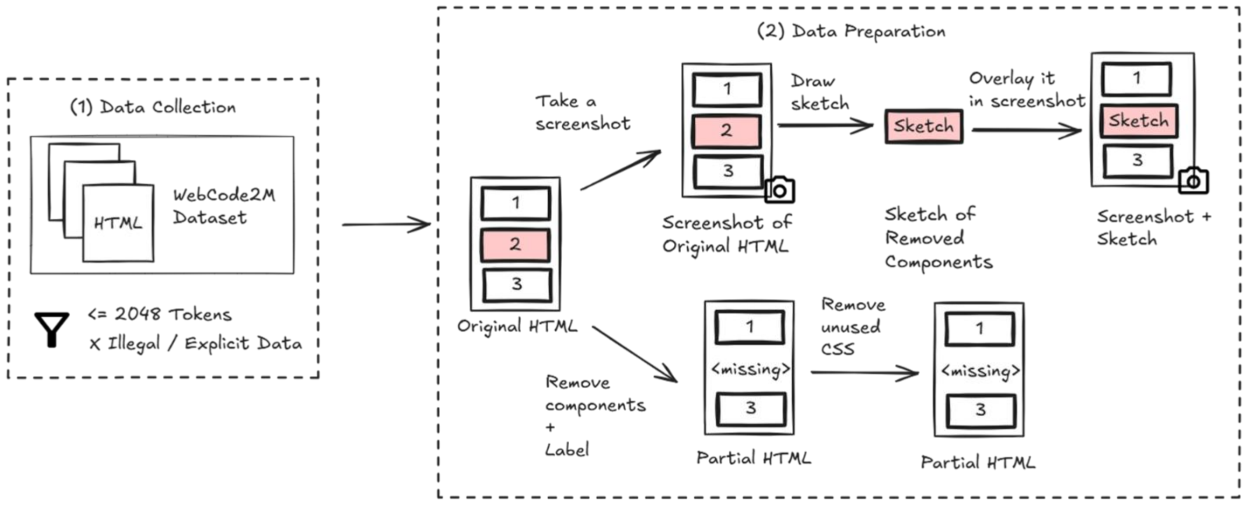


Figure 25: The pipeline for constructing the Low-Fidelity Sketches Dataset.

5.2.2. Prompting Methods

As discussed in 5.1, we evaluate the VLLMs using four different methods: Merging, Merging with Hint, Fixed-sized with Hint, and Varied-sized with Hint.

In the Merging method, we remove the <missing> label in partial HTML, then feed both this partial HTML and a screenshot where the new user design/sketch is merged into the screenshot to the VLLM and ask it to identify the new design/sketch and insert or modify the code anywhere it feels fit.

In the Merging with Hint method, we provide both partial HTML with the <missing> tag and a screenshot where the new user's design/sketch is merged into the screenshot. We ask the VLLM to replace the <missing> tag with the new code it generates. The <missing> tag here acts as a hint for the VLLM to input its changes.

As for the Fixed-sized with Hint method, we provide partial HTML with the <missing> tag and the new design with a fixed width. We arbitrarily picked 1280 pixels as the width, as it is both the default width used in the previous study and the default width set by the tool we used to take screenshots, Playwright.

Lastly, for the Varied-Sized with Hint method, we provide partial HTML with the <missing> tag and the new design/sketch with varied sizes. The main idea is that users may not be aware of the responsiveness or the preferred width they want while preparing their designs/sketches. Hence, we arbitrarily pick a reasonable size for all the designs/sketches. For design, the idea of reasonable size is that since the web pages in our datasets are responsive by default, we pick a size that is long in width, and the content is still understandable in human eyes. On the other hand, for sketches, we tried our best to mimic the website layout captured with a width of 1280 as much as possible. However, the layout in the sketch may somewhat differ from the original if we cannot fit all the contents within the width of the A4 paper. This is the same scenario where the users may only have a rough idea of the sketches while not necessarily thinking of how the website will look under different screen sizes.

Regardless of the methods used, we remove the HTML and CSS comments and minify the HTML by removing any spaces and next-line characters between HTML tags before sending it to the VLLMs. We also ask VLLMs to make minimal and only necessary changes to the existing HTML components and follow the existing styles as much as possible. After the VLLM generate the response, we parse it using regex to extract the HTML and CSS code. All the prompts we used can be found in Appendix D.

5.2.3. Evaluation Metrics

The code generated from partial designs is compared against the original HTML and CSS code and evaluated according to metrics proposed in Design2Code [11]. The idea is to evaluate the VLLMs' HTML responses as screenshots to evaluate their visual similarity. Two types of metrics are used: high-level visual similarity and low-level element matching.

In high-level visual similarity, we provide insight into the semantic similarity between the components with CLIP [76], a vision language model trained to align images and their textual description. Regardless, we are only interested in its image feature embeddings produced by CLIP, not the textual alignment. We obtained it through CLIP-ViT-B/32 by feeding it a square-ish screenshot. To ensure the similarity measure reflects only the visual (non-textual) content, we follow the methodology described in the original metric and apply the inpainting algorithm from Telea [77] to mask the deleted text regions.

For the low-level element catching, as discussed in Design2Code, both the generated and original web pages will be separated into blocks containing textual content by a text detection module. The detected blocks are then matched using Jonker-Volgenant algorithm, which helps us find the best one-to-one pairings between the boxes in R and G based on how similar their text is. Afterwards, the evaluation will be done between these one-to-one block pairs.

In the following formulas, generated web pages' blocks are labelled as G and R for the original.

- Block-Match

  In Design2Code, this measures if LLMs hallucinate about non-existent new elements. In our use case, it also helps measure how well the generated webpage copies all the visual parts from the original and if it includes everything in the sketch without adding any extra made-up elements. This score is measured by comparing the total size of correctly matched blocks to the total size of all blocks, in which the total size includes blocks missed by the generated page and those added incorrectly.

$$\mathbf{match_{block}}(r_p, g_q) = \frac{S(r_p) + S(g_q)}{\sum_{(i,j)\in M}(S(r_i) + S(g_j)) + (\sum_{i\in U_R} S(r_i) + \sum_{j\in U_G} S(g_j))}, \quad (1)$$

$$\mathbf{match_{block}}(R, G) = \sum_{(p,q)\in M} \mathbf{match_{block}}(r_p, g_q). \quad (2)$$

Figure 26: Equations used to calculate the Block-Match metric, as described in Design2Code. The S function returns the total size of the blocks, while U_r and U_g represent the unmatched blocks in R and G.

- Text Similarity: Similarity between the corresponding text boxes.

  For each pair of matching text boxes, we check how similar their content is at the character level. This calculation is done using the character-based Sørensen-Dice score, which is calculated by multiplying the overlapping characters by two and then dividing it by the total characters in both boxes.

- Position Similarity: How closely do the blocks match those in the original?
  For each matched block pair, we normalised their x and y coordinates to [0, 1], then obtained the differences of x coordinates between the pair and then the y coordinates. Between these two differences, we pick whichever has a higher value and use one to minus that value to obtain the position similarity score. Generally, if the position difference is large, then the difference in coordinates is significant, and hence, the score is lower, and vice versa.
- Colour Similarity: How close are the colours generated between the blocks?
  We use CIEDE2000 [78] to assess the colour difference for each matched block pair, which considers the complexities of human vision in colours.

Additionally, we track the number of generations required to produce valid HTML. However, if it takes more than 10 attempts to generate a valid HTML and CSS, we consider the generation unsuccessful.

We use similar metrics for our sketch-to-code but without colour similarity, as sketches are generally considered to be drawn on white paper in previous studies. We also do not consider using the exact colours when replicating the screenshots in our sketches. Hence, evaluating the colour similarity across different methods or VLLMs is unfair for sketch-to-code.

While the metrics above provide high-level and low-level insights on UI similarity, these evaluation metrics did not consider how viewpoints may affect the result. Specifically, Design2Code evaluated the performance in a viewport of 1280x720, a default viewport set by Playwright, the screenshot tool used in the experiment. However, the generated code may not work under all viewports and may perform better or worse in others. We propose evaluating the generated code under different viewports, in which we collect the commonly used viewports from the dataset provided by Statcounter, which collects the data through the web pages that install it for data analytics. Due to the mass computation power required to run evaluations on various viewports, we only pick the top 50% of viewports used on the three devices, desktop, mobile and tablet, as the most interesting viewports in our study. We then adopt a uniform-weight evaluation, where these viewports contribute equal weight regardless of real-world prevalence. This evaluation allows for a fairer model robustness and consistency assessment across all screen sizes and aspect ratios.

5.3. Results and Analysis

We split our discussion into several research questions (RQ) and answered them in the following sections. Due to the relatively smaller dataset for sketches, we focus on the results from partial design-to-code while having a separate RQ specifically targeting partial sketch-to-code (RQ3).

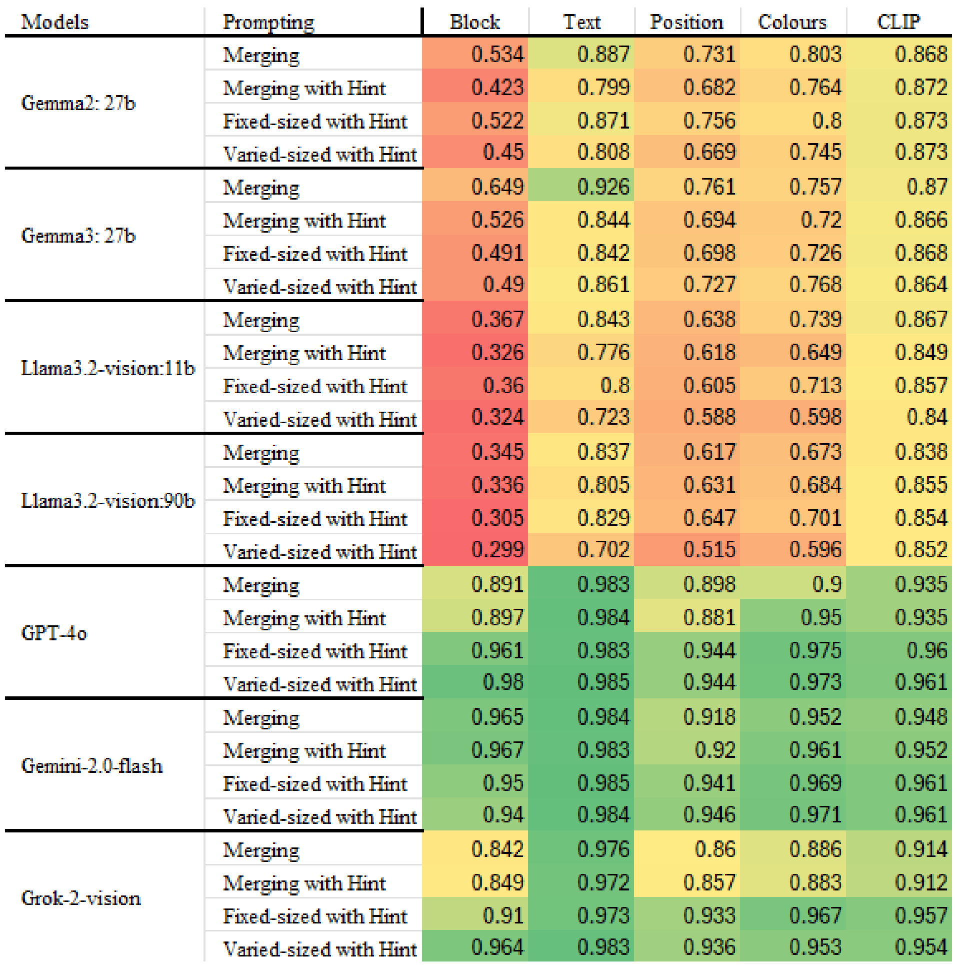
5.3.1. RQ1: Which method consistently produces the most accurate code across input types and devices?

In this section, we look at design-to-code performance for our primary analysis. Since we are interested in the difference between methods, we apply the uniform weight scheme where each viewport contributes equally. We report the mean score across all pages. The standard deviation shown in the table reflects inter-page variance, which is calculated by averaging across all viewports for each page and then computing the standard deviation across pages.

Table 1: Partial design-to-code performance across various models and prompting strategies. Each cell reports the mean ± standard deviation across pages for key layout metrics.

Models	Prompting	Block	Text	Position	Colours	CLIP
Gemma 2: 27b	Merging	0.534 ± 0.326	0.887 ± 0.211	0.731 ± 0.183	0.803 ± 0.269	0.868 ± 0.039
	Merging with Hint	0.423 ± 0.345	0.799 ± 0.285	0.682 ± 0.251	0.764 ± 0.307	0.872 ± 0.041
	Fixed-sized with Hint	0.522 ± 0.345	0.871 ± 0.248	0.756 ± 0.225	0.800 ± 0.293	0.873 ± 0.047
	Varied-sized with Hint	0.450 ± 0.364	0.808 ± 0.299	0.669 ± 0.274	0.745 ± 0.337	0.873 ± 0.041
Gemma 3: 27b	Merging	0.649 ± 0.293	0.926 ± 0.148	0.761 ± 0.144	0.757 ± 0.213	0.870 ± 0.039
	Merging with Hint	0.526 ± 0.320	0.844 ± 0.293	0.694 ± 0.249	0.720 ± 0.302	0.866 ± 0.069
	Fixed-sized with Hint	0.491 ± 0.327	0.842 ± 0.288	0.698 ± 0.254	0.726 ± 0.297	0.868 ± 0.057
	Varied-sized with Hint	0.490 ± 0.313	0.861 ± 0.235	0.727 ± 0.230	0.768 ± 0.260	0.864 ± 0.049
Llama3.2-vision:11b	Merging	0.367 ± 0.313	0.843 ± 0.298	0.638 ± 0.282	0.739 ± 0.309	0.867 ± 0.047
	Merging with Hint	0.326 ± 0.283	0.776 ± 0.329	0.618 ± 0.302	0.649 ± 0.366	0.849 ± 0.049
	Fixed-sized with Hint	0.360 ± 0.346	0.800 ± 0.338	0.605 ± 0.312	0.713 ± 0.386	0.857 ± 0.057
	Varied-sized with Hint	0.324 ± 0.347	0.723 ± 0.376	0.588 ± 0.331	0.598 ± 0.401	0.840 ± 0.068

Llama3.2-vision:90b	Merging	0.345 ± 0.326	0.837 ± 0.269	0.617 ± 0.247	0.673 ± 0.311	0.838 ± 0.067
	Merging with Hint	0.336 ± 0.333	0.805 ± 0.297	0.631 ± 0.257	0.684 ± 0.348	0.855 ± 0.036
	Fixed-sized with Hint	0.305 ± 0.304	0.829 ± 0.272	0.647 ± 0.256	0.701 ± 0.304	0.854 ± 0.047
	Varied-sized with Hint	0.299 ± 0.333	0.702 ± 0.395	0.515 ± 0.349	0.596 ± 0.413	0.852 ± 0.050
GPT-4o	Merging	0.891 ± 0.234	0.983 ± 0.017	0.898 ± 0.058	0.900 ± 0.127	0.935 ± 0.030
	Merging with Hint	0.897 ± 0.263	0.984 ± 0.016	0.881 ± 0.079	0.950 ± 0.059	0.935 ± 0.030
	Fixed-sized with Hint	0.961 ± 0.169	0.983 ± 0.017	0.944 ± 0.049	**0.975 ± 0.039**	0.960 ± 0.022
	Varied-sized with Hint	**0.980 ± 0.120**	**0.985 ± 0.016**	0.944 ± 0.046	0.973 ± 0.046	0.961 ± 0.017
Gemini-2.0-flash	Merging	0.965 ± 0.100	0.984 ± 0.017	0.918 ± 0.067	0.952 ± 0.080	0.948 ± 0.028
	Merging with Hint	0.967 ± 0.092	0.983 ± 0.016	0.920 ± 0.075	0.961 ± 0.057	0.952 ± 0.024
	Fixed-sized with Hint	0.950 ± 0.153	**0.985 ± 0.016**	0.941 ± 0.071	0.969 ± 0.058	0.961 ± 0.019
	Varied-sized with Hint	0.940 ± 0.176	0.984 ± 0.016	**0.946 ± 0.060**	0.971 ± 0.063	**0.961 ± 0.016**
Grok-2-vision	Merging	0.842 ± 0.221	0.976 ± 0.020	0.860 ± 0.077	0.886 ± 0.169	0.914 ± 0.029
	Merging with Hint	0.849 ± 0.224	0.972 ± 0.024	0.857 ± 0.076	0.883 ± 0.141	0.912 ± 0.028
	Fixed-sized with Hint	0.910 ± 0.168	0.973 ± 0.022	0.933 ± 0.050	0.967 ± 0.055	0.957 ± 0.019
	Varied-sized with Hint	0.964 ± 0.129	0.983 ± 0.016	0.936 ± 0.050	0.953 ± 0.074	0.954 ± 0.023



Models	Prompting	Block	Text	Position	Colours	CLIP
Gemma2: 27b	Merging	0.534	0.887	0.731	0.803	0.868
	Merging with Hint	0.423	0.799	0.682	0.764	0.872
	Fixed-sized with Hint	0.522	0.871	0.756	0.8	0.873
	Varied-sized with Hint	0.45	0.808	0.669	0.745	0.873
Gemma3: 27b	Merging	0.649	0.926	0.761	0.757	0.87
	Merging with Hint	0.526	0.844	0.694	0.72	0.866
	Fixed-sized with Hint	0.491	0.842	0.698	0.726	0.868
	Varied-sized with Hint	0.49	0.861	0.727	0.768	0.864
Llama3.2-vision:11b	Merging	0.367	0.843	0.638	0.739	0.867
	Merging with Hint	0.326	0.776	0.618	0.649	0.849
	Fixed-sized with Hint	0.36	0.8	0.605	0.713	0.857
	Varied-sized with Hint	0.324	0.723	0.588	0.598	0.84
Llama3.2-vision:90b	Merging	0.345	0.837	0.617	0.673	0.838
	Merging with Hint	0.336	0.805	0.631	0.684	0.855
	Fixed-sized with Hint	0.305	0.829	0.647	0.701	0.854
	Varied-sized with Hint	0.299	0.702	0.515	0.596	0.852
GPT-4o	Merging	0.891	0.983	0.898	0.9	0.935
	Merging with Hint	0.897	0.984	0.881	0.95	0.935
	Fixed-sized with Hint	0.961	0.983	0.944	0.975	0.96
	Varied-sized with Hint	0.98	0.985	0.944	0.973	0.961
Gemini-2.0-flash	Merging	0.965	0.984	0.918	0.952	0.948
	Merging with Hint	0.967	0.983	0.92	0.961	0.952
	Fixed-sized with Hint	0.95	0.985	0.941	0.969	0.961
	Varied-sized with Hint	0.94	0.984	0.946	0.971	0.961
Grok-2-vision	Merging	0.842	0.976	0.86	0.886	0.914
	Merging with Hint	0.849	0.972	0.857	0.883	0.912
	Fixed-sized with Hint	0.91	0.973	0.933	0.967	0.957
	Varied-sized with Hint	0.964	0.983	0.936	0.953	0.954

Figure 27: Heatmap of partial design-to-code performance across prompting strategies and language-vision models.

