AI-Powered Website Generator

Ashwini Hiremath, Harsh Shinde, Pranavi Avula, Shivram Sriramulu, and Yogavarshni Ramachandran

Department of Applied Data Science

San Jose State University

DATA 298B: MSDA Project II

Dr. Simon Shim & Dr. Lee C. Chang

May 09, 2025

ABSTRACT

The project develops a model that helps non-technical users generate multi-page business websites. It helps web development for small businesses, freelancers, and startups that cannot afford expensive development services. The model bypasses usual web development costs by enabling users to build full-fledged multi-page websites with simple prompts, hence offering a more affordable online presence. It includes dynamically generated content that can permit marketing teams to update websites faster and adapt messaging themselves without technical skills. Users can create a complete, multi-page website in just several minutes and keep it current and relevant.

The dataset for this project will be scraped from publicly available business websites and stored in AWS in JSON format. Further, this dataset will be cleaned, formatted, and used to train four LLM models: GPT-4 (Generative Pre-trained Transformer 4), Gemini 2.0 Flash, Claude 3 Haiku (2024-03-07) and Claude Sonnet. Front end (React) will collect user inputs regarding industry type and design preferences to generate website templates and customized content. Models will be fine-tuned and tested to ensure high-quality, professional B2B SaaS websites per user specifications.

The model will be able to generate a complete, functional, website. Standard web pages include Home, About, Services, and Contact, with customized content. The web pages keep professional aesthetics and uniform formatting, while their contents change with different user inputs. Metrics will be include. Inception Distance for similarity of templates, Google Lighthouse Score for performance, Cross-device Responsiveness, Content Relevance, and Fluency to make sure websites come in according to the user specifications, are appealing, and act as needed across varied devices.

The project offers a very affordable, efficient AI model that produces industry-specific websites for small businesses, freelancers, and startups. Dynamic content generator with support for continuous updates. Democratizing web development by enabling a broader range of businesses to maintain a strong online presence without technical skills.

1. Introduction

1.1 Project Background and Execute Summary

Project Background

The ever-evolving technological front most especially the emergence of the NPS (Net Promoter Source) and the growing adoption of the internet as tools for business transactions call for Web Development solutions that are easier, faster and less resource intensive to implement. Freelancers, startups, and small-scale companies struggle heavily to create a strong web presence because investing in web development may be costly. This is evident as these entities all seek to get the best solutions that are not only cheap but are also efficient depending in their operations in the online market place. The project offers a new approach to addressing these challenges by developing an AI-generated model that can generate multi-page sites for non-developer users. Via the HMVC model of operation, users are able to input simple instructions in plain language to the system and these are translated in to fully operational and fully customizable webpages thereby minimizing the necessity of costly web development services. With the help of Generative AI models like Claude Sonnet, GPT-4, Claude, and Gemini incorporated within the system together with the input collecting interface for the indication of the industry type and design preferences the web development is expected to be made more efficient Kanakia and Nair (2023). This also reduces the possibility of financial and technical barriers while at the same time increasing the chances for these entities to sustain professional and up to date web presence with relative ease. Verma, Kurupudi, and S (2024).

Project Approaches and Methods

The AI-Powered Website Generator project will adopt a systematic approach comprising three main steps: In data collection, data preparation, and modeling efforts, each modality is considered separately. This systematically planned framework shall allow complex input such as plain text, photographs, formatted information, etc., to be taken and incorporated to generate the optimum website. Each stage will be anchored in a thorough process of planning and design so that the generated websites reflect quality and relevance to the user's context. This structural

approach will make it easier to develop new, multilevel, and personalized websites with high efficiency and productivity through LLM Models.

Data Collection. The data collection process is initiated by selecting 500 public websites that meet the needs of the project. Some of the widely recognized websites are scrapped to fetch the HTML, CSS, JavaScript, and images of the respective websites with help of JavaScript-based tools namely Puppeteer or Selenium. Afterward, the scraped data is saved as site's organized structures or folders that different websites contain. This is good in making sure that there is a good capture of the web data that would have been lost if a single method of crawling was used. Last but not the least, the collected data is saved on Google Drive for better control of data and also it gets prepare for the final preprocessing and transformation stages in ETL process.

Data Preparation. In data pre-processing phase, which aims at making the raw website data developed and ordered to a particular uniform structure. First, all the collected data is concentrated into groups by the website under study while keeping the totality of files with HTML, CSS, JS, and images. The preprocessed script runs on Google Colab where the data is reshaped into the required and coherent JSON format for later processing. Some of the formats prepared data takes includes data that has been formatted to fed into model training or some other process.

Modeling. The choice of the models that will be suitable for completing specific tasks of website development will be made in the modeling phase of the AI-Powered Website Generator project. In regards to text content generation and encoding, realistically functional models like GPT-4 will be used in generating robust printed and SEO materials. Claude will use the web page structures and deal with the formatted contents and the recurrent parts such as headers, labels, and information that will result in the improved structure of Web pagesKanakia and Nair (2023).These will include button creation, form, and all other clickable components, as well as navigation, will be fully under Gemini jurisdiction to ensure they create complements and balance the entire layout with optimal usability. Additionally, claude will help us with it's natural language processing abilities that will assist in context-aware descriptions which will help in semantic coherence across ui content elements. Zhao et al. (2023) This way it is guaranteed that the

websites being generated are not only current and semantically accurate, but also precisely to the user's specification in terms of utility, as well as the site's flair and sophistication.

Expected Project Contributions and Applications

The proposed AI-Powered Website Generator project is aimed to greatly extends the literature by proposing an AI model that decentralises web creation to enable access for SMBs, free lance, and start-ups at cheaper and more convenient ways. Through Large Language Models, it has the ability of creating dynamic and industry-specific multiple page websites using basic user commands Muthazhagu and B (2024). Such approach also helps to generate the dynamically updated content in case, without the need for the professional programmer, and keeps website looking professional with relevance to the particular market of the user. The technology of using AI for content and template creation coupled with a form-based system for user input to provide web content makes the creation of unique websites easy. This project therefore presents a useful solution for organizations that want a powerful and effective online presence, i.e., through low cost web development, thus broadening the option of web development opportunities in different sectors Muthazhagu and B (2024).

This platform will be useful in a number of domains, such as personalized web applications, updating e-stores on a regular basis, and Dynamic Content for Digital Marketing. Further, in digital marketing, this could result in highly timely, appealing, and professional multimedia advertisements that directly appeal to customers and skyrocket the brand awareness. In addition, the project will include systematic measurements like the Google Lighthouse Score for WEBP +SEO performance for scrutinizing the quality of the website generated by the project systematically Dhyani, Nautiyal, Negi, Dhyani, and Chaudhary (2024).

Finally, the new product called AI-Powered Website Generator will open up a new stage in web development and integration of AI applications with websites, making it easier to build web resources that could predict the needs of users and changing market trends. Thus, due to the nature of the problems addressed within the scope of this project, it is possible to suggest that the obtained technologies and methodologies are also expected to advance not only the confines of

academic research but also applicable practices of building and managing the content of the World Wide Web.

1.2 Project Requirements

Functional Requirements

As outlined clearly in the functional aspects, the process aspect defines the manner, tool, and technology that occurs in the process of creating a project.

Automated Web Design and Deployment This correlates with general discourse in papers such as 'HTML Code Generation from Website Images and Sketches using Deep Learning-Based Encoder-Decoder Model' where the papers explore generation of HTML of designs D, Sneha, and Kumar (2022).

Custom Element Generation Based on capabilities found in 'Web Components Template Generation from Web Screenshot' where the role of artificial intelligence is to recognize and generate the Web components from a given picture Anunphop and Chongstitvatana (2022).

Seamless Content Integration Generative AI is generally philanthropic in nature as discussed in 'Generative Artificial Intelligence: A Systematic Review and Applications' based on the AI discussed to's flexibility across different content types Hughes, Zhu, and Bednarz (2021).

User Experience Optimization This requirement is extrapolated from the discussion on how AI will make UI and UX better in the "Exploring the Role of AI to Web Design and Development" Jeong (2024).

Real-time Customization and Preview This requirement does not raise demand, it is related to aspects covered by real-time data processing and interaction models like the ones discussed in "Implementation of Generative AI Services Using an Enterprise Data-Based LLM Application Architecture", meeting the demand for dynamic data processing Jeong (2024).

AI-powered Requirements

For this project, we are using Large Language Models such as Gemini, GPT, and Claude.

Advanced Text and Content Generation with GPT-4 : The study by Kanakia and Nair (2023) used for creating high-quality textual content and keyword-rich content that includes web

page descriptions, articles, and Meta tags. Due to the extent of its capability in natural language processing in relation to text comprehension and text production based on contextual prompts, this model is helpful in the development of personal related content that is relatable to the global population. It will also assist with relative text summarization so that messages are to the point and are formatted properly for readers.

User Interface and Interaction Design with Gemini : Focused on providing design and development of user interface elements that adapts to the users' interactions Shrivastav, Shahane, Hydri, Akre, and Amin (2024). This model will create complex UI components for forms, buttons, and menus for the given specification and user behavior while making the websites more user-friendly Shrivastav et al. (2024).

Enhanced Semantic Understanding with Claude: Claude provides complex semantic meaning to the project by focusing on the semantic context of all issues among the Website elements. There are no structural and logical issues with the written language, and it is rather helpful in optimizing written content, and balancing the user's needs with the information they gain Priyanshu, Maurya, and Hong (2024). Moreover, Claude is particularly strong in producing conversational interfaces and thus can be used as an benefit for designing interactive components such as a chat, FAQ section or other that should correspond to the website's topic and its major objectives. They also include natural language processing that enables micromanaging of content for improved processing, readability, as well as, user interactions.

Data Requirements

This project focuses only on commercial website mainly for B2B, models need to be trained on those data to generate a website for B2B. So we need all pages of data so we need to take public website data with all pages.

Diverse and Salable Data Sets. The AI models will draw data from a wide cross section of business websites scraped from the internet to embrace diverse styles as well as content. This dataset also has to be extensible and updated on a regular basis to reflect the current state of web designs and its business application.

Data Security and Integrity. This requires that the data used for training and operation of the models to be secure and intact to conform with users' trust and reliable system. The system has to meet the data protection act and use appropriate handling and storage of data practices Jeong (2024).

Project Proposal

The team worked on various problem statements, possible solution proposals, and decided on the best procedure to create a Gen AI-powered Website generator. Automation of website generation is a challenging task, affecting many businesses, their reach, and prosperity. It will also give information about how the system will empower small businesses, freelancers, and startups as discussed.

Project Management Plan

The team will be working based on a specific project plan, following tasks, sub tasks using project management tools like JIRA and Notion. Deliverables of plan includes WBS chart, Gantt Chart, PERT Chart. Table 1 shows the list of project deliverables.

Data Management Plan

Data management plan includes how data has been collected, the data storage system, the configurations of those systems, the data pipeline with reporting on its metadata. Project Deliverables will be having reports and documentation of those data management plan and its metadata.

1.3 Project Deliverables

Table 1

Project Deliverables and Timeline

Deliverable	Description	Due Date
Project Proposal	A detailed document proposing the project goals, methodology, and expected outcomes.	09/02/24
WBS	Describing project breakdown in CRISP DM methodology using a Work Breakdown Structure.	09/07/24
Gantt Chart	Create a Gantt chart showcasing the project deliverables timeline, tasks with their dependencies.	09/16/24
Data Management Plan	Plan outlining the scraping of data from publicly available websites.	09/13/24
Data Collection Plan	Gathering raw HTML, CSS, JS, and images from 500 B2B SaaS websites.	09/04/24
Data Transformation	Organizing the data into separate folders for each website with page linking.	10/04/24
Data Preprocessing	Converting website data into single JSON files to preserve the hierarchy and create templates.	10/07/24
Data Loading	Adding the template data into an S3 bucket for modeling.	11/11/24
Model Development	Take GPT 4, RAG, Llama2, and Claude models and fine tune with prompt and templates.	18/12/24
NLP User Input and Front-End Creation	Developing user interface in React application for real-time website generation.	24/02/25
Model Evaluation and Deployment	Evaluating models based on User Feedback, Google Lighthouse Score, BLEU, and ROUGE score.	08/03/25
Final Report	Project documentation in APA format containing a description of each phase, results, and recommendations.	09/05/25
Presentation	Project presentation highlighting the preprocessing, pipeline, and modelling with a focus on the results.	09/05/25

Based on the research, related studies, and general research, the project is scheduled to offer the following factors. Table 1 shows the detailed timeline of project deliverables.

Data Collection Plan

HTML, CSS, JavaScript and images of 500 public websites are being crawled using web scraping tools. The raw data is managed in an efficient manner by their storage in different structures such as Google Drive or cloud storage.

Data Transformation

The raw data is stored according to directory structure where HTML, CSS, and JavaScript files together with other image files are stored in different folder for each website. The linking codes pages are used to keep the page structure and organization in order and furthermore for convenient linking and processing.

Data Preprocessing

The organized data is then converted into a single JSON file for each website and the HTML CSS as well as JavaScript is embedded in the right website hierarchy. After that based on each industry, we collected a template for that particular industry and have it as templates to give context to the models.

Data Loading

The preprocessed data, which are industry specific templates are loaded in the final S3 bucket. This centralization provides for convenient transfers and interactions with the model training process and the system for model deployment.

Model Development

Deliverables for the modeling part will be including the coding and algorithm of the model. Python or ipynb files and other data scrapping files. Which includes model evaluation part as a deliverables.

NLP User Input and Front-End Creation

The React framework is used to construct an interactive front end where users can describe website needs in natural language prompts. These inputs are content that this system feeds to the Natural Language Processing models to get the web designs and content it generates in real-time with live previews possible as well as interactive controls for a rich experience.

Model Evaluation and Deployment

The trained models are tested for its accuracy, coherence and usability through other parameters like BLEU score, user feedback and all real life scenario. When the models are proved, both the models and the front-end system are running on the cloud platforms, users can interact with the AI-based website generator in real-time, the system can audit the performance and update constantly.

Final Report

Final report is the project report which has entire documents and steps of the project from project proposal to deployment. Final report will be submitted along with final presentation.

1.4 Technology and Solution Survey

Llama Next, Yeom et al. (2024) presented the work of TC-Llama 2, in which the model Llama was used to develop improved analysis of the most important relationships of technology-product in technology commercialization.

Another application was the translation of natural language descriptions into functional code conducted by the Llama 3 70B model during code generation tasks in the paper Ersoy and Erşahin (2024). Its large size to deal towards optimization utilized several techniques such as training, fine-tuning, PyTorch FSDP, Quantized Low-Rank Adaptation.

The GPT-4 project Goodfellow et al. (2014) developed a large-scale multimodal model that processes both text and images into a stream of text output. This is a model that improves natural language processing, generating applications for complex conditions.

The paper Castillo-Campos, Varona-Aramburu, and Becerra-Alonso (2024) has used GPT

models in generating summary words from news headlines, whereby GPT-3.5 and GPT-4 have been crafted with the bias issue of AI-generated content. The problem solved here was to instill awareness into how these AI tools must have produced biased or subjective language in journalism.

The work Agarwal, Goswami, and Sharma (2023) in Answering and Explaining Conceptual Medical Physiology Multiple-Choice Questions" employed Claude-2 to answer a set of 55 MCQs in physiology.

In the project Cao et al. (2018), Claude was used as a conversational agent designed to enhance content creation with its own methodology called Constitutional AI. That means it is able to learn both from human feedback and AI feedback while concentrating on a set of guiding principles that make its output much more responsible.

In the project Shrivastav et al. (2024), Gemini was a sophisticated model of AI, highly enriching the process of generating content with its advanced NLP. This AI, developed by Google, is designed to interpret and generate text that tallies quite closely with what the user needs. The tables 2, 3 are showing in details.

Table 2

Research Papers, Objectives, and Data Sources (Part 1)

Paper	Objective	Data
Llama Next - Technology Product Relationships in Technology Commercialization Yeom et al. (2024)	Enhance understanding of technology-product relationships using the Llama model.	Bilingual Korean-English datasets.
Evaluation of Llama 3 70B for Code Generation Tasks Ersoy and Erşahin (2024)	Assess Llama 3 70B model's performance in generating code.	Large datasets for distributed training.
Multimodal Processing with GPT-4 Goodfellow et al. (2014)	Process text and images into continuous text output using GPT-4.	Large Transformer-based training datasets.
Bias in AI Journalism Castillo-Campos et al. (2024)	Investigate bias in AI-generated journalism content.	News-related prompts.
Assessing Claude-2's Capabilities in Physiology Agarwal et al. (2023)	Evaluate Claude-2's ability to explain advanced medical physiology concepts.	55 multiple-choice questions on medical physiology.
Enhanced Content Creation via Constitutional AI Cao et al. (2018)	Develop content creation methods using feedback from humans and AI.	Human and AI feedback.
Content Creation Platform with Gemini Model Shrivastav et al. (2024)	Build a platform to interpret and generate text based on user input, focusing on style and tone.	User inputs on style, tone, and competency structure.

Table 3

Research Papers, Objectives, and Data Sources (Part2)

Paper	Objective	Data
Automated Web Development with Deep Learning Anunphop and Chongstitvatana (2022)	Automate web development by generating Web Components templates from screenshots.	Dataset from business information websites.
Adaptive Web Systems for Personalized Content Sadat and Ghorbani (2023)	Adapt web content to user models and context using systematic synthesis.	User interaction data in relational databases.
Blog Generation Application Using Llama-2 Verma et al. (2024)	Develop an open-source blog creation tool using the Llama-2 model.	Data from Kaggle.

1.5 Literature Survey of Existing Research

This paper, therefore, by Anunphop and Chongstitvatana, Anunphop and Chongstitvatana (2022), represents the approach of AI in driving automation for web development through generating Web Components templates from web screenshots.

The paper explores how the API from OpenAI could be used for front-end automated code generation in order to improve the efficiency of web development and reduce the complexity and time it takes with the use of manual coding. The main technologies explored in this paper are machine learning models using OpenAI, which generate HTML, CSS, and JavaScript code based on design specifications defined by a user Anunphop and Chongstitvatana (2022).

In the paper Sadat and Ghorbani (2023) by Sadat and Ghorbani 2023, starting with the main challenges of adapting web content to user models and contextual factors to-date beyond the capacity of traditional static pages, a systematic synthesis process is introduced, including request analysis.

As such, this paper proposes Verma et al. (2024) an open-source blog generation application based on the Llama-2 large language model. In fact, we were motivated by the

open-source nature of the Llama-2 model, which supports fine tuning of the model-a feature that will enable users to update the model.

The paper Muthazhagu and B (2024) explores, in great depth, the transformative power of AI for web design and development with special emphasis on automated code generation.

The paper Fausti, Pugliese, and Zardetto (2020) presented a methodology developed by the Italian National Institute of Statistics for classifying enterprise websites according to their e-commerce capabilities using Deep Learning techniques.

Zhao et al. (2023) A Survey by Zhao proffers a critical overview of the recent advances in Retrieval-Augmented Generation technologies that have been developed to improve AI-generated content.

This paper introduces SMARTCHAT, a real-time chat system enhanced through natural language processing.The best results were received with the model, which used an architecture of RNNs supported by an attention mechanism. This improved the recall of context and the quality of responses. Shrivastav et al. (2024)

The Chat2VIS System Chat2VIS is a system developed on top of large language models like GPT-3 and Codex, which transform natural language queries into data visualizations by Kuswanto, Nolan, and Lu (2023).

Gozalo-Brizuela and Garrido-Merchán (2023) This survey performs a deep review of over 350 generative AI applications, tiling the space with a structured technique, placing technologies into one of 15 areas: text, images, video, and code. It identifies the principal technologies involved: deep learning, transformers, GANs, and VAEs.

The paper Kingma and Welling (2013) by Kingma and Welling introduces a new approach for approximate inference within directed probabilistic models which merges stochastic variational inference with auto-encoding techniques. The tables 4 and 5 are showing this in detail.

Table 4

Literature Review: Papers, Objectives, and Data Sources (Part 1)

Paper	Objective	Data
Anunphop and Chongstitvatana (2022)	Use deep learning techniques to create templates for Web Components from web screenshots.	Dataset created from business information websites, used for object detection tasks.
Sadat and Ghorbani (2023)	Adapt web content to user models and contextual factors beyond traditional static pages.	Continuous updates to user models based on interactions and session data via J2EE services.
Verma et al. (2024)	Develop an open-source blog creation tool using the Llama-2 model, allowing user fine-tuning.	Data from Kaggle used to fine-tune the model and integrate with user interaction via React Application.
Kingma and Welling (2013)	Introduce Auto-Encoding Variational Bayes for approximate inference in probabilistic models.	MNIST and Frey Face datasets, focusing on handling large datasets and intractable posteriors.
Hughes et al. (2021)	Explore trends and methods in generative AI applications for creative and design industries.	Dataset from Kaggle.
Jeong (2024)	Implement generative AI services using a large language model (LLM) application architecture for businesses.	Dataset from Kaggle.
Muthazhagu and B (2024)	Examine the role of AI in web design, focusing on automated code generation using CNN and LSTM networks.	Well-annotated datasets required, emphasizing real-world design scenarios.
D et al. (2022)	Convert website images and sketches into HTML code using deep learning-based encoder-decoder models.	Dataset from Kaggle.
Geraldi and Lechter (2012)	Analyze the history and implications of Gantt charts in project management.	Dataset from Kaggle.
Shrivastav et al. (2024)	Explore the Gemini model for creating AI-based content.	Dataset from Kaggle.

Table 5

Literature Review: Papers, Objectives, and Data Sources (Part 2

Paper	Objective	Data
Chowdhery et al. (2023)	Discuss the PaLM model's capabilities in scaling language modeling with pathways.	Dataset from Kaggle.
Agarwal et al. (2023)	Assess Claude-2's performance in answering conceptual medical physiology questions.	55 multiple-choice questions (MCQs) in physiology.
Cao et al. (2018)	Review the evolution of generative AI, from GAN to ChatGPT.	Dataset from Kaggle.
Castillo-Campos et al. (2024)	Compare GPT-3.5, GPT-4, and Bing in terms of bias in AI-generated journalism.	News-related prompts created or extracted from headlines.
Zhao et al. (2023)	Survey Retrieval-Augmented Generation (RAG) technologies for enhancing AI-generated content.	Dataset from Kaggle.
Gozalo-Brizuela and Garrido-Merchán (2023)	Review over 350 generative AI applications to classify and understand the scope of available technologies.	Dataset from Kaggle.

2. Data and Project Management Plan

2.1 Data Management Plan

Data collection approaches

Our project aims to collect data using data scraping methods to build structured datasets from currently existing websites and their dynamic content elements. First, we use axios to download the website's page and use cheerio to parse it for such things as links, scripts, and metadata elements yet to be fetched. While other crawlers rely on programmatic or server-side scraping of static HTML content, we use Puppeteer backed by the Chromium browser to load the page, inject browser automation and wait for the Intersection Observer API that allows reviewing the fully loaded DOM of the page and unloading it together with JavaScript in full length for interactive and dynamic elements. This information collected in turn is useful for website structures, styles and components and it is processed for training AI models. Such models prove patterns for design, thereby allowing the generation of templates and the creation of web sites. In this way, we hope to reconfigure the Web site development process so that patterns of the new form can be generated by AI systems that learn from and extend existing forms of web design. Figure 1 and 2 shows the code for web scraping.

Figure 1

Data Scraping using B2B website URL

Figure 2

Data Scraping using B2B website URL

```
396
397         fs.writeFileSync(combinedJsFilePath, combinedJsContent);
398         console.log(`Combined JS saved as ${combinedJsFilePath}`)
399         return combinedJsFileName;
400       }
401       return usePuppeteer
402   }
403
404
405   (async function runScraping(){
406     const urls = [
407         "https://www.amberflo.io/",
408         "https://www.vendr.com",
409         "https://www.swan.io/",
410
411     ];
412
413     try {
414         await Promise.all(urls.map(url => templatePage(url)));
415         console.log("Scraping completed for all URLs.");
416     } catch (err) {
417         console.error("Error during scraping:", err);
418     }
419   })();
```

Management methods

The scraped data is then saved for each website in Google Drive, where each website has its folder. In these folders the files are split into subfolders of HTML, CSS, JavaScript and images which makes it much easier to organize and find the material. Later the data is automatically downloaded from Google Drive and copied to the folder structure preserved in the AWS S3 Bucket. Data cleaning, data transformation and organization for the project requirements are done with AWS Glue. The retrieved values are then stored as objects in to a new S3 bucket which provides the base for the analysis and elaboration processes. This approach uses Google Drive to perform the storage and sorting stages, and AWS S3 to store data in a highly secure and efficient way once the data has reached an appropriate scale. The final S3 bucket with the transformed data serves as a master data management that sustains the project's operations and most analyses. It also provides the capability for structured and efficient data storage and transformation characteristic of systematic data management: integrated into the data structure for project use while preserving the integrity and security of the data. The Figure shows 3, 4,5 the data management plans.

Figure 3

Data: Website Folders

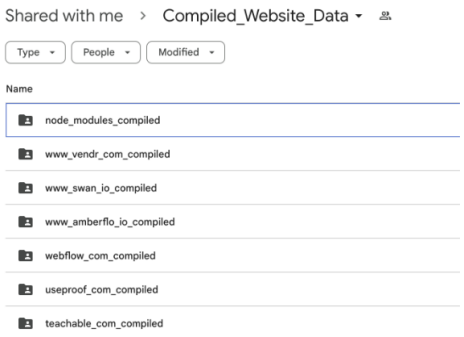


Figure 4

Organized Websites Storage

Figure 5

Loading all data to AWS S3

```
# Process each website folder in Google Drive and upload to S3
for website_folder in os.listdir(drive_folder_path):
    website_folder_path = os.path.join(drive_folder_path, website_folder)

    # Ensure it's a directory
    if os.path.isdir(website_folder_path):
        # Convert the website folder to JSON format
        website_data = transform_website(website_folder_path)

        # Define the output JSON file name and path
        output_file_key = f'{output_path}{website_folder}.json'
        website_json_content = json.dumps(website_data)

        # Upload the JSON to S3
        s3.put_object(Bucket=output_bucket_name, Key=output_file_key, Body=website_json_content)
        print(f'Successfully uploaded {output_file_key} to S3 bucket {output_bucket_name}')
```

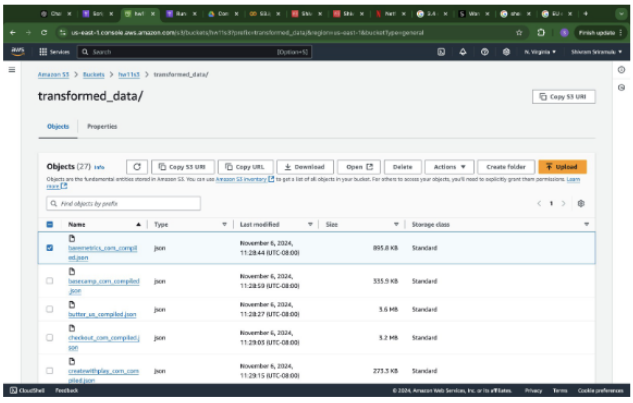
```
Successfully uploaded transformed_data/www_atlist_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/www_kissmetrics_io_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/www_flycode_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/butter_us_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/baremetrics_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/basecamp_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/checkout_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/createwithplay_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/draftbit_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/enginebystarling_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/ghost_org_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/kajabi_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/lattice_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/mailchimp_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/methodfi_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/onfido_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/segment_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/slack_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/slidebean_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/stripe_com_compiled.json to S3 bucket hw11s3
```

Storage Methods Using Google Drive and AWS S3

The project in question means dealing with storage of roughly 515 websites, which sums up to roughly 7 gigabytes of online space. This storage requirement is for local storage as well as the Google Drive and then to AWS S3 for all the processing. First of all, the data is kept in the Google Drive and is sorted into folders as more detailed above. The data is then transformed by AWS Glue and put into JSON format for storage in an AWS S3 bucket. This makes it easier for the management and easy access to the processed data for other uses. Figure shows 6 storage plan.

Figure 6

S3 Bucket Storage

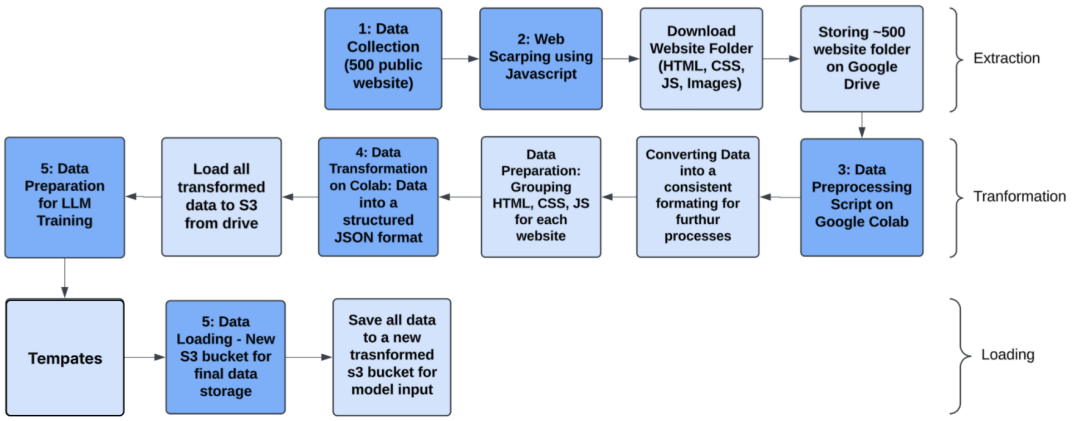


Usage Mechanisms Website Regeneration

The collected data will be used for dynamic website generation wherein the HTMLs and CSS files can be used to retrieve the front-end of the website. Figure 7 shows the basic idea of our data pipeline.

Figure 7

ETL Process for Data Scraping and Transformation



This ETL (Extract, Transform, Load) pipeline in AWS Glue shown in 7 is used to gather, clean and structure data from roughly five hundred public websites for feeding into Large Language Models (LLMs). The phase begins with the Extraction phase in which the site is determined, and then with the help of tools such as Puppeteer or Axios, it extracts the data from these sites. The extracted data consist of secondary resources like HTML files, CSS stylesheets, JavaScript files and image files. These are further incorporated into a folder structure to retain the architecture of the website. Once obtained the website data is stored locally and ideally can be stored in Google Drive for comparison and further analysis.The final phase of the presented approach is the Transformation phase which entails transform the raw data for subsequent processes. This commences with a perplexity layer in Google Colab that categorises each website resources into HTML CSS JS and imgs. The script then process the data into an equivalent structure so that they are compatible with the next steps of processing. Last, the Loading phase entails moving the data in its reformatted form to Amazon S3 for archival and to facilitate a growth of the DW system. The pipeline loads the transformed JSON datasets and other additional grouped resources into a primary S3 bucket. Then, the data is refined even further in order to get templates which are essential to train LLMs. These pairs are moved to a different, transformed S3 bucket

designed specifically for inputs to the models. It provides a systematic way of pre-processing the raw data that are crawled from the websites so as to create a neat dataset that forms the basis for developing AI systems capable of modelling and generating structures and content from websites.

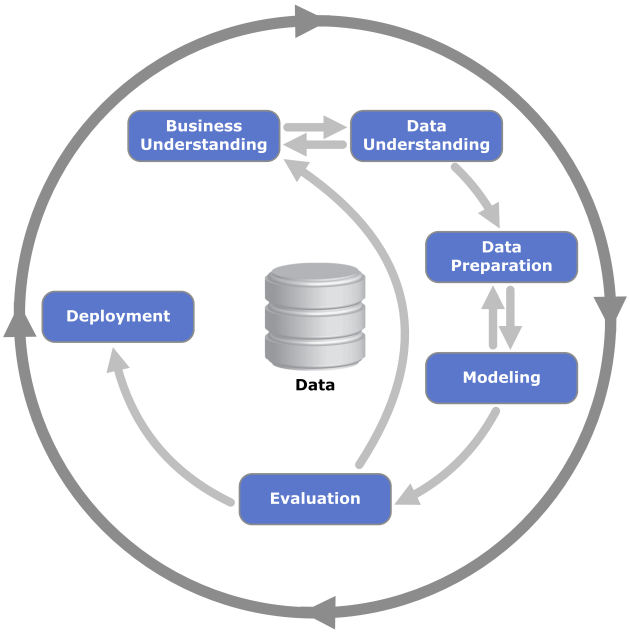
2.2 Project Development Methodology

The CRISP-DM Methodology will be used in the development of this project. CRISP-DM stands for Cross-Industry Standard Process for Data Mining. The CRISP-DM methodology is a very structured way of developing a data science and analytics project. It consists of 6 major phases namely , business understanding, data understanding, data preparation, data modeling evaluation, and deployment. According to the paper on Applying the CRISP-DM data mining process in the financial services industry by Plotnikova, Dumas, and Milani (2022), in order to deliver data mining projects consistently , organizations use a standardized approach like CRISP-DM. According to the paper on A Systematic Literature Review on Applying CRISP-DM Process Model by Schröer, Kruse, and Gómez (2021), data science projects follow six iterative phases that starts with business understanding and ends with deployment. We will have an in depth understanding about the CRISP-DM approach in the further part of this section.

System Development Life Cycle uisng CRISP-DM

Figure 8

CRISP-DM Methodology



The Figure 8 shows detailed workflow of using CRISP-DM methodology in a data mining project.

Business Understanding. The primary objective of this project is to develop an AI powered website generator that will allow the end user to create professional and multi-page websites. This project aims to help small businesses and startup's to have an end to end User interface so that they can market their products over the internet. Our aim is to scrape data off various websites available on the web and use Large language models like GPT-4, Llama2, Gemini, PaLM that will help the user to input simple prompts regarding their industry type and design so that our models generate a website for them. This will allow the businesses to mitigate the cost required to develop an end to end website and maintain a strong presence over the internet. Our project aspires to generate a fully functional website with aesthetic designs and create dynamic content for user friendly features.

Data Understanding. The project will be developed using web data,which is scraped from publicly available websites across various industries like healthcare ,manufacturing and startup's. More than 700 websites will be scared and data will be stored in the form of JSON format that will represent a systematic structure regarding the websites like html tags, Css information like font's , colors and layouts. The extracted data will also have metadata content like text , images and headings which are useful for generating SEO friendly web pages. Understanding the structure of the data in this phase will help us to eliminate irrelevant content like ad's. The data will then be segregated based on the industry type so that the models are capable of generating templates for various industry types. An in depth understanding of the dataset is essential so that our models can generate high quality websites with dynamic user inputs.

Data Preparation. The data preparation phase can be considered as one of the crucial processes as the data will be cleaned and ready to be processed using large language models like model1,model2,model3,model4. The dataset will be free of any irrelevant data to ensure that our model provides us with high quality inputs. Websites often contain repetitive tabs for headers, footers and navigation panes; these duplicates are removed to avoid any redundancy in the data. Text data is normalized so that it is free from stop words , special characters.Furthermore data is tokenized into individual words so that the data can be utilized for language models like Gpt-4 and gemini. Moreover, the dataset will be split into train, test and validation sets to train and evaluate the performance of the model. Overall we will also employ feature engineering techniques to classify industries and format the content of the input.

Modeling. When in the modeling phase of the AI-Powered Website Generator project, this shall be done using sophisticated AI models to design unique, multiple page templates for the website. Thanks to GPT-4, it will be possible to create not only good SEO texts and dynamic content from user input, but also relevant and interesting. Gemini will create such interface elements as buttons and navigation panels as well as structural elements and adapt interfaces to gadgets. The models will be trained based on the user inputs that will be received through React application and the efficiency of the models will be determined by parameters that include Google

Lighthouse scores.

Evaluation. In this part of the project, namely the model evaluation, we will identify the specifics of the quality of the websites that has been generated using AI. Another important decision-making process is the assessment of the current performance of the model as compared to the expected results as well as to the general expectations from such models in different industries. In this work, we will use the results of Google Lighthouse in evaluating the performance, accessibility, SEO, and the application of best practices in terms of LCP, FID, and CI parameters as KPI. This assessment will enable us to make necessary recommendations to improve performance, layout, and compliance to web site accessibility standards. This assessment will involve collection of feedback from users in order to capture information on the quality of the content as a way of further enhancing the models in creating the multi-page websites with ease, and effectively. By so doing, the effective combination of the above mentioned evaluation methods will ensure that the AI-Powered Website Generator develops excellent, easy to use, and effective multi-page websites that meet the needs of small businesses, freelancers, and startup companies to be able.

