AI-POWERED ENGINEERING TOOL

A DESIGN PROJECT REPORT

Submitted by

BARANI DARAN V S

HANS BENEDICT A

LAKSHMI NARAYANAN P

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

JUNE, 2025

AI-POWERED ENGINEERING TOOL

A DESIGN PROJECT REPORT

Submitted by

BARANI DHARAN V S (811722001005)

HANS BENEDICT A (811722001011)

LAKSHMI NARAYANAN P (811722001027)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

JUNE, 2025

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)
SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this design project report titled **"AI-POWERED ENGINEERING TOOL"** is the bonafide work of **BARANI DHARAN V S (811722001005), HANS BENEDICT A (811722001011), LAKSHMI NARAYANAN P (811722001027)** who carried out the design project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other design project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. T.Avudaiappan,M.E., Ph.D.,
HEAD OF THE DEPARTMENT

Department of Artificial Intelligence
K.Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

SIGNATURE

Ms. S.Murugavalli,M.E., Ph.D.,
SUPERVISOR

ASSOCIATE PROFESSOR
Department of Artificial Intelligence
K.Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112

Submitted for the viva-voce examination held on ……

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We jointly declare that the design project report on **"AI-POWERED ENGINEERING TOOL"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF TECHNOLOGY**. This design project report is submitted on the partial fulfilment of the requirement of the award of Degree of **BACHELOR OF TECHNOLOGY**.

Signature

_____

BARANI DHARANI V S

_____

HANS BENEDICT A

_____

LAKSHMI NARAYANAN P

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this design project.

We are glad to credit honourable chairman **Dr. K.RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our design project and offering adequate duration in completing our design project.

We would like to thank **Dr. N. VASUDEVAN, M.E., Ph.D.,** Principal, who gave opportunity to frame the design project the full satisfaction.

We whole heartily thank **Dr. T.AVUDAIAPPAN**, **M.E., Ph.D.,** Head of the department, **ARTIFICIAL INTELLIGENCE** for providing his encourage pursuing this design project.

We express our deep and sincere gratitude to our design project guide **Ms. S.MURUGAVALLI, M.E., Ph.D.,** Department of **ARTIFICIAL INTELLIGENCE,** for her incalculable suggestions, creativity, assistance and patience which motivated us to carry out this design project.

We render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

AI-Powered Engineering Tool presents a novel, intelligent solution for automating the creation of fully functional and visually responsive websites from natural language descriptions. Unlike traditional web development methods that require manual coding and design skills, this AI-driven framework leverages large language models (LLMs) via Together.ai APIs to understand user prompts and generate complete frontend code including HTML, CSS, and JavaScript. The system is built with a modular architecture comprising a user-friendly frontend interface, a Node.js-based backend server, and an AI generation engine, facilitating seamless interaction between user intent and generated output. The core objective of this tool is to democratize web development by enabling users—even those without technical expertise—to design and deploy websites effortlessly. The system pipeline initiates with prompt ingestion and validation, followed by AI-assisted code generation tailored to the provided description. It includes integrated features such as real-time website preview rendering, download as a ZIP package, and optional backend scaffolding. Furthermore, the platform is scalable, allowing future enhancements such as database integration, backend code generation, template customization, and deployment support. Key innovations of this project include prompt-to-code transformation, live preview mechanisms, and zero-setup deployment support. Emphasizing extensibility and user experience, this AI tool offers a fast, reliable, and creative medium for prototyping, teaching, or launching real-world websites—all powered by artificial intelligence.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	-	Artificial Intelligence
API	-	Application Program Interface
IDE	-	Integrated Development Environment
NLP	-	Natural Language Processing
UI	-	User Interface
UX	-	User Experience

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The AI-Powered Engineering Tool is a modern, web-based platform designed to simplify and enhance engineering workflows by leveraging the capabilities of artificial intelligence. Traditionally, engineering tasks such as calculations, report generation, diagram drafting, and technical blogging are handled manually using a combination of software tools like MATLAB, AutoCAD, MS Word, and web searches. These tasks often require domain expertise, extensive time, and access to multiple platforms, leading to inefficiencies and fragmented workflows.