We observe that the open-source models perform much worse than commercial models in all methods, matching the previous study's findings. Across all methods in open-source models, the best method is "Merging", the standard method used in full design-to-code. In Gemma2, Gemma3 and Llama3.2-vision:11b, "Merging" achieves the highest or the second-highest score in all metrics. "Merging" also has a much lower standard deviation than other methods in open-source models, with the lowest or second-lowest standard deviation in Gemma2, Gemma 3 and Llama3.2-vision:11b.

In commercial models, we notice that Fixed-Sized with Hint and Varied-Sized with Hint consistently produce high scores across most metrics with a low standard deviation. This finding is predominantly factual for GPT-4o and Grok-2-vision, where their scores across all but one metric are consistently higher and standard deviations are consistently lower in Fixed-Sized with Hint and Varied-Sized with Hint than the other two methods. This finding shows that commercial models can perform much better with a partial design and partial HTML with a hint of where to input the generated code.

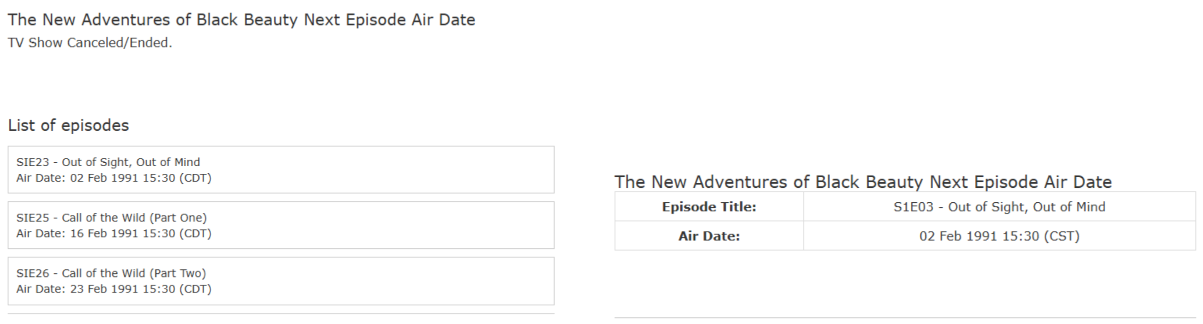
Generally, we notice that “Merging with Hint” has a comparable score to “Merging”. The <missing> tag hint seems to have little effect on the generation quality in the “Merging” method. We also notice relatively strong CLIP scores across all models and all methods, which suggest that the semantic similarity remains decent in partial frontend UI generation tasks regardless of the methods and models used. This result is likely because the partial HTML is fed to the VLLM for reference, and hence, VLLM can consistently achieve a relatively high score if they maintain the relevance between features in the existing codebase.

5.3.2. RQ2: How consistent are the VLLMs in the tasks?

To achieve usability, VLLMs must achieve high evaluation scores and consistently produce valid HTML code, which means they should output high-quality HTML code with a high success rate and minimal generation attempts. In this section, we dive into the variances of the VLLM for each method, their corresponding success rate and the average attempts needed for them to generate a valid HTML.

As shown in Table 1 and Figure 27, we notice an overall poor performance across all open-source models. In open-source models, their block similarity is notoriously low, with only Gemma2 and Gemma3 achieving a score higher than 0.5 in some methods. Moreover, regardless of the methods, they have a much higher variance than the commercial models. Only Gemma 3 achieves a relatively low variance, with 0.148 in text metrics and 0.144 in position metrics in the “Merging” methods, while the others often have higher variances than 0.2, indicating that their output quality is inconsistent.

On the other hand, commercial models perform much better than open-source models. We notice that regardless of the methods used, they all perform better than their open-source counterpart. For example, they all score exceptionally high in text similarity with low variance, suggesting the models are robust in recognising text contents in designs. The heat map also reflects this high text similarity score as the intensity of the green colour. In block metrics, their score is not lower than 0.84. We notice Gemini-20 Flash and GPT-4o as top performers among all commercial models. Gemini-2.0-Flash's scores are similar to GPT-4o, as they achieve the same score and standard deviation for text similarity, although they use different methods. However, GPT-4o provides a lower variance in both "Fixed-Sized with Hint" and "Varied-Sized with Hint" methods, indicating it is slightly more consistent than Gemini-2.0-Flash.



The New Adventures of Black Beauty Next Episode Air Date
TV Show Canceled/Ended.

List of episodes

SIE23 - Out of Sight, Out of Mind
Air Date: 02 Feb 1991 15:30 (CDT)

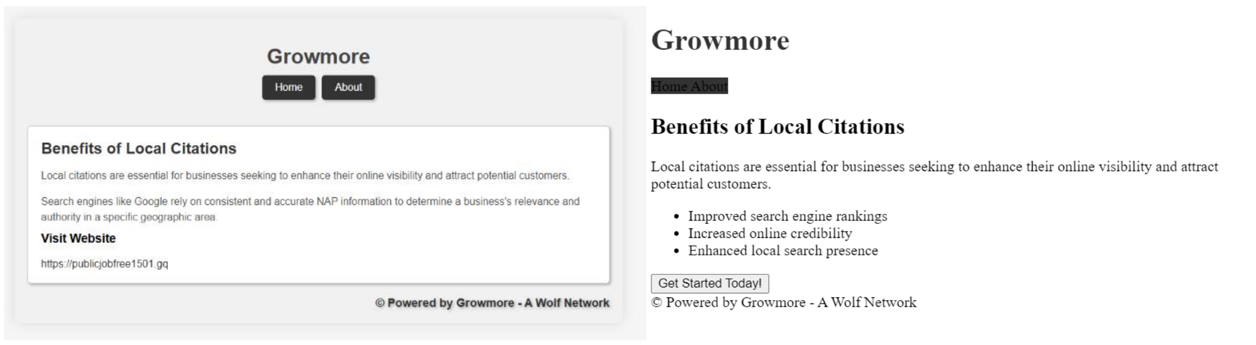
SIE25 - Call of the Wild (Part One)
Air Date: 16 Feb 1991 15:30 (CDT)

SIE26 - Call of the Wild (Part Two)
Air Date: 23 Feb 1991 15:30 (CDT)

The New Adventures of Black Beauty Next Episode Air Date

Episode Title:	S1E03 - Out of Sight, Out of Mind
Air Date:	02 Feb 1991 15:30 (CST)

Figure 28: Web pages generated via the "Merging" method from designs. The left (Gemini-2.0-Flash) has all the components in the sketch, while the right (Llama 3.2 Vision:11b) has missing components.



Growmore
Home About

Benefits of Local Citations

Local citations are essential for businesses seeking to enhance their online visibility and attract potential customers.

Search engines like Google rely on consistent and accurate NAP information to determine a business's relevance and authority in a specific geographic area.

Visit Website

https://publicjobfree1501.gq

© Powered by Growmore - A Wolf Network

Growmore

Home About

Benefits of Local Citations

Local citations are essential for businesses seeking to enhance their online visibility and attract potential customers.

- Improved search engine rankings
- Increased online credibility
- Enhanced local search presence

Get Started Today!
© Powered by Growmore - A Wolf Network

Figure 29: Web pages generated via the "Merging" method from sketches. The left (GPT-4o) maintains the original layout, while the right (Llama 3.2 Vision:90b) does not respect the existing designs and layout.

As we look into the code generated by the open-source models, we notice that they often miss some parts of the designs. For example, they may only generate a small part of the design and get a lower score overall. Moreover, they tend not to maintain the existing layout partial HTML, which leads to low block and position similarity. Since the open-source models achieve a high CLIP score even though the design is significantly modified, we believe CLIP may not be an accurate score to reflect these style errors compared to metrics like Block Similarity and Position Similarity.

Regarding the success rate and the number of generations required, we verify that all commercial models can usually generate the response in one go with a 100% success rate. However, open-source models often require more generations. They sometimes refuse to perform the task, citing a lack of vision, legal restrictions or other limitations. The most notorious model among all open-source models was Gemma2:27b, the only model that fails to generate some results from all methods. By inspecting its output, we notice that it sometimes produces non-relevant results and does not perform tasks as instructed, such as describing the content in the partial HTML instead of generating the code. In contrast, the rest of the open-source models have a 100% success rate, while Gemma 3:27b and Llama3.2-vision:90b require fewer generations than Llama3.2-vision.

Notably, when we compare Llama3.2-vision:90b and its counterpart, Llama3.2-vision, we notice it is much more consistent in its output by having a much lower average number of generations required. It also slightly performs better in all methods other than “Merging”. These findings suggest that models with higher parameters may outperform their lower-parameter counterparts in partial frontend UI generation tasks.

Table 2: The success rate and number of generations required for 7 VLLMs to generate a valid HTML.

Models	Prompting	Success Rate	Avg. # of Generation
Gemma2: 27b	Merging	94%	1.149
	Merging with Hint	92%	1.370
	Fixed-sized with Hint	96%	1.375
	Varied-sized with Hint	98%	1.143
Gemma3: 27b	Merging	100%	1.02
	Merging with Hint	100%	1.02
	Fixed-sized with Hint	100%	1.0
	Varied-sized with Hint	100%	1.0
Llama3.2-vision:11b	Merging	100%	1.38
	Merging with Hint	100%	1.04
	Fixed-sized with Hint	100%	1.22
	Varied-sized with Hint	100%	1.14
Llama3.2-vision:90b	Merging	100%	1.04
	Merging with Hint	100%	1.0

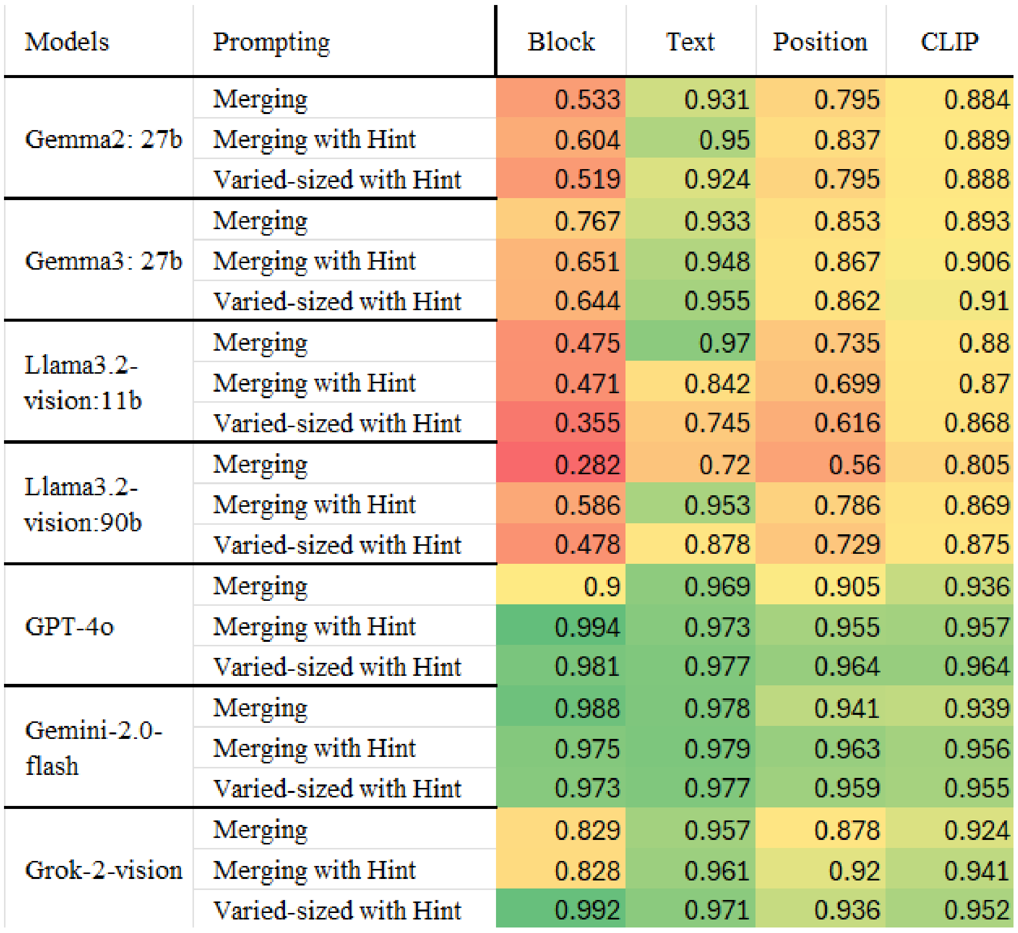
	Fixed-sized with Hint	100%	1.04
	Varied-sized with Hint	100%	1.0
GPT-4o Grok-2-vision Gemini-2.0-flash	Merging	100%	1.0
	Merging with Hint		
	Fixed-sized with Hint		
	Varied-sized with Hint		

5.3.3. RQ3: Does the fidelity of the input affect VLLM's performance?

This section explores whether VLLMs perform similarly when given sketches instead of high-fidelity designs as input.

Table 3: Partial sketch-to-code performance across various models and prompting strategies. Each cell reports the mean ± standard deviation across pages for key layout metrics.

Models	Prompting	Block	Text	Position	CLIP
Gemma2: 27b	Merging	0.533 ± 0.368	0.931 ± 0.098	0.795 ± 0.080	0.884 ± 0.042
	Merging with Hint	0.604 ± 0.285	0.950 ± 0.059	0.837 ± 0.116	0.889 ± 0.041
	Varied-sized with Hint	0.519 ± 0.340	0.924 ± 0.131	0.795 ± 0.150	0.888 ± 0.044
Gemma3: 27b	Merging	0.767 ± 0.277	0.933 ± 0.066	0.853 ± 0.090	0.893 ± 0.046
	Merging with Hint	0.651 ± 0.284	0.948 ± 0.047	0.867 ± 0.089	0.906 ± 0.046
	Varied-sized with Hint	0.644 ± 0.284	0.955 ± 0.038	0.862 ± 0.096	0.910 ± 0.041
Llama3.2-vision:11 b	Merging	0.475 ± 0.327	0.970 ± 0.043	0.735 ± 0.108	0.880 ± 0.052
	Merging with Hint	0.471 ± 0.307	0.842 ± 0.250	0.699 ± 0.287	0.870 ± 0.052
	Varied-sized with Hint	0.355 ± 0.312	0.745 ± 0.377	0.616 ± 0.357	0.868 ± 0.071
Llama3.2-vision:90 b	Merging	0.282 ± 0.313	0.720 ± 0.375	0.560 ± 0.326	0.805 ± 0.057
	Merging with Hint	0.586 ± 0.370	0.953 ± 0.056	0.786 ± 0.121	0.869 ± 0.046
	Varied-sized with Hint	0.478 ± 0.346	0.878 ± 0.242	0.729 ± 0.242	0.875 ± 0.049
GPT-4o	Merging	0.900 ± 0.228	0.969 ± 0.022	0.905 ± 0.077	0.936 ± 0.029
	Merging with Hint	**0.994 ± 0.009**	0.973 ± 0.014	0.955 ± 0.028	0.957 ± 0.021
	Varied-sized with Hint	0.981 ± 0.059	0.977 ± 0.015	**0.964 ± 0.036**	**0.964 ± 0.015**
Gemini-2.0-flash	Merging	0.988 ± 0.030	0.978 ± 0.014	0.941 ± 0.048	0.939 ± 0.040
	Merging with Hint	0.975 ± 0.076	**0.979 ± 0.013**	0.963 ± 0.028	0.956 ± 0.026
	Varied-sized with Hint	0.973 ± 0.075	0.977 ± 0.015	0.959 ± 0.031	0.955 ± 0.030
Grok-2-vision	Merging	0.829 ± 0.251	0.957 ± 0.029	0.878 ± 0.055	0.924 ± 0.026
	Merging with Hint	0.828 ± 0.211	0.961 ± 0.030	0.920 ± 0.050	0.941 ± 0.022
	Varied-sized with Hint	0.992 ± 0.015	0.971 ± 0.021	0.936 ± 0.053	0.952 ± 0.026



Models	Prompting	Block	Text	Position	CLIP
Gemma2: 27b	Merging	0.533	0.931	0.795	0.884
	Merging with Hint	0.604	0.95	0.837	0.889
	Varied-sized with Hint	0.519	0.924	0.795	0.888
Gemma3: 27b	Merging	0.767	0.933	0.853	0.893
	Merging with Hint	0.651	0.948	0.867	0.906
	Varied-sized with Hint	0.644	0.955	0.862	0.91
Llama3.2-vision:11b	Merging	0.475	0.97	0.735	0.88
	Merging with Hint	0.471	0.842	0.699	0.87
	Varied-sized with Hint	0.355	0.745	0.616	0.868
Llama3.2-vision:90b	Merging	0.282	0.72	0.56	0.805
	Merging with Hint	0.586	0.953	0.786	0.869
	Varied-sized with Hint	0.478	0.878	0.729	0.875
GPT-4o	Merging	0.9	0.969	0.905	0.936
	Merging with Hint	0.994	0.973	0.955	0.957
	Varied-sized with Hint	0.981	0.977	0.964	0.964
Gemini-2.0-flash	Merging	0.988	0.978	0.941	0.939
	Merging with Hint	0.975	0.979	0.963	0.956
	Varied-sized with Hint	0.973	0.977	0.959	0.955
Grok-2-vision	Merging	0.829	0.957	0.878	0.924
	Merging with Hint	0.828	0.961	0.92	0.941
	Varied-sized with Hint	0.992	0.971	0.936	0.952

Figure 30: Heatmap of partial sketch-to-code performance across prompting strategies and language-vision models.

