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 298A: MSDA Project

Dr. Simon Shim & Dr. Lee C. Chang

Dec 09, 2024

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: Generative Llama 2 (Large Language Model Meta AI (version 2)), GPT-4 (Generative Pre-trained Transformer 4), PaLM (Pathways Language Model) , and Gemini. Streamlit will collect user inputs regarding industry type and design preferences to store in AWS to generate website templates and customized content. Models will be fine-tuned and tested to ensure high-quality, professional multi-page websites per user specifications.

The model will be able to generate a complete, functional, multi-page 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 include Fréchet 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, multipage 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 Llama 2, GPT-4, PaLM, Gemini and Claude model 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. Llama 2 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 organisations that want an effective powerful online presence such that is through web development at low cost thus broadening the option of web development opportunities across 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, Llama 2, PaLM or 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).

Structured Content Management with Llama 2 : It is concerned more with proper management of well-defined content, which can comprise captions, tags, and labels. Due to its effectiveness in managing organized information, it makes the architectural design of the site more SEO-receptive, and ergonomic making the site more accessible Verma et al. (2024).

Structured Content Management with PaLM : It performs very well when it comes to subsequent text, syntactically coherent with the given context and in multiple languages, and this is especially useful when it comes closer to content translation for multilingual websites. Also, PaLM can be employed for the generation of microcopy, which involves composing buttons in call-to-action and error messages, for the purposes of improving the graphical interface and, hence, user experience Chowdhery et al. (2023). Although PaLM itself does not produce the website layout (HTML/CSS), it collaborates with other models, such as Codex, for which it is critical in the content generation aspect of Web development Chowdhery et al. (2023).

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).

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.	09/15/24
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 prompt-completion pairs.	10/07/24
Data Loading	Adding the preprocessed data into an S3 bucket for modeling.	10/10/24
Model Development	Developing NLP models with GPT 4, RAG, Llama2, and Claude.	12/07/24
NLP User Input and Front-End Creation	Developing user interface in Streamlit for real-time website generation.	12/07/24
Model Evaluation and Deployment	Evaluating models based on Fréchet Inception Distance, Google Lighthouse Score, BLEU, and ROUGE score.	12/07/24
Final Report	Project documentation in APA format containing a description of each phase, results, and recommendations.	12/10/24
Presentation	Project presentation highlighting the preprocessing, pipeline, and modelling with a focus on the results.	12/10/24

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.

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. The main problem the team intends to solve is making costly web development accessible and affordable for non-technical users. The proposed solution is an AI-powered tool that generates fully functional websites based on user inputs. The proposal will cover the project's purpose, expected outcomes, and the technical approach, including model types and the use of AWS for storage and Streamlit for the user interface. It will also give information about how the system will empower small businesses, freelancers, and startups as discussed.

Due date: 09/04/24

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.

Due date: 09/13/24

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.

Due date: 09/04/24

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.

Due date: 10/04/24

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. Also, the prompt-completion pairs are established by defining the input (prompt) as certain website characteristics or features and the output (completion) as the committed HTML, CSS, and JavaScript formatting structures. This makes it easier for the data to be format to achieve the data format required in model usage while embracing diverse use at the same time.

Due date: 10/07/24

Data Loading

The preprocessed data, out of which are the JSON files and the pairs of the completion of the prompts, 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.

Due date: 10/10/24

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.

Due date: 12/07/24

NLP User Input and Front-End Creation

The Streamlit 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.

Due date: 12/07/24

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.

Due date: 12/07/24

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.

Due date: 12/09/24

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 Streamlit.
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

```
app.js
Users > yogavarshniramachandran > Documents > app.js > templatePage
const cheerio = require("cheerio");
const axios = require("axios");
const path = require('path');
const puppeteer = require('puppeteer');
const robotsParser = require('robots-parser');
const fs = require("fs");


async function templatePage(url){
    try {
      let updatedUrl = new URL(url);
      updatedUrl.search = "";
      let scrapeURL = updatedUrl.href;
      scrapeURL = scrapeURL.replace(/\/+$/, "");
      let data;
      try{
        const response = await axios.get(scrapeURL);
        data = response.data
      }
      catch(err){
        console.log("error", err)
      }

      const $ = cheerio.load(data);

      const isNextJs = $('script[src*="next"]').length > 0;
      if (isNextJs) {
        console.log("next js")
      }


      const browser = await puppeteer.launch({timeout: 60000});
      const page = await browser.newPage();
      page.on('requestfailed', () => {});
      await page.goto(scrapeURL, { waitUntil: "networkidle2", timeout: 60000 });

      const links = await page.evaluate((scrapeURL) => {
        const anchors = Array.from(document.querySelectorAll("a"));
```

Figure 2

Data Scraping using B2B website URL

```
        fs.writeFileSync(combinedJsFilePath, combinedJsContent);
        console.log(`Combined JS saved as ${combinedJsFilePath}`)
        return combinedJsFileName;
      }
      return usePuppeteer
  }


  (async function runScraping(){
    const urls = [
        "https://www.amberflo.io/",
        "https://www.vendr.com",
        "https://www.swan.io/",

    ];

    try {
        await Promise.all(urls.map(url => templatePage(url)));
        console.log("Scraping completed for all URLs.");
    } catch (err) {
        console.error("Error during scraping:", err);
    }
  })();
```