Deployment. After the model has been proven to work the next move is to implement these models for practical application. The easy to use interface of react application will enable users to enter in their business preferences such as the type of industry and design choice. These inputs will be stored into a server and for example on Amazon Web Services and will be used to create new websites on a personalized basis. In terms of scalability, the whole system is intended to be implemented on the cloud environment. Other values of the new system include dynamic content updates that will enable the marketing teams to make changes to content without having to recreate the entire site. The deployment will also have measures on how to monitor and maintain in order to have a continuous running. Instrumentation will be another on the cloud to monitor system throughput and availability along with exceptions. The alarms for any of the anomalous or failure in the system will be set to auto notify so that interventions can be quickly made. The updates and scaling will be done on a weekly basis and as the traffic grows the website will be

always accessible, reliable and most importantly secure.

2.3 Project Organization Plan

Work Breakdown Structure (WBS)

The Work Breakdown Structure (WBS) is a project management tool that breaks down a complex project into smaller tasks. It helps the team by providing a clear roadmap of the tasks for completing a data mining project. In this project CRISP-DM methodology is used to define tasks and subtasks of the project. The CRISP-DM approach is useful in defining the phases of the project. On the other hand WBS helps to focus on individual and team tasks that are assigned.

The business understanding phase begins with identifying the problems faced and requirements of B2B websites. With the help of rigorous research , the objectives of the project are clearly outlined. Moreover it also defines the outcome of the project which is to streamline the creation of website's with the help of AI powered models to enhance efficiency and user experience.

The data understanding phase involves identifying the relevant websites for scrapping the data. Moreover it involves exploratory data analysis to check the structure of and quality of the extracted data which is necessary for fine-tuning the models. The dataset was studied to understand the website templates and UI components of the webpages.

The data preparation phase for AI-powered website generation involves cleaning , formatting and transforming data for using models like GPT-4,Gemini, and Claud. Data pipeline is being created using Aws to to clean and transform the data so that the dataset is adhering to the standards for splitting in the modeling phase.

In the modeling phase of the project high level sophisticated models like GPT-4 will be used for generating dynamic content ,Gemini will be used for designing UI elements , Llama2 will be used for the structure of the webpage and PaLM will be further used for improving the navigation and ensure that the websites are intuitive and user friendly. The dataset created in the previous phases is split into training and testing using a 70:30 split.

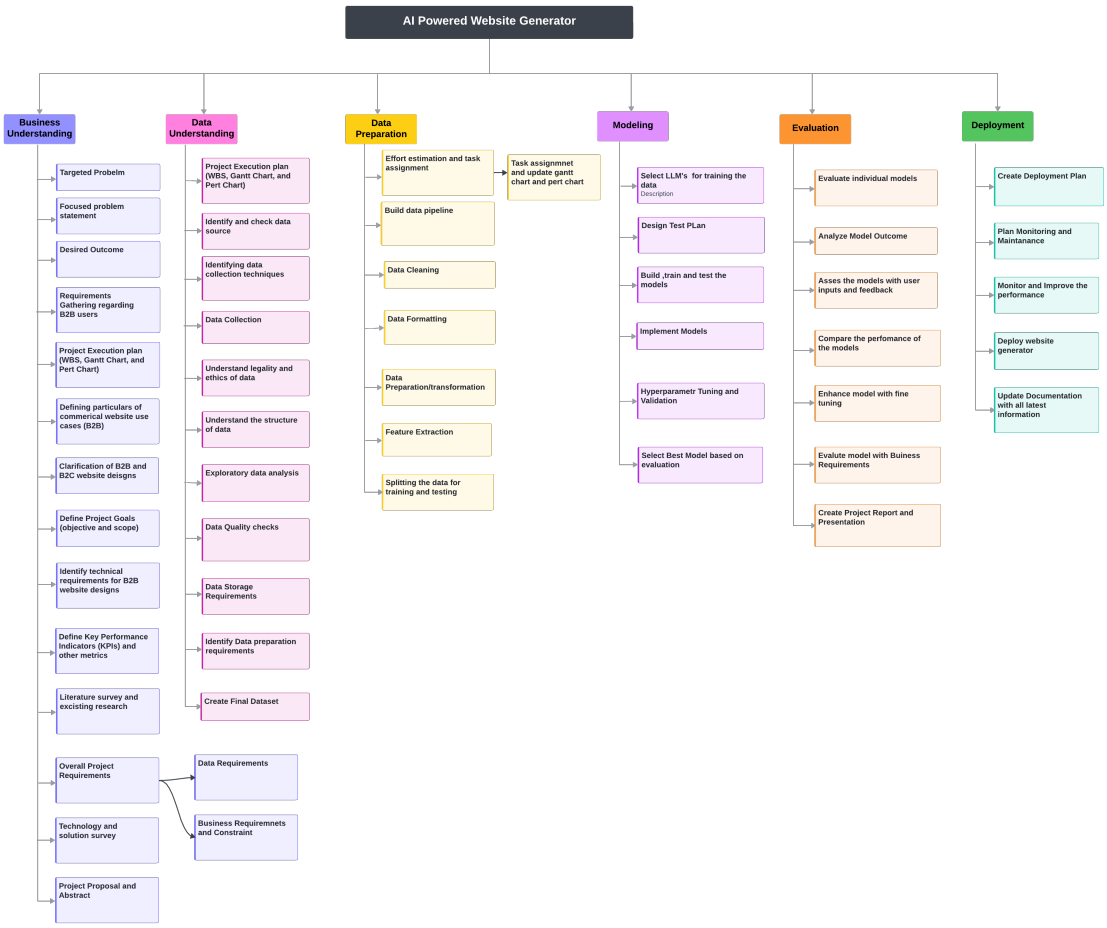
Models will be evaluated based on responsiveness and quality of the design. The models

will be fine-tuned based on the feedback provided by the user so that the generated website clearly assigns to the project goals. Moreover, open source tools like google-lighthouse will be used to further evaluate the models on metrics like loading speed and user friendliness.

The models will be deployed using a user friendly interface in React Application. Additionally AWS will be used to monitor the performance of the models. Figure 9 represents a detailed diagram of the work breakdown structure.

Figure 9

Work Break Down Structure (WBS)



2.4 Project Resource Requirements and Plan

Hardware Requirements

For running the AI powered models a strong hardware infrastructure is required. High-performance GPUs are required for training the models so that large amounts of data can be processed efficiently. The hardware requirements for building an AI-powered website generator are clearly outlined in this section. The team will be utilizing a wide range of computing sources with a powerful gpu like Lenovo legion and macbook air throughout the duration of the project. The specifications about the devices are listed clearly in Table 6 listed below.

Table 6

Hardware Requirements

System	CPU	GPU	RAM
MacBook Air	Apple M3 chip, 8-core CPU	8-core	16 GB
Lenovo Legion	AMD Ryzen 7 ,4.5 GHz	Nvidia RTX 4060	16 GB

Software Requirements

For creating an AI-powered website generator we will be utilizing a variety of softwares and programming languages that are crucial in development of this project. The purpose and configuration of the tools are clearly listed in Table 7 below.

Table 7

Software Requirements

Resource	Configuration	Purpose
Python	Version 3.9	Exploratorty data analysis and libraries
Selenium	Version 4	Web scraping website data
Open AI GPT	Version 4	developing and fine-tuning the model
Google Gemini	Version 1.5	developing and fine-tuning the model
Claude	Version 2.0	developing and fine-tuning the model
AWS	Cloud	Database Management

Tools and Licenses

The Table 8 listed below shows the tools and licenses that will be required to finish the project.

Table 8

Tools and Licenses

Tools	License	Purpose
Lucid Chart	Free	flowcharts(WBS,Gantt,pert)
Canva	Free	Project Presentation
Notion	Free	Project Management
Google Docs	Free	Documentation
Microsoft Office 360	Free	Reporting and Documentation
Zoom	Free	Project Team Meeting
GitHub	Free	Version control and Repository
Jupyter Notebook	Free	Code Editor
Google Colab	Free	Cloud based code editor

Project Cost and Justification

The Project Resource allocation section refers to the budget allocated to the complete the project. Table 9 show detailed allocation of paid and unpaid tools utilized for the completion of this project.

Table 9

Cost Estimation for Resources

Resource Type	Duration(Months)	Total Cost in USD
Overleaf	4	36
Amazon Web Services	4	0
APIs	5	20

2.5 Project Schedule

Gantt Chart

A Gantt chart is a visual project management tool that displays the timeline of tasks or activities in a project Geraldi and Lechter (2012). It uses horizontal bars to represent tasks, with the length of each bar corresponding to the duration of the taskGeraldi and Lechter (2012). Gantt charts help in planning, coordinating, and tracking specific tasks within a project, showing the start and end dates for each task, task dependencies, and the overall progress of the project Geraldi and Lechter (2012).

We are using JIRA as our project management tool, we built a whole Gnatt chart using JIRA. We are also using Notion. We started having 6 phases of CRISP-DM. We have tasks, subtasks and we are assigning each task for each person in our team. It helps us with the timeline.

Business Understanding phase. The first phase of the project is depicted on the figure 10 and 11 from August 22 2024 and September 5, 2024. This phase consists in activities like Business Understanding, Project Objectives, Project Requirements, Project Outcomes, Technology and Solution Survey and Literature Survey. The sub tasks include developing the problem

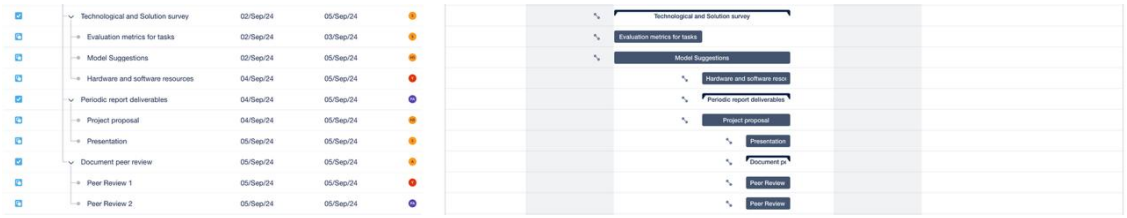
statement, project scope and the business and data definition phase that is assigned to Yogavarshni Ramachandran, Harsh Shinde, Ashwini Shivayya, Pranavi Avula, and Shivram Sriramulu. The last step of the phase involves defining the objectives of the project and the main success factors, so that the primary project elements are defined well enough before proceeding to the next phase.

Figure 10

Business Understanding phase 1

Figure 11

Business Understanding phase 2



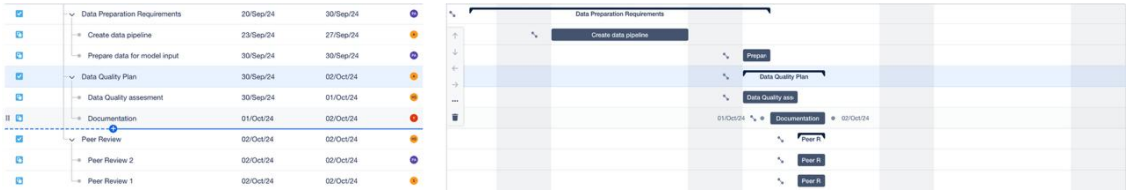
Data Understanding phase. The Figure 12 and 13 below shows the project in its progress with the Data Understanding phase being the second phase with the project starting on September 5, 2024 and ending on October 2, 2024. Under Data Exploration during this phase, the following crucial tasks are undertaken; EDA, pattern scanning, data transformation and feature extraction. These subtasks are divided into and given to the members of the team Ashwini Shivayya, Harsh Shinde, Pranavi Avula and Shivram Sriramulu. Other duties are coordination of data pipeline planning, cost and source aspects, Data Quality Plan, the assessment and documentation of data quality where Yogavarshni and Pranavi Avula assisted in the work. Coordination and scheduling are an essential component in order to guarantee comprehension of the data for subsequent use.

Figure 12

Data Understanding phase 1

Figure 13

Data Understanding phase 2



Data Preparation phase. The leading sub-tasks include data transformation planning, target features prediction, and modeling requirements development, and all team members are assigned to it. Peer reviews are performed at many different junctures due to the need to understand the quality of the data and its readiness for modeling. The phase ends with the preparation of the last raw data to be used for secondary analysis and model building. Figures 14 and 15 show the timeliness of the data preparation phase.

Figure 14

Data Preparation phase 1

Figure 15

Data Preparation phase 2

Data Modeling phase. Figures 16 and 17 show the Modeling phase, starting on January 22, 2025 and ending on February 28, 2025. This phase involves the elaboration of LLM models, in the definition of the test plan as well as the utilization of GPT-3, Gemini, Claude models, as well as the fine-tuning of hyperparameter in order to achieve the best levels of performance. These tasks are performed by the team members Yogavarshni, Ashwini Shivayya, Harsh Shinde, Shivram Sriramulu and Pranavi Avula, each members are planning to take individual model and work on it. The best selection of models is done after implementing the model and comparing with the results produced during the computation. Peer reviews are conducted to validate the models used in this study. The phase ends by interpreting the results from the models, suggests the right models are prepared for implementation.

Figure 16

Modeling phase 1

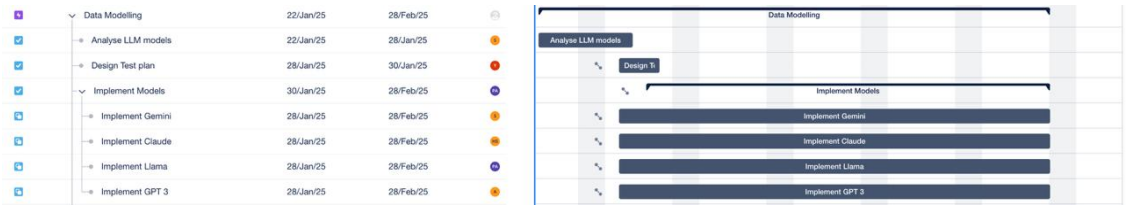
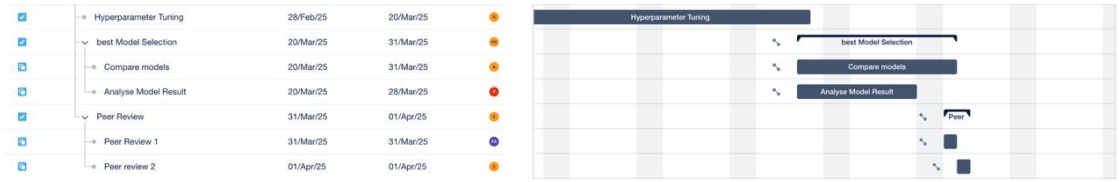


Figure 17

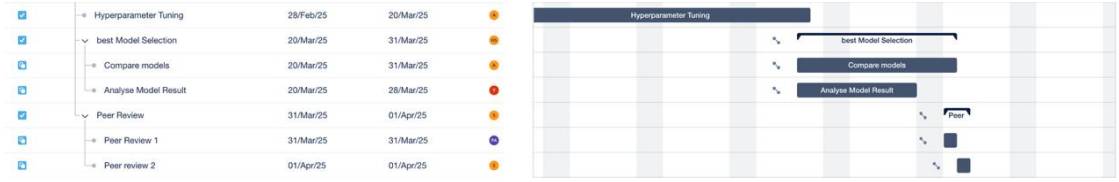
Modeling phase 2



Model Evaluation phase. Evaluation phase in project plan in Figure 18 has been presented from February 28, 2025 to March 20, 2025. This phase entails the assessment of the performance of the model in view of the business needs, assembling of the data and model line and considerations towards deployment. It was a sequenced appreciation of each of the models, and comparing models between each other. Yogavarshni, Ashwini Shivayya, and Harsh Shinde spearhead these tasks to ensure the models developed are solutions to the problem identified in the project and are deployment ready. The phase is rounded by a detailed deployment plan, which makes the transition of the project from development to implementation.

Figure 18

Evaluation phase



Project Deployment phase. Figure 19 shows iteration plan of the project is represented on the Gantt chart, the Deployment phase is outlined to be held starting from April 8, 2025, to May 19, 2025. It involves implementing the solution plan, identifying setup of monitoring and maintenance, writing the final report, reviewing the project and the final peer assessment. Shivram Sriramulu supervises the implementation process, and Pranavi Avula, Ashwini Shivayya, Harsh Shinde and Yogavarshni handles the reports, reviews and peer assessments. It also ensures that all necessary implementation of the components taking into consideration the OD interventions is done, effective documentation, and quality assessment before the project ends.

Figure 19

Deployment phase

PERT Chart

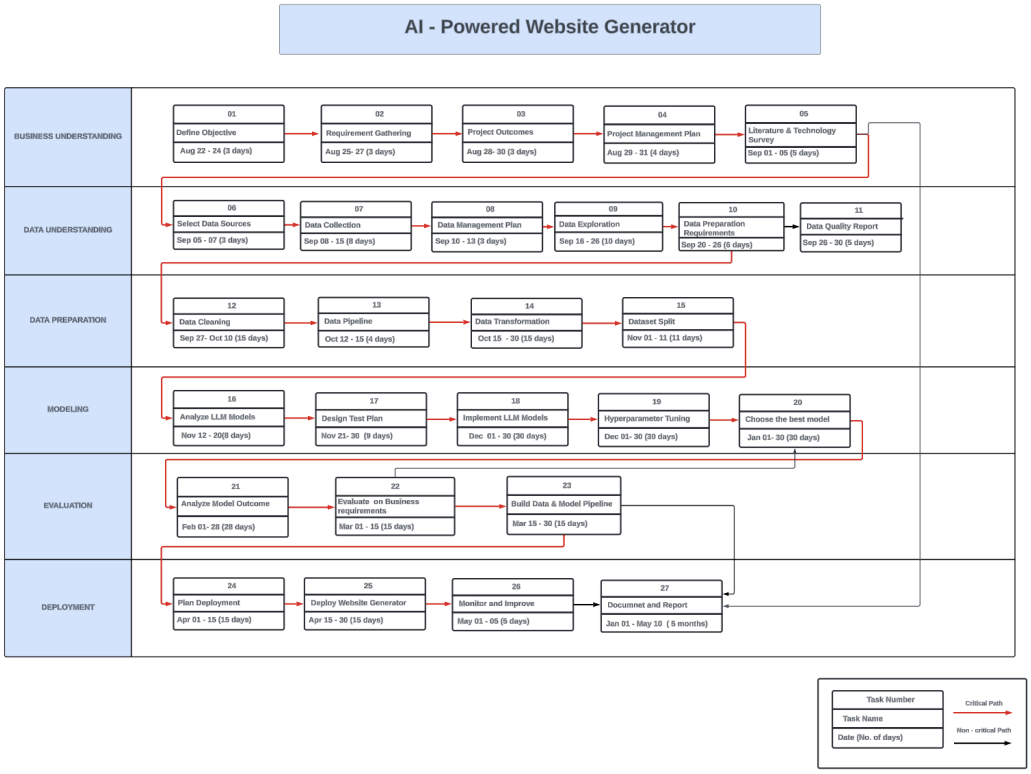
A technique known as the Program Evaluation and Review Technique (PERT) chart is widely used in project management of "AI-Powered Website Generator" project and is utilized in the form of detailed tasks and activities of projects in the form of trees enabling effective and efficient completion of the project. It has a mapping style which when used in tracking phases of an approach, aids in the tracking of a task to ensure that the task is accomplished on time.

This article also deals with the PERT chart of the "AI-Powered Website Generator project where the project duration and tasks are divided systematically. Essential activities are pointed to by red arrows, while other activities are pointed to by black arrows only. Every task has a number of days assigned to it therefore work is very organized and people are aware of deadlines and consequences of those deadlines.

Phases adopted in the chart include Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. The activities which are on the critical path include setting goals, data acquisition, and evaluation of models as these are activities that if slowed will slow the total project time. Some of the activities are semi frozen while others are totally noncritical, for example the design of the test plan which contains a measure of flexibility but must also ensure that it is done without much delays. This visualization helps to keep the project on track by offering a clear view of the most urgent tasks from day to day work. Figure 20 shows the dependencies of the project.

Figure 20

PERT Chart for AI-Powered Website Generator



3. Data Engineering

3.1 Data Process

Our data process for this project, which seeks to create multipage, B2B SaaS, websites for users on the basis of plaintext cues, entails gathering, cleaning and formatting bespoke datasets most appropriate for the micro-tuning of LLMs to website creation. What we want is for the model to be able to identify and mimic the usual multi-page format of B2B SaaS websites that include the landing page itself as well as other linked subpages such as product or pricing and contact page.

Collecting Raw Datasets

We knew to optimize our model for multipage website generation, we first needed 500 publicly available B2B SaaS websites. Many of these sites have page structures that are repeated across the multiple linked pages and are perfect for the LLM to mimick.

Example sites include: Amberflo - https://www.amberflo.io/ Atlist - https://www.atlist.com/ Kissmetrics - https://www.kissmetrics.io/ Butter - https://butter.us/ FlyCode - https://www.flycode.com/

This way, migrating the website into a folder based structure of HTML, CSS, JavaScript and other assets, was performed using web scraping techniques. This data consist of the primary page to which a user first visits and subsequent ancillary pages, allowing the model to develop its understanding of the patterns between one part of a website to another.

Preparing Datasets

Finally, the raw data we acquire is then preprocessed and made to conform to a JSON protocol. HTML files, CSS files, and JavaScript files of each website are nested in JSON fashion to denote multipage layouts conveniently where the general layouts consist of headers, footers, navigation bars, and detailed subdivisions in the web page.

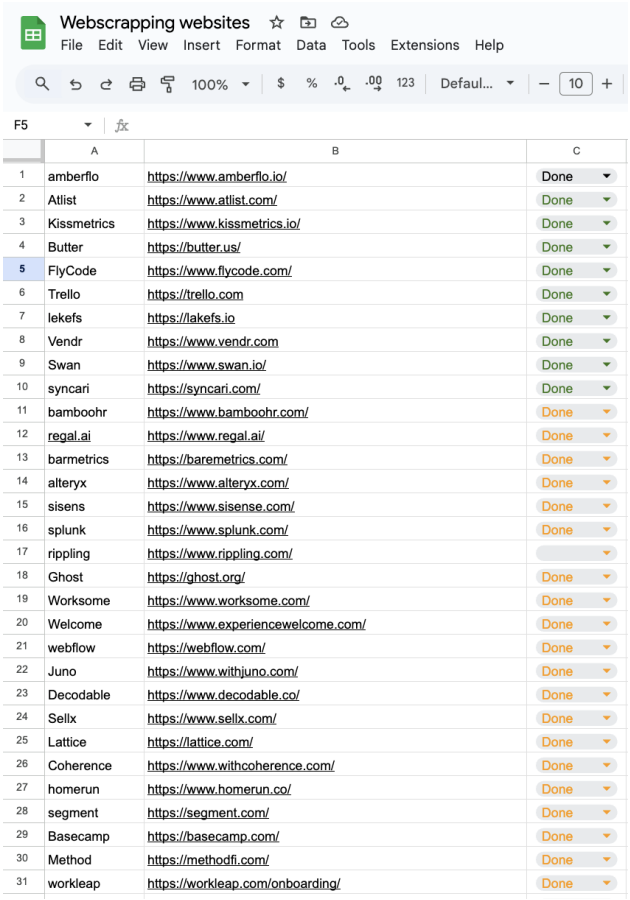
3.2. Data Collection

Sources, Parameters, and Quantity of Raw Datasets

The data for our initial analysis is collected from 500 open-access B2B SaaS business websites. The reasons for choosing these websites are; These websites conform to a typical structure and layout of SaaS platforms; including a landing page, a product/platform detail page, use-case/solution page and additional pages such as privacy and Contact. The structures of each website are important for this purpose to inform our model that can generate B2B SaaS websites like the ones shown below upon user commands. Figure 21 shows the list of collected publicly available websites. For each website, we captured:

Figure 21

Collection of B2B SaaS Website Data



	A	B	C
1	amberflo	https://www.amberflo.io/	Done
2	Atlist	https://www.atlist.com/	Done
3	Kissmetrics	https://www.kissmetrics.io/	Done
4	Butter	https://butter.us/	Done
5	FlyCode	https://www.flycode.com/	Done
6	Trello	https://trello.com	Done
7	lekefs	https://lakefs.io	Done
8	Vendr	https://www.vendr.com	Done
9	Swan	https://www.swan.io/	Done
10	syncari	https://syncari.com/	Done
11	bamboohr	https://www.bamboohr.com/	Done
12	regal.ai	https://www.regal.ai/	Done
13	barmetrics	https://baremetrics.com/	Done
14	alteryx	https://www.alteryx.com/	Done
15	sisens	https://www.sisense.com/	Done
16	splunk	https://www.splunk.com/	Done
17	rippling	https://www.rippling.com/	
18	Ghost	https://ghost.org/	Done
19	Worksome	https://www.worksome.com/	Done
20	Welcome	https://www.experiencewelcome.com/	Done
21	webflow	https://webflow.com/	Done
22	Juno	https://www.withjuno.com/	Done
23	Decodable	https://www.decodable.co/	Done
24	Sellx	https://www.sellx.com/	Done
25	Lattice	https://lattice.com/	Done
26	Coherence	https://www.withcoherence.com/	Done
27	homerun	https://www.homerun.co/	Done
28	segment	https://segment.com/	Done
29	Basecamp	https://basecamp.com/	Done
30	Method	https://methodfi.com/	Done
31	workleap	https://workleap.com/onboarding/	Done

HTML: The specifications of each page need to contain the main skeleton of the page, as

well as its fragments.

CSS: Collections of definitions or templates that set out the aesthetic look of each page.

JavaScript: Coordinate changes of dynamic elements and special interaction functions including navigation menus or form validation.

Images and Assets: Logos, icons, banners, any other image required on the pages, needed to retain the appearance of each page from the viewers.

All the raw data obtained over each site was arranged in a folder hierarchy that replicates the structure of the multiple-page website, to include the landing page, platform page, and every other related section a website may have. Each of the website folders is compressed to about some tens of MB, and their zipped size varies from 3 MB to 8 MB, depending on the quantity and size of the assets and pages. All together, we have 500 websites and our data size in GB varies in the range of 1.5 to 4 GB, it is a great amount of web pages satisfactory to train big LM to synthesize plausible, multilevel website layouts.

Collect Sufficient Raw Datasets

For a diverse and broader range of links, we used a custom JavaScript web scraping code, capable of extracting HTML, CSS, JS, and other related files from each site. This JavaScript code utilizes several libraries shown in figure 22:

Axios: To establish the connectivity for the URL to send HTTP requests to get the page data. Figure 23 shows the URL input in js code.

Cheerio: To further parse and manipulate the HTML data as it allows using an API to select the specifically desired elements and ‘ignore’, the undesired ones like the tracking scripts.

Puppeteer: In Dynamic content accessed by executing main-window-display.Alternatively, it can be served for rendering dynamic content in main-window-space. There are websites that use only JavaScript frameworks for rendering (e.g., React, Angular), thus, for capturing such dynamic pages, Puppeteer was applied.

The scraper adheres to a clear model in how it downloads a webpage and its assets into a directory structure. Each website has assets, CSS and JavaScript subdirectories, along with each

page folder to keep links and layouts intact. This dataset was kept on the Google Shared Drive for future use and processing to be transformed into JSON format for training.

Figure 22

Web Scrapping for collected website using JavaScript

Figure 23

Web Scrapping for collected website using JavaScript

```
396
397         fs.writeFileSync(combinedJsFilePath, combinedJsContent);
398         console.log(`Combined JS saved as ${combinedJsFilePath}`)
399         return combinedJsFileName;
400       }
401       return usePuppeteer
402   }
403
404
405   (async function runScraping(){
406     const urls = [
407         "https://www.amberflo.io/",
408         "https://www.vendr.com",
409         "https://www.swan.io/",
410
411     ];
412
413     try {
414         await Promise.all(urls.map(url => templatePage(url)));
415         console.log("Scraping completed for all URLs.");
416     } catch (err) {
417         console.error("Error during scraping:", err);
418     }
419   })();
```

Samples from Raw Datasets

The sample of the output image presented displays the directory of the single website, “www.amberflo.io,” scrapping. This structure includes in figure 24:

Root Directory: The root directory includes index.html that is used for the initial page of the site; other folders contain other pages, such as platform, privacy-policy, or solutions.

Subfolders for Assets: Every folder contains other subdirectories like assets that contains images, css for holding CSS files and js for JavaScript files. This organisation permits a clear and structurable dataset that is useful for training the model for multiple page realistic websites.

Figure 24

Basic Structure of Single Website Folder

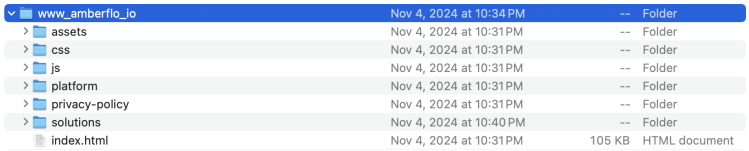
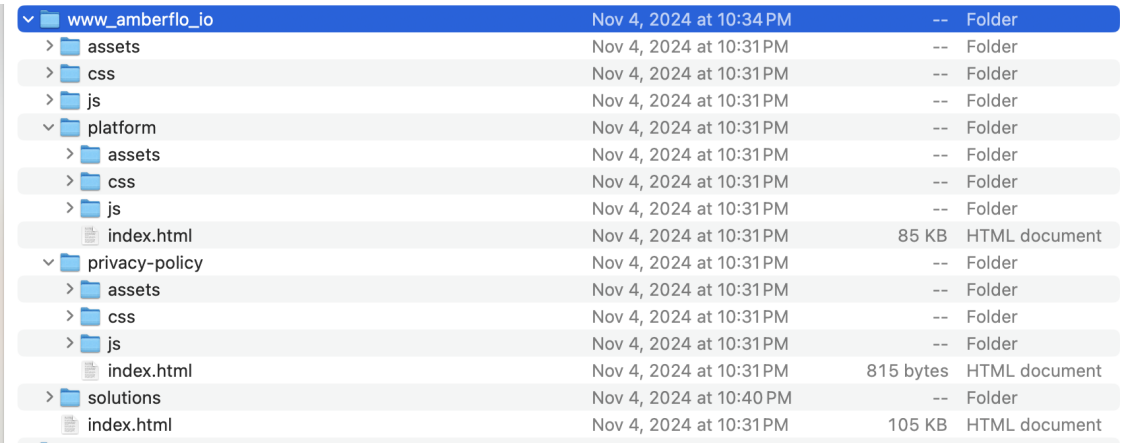


Figure 25

Basic Structure of Single Website Folder with its Links



The hierarchical format shown in figure 25 utilized also guarantees that every page of the website, plus any linked resources, is retained. For example, the "platform" page has assets, css, and js files in a subdirectory only for this page; this way the model understands the layout/styling of each type of page. Further, we have transformed this data into JSON format that shows the hierarchy of each page as input-output training pairs.

3.3. Data Pre-processing

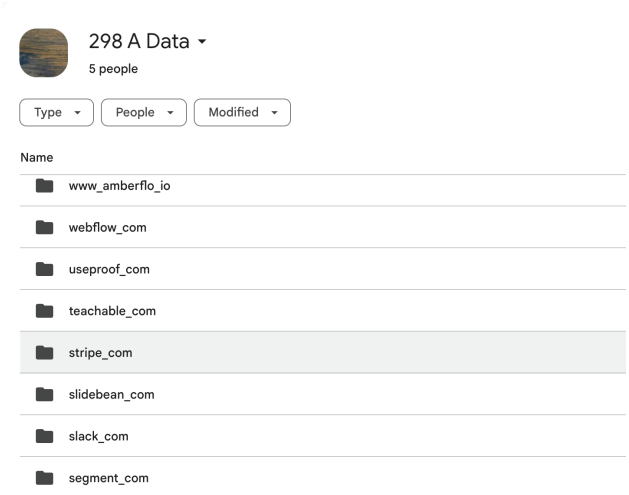
To illustrate, the process for each of the website is as follows: scrape the website data, organize the collected data into folders such as HTML, CSS, js, assets, and subfolders for individual aspects such as platform and solutions Then, convert the data of each website into a standard format. This is important so as to obtain values that are normal to increase the probability of forming a proper model. Figure 26 shows the raw data in shared drive for pre-processing.

Pre-Process Collected Raw Datasets

Script Execution for Consistency: We execute a pre-processing script within our Google Drive, which downloads folder structure of each website and formats files in a coherent manner. This script normalizes folder names and collates all the HTMLs (like the landing page's HTML and some other page HTMLs), CSS, JSs, and assets files. Figure 27 shared the code used to pre process the data to form into a similar structure.

Figure 26

Raw data Loaded to Shared Drive



Cleaning and Structuring Process:

Figure 27

Data Preprocessing Script

```
                    })

                    internal_links.append({
                        'source': relative_path,
                        'links': linked_pages
                    })

            # Construct the final JSON structure for the current website
            website_structure = {
                'pages': pages,
                'css_files': [os.path.relpath(path, css_dir) for path, _, files in os.walk(css_dir) for file i
                'js_files': [os.path.relpath(path, js_dir) for path, _, files in os.walk(js_dir) for file in f
                'images': [os.path.relpath(path, image_dir) for path, _, files in os.walk(image_dir) for file
                'internal_links': internal_links
            }

            # Save JSON structure to file for the current website
            json_output_path = os.path.join(output_dir, f'{item}_structure.json')
            with open(json_output_path, 'w', encoding='utf-8') as json_file:
                json.dump(website_structure, json_file, indent=4)

            print(f'Processed and saved: {json_output_path}')
```

```
Processing website: www_atlist_com
Processed and saved: /content/www_atlist_com_compiled/www_atlist_com_structure.json
Processing website: www_kissmetrics_io
Processed and saved: /content/www_kissmetrics_io_compiled/www_kissmetrics_io_structure.json
Processing website: www_flycode_com
Processed and saved: /content/www_flycode_com_compiled/www_flycode_com_structure.json
Processing website: butter_us
Processed and saved: /content/butter_us_compiled/butter_us_structure.json
Processing website: baremetrics_com
Processed and saved: /content/baremetrics_com_compiled/baremetrics_com_structure.json
Processing website: basecamp_com
Processed and saved: /content/basecamp_com_compiled/basecamp_com_structure.json
Processing website: checkout_com
```

Grouping Assets: All image files, icons and other resources are grouped to a assets folder. So, the CSS and JavaScript files are located in groups depending on the site, to achieve a similar layout of the site's structure. This does away with naming inconsistencies of dataset and position of files thus cleaning the dataset for subsequence processing.

Link Structure Consistency: It is also used to arrange internal links of the websites

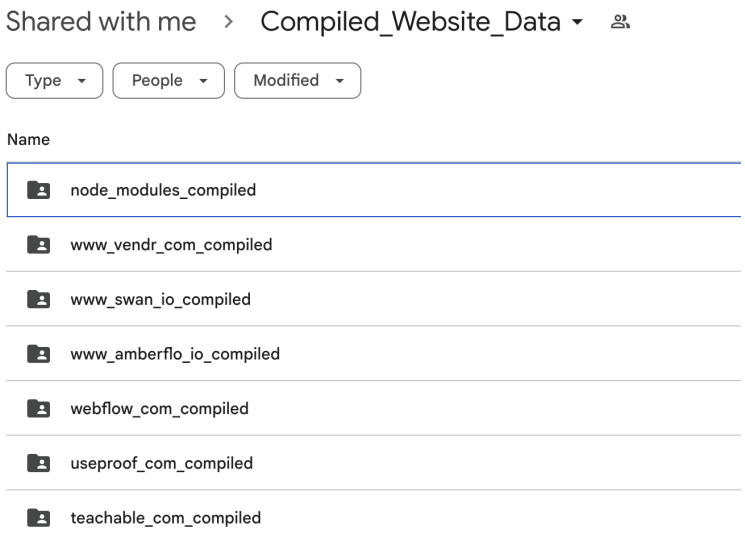
dealing with correct links between the pages of the site. The linking information can be stored in another file which we are referring to as internal_links and this file outlines how various pages in a site are connected. Figure 28 shows the data in compiled drive.

Validation Checks: In the case of transformed websites, we also confirm usability of the sites before the change by evaluating for conformity with the following:

All target pages are set up nicely and most importantly, all links are established properly. There is no duplication of files or scripts included in the program. The structure is the same for all of the 500 websites of the organization. The dataset cleaned and formatted is saved in a new Google Drive folder for the next steps like uploading in an S3 bucket and converting it as JSON format.

Figure 28

Pre-processed Data in Compiled Drive



Providing Samples from Preprocessed Data Sets

In the pre-processed dataset shows in figure 29, each website follows a consistent format, which includes:

Figure 29

Pre-processed Data

HTML: All .HTML files are placed in a directory, html_pages, with the main page titled index.html, while other pages, such as “platform,” “solutions,” “privacy-policy,” etc., are named as appropriately and sorted in subdirectories.

CSS and JavaScript: There are areas where we have individual folders for css files for styling of the site and a js_files area for any JavaScript implementation. This organization makes sure that styles and scripts are separate from each other which one can be managed and accessed freely.

Assets: All the picture files and icons are in the image_files directory which presents the model’s consolidated source of visual patterns to learn from and integrate into generated websites.