The AI-Powered Engineering Tool aims to unify and automate these processes by offering an all-in-one solution. At its core, the system utilizes advanced natural language processing models (like OpenAI's GPT) to interpret user input and perform a variety of engineering-related tasks—ranging from solving equations and generating formatted reports to creating diagrams and summarizing technical concepts. The platform also includes a blog module, where AI helps generate and format high-quality educational content, tutorials, and engineering insights, reducing the effort needed to maintain academic or project-related documentation.

Designed with accessibility and ease of use in mind, the interface is built using Streamlit or a similar front-end framework to allow intuitive user interactions. Whether you're a student, educator, or professional engineer, this tool helps reduce manual work, enhances productivity, and delivers consistent, professional-quality outputs across tasks. By combining engineering logic, natural language understanding, and automation, this system is an ideal assistant in educational institutions, research centers, and small to mid-sized engineering firms.

1.2 OBJECTIVE

The primary objective of the AI-Powered Engineering Tool project is to create an intelligent, accessible, and integrated platform that simplifies and automates routine engineering tasks using artificial intelligence. The tool is designed to support students, educators, and practicing engineers by providing a centralized system where they can perform calculations, generate reports, create technical content, and access learning materials without switching between multiple software applications. By leveraging advanced AI technologies such as natural language processing and machine learning, the system enables users to interact with it through simple, conversational input and receive accurate and context-aware responses.

This project aims to eliminate the dependence on traditional tools that often require specialized knowledge, such as MATLAB, AutoCAD, or graphic design software, which can be a barrier for non-experts or students. Instead, users can input problems or queries in plain language, and the system will generate appropriate outputs—be it a solved equation, a formatted report, a concept explanation, or a code snippet. Additionally, the tool supports the generation of educational blogs and technical documentation, allowing users to produce and publish high-quality content quickly and easily.

The core objective is to enhance learning and productivity by reducing the time and effort required to perform repetitive tasks. With built-in modules for unit conversion, data visualization, and report formatting, the platform not only serves as a smart calculator but also as a creative assistant. The interface is developed with user-friendliness in mind, ensuring that individuals from various engineering backgrounds and levels of technical expertise can effectively utilize its features. Ultimately, the project seeks to demonstrate the potential of AI in transforming the engineering landscape—making workflows faster, smarter, and more inclusive.

CHAPTER 2

LITERATURE SURVEY

2.1 THE INTEGRATION OF AI IN SOFTWARE ENGINEERING PHASES AND ACTIVITIES

Usman K. Durrani

This proposed model reviews the use of Artificial Intelligence (AI) in software engineering from 2013 to 2023. It highlights the application of AI techniques such as machine learning, deep learning, and natural language processing across various phases including requirements, design, development, testing, deployment, and maintenance. The study focuses on tasks like defect prediction, code generation, and test optimization, and evaluates tools like CodeBERT and GPT based on their efficiency. A conceptual model for AI-driven software development is also proposed.

Merits

- AI models can predict defects, estimate effort, and classify requirements with high precision.
- Tasks such as test generation, bug fixing, and code documentation are automated, reducing manual effort.
- Developers save time through AI-assisted coding tools (e.g., Copilot, CodeWhisperer).

Demerits

- Expert systems and predictive analytics help in making informed architectural and design choices.
- Performance heavily depends on the quality and quantity of training data.
- Incorporating AI into traditional SE workflows requires additional tools, skills, and infrastructure.