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

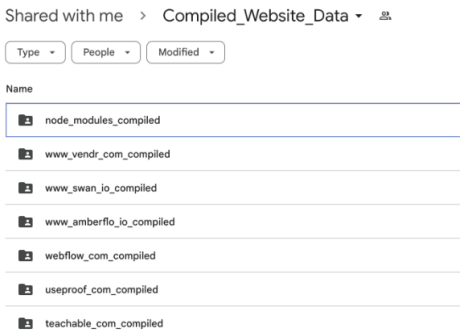


Figure 4

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}')
```

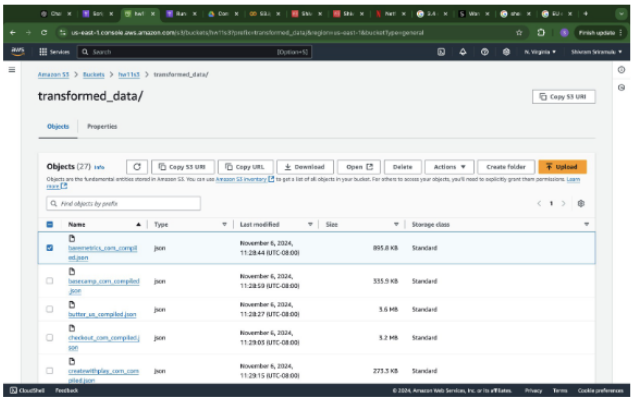
```
Successfully uploaded transformed_data/www_atlist_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/www_kissmetrics_io_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/www_flycode_com_compiled.json to S3 bucket hw11s3
Successfully uploaded transformed_data/butter_us_compiled.json 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
```

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

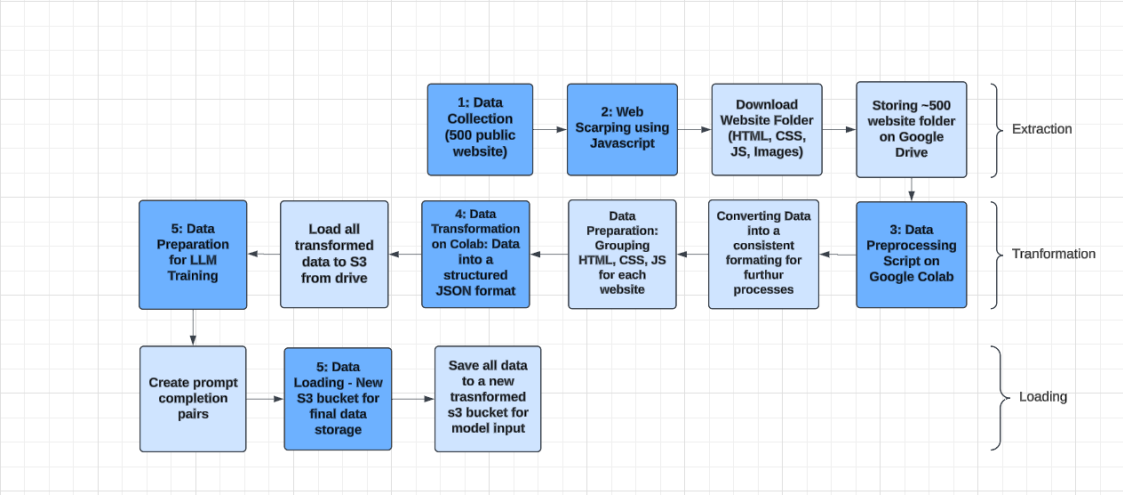


Usage Mechanisms Website Regeneration

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. The processed data is further shaped into a structured JSON format, which models the Inter-Component Relations, ICR, between different components of a website, such as links, styles or scripts. This kind of data format means that any dataset derived from a structured format is optimal for handling in a mechanistic manner, specifically in model formation.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 beasmus prompt-completion pairs 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.

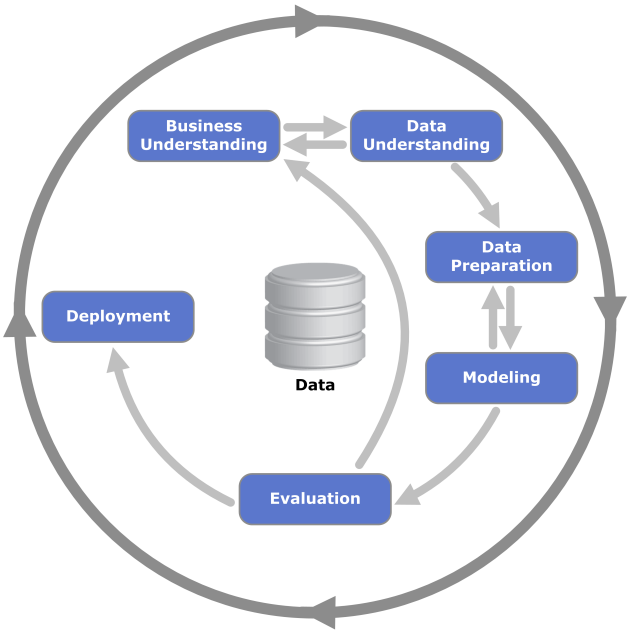
2.2 Project Development Methodology

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. Llama 2 will deal with structured content and give out standard headings, labels and metadata to avoid compromising the professional look of the resultant pages. Finally, PaLM will

improve website organization and navigation, and adjust the layouts to be responsive and easy to navigate.The used dataset which contains text scraped from the publicly available business websites will be divided into the training and testing sets for the efficiency of the models. Most of the time, the data will be divided into training data and testing data in the ratio of 70:30. This division is useful in training the models on a large data set while reserving part of the data for verifying the results as well as testing the generality of the models. The models will be trained based on the user inputs that will be received through Streamlit 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. As part of the cross-device availability check, based on available best practice's, we will ensure that the websites generated using the tools supported by this project work and look equally well on desktops, tablets, and mobile phones. This is especially important in modern context in which many users interact with multiple devices and expect the experience to be consistent Consequently, we will assess content relevancy to determine whether the generated text corresponds to users' inputs. 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 Streamlit 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, Llama-2 and Palm . 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.

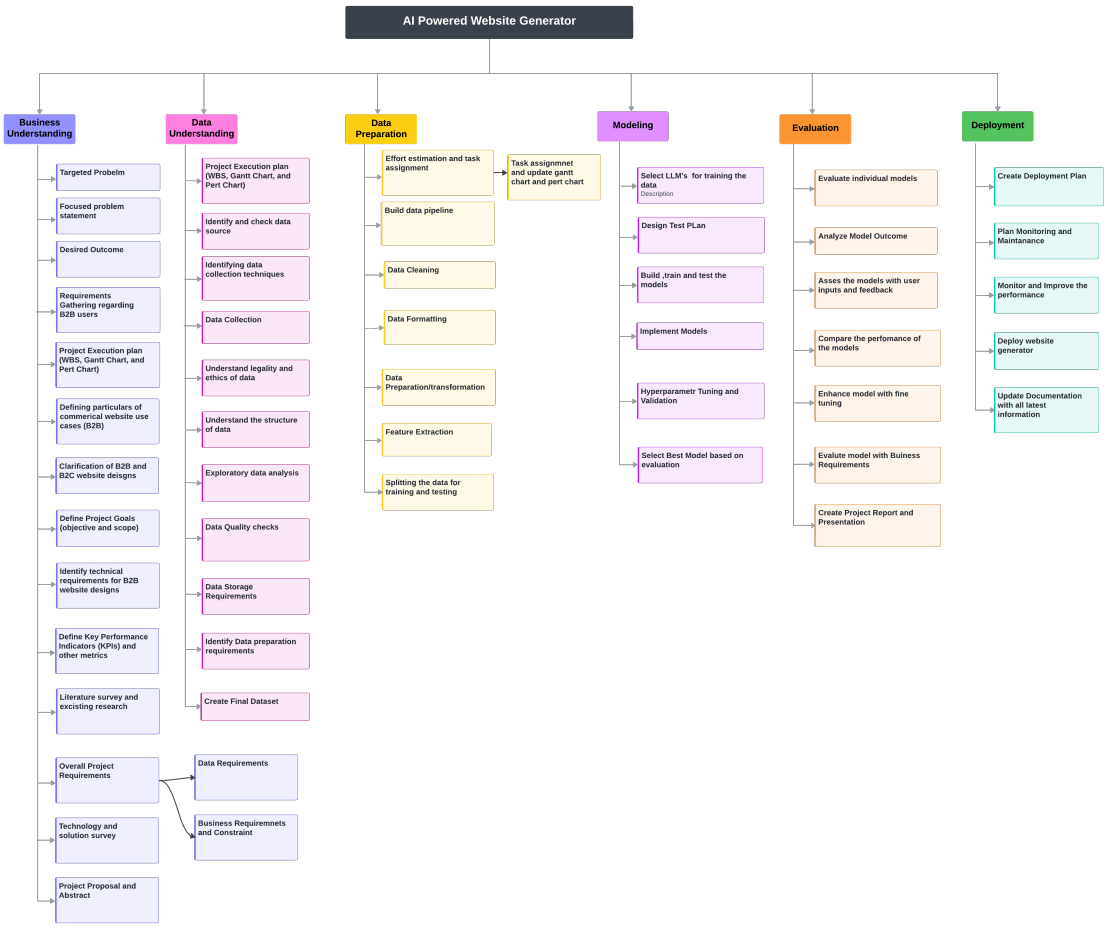
In the modeling phase of the project high level sophisticated models like GPT-4 will be used for generating dynamic content ,Gemini will be used for designing UI elements , Llama2 will be used for the structure of the webpage and PaLM will be further used for improving the navigation and ensure that 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 streamlit. 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
Meta Llama	Version 2	developing and fine-tuning the model
PaLM	Version 2.6	developing and fine-tuning the model
Claude	Version 2.0	developing and fine-tuning the model

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

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

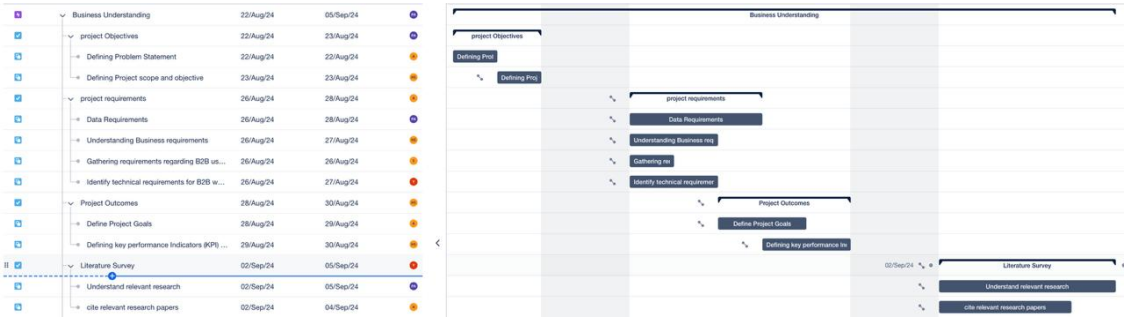
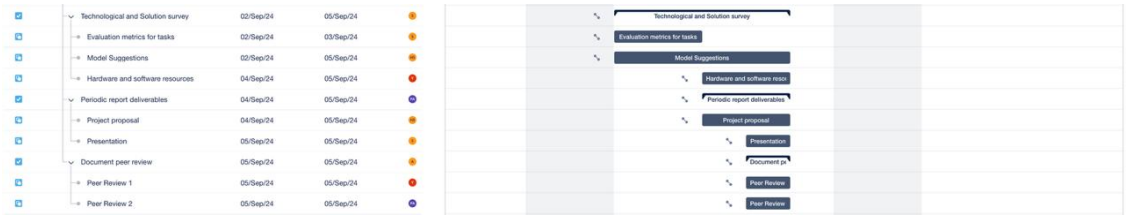


Figure 11

Business Understanding phase 2



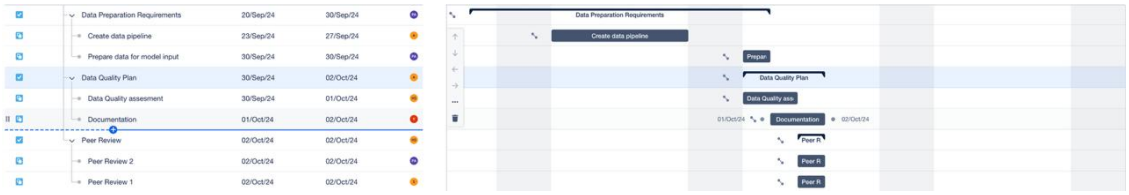