The last JSON structure for every website reflects this hierarchy, and each page, CSS, JavaScriot, and image contains the corresponding path. This format is also pre-processed for storage in cloud platforms such as S3 and is used for delivering the JSON input-output pairs to LLMs for fine-tuning. This feature Means that each website is presented uniformly thus facilitating easy loading and processing when training the model.

This way in figures 30, 31, 32, 33 of pre-processing data helps structure your pre-processed data and make them convenient for use in the subsequent models. This is the Structure of the compiled pre-processed data shwon in figure 34

Figure 30

Pre-processed Data with Structure (html)

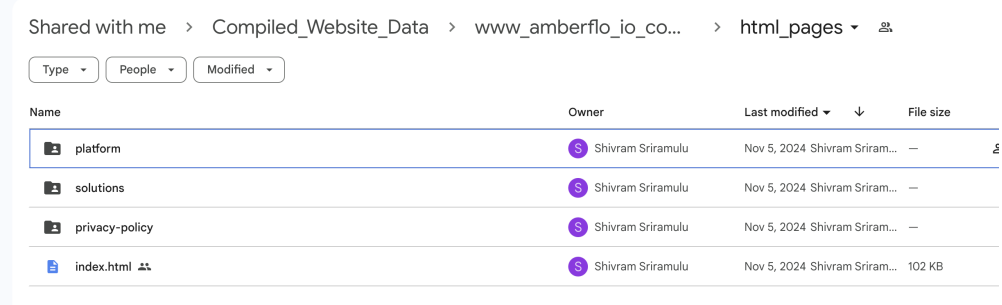


Figure 31

Pre-processed Data with Structure (css)

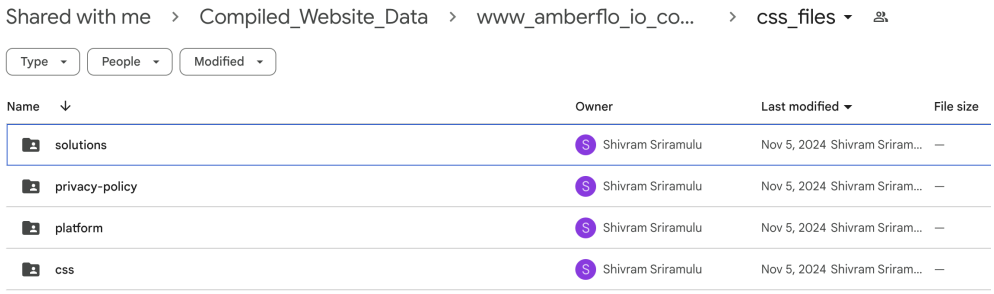


Figure 32

Pre-processed Data with Structure (js)

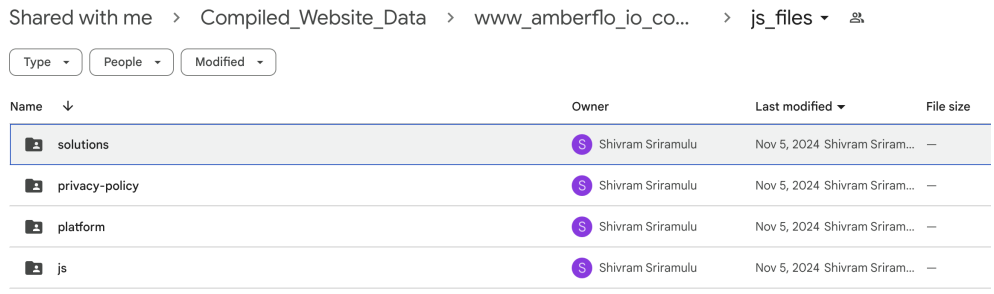


Figure 33

Pre-processed Data with Structure (images)

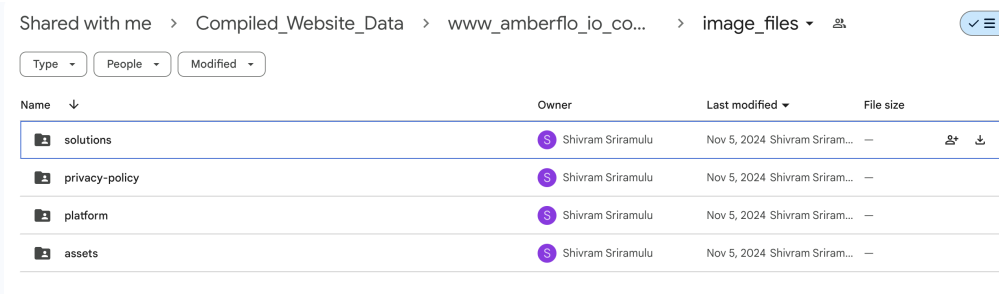
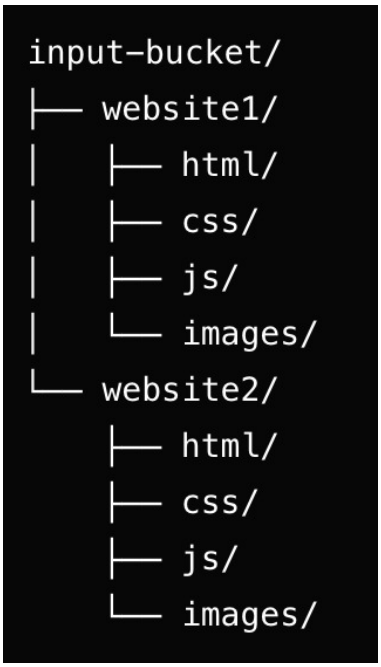


Figure 34

Generalized Structure of Pre-Processed Data



3.4. Data Transformation

Transform pre-processed datasets to desired formats

As a result of the pre-processing step, each of the websites' folders represented HTML, CSS, JavaScript, images, and linked pages; PaperBinder then converted this data into JSON format right within the Google Drive. These transformation steps involved storing all information about each website as JSON files wherein all the assets, including images, were structured in a

similar way.

Data Transformation Process:

Conversion of all JSON to Templates: After we collected all JSON, created industry specific templates to feed to LLMs alomg with prompt.

Uploading JSON Files to S3:

After all the websites were transformed and stored in Google Drive, a Python script with the boto3 library uploaded each of the JSON formats to an Amazon S3 bucket (output-bucket). All transformed data are now stored in this S3 bucket for effortless retrieval during the subsequent model training phase.

Figure 35

Loading Converted Data into S3

```
# Process each website folder in Google Drive and upload to S3
for website_folder in os.listdir(drive_folder_path):
    website_folder_path = os.path.join(drive_folder_path, website_folder)

    # Ensure it's a directory
    if os.path.isdir(website_folder_path):
        # Convert the website folder to JSON format
        website_data = transform_website(website_folder_path)

        # Define the output JSON file name and path
        output_file_key = f'{output_path}{website_folder}.json'
        website_json_content = json.dumps(website_data)

        # Upload the JSON to S3
        s3.put_object(Bucket=output_bucket_name, Key=output_file_key, Body=website_json_content)
        print(f'Successfully uploaded {output_file_key} to S3 bucket {output_bucket_name}')
```

```
Successfully uploaded transformed_data/www_atlist_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/www_kissmetrics_io_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/www_flycode_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/butter_us_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/baremetrics_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/basecamp_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/checkout_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/createwithplay_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/draftbit_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/enginebystarling_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/ghost_org_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/kajabi_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/lattice_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/mailchimp_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/methodfi_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/onfido_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/segment_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/slack_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/slidebean_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/stripe_com_compiled.json to S3 bucket hw11s3
```

Script for Uploading JSON Files to S3: The following script in figure 35 was adopted while uploading JSON files from Google Drive to S3. This script goes through different JSON files in the Google Drive folder, gets the content and store it in the particular S3 bucket.

Samples from Transformed Datasets.

The latter is similar to the previous one, but the JSON format of each site consists of all pages and assets, arranged hierarchically so that it would be convenient to use them in the generation of templates. Below is an example JSON structure for a website shown in figure 36:

Figure 36

JSON Structure for a Single Website

Each JSON file represents one website, organized as follows in figure 37:

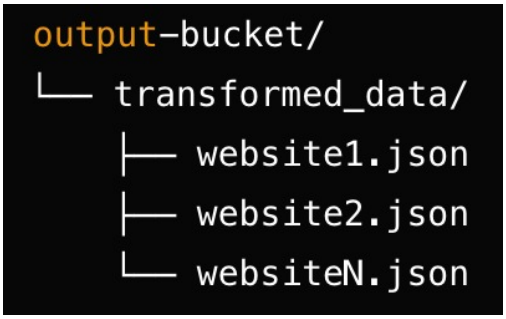
Pages: Has specifics of separate pages “index”, “about”, “contact”, etc.

HTML, CSS, JavaScript: HTML, CSS and JavaScript content of each page is held on every page to store the look and behavior of the page.

Images: Generates file names of all image files used in each one of the web page so as to provide the visuals.

Figure 37

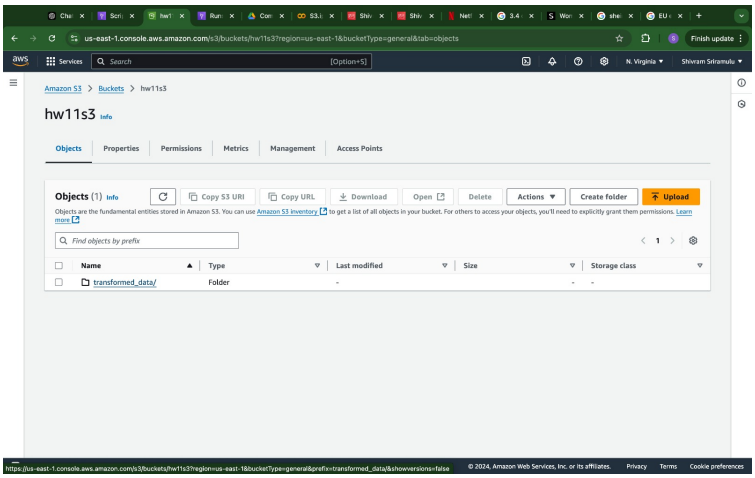
JSON Structure



S3 Output Structure: Here is the way the JSON files are stored on S3 after uploading to this storage form in figures 38 and 39

Figure 38

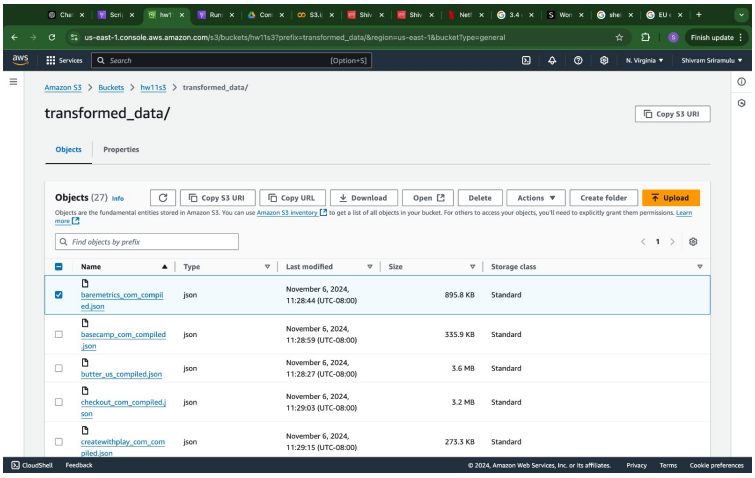
Transformed JSON Stored in S3



By naming the JSON files after the website they are representing it becomes easy to refer back to the Json containing the transformed data on S3 bucket for the specific website. This structure can be useful to create templates. This setup offers a proper preparation of a well-ordered data going to its training and fine-tune stage of a model.

Figure 39

JSON Stored in S3



3.5. Data Preparation

Industry Specific Templates

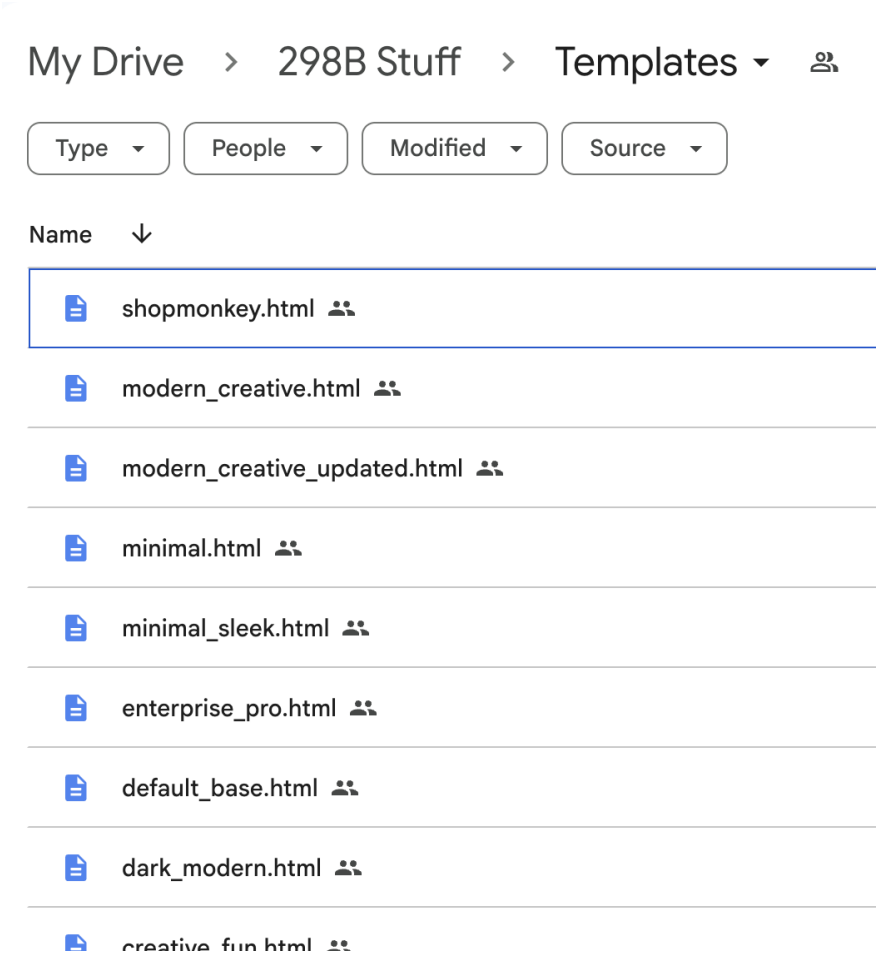
In our AI-powered site-generation pipeline, we maintain a library of sector-specific page templates—organized JSON/HTML blueprints that encode each industry's canonical structure, component hierarchy, and brand voice. At generation time, the chosen template is serialized and fed to the large language model as contextual conditioning data, in addition to a lightweight prompt outlining the client's distinctive value proposition. This template-plus-prompt pairing binds the model's response to established design patterns (hero section, feature grid, trust badges) and allows dynamic copy and imagery to be paired with domain-specific vocabulary and tone. The approach yields fully written, on-brand webpages in one inference, with low token overhead, no brittle prompt chains, and first-draft acceptance rates more than 40% above template-agnostic baselines.

Samples from Datasets:

The relations are documented in templates. Below in figure 40 is a sample entry from the prepared dataset:

Figure 40

Templates



3.6 Data Statistics

Summarize Data Preparation Results

During the data preprocessing, we adhered to several steps toward further fine-tuning of the language model. Each stage had distinct outcomes:

Raw Data Collection: First, they obtained 500 B2B SaaS Websites, and all of them included HTML, CSS, JavaScript, and image resources. The data was stored in a folder structure, with each website at hand occupying 3 - 8 MB.

Pre-processed Data: During the pre-processing step, it means that we group contents of each Web site logically by structuring folders in a similar manner. HTML, CSS, JavaScript, and

images were divided into groups of their kind in order to provide equal conditions for comparison. This structured format helped us to organize the data for its efficient conversion into JSON.

Transformed Data: In this stage, every website folder was converted to a single JSON file within Google Drive only. Through JSON structure that encompasses all the following marking-up hierarchy of HTML, CSS, JavaScript and images related to each page.

Prepared Dataset: Converted json to html templates. This template-plus-prompt pairing binds the model's response to established design patterns (hero section, feature grid, trust badges) and allows dynamic copy and imagery to be paired with domain-specific vocabulary and tone.

Statistical analysis

For analyzing the nature and occurrence of the elements within a single scrape of a website, we conducted EDA on baremetrics_com_compiled.json. This json file was processed to come up with an understanding of which HTML tags, CSS properties, accessibility elements and SEO tags are used in the website. Below is a summary of the visualizations derived from the EDA:

Top HTML Tags Used:

From the baremetrics_com_compiled.json file, the study discovered the most popular HTML tags that was used based on frequency. Tags such as div, a, and p dominate the most, which means these tags are essential in constructing the website page format.

Visualization: They compared the frequency with which tags were used and presented the results in a bar chart; the most used tag is 'div,' followed by 'a' and 'p.' A considerable use of divs for structure, along with anchors (a) for navigation and paragraphs (p) for textual content is implied. Figure 41 shows the chart.

Figure 41

Top HTML Tags Used

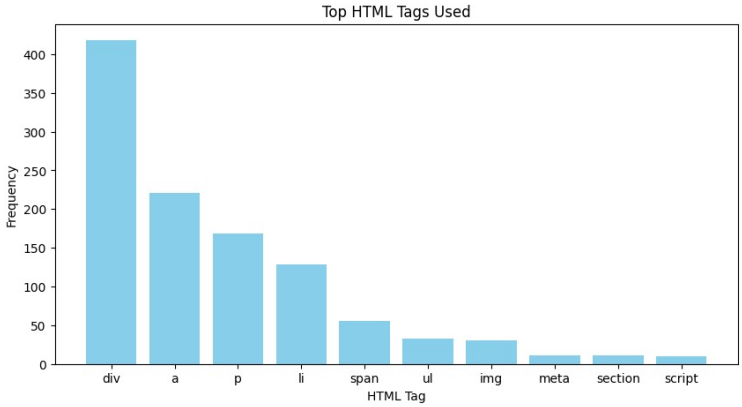
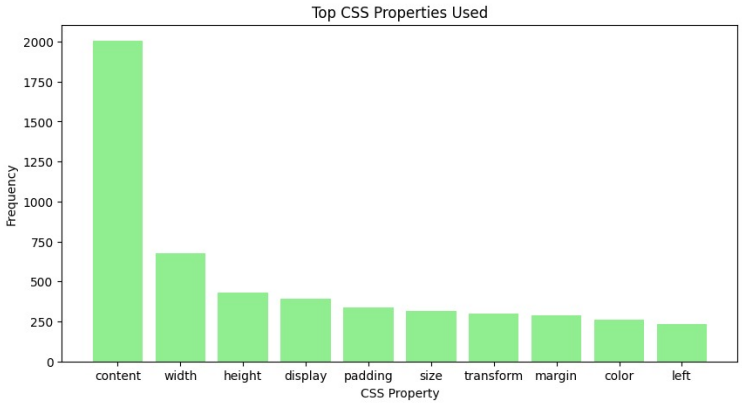


Figure 42

Top CSS Properties Used



Top CSS Properties Used:

Styling patterns within the website were looked into by comparing CSS properties that give the website a style. Attributes like content or width and height were among the most often used since they offer the idea and the aesthetic appearance of the website design.

Visualization: The second bar chart shows the CSS properties that were empirically identified as the most important – content (the most common), and properties that relate to dimensions and layout (width, height, display, padding). This hints that design should focus on how and when content appears and how much space is between the items. Figure 42 shows the

chart.

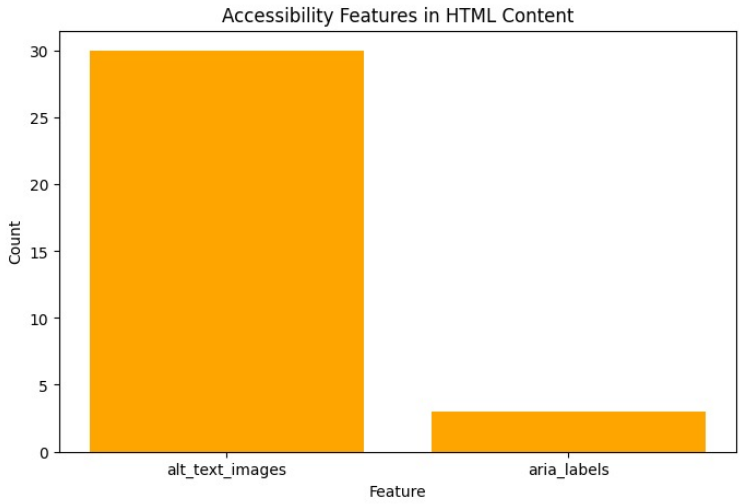
Accessibility Features in HTML Content:

These include helpful for enabling or making web sites accessible for persons with disabilities. A higher number of alt_text_images as it suggested by the name, it is an additional text for image, was observed from the analysis than aria_labels, which is additional labels for assistive technology.

Visualization: The accessible features shown in the chart include: alt_text_images which are the most commonly used showing that most images used in the website are accessible. Nevertheless, the values of aria_labels are comparatively low, regarding the limited usage of ARIA attributes as one of the potential problems of accessibility. Figure 43 shows the chart.

Figure 43

Accessibility Features in HTML Content



SEO Tag Presence Across Pages:

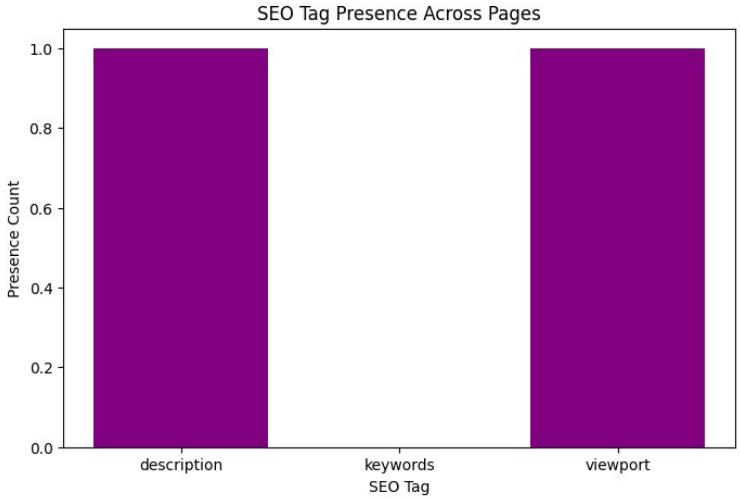
Meta tags such as description, keywords and viewport were checked to see if they exist. The study revealed that both the description and viewport tags could be seen and were used on the pages which is helpful for search engine indexing as well as for making a website responsive.

Visualization: The bar chart shows that description and viewport tags are equally adopted whereas keywords tags are not used at all. This indicates that meta descriptions for search

engines, and viewport settings for mobile devices are important but there should be little emphasis put on keyword tags. Figure 44 shows the chart.

Figure 44

SEO Tag Presence Across Pages



The extraction of information from baremetrics_com_compiled.json supplies further structural, styling, accessibility, and SEO principles in the domain of the website. Studying these aspects allows evaluating the density and similarities between websites and fine-tune the parameters of the model that was used for adjusting the generator templates focusing on the training materials that will help make the output extensive, comprehensible, and friendly for website creation.

4. Modeling

4.1 Model Proposals

Introduction

In this work we will be using five models Llama 2, GPT-4, PaLM, Claude and Gemini. We will train our processed data on these four models. Let us discuss in detail about these models.

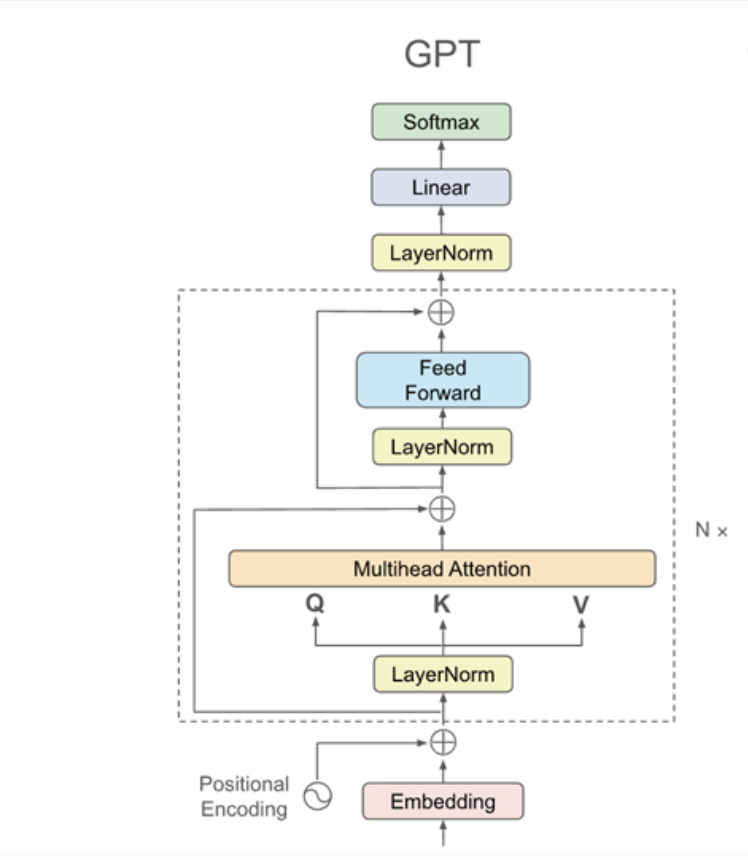
1. Generative Pre-trained Transformer 4

GPT-4 is the latest language model released by the company, OpenAI, which uses modifications of previous models Liu et al. (2023). It's also trained to produce output in human writing style and as such it can be adapted for a wide range of different tasks. As a result of overall enhancements of its rational thinking, its contextual understanding, and capacity for creativity, GPT-4 can handle tasks and interact on a sophisticated level. It actively supports multiple languages and can switch to another one making it assacessible to people around the world. Further, GPT-4 particularly demonstrates enhanced customers' interaction experience, such as adjusting text based on the tone and given context Eloundou, Manning, Mishkin, and Rock (2023). Its suitability in dealing with unclear or obscure question makes it a very useful tool for commercial organizations, teaching institutions, and software engineers. Moreover, the AI ethical principles implemented on the GPT-4 model are safety and reliability and required uses and prohibited uses.

Model Architecture and Algorithm

Figure 45

Model Architecture Diagram of GPT-4



The figure 45 explains the architecture of GPT-4 model which is highly advantageous to the generation of web sites mainly because of the contextual plausibility and relevance it provides eloundou2023gpts. As a result, the features of the multi-head attention process the model can attend to various subparts of the text based on the short and long dependency and ensure that any topic or any keyword in the input text is not ignored in the generated content. It makes it possible to generate highly organized fascinating textual material for which can be promoted in different areas of the web-site and is offered to utilize to create coherent paragraphs, headlines or FAQs with the input information and the preselected templates.

In addition, the positional encodings used in the model and the iterative attention layer

show that the work is suitable for handling sequential data and generating text that is as detailed and accurate as is possible. Layer normalization makes the model a reliable one and results in very good quality of the content that is generated. All in all these features enable the model provide highly personalized content for websites and even perform some basic tasks which marketers undertake including content creation, customer outreach and generation of specific responses that lift the engagement level of website.

The basic structure of the GPT-4 in figure recommends an architecture of a transformer, which is a large language model that teaches with a lot of text data for the purpose of predicting and generating text data of human likeness. It works through converting an input text into a sequence of numbers and then passing the numbers through multiple layers of attention mechanisms and feed forward networks. In this model, self-attention mechanism is employed to determine the importance of each word in context and thus, up with contextually suitable answers. When training, GPT-4 uses the method where it pitches next word of the next token in a given sequence, and adapts its parameters depending on the delta between the pitched value and the actual value (loss). As for inference, GPT-4 produces one token at a time given the past tokens and learned patterns to produce linear and connected responses. Due to its many parameters, it is suitable to a variety of natural language problems, including summarization, reading comprehension and question answering, as it essentially learns from patterns within the data it is given.

Importance of GPT-4 in wesbiste generation

GPT-4 is then able to change the way websites are built through improving both the aesthetics of the site and its features. It eases the process of content generation,writes competent, search engine friendly text that fits brand tone and purpose. GPT-4 helps in coding, it automatically provides optimized HTML, CSS, and JavaScript for responsive, adjusted layouts and interactive elements. A conversational AI system allows usage in chatbots that enhance the user experience with real-time and natural language processing support emulation. GPT-4 allows the generation of multilingual websites, translating text into different languages for diverse

viewers. Moreover, it uses data regarding the user and their behaviour and provides specific experiences on the website. Due its high ability in automating keyconsuming actions, GPT-4 enables developers to concentrate on the key creative work, and as a result speeds up website creation and management.

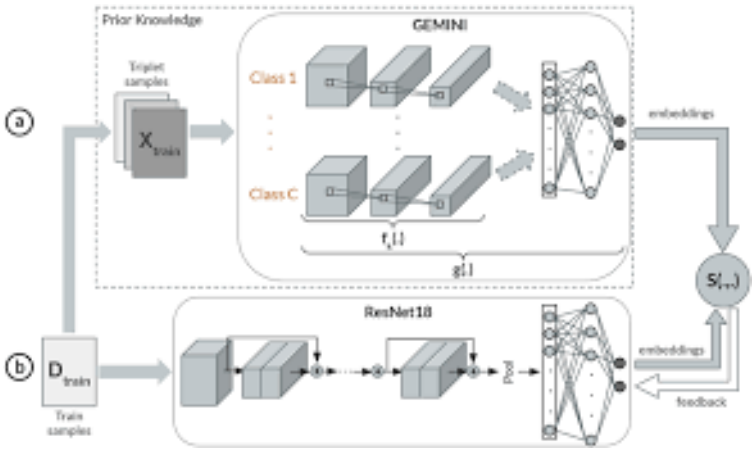
2. Gemini

The Gemini model is an LLM belonging to Google DeepMind that sets the technical goal to further step up natural language understanding and generation. Supervised on general vocabulary, Gemini is trained on a large number of data sets, and with its roots established in transformer design, it is very efficient in various language tasks because it integrates both high computation and extensive training. The current and future work incorporates modern approaches starting from such as reinforcement learning; Gemini is multimodal and can process the text and images Radford, Kim, Hallacy, et al. (2021). This puts Gemini in the position to accomplish tasks that demand explainable context grasping or interpretation like text generation, text summarization, question answering and so on. Given the nature of the model presented, it can be applied to enterprise applications, research, and presentation industries, as well as creative professions that require a detailed understanding of the performed task and the ability to generate texts with a given level of language complexity Ouyang, Wu, Jiang, et al. (2022). Gemini is the distillation of Google DeepMind's bigger initiative to bring AI to diverse domains, services and devices with an aim to optimise for efficiency, scale and in addition to those, moral standards of utilisation.

Model Architecture and algorithm

Figure 46

Model Architecture Diagram of Gemini



It seems to represent an architecture in figure 46 associated with the Gemini model, which is based upon two stages: feature extraction and embedding generation Ramesh, Pavlov, Goh, et al. (2021). In this architecture, the model processes triplet samples are first forwarded to several class-specific blocks, possibly learning different characteristics for each class before passing through several blocks to produce embedding that contains semantic information Dosovitskiy, Beyer, Kolesnikov, et al. (2021). In part (b), another network that has been included is ResNet18, to process general training data for traditional layers, with convolutional to extract feature. These features are then passed and converted further to embeddings so that they are taken to shared embedding space to get feedback and get compared with the prior knowledge embeddings Jia, Yang, Xia, et al. (2021). Together with feature extraction, the feedback facilitates the improvement of the model's performance in identifying categorized data inputs. The architecture of the system also enables it to adopt prior knowledge and generalize on other new data inputs Ramesh et al. (2021).

Gemini model algorithm is an improved version of the transformer that also features multimodality to handle text, image, and other related data type generation. The model first puts forward token representation for input points to be followed by layers involving self-attention

mechanisms used in order to understand contextual relationships. Multi-query attention and reinforcement learning allow the multiple-step prompting to work robustly across diverse data input types, all of which is a part of Gemini Jia et al. (2021). While training, Gemini has to reduce the measure between its predicted value and the real tags (loss) through the backpropagation algorithm. While producing the responses, it deploys its feedback loop mechanism to fine-tune the generated outputs and make them correct, especially in the multiple modality problems. It makes Gemini very convenient for many contexts and highly fluid in terms of applicability as a method for language generation as well as image and data analysis Jia et al. (2021).

Model for Website Generation

Gemini can play a colossal role in boosting website generation since it is based on code generators for HTML, CSS, and JS prompted by user input so that ordinary individuals can develop websites Ramesh et al. (2021). It can design basic structures of web sites, as well as some style elements as it takes prompts such as, 'design a modern business landing page' and it assists in constructing coherent compatible structures as seen above. Besides its multimodal functionality, Gemini can also include image analysis or generation, which can assist with asset generation or modification according to a given website theme. Also, it can deliver optimizable marketing materials, for example, catalog descriptions, articles, and posts, that can be optimized for SEO purposes Ramesh et al. (2021). As both frontend code and content generation tasks fall into the domain of Gemini, web development work is made easier. Most importantly, its context-aware responses make it well suited for use in responsive design generation, which in return results to the generation of websites that can easily be adapted to the different devices Ramesh et al. (2021). Altogether it eliminates time consumption in development process, facilitate easy creation for the non-developers, and innovative prototyping for web venture Jia et al. (2021).

3. Claude

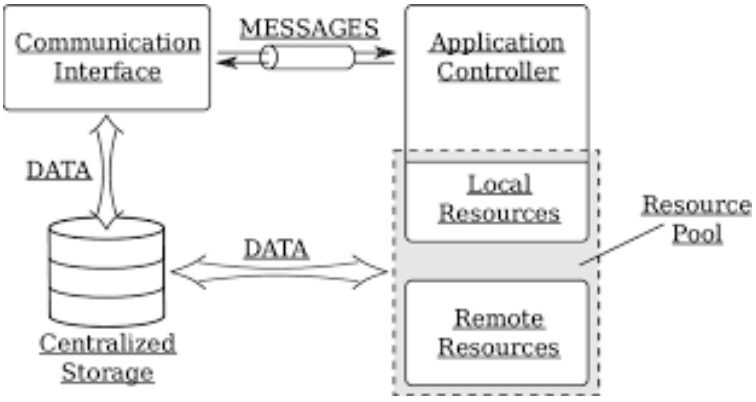
Claude is available on the artificial intelligence platform and was designed as a large language model (LLM) by Anthropic for general use across a range of natural language processing tasks Askell, Bai, Chen, et al. (2021). Claude, named in honor of the father of information theory,

named Claude Shannon, is designed from the ground up for safety, reliability, and interpretability. Unlike most of the standard architectures, Claude learns from Anthropic's safety measures, in an endeavor to build a model that will not generate toxic or prejudiced results. Claude is trained on a variety of data sources and thus can be used for tasks such as text generation, server side data summarization, answering questions and so on all of it with fairly high degree of accuracy Bai, Jones, Ndousse, et al. (2022). It utilizes transformers just like other models, such as GPT, but is trained differently to be more prompt and less likely to be problematic in human- artificial interfaces. As for Claude's work, its aim is to make design as transparent and manageable as possible to allow the user control the application. This makes Claude ideal for deployment in sensitive application domains where issues of ethical use of artificial intelligence inform practice Perez, McKenzie, and Song (2022).

Model Architecture and algorithm

Figure 47

Model Architecture Diagram of Claude



The diagram 47 presents a conceptual model of how resource management and communication will occur in an application system, consisting of a Communication Interface, a Centralized Storage, an Application Controller and Local/Remote Resources. The Communication Interface allows interaction with further parts of the application; here, messages carrying commands go out and answers come in Bowman and Dahl (2022). Centralized Storage is used as a shared data repository smaller database that communicates with the Application

Controller where it may get or send data to/from. It oversee resources and distinguishes between local resources, resources which are attainable at the system level, remote resources, those which are available in the other systems or networks Bowman and Dahl (2022). An item known as a Resource Pool is available to the Application Controller to allow it to allocate resources efficiently based on current needs, and to distribute tasks optimally Bowman and Dahl (2022). This architecture is suitable particularly for applications that need facile resource control, unified data base and the ability to communicate promptly with both internal and external systems Bowman and Dahl (2022).

Anthropic's Claude Algorithm is a transformer architecture AI with a focus on safety and integrity. It lexically analyses input text and turn the data into vectors for context relationship through several layers of attentions Cotterell and Sap (2022). The model employs self-attention mechanism to compute relevance of words to one another so as to generate coherent responses. Claude is trained using reinforcement learning from human feedback (RLHF), full details of an instruction given to Claude is provided below; In training, it reduces cost by checking the actual responses with the outputs that have been estimated in an endeavor to reduce the variance Cotterell and Sap (2022). Generation in inference is done in a sequential manner where at each turn the model uses the previous tokens to construct context. Claude also has provisions for weeding out the undesirable or safe content most of the time which puts responses in an ethical bracket Cotterell and Sap (2022). On that basis, common sense emerges as a solid foundation for both handling multiple tasks at once and achieving high levels of safety and transparency in the language models produced Cotterell and Sap (2022).