2.2 AI-DRIVEN SMART INDUSTRIAL EQUIPMENT LIFECYCLE: A COMPREHENSIVE REVIEW

Xiaohong Zhang

This proposed model presents a comprehensive literature review on the application of Artificial Intelligence (AI) techniques throughout the lifecycle of industrial equipment in smart manufacturing. The lifecycle is categorized into three main phases: Beginning of Life (BOL), Middle of Life (MOL), and End of Life (EOL). AI techniques such as machine learning, deep learning, and optimization algorithms are applied to tasks including product design, predictive maintenance, quality control, and recycling decision-making. The performance of AI-driven systems is analyzed based on their ability to improve efficiency, reduce downtime, and enhance sustainability in industrial operations. Tools and technologies used in AI integration are evaluated in terms of real-time analytics, predictive capabilities, and lifecycle impact.

Merits

- AI-driven automation leads to faster and more accurate manufacturing processes.
- Early detection of potential failures reduces downtime and maintenance costs.
- Optimized resource utilization and waste reduction contribute to eco-friendly operations.

Demerits

- Handling vast amounts of heterogeneous data from various sources requires robust infrastructure.
- Seamless incorporation of AI into existing manufacturing systems can be technically challenging.
- Need for specialized personnel to develop, implement, and maintain AI solutions.

2.3 A RESEARCH AGENDA FOR GENERATIVE AI IN SOFTWARE ENGINEERING

Nguyen-Duc, Hosek, and Franch

This proposed model presents a comprehensive research agenda for the integration of Generative Artificial Intelligence (GenAI) in software engineering. It covers multiple software engineering phases including requirements engineering, design, implementation, quality assurance, maintenance, and project management. GenAI techniques are explored for tasks such as automated requirement elicitation, code generation, bug detection, testing, and project planning. The study identifies key challenges related to usability, ethics, transparency, sustainability, and industry adoption. A set of 78 open research questions is proposed to guide future investigations into effective and responsible use of GenAI tools. The paper also discusses human-AI interaction and governance frameworks to ensure trustworthy deployment of GenAI in software.

Merits

- GenAI tools can automate repetitive tasks, allowing developers to focus on more complex problems.
- Automated code generation and review processes can lead to higher-quality software.
- GenAI can speed up various phases of software development, from design to deployment.

Demerits

- The effectiveness of GenAI is heavily reliant on the quality and diversity of the data it is trained on.
- GenAI may struggle with understanding the broader context of software projects, leading to suboptimal solutions.
- The use of GenAI raises issues related to bias, accountability, and transparency in software development.

2.4 ARTIFICIAL INTELLIGENCE IN DIGITAL TWIN SYSTEMS: A SYSTEMATIC MAPPING STUDY

A. M. Da Silva

This proposed model presents a systematic mapping study on the integration of Artificial Intelligence (AI) in digital twin systems. It explores how AI techniques are applied across various components of digital twins, including data acquisition, modeling, simulation, and decision-making. Methods such as machine learning, deep learning, and optimization algorithms are employed for tasks like predictive maintenance, anomaly detection, and system optimization. The study analyzes and categorizes AI approaches based on their application domains, models used, and performance metrics. Tools and frameworks relevant to AI-enabled digital twins are reviewed and evaluated. The paper also discusses challenges and future directions for advancing AI integration within digital twin technologies.

Merits

- AI enhances digital twins by enabling accurate prediction of equipment failures and system behaviors.
- AI techniques support continuous monitoring and rapid response to changing system conditions.
- AI-driven analytics optimize performance, reduce downtime, and improve resource utilization.

Demerits

- AI integration requires large volumes of high-quality, labeled data, which can be difficult to obtain.
- Advanced AI models often demand significant computational resources, increasing costs and infrastructure needs.
- Combining AI models with existing digital twin platforms can be complex and time-consuming.

2.5 GENERATIVE AI TOOLS FOR LITERATURE REVIEWS IN ENGINEERING RESEARCH.

Austin Olom Ogar

This proposed model presents a comprehensive overview of generative AI tools designed to support literature reviews in engineering research in 2024. These AI techniques are applied across various stages of the literature review process, including data extraction, summarization, citation management, and trend analysis. Tools leveraging natural language processing, machine learning, and deep learning are used to automate and enhance the efficiency of reviewing vast amounts of academic literature. The selected AI tools are analyzed and compared based on their usability, accuracy, and ability to handle engineering specific content. Frameworks and platforms such as GPT-based models and other generative AI applications are evaluated for their effectiveness in streamlining literature review workflows.