Data Understanding phase. The Figure 12 and 13 below shows the project in its progress with the Data Understanding phase being the second phase with the project starting on September 5, 2024 and ending on October 2, 2024. Under Data Exploration during this phase, the following crucial tasks are undertaken; 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. Third Phase is the Data Preparation, step being take from October 2, 2024 to November 6, 2024. This phase includes procedures like peer review of data, data scrubbing, data normalization and data partitioning into training, validation and test sets. Leading sub tasks include data transformation planning, target features prediction, and modeling requirements development, all team members are being assigned to it. Peer reviews are done at many different junctures due to the need to understand the quality of the data and its preparedness for modeling. The phase ends with the preparation of the last raw data to be used for secondary analysis and building of models. Figure 14 and 15 shoes the timeliness of data preparation phase.

Figure 14

Data Preparation phase 1

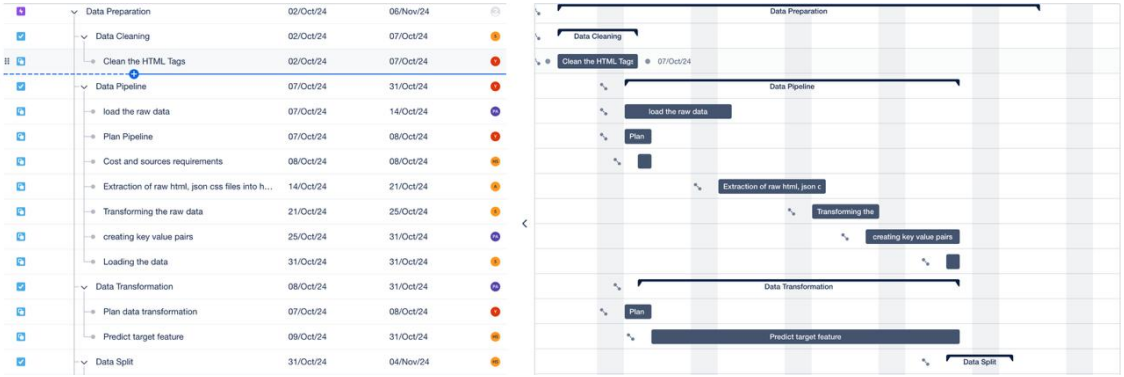


Figure 15

Data Preparation phase 2

Data Modeling phase. Figure 16 and 17 shows the Modelling 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, Llama and PaLM models, as well as the fine-tuning of hyperparameters 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 model selection is done after implementing the model by comparing with the results produced during the computation. Peer reviews are conducted to validate the models being used in this study. The phase ends by interpreting the results from the models, guaranteeing the right models are prepared for implementation.

Figure 16

Modeling phase 1

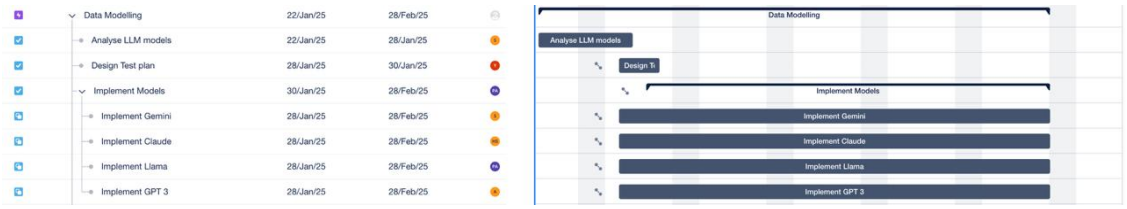
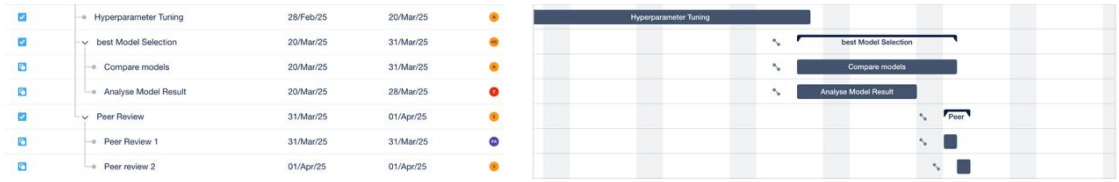


Figure 17

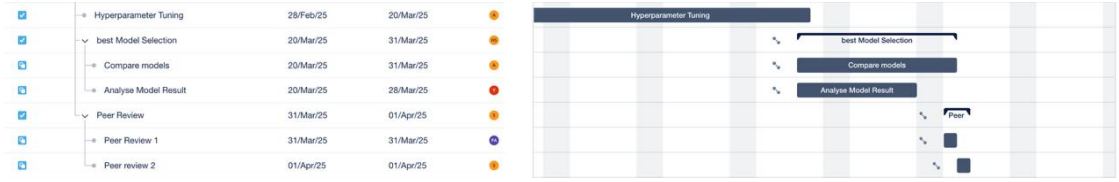
Modeling phase 2



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

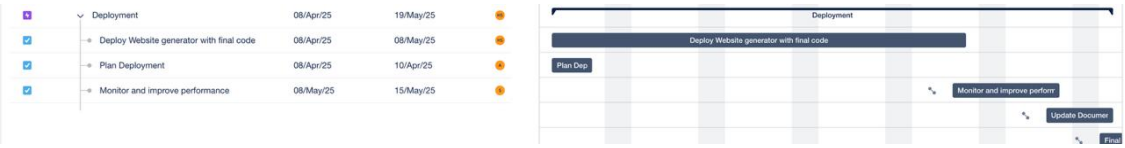
Evaluation phase



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

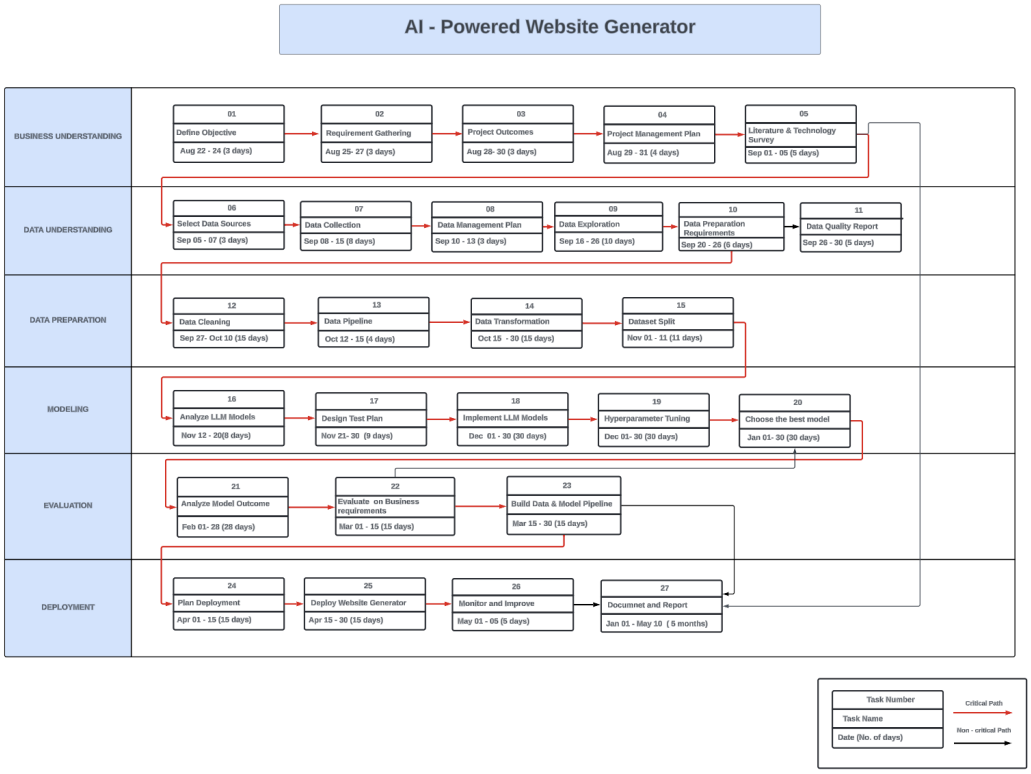
A technique known as the Program Evaluation and Review Technique (PERT) chart is widely used in project management of "AI-Powered Website Generator" project and is utilized in the form of detailed tasks and activities of projects in the form of trees enabling effective and efficient completion of the 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 Training, Validation and Test 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. This structure offered the LLM the clear format and guidelines through which the LLM can learn and replicate Multi-Page Site designs.

Prompt-Completion Pairs for LLM Training: In the training process with multipage

designs, we employ prompt-completion pairs. Each pair includes:

Prompt: Applicant-created input which refers to the nested-page web design like: “Create a clean modern B2B SaaS website with the home page, services page, pricing page, and contact page."

Completion: The JSON blueprint of a web site that includes each page of the site and its HTML, CSS, and Javascript in a nested tree.

This dataset of five hundred websites will be split into training and validation datasets . During testing, we’ll input to the model only user prompts to check its capacity to generate coherent multipage layouts on its own. This approach also looks at the context of the B2B SaaS websites to make certain that the model also considers the interconnection of the multiple interconnected pages out of the single landing page structure.

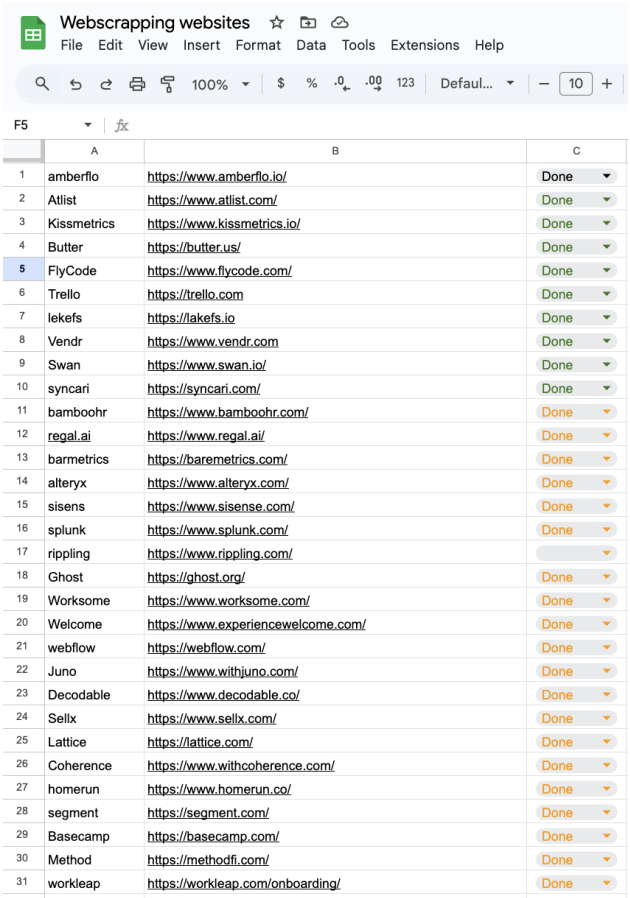
3.2. Data Collection

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



Webscrapping websites

File Edit View Insert Format Data Tools Extensions Help

	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