While interpreting this table, it is important to note that the components we removed from the sketch differ from those we removed from the design. Therefore, a direct comparison between this table and the previous design-based results is inappropriate. However, we can still draw meaningful insights by examining relative performance across methods and models. Specifically, we want to draw insights into whether the methods that perform well on high-fidelity designs also perform well in sketches and whether VLLMs that perform well on high-fidelity designs maintain strong performance on sketch-based inputs.

We notice that for open-source models, while "Merging" still has a relatively high score compared to other methods in Gemma3:27b and Llama3.2-vision, "Merging with Hint" seems to perform better in open-source models, especially in Gemma:27b and Llama3.2-vision. Generally, the "Merging with Hint" method performs better than "Merging" regardless of the models used, with only slight exceptions, such as Block Similarity in Gemma3, which differs from the findings in partial design-to-code. Regardless, the open-source models perform worse than commercial models, and the top performers are still GPT-4o and Gemini-2.0-flash. However, Gemini-2.0-flash is more consistent than GPT-4o because of its lower variance. Grok-2-vision also has a comparable result, although slightly lower than GPT-4o and Gemini-2.0-flash; it also has a relatively high variance. For instance, it has the highest variance across all commercial models for block similarity in the "Merging" method. The CLIP scores are relatively stable across all models, similar to the partial design-to-code.

Regarding the success rate and the average generation required from successful generations, all models successfully generated responses in one go with a 100% success rate, with only Gemma3:27b requiring 1.067 attempts for "Merging", 1.133 attempts for "Merging with Hint" and 1.6 attempts for "Varied-Sized with Hint". Due to the limited size of our datasets, we cannot draw any insights on whether generating from sketches is more effortless than generating from designs. However, these data still show that VLLMs can consistently generate valid code with all these methods.

5.3.4. RQ4: How responsive is the code generated by VLLMs?

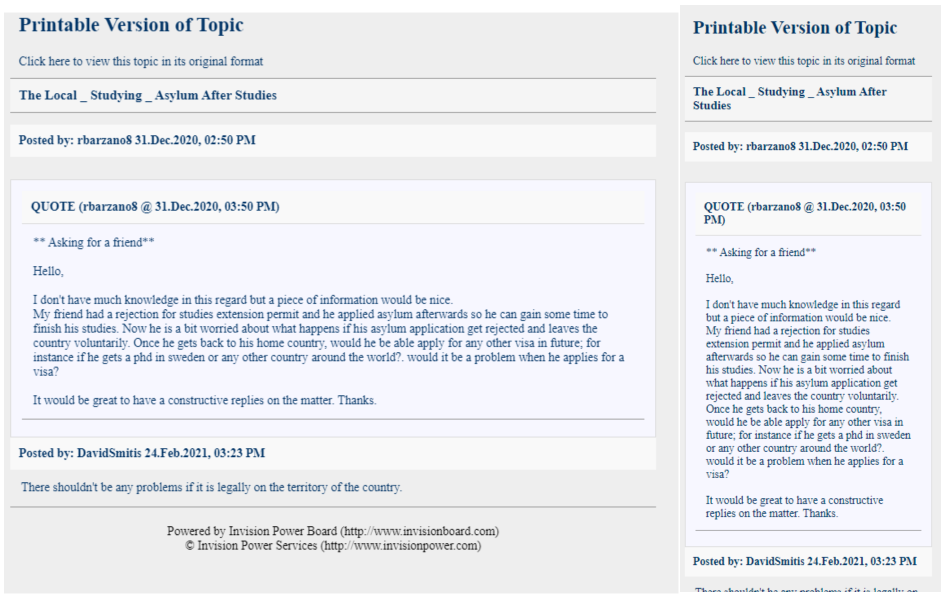
We evaluate how well the generated website adapts to the Responsive Web Design by calculating the inter-viewport variance, which is calculated by averaging the results across all pages for each viewport and then computing the standard deviation across these viewport-levels means. A lower variance indicates more consistent performance across different screen sizes, reflecting better responsiveness. We limit this evaluation to the partial design-to-code setting, as its dataset is larger than the partial sketch-to-code.

Table 4: Visual metrics' inter-viewport variance (standard deviation) across various VLLM models and prompting strategies.

Models	Prompting	Block Score Std	Text Score Std	Position Score Std	Colour Score Std	Clip Store Std
Gemma2: 27b	Merging	0.013	0.001	0.014	0.002	0.005
	Merging with Hint	0.012	0.004	0.007	0.004	0.004
	Fixed-sized with Hint	0.010	0.000	0.007	0.000	0.003
	Varied-sized with	0.010	0.000	0.005	0.000	0.004

	Hint					
Gemma3: 27b	Merging	0.011	0.004	0.014	0.003	0.004
	Merging with Hint	0.006	0.000	0.012	0.001	0.003
	Fixed-sized with Hint	0.007	0.001	0.002	0.000	0.007
	Varied-sized with Hint	0.011	0.000	0.004	0.001	0.004
Llama3.2-vision:11b	Merging	0.012	0.000	0.004	0.000	0.005
	Merging with Hint	0.014	0.004	0.005	0.004	0.006
	Fixed-sized with Hint	0.008	0.000	0.004	0.000	0.003
	Varied-sized with Hint	0.012	0.000	0.006	0.000	0.007
Llama3.2-vision:90b	Merging	0.015	0.004	0.012	0.003	0.010
	Merging with Hint	0.019	0.000	0.007	0.001	0.005
	Fixed-sized with Hint	0.013	0.000	0.009	0.000	0.004
	Varied-sized with Hint	0.007	0.000	0.005	0.000	0.003
GPT-4o	Merging	0.003	0.000	0.003	0.000	0.003
	Merging with Hint	0.003	0.000	0.011	0.000	0.004
	Fixed-sized with Hint	0.003	0.000	0.003	0.000	0.003
	Varied-sized with Hint	0.003	0.000	0.003	0.001	0.003
Gemini-2.0-flash	Merging	0.001	0.000	0.002	0.000	0.003
	Merging with Hint	0.001	0.000	0.006	0.000	0.002
	Fixed-sized with Hint	0.004	0.000	0.002	0.000	0.003
	Varied-sized with Hint	0.002	0.000	0.004	0.000	0.002
Grok-2-vision	Merging	0.015	0.000	0.010	0.002	0.004
	Merging with Hint	0.004	0.001	0.010	0.001	0.005
	Fixed-sized with Hint	0.004	0.000	0.006	0.001	0.004
	Varied-sized with Hint	0.005	0.000	0.003	0.001	0.003

From the results, we notice an overall low variance between the models. These results reflect that VLLMs' output is generally responsive since they achieve a similar score across all viewports. Moreover, even open-source models that performed poorly in the previous evaluation also had low variance. However, that may be attributed to the partial HTML we provide being already responsive, and hence, the models can still generate responsive output, given that they do not entirely modify or remove the CSS that handles responsiveness.



Printable Version of Topic

Click here to view this topic in its original format

The Local _ Studying _ Asylum After Studies

Posted by: rbarzano8 31.Dec.2020, 02:50 PM

QUOTE (rbarzano8 @ 31.Dec.2020, 03:50 PM)

** Asking for a friend**

Hello,

I don't have much knowledge in this regard but a piece of information would be nice.
My friend had a rejection for studies extension permit and he applied asylum afterwards so he can gain some time to finish his studies. Now he is a bit worried about what happens if his asylum application get rejected and leaves the country voluntarily. Once he gets back to his home country, would he be able apply for any other visa in future; for instance if he gets a phd in sweden or any other country around the world?. would it be a problem when he applies for a visa?

It would be great to have a constructive replies on the matter. Thanks.

Posted by: DavidSmitis 24.Feb.2021, 03:23 PM

There shouldn't be any problems if it is legally on the territory of the country.

Powered by Invision Power Board (http://www.invisionboard.com)
© Invision Power Services (http://www.invisionpower.com)

Printable Version of Topic

Click here to view this topic in its original format

The Local _ Studying _ Asylum After Studies

Posted by: rbarzano8 31.Dec.2020, 02:50 PM

QUOTE (rbarzano8 @ 31.Dec.2020, 03:50 PM)

** Asking for a friend**

Hello,

I don't have much knowledge in this regard but a piece of information would be nice.
My friend had a rejection for studies extension permit and he applied asylum afterwards so he can gain some time to finish his studies. Now he is a bit worried about what happens if his asylum application get rejected and leaves the country voluntarily. Once he gets back to his home country, would he be able apply for any other visa in future; for instance if he gets a phd in sweden or any other country around the world?. would it be a problem when he applies for a visa?

It would be great to have a constructive replies on the matter. Thanks.

Posted by: DavidSmitis 24.Feb.2021, 03:23 PM

Figure 31: Example of a website generated by GPT-4o, rendered across different viewports. The layout and content remain visually consistent across wide and narrow screen sizes.

5.4. Discussion

This section introduces partial frontend UI generation from designs and sketches. We propose four prompting techniques: “Merging”, “Merging with Hint”, “Fixed-Sized with Hint”, and “Varied Size with Hint”.

Our evaluation reveals that open-source models struggle in partial frontend UI generation tasks, with the best method being “Merging”. On the other hand, commercial models perform well in both tasks, especially with the “Fixed-Sized with Hint” and “Varied-Sized with Hint” methods. The open-source models in our experiments are not yet ready for such generation tasks with low performance and low consistency, mainly because of the messing out of the existing layout and the inability to provide a valid HTML after multiple generations.

We suspect that “Fixed-Sized with Hint” and “Varied-Size with Hint” perform better in commercial models because the VLLMs only need to parse a smaller image. However, in “Merging” and “Merging with Hint”, the models will need first to identify the location of the new design/sketch before the generation. This also means that providing a <missing> tag as a hint is enough for these commercial VLLMs to identify where to input the components instead of the entire image.

However, both “Fixed-Sized with Hint” and “Varied-Size with Hint” do not work well in open-source models, which we suspect is due to there being too many parts that require attention, such as the location of the new design/sketch, the <missing> tag in the partial HTML and the previous styles used in the partial HTML. These probably overwhelm the open-source VLLMs, which have much smaller parameters than the commercial models, and hence, they are not built for such complex tasks that require attention to all input components (partial HTML and design/sketch) simultaneously.

In the website builder context, we are glad that given the correct method and suitable models, the original design remains mostly unmodified with the additional changes. Moreover, the generated web pages remain responsive, given that the original HTML is already responsive. These findings show that partial design-to-code and partial sketch-to-code are practically possible and can be integrated into the current website builders.

Since we did not use any fine-tuned models in our experiments, smaller fine-tuned models may perform better in such tasks, as shown in WebCode2M, where they fine-tuned Pix2Struct with their datasets and achieved comparable results to some commercial models that they compared against, such as Claude and Gemini.

6. System Design

6.1. Implementation Scope

6.1.1. Functional Requirements

As discussed in Chapter 4.4, we plan to implement the following features to identify their technical viability:

1. Design-to-code & Sketch-to-code
    a. Develop a UI for both ordinary sketch-to-code and partial sketch-to-code.
    b. Create a corresponding backend service to handle the images and HTML sent from the UI.
2. Plugins-of-Plugins Systems
    a. Develop a system that allows users to add customisation on top of plugins. We will use the sketch-to-code plugin as an example.
    b. Implement a system that allows users to parse sketches with any LLMs they prefer.
    c. Implement a system that allows users to add custom sketches parsing logic to sketch-to-code.
3. HTML Importation Feature
    a. Have a system that allows users to convert their existing HTML code into the drag-and-drop editor and from the drag-and-drop editor back to the HTML code.
4. Automated Documentation Generation
    a. Provide a system that allows automated documentation generation as the code is created, modified or deleted over time.

These implementations are released under the MIT license to embrace the open-source spirit. By releasing the source code, our solution can have better customizability since interested parties can build and improve upon the solution.

6.1.2. Non-functional Requirements

As our proof-of-concept is a general website builder, our proposed solution shall be user-friendly to users with limited technical experience and advanced features for customisation for experienced developers. A good reference can be how Wix separates both Wix Editor and Wix Studio Editor to support the needs of both novice users and experienced developers. However, we also need to consider overcoming the restrictions of "upgrading" if the user has changed their requirements and requires more customizability. A potential solution may be a simplified UI with a collapsible menu, making it easier for novice users while still allowing advanced users to modify the details.

6.2. Design of Frontend Architecture

6.2.1. Plugin vs Modifying Core

As mentioned in WordPress Market Analysis, WordPress encourages the use of plugins, which allows updates of both WordPress and plugins separately without conflict. Hence, following this design approach, most of our proposed solutions will be built as plugins for existing website builders unless it is technically impossible. The plugin architecture also brings extra advantages, such as cross-platform availability. Our plugins for different website builders can share the same codebase or servers but with different platform-specific adapters. With this architecture, even when users leave website builders for custom websites, they can still connect to the plugin through APIs or implement a similar adapter. This architecture also helps mitigate the vendor lock problem since the plugins will be available everywhere. We discuss plugin architecture more later in this thesis, 5.4 System Architecture.

6.2.2. WordPress vs GrapesJS

WordPress, as introduced in Chapter 2.1.2, is an open-source, general-purpose dynamic website builder. With its significant plugin ecosystems and large user base, it feels like the perfect website builder as our targeted development platform. Moreover, since WordPress is a proven production-ready and battle-tested product, many features can be inherited, and there is no need to recreate the wheels.

GrapesJS, on the other hand, is a growing static website builder. It has a drag-and-drop editor that supports blocks mode, like WordPress, and absolute mode, similar to the Wix editor. Moreover, it allows import from HTML, translating the HTML elements into its internal components supporting drag and drop. However, as a growing website builder, the options provided in the editor are slightly complicated and not straightforward for non-technical users. For instance, there is no button to align the elements to the middle.

After some consideration, we built our solutions on the GrapesJS website builder. First, it is much more lightweight than WordPress as it is just a simple static website builder and does not have complex features such as dashboards or access control. It would be much easier to extend its functionality compared to WordPress. Secondly, its capability of importing HTML and CSS code provides at least two benefits that align with our project vision.

1. Users can directly modify the HTML and CSS while allowing drag and drop. This feature addresses the P18 respondent's suggestion in the user survey, where he proposed that it would be easier to implement some features in pure HTML and CSS. If a feature is more straightforward to do in HTML, the user can go directly for it.
2. It allows non-technical users to make minor modifications to the existing layout of the websites, even though they may lack technical knowledge.

However, it does not mean that we would be abandoning WordPress as an option. With the proposed plugin architecture, our solutions can work in different website builders or other platforms with suitable adapters. As proof of concept for this idea, we would first implement a plugin for GrapesJS.

6.2.3. Choice of Coding Language

As our target is a proof-of-concept website builder, we keep things simple with pure HTML and CSS. We use JavaScript for simple scripting on the front end.

6.3. Design of Backend Architecture

We require a backend service to provide some of the features for our plugins, such as sketch-to-code features to perform HTML generation with LLMs. In this section, we discuss our design decision for the backend service.

6.3.1. Choice of Coding Language

There are many languages available for backend development. There are compiled languages such as C++, Golang, and Rust, as well as interpreted languages such as Python, PHP, and JavaScript, or hybrid languages such as Java. Compiled language promises a much higher performance as the code is pre-compiled as machine code. Hence, compiled languages are usually preferred for scenarios where performance is the most important factor. However, since there are generally more AI-related tools and libraries in Python, we chose Python as the coding language for our server.

6.3.2. Choice of Tools and Frameworks

Two main backend frameworks in Python catch our attention: Flask and FastAPI. Flask is more established and has better community support, while FastAPI is a relatively young framework. However, we ultimately chose FastAPI because it can automatically generate OpenAPI documentation for all the available routes. This documentation will make it much easier for interested individuals to modify and extend the existing system.

6.4. System Architecture

As discussed, we would like our plugins to be available to multiple website builders or custom platforms. Hence, we propose having adapters for different platforms, which then talk to the plugin server to perform specific actions. In this case, our plugin server handles all the business logic, while the adapters are merely a shell that sends requests to the users and provides representations after parsing the server responses. Since the plugin server handles all communication and adapters are merely a shell for access to the features, any services that can send HTTP requests and receive HTTP responses can easily integrate into our proposed architecture as long as they communicate with the plugin server. In our proof-of-concept website builder, we implement only the plugin server and an adapter that works in GrapesJS due to time and resource limitations.