Merits

- Automates time-consuming tasks like data extraction, summarization, and citation organization, speeding up the literature review process.
- Reduces human errors in identifying relevant papers and extracting key information.
- Capable of processing and analyzing vast amounts of literature that would be impractical manually.

Demerits

- Effectiveness is limited by the quality and scope of available digital literature databases.
- May miss nuanced meanings or context-specific information crucial for in-depth analysis.
- AI models can inherit biases present in training data, affecting the neutrality of results.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In the current or traditional engineering environment, completing tasks such as design calculations, report creation, code prototyping, and technical content development typically requires the use of multiple separate software tools. Common platforms include MATLAB or Excel for computations, AutoCAD for diagramming, LaTeX or Microsoft Word for documentation, and various online blogs or YouTube channels for learning. These tools often operate independently, forcing engineers, students, and educators to manually transfer data or results between applications.

This disconnected workflow not only consumes valuable time but also introduces the risk of inconsistency and human error. Moreover, most of these tools require a significant level of technical skill or specialized training, making them less accessible to beginners or users with limited backgrounds in engineering software. For example, generating a formatted report often involves manual typing, formatting, and referencing across different platforms—an effort that can be repetitive and error-prone.

3.1.1 Demerits

- Requires the use of multiple, disconnected tools for calculations, content writing, and visualization.
- Demands advanced technical knowledge or experience with software like MATLAB, AutoCAD, or Latex.
- Lacks AI-powered features such as natural language queries or automated report generation.
- Manual documentation and formatting processes are slow and error-prone.

3.2 PROPOSED SYSTEM

The proposed system, AI-Powered Engineering Tool, is an intelligent and interactive web-based platform designed to assist students, educators, and engineers in performing core engineering tasks such as solving equations, creating technical reports, and generating learning content—quickly and accurately. It addresses the inefficiencies of the traditional workflow, which involves switching between multiple tools like MATLAB, AutoCAD, and Word, by consolidating these functions into a single unified interface powered by artificial intelligence.

The user interface is developed using Streamlit or a similar Python-based framework, enabling real-time interaction through forms and text input boxes. The system can generate structured content, such as lab reports, design summaries, calculation steps, and technical blogs, which users can download or share directly. Future enhancements may include AI-generated diagrams, graphs, and simulations to further reduce dependency on manual design tools.

3.2.1 Merits

- AI-Powered Response Engine interprets technical queries and delivers solutions instantly.
- Generates structured content such as reports, abstracts, or tutorials automatically.
- Ideal for students, educators, and engineers in institutions, startups, and training centers.
- It can be extended to include diagrams, simulations, and interactive widgets for advanced use.

CHAPTER 4

SYSTEM SPECIFICATIONS

4.1 HARDWARE SPECIFICATIONS

- Processor : AMD Ryzen 5 5600H (6 cores / 12 threads)
- Graphics : NVIDIA GTX 1650 (4GB VRAM)
- RAM : 16 GB DDR4
- Storage : 512 GB SSD + 1 TB HDD
- Network : Wi-Fi 6, Bluetooth 5.0
- Display : 15.6" Full HD (1920x1080) IPS

4.2 SOFTWARE SPECIFICATIONS

- Operating System : Windows 11
- Node.js : v20.x
- Frontend Tools : React.js, Tailwind CSS, Vite
- Backend : Node.js (Express.js)
- AI API : Together.ai or OpenAI API
- Database : MongoDB or SQLite.

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

The AI Website Generator runs on a Windows 11 system powered by an AMD Ryzen 5 5600H CPU, GTX 1650 GPU, 16 GB RAM, and SSD storage. It uses a Node.js backend connected to Together.ai or OpenAI for generating HTML, CSS, and JavaScript from user prompts. The frontend is built with React.js and Tailwind CSS, communicating via REST APIs. Features include prompt validation, regeneration, live preview, optional code formatting, and ZIP export, making it a complete AI-powered web development tool.