```
app.js
Users > yogavarshniramachandran > Documents > app.js > templatePage
const cheerio = require("cheerio");
const axios = require("axios");
const path = require('path');
const puppeteer = require('puppeteer');
const robotsParser = require('robots-parser');
const fs = require("fs");


async function templatePage(url){
    try {
      let updatedUrl = new URL(url);
      updatedUrl.search = "";
      let scrapeURL = updatedUrl.href;
      scrapeURL = scrapeURL.replace(/\/+$/, "");
      let data;
      try{
        const response = await axios.get(scrapeURL);
        data = response.data
      }
      catch(err){
        console.log("error", err)
      }

      const $ = cheerio.load(data);

      const isNextJs = $('script[src*="next"]').length > 0;
      if (isNextJs) {
        console.log("next js")
      }


      const browser = await puppeteer.launch({timeout: 60000});
      const page = await browser.newPage();
      page.on('requestfailed', () => {});
      await page.goto(scrapeURL, { waitUntil: "networkidle2", timeout: 60000 });

      const links = await page.evaluate((scrapeURL) => {
        const anchors = Array.from(document.querySelectorAll("a"));
```

Figure 23

Web Scrapping for collected website using JavaScript

```
            fs.writeFileSync(combinedJsFilePath, combinedJsContent);
            console.log(`Combined JS saved as ${combinedJsFilePath}`)
            return combinedJsFileName;
        }
        return usePuppeteer
    }


    (async function runScraping(){
      const urls = [
            "https://www.amberflo.io/",
            "https://www.vendr.com",
            "https://www.swan.io/",

      ];