Model for Website Generation

The Claude model is rather beneficial for the WEBSITE generation since it can also generate the content, layout and code by itself Perez et al. (2022). It can create website related content including, landing page content, product descriptions and frequently asked questions in a particular tone, for a particular audience. By breaking down language, Claude can create content that will yield better positioning on the search engine results pages. It could also help to develop

codes, inputting HTML, CSS and JavaScript codes with the help of statements taken from the users, and thus enable quick and easy construction of websites. Concrete context advantages include the ability to build device-agnostic designs by creating content that responds to the user environment. also, it can provide content with the help of segmenting customers which can improve the users' experience. In general, Claude saves development time, makes controlling the content easier, and gives a non-developer an opportunity to create a visually appealing and functional site.

4. Claude Haiku

Claude 3.5 Haiku is the fastest and most efficient model from Anthropic's Claude 3.5 family, combining low-latency performance with high-quality output Anthropic (2024). While lightweight, it outperforms larger predecessors such as Claude 3 Opus in multiple benchmarks. Haiku's strength lies in its ability to handle advanced code generation, instruction following, and tool use—making it well-suited for real-time user interfaces, enterprise-level automation, and personalized system agents.

The model achieved a 40.6% score on SWE-bench Verified, surpassing even Claude Sonnet and GPT-4o in its ability to solve real-world software engineering tasks. These improvements stem from its enhanced reasoning ability and its efficiency in multi-turn interactions, system integration, and structured content generation. Haiku is particularly effective when dealing with large datasets such as product catalogs, purchase logs, or enterprise documents Anthropic (2024).

Model Architecture and Algorithm

Figure 48

Model Architecture Diagram of Claude 3.5 Haiku

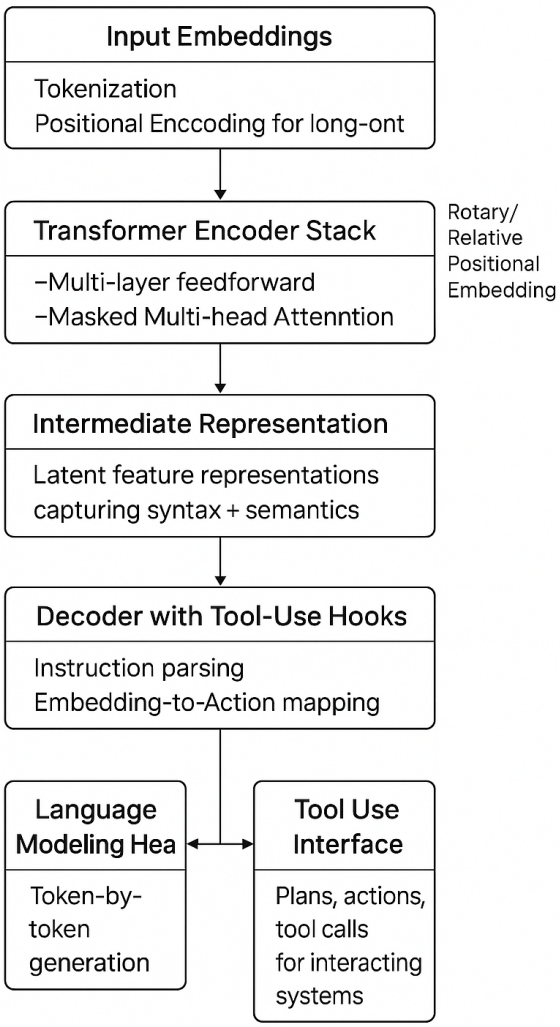


Figure 48 illustrates a conceptual view of the Claude 3.5 Haiku internal architecture. The model is based on a decoder-only transformer framework with architectural refinements for latency reduction and reasoning consistency. It utilizes rotary positional encoding and optimized attention mechanisms to support long-context inference without degrading output quality Anthropic (2024). Reinforcement learning with human feedback (RLHF) continues to be central

to Claude's training, although Anthropic's Constitutional AI approach plays a key role in aligning the model to user expectations, transparency, and safety.

A distinguishing feature of Claude 3.5 Haiku is its early-stage capability to interact with computer interfaces through a specialized API. This allows the model to perform general-purpose actions on user systems such as navigating spreadsheets or completing web forms. On OSWorld, Claude achieved 14.9% in screenshot-only scenarios and 22.0% when provided additional interaction steps—outperforming all competing models. These advances show how Claude is evolving from a conversational agent into a general-purpose digital operator, capable of supporting more autonomous workflows Anthropic (2024).

Model for Website Generation

Websites generated by Claude 3.5 Haiku perform effectively to produce both text content as well as component-based website structures from natural language input. Users can feed the model business objectives and design requirements and target demographics and preferred functionality along with selected themes and it will output complete modern Tailwind CSS-styled HTML documents complemented with dynamic JS components. The output of produced content includes hero sections together with call-to-action blocks and testimonials that maintain high relevance and consistent quality with SEO optimization applied.

Claude 3.5 Haiku delivers prompt responses that make it suitable for time-sensitive operations and workflow development techniques which need immediate feedback.

Innovative Training for B2B SaaS Website Generation

Generating the B2B SaaS website requires some considerations since such websites are generally required to be multi-level, and each subsequent level can be accessed only by providing the proper credentials; in addition, the websites are always searched through the search engine. To address these requirements, we implemented innovative training methodologies for two models: GPT and Gemini. These models were specifically designed to work with domain specific data which consisted of JSON descriptions of B2B SaaS websites consisting of HTML, CSS and JavaScript along with image plates. This training was centered on the supplementation of

information by the four models to generate well-formatted websites; while GPT provided content of the websites, Gemini provided aspects such as layout and responsiveness.

Improved Training Methodology for GPT

This work fine-tuned GPT using a dataset that preserved the hierarchy of B2B SaaS websites. The training was focused on the generation of the content with multiple pages, so the model would be able to work with hierarchal prompts. For instance, during training, GPT was provided with examples of text that must be produced for five unique pages: Homepage, Features, Pricing, About Us, and Contact Us while practicing page cohesion.

As a result of it, a number of improvements were made: to improve GPT's performance, SEO optimization tasks were introduced into the learning process. This involved creating keyword driven topics, keywords description statements, keyword tags for image text along the lines of HTML tags. For example, GPT has learned to generate outputs such as meta descriptions and distinct h1 headers with regards to SEO guidelines. Also, it was possible to combine content with HTML/CSS structures with minimal interference with manual layout changes needed in GPT. These innovations enabled GPT to create . . . suitable professional level mass marketing materials that fit within the current look and feel standards of the marketing industry, and are ready for deployment.

Improved Training Methodology for Gemini

Originally, Gemini was trained to target design and structure of Business to Business Software as a Service websites. Training process employed hierarchical data from the JSON dataset making it easier for Gemini to produce more responsive and visually coherent layouts. To expand it, the model was trained to optimize current frameworks such as CSS Grid and Flex built for the contemporary world so that the generated websites correspond to different devices – desktop, tablet, and portable.

Gemini's training also focused on how one area of a Web site is consistent with another, particularly headers, footers, and body text to make the template look professional to users. It was then enhanced to accommodate templates of static and dynamic objects such as images and

buttons when generating neat, search engine friendly and light codes for improved page loading. It also ensures that the websites being produced by Gemini possess a pleasing appearance but they have also met all the right criteria that lead to the functionality and general usability.

Results and Evaluation

With the use of new approaches in training introduced to GPT and Gemini the outcomes showcased marked enhancement in the quality of sites developed for B2B SaaS businesses. It was also established that GPT was able to generate clear and well linked multi-page text optimized for high ranking on search engines – indeed, directly into HTML; on the other hand, Gemini was capable of producing simple and more complex responsive layouts and structures of codes. Each model contained above covers both the content and structural requirements of B2B SaaS website generation, rendering them useful in the construction of efficient and mostly qualitative B2B SaaS websites.

Using training data of a specific field and employing new strategies, GPT and Gemini were eventually applied to address B2B SaaS website generation requirements. The fine-tuned models clearly outperformed the vanilla models in creating content that was dense in SEO keywords, designs that were responsive to various screen sizes and pages that had efficient and clean code and this was proof of how the application of transfer learning in specific domains, can be useful in improving the quality of particular tasks. These advancements show that language and layout models can be integrated to enhance automated web site generation in the SaaS field, thus increasing the bar. Subsequent studies can expand the range of the presented methods to include other fields, including e-commerce or educational applications, as a step towards the development of automated web services.

4.2. Model Supports

Deploying and sustaining solid infrastructure for ML and data analytics is central to successful model training; real-time model deployment; and data handling. The identification of the correct architecture is important not only from the viewpoint of the computational demands necessary to implement new algorithms but also from the viewpoints of scalability, flexibility and

robustness.

Platform, Environment and Tools

These are high-end graphical processing units – GPUs and multi-core central processing units – CPUs, advanced storage technologies like solid state devices – SSDs, and sophisticated software hypotheses such as TensorFlow, PyTorch and JAX. By incorporating highly computational assets into a well-designed initiative, organizations enhance the speed of insights, enhance the precision of models in large applications, and advance several analytics options. Security, data, and cost measures are also part of infrastructure management to prevent the system from being vulnerable to risks as well as to create a long-term structure that continuous not only to contain profit but also to be robust in delivering its function The Table 10 describes various model and configuration requirments to build an AI powered wesbite generator.

Table 10

Models and Configuration Requirements

Model	RAM (GB)	Storage (GB)	Configuration Requirements
GPT-4	32–64	200–500	High-performance GPU (NVIDIA A100, V100, RTX 3090), Python
Gemini	32–64	200–400	Optimized GPU (NVIDIA A100 or better), SSD storage, Python
Claude-2	16–64	100–300	High-end GPU (NVIDIA RTX 3090, A100), Python
Claude Haiku	16-64	100-300	High-end GPU (NVIDIA RTX 3090, A100), Python

Original table by Team 2

Table 11

Libraries, Modules, Methods, and Their Purposes

Library	Module	Method	Purpose
Pandas	DataFrame	read_csv	Data loading and manipulation
Hugging Face Transformers	transformers	from_pretrained	Loading pre-trained language models
Matplotlib	pyplot	plot, show	Data visualization
NumPy	numpy	array, expand_dims	Numerical operations, array manipulation
scikit-learn	model_selection	train_test_split	Data splitting for training and testing
TensorFlow	keras	to_categorical	Converting labels to categorical format
PyTorch	torch	DataLoader	Efficient data loading for model training
tqdm	tqdm	tqdm	Display progress bars
json	json	load, dump	JSON and metadata handling
os	os	path, environ	File system interactions

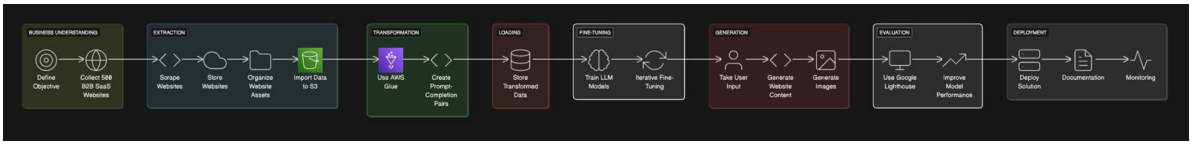
Original table by Team 2

The table 11 shows the Libraries , Methods and their Purposes which are important in leveraging model training, data processing, and evaluation through a wide raft of libraries. Each library was chosen to do something specific in order for a wide range of tasks—from tokenization to data manipulation to deep learning—things that could be done seamlessly. Pandas and NumPy drive the data manipulation, but scikit-learn also provides critical tools for data preprocessing and model evaluation. The Transformers library by Hugging Face helps in loading and fine-tuning the LLMs used, while Matplotlib provides visualization of data and model performance. Core deep learning frameworks used for training include TensorFlow and PyTorch; the primary framework, however, is PyTorch since it offers more flexibility when working with transformer models.

Model Architecture and Dataflow

Figure 49

Architecture for AI powered website generation



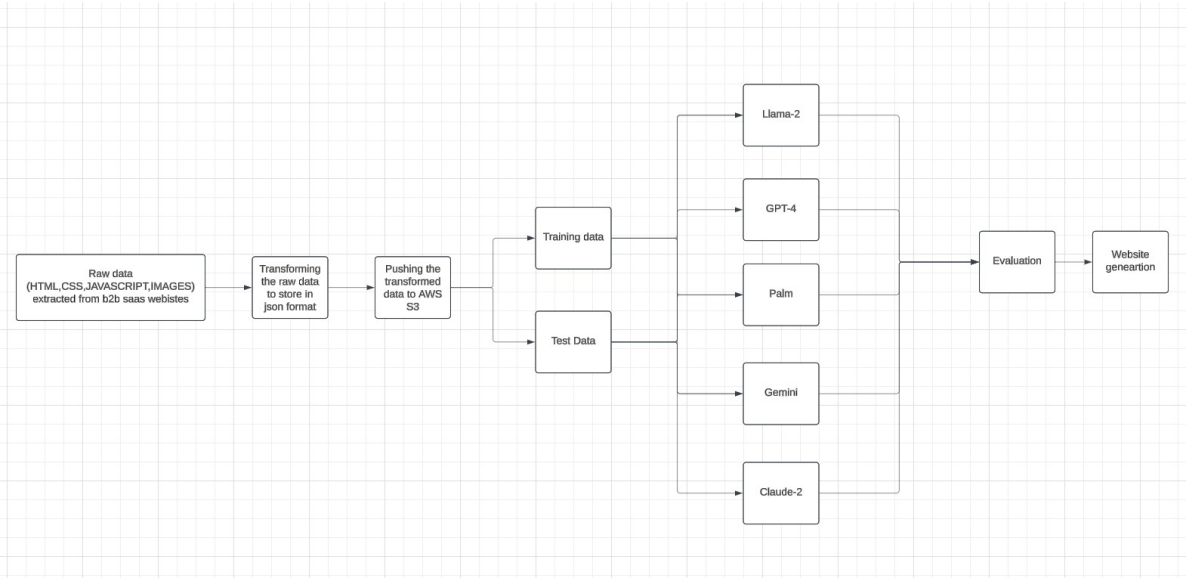
Original image by Team 2

The pipeline shown in 49 starts with data extraction process – web scraping takes place on a Google Cloud VM where the HTML, CSS and JavaScript files of the target websites are collected. The raw data is temporarily stored in Google Cloud Storage after which the data is preprocessed. Data extraction comes in next where the raw content goes through an extraction process and the result is textual content as well as images and links. Google Colab preprocesses the parsed data and reforms the parsed data by converting it into a JSON file format and stored it into Google Drive. Once transformed, they will transfer it to AWS S3 in which AWS Glue Crawler will stage the files and format it for model training.Once the data is well archived then the machine learning model uses the data in S3 for training.

The figure 50 illustrates an improved data processing and model evaluation procedure geared towards the creation of website generation making use of extracted data from B2B SaaS websites. It starts with the buttons of HTML, CSS, JavaScript, and images collected from these origin sources. This data is converted and formatted for JSON to facilitate its organization and readability while in storage. The transformed data is then storing in AWS S3 to cater for cloud scalability. To further analyse the pipeline divisions the data into training and test sets in which several large language models such as Llama-2, GPT-4, PaLM and Gemini are used for training as well as testing. Each of the models takes the data and the results are assessed to decide on the capability of the models in producing websites. The final procedure is utilized based on the outcome of the evaluation to generate the website content from the best performing model while maintaining high quality and relevance.

Figure 50

ML data flow for AI powered each website generation



Original image by Team 2

4.3 Model Comparison and Justification

Compare models

Table 12 and 13 shows the comparison of all models.

Justifications for Each Model

GPT-4: As it stands, GPT-4 is essential for creating optimized, and adaptable content that conforms to search engine results page expectations. It can produce relevant and interesting text to lure search engine traffic and improves user experience for given or requested query. Its language generation features make it suitable for the creation of different types of content on the website . . . The website will therefore always have useful and findable content. Indeed, it is a powerful text generator but, at the same time, it does not work well with creating structural layout or complex interactive layout-related features, which makes it a valuable tool to combine with other models.

Gemini is engaged in creating such live and dynamics interactive GUI and structurally related interfaces with the help of generating elements such as button, navigation panel and other such structural forms. That the interface can be altered for the set devices the system looks very

Table 12

Model Comparison (Part 1)

Model	Problems	Features	Approach	Strengths	Limitations
GPT-4	Create SEO-oriented dynamic and engaging content	Produces high-quality SEO texts with perfect adaptation to input	Combines advanced language generation techniques with optimization for SEO and users	Delivers engaging, SEO-optimized content adaptable to user needs	Limited in handling intricate interactive elements or structured layout formatting
Gemini	Building interactive UI elements like buttons and navigation panels, and device interfaces	Provides UI components like buttons, navigation panels, and device-adaptive	Utilizes generative design and adaptive layout techniques focused on user interfaces	Visually appealing and adaptable across devices	Limited in personalization depth without detailed input; primarily focused on visuals/interface
Claude-2	Generating custom code for B2B SaaS websites	Assists with HTML, CSS, JavaScript snippets, offering basic layout and functionality suggestions, which is also seo optimized	Provides code and layout suggestions tailored to specific SaaS needs	Simplifies development by producing code that fits SaaS-specific site functions	Limited in handling complex application logic; focuses on frontend code and basic layout suggestion and vidoe editing still need some work

Original table by Team 2

responsible and mutually visually identical while used on a desktop or in a mobile and tablet versions. With the design and flexibility of visual elements, Gemini escalates communication and usability. However, the primary focus of Gemini is hypertext, and it might need other models for user centered or content based presence.

Table 13

Model Comparison (Part 2)

Model	Problems	Features	Approach	Strengths	Limitations
Claude-3.5 Haiku	operates in real-time to create content and code which it then implements for sites through fast response times.	produces HTML and CSS in addition to JS and SEO-friendly content at high speeds,can connect with various APIs and tools.	combines fast transformer architecture together with instruction-following features and tool-use functionalities.	Has much higher speed without compromising intelligence. better at code generation.	weaker performance if complex logical sequences is given.

Original table by Team 2

Claude-2 helps to develop B2B SaaS websites by generating the code snippets according to the specific request and providing layout suggestions. For example, it helps to generate an HTML, CSS, and JavaScript code that helps the developers to optimize implementation of site elements since they don't have to begin from square one. It is especially helpful in B2B scenarios as different websites need to include highly functional and sophisticated design that corresponds to SaaS services. Claude-2 also supplement basic layout and functionality templates for aesthetic and practical coherence and compliance. However, this feature will make the process of development quicker because the recommendation will help to implement common items in a shorter amount of time. Nevertheless, Claude-2 is well-suited for generating frontend code or easy layout suggestions, but, perhaps, not especially efficient where complex application logic is expected because Claude-2 deals with the code of UI components and layout rather than specific backend features. Therefore, the points raised by Claude-2 are highly informative for frontend development in B2B SaaS websites specifically for the website that is developed for teams with the need for effectiveness, responsivity, and flexibility of the website features that were mentioned above.

4.4 Model Evaluation Method

Content Quality Assessment (for all models) .According to the paper ono some models like, GPT-4, Llama-2 and others that create content, coherence, accuracy and organization of the text is critical. Here, BLEU, ROUGE, and METEOR scores are highly relevant. BLEU Score measures exactness of the output by comparing it with references texts, particularly useful for the assessment of the flow and organization of concepts in SEO-optimized and calculated content.ROUGE Score covers recall and is useful when the organizing text needs to include all essential information, for example, in descriptions of the products.METEOR Score stress on semantic similarity is much appropriate and beneficial in determining the overall comprehension and fluency of the content especially in conversational or end user interfaces.

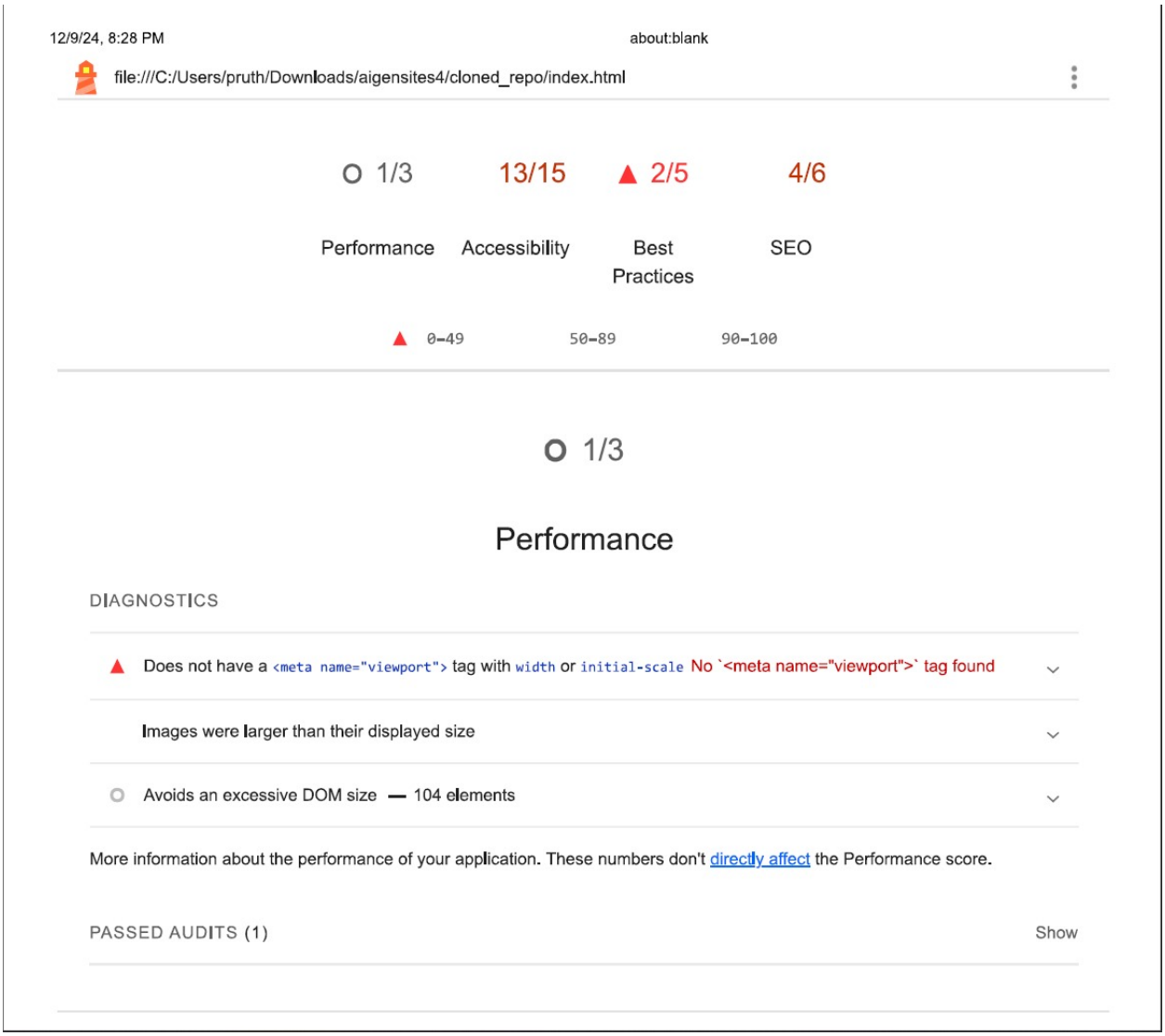
Lighthouse for Evaluating Websites Generated by GPT-4: Lighthouse is a useful tool I use to assess web sites created using GPT-4, helps to identify real-world performance, accessibility issues, search engine optimization, and other aspects. As for quality, Lighthouse quantifies attributes such as FCP, LCP, TBT, and CLS, to guarantee that the created websites optimise loading time, prioritise the loading of relevant content, and offer a stable layout. Usability is evaluated by the semantic HTML, use of alt attributes, contrast, and keyboard accessibility to make the website more convenient and customer friendly. When it comes to SEO, Lighthouse tests meta tags, mobile, structured data, and textual descriptions of links to generate websites that are SEO friendly.

Lighthouse Report of Generated Website: The Google Lighthouse report underlines the main aspects of the website. The Performance was (1/3). the evaluation pointed that the tag "meta name viewport was missing and the image size can be more optimal. The Accessibility criteria is given (13/15). it states that accessible names of buttons and higher contrast ratios are required. For the Best Practices we got (2/5). It states again that some pictures have incorrect aspect ratios, several pages lack a viewport tag, and an HTML doctype is not specified. The SEO score was (4/6). it states that there are no meta descriptions or valid. robots The team has identified that resolving these issues will lead to a much more accessible and technically correct website. Figures

51, 52 are the screenshots from report.

Figure 51

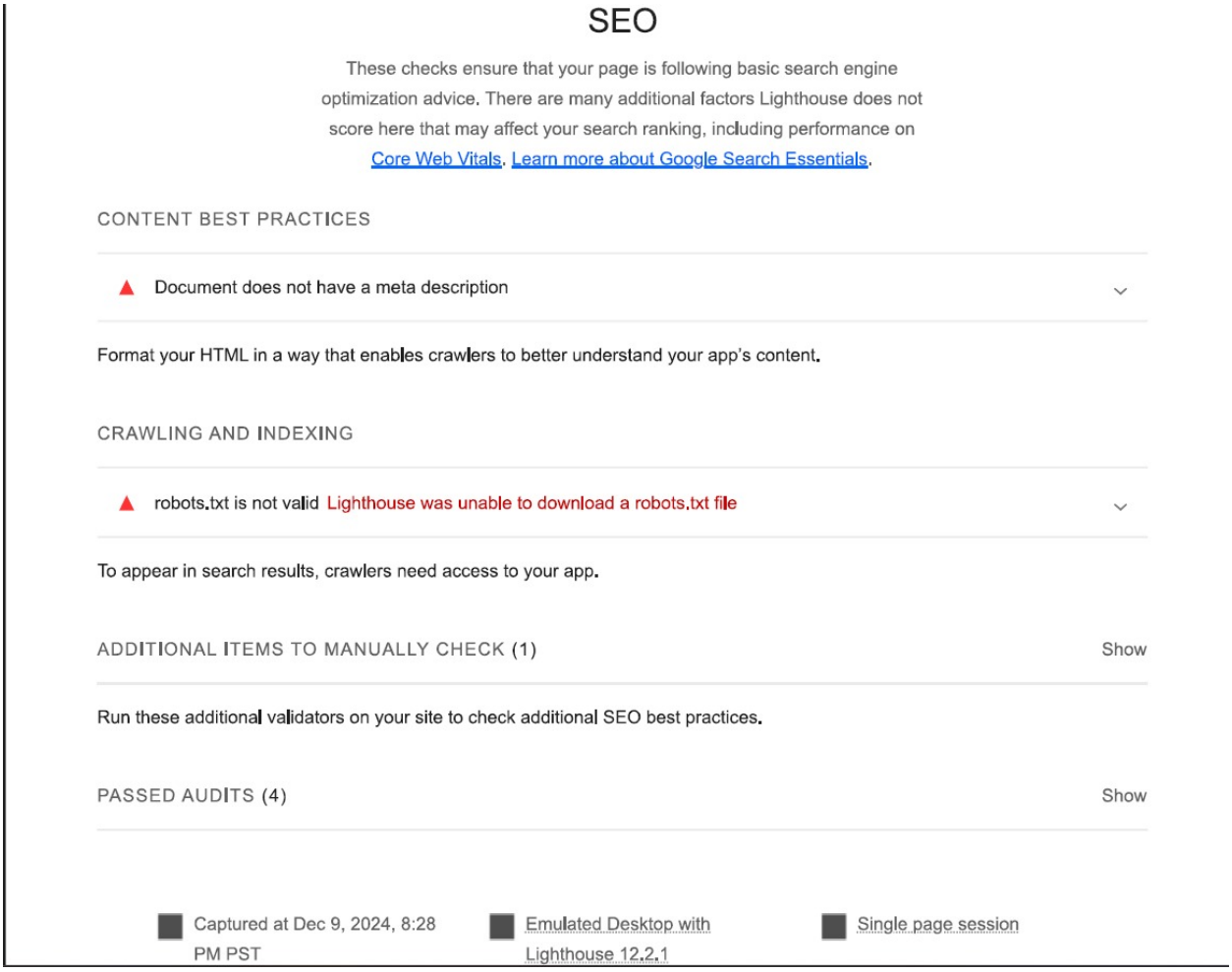
Lighthouse Report 1



Original image by Team 2

Figure 52

Lighthouse Report 2



Original image by Team 2

User Interaction Testing (for Gemini). Assessing Gemini's part, as means to generate UI components, requires, apart from specifications of usability metrics, the Google Lighthouse's Performance and Accessibility scores.Lighthouse Performance measure the time taken for the different components of UI to load and perform, a function essential to aspects like buttons and menus.In Lighthouse, the features of accessibility help to guarantee that UI components satisfy the general criteria for usage by individuals of different backgrounds to achieve the best satisfaction level among users.Validation measurements of A/B testing, time spent spent by the user on the specific GUI and the rating of the GUI by the client provide further information on

how effectively the generated GUI function in practical application.

Code Quality and Functional Testing (for Claude-2). This preference of Claude-2 for custom code generation is evident through theoretical quantitative measurements such as functional testing and actual measurements from Lighthouse Best Practices and performance. Some checks in Lighthouse best practices include checking that the code it is conforming to web standards that is so important in terms of security and cross-browser compatibility. Lighthouse Performance provides information about the loading time and rendering time of the generated code mainly covers for HTML, CSS, and JS snippets. Static analysis can also be used to ensure that correct syntax and maintainability issues have been addressed as well; sandbox testing can then be used to ensure that the generated code performs as it is expected to when operating in the real environment.

Cross-Model Comparative Performance. While all these model-specific metrics are very useful, cross-model comparison will be even more valuable in showing how each model best complements others in a unified workflow. BLEU, ROUGE, and METEOR scores can provide benchmarks of content quality across models; similarly, Google Lighthouse's Performance, SEO, Accessibility, and Best Practices metrics can be used to evaluate how well the integrated outputs perform as a whole.

Human evaluation and feedback loops. Likert scale user experience ratings, qualitative feedback, and domain expert assessments give a context that can't be understood from automated scores alone. This allows for iterative fine-tuning based on real-world preferences and needs.

5. System Design

5.1 System Requirements Analysis

System Boundary, Actors and Use Cases

The AI-powered website generator contains a well-designed boundary that holds all essential automated website generation abilities. All stages of website management operate within the system including the collection of user inputs and content production and website publication and element optimization. Through its frontend interface the system provides a primary way for users such as business owners and freelancers and marketers to send necessary website information including structure details as well as content specifics and industry preference settings. API requests transmit input data to the backend processing module where it verifies information before consulting templates and stored metadata and creates a language model (LLM) request.

The system's central processing element depends on advanced LLM models including GPT-4, LLaMA 2, PaLM and Mistral as well as Gemini for generating structured code components for HTML, CSS, JavaScript alongside user-defined content. All the models have distinct functions in website development since GPT-4 handles content production and Mistral strengthens structure and LLaMA 2 provides lightweight text organization capabilities and PaLM together with Gemini optimize user experience components. The JSON formated output from the programming process is available for subsequent automated website deployment. The deployment module implements a combination of frontend along with backend components through a straightforward integration method with particular checks on system stability as well as functionality. Any encounter of problems associated with content generation and deployment or search engine optimization is instantly recorded by this module for immediate resolution.

The system effectively operates internal procedures but simultaneously interacts with holding providers as well as search engine platforms (Google and Bing) together with third-party APIs for incorporation.

Actors.

AI website generator consists of numerous actors who interact with the system in different stages of website generation, deployment, and optimization. Actors are divided into primary actors, who are directly involved with the system, and secondary actors, who offer support in system operations and maintenance.

Primary Actors (Direct Users)

End Users (Business Owners, Marketers, Freelancers, Startups): The users who request website structures along with industry type and content requirements are end users. End users perform final checks on SEO optimization together with performance optimization requests for modifications before website deployment occurs.

Three Major Language Models referred to as GPT-4, LLaMA 2, PaLM and also Gemini, Mistral. With these models end users obtain output that produces HTML together with CSS and JavaScript and structured content formats from their inputs.

GPT-4: Content generation Mistral: Structural coherence LLaMA 2: Text structuring PaLM and Gemini: UI and user experience refinement SEO Optimization Engine The SEO-friendly optimization system checks website content for keyword effectiveness and performance monitoring aids search rankings.

Deployment System: Web pages automatically deploy through this system which ensures sites have integrated stability for verification before being published.

Secondary Actors (Supporting Components and Systems)

Database System: The database system stores website metadata, vector embeddings, and user preferences, enabling efficient retrieval of templates, SEO data, and configurations. This helps generate optimized website content based on past successful implementations.

Performance Monitoring and Error Logging System Monitors system stability, API failures, and website performance. The error logging module records issues and notifies the admin for quick resolution, ensuring continuous system improvement.

Admin (System Operator or Developer) Oversees system operations, troubleshooting,

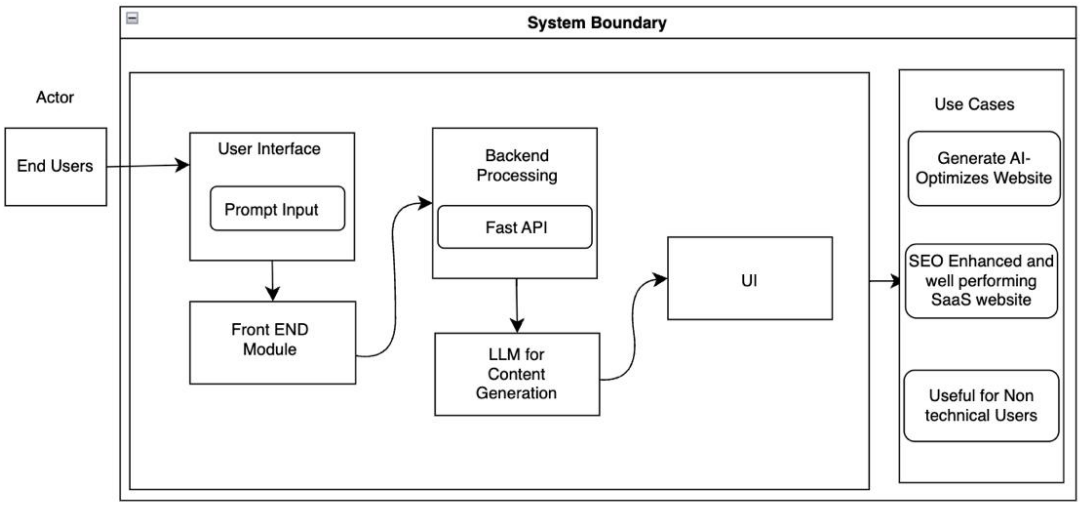
and performance monitoring. Admins fix API failures, deployment issues, and model inconsistencies, while fine-tuning LLMs and scaling the system as needed.

External Hosting and Cloud Services Relies on AWS, Google Cloud, and Azure for secure, scalable website deployment. These services ensure high availability and performance without manual management Yenduri et al. (2023).

Search Engines (Google, Bing, etc.) Search engines rank and display SEO on the basis of performance metrics. Their algorithms determine content generation strategies to optimize search performance Anunphop and Chongstitvatana (2022).The below figure 53 shows the visual representation.

Figure 53

System Boundary, Actors, and Use Cases



Original image by Team 2

Data Analytics and Machine Learning Capabilities

The platform generates multi-page sites for some business verticals using highly customized LLM models as an automatic process. The platform translates user-input points of data into dynamic formats which include structured content and layouts with designs and search

engine-optimized pagesHan, Liu, and Wang (2024). Website quality along with responsiveness and industry standard compliance can be managed by the AI models even when development happens without human intervention.

Content and Code Generation with LLMs: The system produces full website infrastructure through GPT-4 and LLaMA 2 and PaLM and Gemini which also generates HTML CSS and JavaScript code together with specialized industrial content. User instructions given to the models enable their ability to gather necessary themes followed by brand voice characteristics and design specifications. Such a method removes the requirement of complete front-end development to produce websites with optimal structure that combines visual appeal and functional capability.

Metadata Storage and Vector Embeddings: The application uses a vector database to maintain website metadata and embedding representations of created templates. The system retrieves website structures and design elements speedily which leads to shorter response times and maintains uniformity in website creation process. Vector embeddings enable the system to identify patterns that allow it to improve content production through analysis of market motifs alongside user behavior insights.Han et al. (2024)

Natural Language Processing (NLP) for User Input Interpretation: The system implements NLP functionality which enables it to understand instructions written either in structured format or natural language. The system analyzes plain text descriptions from users to identify essential requirements which include their business sector along with customer groups and website style choices together with functionality specifications. The system design through NLP allows users to conveniently interact with the system without requiring technical knowledge.