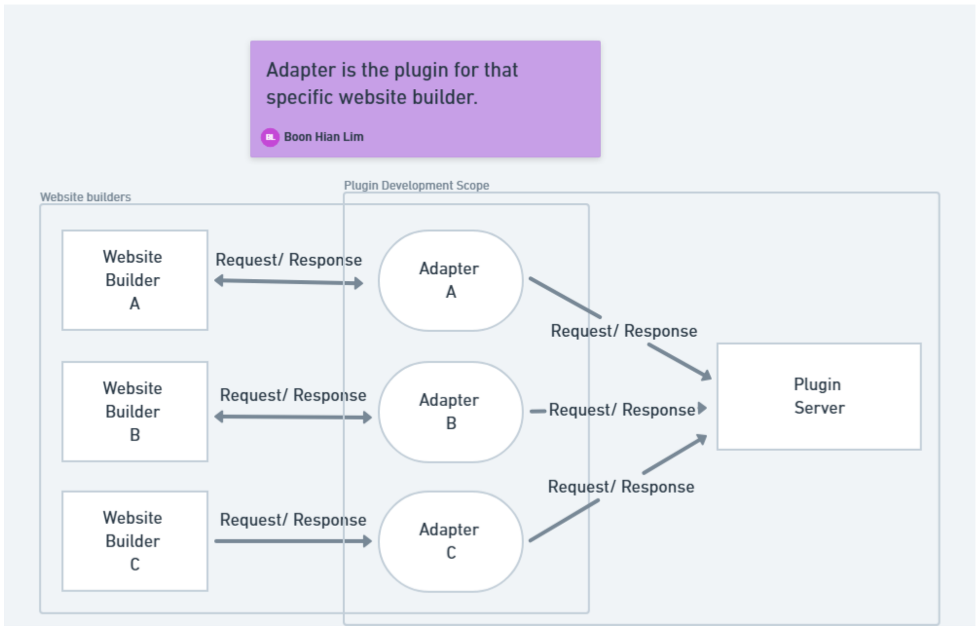


Figure 32: System Architecture of Plugin.

7. Implementation

7.1. Full Frontend UI Generation

We have established that frontend UI generation from designs or sketches is technically feasible in prior studies and our experiments in Section 4.3. This section discusses our approach to implementing full frontend UI generation in our proof-of-concept website builder.

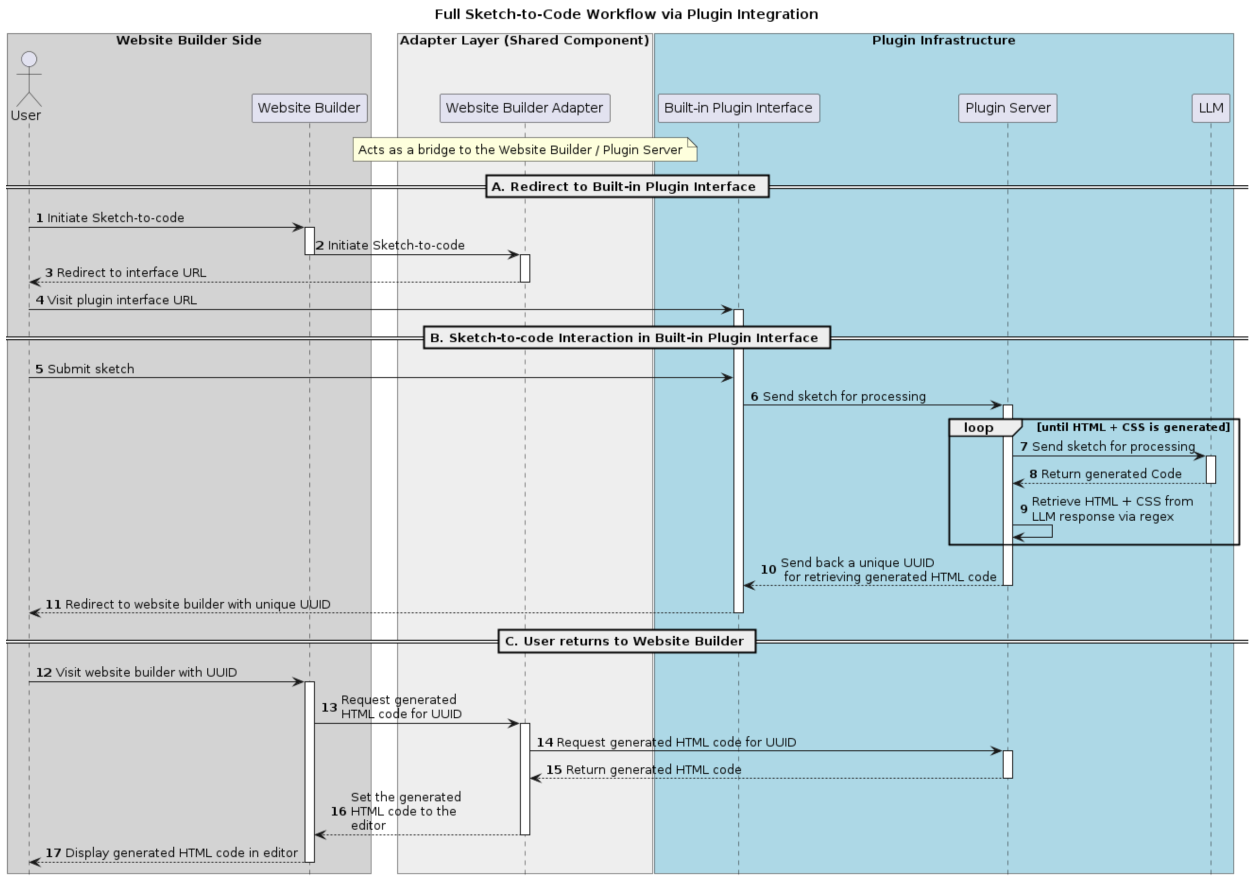

Full Sketch-to-Code Workflow via Plugin Integration
Website Builder Side
Adapter Layer (Shared Component)
Plugin Infrastructure
User
Website Builder
Website Builder Adapter
Built-in Plugin Interface
Plugin Server
LLM
Acts as a bridge to the Website Builder / Plugin Server
A. Redirect to Built-in Plugin Interface
1 Initiate Sketch-to-code
2 Initiate Sketch-to-code
3 Redirect to interface URL
4 Visit plugin interface URL
B. Sketch-to-code Interaction in Built-in Plugin Interface
5 Submit sketch
6 Send sketch for processing
loop [until HTML + CSS is generated]
7 Send sketch for processing
8 Return generated Code
9 Retrieve HTML + CSS from LLM response via regex
10 Send back a unique UUID for retrieving generated HTML code
11 Redirect to website builder with unique UUID
C. User returns to Website Builder
12 Visit website builder with UUID
13 Request generated HTML code for UUID
14 Request generated HTML code for UUID
15 Return generated HTML code
16 Set the generated HTML code to the editor
17 Display generated HTML code in editor

Figure 33: A sequence diagram showing the full sketch-to-code workflow across the website builder, adapter layer, and plugin infrastructure using a plugin server and LLM.

Figure 33 illustrates the flow for full frontend UI generation workflow in the system. To allow users to initiate the flow, we added a button to the panel on the top right labelled “Sketch to Code” in the GrapesJS drag-and-drop editor. By clicking it, users will be redirected to a page to upload their sketches and pick which LLMs to process them.

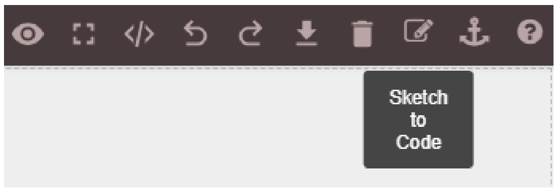
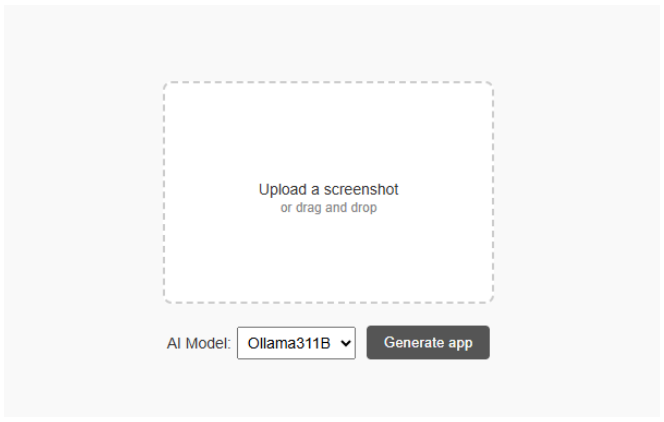


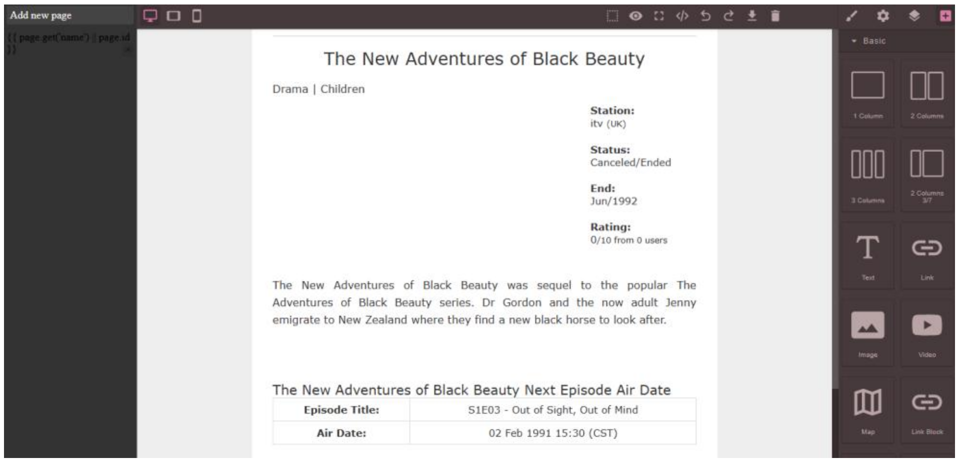
Figure 34: A screenshot of the proof-of-concept website builder, highlighting the Sketch-to-Code button on the panel's top right corner.



AI Model: Ollama311B

Figure 35: A screenshot of the sketch upload pages in the proof-of-concept website builder.

As discussed in Chapter 5.3.2, sometimes the model may not correctly generate the sketches. We will try again if our regex parser fails to extract any HTML code and ask the selected LLM to regenerate its response. After successful generation, the users will be led back to the drag-and-drop editor while the generated HTML replaces the canvas.



The New Adventures of Black Beauty

Drama | Children

Station:
itv (UK)

Status:
Canceled/Ended

End:
Jun/1992

Rating:
0/10 from 0 users

The New Adventures of Black Beauty was sequel to the popular The Adventures of Black Beauty series. Dr Gordon and the now adult Jenny emigrate to New Zealand where they find a new black horse to look after.

The New Adventures of Black Beauty Next Episode Air Date

Episode Title:	S1E03 - Out of Sight, Out of Mind
Air Date:	02 Feb 1991 15:30 (CST)

Figure 36: A screenshot of the proof-of-concept website builder using Grapes.js, displaying the generated HTML code in the drag-and-drop editor.

7.2. Partial Frontend UI Generation & Plugins-of-Plugins System

To provide insight into how the plugins-of-plugins system may work, we implemented such a system in the partial sketch-to-code functionality, allowing extensions to decide how to generate the UI from sketches, such as merging or the separate method, as introduced in Section 4.3.

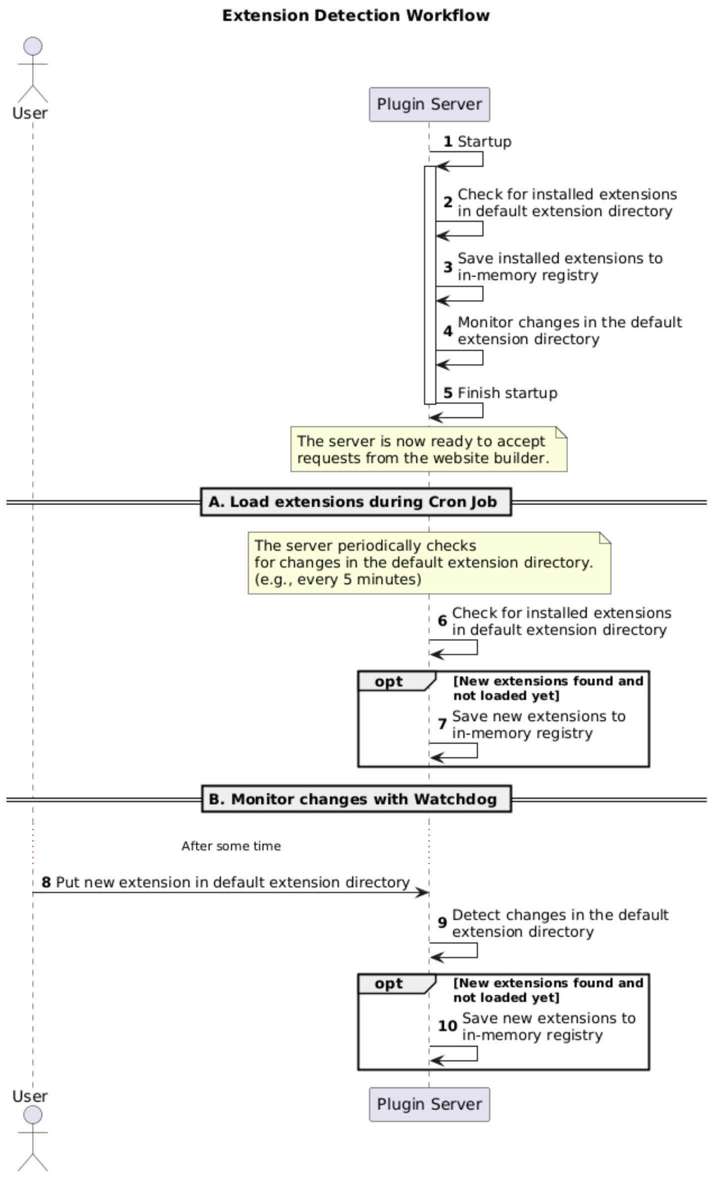
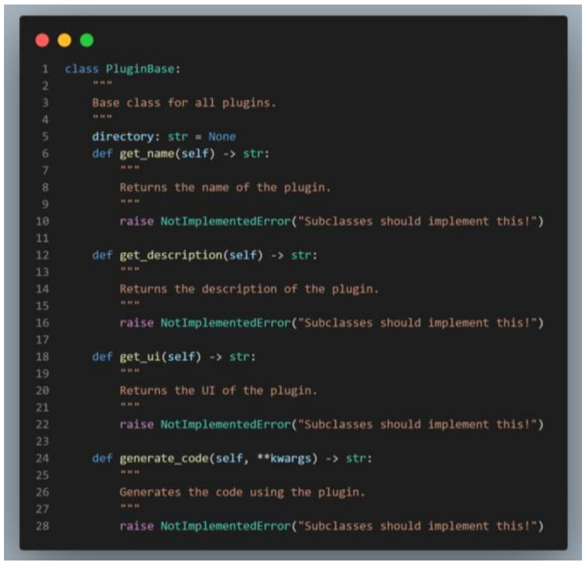

Extension Detection Workflow
User
Plugin Server
1 Startup
2 Check for installed extensions in default extension directory
3 Save installed extensions to in-memory registry
4 Monitor changes in the default extension directory
5 Finish startup
The server is now ready to accept requests from the website builder.
A. Load extensions during Cron Job
The server periodically checks for changes in the default extension directory. (e.g., every 5 minutes)
6 Check for installed extensions in default extension directory
opt
[New extensions found and not loaded yet]
7 Save new extensions to in-memory registry
B. Monitor changes with Watchdog
After some time
8 Put new extension in default extension directory
9 Detect changes in the default extension directory
opt
[New extensions found and not loaded yet]
10 Save new extensions to in-memory registry
User
Plugin Server

Figure 37: A sequence diagram illustrating how the plugin server detects and registers new extensions.

We need to tackle two main points: extension detection and extensibility. Regarding extension detection, we propose a simple system to import the extensions. During system startup, we check the default extensions directory and add them to the system before accepting incoming requests. Afterwards, we monitor changes in the said directory through a cron job (periodic check) and file system watchers to load new extensions into the memory during runtime.

For extensibility, we define a shared abstract class in Python, which defines the methods (hooks) the plugin server will use to interact with the extensions. All extensions are expected to implement this class in their main.py, which is the default entry point of the extensions. This class also has a directory attribute, which is a unique identifier that differentiates between extensions. It is assigned by the plugin server to all of the extensions when the plugin server adds them to the system.



```python
class PluginBase:
    """
    Base class for all plugins.
    """
    directory: str = None
    def get_name(self) -> str:
        """
        Returns the name of the plugin.
        """
        raise NotImplementedError("Subclasses should implement this!")

    def get_description(self) -> str:
        """
        Returns the description of the plugin.
        """
        raise NotImplementedError("Subclasses should implement this!")

    def get_ui(self) -> str:
        """
        Returns the UI of the plugin.
        """
        raise NotImplementedError("Subclasses should implement this!")

    def generate_code(self, **kwargs) -> str:
        """
        Generates the code using the plugin.
        """
        raise NotImplementedError("Subclasses should implement this!")
```

Figure 38: PluginBase, the shared interface and structure for all plugins.