      try {
            await Promise.all(urls.map(url => templatePage(url)));
            console.log("Scraping completed for all URLs.");
      } catch (err) {
            console.error("Error during scraping:", err);
      }
    })();
```

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

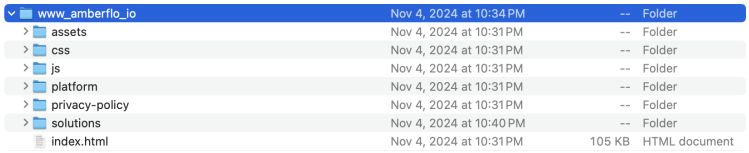
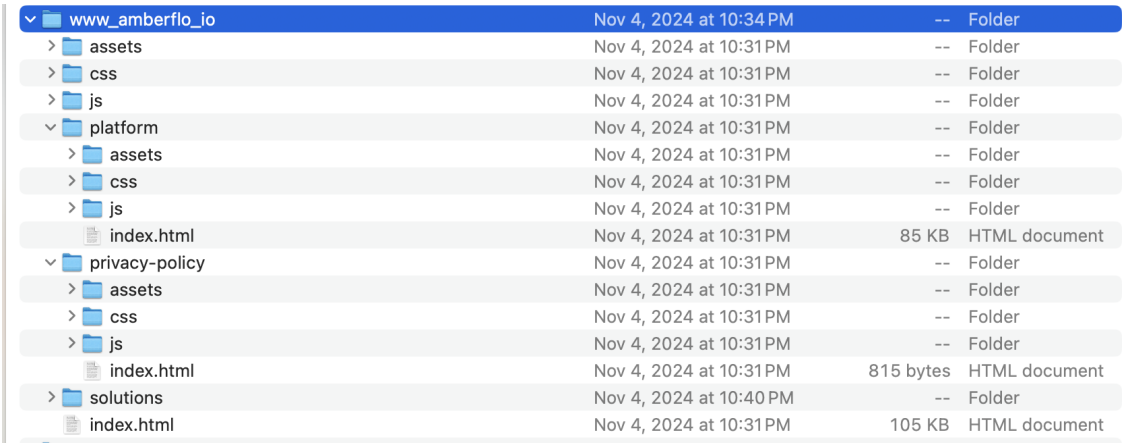


Figure 25

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

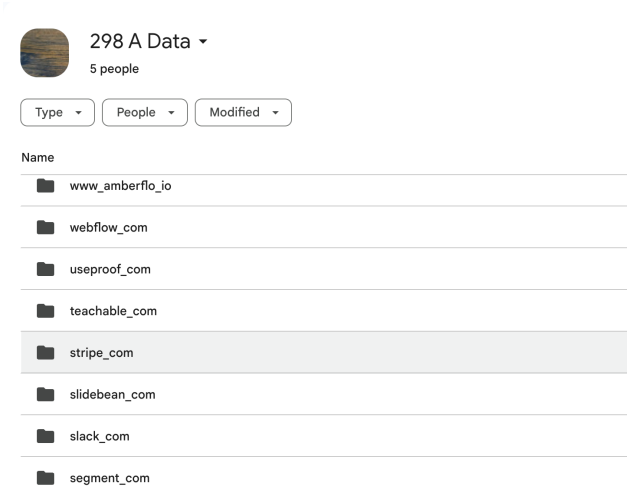
To illustrate, the process for each of the website is as follows: scrape the website data, organize the collected data into folders such as HTML, CSS, js, assets, and subfolders for individual aspects such as platform and solutions Then, convert the data of each website into a standard format. This is 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.

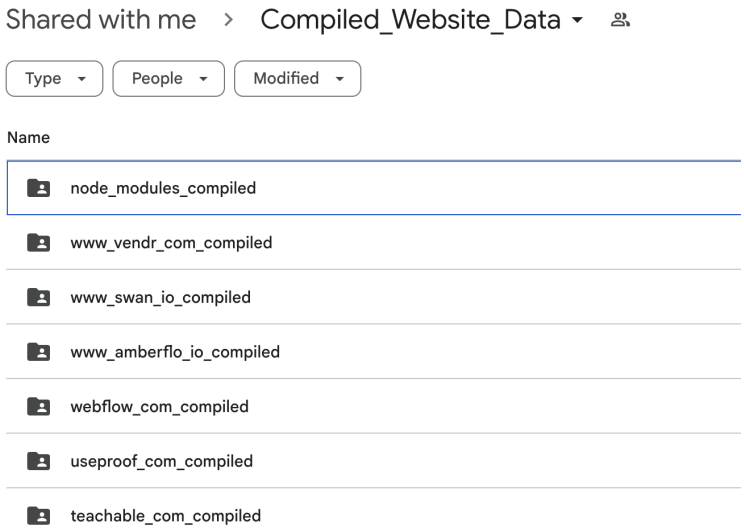
Link Structure Consistency: It is also used to arrange internal links of the websites dealing with correct links between the pages of the site. The linking information can be stored in another file which we are referring to as internal_links and this file outlines how various pages in a site are connected. 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)

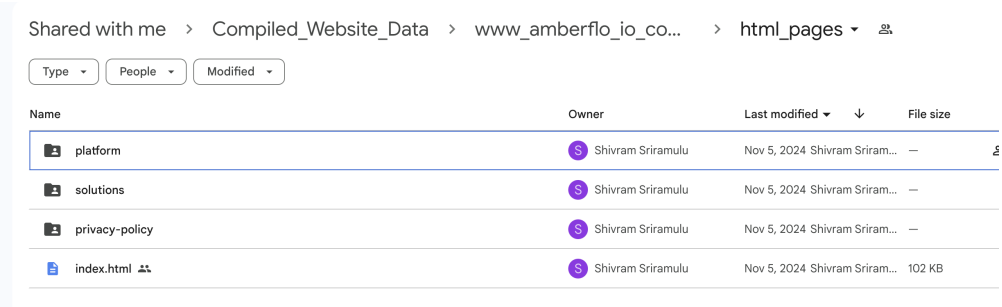


Figure 31

Pre-processed Data with Structure (css)

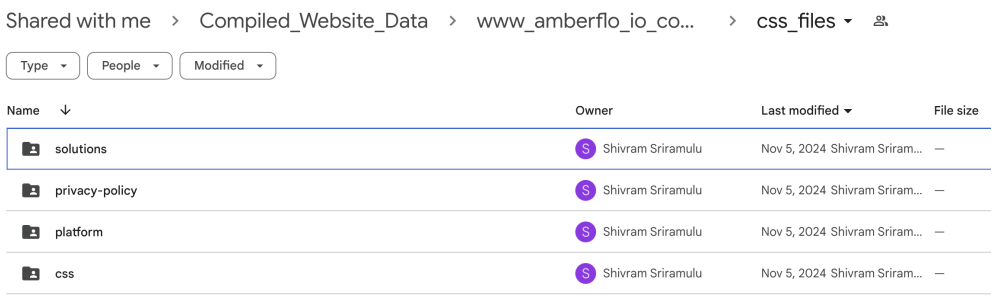


Figure 32

Pre-processed Data with Structure (js)

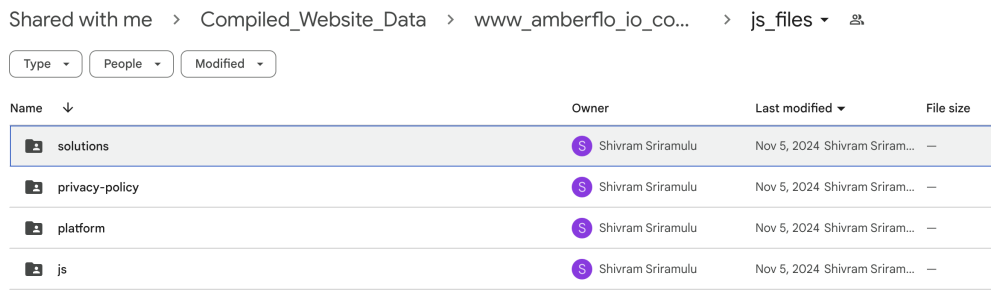


Figure 33

Pre-processed Data with Structure (images)

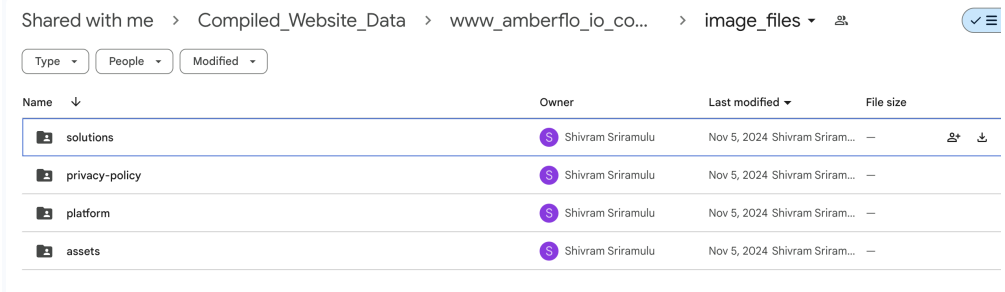
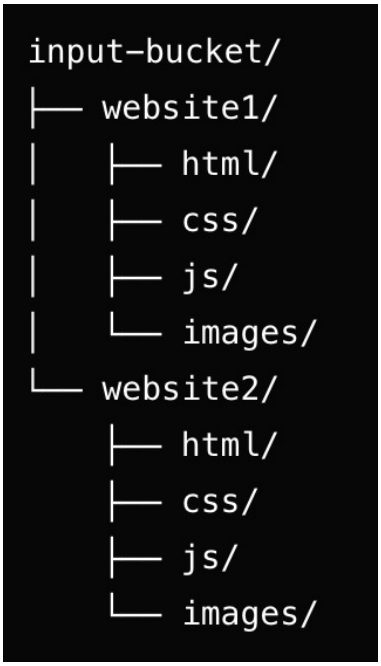


Figure 34

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 Pre-processed Data to JSON: Specifically, the following script was run within Google Drive: each website folder – encompassing HTML, CSS, JS, images etc. files – was converted into a single JSON file. This JSON file represents the entire structure and content of a website, organized as follows:

Page Structure: Every page (example 'index', 'about', 'contact') has their own html, css

and javascript content.

Image and Asset References: All pictures, graphics, and components applicable to every page are embodied in the JSON to ensure each view and functionality of the site.

Internal Link Structure: It also shows how exactly the pages are connected, so the general structure of the website is seen in the JSON. The conversion ended up in a creating a Google Drive folder which contained one JSON file for each website. The name of each JSON file was derived from the name of the website with which the content was extracted from to ease identification when dealing with various files.

Uploading JSON Files to S3:

After all the websites were transformed to JSON format 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 prompt-completion pairs. Below is an example JSON structure for a website shown in figure 36:

Figure 36

JSON Structure for a Single Website