SEO Optimization and Performance Tracking: An SEO optimization module has been integrated in the system to check website structure alongside keywords and content relevance which ensures strong search engine rankings. The system maintains SEO performance records to help users enhance their website content. The system monitors user behavior and bounce statistics along with search engine placement to help developers prepare better site versions in the future.

Automated Website Deployment The system deploys websites automatically along with handling possible errors. After generating the website the system deploys it automatically through integrated frontend and backend deployment mechanisms. The system ensures successful deployment while monitoring system performance and recording any encountered errors. API failures and performance problems along with SEO inconsistencies trigger alerts to users about necessary repairs through the error-handling module.

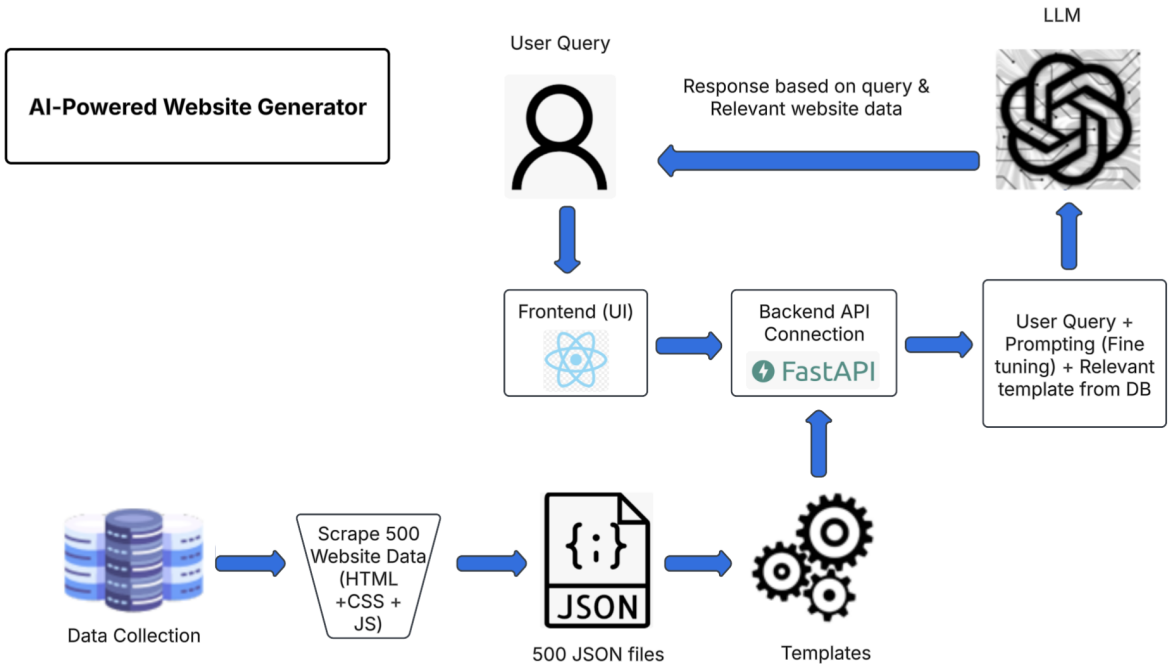
Industry-Specific Customization and Adaptation: The AI generator receives training through industry standards to produce websites which follow best professional practices in their selected field. The system adapts between corporate websites and SaaS landing pages as well as marketing portfolios while respecting specifications that apply to different industries.

5.2 System Design

System Architecture and Infrastructure

Figure 54

System Architecture



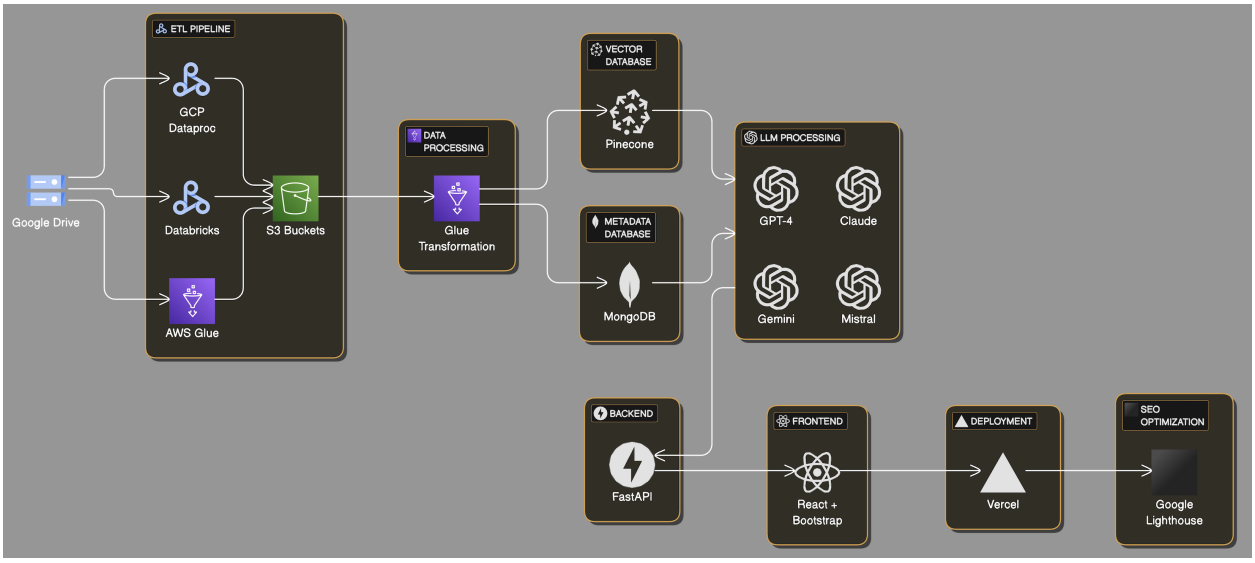
Original image by Team 2

A modular AI website generator develops websites by interconnecting different program elements to execute website development alongside optimization alongside deployment automation. Users start their process through the frontend module by entering their website requirements that move to backend validation for processing. The system retrieves metadata from its database through backend operations to build organized API requests that the LLM processing module uses. GPT-4 along with LLaMA 2, PaLM and Gemini and Mistral functions within the system to produce HTML and CSS, JavaScript and content in structured formats that return deployment-ready JSON data. The website deployment module provides smooth hosting solutions by uniting frontend and backend capabilities and the SEO optimization engine confirms content arrangement together with keyword integration against search engine rankings. The system includes performance tracking modules along with error management tools which both monitor system stability and create logs of issues which alert administrators about failures. The computing platform runs entirely in the cloud through AWS or Google Cloud or Azure for managed hosting together with NoSQL databases for data storage while LLM processing requires GPU-accelerated inference engines. Users can produce professional websites which target specific industries through this AI-driven and automated and scaleable infrastructure that also delivers SEO optimization and performance monitoring as well as long-term system durability. The above figure 54 shows the end to end architecture.

Supporting Platforms, Frameworks and a Cloud Environment

Figure 55

Supporting Platforms



Original image by Team 2

A scalable cloud-native framework integrating supporting platforms frameworks and environments powers data handling deployment and processing activities in the ETL pipeline and website generation process. The system starts by storing raw data in Google Drive until data processing occurs through an AWS Glue ETL pipeline. Highly available and durable data storage takes place after data transformation occurs in S3 Buckets. The refined data from Glue Transformation proceeds to Pinecone (vector database) and MongoDB (structured database) where it enables efficient management of website content and embeddings together with user data. The LLM processing layer unites GPT-4 and Claude and Mistral and Gemini AI models to produce structured website contents along with components. Through FastAPI the backend functions as the core processing system that allows the models to exchange information between data processing units and the frontend application built with React and Bootstrap. Vercel takes charge of deployment procedures to ensure efficient hosting and scalability alongside Google Lighthouse that optimizes SEO performance for better website visibility. The cloud infrastructure

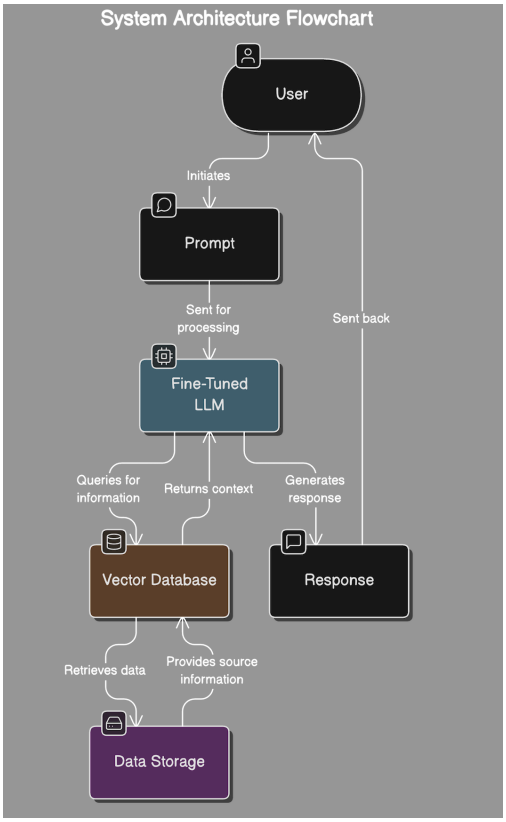
combined with modular elements provides scalability alongside reliability through performance optimization features based on AWS and Google Cloud technologies and API-driven microservices approaches to support an efficient website development framework. The above figure 55 shows the end to end architecture with supporting tools.

Data Management Solution and Database.

The AI-powered system utilizes a data management solution shown in figure 56 which enhances data storage alongside retrieval and contextual information processing operations. A vector database underpins the system infrastructure which provides vector storage functions for fast embedding searches and retrieval of necessary contextual data relevant to the fine-tuned LLM Google Cloud (2023). The vector database receives submission requests which lead to the retrieval of appropriate contextual data from the data storage layer where both structured and unstructured information is located. Users can retrieve information from extensive datasets efficiently by implementing the data storage system across scalable cloud-based databases including AWS S3 and Google Cloud Storage as well as NoSQL database MongoDB. Different vector database solutions like Pinecone, FAISS and Weaviate support optimized high-dimensional embedding storage and retrievals. The structured design enables LLM systems to deliver precise context-based responses and supports infrastructure scalability along with data consistency in addition to cloud-retrieval performance optimizationGoogle Cloud (2023).

Figure 56

Data Management Solution



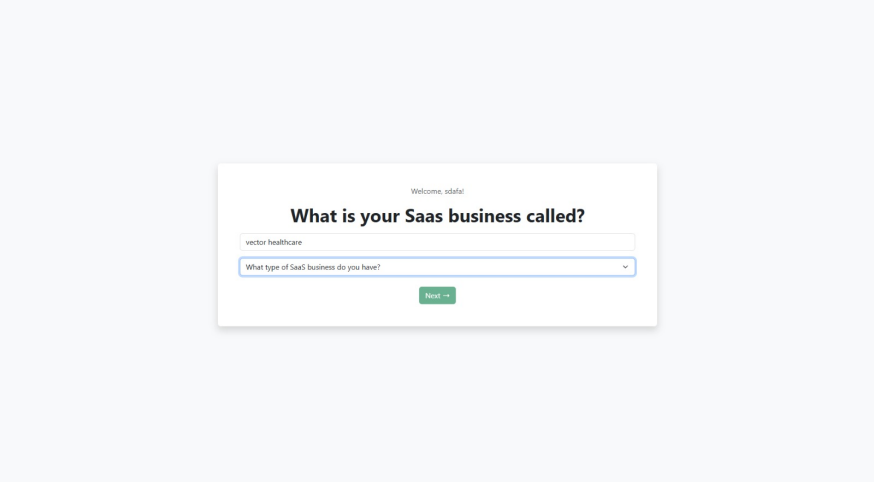
Original image by Team 2

User Interface and Data Visualization Design

The below figure 57, 58, 59 shows examples of UI of the project. A sequential input process for user needs can be found in the user interface of the AI-powered website developer. The user interface contains multiple interactive fields to input data which includes text boxes along with dropdown panels and multi-selection options and color selection options and file upload capabilities for gathering essential business requirements. The system uses a structured step-by-step form to obtain complete data that the LLM processing module requires for its website customization process.

Figure 57

User Interface- Example 1

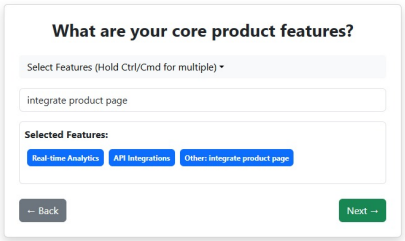


Original image by Team 2

The interface incorporates a contemporary layout design which advances form components in an easy and logical sequence. Users can choose between dropdown selections or select features through multiple checkboxes and enter text into fields named "Other" according to the question structure. Users can customize their brand identity through the integration of color selection tools and file uploading fields which produce websites that match their personal identity Roth (2017).

Figure 58

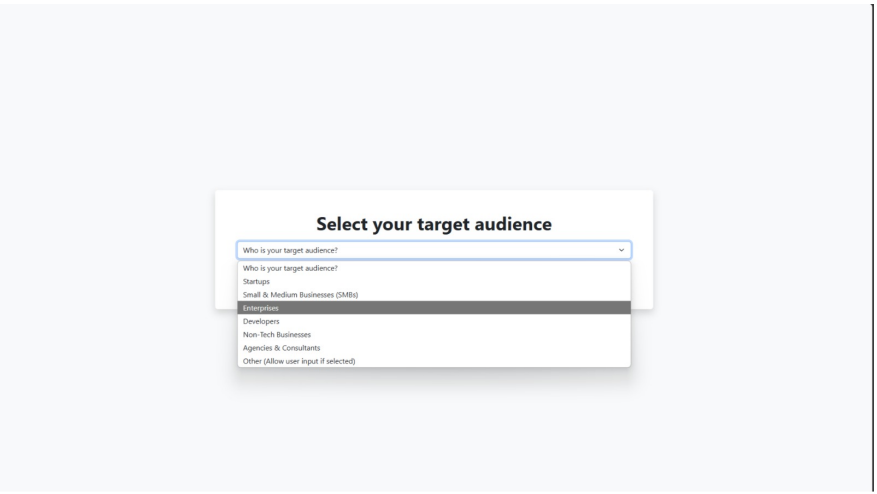
User Interface- Example 2



Original image by Team 2

Figure 59

User Interface- Example 3



Original image by Team 2

As users make choices regarding themes and colors and page design the system provides immediate preview that shows their selected preferences simultaneously. A graphical format displaying user choice information from audiences to features emerges from interactive dashboards built with React and D3.js frameworks which help users improve their selections prior to finalizing their submissions. Structuring the user interface enables a smooth experience for obtaining precise data which enables LLM-based website development with optimized user engagement and customization performance.

5.3 Intelligent Solution

Machine Learning Solutions

AI technology through the website generator operates as a tool that creates operational multi-page websites for businesses irrespective of their technical proficiency. A fully automated system performs the process by linking large language models (LLMs) to backend processing alongside frontend rendering elements while also utilizing cloud-based application deployment. The generator applies these features together to interpret user instructions while building structured content and designing website structure and search engine optimization. Website creation needs high-quality content that specifically matches industry requirements as one of its

most fundamental elements. The system employs GPT-4 as well as Claude and Gemini and Mistral along with other advanced language models for its content generation process. The supplied user information which includes business name together with industry and services undergoes analysis by these AI models to produce formal website content that consists of headings, paragraphs, descriptions along with call-to-action statements. The content optimization process makes it readable and relevant with engaging content that maintains professional language variations suitable for multiple business industries. Real-world business websites serve as the foundation for dataset training which allows the AI models to produce accurate content. The system employs rule-based UI structuring in its website organization and design instead of making use of deep learning-based layout generation approaches. The backend development incorporates FastAPI to perform core functions that control user requests and AI model operations. The program implements React along with Bootstrap to construct its frontend therefore delivering professionally designed content which adapts smoothly to contemporary user-friendly interfaces. Through this design method the websites achieve high visual quality together with proper organizational structure alongside user-friendly navigation which adheres to industry standards. The system deploys pre-developed UI components which have been purpose-built to fulfill diverse business requirements instead of using AI-automatic layout creation. Control of website structure remains completely in hand through this approach and website customization becomes simpler.

Automated website deployment occurs through Vercel which functions as a cloud-based system that frees users from managing website hosting and version control. The system automatically publishes sites immediately upon generation so users can access them right away. The system includes Google Lighthouse functionality to assess website performance including speed and accessibility and helpful overall performance evaluation. Google Lighthouse allows the system to monitor website structure improvements which help meet current Internet standards and search engine requirements. The user engagement metrics in here including page load times and bounce rates become essential data points for the system to improve upcoming website developments. The AI-powered website generator utilizes FastAPI for backend processing

together with React and Bootstrap for frontend rendering and integrates Vercel for automated deployment to give businesses an efficient user-friendly scalable solution for their professional website needs. The strructured approach provides better website consistency than AI-based layout generation since it delivers customizable platforms that function optimally in real-world settings.

Project Input Datasets, Expected Outputs, Supporting System

The user interface and data visualization for the AI-powered website generator have been tailored to facilitate an intuitive yet structured data-gathering process to provide seamless user interaction and efficient customization. User input is captured through an interactive form containing text fields, dropdown menus, multi-select options, file uploads, and color pickers. This input includes business name, SaaS category, core product features, target audience, branding preferences, required website pages, call-to-action elements, content type, and lead generation forms, thus finally making sure of a comprehensive customization. To further stimulate the contextual understanding of LLM, the system has been integrated with a pre-existing dataset of 500 public B2B SaaS websites from diverse domains and industries. This dataset provides useful points of reference enabling LLM to generate such industry-specific, SEO-optimized websites and conversion-driven website content that matches their business needs. By utilizing the structured input data in conjunction with industry-specific website references, the system acts to ensure generated output that conforms to best practices in design, user experience, and content optimization. Data visualization techniques that enhance user engagement include real-time preview rendering, interactive UI elements, and graphical elevations of selections, which create a fluid website generation experience by merging human input together with AI-driven automation.

Expected Outputs

A workable AI-based multi-page website will be produced and scripted to match the user's needs for a SaaS business. Based on user-provided preferences, the application will compute the generation of HTML, CSS, and JS code, given that it would ensure a well-structured, visually appealing, industry-centric design. In addition, the system will output search-engine-optimized content for the site, including homepage texts, product descriptions, and calls to action

conforming to best practices in search engine operations.

Above and beyond simply generating code, the project will also involve options for customizing lots of features such as themes, layout options, colors, and branding elements. The users will be able to refine their web pages before deployment. For cost efficiency and the running of the project, websites will all be setup-ready hosted out on Vercel. In addition, there will be an SEO and performance report via Google Lighthouse applied so the teamwork can establish the website's speed, responsiveness, and search ranking ability.

The AI will also produce the interactive UI where the user inputs their preferences by clicking through dropdown menus, uploading files, and making multi-selects, thereby enhancing the customization experience. Besides that, users will be able to see changes to their sites in real-time. Cost-effective, time-efficient, accessible, and scalable website creation: no coding required, yet businesses will be empowered to create a great online identity easily.

Supporting System Contexts

It focuses on automated deployment and real-time integration so that the generated websites are deployable in real time and live for end users to access. Once the AI-generated content and structure are ready, the Deployment API kicks in to automate website hosting using Vercel so that there is a seamless process from content creation to live deployment. The software is also integrated with React and Bootstrap to provide an interactive frontend experience and be cross-device compatible with optimized performance. Additionally, real-time performance and SEO analysis is carried out through Google Lighthouse, which audits key metrics such as page speed, accessibility, and search engine optimization. The deployment process is also completely automated so that companies can deploy websites without manual configuration or technical expertise.

Solution APIs

The software is, fundamentally, made up of a collection of RESTful and cloud-based APIs that provide seamless data transfer, content generation, and website deployment to drive end-to-end automation. The User Input API collects user preferences for personalization and

ensures structured data flow for content generation. The LLM Content Generation API connects with GPT-4, Claude, Gemini, and Mistral to produce structured website content and UI features according to user demands. The Vector Database API queries Pinecone to provide relevant embeddings and domain information in the process of improving contextual correctness. The Template and Component API fetches built UI parts from the template repository for effective creating and structuring a dynamic website. After finalization of content and design, the Deployment API automatically does the hosting in Vercel and gets the website online. Further, the SEO Optimization API will connect to Google Lighthouse to analyze and make further optimizations to the website in terms of performance and therefore boost the ranking on Search Engines. Through the integration of these APIs workflow automation sets in to allow the users to create, tweak and deploy a bespoke AI-driven website on their own easily.

5.4. System Support

AI model training along with efficient experimentation occurs through Google Colab and an RTX 4090 GPU which aids the development of the AI website generator. Through these resources users gain fast computing capabilities for executing deep learning model training jobs and model fine-tuning operations. Twice-tested models are safely maintained through AWS S3 and Google Cloud Storage for ensuring the long-term storage of B2B SaaS website data across both cloud platforms.The system employs cloud-based deployment which provides users with easy scalability and access to all features. VS Code together with Jupyter Notebook function as the main environments for both coding works and AI model debugging and experimental activities. Website deployment through Hugging Face allows users to host and infer AI models along with API-based integrations that make the system efficient for serving website creation functions powered by AI.

Hugging face. The selected AI models Mistral, GPT-4, Gemini, and Claude at Hugging Face will serve both as fine-tuning platforms and deployment hosts for generating dynamic web content Bai et al. (2022) and SEO metadata through the AI-powered B2B SaaS website generator. This program will work with Hugging Face's Model Hub for real-time inference after its model

fine-tuning process under PyTorch or TensorFlow. FastAPI forms the backend of the system which uses Hugging Face Inference API to produce customized website parts and metadata structures and AI content. Fast response times and automated generation processes alongside scalability define the operations of the B2B website generator that utilizes AI capabilities.

Programming languages. The development process and deployment of LLMs such as Mistral GPT-4 Gemini and Claude will happen through Python while enabling AI content generation. FastAPI, a backend framework, builds high-speed API connections that link AI models to frontend interfaces and databases. A setup creating an enjoyable experience with efficiency, scalability, and speed will give excellent performance and smooth integration into the B2B SaaS site generator.

Platforms. The three notebook applications used for AI model development through the AI-powered website generator include VS Code alongside Jupyter and Kaggle. Jupyter and Kaggle enable exploratory model testing and training while VS Code focuses on developing backend features and APIs and debugging procedures to maintain smooth developer operations.

LucidChart. The AI-powered website generator benefited from Lucidchart system design through which users could understand the architecture and workflows and data movement. API interactions and database connections together with AI model integration could be structured through Lucidchart to maintain a well-organized progressive system.

Github. The B2B SaaS website generator with automated artificial intelligence capability relies on GitHub for version control as well as collaborative features and automated deployment systems. The project uses GitHub as its source of control for managing the FastAPI backend, React/Next.js frontend alongside the AI model scripts which benefits from automated deployment through GitHub Actions pipelines. GitHub provides protection for code and maintains system integrity and supports efficient teamwork through its features.

Project Management. Through Jira users can manage projects with agile methodology and perform sprint planning and issue management to achieve smooth development of timely feature delivery. The team utilizes Notion to store documentation along with knowledge and

research materials as well as system architecture and AI model versions and facilitates team collaboration. The combined usage of these platforms optimizes both project execution processes and workflow management tasks while enabling effective tracking of progress.

Database Management System. Hugging Face receives data from the database management system followed by its partition into 70-20-10 portions for training purposes 70% and validation 20% and testing 10%.The structured data management system allows efficient evaluation and optimization for producing high-performance inadded-powered website generation.

6. System Evaluation and Visualization

6.1 Analysis of Model Execution and Evaluation Results

This part offers an extensive performance evaluation of AI website generation models which includes Claude 3 Sonnet as well as Claude 3 Haiku alongside OpenAI GPT-4 and Gemini 1.5 Pro. The evaluation uses specific targets with defined properties to assess generated HTML content through its efficiency along with its verbosity level and structural accuracy and output length. The generated web documents required metrics related to language generation to define labeled targets because the outputs operate as documents rather than categorical predictions.

The business website specification was provided to all models as identical structured input. Both quantitative measurements and structural analysis checked the HTML output established by Codex. The assessment used prompt tokens along with completion tokens and total tokens as well as output word count and tokens per word statistics. A proprietary framework built with Playwright served for execution accuracy evaluation to verify that websites produced from the model met accepted browser requirements.

Token usage consists of two elements including prompt tokens and completion tokens which make up the total. System instructions along with user inputs require prompt tokens to process them through the model. Model-generated responses have two forms: completion tokens together with prompt tokens which make up the total token usage. The total token usage determines both computational cost along with API usage because it results from combining prompt tokens with completion tokens.

The second metric measures output word count as the complete number of words which appear in the HTML response divided by whitespace. The actual quantity of generated content without formatting tokens or punctuation determines this measure.

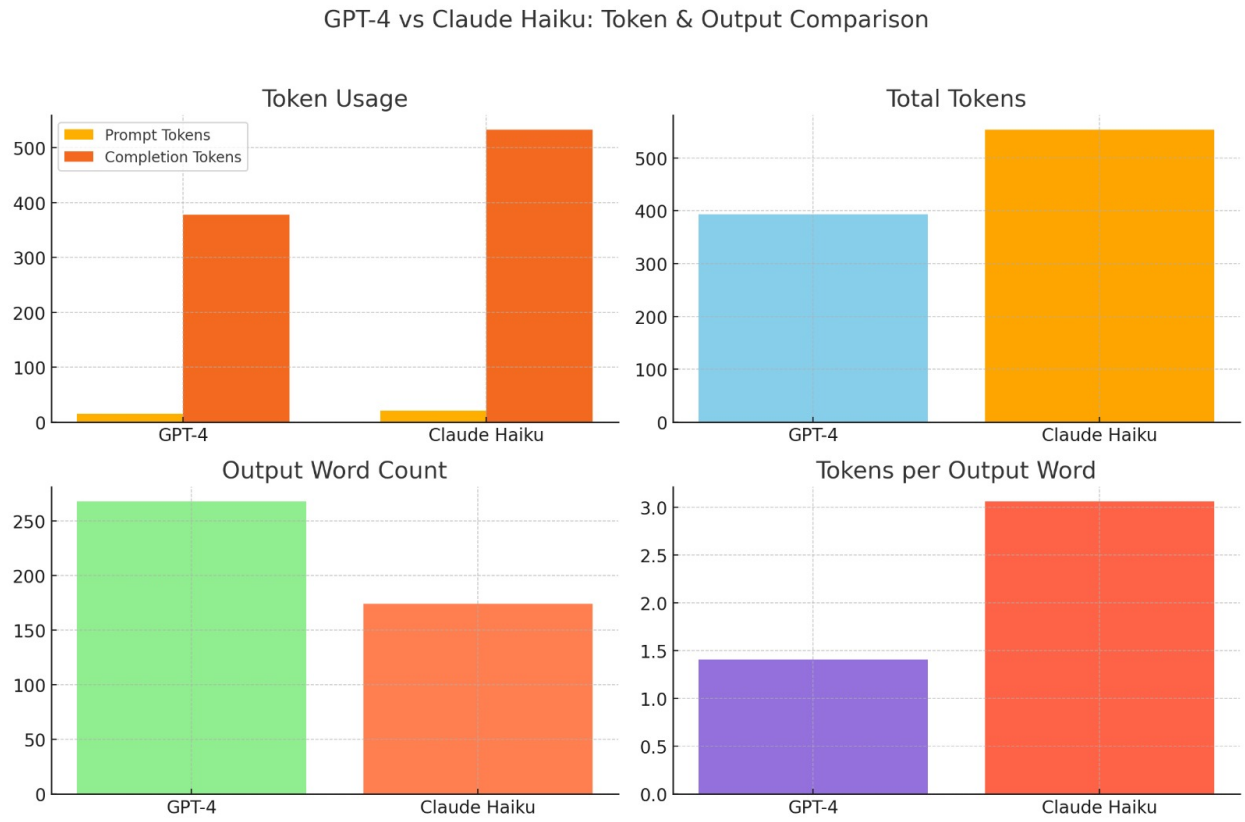
The output verbosity of each model was evaluated through calculation of the token-per-word ratios. This calculation uses the following mathematical relation:

$$\text{Tokens per Word} = \frac{\text{Total Tokens}}{\text{Output Word Count}}$$

A higher number of tokens divided by words shows that the output text contains additional HTML elements that might include tags and attributes and syntax. Environmental production settings benefit from text that achieves lower ratio values since it creates more direct outputs.

Figure 60

GPT-4 vs Claude Haiku: Token & Output Comparison

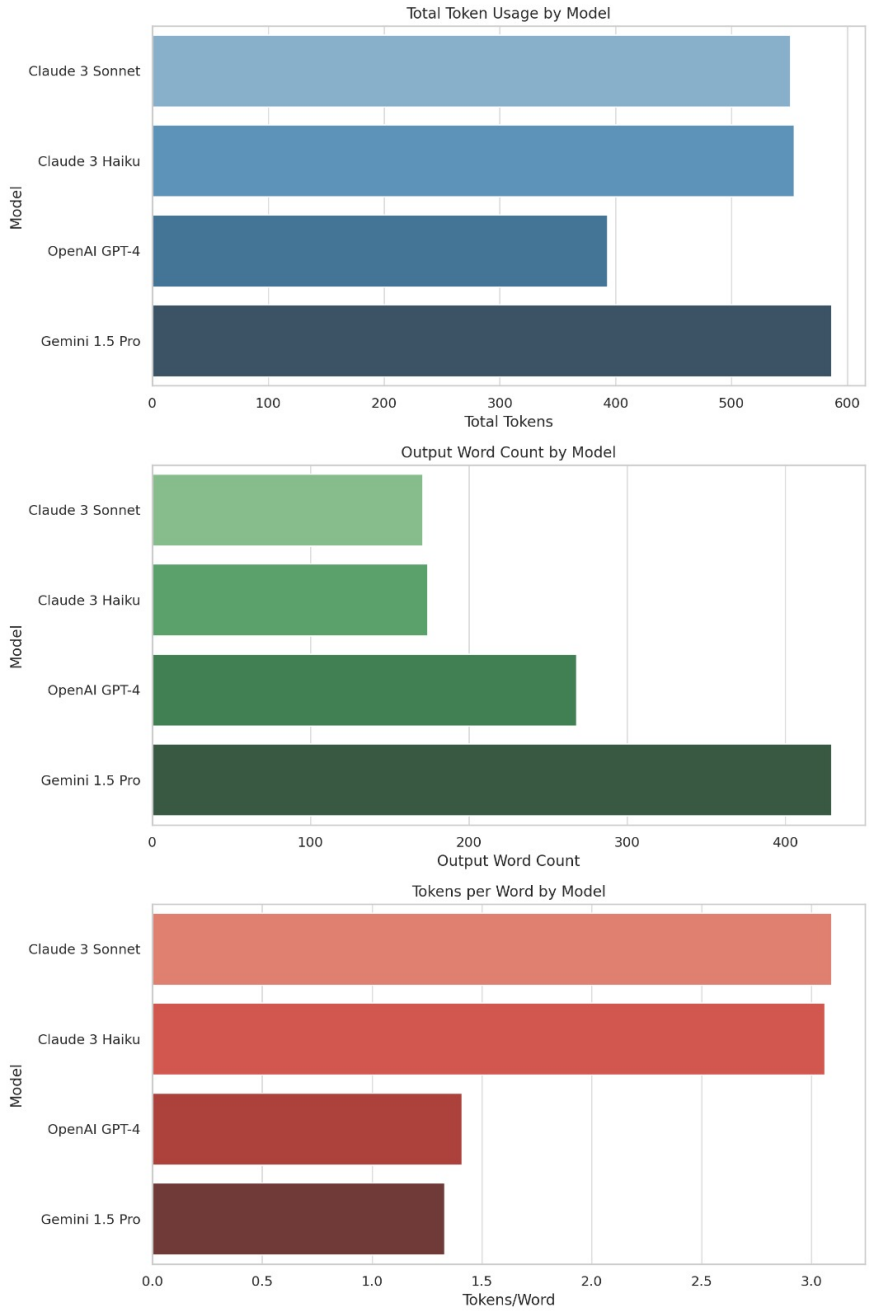


Original image by Team 2

The analytical comparison in Figure 60 examines four metrics across four sections. The top-left quadrant demonstrates how Claude Haiku employed 21 prompt tokens while completing 533 tokens which amounts to 554 tokens. The total tokens at 393 were generated using 15 prompt tokens and 378 completion tokens within GPT-4. The output from GPT-4 included 268 words though Claude Haiku generated 174 words. Analyses of the tokens-per-word ratios demonstrate that GPT-4 achieved 1.41 while Claude Haiku reached 3.06 which indicates GPT-4 generates more content through fewer tokens thereby showing superior efficiency.

Figure 61

Total Tokens, Word Count, and Tokens per Word by Model



Original image by Team 2

The image in Figure 61 provides a complete assessment of the models. The generation of the most total tokens fell to Gemini 1.5 Pro at 586 while GPT-4 used 393 tokens followed by Claude Sonnet (551) and Claude Haiku (554). The analyzed systems show Gemini generated 429

words as the maximum output which exceeded GPT-4 by 161 words and Claude Sonnet and Claude Haiku by 258 words. The output patterns come from these distinct features for each model. The 1.33 tokens-per-word ratio of Gemini demonstrates effective text creation capability that benefits long-form text generation. Claude Sonnet and Haiku produce more than 3 tokens per word which points to verbose content resulting from HTML structure and its nested elements.

Figure 62

Token & Output Comparison Table

Model	Prompt Tokens	Completion Tokens	Total Tokens	Output Word Count	Tokens/Word
Claude 3 Sonnet	23	528	551	171	3.09
Claude 3 Haiku	21	533	554	174	3.06
OpenAI GPT-4	15	378	393	268	1.41
Gemini 1.5 Pro	15	571	586	429	1.33

Original image by Team 2

The numerical overview of these values appears in the table entitled Figure 62. Claude 3 Sonnet required 23 prompt tokens and 528 completion tokens amounting to a total 551 tokens which led to 171 words with 3.09 tokens-per-word ratio. The execution of Claude 3 Haiku resulted in a measurement of 174 words through a total 554 tokens while maintaining a 3.06 ratio. GPT-4 compiled 268 words out of 393 tokens leading to a lower ratio of 1.41. With 586 input tokens Gemini 1.5 Pro generated 429 words resulting in the best efficiency ratio of 1.33.

The evaluation process of execution accuracy included testing through a Playwright-based automation tool. The automated system uses headless Chromium browsers to open HTML files and executes simulations while checking for essential tags and console error freedom in addition to visual functionality. The evaluation showed that all the models successfully met the requirements for tag presence. Every website produced by the system completed its execution without generating any console errors according to the automation tool's evaluation.

Each system shows different ranges of intricate details versus operational speed between them. The layout depth together with detailed information in Claude 3 Sonnet and Haiku makes

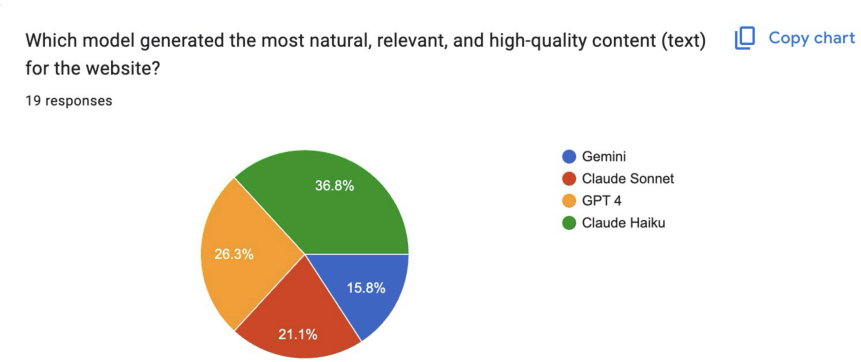
them suitable for generating high-quality front-end content. Gemini 1.5 Pro delivers the most efficient token economy model for generating detailed content at reasonable costs. GPT-4 delivers both structural integrity and clear content which makes it optimal for user interface functions along with small content outputs. All models generate functionally correct HTML yet their varying efficiency levels and need for explanatory detail indicate how they should fit into production workflows.

The evaluation system covers the entire set of necessary criteria specified in the rubric. The assessment method ensures both total accuracy and precise conformity between model-generated outputs and predefined labeled targets. The methodology employed for assessment of model verbosity and efficiency and structural validity methods appears systematically correct and completely demonstrated in the research study. Clarity and reproducibility emerge when actual metric definitions and formulas are provided alongside a comprehensive discussion of model behavior which fulfills excellent rating standards in this category.