The partial frontend UI generation system works similarly to the full sketch-to-code feature but with extra customisability thanks to the plugin-of-plugin system. The user first drag-and-drops to insert an image into the current canvas, and the user will be prompted to upload any image/sketch for that component. The users can then initiate the partial sketch-to-code flow by clicking the panel button and selecting the partial sketch-to-code flow they wish to initiate. After the user chooses a particular flow, the plugin server will receive that generation request and pass it to the relevant extension for further generation through the generate_code() hook. After successfully generating messages the server receives, the adapter will bring the users back to the drag-and-drop editor while the generated HTML replaces the canvas.

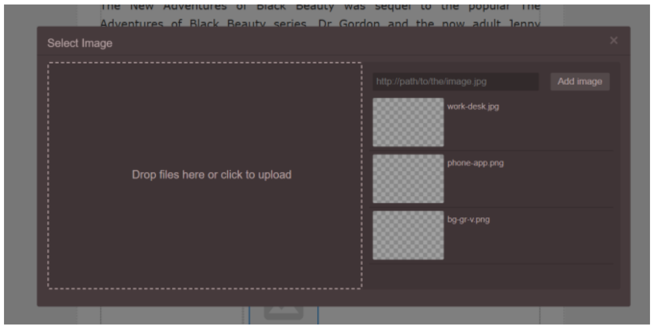

Select Image
Drop files here or click to upload
http://path/to/the/image.jpg
Add image
work-desk.jpg
phone-app.png
bg-gr-v.png

Figure 39: Image upload dialogue in the visual editor interface. Users can drag and drop files or select existing assets from a media list for use within their designs.

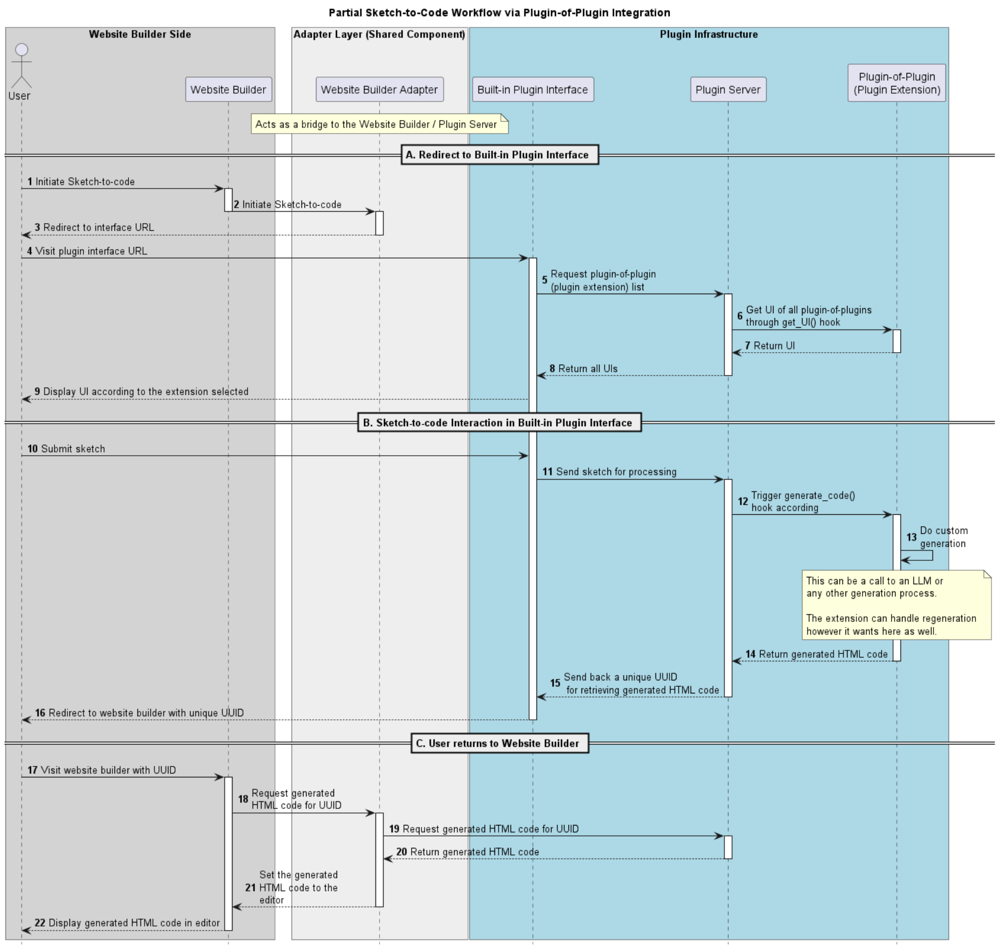
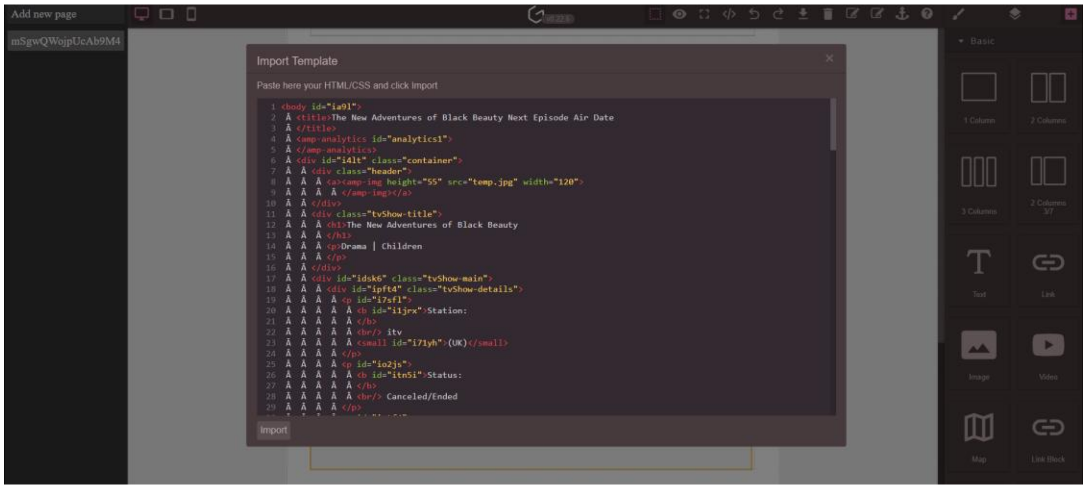

Partial Sketch-to-Code Workflow via Plugin-of-Plugin Integration
Website Builder Side
Adapter Layer (Shared Component)
Plugin Infrastructure
User
Website Builder
Website Builder Adapter
Built-in Plugin Interface
Plugin Server
Plugin-of-Plugin (Plugin Extension)
Acts as a bridge to the Website Builder / Plugin Server
A. Redirect to Built-in Plugin Interface
1 Initiate Sketch-to-code
2 Initiate Sketch-to-code
3 Redirect to interface URL
4 Visit plugin interface URL
5 Request plugin-of-plugin (plugin extension) list
6 Get UI of all plugin-of-plugins through get_UI() hook
7 Return UI
8 Return all UIs
9 Display UI according to the extension selected
B. Sketch-to-code Interaction in Built-in Plugin Interface
10 Submit sketch
11 Send sketch for processing
12 Trigger generate_code() hook according
13 Do custom generation
This can be a call to an LLM or any other generation process.
The extension can handle regeneration however it wants here as well.
14 Return generated HTML code
15 Send back a unique UUID for retrieving generated HTML code
16 Redirect to website builder with unique UUID
C. User returns to Website Builder
17 Visit website builder with UUID
18 Request generated HTML code for UUID
19 Request generated HTML code for UUID
20 Return generated HTML code
21 Set the generated HTML code to the editor
22 Display generated HTML code in editor

Figure 40: Sequence diagram of Partial Frontend UI Generation using plugin-of-plugin architecture.

7.3. HTML Importation

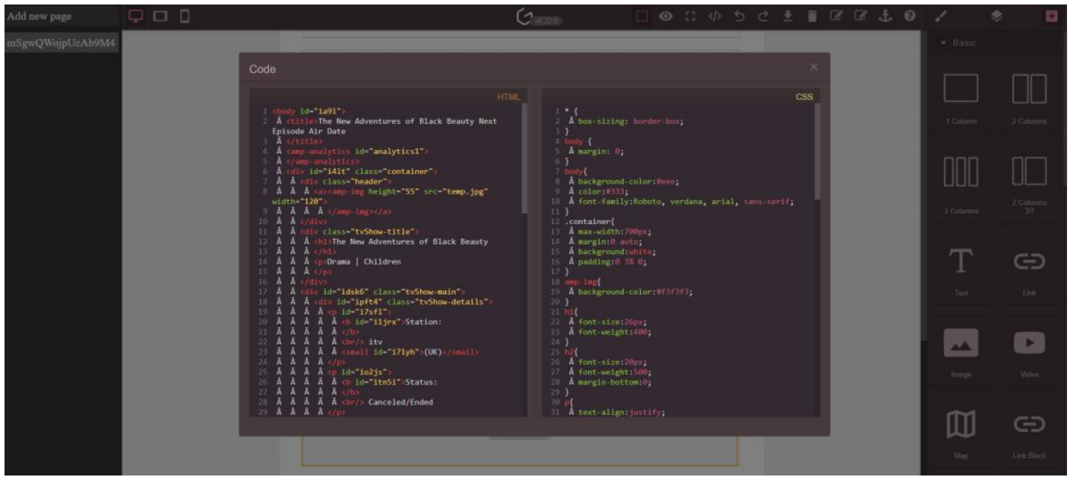
HTML importation is an in-built feature provided by GrapesJS website builder. It provides an interface for users to import any HTML codebase and match it with its internal components to convert them into components that can be drag-and-droped in the editor. Similarly, the user can retrieve the code using its side-by-side HTML and CSS code interface to obtain the HTML code of the design in the editor.



Paste here your HTML/CSS and click Import

```html
<body id="ia9l">
 Â <title>The New Adventures of Black Beauty Next Episode Air Date
 Â </title>
 Â <amp-analytics id="analytics1">
 Â </amp-analytics>
 Â <div id="i4lt" class="container">
 Â  Â <div class="header">
 Â  Â  Â <a><amp-img height="55" src="temp.jpg" width="120">
 Â  Â  Â  Â </amp-img></a>
 Â  Â </div>
 Â  Â <div class="tvShow-title">
 Â  Â  Â <h1>The New Adventures of Black Beauty
 Â  Â  Â </h1>
 Â  Â  Â <p>Drama | Children
 Â  Â  Â </p>
 Â  Â </div>
 Â  Â <div id="idsk6" class="tvShow-main">
 Â  Â  Â <div id="ipft4" class="tvShow-details">
 Â  Â  Â  Â <p id="i7sfl">
 Â  Â  Â  Â  Â <b id="i1jrx">Station:
 Â  Â  Â  Â  Â </b>
 Â  Â  Â  Â  Â <br/> itv
 Â  Â  Â  Â  Â <small id="i71yh">(UK)</small>
 Â  Â  Â  Â </p>
 Â  Â  Â  Â <p id="io2js">
 Â  Â  Â  Â  Â <b id="itn5i">Status:
 Â  Â  Â  Â  Â </b>
 Â  Â  Â  Â  Â <br/> Canceled/Ended
 Â  Â  Â  Â </p>
```

Figure 41: Template import interface in GrapesJS website builder. Users can paste raw HTML/CSS into the editor to import and render structured web content.

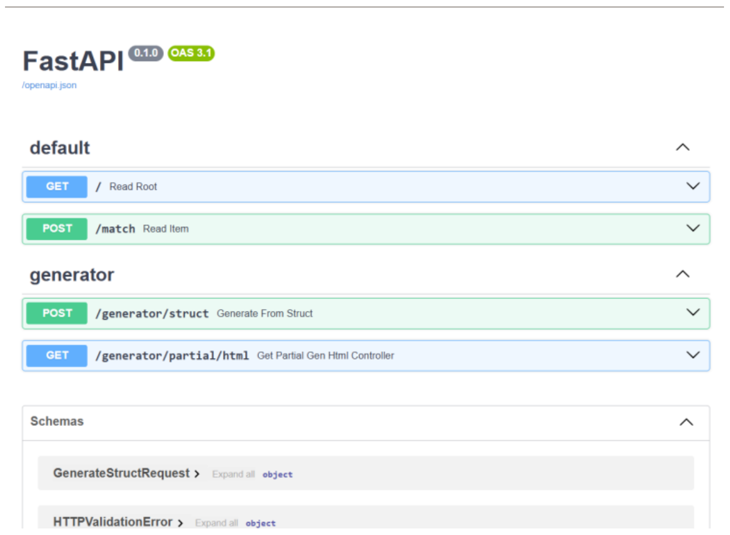


```html
<body id="ia9l">
 Â <title>The New Adventures of Black Beauty Next
Episode Air Date
 Â </title>
 Â <amp-analytics id="analytics1">
 Â </amp-analytics>
 Â <div id="i4lt" class="container">
 Â  Â <div class="header">
 Â  Â  Â <a><amp-img height="55" src="temp.jpg"
width="120">
 Â  Â  Â  Â </amp-img></a>
 Â  Â </div>
 Â  Â <div class="tvShow-title">
 Â  Â  Â <h1>The New Adventures of Black Beauty
 Â  Â  Â </h1>
 Â  Â  Â <p>Drama | Children
 Â  Â  Â </p>
 Â  Â </div>
 Â  Â <div id="idsk6" class="tvShow-main">
 Â  Â  Â <div id="ipft4" class="tvShow-details">
 Â  Â  Â  Â <p id="i7sfl">
 Â  Â  Â  Â  Â <b id="i1jrx">Station:
 Â  Â  Â  Â  Â </b>
 Â  Â  Â  Â  Â <br/> itv
 Â  Â  Â  Â  Â <small id="i71yh">(UK)</small>
 Â  Â  Â  Â </p>
 Â  Â  Â  Â <p id="io2js">
 Â  Â  Â  Â  Â <b id="itn5i">Status:
 Â  Â  Â  Â  Â </b>
 Â  Â  Â  Â  Â <br/> Canceled/Ended
 Â  Â  Â  Â </p>
```

```css
* {
 Â box-sizing: border-box;
}
body {
 Â margin: 0;
}
body{
 Â background-color:#eee;
 Â color:#333;
 Â font-family:Roboto, verdana, arial, sans-serif;
}
.container{
 Â max-width:700px;
 Â margin:0 auto;
 Â background:white;
 Â padding:0 3% 0;
}
amp-img{
 Â background-color:#f3f3f3;
}
h1{
 Â font-size:26px;
 Â font-weight:400;
}
h2{
 Â font-size:20px;
 Â font-weight:500;
 Â margin-bottom:0;
}
p{
 Â text-align:justify;
```

Figure 42: Side-by-side HTML and CSS code interface in a visual web builder. Users can view and retrieve both structure and styling here.

7.4. Automated Documentation Generation


FastAPI 0.1.0 OAS 3.1
/openapi.json

default

GET / Read Root

POST /match Read Item

generator

POST /generator/struct Generate From Struct

GET /generator/partial/html Get Partial Gen Html Controller

Schemas

GenerateStructRequest > Expand all object

HTTPValidationError > Expand all object

Figure 43: OpenAPI 3.1 documentation interface automatically generated by FastAPI.

By using FastAPI, we have automatically generated OpenAPI documentation that can be accessed through the route “/docs” in the server. This generation is possible as FastAPI is based on the OpenAPI specification [78]. Since OpenAPI is a common standard used in the industry for API documentation, existing tools in the market can automatically generate clients to interact with the server, such as OpenAPI Generator [79] or openapi-ts [80].

8. Conclusion

8.1. Summary of Achievement

We have successfully identified limitations in website builders, with the principal recurring limitation being a lack of customizability. The others include editor issues, lack of TypeScript support, accessibility issues, poor documentation, and limited integration with external tools raised in the case study and user survey. Hence, we propose having sketch-to-code, plugins-of-plugins functionality and HTML importation feature to tackle customizability. Regarding the versioning tool problem, we advise having a versioning tool integrate with Git since it is the industry standard in traditional website development.

Through our experiments, we verify that partial UI generation from design and sketches is technically feasible. The best models in our experiments are Gemini2.0-flash or GPT-4o, using the "Fixed-Sized with Hint" and "Varied-Sized with Hint" methods. Commercial models outperform open-source models in both partial sketch-to-code and partial design-to-code. We suspect this contributes to the lack of adaptation of such features since commercial models may have varied pricing depending on the vendors. We also found that the web pages generated from the models generally follow Responsive Web Design, which is proved by the low variance across viewports.

We further prove that all of our proposed features, UI generation from sketches, plugins-of-plugins functionality, HTML importation feature and automated documentation generation feature, are all engineering possible by implementing them in our proof-of-concept website builder. We build some of these features as plugins on GrapesJS. Some of these features are built into our chosen tools, such as the HTML importation feature in GrapesJS that maps the HTML tag to its internal drag-and-drop components. Releasing our solutions as open-source provides customizability as users can modify, install and uninstall them accordingly without hassle. Users can further build their adapters if they wish to use our proposed solutions on some unsupported platforms.

8.2. Future Work

We have discussed many topics in this thesis, and each can be further explored, considering that not all the limitations and exploration are covered. We propose that future research should explore the following directions:

8.2.1. Experiment

1. The current open-source models we use are relatively small. Those models with higher parameters may perform better in partial sketch-to-code tasks.
2. The current testing dataset for partial frontend UI generation is relatively small compared to the existing study in the entire generation. Future studies can propose an automated pipeline to produce more designs and sketches to evaluate LLMs' performance.
3. We have not fine-tuned any models we use for the partial sketch-to-code task. Fine-tuned models may achieve a better performance than what we have now.
4. Future research can explore generating multiple components in different parts of the Document Object Model (DOM) tree at the same time and evaluate the effectiveness of LLMs.
5. While we establish that the code generated follows RWD, the websites in our datasets are also responsive, which is not the case for all websites. Given websites that do not follow RWD, it is unclear whether the LLMs can generate a website with responsiveness in mind.