```
{
    "website_name": "website1",
    "pages": {
        "index": {
            "html": "<html>...</html>",
            "css": "body { ... }",
            "js": "document.addEventListener('DOMContentLoaded', function() { ...
            "images": [
                "website1/images/logo.png",
                "website1/images/banner.jpg"
            ]
        },
        "about": {
            "html": "<html>...</html>",
            "css": "body { ... }",
            "js": "document.addEventListener('DOMContentLoaded', function() { ...
            "images": [
                "website1/images/team.jpg"
            ]
        },
        "contact": {
            "html": "<html>...</html>",
            "css": "body { ... }",
            "js": "document.addEventListener('DOMContentLoaded', function() { ...
            "images": [
                "website1/images/contact_icon.png"
            ]
        }
    }
}
```


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

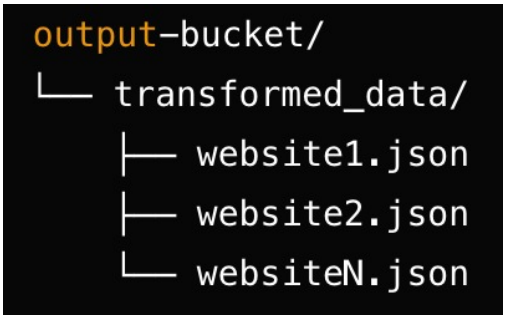
Pages: Has specifics of separate pages “index”, “about”, “contact”, etc.

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

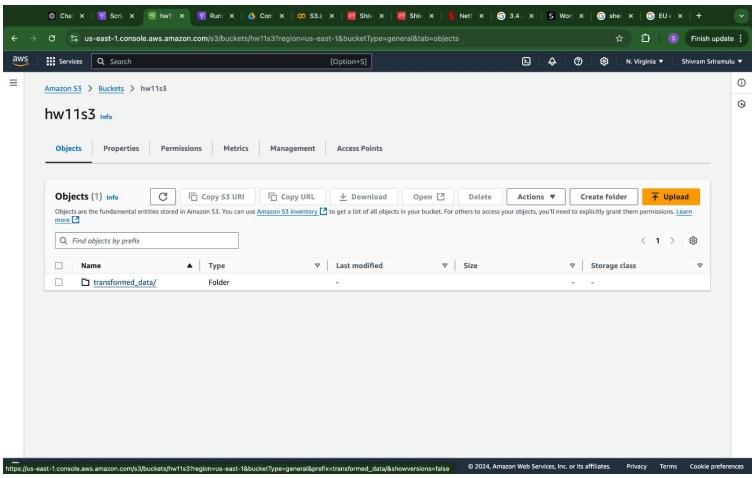
JSON Structure



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

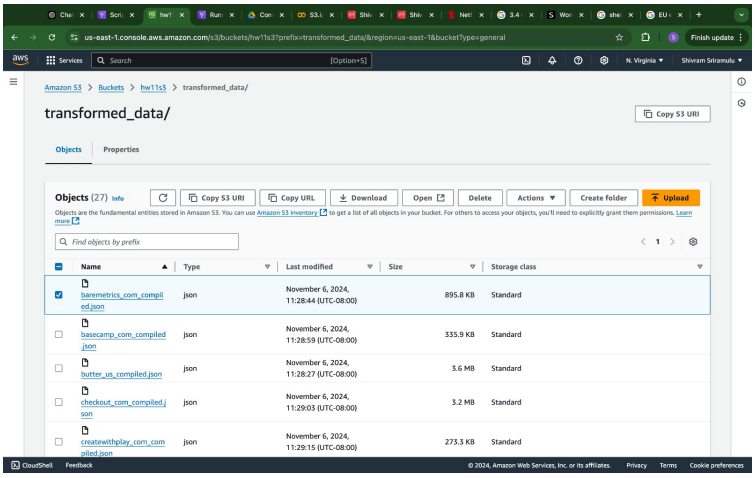
Transformed JSON Stored in S3



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 pairs prompt-completion for training, where the prompt is a textual prompt of a target website and the completion is the JSON representation of a real website. 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

Create Training, Validation and Test Data Sets by Using Transformed Data Sets

The subsequent step of this data preparation process was to create Prompt-Completion Pairs for fine-tuning the language model (LLM), where the originally pre-processed data was first restructured to a JSON format and stored on S3 bucket. A single prompt-completion pair reflects how the model should generate an output for an input which it a crucial aspect when tuning the model.

Prompt-Completion Pair Structure:

Prompt: The prompt can be described as an instruction or a request for the model given in the form of a description. For our use case, the prompt defines what the model ought to create or produce. For instance, a prompt might look like: >Create a web site by the name ‘website1’ with the pages and assets. This prompt instructs the model in terms of the structure that the created website has to possess and the kind of content to generate using the given website data.

Completion: The completion field is the JSON form of the website which the model is required to create based on the given prompt. It includes: Website Name: The name of the website. Pages: Lists of each of the page within the website, whereby each page designated:

HTML: The HTML tag of the page. CSS: The styling information that specific how the aspect of the page will be the displayed. JavaScript: The interactivity of the Operational part of the page and habits of visitors/Guests. Images: Links to image files for the page so that the model understands included visual elements. This arrangement helps the model to capture how best to build a whole web layout from a basic text description. Figure 40 shared the code used.

Figure 40

Prompt-Completion Code

```
# Process each JSON file in the input bucket
bucket = s3.Bucket(input_bucket_name)
for obj in bucket.objects.filter(Prefix=input_path):
    if obj.key.endswith('.json'):
        # Read and load the JSON file
        json_content = obj.get()['Body'].read().decode('utf-8')
        website_data = json.loads(json_content)

        # Create prompt-completion pair
        pair = create_prompt_completion(website_data)
        prompt_completion_pairs.append(pair)

# Save the prompt-completion pairs as JSONL format in the output bucket
output_data = '\n'.join(json.dumps(pair) for pair in prompt_completion_pairs)
s3.Object(output_bucket_name, output_file_key).put(Body=output_data.encode('utf-8'))