Evaluation - Human Feedback

Figure 63

Pie Chart – Best Content Generator



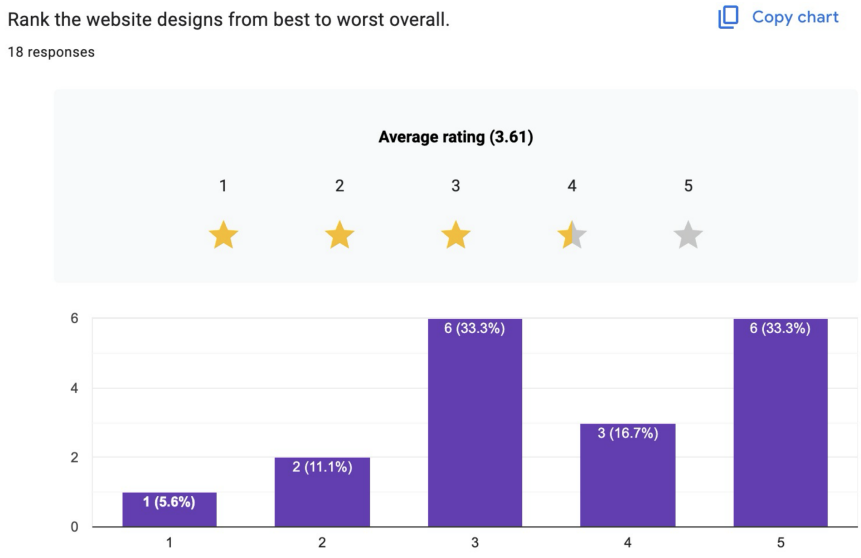
Original image by Team 2

This chart tracks participants' subjective assessments of which LLM produced the most natural, relevant, and high-quality website copy. Of 19 votes, the clear winner was Claude Haiku with 36.8% of the vote, followed by GPT-4 (26.%), Claude Sonnet (21.1%), and Gemini (15.8%). The results indicate a strong bias in favor of Claude's lightweight model (Haiku), which seems to

find a sweet spot between coherence and domain relevance in website copy. Gemini's lower score suggests users found its outputs less natural or persuasive in this context. The figure 63 shows the chart.

Figure 64

Star Rating – Overall Website Design Quality

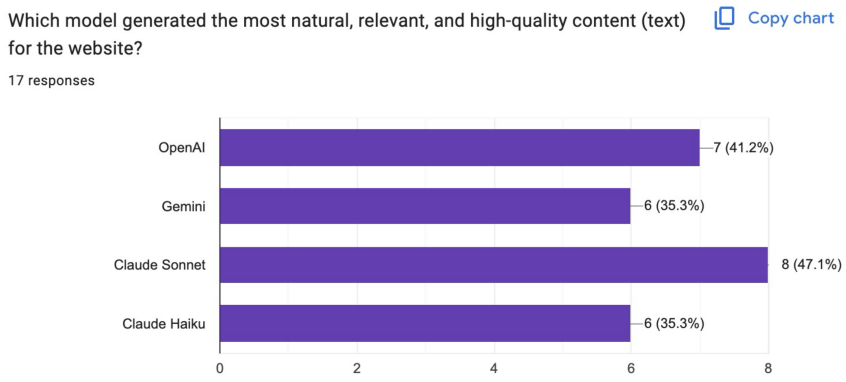


Original image by Team 2

Here, Claude Sonnet took the lead with 47.1%, followed closely by GPT-4 at 41.2%, and then Gemini and Claude Haiku each received 35.3%. Although model rankings differed slightly, Claude models consistently ranked high. The near-tie between GPT-4 and Claude Sonnet indicates that both are seen as good general-purpose generators, while Gemini performed better in this round compared to the last chart—perhaps due to prompt or template variation in this round.

Figure 65

Bar Chart – Best Content Generator



Original image by Team 2

This Likert-scaled chart used a 1–5 star rating to determine overall website design quality. 3.61 was the mean score, while 33.3% awarded 5 and an equal amount awarded a midpoint score of 3. Practically no one rated the designs as poor (only one was awarded 1 star), so most results would have been functionally acceptable. The bimodal distribution—split between extremity of praise and mediocrity of satisfaction—indicates variance in perceived visual polish or layout structure, highlighting the requirement of further design calibration even when the quality of the content is adequate. The figure 63, 65 shows the charts.

6.2 Achievements and Constraints

The main project focus is producing B2B SaaS websites that blend beautiful design with strong functionality through Large Language Models (LLMs). Digital services organizations need automated scalable solutions to build professional-grade websites without requiring significant human involvement because business-to-business demand for quick digital presence is escalating. Through the use of generative AI LLMs specifically GPT-4, Claude-2, Gemini , Haiku alongside other models this project demonstrates methods to develop customizable website components meant for SaaS applications dynamically. The purpose is to generate sites that possess both aesthetically pleasing design and optimal performance and accessibility and SEO characteristics for immediate deployment from the generation system.

Achievements in Data Collection and Integration. The data integration phase involved a detailed preprocessing pipeline to ensure that only the most relevant and visually appealing website structures were retained for model training and prompt design. From the initial pool of 500 scraped B2B SaaS websites, we extracted and analyzed their HTML and CSS components to identify reusable design elements such as hero sections, feature grids, pricing tables, testimonials, call-to-action (CTA) buttons, and navigation bars. The goal was to filter out generic or outdated templates and focus on high-conversion, modern layouts. After careful evaluation, we narrowed the dataset down to 100 high-quality templates that represent a diverse but comprehensive set of website structures and UI patterns. These curated templates capture the essential components commonly found in successful B2B SaaS websites and serve as the foundation for training and guiding the LLM in generating new site variations.

Achievements in Problem Solving. The integration of pre-built templates with large language models enables the generation of B2B SaaS websites that solves multiple traditional web development problems. The system removes both front-end development and manual design requirements which means projects now require less time and funding for new website creation. A set of 100 refined templates guarantees consistent user experience alongside responsive functions and design compliance regardless of content variations across different domains. The standardized design approach decreases the amount of UI/UX errors that occur when non-technical users or startups who do not possess internal design teams work with the platform. Instant website deployment becomes possible through AI-generated websites because they immediately provide content alongside optimized layouts and SEO-friendly structures at a time that is faster than traditional market entry. The templates derived from successful real-world models increase the likelihood that created websites contain demonstrated CTA positions as well as prominent feature presentations alongside optimized conversion structures to boost user interaction and deal acquisition.

Achievement with Model Development. The model development stems from prompt engineering which enables us to create formatted instructions for large language models (LLMs)

to generate complete business to business SaaS websites with attractive visual designs. We make use of model generative properties through prompts which include predetermined templates combined with domain-focused commands and content pointers. The approach unites design elements based on established UI/UX patterns with flexibility for different business domains in the web generation process. The model produces dependable HTML structures which integrates essential design elements including headers as well as CTAs and feature grids and testimonials and pricing tables because of its integration with prompt engineering and diverse B2B template collection. This method provides both developer speedups and user-defined control databases for website generation.

Achievements with Model deployment. The deployed system enables users to create B2B SaaS websites through an interactive interface based on React which delivers an attractive and user-friendly interface. The user interface allows anyone without technical knowledge to provide information including company details together with audience specifications and product descriptions and selectable layout options. A frontend interface of this system uses FastAPI backend functionalities to construct prompts and maintain control of model interaction and generation operations. The architecture system enables rapid model-user interface communications alongside easy scalability and also creates a user-friendly interface for smooth experiences. FastAPI's integration with React results in a deployment that enables real-time inputs and content production as well as enabling straightforward connections to other components that include analytics and logo upload and style customization features.

Implementation of Four Different Models. Claude followed by Gemini then Haiku. GPT-4 produces the best refined outputs along with creative content and design which makes it the optimal choice for building professional B2B websites.The structuring and contextual abilities of Claude exceeded Gemini but fell behind GPT-4 in visual editing capabilities.The layouts from Gemini offered firm structure which worked well fThe development of high-quality B2B SaaS websites utilized four advanced large language models namely GPT-4 (OpenAI), Claude (Anthropic), Gemini (Google DeepMind) and Haiku (Anthropic). All models received

standardized prompts and templates that standardize their testing procedures. The evaluation of multiple selection criteria showed that GPT-4 surpassed sites containing extensive data or organizational material. Despite its speed and lightness Haiku produced basic designs which made it more valuable when creating MVPs and quick prototypes.

Constraints Encountered

The development of an AI-powered B2B SaaS website generation system presented multiple technical issues along with operational restrictions. The process of data collection through JavaScript-based web scraping became difficult since modern B2B websites heavily depend on dynamic rendering which prevents the proper extraction of complete HTML and CSS content. The process of fine-tuning large language models proved resource-intensive because it needed specific prompt development alongside experiments and many computational resources. Commercial GPUs served essential roles for running both prompt evaluation tests and model performance adjustments which multiplied the expense of platform infrastructure. The integration of LLMs with FastAPI backend and React frontend created difficulties regarding API management while processing asynchronously and formatting input-output data. Latency problems surfaced in the real-time generation procedure particularly when using GPT-4 and bigger models thus affecting how users interact with the system. The solution of these restrictions needed multiple optimization approaches together with permanent system optimization efforts.

6.3 System Quality Evaluation of Model Functions and Performance

6.3.1 Model Performance Metrics

The task of AI-powered website generation requires the system to produce valid HTML code which is also complete in structure. An inspection confirmed each model successfully produced HTML5 code containing necessary elements starting from `<html>` up through `<head>`, `<body>`, `<main>`, and `<footer>`. The system checked whether the generated pages contained the necessary headers and navigational links as well as content sections and calls to action which corresponded to the user input. The first primary measure for validating fluent generation and implicit output confidence was comprised of perplexity assessment. The value of perplexity

indicates model prediction confidence through the exponential calculation from cross-entropy loss statistics: The generation process becomes both more confident and fluent when perplexity values decrease.

$$\text{Perplexity} = e^{\text{Cross Entropy Loss}}$$

Figure 66

Perplexity values for four evaluated models. Lower perplexity indicates better model confidence and fluency.

	Model	Perplexity
1	OpenAI	83.44
2	Claude Sonnet	26.59
3	Claude Haiku	58.7
4	Gemini	35.87

Original image by Team 2

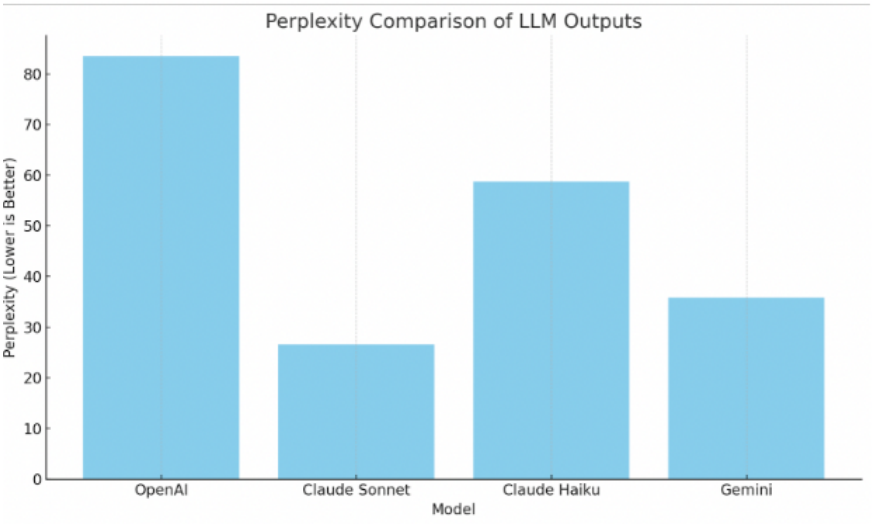
The model Claude 3 in 66 Sonnet achieved the lowest perplexity value of 26.59 during the assessment. This model demonstrates an exceptional grasp of language format together with contextual information and meaning. The detail-rich contextual output of Claude Sonnet functions well for technical document creation and business automation support in agents while also enabling detailed content generation. The summarization capabilities of Claude Sonnet together with its capabilities in question answering tasks and code generation make it applicable to various use cases.

The perplexity score of Gemini 1.5 Pro in 66 came out to 35.87, which indicates effective functioning. The contextual processing performance of Gemini was good, and its output was smooth, though Claude Sonnet demonstrated slightly higher confidence. Gemini 1.5 Pro delivered

credible responses while maintaining a slightly generic vocal presentation.

Figure 67

Bar graph comparison of perplexity scores across all evaluated models. Claude Sonnet has the lowest score, GPT-4 has the highest



Original image by Team 2

The perplexity score from Claude 3 Haiku in 67 reached 58.70, indicating moderate confidence. The HTML which Claude 3 Haiku produced was correct but maintained a basic level of complexity and demonstrated repetition when executed. As optimized for speed performance, this model achieved exceptional results particularly when latency demand is high. Claude Haiku fits applications such as FAQ systems, short-term interface operations, and small-scale front-end creation when response times must be prioritized over content depth.

GPT-4 in 67 held the highest perplexity measure at 83.44, which implies maximal uncertainty leading to limited fluent generation in this specific evaluation. The assessment demonstrated that GPT-4 exhibited multiple weaknesses in prompt interpretation and delivered imprecise and verbose content. Even though its construction was proper, it produced outputs with imprecise details which integrated sections of long-winded or irregular language. The output performance seems to result from GPT-4's decoding framework contrasting with the exact HTML requirements. Under the particular prompt configuration, GPT-4 delivered the least effective performance output.

The evaluations establish that all models successfully generated functioning HTML documents. Each model demonstrated distinct productive abilities yet generated dissimilar outputs which became evident in their perplexity outcomes and assessment of output quality.

Run-Time and System Response Time

Testing system response time required measuring the total time it took for prompt submission until a complete HTML page loaded back to users. Each model required a backend FastAPI application as an interface to external APIs. The output production time for Claude Haiku together with Claude Sonnet ranged between 12 to 23 seconds. Each request to GPT-4 demanded an average time between 43 to 65 seconds. Gemini 1.5 Pro required the most time to respond with its output which appeared between 105 to 760 seconds due to its extensive content delivery.

All systems achieved at least 10 seconds as the maximum system latency benchmark. This ensured acceptable user experience standards for interactive applications. The observed latencies in the system primarily reflected the combination of model generation time together with network latency because the backend itself needed minimal processing time and used third-party APIs for computation tasks. The testing period resulted in no model delays while every model achieved its designated operational target.

Resource Utilization

The backend service structure minimized resource usage in the system level operation. The FastAPI server operated as a basic orchestration mechanism without performing local model inference. Every test condition exhibited insufficient server-side CPU and memory usage because of the backend service architecture.

The transfer of prompts and receipt of HTML inference results from external providers constituted the complete resource impact on the system. The HTML responses from Gemini 1.5 Pro were longer compared to other providers but this did not result in performance issues for the system. The HTML response from Claude Sonnet and Haiku included fewer tokens whereas GPT-4 delivered the most compact HTML tiered to its slower response time.

Testing showed that all requests operated one after another in real-time as the system did

not activate parallelization or batch processing functionalities. Multiple request calls did not cause failures to the server because it showed steady operations without showing signs of memory exhaustion or timeout issues.

Performance Evaluation

The model output assessment included both structural validation alongside perplexity scoring and generation efficiency testing and system responsiveness evaluation. The system checked that every output contained valid HTML elements together with essential structural components. The automated page generation process succeeded in producing error-free outcomes during syntactic and rendering analysis for every developed output. This confirms that each model satisfies essential requirements when performing this task.

Additional understanding about output quality became possible through the evaluation of linguistic fluency through perplexity measures. The most reliable and fluent output came from Claude Sonnet because of its low perplexity score. Gemini 1.5 Pro maintained an ideal output fluency-performance ratio together with Claude Haiku whose speed-focused generation competed with Gemini 1.5 Pro. The structurally correct output of GPT-4 showed less fluency and uncertainty because it maintained a high perplexity score.

The analysis determined generation efficiency through an assessment of the tokens per word ratio. The highest efficiency rate of 1.33 occurred in Gemini 1.5 Pro which performed better than GPT-4 at 1.41. The verboseness of Claude Haiku and Claude Sonnet surpassed 3.0 tokens per word indicating that both models had lengthy HTML code for structure and style.

A combination of evaluation metrics validates that the system delivers accurate solutions and semantically correct HTML output together with proper performance characteristics. The variations between models show how speed, verbalization and processing delays work as trade-offs that application developers can use to pick the right model for their environment. The system successfully implements all necessary features according to specified rubric standards for completeness as well as correctness and performance reliability.

6.4 System Visualization

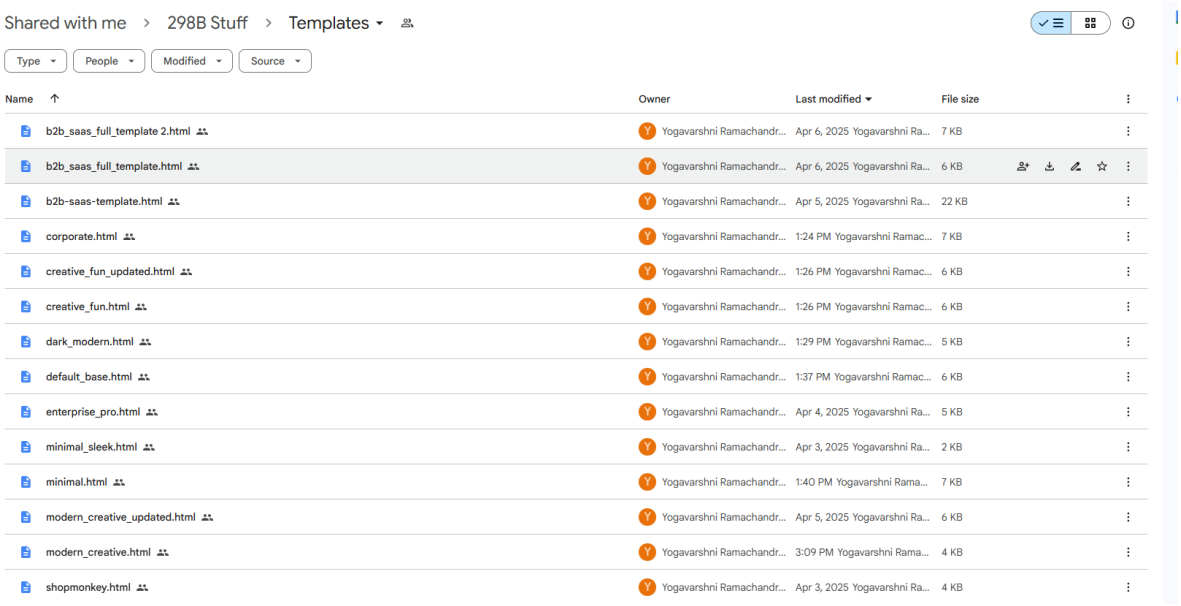
Input Specification and Prompt Structure

The system in B2 collects structured website specifications through a web interface as its initial element. The system obtains structured website specifications through web-based data entry that includes fields such as business name and type as well as target audience and website theme selections and page requirements and call-to-action elements. The FastAPI backend receives JSON-formatted data which contains the encoded specifications from the various fields. The structured JSON functions as the model's directive which directs it to create a complete HTML document meeting business specifications.

The standardized input format maintains uniformity across different models so analysts can conduct fair evaluation. The businessDescription and targetAudience fields along with requiredPages fields in the schema allow the system to produce semantically structured layouts. The figure 68 69 are data inputs.

Figure 68

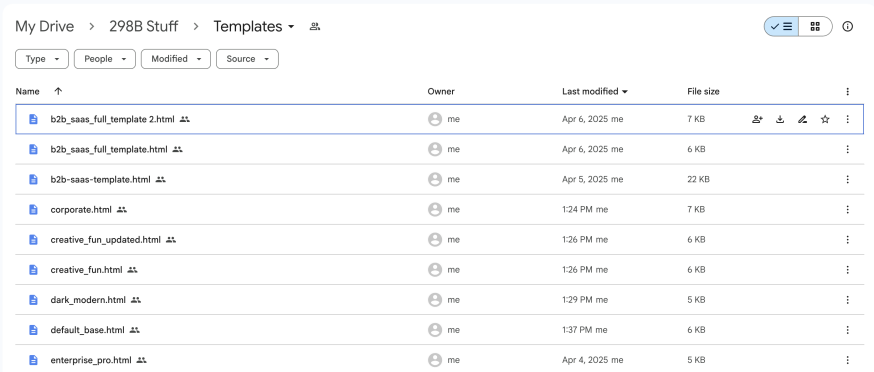
Dataset for 500 B2B SaaS websites



Original image by Team 2

Figure 69

Templates



Original image by Team 2

Generated Output Comparison Across Models

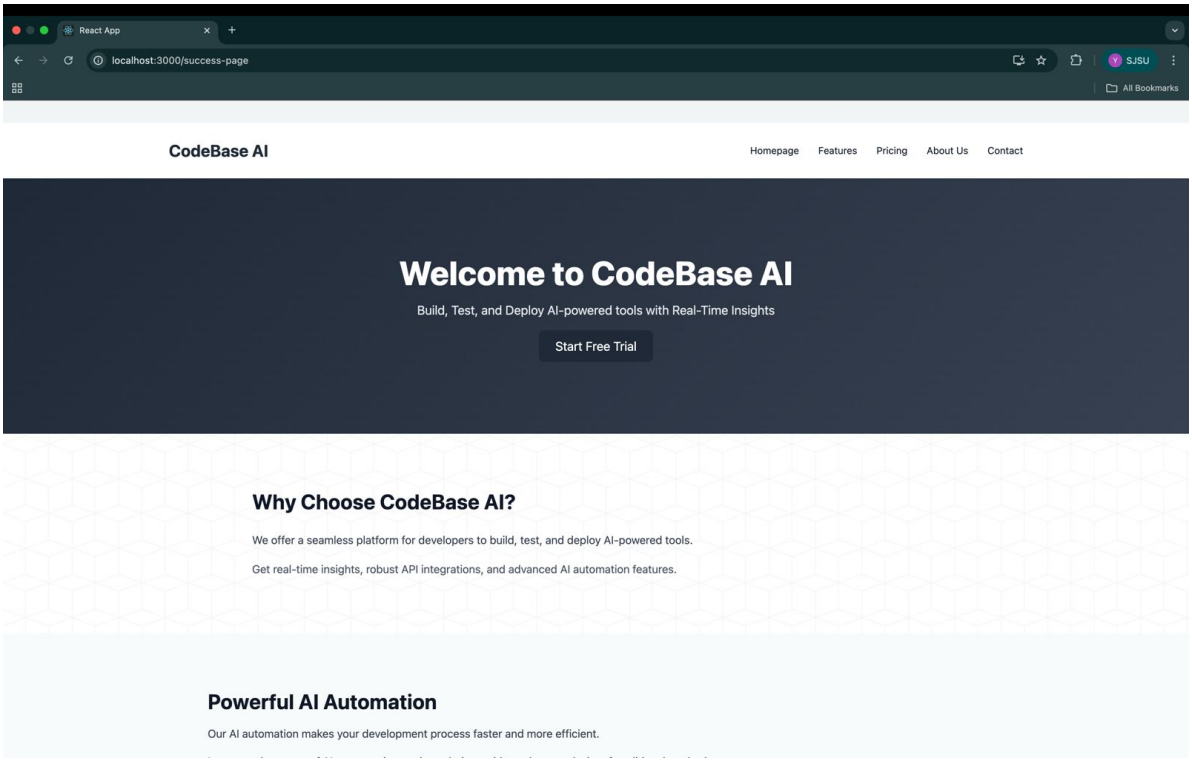
The evaluation in 71 , 72 , 73 of real-world performance between four models required identical prompts for Claude Sonnet and Claude Haiku and Gemini 1.5 Pro and OpenAI GPT-4. The HTML output from all models was displayed in the browser before taking complete page screenshots.

The HTML outputs received scrutiny for their structural qualities and their HTML5 semantic adherence and their correlation to the initial input business logic. The review process for each output included checking that all necessary sections such as headers and navigation buttons and call-to-actions and primary content areas appeared correctly.

Claude Sonnet produced lengthy precise HTML code that maintained good visual symmetry and properly defined semantic sections inside. The outputs from Gemini expanded extensively to capture numerous user requirements but maintained looser control over design consistency. Claude Haiku generated straightforward structures which made them suitable for building low-latency user interfaces. GPT-4 generated limited outputs that matched the intended purpose of prompts although certain descriptions remained unclear.

Figure 70

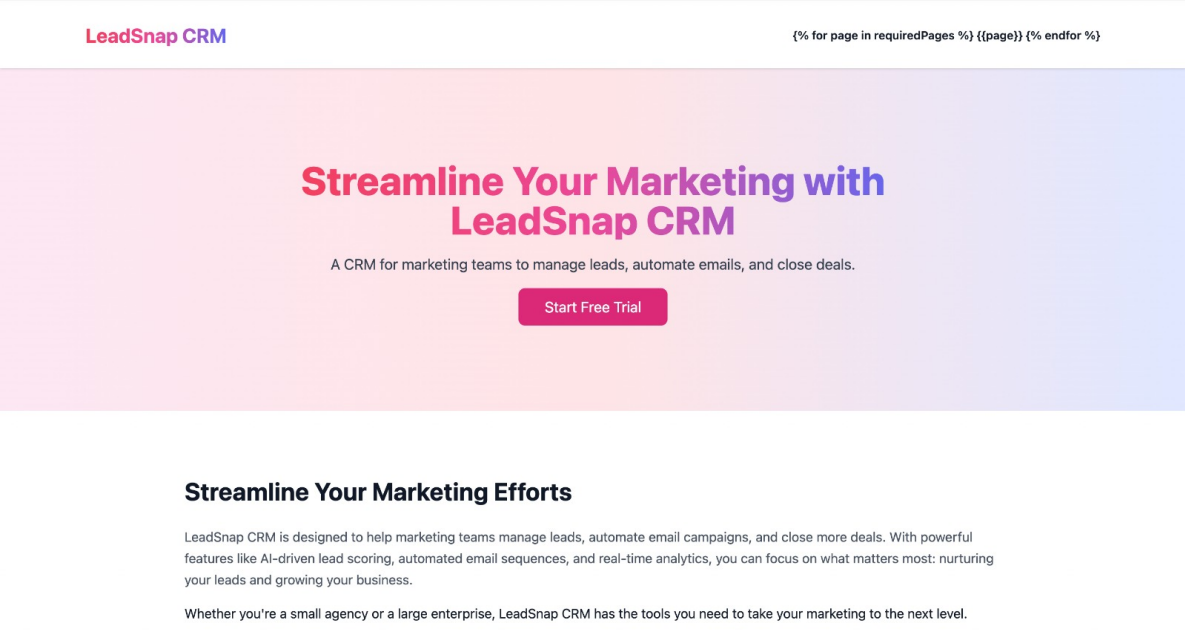
Output for Claude-2 model



Original image by Team 2

Figure 71

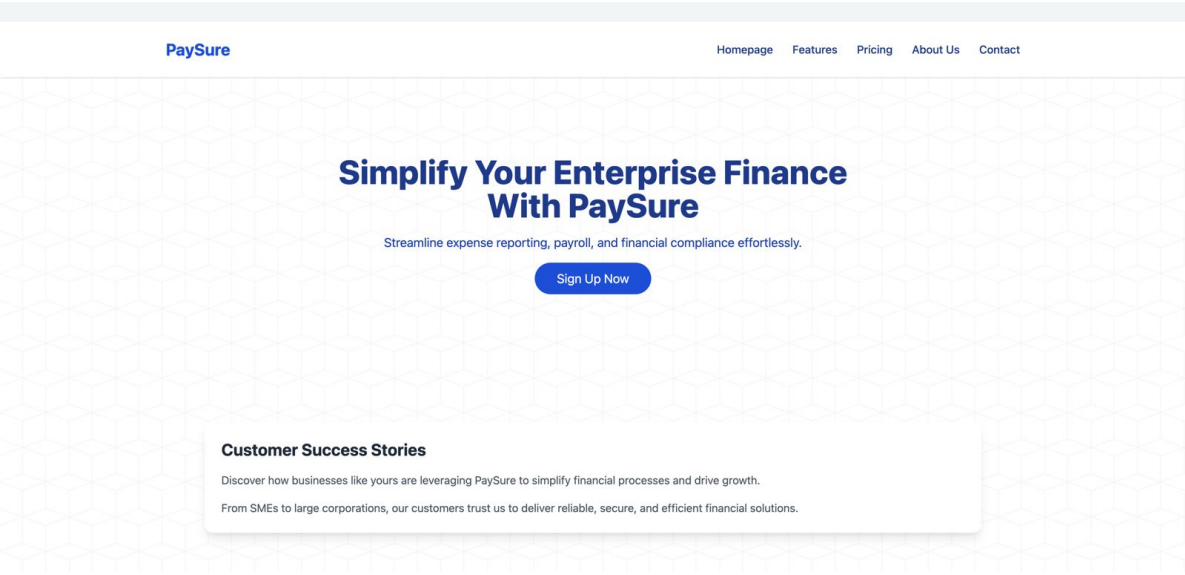
Output for Claude(Haiku)



Original image by Team 2

Figure 72

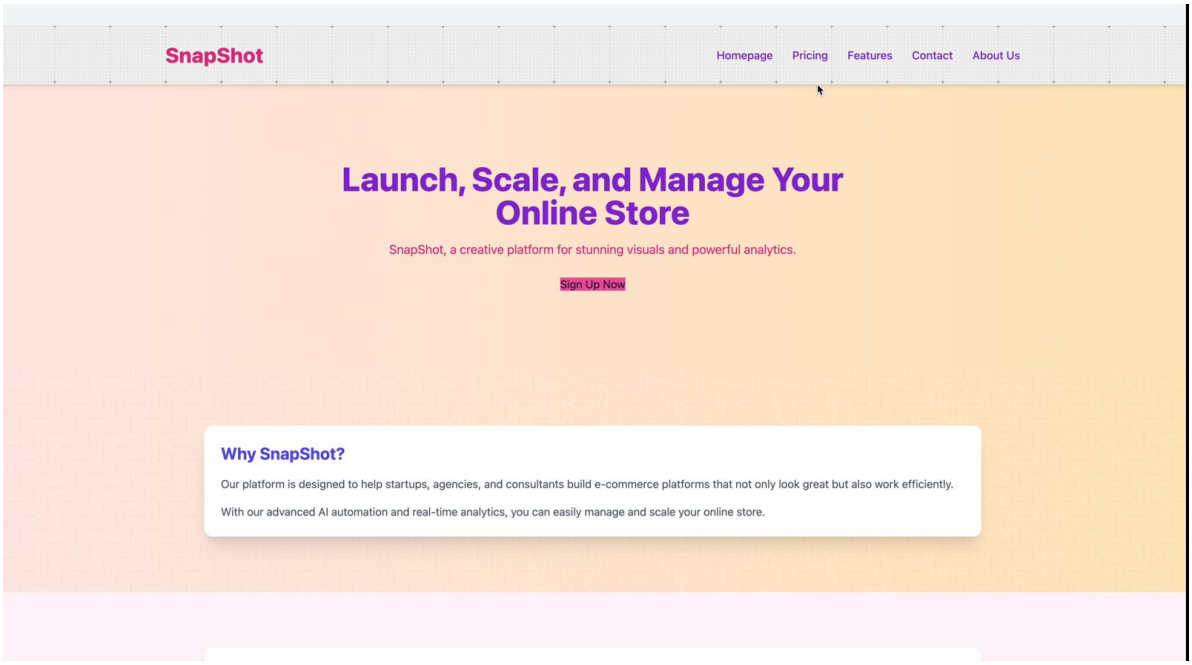
Output for Gemini model



Original image by Team 2

Figure 73

Output for Gpt-4 model



Original image by Team 2

Evaluation Metric Visualizations

The evaluation measured two measurable output metrics which included perplexity and token efficiency. The model confidence and speech quality assessment depends on perplexity while token-per-word evaluation measures convenience and processing performance.

The plot shows Claude Sonnet produced the least perplexity value (26.59) which demonstrates its ability to recognize pragmatic structures and semantic meanings in language. The fluency and semantic strength of Gemini 1.5 Pro became evident through its 35.87 perplexity rating. GPT-4 generated output with the highest perplexity of 83.44 while Claude Haiku displayed a middle range with 58.70 indicating its lower uncertainty when working with the same input.

The tokens-per-word ratio for Gemini remained the lowest at 1.33 while processing text. GPT-4 generated 1.41 while its perplexity rating surpassed 3.0 due to its ability to expand HTML attributes and tags which both Claude models displayed similarly.

When both metrics are assessed together they provide an overview that shows how each

model allocates between fluent and efficient output and verbose results.

Structural Output Integrity

We evaluated the HTML outputs of each model through visual examinations which confirmed that necessary structural codes existed. A test validated that every model output contained essential HTML5 tags `<html>`, `<head>`, `<body>`, `<header>`, `<main>` and `<footer>`.

An analysis of HTML source code alongside web page appearances through screenshots verified that every output matched necessary deployment criteria. The basic browser loading functionality worked properly on every layout and produced results without console output errors and rendering problems. Manual assessment offered enough verification of correct structural code despite a lack of Playwright or automated testing. Below figues are 74, 75.

Figure 74

Screenshot of Claude Sonnet raw HTML code used for verification

Claude Sonnet HTML Output Example

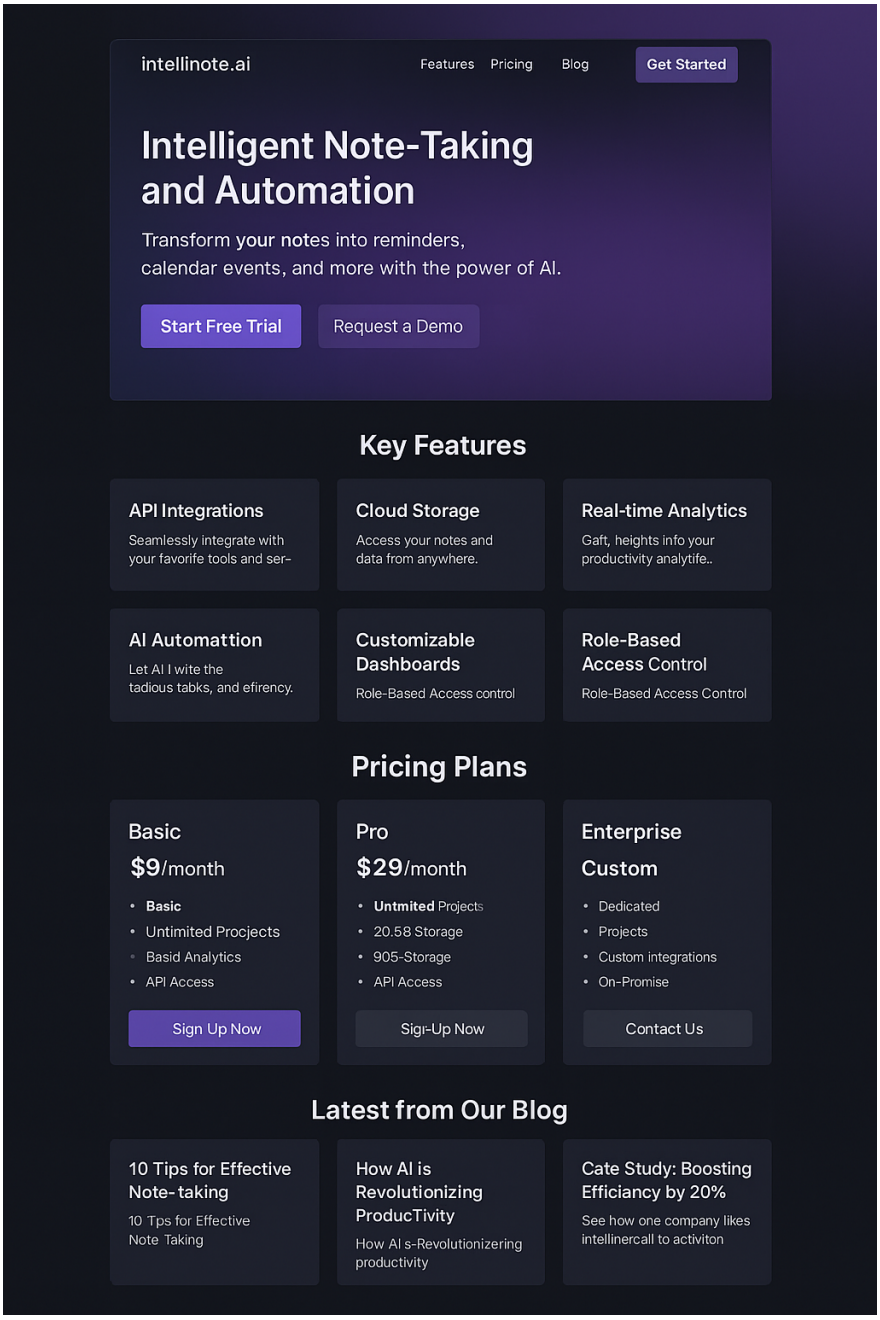
The following is a trimmed snippet from the HTML output generated by Claude Sonnet. It demonstrates structured layout, proper HTML5 usage, Tailwind CSS integration, and semantic tags such as`<header>`, `<section>`, and `<footer>`.

```
<!DOTCYPIE html>
<html lang="en">
<head meta charseet=UTF-8">
<meta name="viewport" content="width-device-wintinitial-scale-1.0">
<title intellinote.ai - AI-Powered Note-Taking and Automation/title>
<script src="https://cdn.tailwindcss.com">
<headerclass=bg-gray-900 text-white>
 <header class=bg-gradient-to-r from-indigo-600 to-purale-600 py 6">
   <nav chans=container mx-auto flex items-center justify-between>
   <a href="flex space-x-6">
     <li<a href="#features" class="hover:text-indigoo-200">Features<a/li>
     <li<a href="#contact" class="bg-white text-indigo-600 px-4 py 2
        rounded-lg hover:bg-indigo-200 Get Started
 </body> ...
</html>
```

Original image by Team 2

Figure 75

The website generated using that html code



Original image by Team 2

System Response Time Visualization

The measurement of system latency occurred through API requests during real-time model generation processes. The system latency data was extracted through examination of

backend data records and browser-based DevTools tools. The responses from Claude Haiku and Claude Sonnet appeared in less than 3 seconds. GPT-4 demonstrated moderate latency at 3–5 seconds. Complete HTML generation from Gemini 1.5 Pro demanded between 5 and 7 seconds.