8.2.2. Implementation

1. We build our prototype on GrapesJS, a static website builder. Implementing similar ideas like sketch-to-code or plugins-of-plugins in a dynamic website builder like WordPress or Wix may bring unique challenges.
2. Our prototype relies on GrapesJS and hence inherits limitations from GrapesJS as well. For instance, its drag-and-drop editor is unreliable. When switched to absolute-position mode, the components placed will conflict with those placed in block mode and overlap. Future work can focus on improving the GrapesJS editor for better usability or changing to using other website builders that provide a better drag-and-drop experience.

Reference

[1] E. Ageeva, T. C. Melewar, P. Foroudi, and C. Dennis, “Evaluating the factors of corporate website favorability: a case of UK and Russia,” *Qualitative Market Research*, vol. 22, no. 5, pp. 687–715, 2019, doi: 10.1108/QMR-09-2017-0122.

[2] A. H. S. Ng, “What website builders can and cannot do,” 2023, Accessed: Aug. 31, 2024. [Online]. Available: https://hdl.handle.net/10356/171901

[3] “Wix Editor Constantly Crashing - Ask the community,” Community Support Forum | Wix Studio. Accessed: Jan. 21, 2025. [Online]. Available: https://forum.wixstudio.com/t/wix-editor-constantly-crashing/21617

[4] J. K. L. Yeo, “What website builders can and cannot do,” 2024, Accessed: Aug. 31, 2024. [Online]. Available: https://hdl.handle.net/10356/175248

[5] A. Ko, “How to Create a Website From Scratch in 11 Steps (for Beginners),” Wix Blog. Accessed: Mar. 09, 2025. [Online]. Available: https://www.wix.com/blog/how-to-build-website-from-scratch-guide

[6] J. Hitchcock, “What Is Shopify and How Does It Work? (2025) - Shopify Singapore,” Shopify. Accessed: Jan. 24, 2025. [Online]. Available: https://www.shopify.com/sg/blog/what-is-shopify

[7] Medium, “About Medium,” Medium. Accessed: Jan. 24, 2025. [Online]. Available: https://medium.com/about

[8] Blackboard, “Blackboard Learn - Low-maintenance. Modern. A new kind of LMS. | Anthology.” Accessed: Jan. 24, 2025. [Online]. Available: https://www.anthology.com/en-apac/products/teaching-and-learning/learning-effectiveness/blackboard-learn

[9] J. Forbes, “Web Builders and the Future of the Web Development Industry,” 2019, Accessed: Mar. 08, 2025. [Online]. Available: http://www.theseus.fi/handle/10024/260767

[10] Y. Gui *et al.*, “WebCode2M: A Real-World Dataset for Code Generation from Webpage Designs,” Feb. 22, 2025. doi: 10.1145/3696410.3714889.

[11] C. Si, Y. Zhang, R. Li, Z. Yang, R. Liu, and D. Yang, “Design2Code: Benchmarking Multimodal Code Generation for Automated Front-End Engineering,” Feb. 09, 2025, *arXiv*: arXiv:2403.03163. doi: 10.48550/arXiv.2403.03163.

[12] J. DeRosa, “Part 1 Getting Started with DIY Websites,” in *Building DIY Websites for Dummies*, United States: John Wiley & Sons, Incorporated, 2024.

[13] “40% of the web uses WordPress,” W3Tech. Accessed: Jan. 26, 2025. [Online]. Available: https://w3techs.com/blog/entry/40_percent_of_the_web_uses_wordpress

[14] J. Renouard, “Website Builders Market Share: A Deep Dive,” Website Builder Expert. Accessed: Aug. 31, 2024. [Online]. Available: https://www.websitebuilderexpert.com/website-builders/website-builders-market-share/

[15] BuiltWith, “BuiltWith,” BuiltWith. Accessed: Jan. 21, 2025. [Online]. Available: https://builtwith.com/

[16] BuiltWith, “CMS technologies Web Usage Distribution on the Entire Internet,” CMS Usage Distribution on the Entire Internet. Accessed: Jan. 21, 2025. [Online]. Available: https://trends.builtwith.com/cms/traffic/Entire-Internet

[17] WordPress, *WordPress*. (Jan. 25, 2025). PHP. WordPress. Accessed: Jan. 26, 2025. [Online]. Available: https://github.com/WordPress/WordPress

[18] Wix, “Your website, your business, your future ∣ Wix.com,” wix.com. Accessed: Aug. 31, 2024. [Online]. Available: https://www.wix.com/

[19] “About,” WordPress.org. Accessed: Aug. 31, 2024. [Online]. Available: https://wordpress.org/about/

[20] WordPress, “WordPress Block Editor – Documentation.” Accessed: Jan. 26, 2025. [Online]. Available: https://wordpress.org/documentation/article/wordpress-block-editor/

[21] WordPress, “Roles and Capabilities,” Documentation. Accessed: Jan. 26, 2025. [Online]. Available: https://wordpress.org/documentation/article/roles-and-capabilities/

[22] “About Us,” WordPress.com. Accessed: Jan. 26, 2025. [Online]. Available: https://wordpress.com/about/

[23] “WordPress Hosting by WP Engine,” WP Engine. Accessed: Jan. 26, 2025. [Online]. Available: https://wpengine.com/wordpress-hosting/

[24] “Managed Hosting for WordPress | Fast Loading & Secure,” Hostinger. Accessed: Jan. 26, 2025. [Online]. Available: https://www.hostinger.com/wordpress-hosting

[25] “Amazon Lightsail - Launch a WordPress Site in Minutes,” Amazon Web Services, Inc. Accessed: Jan. 26, 2025. [Online]. Available: https://aws.amazon.com/lightsail/projects/wordpress/

[26] BuiltWith, “WP Engine Usage Statistics.” Accessed: Jan. 26, 2025. [Online]. Available: https://trends.builtwith.com/hosting/WP-Engine

[27] “About WP Engine,” WP Engine. Accessed: Jan. 26, 2025. [Online]. Available: https://wpengine.com/about-us/

[28] “Introduction to Plugin Development – Plugin Handbook | Developer.WordPress.org,” WordPress Developer Resources. Accessed: Jan. 26, 2025. [Online]. Available: https://developer.wordpress.org/

[29] Jordy Meow, “Media File Renamer: Rename for better SEO (AI-Powered),” WordPress.org. Accessed: Aug. 31, 2024. [Online]. Available: https://wordpress.org/plugins/media-file-renamer/

[30] Jordy Meow, “AI Engine,” WordPress.org. Accessed: Aug. 31, 2024. [Online]. Available: https://wordpress.org/plugins/ai-engine/

[31] “WooCommerce Bookings - Booking & Reservations WordPress Plugin,” WooCommerce. Accessed: Jan. 26, 2025. [Online]. Available: https://woocommerce.com/products/woocommerce-bookings/

[32] “WooCommerce Subscriptions,” WooCommerce. Accessed: Jan. 26, 2025. [Online]. Available: https://woocommerce.com/products/woocommerce-subscriptions/

[33] “Wix Blog: Importing Blog Posts from WordPress to the Wix Blog | Help Center | Wix.com.” Accessed: Jan. 27, 2025. [Online]. Available: https://support.wix.com/en/article/wix-blog-importing-blog-posts-from-wordpress-to-the-wix-blog

[34] ReCorp, “Export WP Page to Static HTML/CSS,” WordPress.org. Accessed: Jan. 27, 2025. [Online]. Available: https://wordpress.org/plugins/export-wp-page-to-static-html/

[35] E. Halim, M. Hebrard, H. Hartono, K. O. Halim, and W. Russel, “Exploration WordPress as E-Commerce RAD-CMS for SMEs in Indonesia,” in *2020 International Conference on Information Management and Technology (ICIMTech)*, Aug. 2020, pp. 818–823. doi: 10.1109/ICIMTech50083.2020.9211122.

[36] A. H. Kusumo, “Has Website Design using Website Builder Fulfilled Usability Aspects? A Study Case of Three Website Builders,” in *Proceedings of the 4th International Conference on Informatics, Technology and Engineering 2023 (InCITE 2023)*, Yogyakarta: Atlantis Highlights in Engineering, Nov. 2023, pp. 545–557. Accessed: Jan. 21, 2025. [Online]. Available: https://doi.org/10.2991/978-94-6463-288-0_45

[37] S. K. Oswal and H. K. Oswal, “Examining the Accessibility of Generative AI Website Builder Tools for Blind and Low Vision Users: 21 Best Practices for Designers and Developers,” in *2024 IEEE International Professional Communication Conference (ProComm),* Jul. 2024, pp. 121–128. doi: 10.1109/ProComm61427.2024.00030.

[38] Z. B. Palmer and S. K. Oswal, “Constructing Websites with Generative AI Tools: The Accessibility of Their Workflows and Products for Users With Disabilities,” *Journal of Business and Technical Communication,* vol. 39, no. 1, pp. 93–114, Jan. 2025, doi: 10.1177/10506519241280644.

[39] Y. Jiang, J. Gao, J. Zhang, L.-X. Zhu, and R.-Y. Zhang, “Research on Tag-Styled Website Builder System,” *Jisuanji Xitong Yingyong = Computer Systems and Applications,* no. 7, 2017.

[40] Anton. A. Sutchenkov and A. I. Tikhonov, “Site Generator for Small Open and Private Online Courses,” in *2020 International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE)*, Mar. 2020, pp. 1–5. doi: 10.1109/REEPE49198.2020.9059197.

[41] Mamatha SK, Gurukiran AC, Basavaraj, and Meghana, “NO-CODE WEB DEVELOPMENT,” *IRJMETS*.

[42] H. Petersen, “From Static and Dynamic Websites to Static Site Generators”.

[43] D. Rathore and N. Singhal, “Web Design Dilemma: A Comprehensive Guide to Adaptive and Responsive Design,” in *2024 IEEE International Students’ Conference on Electrical, Electronics and Computer Science (SCEECS)*, Feb. 2024, pp. 1–5. doi: 10.1109/SCEECS61402.2024.10482052.

[44] N. Yousaf, A. Arshad, M. Nouman, and U. Arshad, “Towards adaptive and responsive web design: A systematic literature review,” *Language*, vol. 1, p. 40, 2018.

[45] G. Bekmanova *et al.*, “Requirements for the Development of a Website Builder with Adaptive Design,” in *2024 9th International Conference on Computer Science and Engineering (UBMK)*, Oct. 2024, pp. 265–270. doi: 10.1109/UBMK63289.2024.10773412.

[46] Wix, “Wix Editor Request: Responsive Sites | Help Center | Wix.com,” Wix Editor Request: Responsive Sites. Accessed: Jan. 25, 2025. [Online]. Available: https://support.wix.com/en/article/wix-editor-request-responsive-sites

[47] J. A. Landay and B. A. Myers, “Sketching interfaces: toward more human interface design,” *Computer*, vol. 34, no. 3, pp. 56–64, Mar. 2001, doi: 10.1109/2.910894.

[48] J. Seifert, B. Pfleging, E. del Carmen Valderrama Bahamóndez, M. Hermes, E. Rukzio, and A. Schmidt, “Mobidev: a tool for creating apps on mobile phones,” in *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, in MobileHCI ’11. New York, NY, USA: Association for Computing Machinery, Aug. 2011, pp. 109–112. doi: 10.1145/2037373.2037392.

[49] A. Robinson, “Sketch2code: Generating a website from a paper mockup,” May 09, 2019, *arXiv*: arXiv:1905.13750. doi: 10.48550/arXiv.1905.13750.

[50] V. Jain, P. Agrawal, S. Banga, R. Kapoor, and S. Gulyani, “Sketch2Code: Transformation of Sketches to UI in Real-time Using Deep Neural Network,” Oct. 20, 2019, *arXiv*: arXiv:1910.08930. doi: 10.48550/arXiv.1910.08930.

[51] K. Lee *et al.*, “Pix2Struct: Screenshot Parsing as Pretraining for Visual Language Understanding,” Jun. 15, 2023, *arXiv*: arXiv:2210.03347. doi: 10.48550/arXiv.2210.03347.

[52] R. Li, Y. Zhang, and D. Yang, “Sketch2Code: Evaluating Vision-Language Models for Interactive Web Design Prototyping,” Oct. 21, 2024, *arXiv*: arXiv:2410.16232. doi: 10.48550/arXiv.2410.16232.

[53] “关于 FOGG,” FOGG Boon Hian Copy. Accessed: Mar. 22, 2025. [Online]. Available: https://www.fogg.com.sg/en/about

[54] A. A. Almazroi, “A Systematic Mapping Study of Software Usability Studies,” *IJACSA*, vol. 12, no. 9, 2021, doi: 10.14569/IJACSA.2021.0120927.

[55] “Wix Multilingual | Wix App Market | Wix.com.” Accessed: Mar. 23, 2025. [Online]. Available: https://www.wix.com/app-market/wix-multilingual?searchLocation=standalone-header

[56] Wix, “Wix Multilingual Request: Changing Your Site’s Main Language | Help Center | Wix.com.” Accessed: Jan. 26, 2025. [Online]. Available: https://support.wix.com/en/article/wix-multilingual-request-changing-your-sites-main-language

[57] Wix, “Wix Bookings | Wix App Market | Wix.com,” Wix Bookings. Accessed: Jan. 25, 2025. [Online]. Available: https://www.wix.com/app-market/web-solution/bookings

[58] Wix, “About Velo by Wix.” Accessed: Jan. 26, 2025. [Online]. Available: https://dev.wix.com/docs/develop-websites/articles/getting-started/about-velo-by-wix

[59] Wix, “About Velo.” Accessed: Jan. 26, 2025. [Online]. Available: https://dev.wix.com/docs/velo/articles/getting-started/about-velo

[60] A. Amery, “Wix vs. Wix Studio: Which should you choose?,” Wix Blog. Accessed: Jan. 26, 2025. [Online]. Available: https://www.wix.com/blog/wix-vs-wix-studio

[61] “About the Wix Bookings API.” Accessed: Mar. 23, 2025. [Online]. Available: https://dev.wix.com/docs/velo/apis/wix-bookings-v2/introduction

[62] “Developer Preview.” Accessed: Mar. 23, 2025. [Online]. Available: https://dev.wix.com/docs/velo/articles/api-overview/developer-preview

[63] “Wix Editor: Adding a Custom Element to Your Site | Help Center | Wix.com.” Accessed: Mar. 23, 2025. [Online]. Available: https://support.wix.com/en/article/wix-editor-adding-a-custom-element-to-your-site

[64] “Using custom elements - Web APIs | MDN.” Accessed: Mar. 23, 2025. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Web_components/Using_custom_elements

[65] K. Goldstein, “Wix ADI: How Design AI Elevates Website Creation for Everyone,” Wix Blog. Accessed: Aug. 31, 2024. [Online]. Available: https://www.wix.com/blog/wix-artificial-design-intelligence

[66] “Freeze or Crashing Editor - Ask the community,” Community Support Forum | Wix Studio. Accessed: Jan. 26, 2025. [Online]. Available: https://forum.wixstudio.com/t/freeze-or-crashing-editor/7194

[67] “Wix Editor Crashing? - Ask the community,” Community Support Forum | Wix Studio. Accessed: Jan. 26, 2025. [Online]. Available: https://forum.wixstudio.com/t/wix-editor-crashing/36384

[68] “Editor Performance: Best Practices | Help Center | Wix.com.” Accessed: Jan. 26, 2025. [Online]. Available: https://support.wix.com/en/article/editor-performance-best-practices

[69] “Viewing and Managing Your Site History | Help Center | Wix.com.” Accessed: Mar. 23, 2025. [Online]. Available: https://support.wix.com/en/article/viewing-and-managing-your-site-history

[70] “What is a Lottie animation?,” LottieFiles. Accessed: Mar. 24, 2025. [Online]. Available: https://lottiefiles.com/what-is-lottie

[71] “Studio Editor: About CSS Editing | Help Center | Wix.com.” Accessed: Mar. 24, 2025. [Online]. Available: https://support.wix.com/en/article/studio-editor-about-css-editing

[72] “Importing Figma designs into Builder,” Builder.io. Accessed: Mar. 24, 2025. [Online]. Available: https://www.builder.io/c/docs/import-from-figma

[73] *versionpress/versionpress*. (Jan. 17, 2025). PHP. VersionPress. Accessed: Jan. 27, 2025. [Online]. Available: https://github.com/versionpress/versionpress

[74] Infragistics, “Design-to-Code Tool For High-Performance Apps,” App Builder. Accessed: Mar. 24, 2025. [Online]. Available: https://www.appbuilder.dev/design-to-code

[75] *uncss/uncss*. (Mar. 30, 2025). JavaScript. UnCSS. Accessed: Apr. 01, 2025. [Online]. Available: https://github.com/uncss/uncss

[76] A. Radford *et al.*, “Learning Transferable Visual Models From Natural Language Supervision,” Feb. 26, 2021, *arXiv*: arXiv:2103.00020. doi: 10.48550/arXiv.2103.00020.

[77] A. Telea, “An Image Inpainting Technique Based on the Fast Marching Method,” *Journal of Graphics Tools*, vol. 9, no. 1, pp. 23–34, Jan. 2004, doi: 10.1080/10867651.2004.10487596.