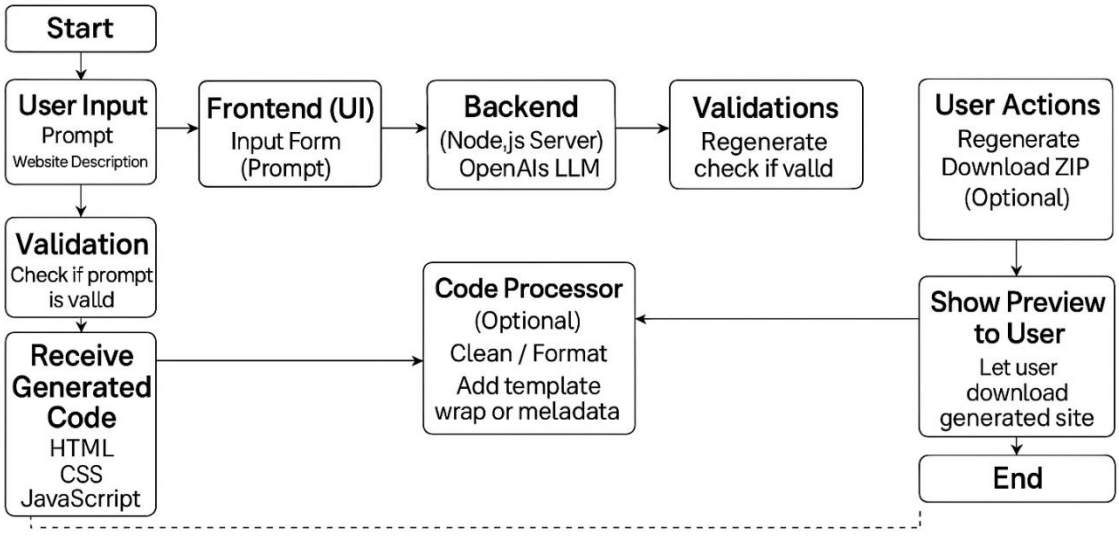

Fig.No.5.1 System Architecture

The AI Engineering Tool functions as an end-to-end system that allows users to generate complete websites from plain text prompts using the power of AI, combined with a responsive frontend and a capable backend infrastructure. The process begins on the frontend, where the user is presented with a simple yet modern user interface developed using React.js and styled with Tailwind CSS. The user provides a prompt—this could be a short description like "a personal portfolio site for a designer" or a more detailed instruction set specifying components, layout, or design preferences. This prompt is then passed through a validation layer which ensures the input is not blank or structurally invalid. If it fails validation, the system prompts the user to either modify or regenerate their input.

Once the prompt passes validation, it is sent via RESTful API to a Node.js backend. The backend acts as a communication bridge between the frontend and the AI model. Here, the prompt is packaged and sent to a third-party AI model API, such as Together.ai or OpenAI's GPT-based service, which processes the request and generates code based on the user's description. The generated response typically includes a set of files in HTML, CSS, and JavaScript, sufficient to create a standalone static website.

After receiving the AI-generated code, the backend optionally passes it to a code processing module, which enhances the raw output. This processor may reformat the code using tools like Prettier, add comments, inject metadata, or wrap the code inside a predefined template (like a responsive layout or SEO-ready header). Once the code is processed, it is returned to the frontend where a live website preview is rendered dynamically. This gives users an immediate visual of how their prompt has been translated into a functioning web page.

At this stage, users can interact with the result through a set of action buttons. They can choose to regenerate the site, edit the original prompt to refine results, download the full website as a ZIP archive, or optionally connect a backend for added functionality (such as form handling or dynamic content). The download package typically includes all frontend files neatly organized for local hosting or deployment.