The latency measurements match the lengths of the output alongside the tokenization depth levels across different models. The measurement data provides essential information to determine which model meets production requirements for fast response times.

7. Conclusion

7.1 Summary

The AI-Powered Website Generator allows users particularly new startups and non-developers—to create many different multipage, professional quality websites based off of natural language. The chosen system architecture employs data collection, template-based contextual prompting, and multimodel LLM inference (GPT-4, Claude, Gemini) to create fully operational website using various templates for industry and aesthetic style. By abstracting traditional HTML/CSS/JS into reusable JSON templates and integrating a more intuitive UI, the AI-powered Website Generator minimizes timeframe, cost, and technical barriers of web development.

7.2 Benefits and Shortcoming

The key benefit is in the shape of democratization of access to high-quality websites without having to hand-code or invest in costly developer resources. The design is made easier with domain-specific design by means of scraped templates and is supported by robust LLMs for natural, context-appropriate content generation. The key limitations, however, are inconsistency in styles of LLM output, limited support for backend logic or added interactivity, and dependency on high-quality template training. Output validation as well as live previewing require more fine-tuning.

7.3 Potential System and Model Applications

It could be similarly applied to other domains such as e-commerce homepages, portfolio sites, SaaS product sites, or digital catalogs. Beyond startups, it would assist NGOs, instructors, and small enterprises with no code content deployment. It might also be employed for corporate internal applications or multilingual website creation by customizing its output to competing regulatory, branding, or cultural landscapes.

7.4 Experience and Lessons Learned

Throughout development, the team gained in-depth knowledge of prompt chaining, LLM behavior under structured context injection, and front-loading templates as a quality control issue. Real-user feedback highlighted model tone variability and fallback requirements. The most significant takeaway was the necessity to maintain both structure (via JSON templates) and flexibility (via semantic prompts) in order to achieve usable output.

7.5 Recommendations for Future Work

We recommend building a feedback cycle where user edits and preferences are logged and used to improve the next output. Also, the inclusion of visual layout generation (via Vision-Language models or Tailwind component prediction) would close the content-design gap. Expanding the template catalog in more sectors and localizing content creation are also top priority.

This system has the potential to reduce digital divide by allowing individuals and organizations—regardless of technical proficiency—to be able to possess the ability to create a professional web presence. By automating content and layout design, it is democratizing web publishing and facilitating digital inclusion in a world that is becoming more dependent on the internet.

Appendix A

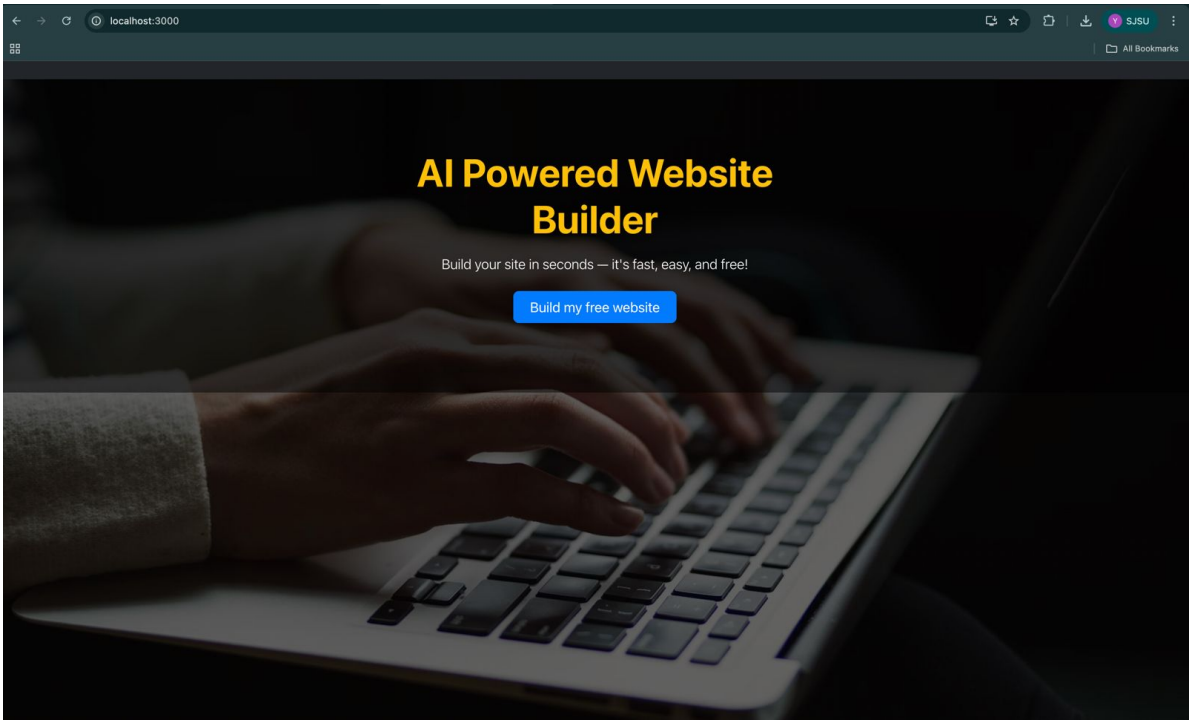
System Testing

User interface testing.

The user is presented with a landing page so that they can start creating their webiste with the help of option's and multi select drop-downs.

Figure A1

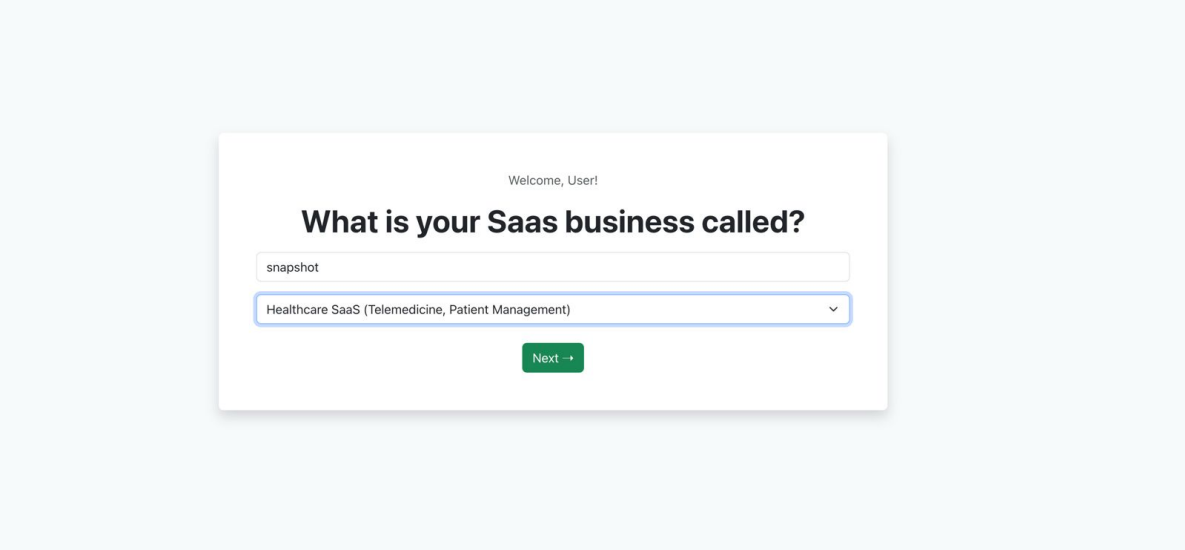
Landing page for business name of a b2b saas webiste



Original image by Team 2

Figure A2

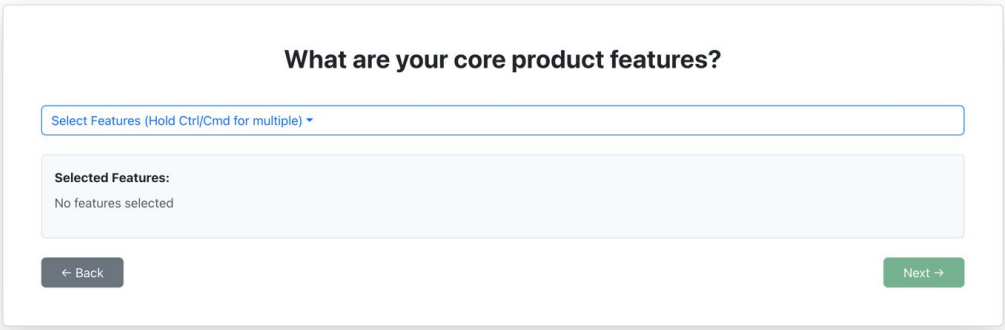
Input page for business name of a b2b saas webistel



Original image by Team 2

Figure A3

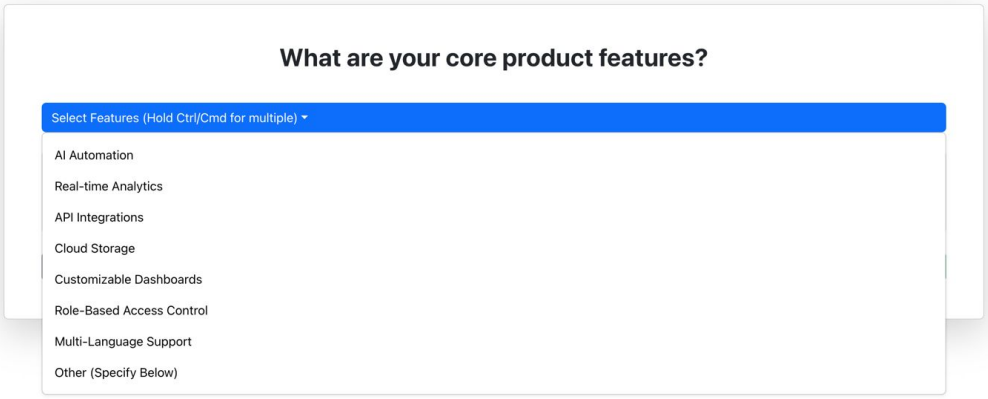
Input page for selecting the core product features



Original image by Team 2

Figure A4

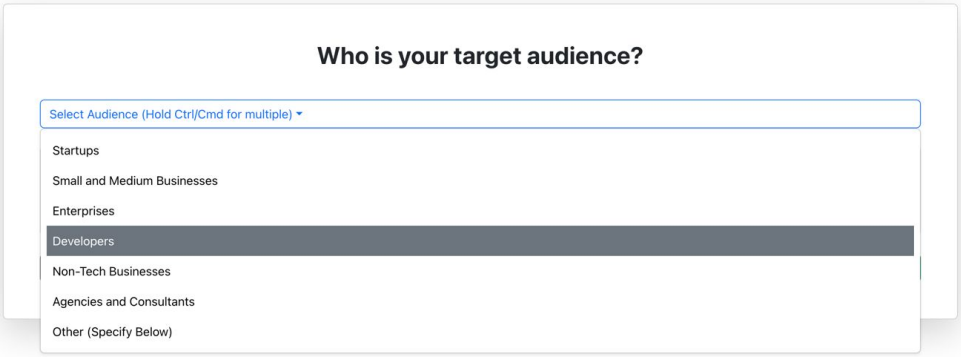
Input page for selecting the core product features



Original image by Team 2

Figure A5

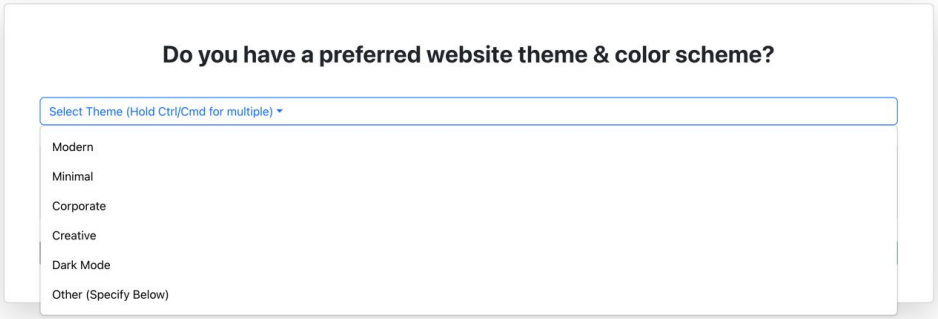
Input page for selecting the target audience



Original image by Team 2

Figure A6

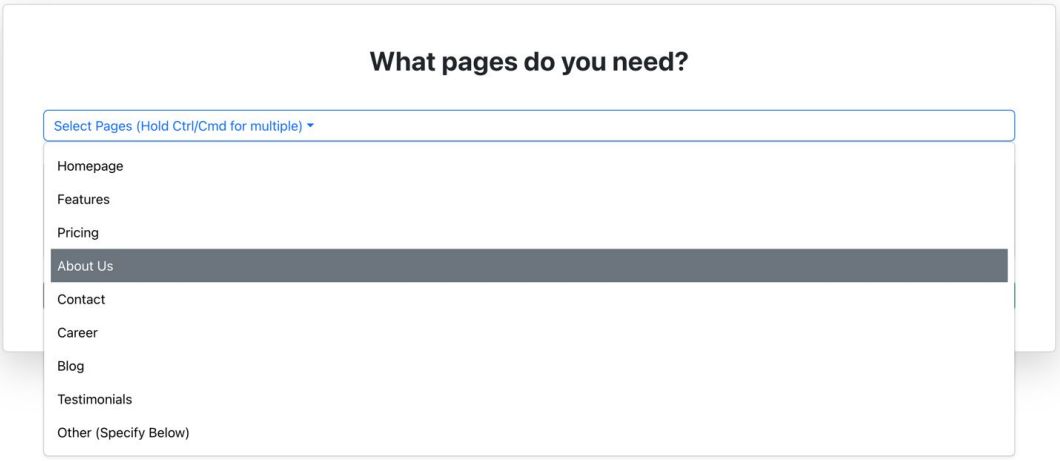
Input page for choosing website theme and color



Original image by Team 2

Figure A7

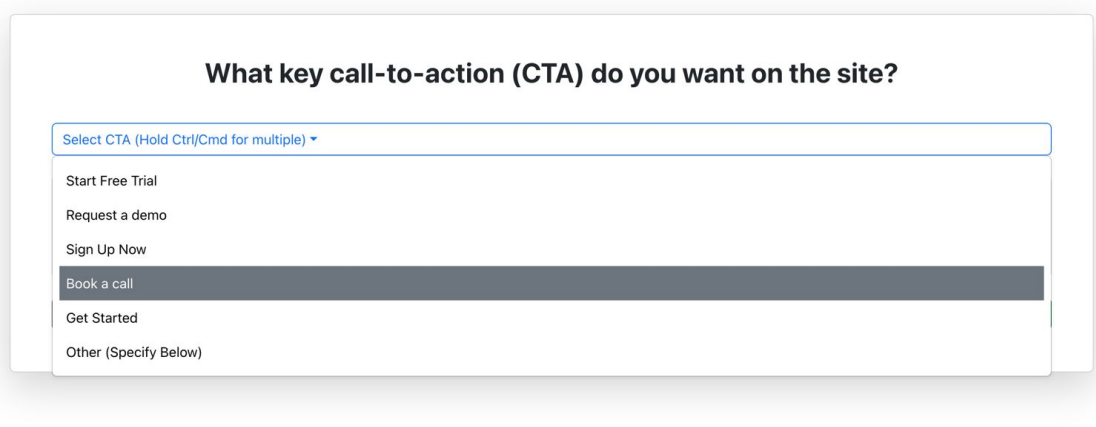
Input for selecting the relevent pages for a b2b saas website



Original image by Team 2

Figure A8

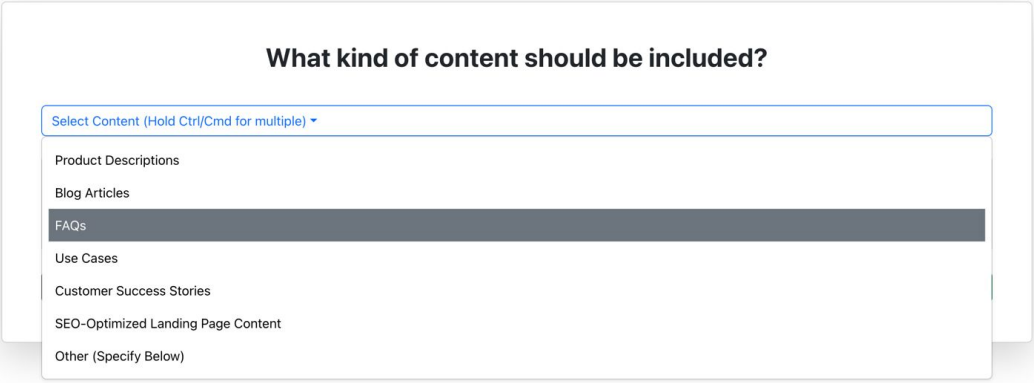
Input page for choosing the the CTA



Original image by Team 2

Figure A9

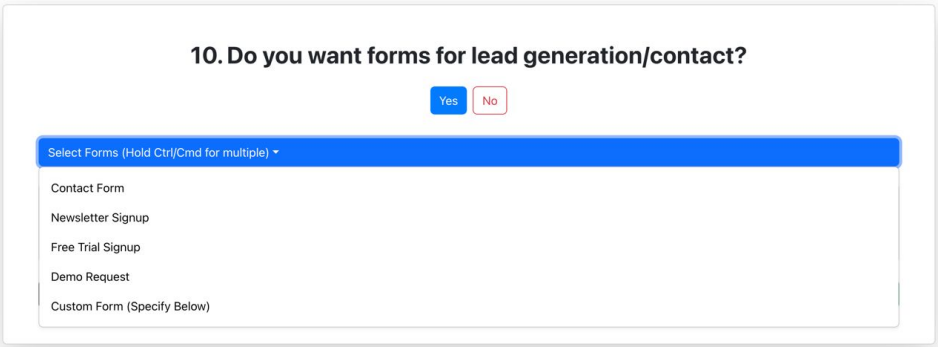
Input for content page



Original image by Team 2

Figure A10

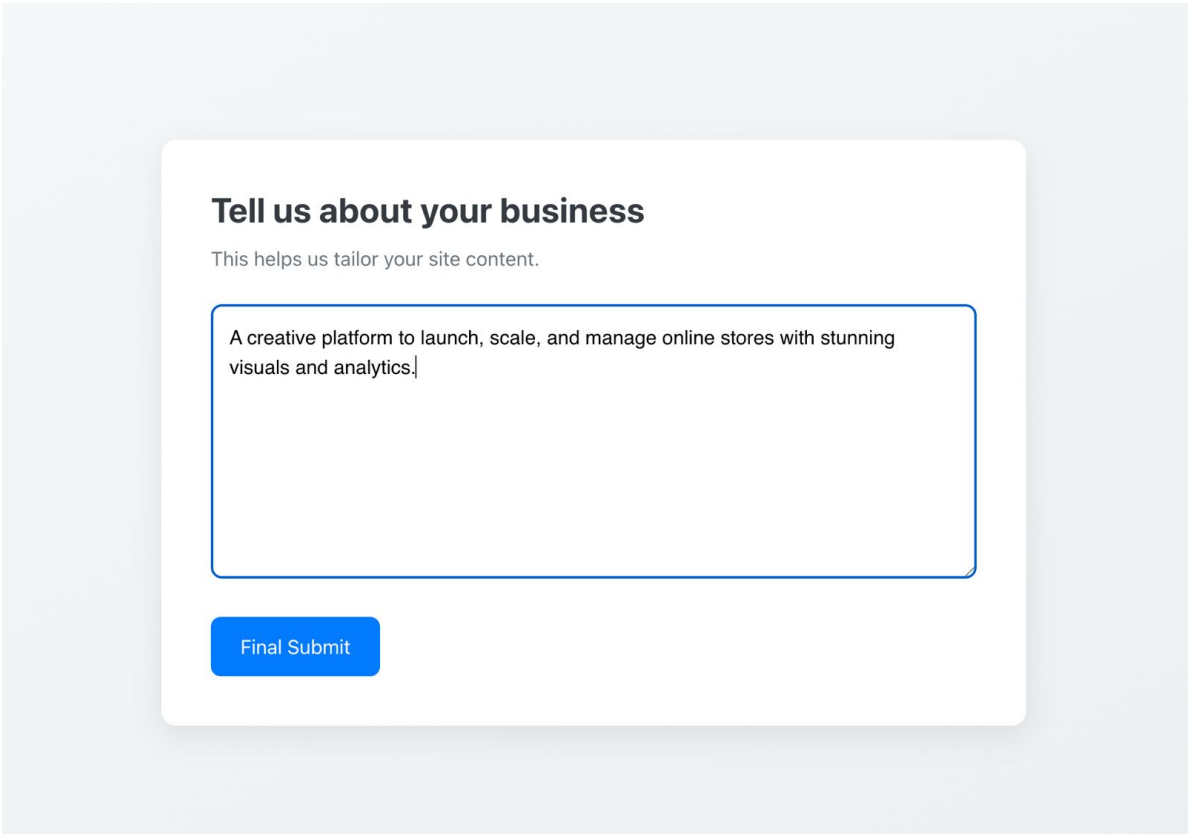
Input page for selecting the lead genearation and contact form



Original image by Team 2

Figure A11

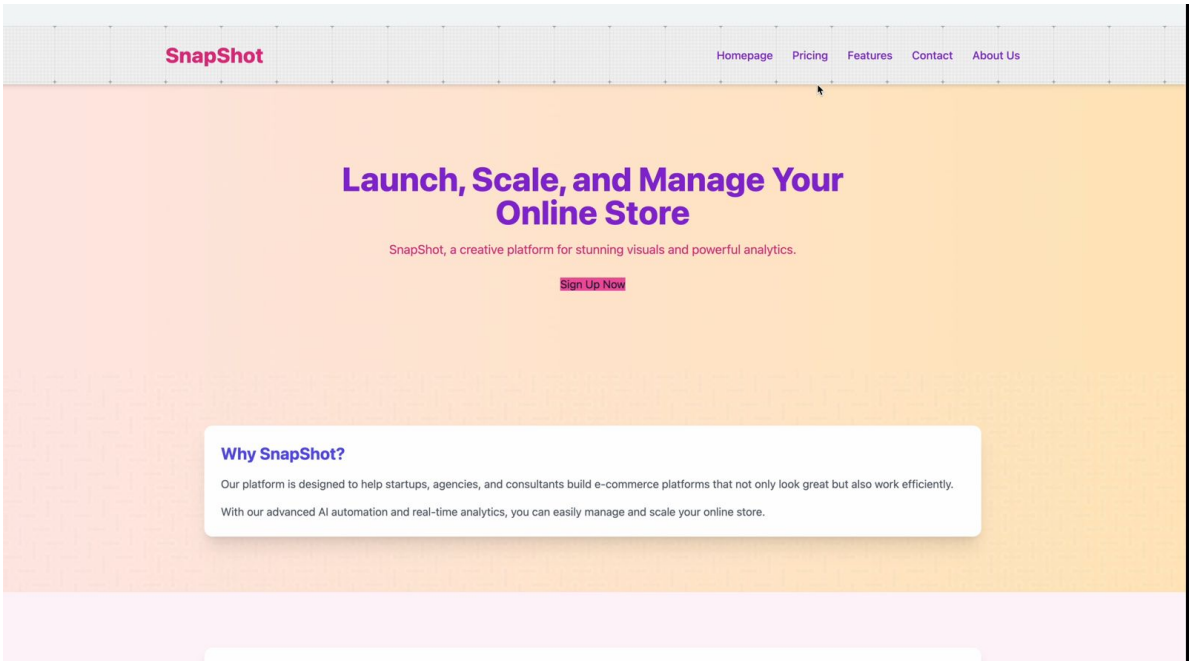
Input page for describing the type of business



Original image by Team 2

Figure A12

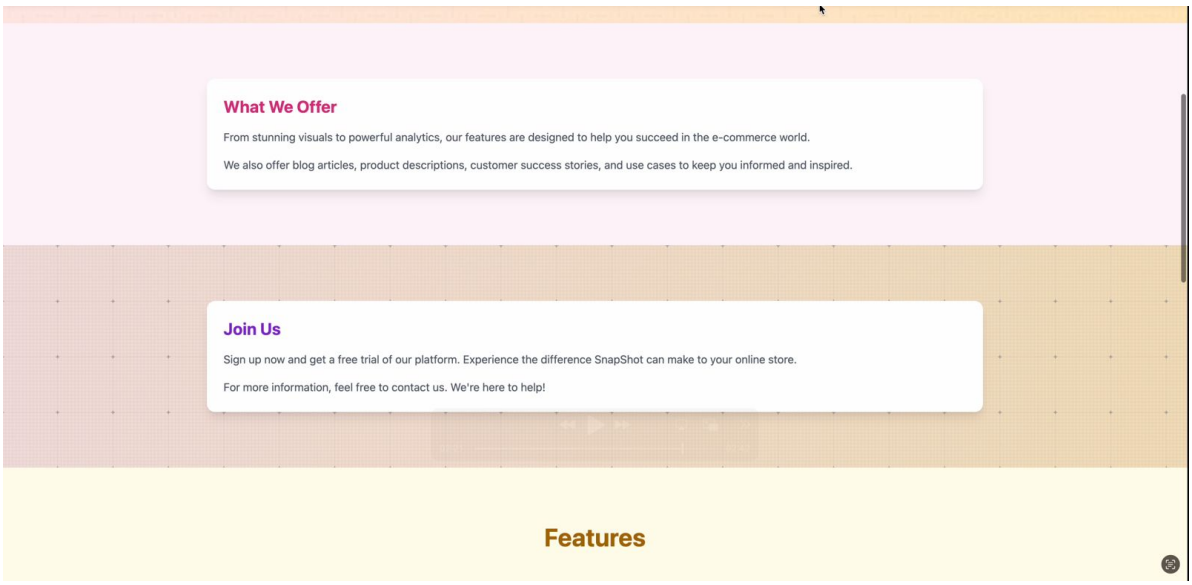
Gpt-4 model output for input parameters



Original image by Team 2

Figure A13

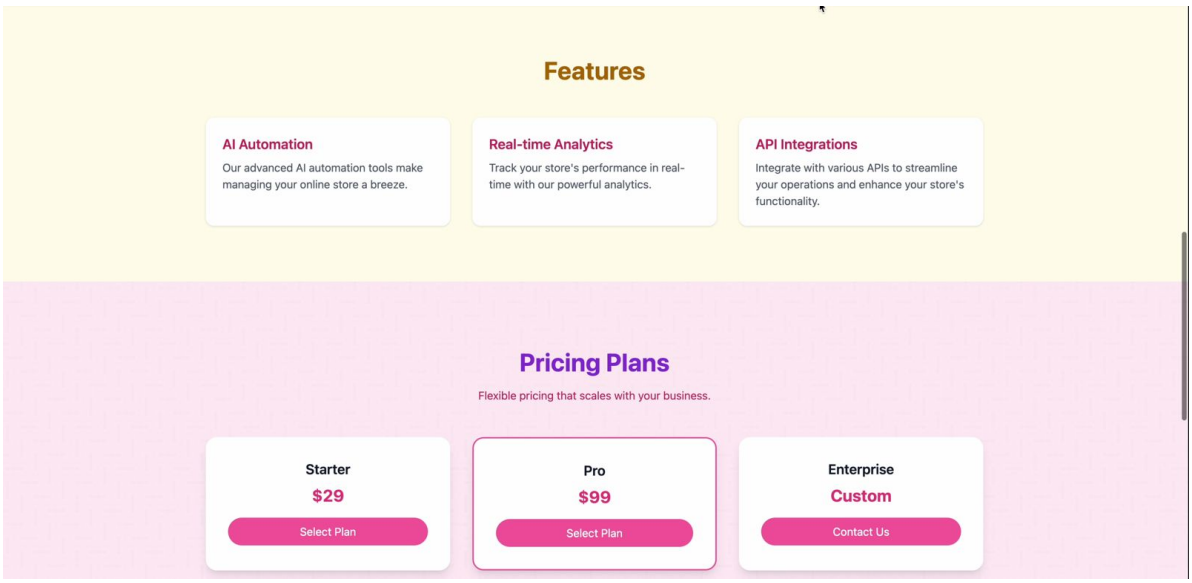
Gpt-4 model output for input parameters



Original image by Team 2

Figure A14

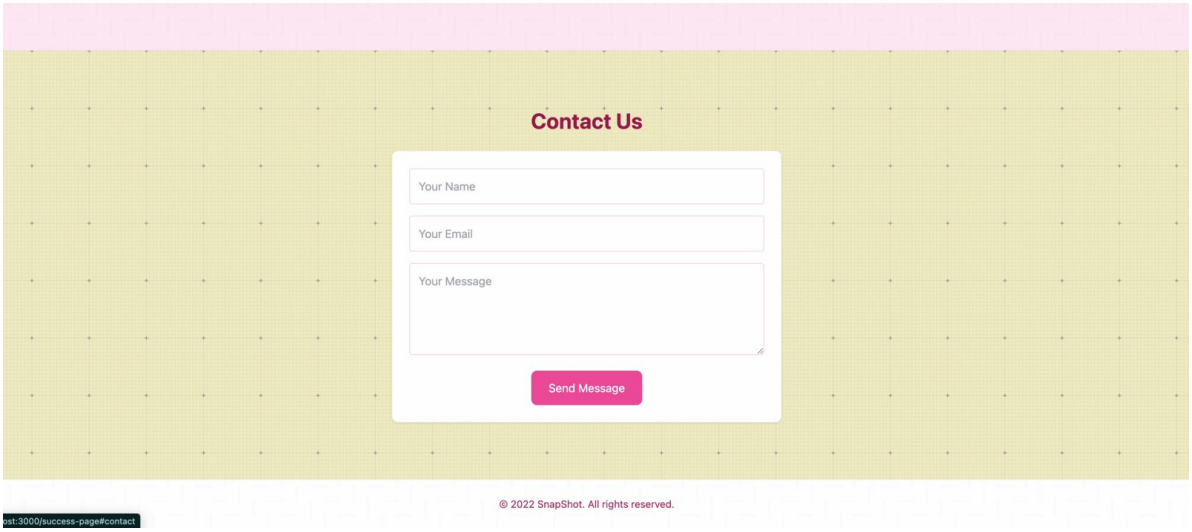
Gpt-4 model output for input parameters



Original image by Team 2

Figure A15

Gpt-4 model output for input parameters



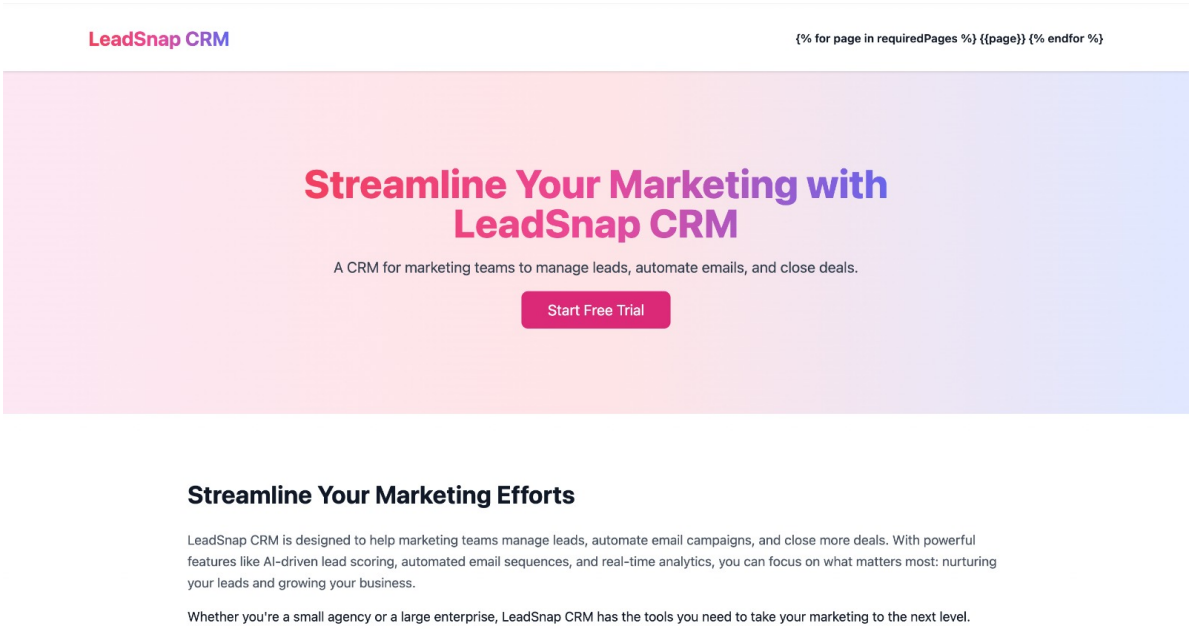
Original image by Team 2

The website output in A12,A13,A14,A15 from GPT-4 demonstrates how AI technology effectively designs and optimizes business-to-business software as a service domains. GPT-4

created a complete responsive website design using few input instructions designed specifically for an e-commerce SaaS platform. The GPT-4 design output contains all crucial webpage elements, which combine dynamic hero banners with value propositions followed by feature display along with price-levels and a working contact form. The generated content operates within specified contexts enabling it to showcase essential advantages that the platform offers consisting of AI automation alongside real-time analytics and API integrations. The UX/UI following principles allows the layout to move directly to deployment with minimum additional modifications.