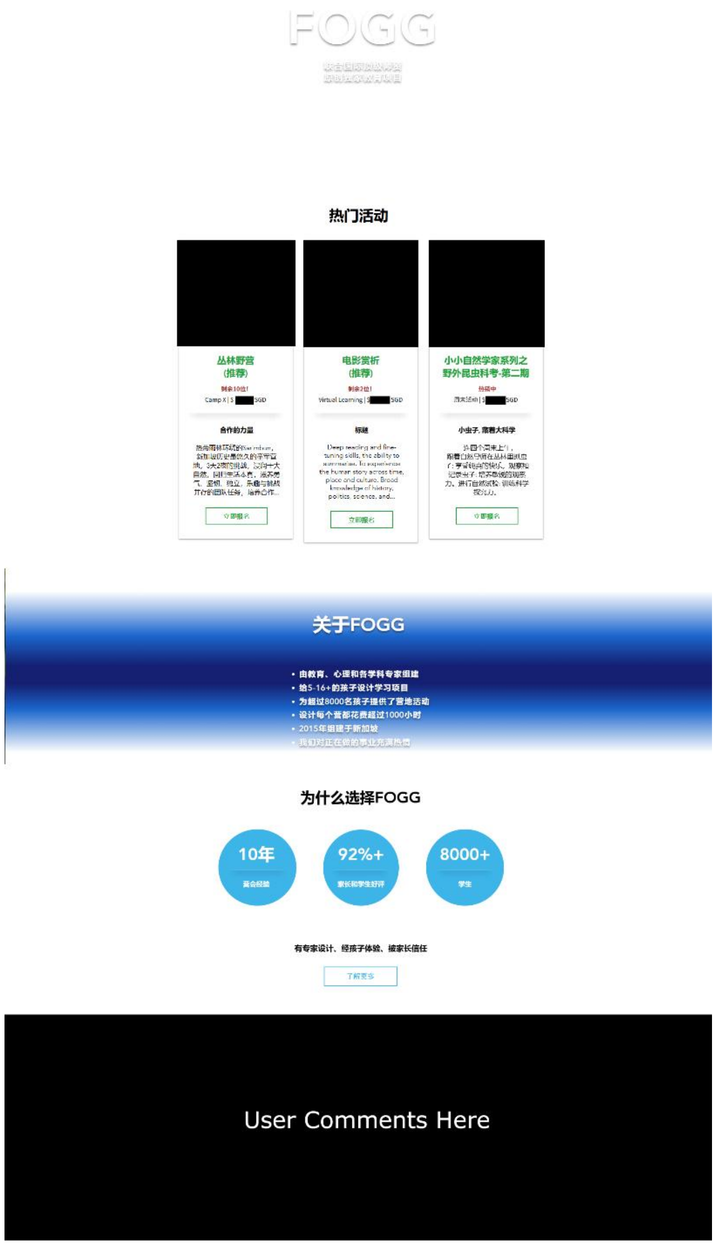
[78] “Generate Clients - FastAPI.” Accessed: Apr. 02, 2025. [Online]. Available: https://fastapi.tiangolo.com/advanced/generate-clients/

[79] “Hello from OpenAPI Generator | OpenAPI Generator.” Accessed: Apr. 02, 2025. [Online]. Available: https://openapi-generator.tech/

[80] *hey-api/openapi-ts*. (Apr. 02, 2025). TypeScript. Hey API. Accessed: Apr. 02, 2025. [Online]. Available: https://github.com/hey-api/openapi-ts

Appendices

A. Screenshots of Revamped FOGG Web Pages



FOGG

热门活动

[redacted]

丛林野营
(推荐)

剩余10位！

Camp X | $[redacted]SGD

合作的力量

热带雨林环境的Sarimbun，新加坡历史最悠久的露营营地，3大2夜的挑战，回归十大自然，回归生活本真，感受勇气、坚韧、独立，乐趣与挑战并存的团队任务，培养合作...

立即报名

[redacted]

电影赏析
(推荐)

剩余2位！

Virtual Learning | $[redacted]SGD

标题

Deep reading and fine-tuning skills, the ability to summarise. To experience the human story across time, place and culture. Broad knowledge of history, politics, science, and...

立即报名

[redacted]

小小自然学家系列之
野外昆虫科考-第二期

热销中

周末活动 | $[redacted]SGD

小虫子，藏着大科学

共四个周末上门，跟着自然导师在丛林里抓虫子：享受纯真的快乐，观察和记录虫子：培养敏锐的观察力，进行自然试验：训练科学探究力。

立即报名

关于FOGG

- 由教育、心理和各学科专家组建
- 给5-16+的孩子设计学习项目
- 为超过8000名孩子提供了营地活动
- 设计每个营都花费超过1000小时
- 2015年组建于新加坡
- 我们对正在做的事业充满热情

为什么选择FOGG

10年
营会经验

92%+
家长和学生好评

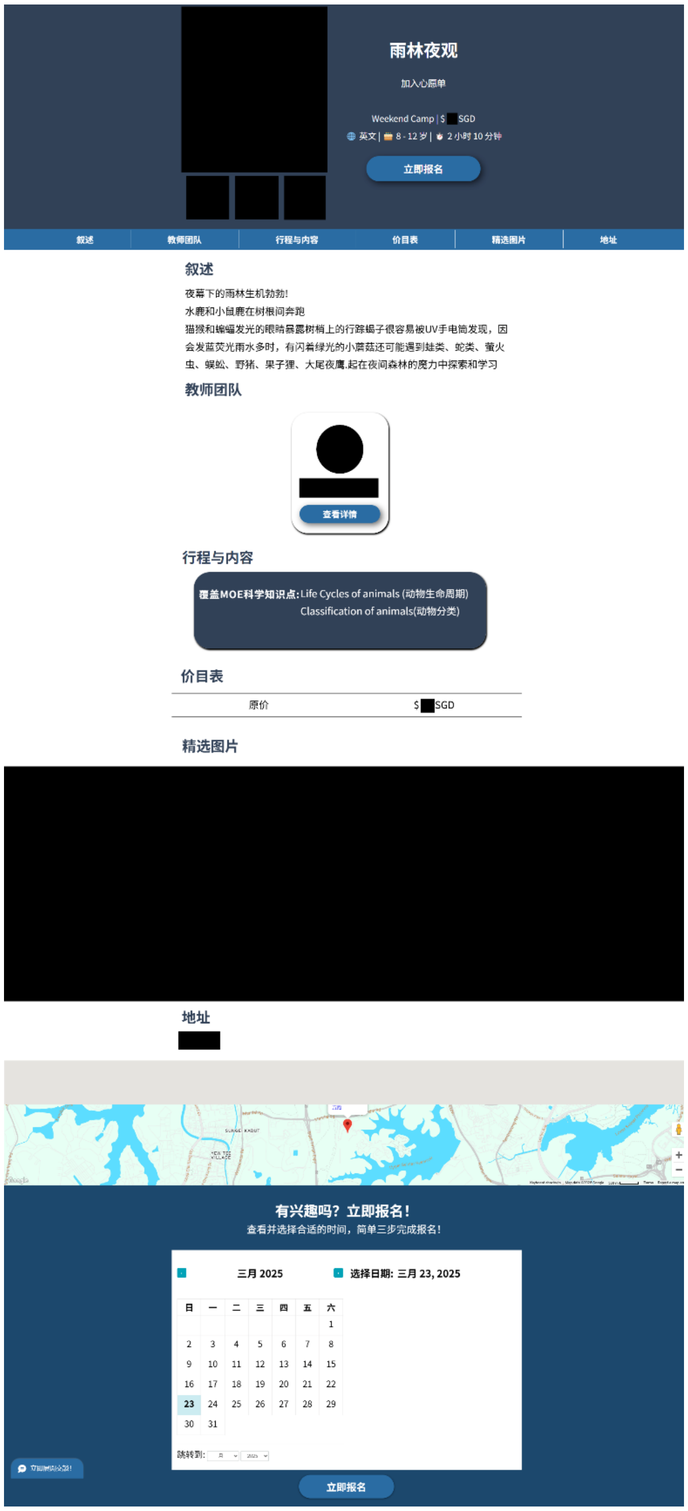
8000+
学生

有专家设计、经孩子体验、被家长信任

了解更多

User Comments Here

Figure 44: Screenshot of the revamped main page with sensitive or identifiable content redacted or replaced for privacy.



[redacted] [redacted] [redacted] [redacted]

雨林夜观

Weekend Camp | $ [redacted] SGD
英文 | 8 - 12 岁 | 2 小时 10 分钟

叙述

夜幕下的雨林生机勃勃!
水鹿和小鼠鹿在树根间奔跑
猫猴和蝙蝠发光的眼睛暴露树梢上的行踪蝎子很容易被UV手电筒发现，因会发蓝荧光雨水多时，有闪着绿光的小蘑菇还可能遇到蛙类、蛇类、萤火虫、蜈蚣、野猪、果子狸、大尾夜鹰.起在夜间森林的魔力中探索和学习

教师团队

[redacted] [redacted]

行程与内容

覆盖MOE科学知识点: Life Cycles of animals (动物生命周期)
Classification of animals(动物分类)

价目表

原价	$ [redacted] SGD

精选图片

[redacted]

地址

[redacted]

有兴趣吗? 立即报名!
查看并选择合适的时间，简单三步完成报名!

三月 2025 选择日期: 三月 23, 2025

日	一	二	三	四	五	六
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Figure 45: Screenshot of the revamped camp details page with sensitive or identifiable content redacted or replaced for privacy.

B. Questions in the User Survey

1. *How many years of experience do you have in general web development (including website builders, HTML, CSS, Javascript, React, Vue or any kind of web development technology)? [MCQ: 0 (No Experience At All), 1-2 years, 3-5 years, 6-10 years, More than 10 years]*
2. *How many years of experience do you have in custom-coded (non-website builder) web development (e.g., writing HTML, CSS, JavaScript, backend programming)? [MCQ: 0 (No Experience At All), 1-2 years, 3-5 years, 6-10 years, More than 10 years]*
3. *How many years of experience do you have with website builders (No code platform e.g. Wix, WordPress, Webflow, Shopify, etc.)? [MCQ: 0 (No Experience At All), 1-2 years, 3-5 years, 6-10 years, More than 10 years]*
4. *Which website builders have you used before? (Select all that apply) [Multiple Options: WordPress, Wix, SquareSpace, Weebly, Shopify, Webflow, Google Sites, Others]*
5. *What do you use website builders for? (Select all that apply) [Multiple Options: Building a Small / Medium Business (SME) website, Creating an e-commerce store, Building a portfolio website, Blogging, Personal / School projects, Prototypes, Other]*
6. *How important or not important are the following reasons for your use of website builders? [5-point Likert scale, ranging from Not Important to Very Important. Options: Drag-and-drop editor, Pre-built templates, SEO optimization features, Built-in hosting and domain management, Plugins, Faster development time, Lower cost compared to hiring a developer]*
7. *Are there any other reasons why you choose to use a website builder? [Open-ended]*
8. *How significant or not significant are the following limitations in your experience with website builders? [5-point Likert scale, ranging from Not a Problem to Critical Issue. Options: Poor customization, Poor editing experience, Vendor lock (Hard to shift away from current platform), Performance concerns regarding the created websites, Limited integrations with external tools, Poor history tracking features, Poor documentation]*
9. *Are there any additional limitations you face when using website builders? [Open-ended]*

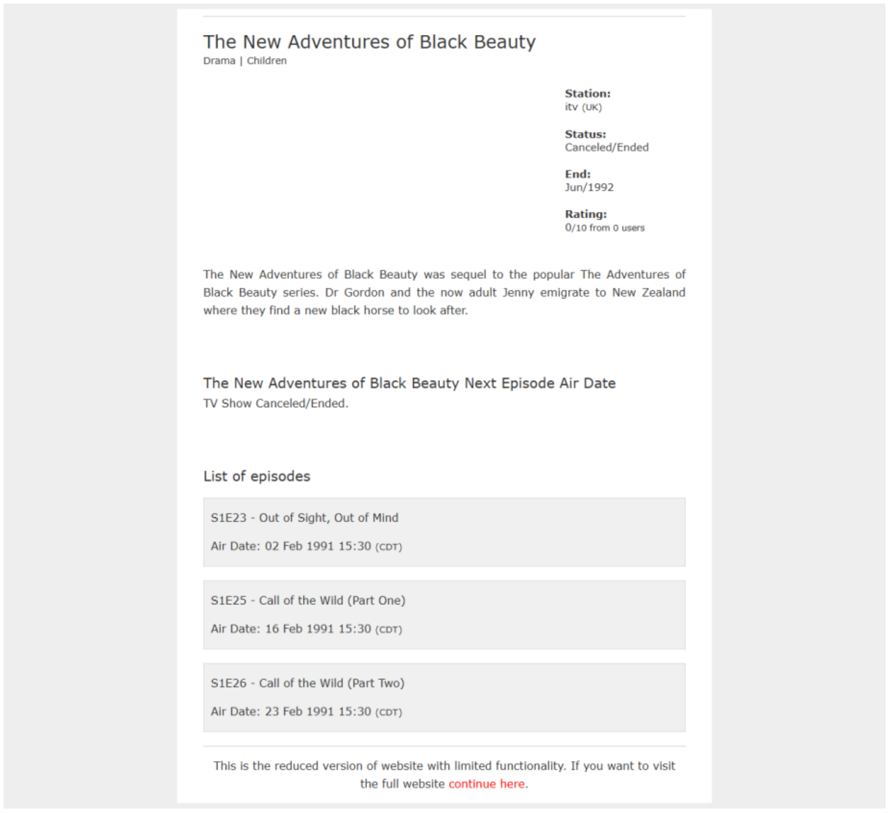
10. *Sketch-to-code functionality: Imagine sketching a rough website layout on paper—just boxes for buttons, text areas, and images. Now, what if an AI could instantly turn that sketch into a real website layout? With Sketch-to-Code functionality, you can:Draw a wireframe by hand or on a digital tablet. Let AI recognize your design and generate the structure automatically. Save time by skipping manual coding or dragging elements around. [5-point Likert scale, ranging from Very Useful to Not Useful At All]*
11. *Plugins-of-Plugins: Imagine having extensions that enhance the plugins or tools you already use in your website builder. Instead of being limited by built-in functionality, you can extend and customize plugins to better fit your needs. With Plugins-of-Plugins, you can: Custom Payment Methods – Add PayPal, QR Payments, or direct Bank Transfers to an existing checkout plugin. Custom Checkout Flow – Modify the purchase process, such as implementing a one-click checkout, multi-step form, or post-purchase surveys. Custom Media Plugins – Extend a gallery plugin to support interactive 3D models, before-after sliders, or auto-generated portfolio previews. [5-point Likert scale, ranging from Very Useful to Not Useful At All]*
12. *Version control: Imagine you're working on a website, and you make a few changes—maybe you update the design, add some text, or tweak the layout. Then, you realize, "Oops! I liked the previous version better." With Version Control, you can: See all past versions of your website. Go back to an earlier version if something goes wrong. Work together with others without accidentally overwriting each other's changes. Think of it like Google Docs' "Version History", where you can undo changes and see who edited what—except it's for your website. [5-point Likert scale, ranging from Very Useful to Not Useful At All]*
13. *Offline Editor: No network connection is necessary to edit the web page. All your changes will be save locally, and a network connection issue won't cause any data loss. With Offline Editor, you can: Edit your website even without Wi-Fi—no internet needed. Save changes locally on your computer. Sync updates later when you're back online. [5-point Likert scale, ranging from Very Useful to Not Useful At All]*

14. *Live Collaboration: Imagine working on a website with your team or friends—at the same time, from anywhere. With Live Collaboration, you can: Edit the website together in real time—like Google Docs for web design. See changes instantly, without waiting for someone to finish their part. Work efficiently as a team, whether you're a designer, developer, or content creator. [5-point Likert scale, ranging from Very Useful to Not Useful At All]*
15. *AI Image / Text Generation: If you require a picture, you can just key in your requirements and a image is generated. You can use AI to improve any text you wrote as well. With AI Image / Text Generation, you can: Generate AI images by simply describing what you need (e.g., "a modern office background" or "a nature-themed banner"). Enhance your website text—AI can rephrase, improve clarity, or make your writing more engaging. [5-point Likert scale, ranging from Very Useful to Not Useful At All]*
16. *AI recommendation system for templates: Not sure where to start? Instead of browsing through hundreds of templates, let AI recommend one that fits your needs. With AI-powered template suggestions, you can: Describe your website (e.g., "A portfolio for a photographer"), and Get a ready-made template, so All you need to do is fill in your content. [5-point Likert scale, ranging from Very Useful to Not Useful At All]*
17. *Any additional comments or features suggestion for website builders? [Open-ended]*
18. *Which of the following role describe you the best? [MCQ: Developer, Designer, Student, Business Owner, Product Manager, Hobbyist, Other]*
19. *What is your primary industry? [MCQ: IT (Information Technology), E-commerce, Creative (Design, Art, etc), Business and Marketing, Education, Other]*

Listing 1: A questionnaire used to gain insights from experienced users.

We present this list of questions to experienced users. We label respondents as experienced users if they choose any option other than 0 (No experience at all) for the website builder experience question. We omit the questions used for respondents with no experience since their responses have not been analysed in this thesis.

C. Examples of Experiment Dataset


The New Adventures of Black Beauty
Drama | Children

Station:
itv (UK)

Status:
Canceled/Ended

End:
Jun/1992

Rating:
0/10 from 0 users

The New Adventures of Black Beauty was sequel to the popular The Adventures of Black Beauty series. Dr Gordon and the now adult Jenny emigrate to New Zealand where they find a new black horse to look after.

The New Adventures of Black Beauty Next Episode Air Date
TV Show Canceled/Ended.