CHAPTER 6

MODULES DESCRIPTION

6.1 PROMPT INTERPRETATION MODULE

The Prompt Interpretation Module is responsible for analyzing and understanding the user-generated prompts within the AI-powered engineering tool. Once a structured prompt is submitted, this module parses the input to extract key components such as the engineering task, domain, relevant parameters, constraints, and desired output format. It translates the natural language prompt into a machine-readable format that can be effectively processed by the AI engine. This module ensures that the user's intent is accurately captured, even if the prompt varies in structure or wording. It uses techniques such as keyword extraction, semantic analysis, and context recognition to understand complex engineering queries. By bridging the gap between human input and AI logic, the Prompt Interpretation Module plays a critical role in ensuring accurate, relevant, and domain-specific AI responses. This enhances the reliability and effectiveness of the tool in solving real-world engineering problems.

This structured representation is essential for the AI engine to accurately interpret user requirements and generate precise engineering solutions. By ensuring that the prompt is thoroughly understood and properly formatted, the module reduces ambiguity and improves the overall effectiveness and reliability of the tool. It acts as a bridge between human input and AI computation, enabling users with varying expertise to interact seamlessly with the system and receive meaningful results tailored to their specific engineering problems.

Using natural language processing (NLP) techniques such as keyword extraction, semantic analysis, and pattern recognition, the module converts unstructured and often varied user inputs into a structured, machine-readable format. When a user submits a prompt, this module analyzes the text to identify the

core engineering task, such as simulation, optimization, or design generation.It also detects the relevant engineering domain—mechanical, electrical, civil, or others—and extracts important parameters like measurements, material specifications, constraints, and expected output formats.

Packages used in prompt interpretation module

In the Prompt Interpretation Module of an AI-powered engineering tool these packages support natural language processing (NLP), data parsing, and integration with AI models.

Natural Language Processing (NLP)

- **spaCy**
  Used for tokenization, part-of-speech tagging, named entity recognition, and dependency parsing to understand the structure of user prompts.
- **sentence-transformers**
  Generates semantic embeddings to compare user input against predefined task categories or templates.
- **NLTK(NaturalLanguageToolkit)**
  Useful for basic NLP tasks like stemming,and keyword extraction.

Machine Learning / AI Integration

- **scikit-learn**
  For classification or intent detection models to map prompts to specific engineering tasks.
- **OpenAI**
  For directly using or fine-tuning large language models to interpret and process prompts.

Functions

- clean_prompt (prompt_text)
- extract_intent (prompt_text)

The Prompt Interpretation Module comprises several essential functions that work together to analyze, process, and structure user input effectively. The first critical function is prompt cleaning, which preprocesses the raw input by removing unnecessary characters, extra spaces, and noise, ensuring that the text is in a consistent format for further analysis.

Following this, the intent extraction function identifies the core action or task that the user wants the AI to perform, such as simulation, optimization, or code generation. This is often achieved through keyword matching or machine learning classification techniques. Parallel to this, the domain detection function determines the relevant engineering field, whether mechanical, electrical, civil, or computer engineering, by analyzing domain-specific keywords or using semantic similarity measures. Another vital function is parameter extraction, which locates and captures important numerical values and design specifications mentioned in the prompt, such as dimensions, voltages, or flow rates, using pattern recognition techniques like regular expressions.

The module also includes a constraints identification function that isolates any limitations or requirements specified by the user, such as material types, safety factors, or budget limits, which are crucial for guiding the AI's solution approach. To complete the interpretation, the output type inference function deduces the desired format of the results, such as graphs, code snippets, CAD models, or reports, ensuring that the AI's output aligns with user expectations.

6.2 WEBSITE CODE GENERATOR MODULE

The Website Code Generator Module is a sophisticated component of the AI-powered engineering tool designed to automate the creation of web-based user interfaces and engineering dashboards based on user requirements. This module leverages advanced AI techniques to interpret high-level design specifications, feature requests, and functional requirements provided in natural language or structured prompts, and then translates them into fully functional, clean, and efficient front-end code. It supports multiple web technologies such as HTML, CSS, JavaScript, and popular frameworks like React or Angular, enabling the generation of responsive and interactive web pages tailored to engineering applications.