print(f"Successfully saved prompt-completion pairs to {output_bucket_name}/{output_file_key}")
```

The specific volumes of the Cartesian product prompt-completion are stored in a .jsonl (JSON Lines) format file where every line consists of the prompt-completion pair. This format is convenient for LLM fine-tuning because the prinup can be processed sequentially and in batches as needed.

Dataset Splits for Training and Testing:

Training Set: 412 websites (80%) Many of the prompt-completion pairs are to prepare the model to get the experience of the structure and information that is on B2B SaaS sites.

Validation Set: 52 websites (10%) A portion of this data is employed to evaluate the model on which check up is made when training is in progress to prevent over-training.

Test Set: 51 websites (10%) The test set includes example prompts as a means using which the effectiveness of the model that shall have been developed will be evaluated. In a production scenario, the test phase might mean a situation where the user inputs a prompt into the system and the model generates a website.

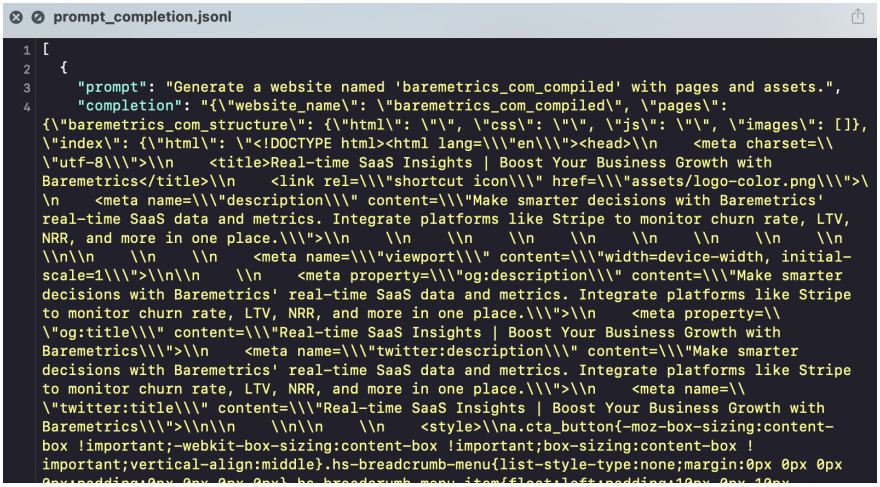
In this setup, a user prompt, as a test input, comes from the front-end application, and we can observe the extent to which the model can design a website structure when it processes real user requests.

Samples from Datasets:

The relations are documented in distinct JSON files where each line corresponds to an instance of the prompt-completion pair. Below in figure 41 is a sample entry from the prepared dataset:

Figure 41

Prompt-Completion Pair Example for Sample



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: We generated the prompt-completion pairs in JSONL format for tuning of the language model. Every pair included the textual description of what the model should create – a website of some kind, and the JSON structure of the creation's actualization. These pairs were then copied to an S3 bucket for training of the mentioned model which will be described later in the chapter.

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 42 shows the chart.

Figure 42

Top HTML Tags Used

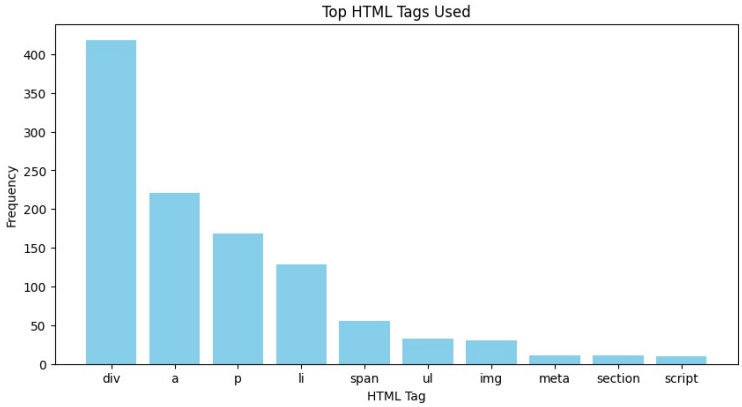
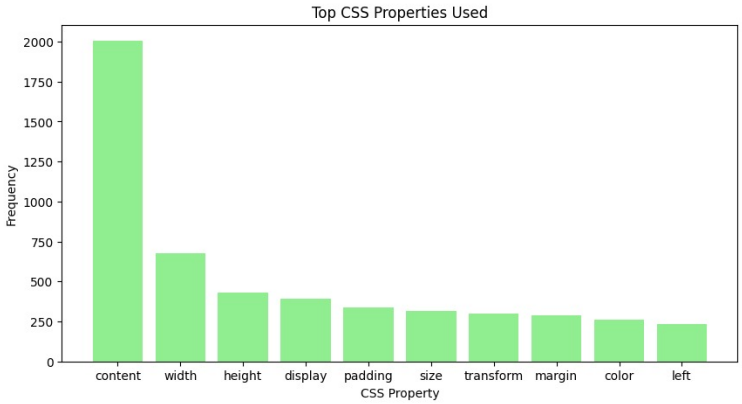


Figure 43

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 43 shows the

chart.

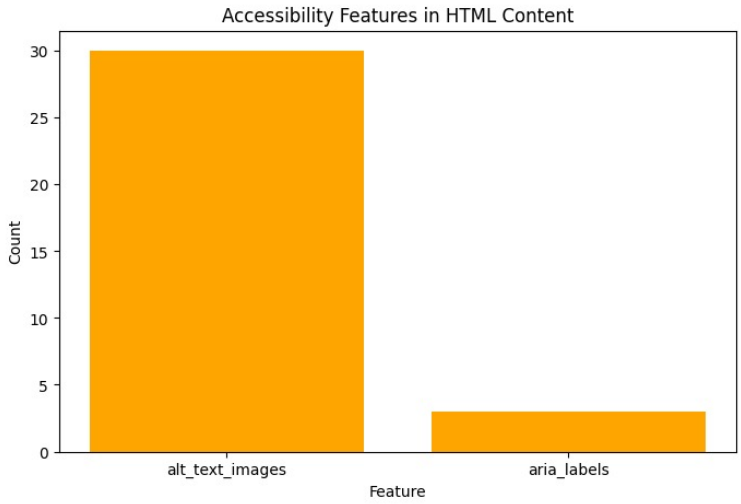
Accessibility Features in HTML Content:

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 44 shows the chart.

Figure 44

Accessibility Features in HTML Content



SEO Tag Presence Across Pages:

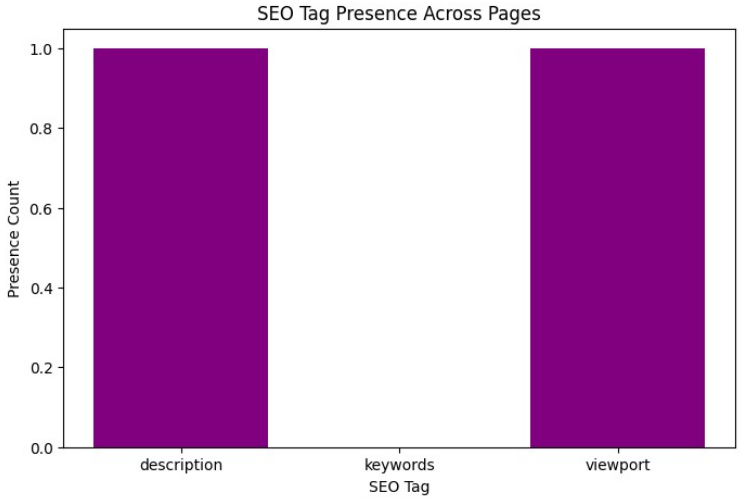
Meta tags such as description, keywords and viewport were checked to see if they exist. The study revealed that both the description and viewport tags could be seen and were used on the pages which is helpful for search engine indexing as well as for making a website responsive.

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 45 shows the chart.

Figure 45

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. Large Language Model Meta AI 2

LLaMA (Large Language Model Meta AI) is a series of basic language models designed by Meta AI (previously Facebook AI). The model generates human like text as it predicts the next word in a sequence, its insight makes it apt for applications in summarization, translation and content generation. LLaMA models originate from transformer architectures and are designed for high efficiency and extendibility. They are offered in many sizes, often from 7B (7 billion parameters) up to 70B, to fit different application needs Anunphop and Chongstitvatana (2022).

Optimizations which enhance the performance of the system and the reliabilities and make LLaMA 2 the second generation of the software. It employs state-of-the-art practices of rotary positional encoding, SwiGLU activation functions, and root mean square normalization for stability and speed Cao et al. (2018). In inference, LLaMA 2 uses subsequence retrieval in the form of a key-value cache table, which boosts its speed as it read through long sequences of text inputs. This makes LLaMA models ideal for academic use and powering practical applications with flexibility and power within a small framework Castillo-Campos et al. (2024).

Model Architecture

The image 46 represents the architecture of a transformer based language model probable to represent the structure and data flow of a model in both training and inference stages.

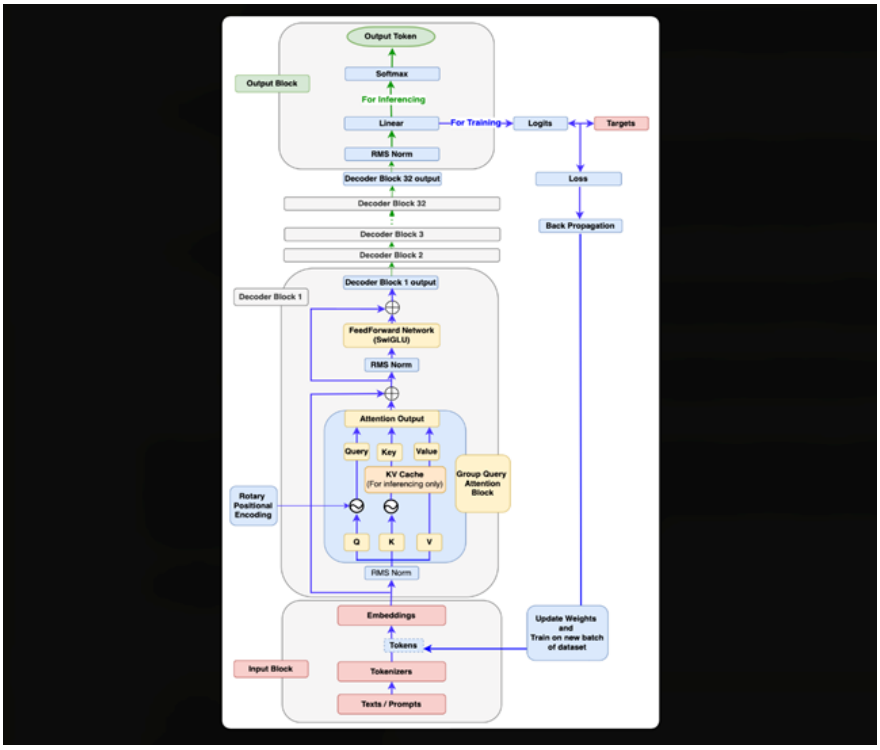
The LLaMA 2 architecture is initiated with an Input Block in which text prompts are converted to numerals that the model can comprehend. These tokens are then passed to a high-dimensional space which gives the meanings as represented by the text. To these embeddings, Rotary Positional Encoding is added in order to incorporate positional information

which allows the model to operate with sequences Chowdhery et al. (2023).

The architecture constitutes some Decoder Blocks the example of which is Blocks 1-32. Every decoder block is coupled with an 'Attention Mechanism' which takes the form of Query (Q), Key (K), Value (V), matrices to scale the weight of tokens of the sequence Wu et al. (2023). Next, in an attention mechanism, a FeedForward Network (SwiGLU) is incorporated to add non-linearity to increase model capacity. Moreover, called the Group Query Attention Block is applied to enhance the query processing at the inference stage by uniting similar queries. During training, RMS Norm (Root Mean Square Layer Normalization) is invoked at various stages with an aim to stabilize the training process as well as to shorten training time Chowdhery et al. (2023).

Figure 46

Model architecture for Llama-2



In the Output Block the output from the last decoder block is passed to a linear layer to generate logits, this is a vector of raw prediction scores for the token sequence in the vocabulary Castillo-Campos et al. (2024). During inference, logits are transformed into probabilities using a

softmax function, to govern token generation. While learning each of the logits is compared with the target token in order to calculate the loss, which is the measure of accuracy of the model. This loss is then propagated back through the network in order to update the model parameters, and increase the model's accuracy. Also, a Key-Value Cache is employed in inference to store previous computation in order to enhance sequential processing by avoiding look-up attention scores Dhyani et al. (2024).

This architectural setup is built to train and inference in parallel, enabling LLaMA 2 to process huge sequences as further discussed above, and yet the performance is not compromised Dhyani et al. (2024).

Algorithm

The LLaMA 2 algorithm in figure 48 first pre-processes input text to tokens, which are then transformed by the toolkit to high dimensions of vectors. To incorporate the sequence order information, we add rotary positional encoding. The tokens go through a series of decoder iteration blocks – each of these blocks includes an attention component (to listen to specific parts of the sequence), a feedforward network utilizing the SwiGLU activation function for non-linearity and capacity extension, and RMS normalization for stable learning. Inference has a novel Group Query Attention to enhance query processing. At the end of the decoder part, a linear layer is used to pruduce the logits which is used for token prediction. In training, the calculated logits are compared to target tokens for the purpose of computing the loss and use backpropagation to build new model weights. In inference, softmax turns logits to probabilities for sampling the next token, and the keys-value format saves previous computation results. This process goes on until tokens are generated or until batches are attained to train until the sequence is complete Cao et al. (2018).

Website Generation Task

For generating websites it can highly be beneficial to use the language understanding capabilities of LLaMA 2 coupled with the text generation to automate many aspects of website development. It can create HTML, CSS, and JavaScript altogether from simple commands and help the users to create the web site layouts, styles or even functions without knowing the deep

programming. This makes it easy to implement the concept of rapid prototyping whereby the designer together with the developer can model website structures to be adjusted He et al. (n.d.).

Furthermore, LLaMA 2 can write content that users want in their websites including products descriptions, FAQs and blog posts that can be written in certain tone and with certain themes in mind. It can as well assist in placing pertinent content on the website to improve visibility since it has gone through an initial SEO-friendly prompt. Arising from these contextual references, the model is in a position to come up with receptive and friendly codes that enhance website usability, across devices. In general, LLaMA 2 can be suggested as a tool that can facilitate the website generation and make it rather unintense in terms of the expert's workload Cao et al. (2018).

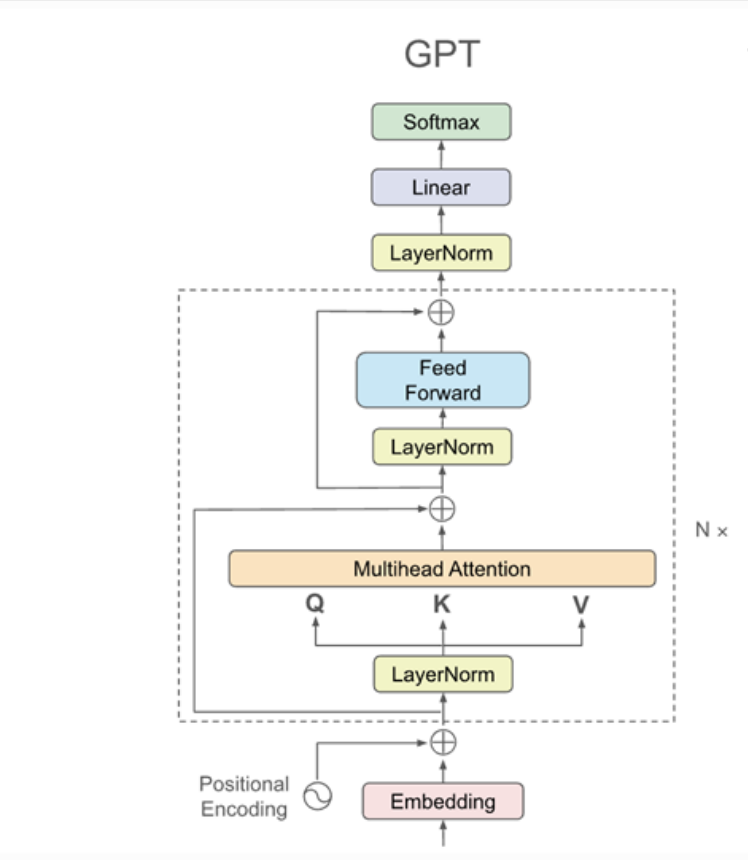
2. Generative Pre-trained Transformer 4

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 47

Model Architecture Diagram of GPT-4



The figure 47 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.

Figure 48

Algorithm for Llama-2

```
function LLaMA2_Model(input_text, mode="train"):
    # Step 1: Tokenization
    tokens = Tokenizer(input_text)