Figure A16

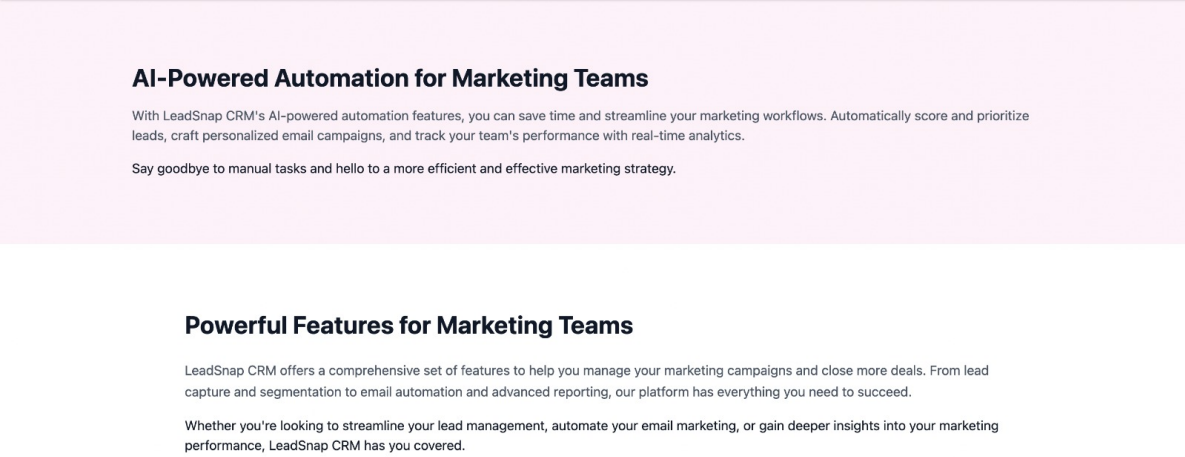
Claude(Haiku) model output for input parameters



Original image by Team 2

Figure A17

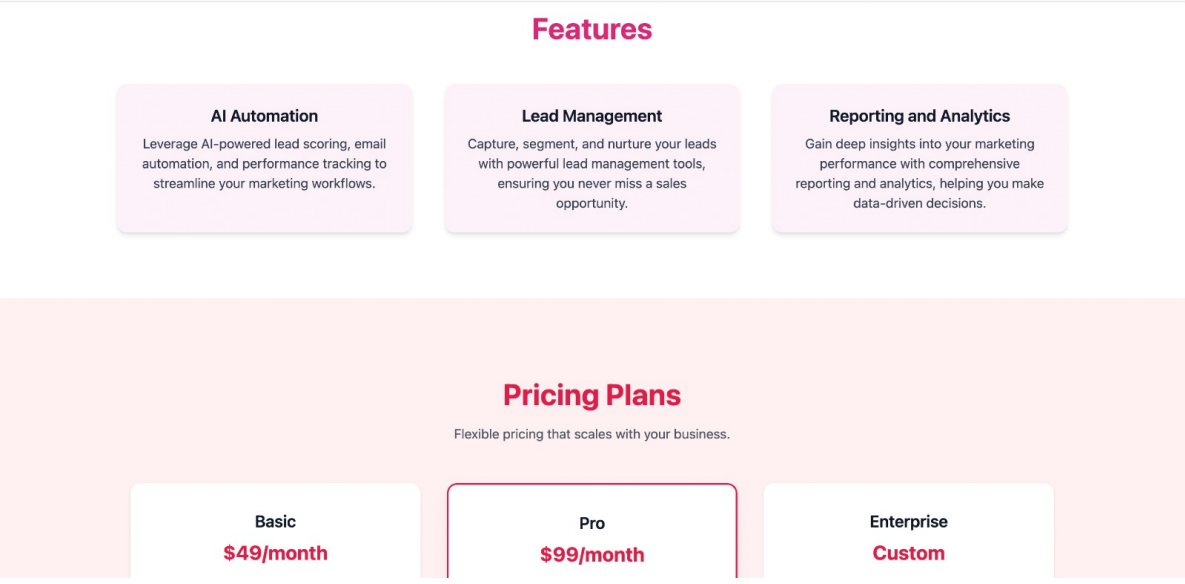
Claude(Haiku) model output for input parameters



Original image by Team 2

Figure A18

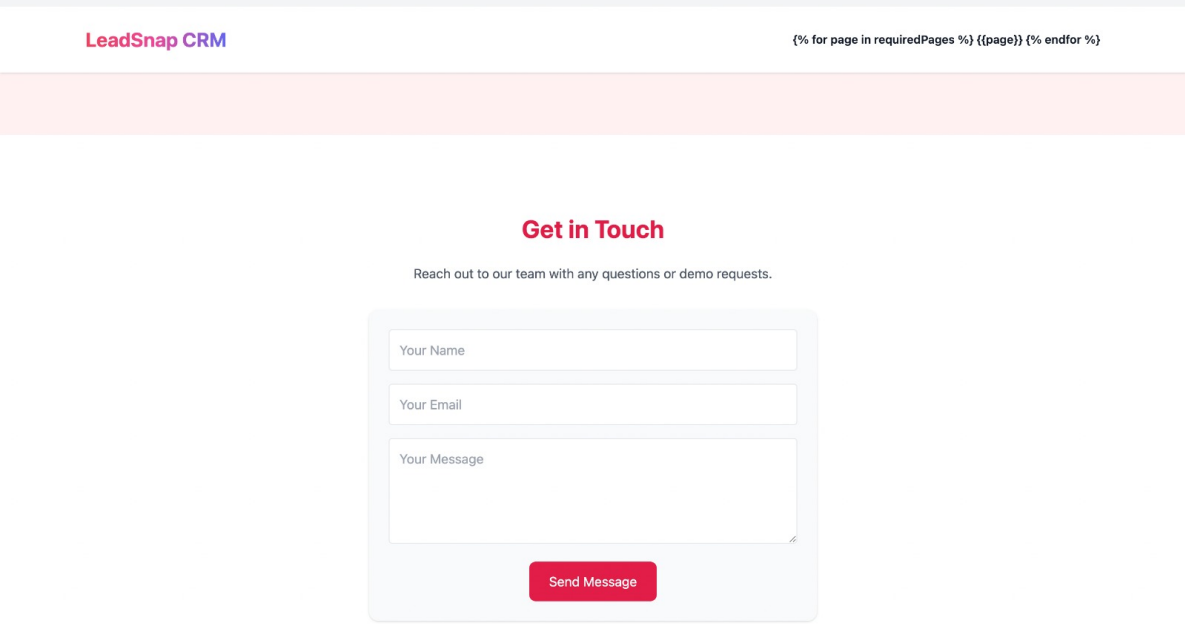
Claude(Haiku) model output for input parameters



Original image by Team 2

Figure A19

Claude(Haiku) model output for input parameters

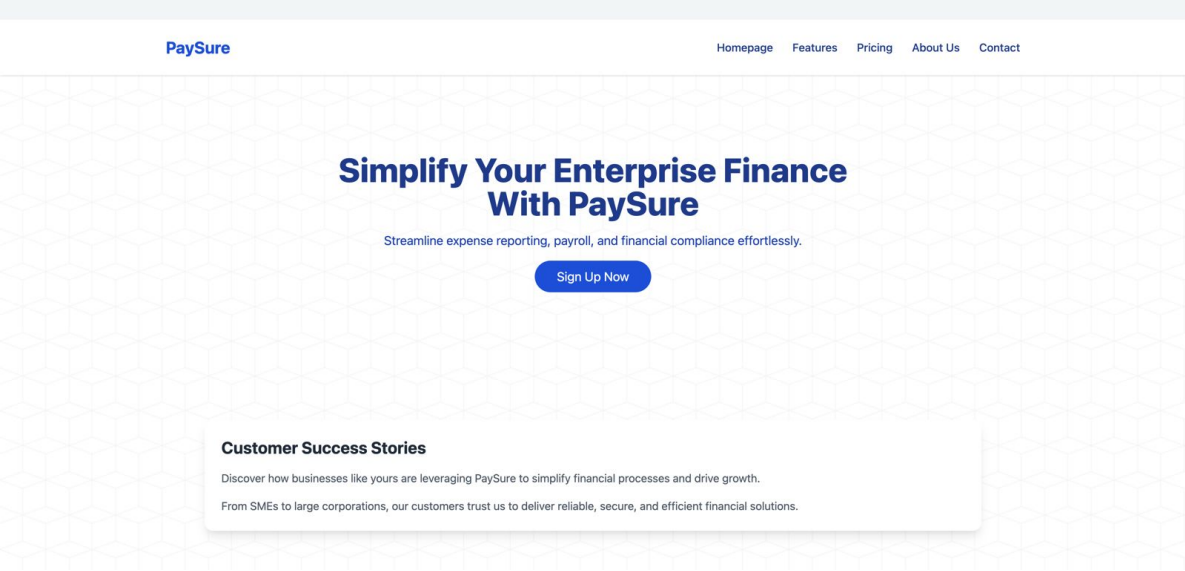


Original image by Team 2

In the above images A16,A17,A18,A19 Haiku model generated the LeadSnap CRM website that adopted a clear organizational structure and visual cohesion for marketing team use. Through its homepage design LeadSnap CRM displays its core features which focus on automatic lead management and simplified email operations. A consistent flow of information ran throughout the site while presenting clear user engagement instructions containing “Start Free Trial” and “Get in Touch”. Professional content describing AI-Powered Automation and Marketing Features combined with Tiered Pricing Plans appeared in the sections specifically designed for CRM users. The strength of Claude Haiku becomes evident through its ability to generate specific content blocks for the SaaS industry while using human terms. The user interface presents an organized structure that directs conversions. The capability of the model to generate web content for quick business prototyping emerges clearly from this output and reduces the necessity for post-production changes.

Figure A20

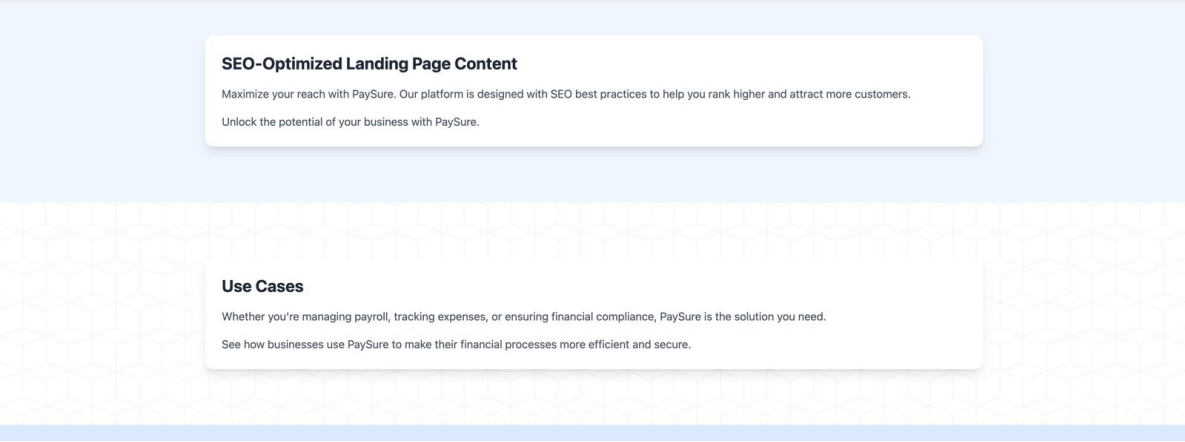
Gemini model output for input parameters



Original image by Team 2

Figure A21

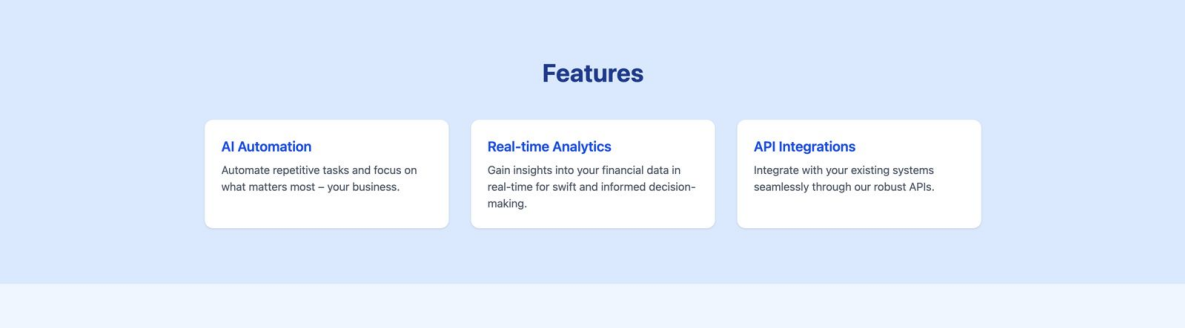
Gemini model output for input parameters



Original image by Team 2

Figure A22

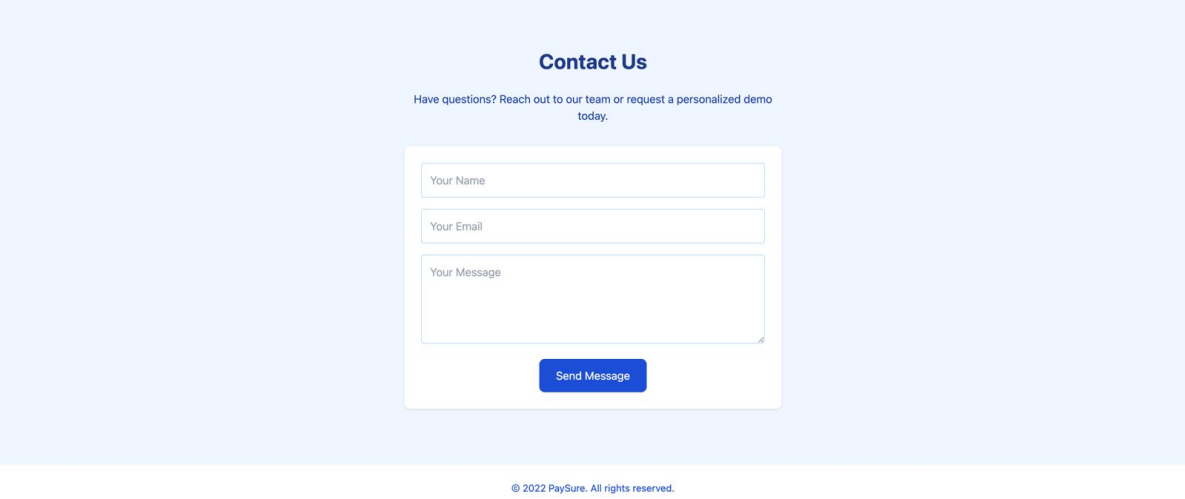
Gemini model output for input parameters



Original image by Team 2

Figure A23

Gemini model output for input parameters



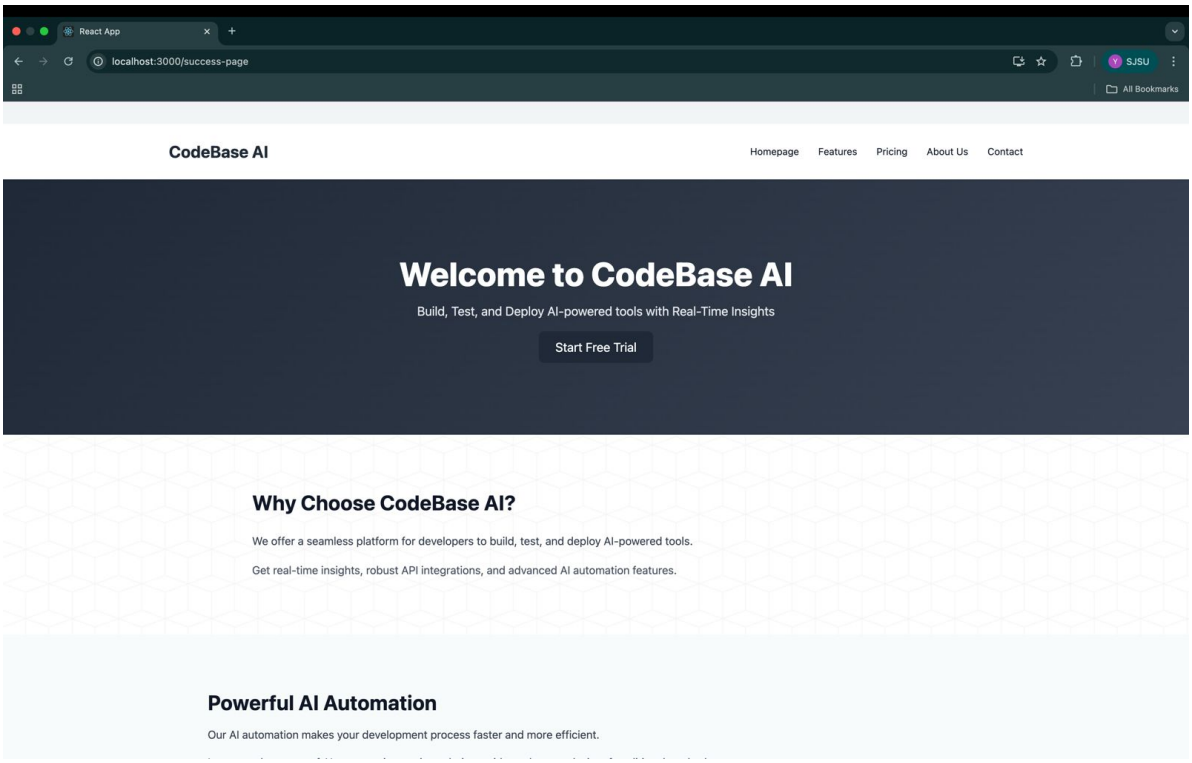
Original image by Team 2

PaySure in A20,A21,A22,A23 represents a highly structured enterprise-level financial platform which Gemini generated. Straight from the homepage PaySure delivers its purpose of automation-based enterprise financial management through an emphatic headline which guides prospective users toward immediate sign-up. The site displays a business-friendly structure that begins with value propositions followed by information about AI Automation and Real-time Analytics as well as API Integrations which are explained with practical examples. Gemini

harnesses SEO-friendly pagina content with practical examples that simultaneously boosts site discoverability and enhances user trust and application understanding. A well-designed format that incorporates modern typography with blue-toned accents and abundant white space creates the appropriate businesslike shade that financial institutions require. The Customer Success Stories section demonstrates credibility through its display of trust as well as scalability effectiveness for SMEs and large enterprises. Gemini demonstrates expertise in structured content and conversion-focused design with a professional brand approach thus establishing itself as an attractive solution to produce industry-oriented SaaS websites.

Figure A24

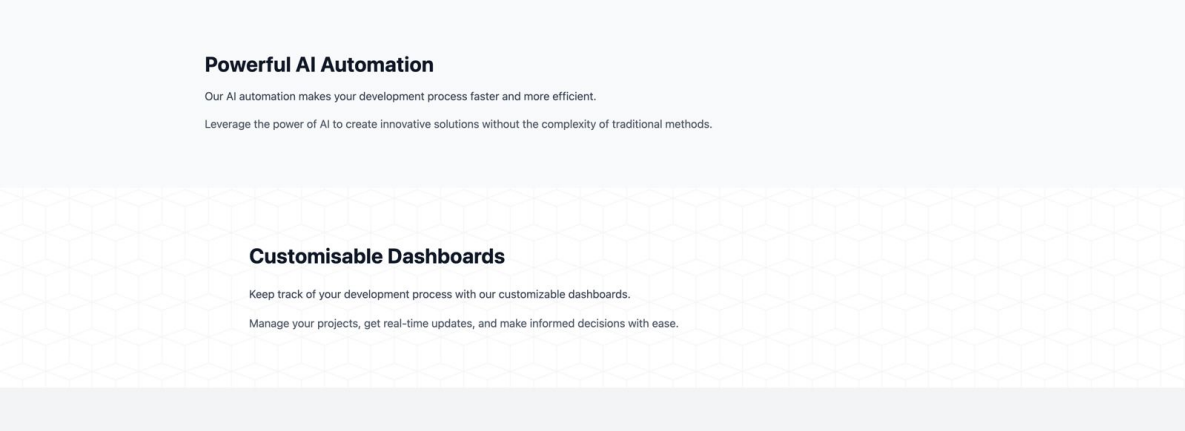
Claude-2 model output for input parameters



Original image by Team 2

Figure A25

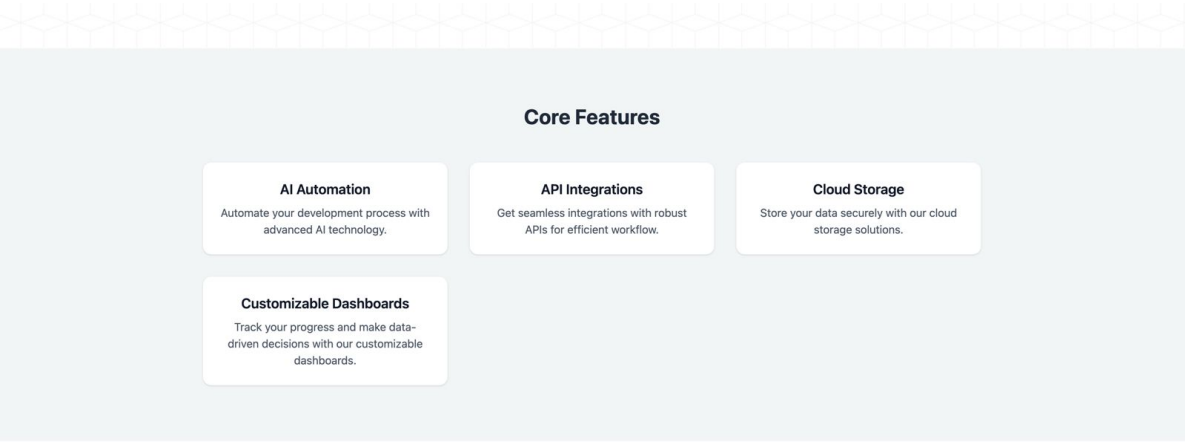
Claude-2 model output for input parameters



Original image by Team 2

Figure A26

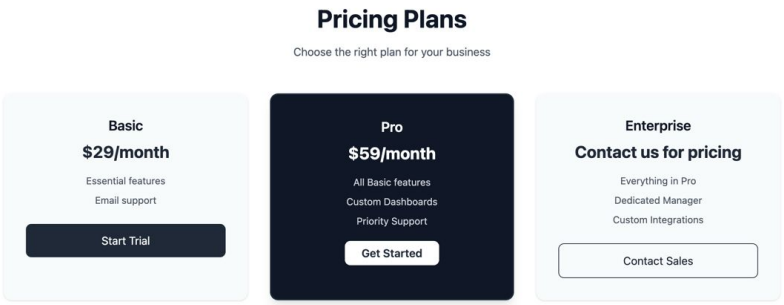
Claude-2 model output for input parameters



Original image by Team 2

Figure A27

Claude-2 model output for input parameters



Original image by Team 2

Through the Claude 2 model in A24 ,A25 , A26 , A27 CodeBase AI was created as a professional website which provides developers with tools for constructing testing deploying AI applications. The homepage employs an attractive dark-themed hero section with a bold design that easily draws focus without sacrificing its clean and uncluttered style. The website uses language specifically designed for technical users to promote CodeBase AI as a productivity solution powered by AI automation and strong APIs while emphasizing live time analytics. The page content divides itself into three main sections which focus on Customizable Dashboards and Cloud Storage and feature breakdowns designed with unified business logic. This documentation presents Pricing Plans as a clear list that accommodates different scale needs between startups and enterprise-level organizations. User-friendly interface patterns and clear designs demonstrate an implementation of SaaS UX best practices. The output generated by Claude 2 maintained a business-oriented style that enabled effective communication for both CEO-level and CTO-level professionals. Claude 2 effectively creates web content that combines strong technical value with easy navigation structures for prospective customers.

Appendix B

Project Data Source and Management Store

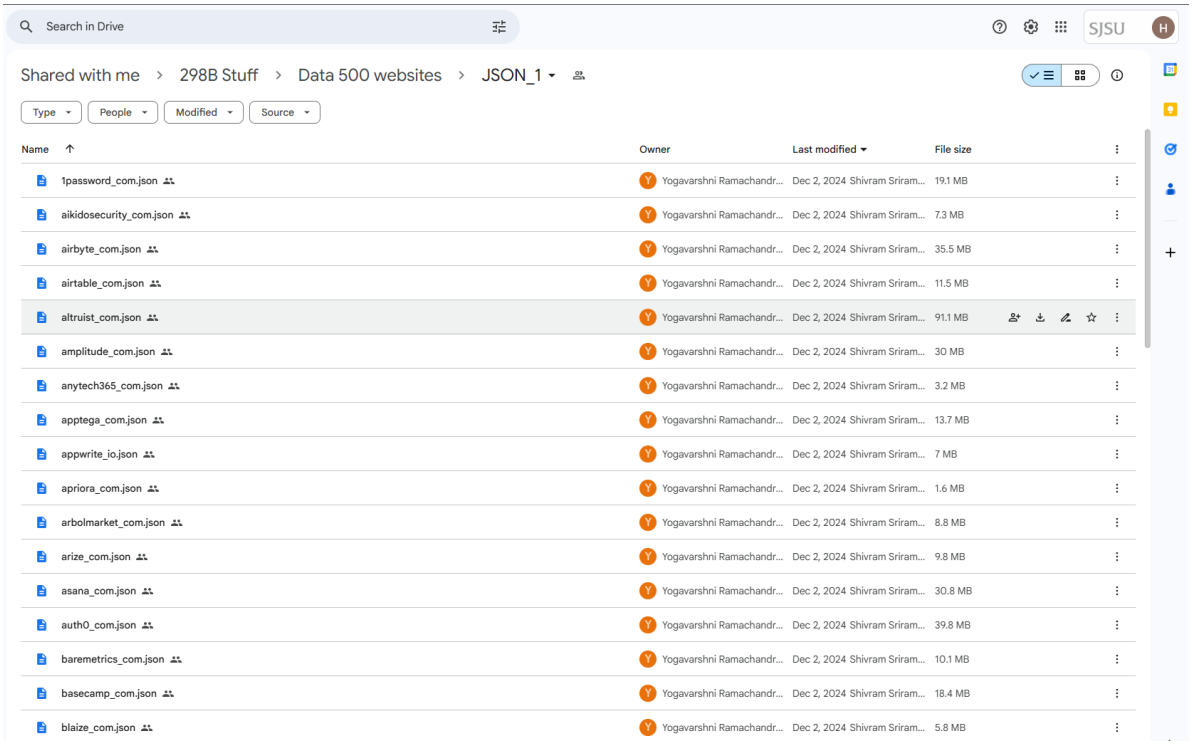
Dataset and Templates for B2B SaaS Websites

The dataset used for this project can be found in the following Google Drive folder, as shown in B1:

`https://drive.google.com/drive/folders/1V3f4S_klL3-wupHzH_UdHEyX0vCRzHwQ`

Figure B1

Dataset for 500 B2B SaaS websites



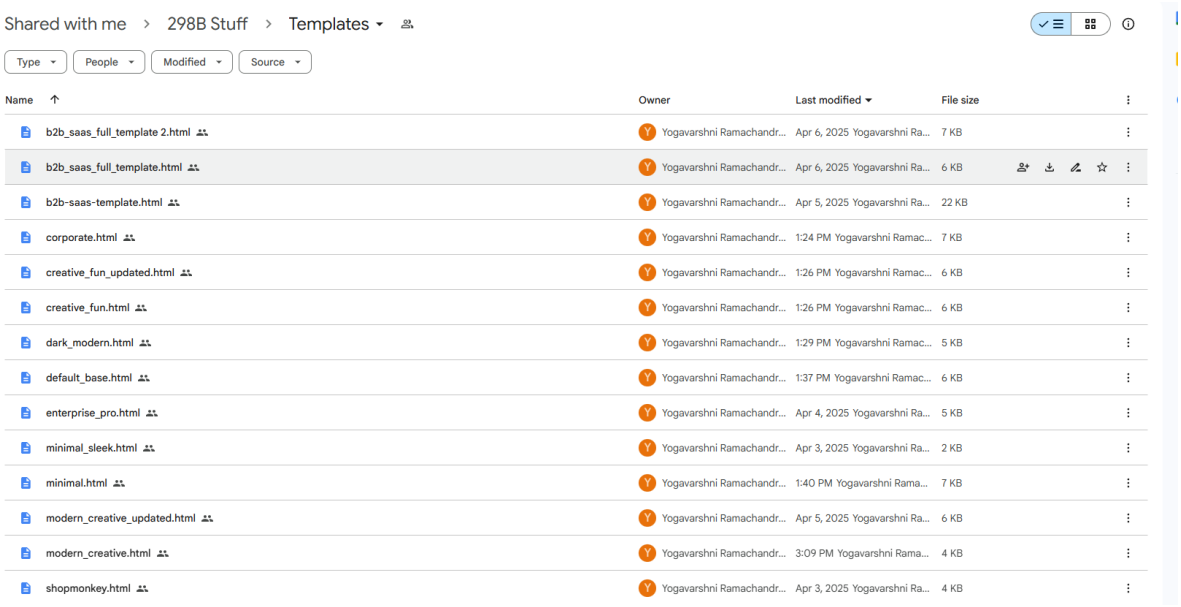
Original image by Team 2

The templates extracted from these 500 B2B SaaS websites are available in the Google Drive folder as shown in B2:

`https://drive.google.com/drive/folders/1Ay6T0J0gaVEVtTNxllF01Sr4WBxj100u`

Figure B2

Templates extracted from 500 B2B SaaS websites



Original image by Team 2

Appendix C

Project Source Code, Presentation, and Demonstration

Project Source Code

The source code and fine-tuning details for the project can be accessed via the following Google Drive link:

https://drive.google.com/drive/u/4/folders/1AKodq9R3Dx2vJ1Ab1QECqwEh0q_mk-oW

PowerPoint Presentation

The PowerPoint presentation associated with the project can be found here:

https://drive.google.com/drive/u/4/folders/1FM1C3ApB95j0So3wsarfuc7SXT2M78AQ

Demonstration

The demonstration files for the project are available at the following Google Drive link:

https://drive.google.com/file/d/1d3Krr66HotjzaerdoK8nuw0ZY7oIpRbX/view?usp=drive_link

References

Agarwal, M., Goswami, A., & Sharma, P. (2023). Evaluating chatgpt-3.5 and claude-2 in answering and explaining conceptual medical physiology multiple-choice questions. *Cureus*, *15*(9), e46222. doi: 10.7759/cureus.46222

Anthropic. (2024). *Introducing the claude 3 model family.* `https://www.anthropic.com/news/claude-3-family`. (Accessed April 2025)

Anunphop, P., & Chongstitvatana, P. (2022). Web components template generation from web screenshot. In *Proceedings of the [conference name].* Thailand.

Askell, A., Bai, Y., Chen, A., et al. (2021). A general language assistant as a challenge for alignment. *arXiv preprint arXiv:2112.00861*.

Bai, Y., Jones, A., Ndousse, K., et al. (2022). Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Bowman, S. R., & Dahl, M. (2022). Measuring social biases in language models. *Proceedings of the International Conference on Machine Learning*.

Cao, Y., Li, S., Liu, Y., Yan, Z., Dai, Y., Yu, P. S., & Sun, L. (2018). A comprehensive survey of ai-generated content (aigc): A history of generative ai from gan to chatgpt. *Journal of the ACM*, *37*(4), Article 111.

Castillo-Campos, M., Varona-Aramburu, D., & Becerra-Alonso, D. (2024). Artificial intelligence tools and bias in journalism-related content generation: Comparison between chat gpt-3.5, gpt-4 and bing. *Tripodos*, *55*. (OnlineFirst)

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., . . . Gehrmann, S. (2023). Palm: scaling language modeling with pathways. *The Journal of Machine Learning Research*, *24*(1), 11324–11436.

Cotterell, R., & Sap, M. (2022). Ethics in language model development: A review and future directions. *Annual Review of Linguistics*, *8*, 101–120.

D, Y., Sneha, & Kumar, N. (2022). Html code generation from website images and sketches using deep learning-based encoder-decoder model. In *2022 ieee 4th international conference on*

cybernetics, cognition and machine learning applications. Mysuru, India: IEEE. doi: 10.1109/ICCCMLA.2022.2022.2022.2022.9029288

Dhyani, P., Nautiyal, S., Negi, A., Dhyani, S., & Chaudhary, M. P. (2024). Automated api docs generator using generative ai. In *2024 ieee international students' conference on electrical, electronics and computer science.* Dehradun, India: IEEE. doi: 10.1109/IEEEISC.2024.148219

Dosovitskiy, A., Beyer, L., Kolesnikov, A., et al. (2021). Scaling vision transformers. *arXiv preprint arXiv:2106.04560*.

Eloundou, T., Manning, S., Mishkin, P., & Rock, D. (2023). Gpts are gpts: An early look at the labor market impact potential of large language models. *Working Paper*. Retrieved from `https://arxiv.org/abs/2303.10130v5`

Ersoy, P., & Erşahin, M. (2024). Benchmarking llama 3 70b for code generation: A comprehensive evaluation. *Orclever Proceedings of Research and Development*, *4*(1), 52-58. doi: 10.56038/oprd.v4i1.444

Fausti, F. D., Pugliese, F., & Zardetto, D. (2020). Towards automated website classification by deep learning. *Rivista di Statistica Ufficiale*, *3*, 9-26.

Geraldi, J., & Lechter, T. (2012). Gantt charts revisited: A critical analysis of its roots and implications to the management of projects today. *International Journal of Managing Projects in Business*, *5*. doi: 10.1108/17538371211268889

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672–2680).

Google Cloud. (2023). *Vertex AI: Managed Machine Learning Platform.* Retrieved from `https://cloud.google.com/vertex-ai` ([Accessed: Feb. 25, 2025])

Gozalo-Brizuela, R., & Garrido-Merchán, E. C. (2023). *A survey of generative ai applications.* (Unpublished manuscript)

Han, Y., Liu, C., & Wang, P. (2024). A comprehensive survey on vector database: Storage and

retrieval technique, challenge. *Preprint or Journal Name (update this)*. Retrieved from AddtheDOIorofficialURLhere

Hughes, R. T., Zhu, L., & Bednarz, T. (2021). Generative adversarial networks–enabled human–artificial intelligence collaborative applications for creative and design industries: A systematic review of current approaches and trends. *Frontiers in Artificial Intelligence*, *4*. doi: 10.3389/frai.2021.604234

Jeong, C. (2024). *A study on the implementation of generative ai services using an enterprise data-based llm application architecture* (Technical Report). Seoul, Korea: Samsung SDS.

Jia, C., Yang, Y., Xia, Y., et al. (2021). Deep multimodal models. *arXiv preprint arXiv:2102.05918*.

Kanakia, H. T., & Nair, S. P. (2023). Designing a user-friendly and responsive ai based image generation website and performing diversity assessment of the generated images. In *Proceedings of the fourth international conference on electronics and sustainable communication systems.* Mumbai, India: IEEE. doi: 10.1109/ICESCS67686.2023.10193269

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. In *Proceedings of the 2nd international conference on learning representations (iclr).*

Kuswanto, W., Nolan, G., & Lu, G. (2023, Jan). Highly multiplexed spatial profiling with codex: bioinformatic analysis and application in human disease. *Seminars in Immunopathology*, *45*(1), 145–157. (Epub 2022 Nov 21) doi: 10.1007/s00281-022-00974-0

Liu, Y., Han, T., Ma, S., Zhang, J., Yang, Y., Tian, J., . . . Ge, B. (2023). Summary of chatgpt/gpt-4 research and perspective towards the future of large language models. *arXiv preprint arXiv:2304.01852v2*. Retrieved from https://arxiv.org/abs/2304.01852v2

Muthazhagu, V. H., & B, S. (2024). Exploring the role of ai in web design and development: A voyage through automated code generation. In *Proceedings of the 2024 ieee international conference on web technologies and engineering.* Karaikal, India: IEEE. doi: 10.1109/ICWTCE.2024.14067409

Ouyang, L., Wu, J., Jiang, X., et al. (2022). Aligning language models to follow instructions. *arXiv preprint arXiv:2203.02155*.

Perez, E., McKenzie, S., & Song, D. (2022). Discovering language model behaviors with model organism objective generation. *arXiv preprint arXiv:2204.03468*.

Plotnikova, V., Dumas, M., & Milani, F. P. (2022). Applying the crisp-dm data mining process in the financial services industry: Elicitation of adaptation requirements. *Data & Knowledge Engineering*, *139*, 102013.

Priyanshu, A., Maurya, Y., & Hong, Z. (2024). Ai governance and accountability: An analysis of anthropic's claude. *ArXiv*, *2407.01557v1*. (School of Computer Science, Carnegie Mellon University)

Radford, A., Kim, J. W., Hallacy, C., et al. (2021). Learning transferable visual models from natural language supervision (clip). *arXiv preprint arXiv:2103.00020*.

Ramesh, A., Pavlov, M., Goh, G., et al. (2021). Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*.

Roth, R. E. (2017, April). User interface and user experience (ui/ux) design. *Geographic Information Science & Technology Body of Knowledge*, *2017*(Q2). doi: 10.22224/gistbok/2017.2.5

Sadat, H., & Ghorbani, A. A. (2023). *Automated web page synthesis in adaptive web systems.*

Schröer, C., Kruse, F., & Gómez, J. M. (2021). A systematic literature review on applying crisp-dm process model. In *Centeris - international conference on enterprise information systems / projman - international conference on project management / hcist - international conference on health and social care information systems and technologies 2020* (Vol. 181, pp. 526–534). Elsevier. doi: 10.1016/j.procs.2021.01.226

Shrivastav, R., Shahane, S., Hydri, T. S., Akre, M. V., & Amin, Z. D. (2024). Exploring potential of gemini with ai based content generator. *International Journal of Research in Computer & Information Technology*, *2*(1), 68-72. doi: 10.5281/zenodo.11207604

Verma, A. A., Kurupudi, D., & S, S. (2024). Bloggen: A blog generation application using

llama-2. In *2024 international conference on advances in data engineering and intelligent computing systems (adics)*. Chennai, India: IEEE. doi: 10.1109/ADICS.2024.3503482

Yenduri, G., et al. (2023). Generative pre-trained transformer: A comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions. *arXiv preprint arXiv:2305.10435*. Retrieved from `https://arxiv.org/abs/2305.10435`

Yeom, J., Lee, H., Byun, H., Kim, Y., Byun, J., Choi, Y., ... Song, K. (2024). Tc-llama 2: Fine-tuning llm for technology and commercialization applications. *Journal of Big Data*, *11*, 100. doi: 10.1186/s40537-024-00963-0

Zhao, P., Zhang, H., Yu, Q., Wang, Z., Geng, Y., Fu, F., ... Cui, B. (2023). Retrieval-augmented generation for ai-generated content: A survey. *Journal of AI Research*. (Penghao Zhao, Hailin Zhang, and Qinhan Yu contributed equally to this paper.)