The module typically integrates natural language processing (NLP) to understand the user's intent and design preferences, and it uses code synthesis algorithms that assemble reusable components, UI elements, and layouts according to best practices and project-specific constraints. Additionally, it incorporates features for embedding engineering visualizations such as graphs, CAD viewers, or simulation results, ensuring that generated websites are not only visually appealing but also highly functional for technical users. Error checking and optimization routines are embedded within the module to ensure that the generated code is clean, performs well, and is compatible across different browsers and devices.

By automating the web development process, this module significantly reduces the time and effort required to build custom engineering tools, enabling engineers and designers without extensive coding experience to quickly deploy user-friendly interfaces that facilitate interaction with complex AI-driven simulations and analyses. Overall, the Website Code Generator Module acts as a powerful bridge between engineering requirements and practical web applications, enhancing the accessibility and usability of AI-powered solutions.

Packages used in website code generator module

The Website Code Generator Module utilizes several powerful software packages to interpret user inputs and generate accurate, efficient, and responsive web code. One of the core packages is the transformers library by Hugging Face, which is used to access and run pre-trained large language models like GPT or Codex for generating HTML, CSS, and JavaScript code directly from user instructions.

These models help in understanding user requirements and synthesizing context-aware, readable front-end code. The Jinja2 package is often used for templating purposes; it allows the module to insert generated content into pre-defined HTML structures or UI layouts, ensuring consistency in design and modular code reuse.

For formatting and beautifying the generated code, the jsbeautifier package is employed, which helps enhance code readability and maintainability by structuring HTML, CSS, and JavaScript files with proper indentation and spacing.

To validate and lint JavaScript code, eslint can be integrated via subprocess or API calls; it checks for syntax errors, undefined variables, and code style issues to ensure that the generated code is error-free and follows modern web development standards. The React library (used via Node.js and its ecosystem) plays a central role in generating dynamic and component-based UIs.

When generating React-based code, packages like create-react-app or Vite are used to scaffold the project structure, and UI components are constructed using JSX syntax. In addition, Tailwind CSS is frequently included for styling purposes because it allows the generator to apply pre-defined utility classes to elements, speeding up UI design and ensuring responsive behavior across devices.

The Flask or FastAPI frameworks may be used on the backend to host the module and expose an API endpoint where user prompts are submitted and the

generated code is returned. For handling JSON data and API communication, the built-in json and requests libraries are used.

Finally, logging is integrated to track user requests, generation errors, and system performance, which helps during debugging and module improvement. Together, these packages form a comprehensive toolchain that enables the Website Code Generator Module to translate engineering-oriented user prompts into complete, functional, and responsive web code in real time.

Functions

The Website Code Generator Module consists of several core functions that work together to transform user prompts into functional front-end code. The first key function is parse_user_prompt(prompt_text), which is responsible for analyzing the user's natural language input and identifying relevant details such as the desired layout, UI components (like buttons, forms, or charts), styling preferences, and target technology (e.g., HTML, React, or Tailwind CSS).

The extract_ui_elements(parsed_data) function takes the parsed result and isolates individual components or sections to be generated—such as headers, navigation bars, cards, or input forms—based on keywords and contextual clues. After that, the generate_code_snippets(element_list) function plays a crucial role by converting each extracted UI element into corresponding code snippets using templates or AI-based code generation models.

For instance, if the prompt requests a form with validation, this function generates a structured HTML or JSX block along with any required JavaScript. Following code generation, the apply_styles(snippets, style_type) function is used to add styling based on user preference—whether using inline CSS, external stylesheets, or utility-first frameworks like Tailwind CSS—ensuring the visual appearance matches the requested design.