    # Step 2: Embedding and Positional Encoding
    embeddings = EmbeddingLayer(tokens)
    embeddings = RotaryPositionalEncoding(embeddings)

    # Step 3: Decoder Blocks
    for each decoder_block in DecoderBlocks:
        # Attention Mechanism
        Q, K, V = Attention(embeddings)
        attention_output = SelfAttention(Q, K, V)

        # Group Query Attention (Inference Only)
        if mode == "inference":
            attention_output = GroupQueryAttentionBlock(attention_output)

        # FeedForward and RMS Norm
        ff_output = FeedForwardNetwork(attention_output)
        embeddings = RMSNorm(ff_output + embeddings) # Residual connection

    # Step 4: Output Layer and Logits
    logits = LinearLayer(embeddings)

    # Step 5: Training or Inference
    if mode == "train":
        loss = ComputeLoss(logits, target_tokens)
        Backpropagate(loss)
        return loss
    else: # Inference
        probabilities = Softmax(logits)
        next_token = Sample(probabilities)
        UpdateKeyValueCache(Q, K, V)
        return next_token

# Main loop for batch processing (training) or token generation (inference)
function main(input_data, mode="train"):
    if mode == "train":
        for each batch in input_data:
            loss = LLaMA2_Model(batch, mode="train")
            UpdateWeights(loss)
    else:
        for each prompt in input_data:
            output_sequence = []
            while not end_of_sequence:
                next_token = LLaMA2_Model(prompt, mode="inference")
                output_sequence.append(next_token)
                prompt = UpdatePromptWith(next_token)
            return output_sequence
```

The basic structure of the GPT-4 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.

3. Gemini

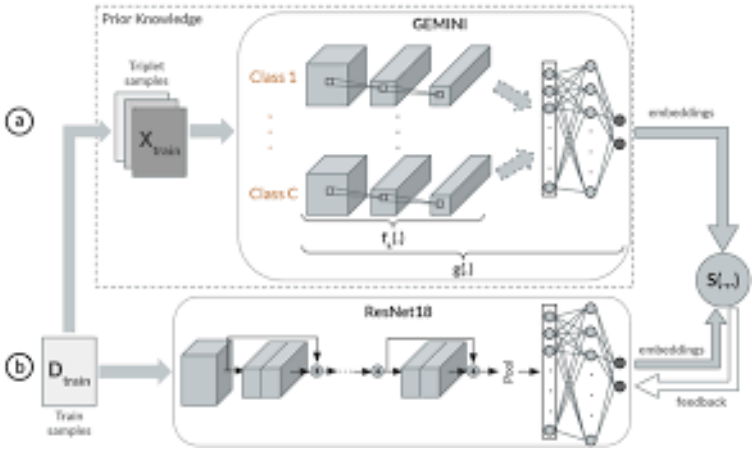
The Gemini model is an LLM belonging to Google DeepMind that sets the technical goal to further step up natural language understanding and generation. Supervised on general vocabulary, Gemini is trained on a large number of data sets, and with its roots established in

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 49

Model Architecture Diagram of Gemini



It seems to represent an architecture in figure 49 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).

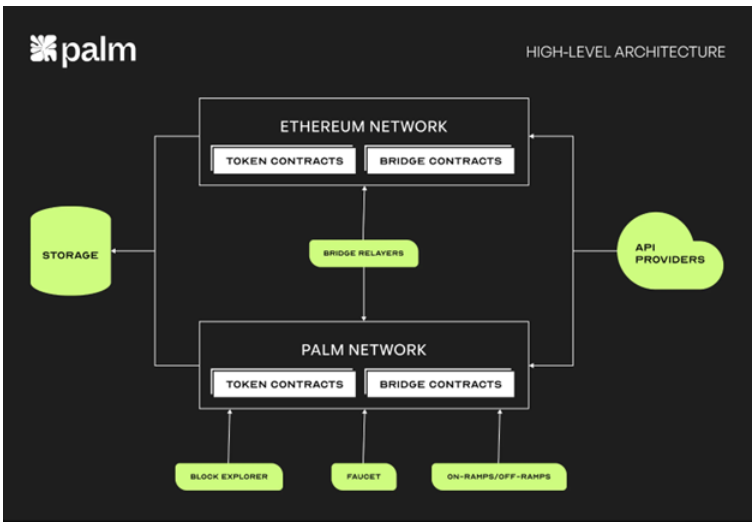
4. The Pathways Language Model (PaLM)

The Pathways Language Model (PaLM) is an advanced language model created by Google with the purpose of being an underlying system for multiple applications of artificial intelligence. By leveraging PaLM's access and emphasis towards human writing, PaLM applies across a spectrum from conversational AI to complicated content creation. It employs transformer-based structure architecture to make it to work with big data feeds effectively Chen et al. (2023). PaLM is lauded for its ability to reason about it contextually, to bring a sense of realism to a communication stream. First, it is also scalable to handle large amounts of data to learn and fine-tune model results from the provided data Briakou, Cherry, and Foster (2023). As a result we can see it can handle high accuracy language tasks because of the algorithms used. This is the model that is intended to understand finer nuances, feelings and intentions in the text. It is important for the domains that require an accurate, timely, and dynamic approach Askell, Bai, Chen, Drain, et al. (2021).

Model Architecture and algorithm

Figure 50

Model Architecture Diagram of Palm



The model named Palm network in figure 50 assists in wholesome communication with Ethereum network which is particularly good for websites that may require implementing elements of dApps, tokens, or even blockchain. Palm model reusable token and bridge contracts help the Palm model to integrate well between Palm and Ethereum. This means that if Palm network is integrated with a website, the website can easily perform token transfer, smart contract execution and asset exchange between Palm and another ecosystem. The bridge relayers have an immense responsibility of making sure that transactions work in one direction from the Palm network to Ethereum while enabling websites to leverage Ethereum's massive blockchain asset and user base while reaping in on the efficiency and cost-effectiveness of the Palm Network.

According to Chowdhery et al. (2023)there are some beneficial technologies that enhance the interaction of the blockchain and its function on websites through the Palm network. Some of the features are a block explorer that assists the users trace the transactions with transparency and the faucet enables customers to get the test tokens for testing or development tasks that are easier for developers to develop and test. On-ramps and off-ramps are helpful for websites that wish to offer helpful financial instruments, which will help the changeover between crypto and fiat to be

seamless. Besides making it easier to interface with the blockchain, this architecture also cuts the cost of the transaction as well as makes it faster than implementing the process of implementing blockchain technology on the website enabling the websites to bring the novel decentralized services to their consumers.

Model for Website Generation

Based on generative modeling and supervised learning, the PaLM uses the transformer to work with sequential data and generate semantically similar text. These non-technical newspaper articles incorporate sophisticated language patterns and semantic dependencies that the model acquires through pre-training development via large datasets and subsequent fine-tuning. It has functionalities of accepting user inputs on the type of industry and design preferences to help in the generation of the full featured multi-page content generation; comprising of Home, About, Services, and Contact among others. Moreover, updates on facilities, courses, and programs, as well as on other promotional contents, can be done easily and constantly through the creation of specially-designated templates that adhere to PaLM templates for quality and professional look of the sites as well as formatting to ensure that all final products look professional and uniform in design and appearance.

5. Claude

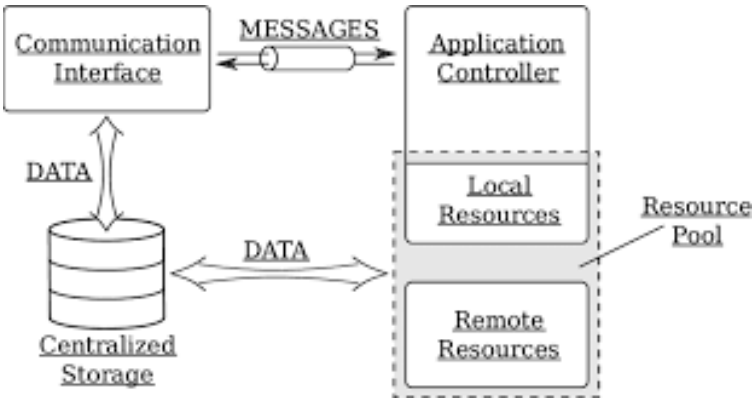
Claude is available on the artificial intelligence platform and was designed as a large language model (LLM) by Anthropic for general use across a range of natural language processing tasks Askell, Bai, Chen, et al. (2021). Claude, named in honor of the father of information theory, named Claude Shannon, 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 51

Model Architecture Diagram of Claude



The diagram 51 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 pagesCotterell and Sap (2022). 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 Perez et al. (2022). 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' experienceCotterell and Sap (2022). 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 Perez et al. (2022).

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
Llama 2	16–32	50–200	GPU (NVIDIA RTX 3090, A100, or similar), Python
GPT-4	32–64	200–500	High-performance GPU (NVIDIA A100, V100, RTX 3090), Python
PaLM	32–128	500–1000	Multi-GPU setup (NVIDIA A100, V100), 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

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

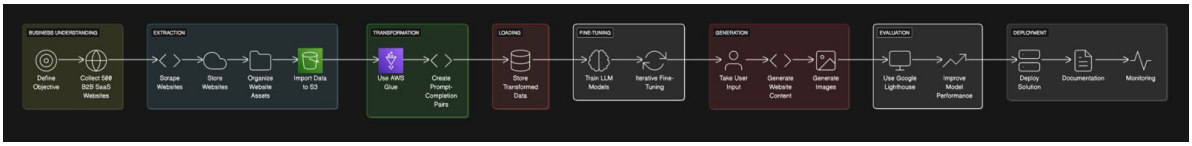
The table 11 shows the Libraries , Methods and their Purposes which are important in leveraging model training, data processing, and evaluation through a wide raft of libraries. Each library was chosen to do something specific in order for a wide range of tasks—from tokenization to data manipulation 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 52

Architecture for AI powered website generation



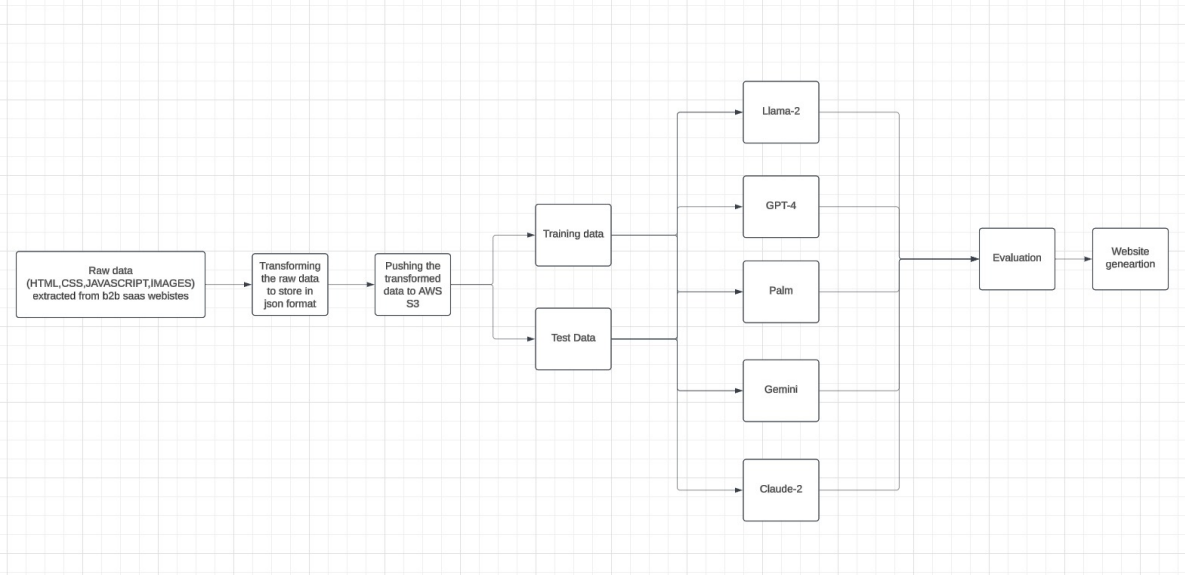
The pipeline shown in 52 starts with data extraction process – web scraping takes place on a Google Cloud VM where the HTML, CSS and JavaScript files of the target websites are collected. The raw data is temporarily stored in Google Cloud Storage after which the data is preprocessed. Data extraction 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. This step involves the use of GPU for computing where frameworks such as PyTorch or JAX are used for training deep learning models. After the model is trained, it is served using Streamlit a web application frame work used for real time prediction. By this arrangement, the pipeline effectively arranges for harvesting, preparing, training and deploying of a machine learning model for use in such tasks as generation of website content or classification in what can be offered in a web interface.

The figure 53 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 53

ML data flow for AI powered each website generation



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 . . .

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
Llama-2	Ensuring structured content with professional format	Standardizes headings, labels, and metadata for content	Focuses on structured data outputs, yielding polished and professional results	Consistently produces professional, structured content	Limited in creating conversational or dynamic text suitable for diverse audiences

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 responsible and mutually visually identical while used on a desktop or in a mobile and tablet

Table 13

Model Comparison (Part 2)

Model	Problems	Features	Approach	Strengths	Limitations
PaLM	Enhancing website design, layout, navigation, and responsiveness	Adjusts layouts for intuitive navigation and logical site structure	Applies layout optimization and responsive design principles	Improves usability with seamless navigation and responsive layouts	Limited in executing complex layout customization without additional guidance
Claude-2	Generating custom code for B2B SaaS websites	Assists with HTML, CSS, JavaScript snippets, offering basic layout and functionality suggestions	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

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.

Llama-2 provides important follow-up services to the professionalization of content by creating consistent headings, labels, and metadata. This structured output allows website content to look almost perfectly professional and well-arranged like other well-structured website content output. Another advantage of Llama-2 is that through reinforcement learning from human feedback Llama-2 is able to modify to the various formatting requirements and still produce well structured work. This is good when working with structured data, and when the content implies a hierarchy, such as documents, reports, or even web pages. But Llama-2 is not so useful for building interactive elements, it files under text formatting and content structuring.

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.

PaLM. According to the paper Singhal et al. (2022) palm is a highly efficient AI tool which may be used for increasing the level of usability of the Internet site and its construction, construction, structure, and adaptability, etc. These are its potentiality for the creation of the natural designs that improve site navigation and easy site organization as a way to assist users in finding the information they might interest and/or engage contents of the site. It also applies the layout optimisation principles of a responsive design in order to guarantee that the PaLM developed websites are highly responsive to the ubiquitous devices and displays including the mobile and desktop. The changes in design of PaLM serve to reduce the likelihood of the situation where a user may have a less than pleasant interaction with the application. However, it is shown in the present study that with increasing the dimensionality of layout design problem and specializing in it, PaLM does not outperform E-DK beyond ordinary layout constraints in the objective function and design flexibility. In such cases, it may need some more guidance; or user intervention to deal with certain complexity of designs. Therefore PaLM is most effective in the application that enhances the sites designed to adhere to the best practice and guidelines while permitting changes and inventions where and when needed to control the delivery and

accessibility for the user.

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 54, 55 are the screenshots from report.

Figure 54

Lighthouse Report 1

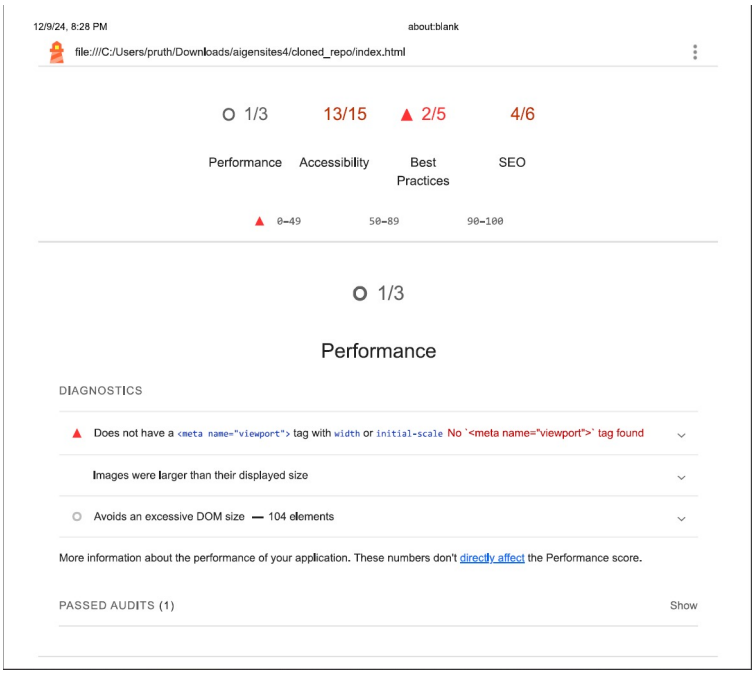
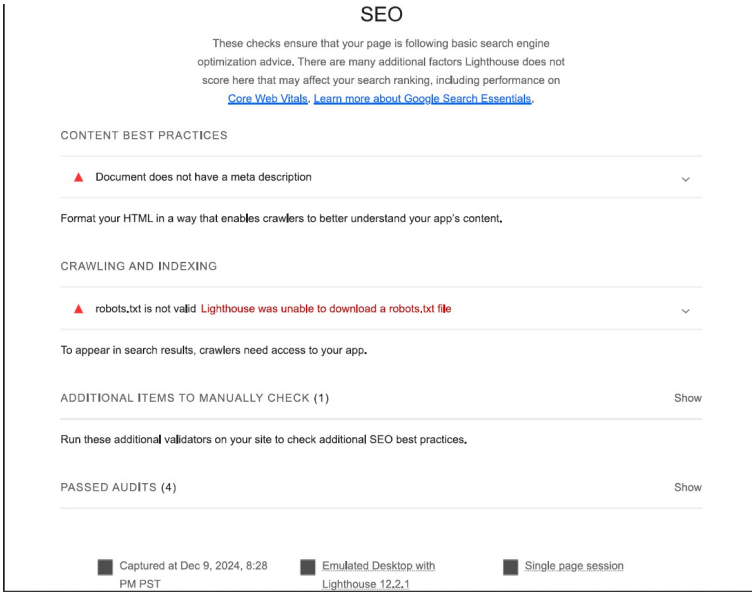


Figure 55

Lighthouse Report 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.

Structure and Formatting Consistency (for Llama-2). Llama-2 consistency in maintaining professional formatting can be measured with ROUGE and METEOR scores, accompanied by Google Lighthouse SEO and Accessibility metrics:ROUGE and METEOR scores measure the coherence and relevance of structured headings, labels, and metadata against ideal reference texts through output comparisons.Structured content with Lighthouse SEO ensures the best practices for search visibility by using proper headings, labels, and metadata.Accessibility checks in Lighthouse confirms that the structure and formatting are user-friendly and accessible to people with disabilities.

Navigation and Layout Optimization (for PaLM). According to the paper Bi et al. (2020)The PaLM focus on website navigation and layout can be easily and swiftly evaluated using Google Lighthouse metrics for Performance, Best Practices, and Accessibility.Performance evaluates load times and responsiveness, which is critical for smooth navigation and a positive user experience.Accessibility checks That the navigation elements are intuitive and accessible to all users make the site usable.Best Practices: Lighthouse runs code quality and security audits on navigation elements to help ensure they're optimized for real users. More metrics provide insight into the prowess of PaLM in chaperoning users, such as click path analysis and user journey mapping.

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. The 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.

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

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., Drain, D., Ganguli, D., Henighan, T., . . . Kaplan, J. (2021). A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861v3*. Retrieved from `https://arxiv.org/abs/2112.00861v3`

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*.

Bi, B., et al. (2020). Palm: Pre-training an autoencodingautoregressive language model for context-conditioned generation. *arXiv preprint arXiv:2004.07159*.

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

Briakou, E., Cherry, C., & Foster, G. (2023). Searching for needles in a haystack: On the role of incidental bilingualism in palm's translation capability. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 9432–9452). Retrieved from `https://aclanthology.org/2023.acl-long.619`

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)

Chen, X., Wang, X., Changpinyo, S., Piergiovanni, A., Padlewski, P., Salz, D., . . . Soricut, R. (2023). Pali: A jointly-scaled multilingual language-image model. In *Proceedings of the international conference on learning representations (iclr).* Retrieved from https://arxiv.org/abs/2209.06794

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.

Chowdhery, A., et al. (2023). Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, *24*(1144). Retrieved from https://jmlr.org/papers/v24/22-1144.html

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).

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

He, A., Key, D., Bulling, M., Chang, A., Shapiro, S., & Lee, E. (n.d.). Hlstransform: Energy-efficient llama 2 inference on fpgas via high level synthesis. *arXiv preprint arXiv:2309.12345*.

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*.

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

Singhal, K., et al. (2022). Large language models encode clinical knowledge. *arXiv preprint arXiv:2212.13138*.

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

Wu, C., Lin, W., Zhang, X., Zhang, Y., Wang, Y., & Xie, W. (2023). Pmc-llama: Towards building open-source language models for medicine. *arXiv preprint arXiv:2304.14454v3*. Retrieved from `https://arxiv.org/abs/2304.14454`

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.)

Appendix

Appendix A: Supplementary URL

For additional information, refer to the following URL:

- `https://drive.google.com/drive/u/3/folders/1V3f4S_klL3-wupHzH_UdHEyX0vCRzHwQ`