List of episodes

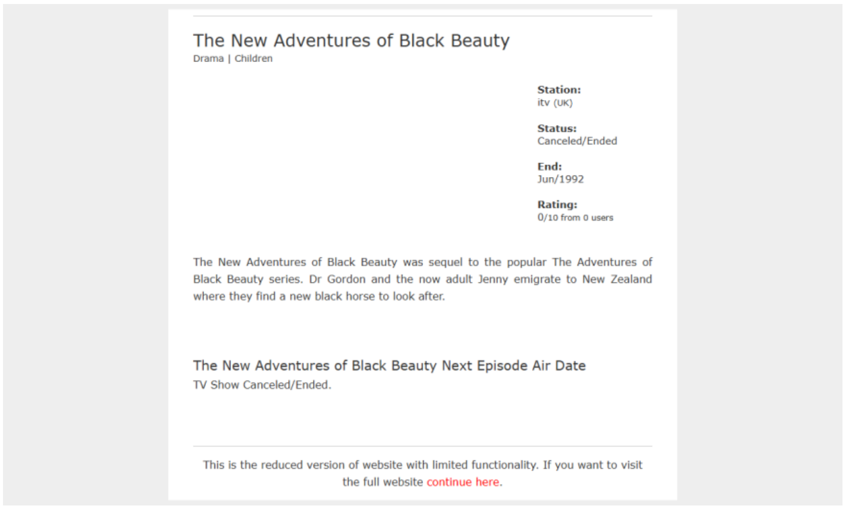
S1E23 - Out of Sight, Out of Mind
Air Date: 02 Feb 1991 15:30 (CDT)

S1E25 - Call of the Wild (Part One)
Air Date: 16 Feb 1991 15:30 (CDT)

S1E26 - Call of the Wild (Part Two)
Air Date: 23 Feb 1991 15:30 (CDT)

This is the reduced version of website with limited functionality. If you want to visit the full website continue here.

Figure 46: A screenshot of the original web page from the WebCode2M database. This screenshot is also used in the design-to-code “Merging” method.


The New Adventures of Black Beauty
Drama | Children

Station:
itv (UK)

Status:
Canceled/Ended

End:
Jun/1992

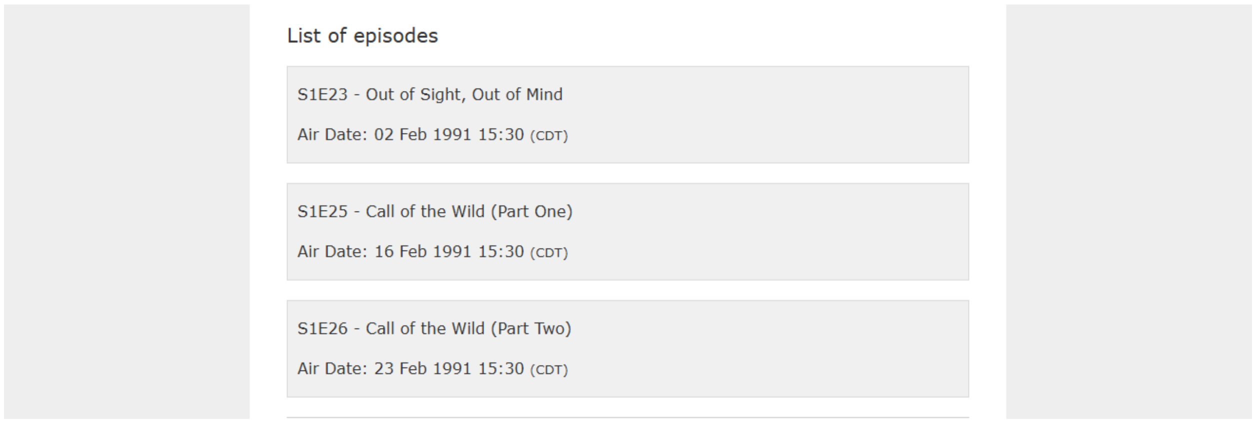
Rating:
0/10 from 0 users

The New Adventures of Black Beauty was sequel to the popular The Adventures of Black Beauty series. Dr Gordon and the now adult Jenny emigrate to New Zealand where they find a new black horse to look after.

The New Adventures of Black Beauty Next Episode Air Date
TV Show Canceled/Ended.

This is the reduced version of website with limited functionality. If you want to visit the full website continue here.

Figure 47: A screenshot of the web page after removing the episode list component from Figure 46.


List of episodes

S1E23 - Out of Sight, Out of Mind

Air Date: 02 Feb 1991 15:30 (CDT)

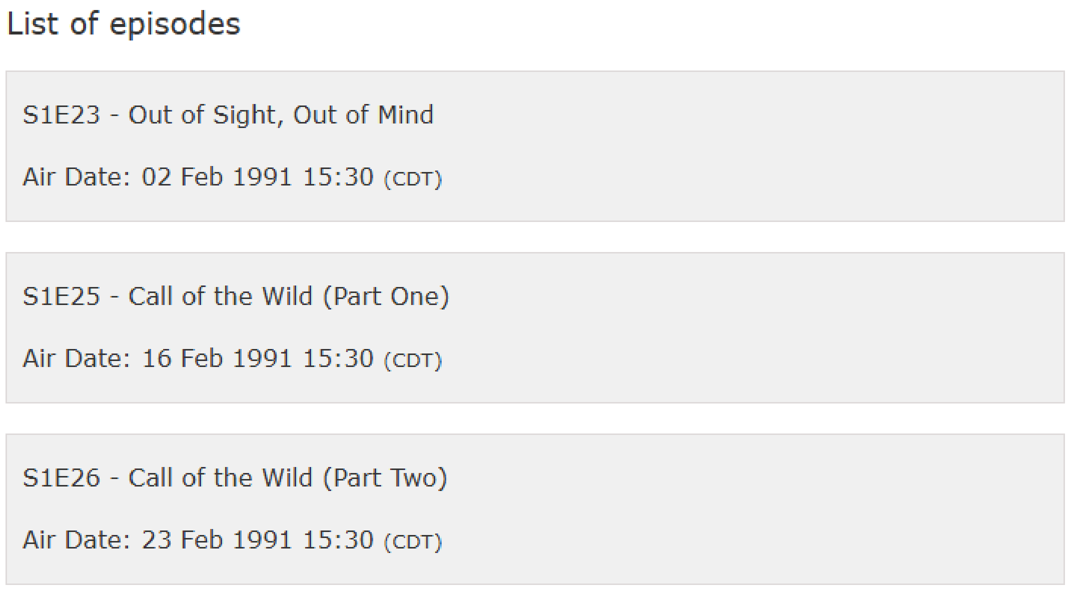
S1E25 - Call of the Wild (Part One)

Air Date: 16 Feb 1991 15:30 (CDT)

S1E26 - Call of the Wild (Part Two)

Air Date: 23 Feb 1991 15:30 (CDT)

Figure 48: A screenshot of the removed components in the default viewport, created from Figure 46. This screenshot is used in the design-to-code “Fixed-sized with Hint” method.


List of episodes

S1E23 - Out of Sight, Out of Mind

Air Date: 02 Feb 1991 15:30 (CDT)

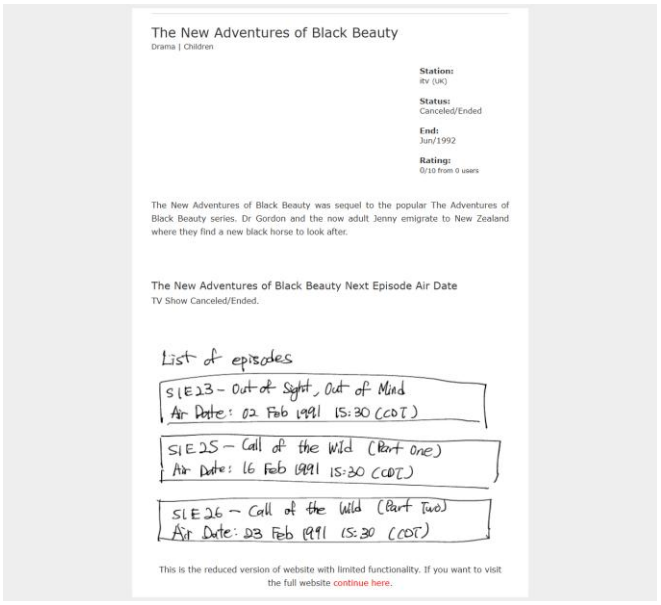
S1E25 - Call of the Wild (Part One)

Air Date: 16 Feb 1991 15:30 (CDT)

S1E26 - Call of the Wild (Part Two)

Air Date: 23 Feb 1991 15:30 (CDT)

Figure 49: A screenshot of the removed components without the sides, created from Figure 46. This screenshot is used in the design-to-code “Varied-sized with Hint” method.



The New Adventures of Black Beauty

Drama | Children

Station:
itv (UK)

Status:
Canceled/Ended

End:
Jun/1992

Rating:
0/10 from 0 users

The New Adventures of Black Beauty was sequel to the popular The Adventures of Black Beauty series. Dr Gordon and the now adult Jenny emigrate to New Zealand where they find a new black horse to look after.

The New Adventures of Black Beauty Next Episode Air Date

TV Show Canceled/Ended.

List of episodes

S1E23 - Out of Sight, Out of Mind
Air Date: 02 Feb 1991 15:30 (CDT)

S1E25 - Call of the Wild (Part One)
Air Date: 16 Feb 1991 15:30 (CDT)

S1E26 - Call of the Wild (Part Two)
Air Date: 23 Feb 1991 15:30 (CDT)

This is the reduced version of website with limited functionality. If you want to visit the full website continue here.

Figure 50: The modified screenshot with a hand-drawn overlay replacing the episode list section, created from Figure 46. This screenshot is used in the sketch-to-code “Merging” method.

D. Relevant Prompts Used in Experiments

System Prompt: ''' You are an expert web developer who specializes in HTML and CSS. A user will provide you with the HTML code of the current webpage, as well as a screenshot of the new webpage design.
Your task is to convert the new design into HTML and CSS code, based on the provided screenshot and the existing HTML code. You should modify only the necessary part, leaving the rest of the page unchanged. Include all CSS code in the HTML file itself.
Do not hallucinate any dependencies to external files. Pay attention to things like size and position of all the elements, as well as the overall layout.
If you need to include new images, use "temp.jpg" as the placeholder name. As a reminder, the "temp.jpg" placeholder is very large (1920 x 1080). So make sure to always specify the correct dimensions for the images in your HTML code, since otherwise the image would likely take up the entire page.
You should aim to make the new webpage as responsive as possible. You must response with only the final HTML + CSS code in one piece, and nothing else.'''

User Prompt: ''' Here is a screenshot of the new design for the webpage, as well as the existing HTML and CSS code. Please update the HTML and CSS code accordingly to the screenshot. Make sure to maintain the overall layout and design consistency. You must response with only the final HTML + CSS code in one piece, and nothing else.
\n''' + <<partial_html_code>>

Listing 2: Prompts given to VLLMs for the “Merging” method for partial design-to-code.

System Prompt: ''' You are an expert web developer who specializes in HTML and CSS. A user will provide you with the HTML code of the current webpage, as well as a screenshot of the new webpage design.

Some components in the new design are missing and need to be generated.

Your task is to convert identify the missing components in the new design and generate the HTML and CSS code for them, based on the provided screenshot and the existing HTML code. There is one single <missing></missing> tag in the existing HTML code which the new components should be inserted into, and you must replace it with the HTML code of the new design.

You should modify only the necessary part, leaving the rest of the page unchanged. Include all CSS code in the HTML file itself.

Do not hallucinate any dependencies to external files. Pay attention to things like size and position of all the elements, as well as the overall layout.

If you need to include new images, use "temp.jpg" as the placeholder name. As a reminder, the "temp.jpg" placeholder is very large (1920 x 1080). So make sure to always specify the correct dimensions for the images in your HTML code, since otherwise the image would likely take up the entire page.

You should aim to make the new webpage as responsive as possible. You must response with only the final HTML + CSS code in one piece, and nothing else.'''

User Prompt: ''' Here is a screenshot of the new design for the webpage, as well as the existing HTML and CSS code. Please update the HTML and CSS code accordingly to the screenshot. Make sure to maintain the overall layout and design consistency. You must response with only the final HTML + CSS code in one piece, and nothing else.

\n''' + <<partial_html_code>>

Listing 3: Prompts given to VLLMs for the "Merging with Hint" method for partial design-to-code.

System Prompt: ''' You are an expert web developer who specializes in HTML and CSS. A user will provide you with the HTML code of the current webpage, as well as a screenshot of partial new webpage design.
Your task is to convert the partial new design into HTML and CSS code, and insert it into the existing HTML code. There is one single <missing></missing> tag in the existing HTML code, and you must replace it with the HTML code of the partial design.
You should modify only the necessary part, leaving the rest of the page unchanged. Include all CSS code in the HTML file itself.
Do not hallucinate any dependencies to external files. Pay attention to things like size and position of all the elements, as well as the overall layout.
If you need to include new images, use "temp.jpg" as the placeholder name. As a reminder, the "temp.jpg" placeholder is very large (1920 x 1080). So make sure to always specify the correct dimensions for the images in your HTML code, since otherwise the image would likely take up the entire page.
You should aim to make the new webpage as responsive as possible. You must response with only the final HTML + CSS code in one piece, and nothing else.'''

User Prompt: ''' Here is a screenshot of the partial new design for the webpage, as well as the existing HTML and CSS code. Please update the HTML and CSS code accordingly to the screenshot. Make sure to maintain the overall layout and design consistency. You must response with only the final HTML + CSS code in one piece, and nothing else.
\n''' + <<partial_html_code>>

Listing 4: Prompts given to VLLMs for the “Fixed-Sized with Hint” and “Varied-Sized with Hint” methods for partial design-to-code.

System Prompt: ''' You are an expert web developer who specializes in HTML and CSS. A user will provide you with the HTML code of the current webpage, as well as a screenshot of the new webpage design. Note that some components are in sketch format in the screenshot.
Your task is to convert the new design into HTML and CSS code, based on the provided screenshot and the existing HTML code. You should modify only the necessary part, leaving the rest of the page unchanged. Include all CSS code in the HTML file itself.
Do not hallucinate any dependencies to external files. Pay attention to things like size and position of all the elements, as well as the overall layout.
If you need to include new images, use "temp.jpg" as the placeholder name. As a reminder, the "temp.jpg" placeholder is very large (1920 x 1080). So make sure to always specify the correct dimensions for the images in your HTML code, since otherwise the image would likely take up the entire page.
You should aim to make the new webpage as responsive as possible. '''

User Prompt: ''' Here is a screenshot of the new design for the webpage, as well as the existing HTML and CSS code. Please update the HTML and CSS code accordingly to the screenshot. Make sure to maintain the overall layout and design consistency.
\n''' + <<partial_html_code>>

Listing 5: Prompts given to VLLMs for the “Merging” methods for partial sketch-to-code.

System Prompt: ''' You are an expert web developer who specializes in HTML and CSS. A user will provide you with the HTML code of the current webpage, as well as a screenshot of the new webpage design. Note that some components are in sketch format in the screenshot.
Some components in the new design are missing and need to be generated.
Your task is to convert identify the missing components in the new design and generate the HTML and CSS code for them, based on the provided screenshot and the existing HTML code. There is one single <missing></missing> tag in the existing HTML code which the new components should be inserted into, and you must replace it with the HTML code based on the sketch in the screenshot.
You should modify only the necessary part, leaving the rest of the page unchanged. Include all CSS code in the HTML file itself.
Do not hallucinate any dependencies to external files. Pay attention to things like size and position of all the elements, as well as the overall layout.
If you need to include new images, use "temp.jpg" as the placeholder name. As a reminder, the "temp.jpg" placeholder is very large (1920 x 1080). So make sure to always specify the correct dimensions for the images in your HTML code, since otherwise the image would likely take up the entire page.
You should aim to make the new webpage as responsive as possible. You must response with only the final HTML + CSS code in one piece, and nothing else.'''

User Prompt: ''' Here is a screenshot of the new design for the webpage, as well as the existing HTML and CSS code. Please update the HTML and CSS code accordingly to the screenshot. Make sure to maintain the overall layout and design consistency. You must response with only the final HTML + CSS code in one piece, and nothing else.
\n''' + <<partial_html_code>>

Listing 6: Prompts given to VLLMs for the “Merging with Hint” methods for partial sketch-to-code.

System Prompt: ''' You are an expert web developer who specializes in HTML and CSS. A user will provide you with the HTML code of the current webpage, as well as a sketch of partial new webpage design.
Your task is to convert the sketch into HTML and CSS code, and insert it into the existing HTML code. There is one single <missing></missing> tag in the existing HTML code, and you must replace it with the HTML code of the sketch.
You should modify only the necessary part, leaving the rest of the page unchanged. Include all CSS code in the HTML file itself.
Do not hallucinate any dependencies to external files. Pay attention to things like size and position of all the elements, as well as the overall layout.
If you need to include new images, use "temp.jpg" as the placeholder name. As a reminder, the "temp.jpg" placeholder is very large (1920 x 1080). So make sure to always specify the correct dimensions for the images in your HTML code, since otherwise the image would likely take up the entire page.
You should aim to make the new webpage as responsive as possible. You must response with only the final HTML + CSS code in one piece, and nothing else.'''

User Prompt: ''' Here is a screenshot of the sketch of partial new webpage design, as well as the existing HTML and CSS code. Please update the HTML and CSS code accordingly to the screenshot. Make sure to maintain the overall layout and design consistency. You must response with only the final HTML + CSS code in one piece, and nothing else.
\n''' + <<partial_html_code>>

Listing 7: Prompts given to VLLMs for the "Varied-Sized with Hint" methods for partial sketch-to-code.