6.3 PREVIEW AND EXPORT MODULE

The Preview and Export Module is a critical component of the AI-powered engineering tool that enables users to visually inspect and obtain the output generated by various functional modules, such as simulation results, design outputs, or auto-generated code. This module provides a dynamic preview interface where users can interact with the results in real time before finalizing or exporting them. Whether it's a rendered 2D/3D engineering model, a circuit diagram, a graph, or a web-based UI layout, the preview system ensures that the output matches the user's expectations and specifications. It is built using front-end libraries and rendering engines like React, Three.js, or Plotly, which allow interactive visualization of CAD models, data charts, or structured web content. In the case of generated code, it often includes a live code editor or viewer where users can modify and re-preview the changes instantly.

Once the output is verified, the export functionality allows users to download the results in multiple formats such as PDF, PNG, HTML, JSON, or even project bundles like .zip files. This is made possible through backend services written in Flask or FastAPI, which handle export requests, generate the required files, and manage file storage and download links securely. For code-based exports, tools like jszip or html2pdf are integrated to bundle or convert the code output appropriately. The module also includes functions to ensure compatibility across different browsers and devices, offering a seamless and responsive user experience. Moreover, it logs user interactions and preferences to improve future previews and support version control for exported files. Overall, the Preview and Export Module plays a key role in making the AI-generated outputs actionable and user-friendly, empowering engineers to review, iterate, and deploy solutions with confidence and efficiency.

The Preview and Export Module is designed to provide real-time visualization and flexible output handling for AI-generated engineering results. It supports interactive previews of models, diagrams, web interfaces, or code using tools like React, Three.js, and Plotly. The module allows users to export outputs in multiple formats, including PDF, HTML, JSON, PNG, and ZIP, using backend frameworks like Flask or FastAPI. It ensures cross-platform compatibility, responsive design, and secure file handling, making the output review and download process seamless, efficient, and user-friendly.

Packages used in preview and export module

The Preview and Export Module makes use of several essential packages and libraries, each serving a distinct purpose in enabling real-time output visualization and flexible export options. One of the most important packages used on the front end is React, a JavaScript library for building responsive and interactive user interfaces. It is used to dynamically render previews of code-generated web layouts, forms, dashboards, or design elements.

For engineering-specific visualizations, the Three.js library is integrated to render interactive 3D models and simulations directly in the browser, making it ideal for mechanical, architectural, or structural engineering outputs. In scenarios involving data visualization (such as charts, graphs, and plots), Plotly.js or its Python counterpart plotly is used for creating real-time, interactive graphs that reflect engineering data outputs.

For live code preview and editing, packages like Monaco Editor (used in VS Code) are incorporated into the UI, enabling users to modify and test generated HTML, CSS, or JavaScript before exporting. On the backend,

Flask or FastAPI is used to build APIs that handle user export requests, generate files on the server side, and manage download processes.

When it comes to exporting HTML or graphical content to downloadable formats like PDF or image files, the html2pdf.js library is used on the client side, while reportlab or pdfkit is used in Python on the backend for server-side PDF generation.

For exporting full code projects, including multiple files, folders, and dependencies, jszip is used to dynamically generate ZIP files in the browser, while Python's built-in zipfile module may be used for backend-based packaging. Additionally, the Pillow library in Python is used for image processing when exporting visual content as PNG or JPEG.

The json module is commonly used to format structured data outputs, while the logging package ensures all export and preview actions are recorded for debugging and audit purposes. Together, these packages enable the module to offer rich, interactive previews and smooth, multi-format export capabilities, ensuring a user-friendly and technically robust experience.

Functions

The Preview and Export Module includes several key functions that work together to handle the rendering, review, and export of AI-generated outputs in a user-friendly and efficient manner. The process typically begins with the render_preview(data) function, which dynamically displays the generated output—such as a web page, 3D model, code snippet, or graph—on the user interface using front-end rendering libraries.

For code-based outputs, the load_code_editor(content) function initializes an interactive editor (like Monaco Editor), allowing users to view and modify

HTML, CSS, or JavaScript before exporting. In case of 3D visualizations or CAD-like models, the init_model_viewer(model_data) function is used to load and manipulate 3D files (e.g., using Three.js), offering interactive rotation, zoom, and pan features.

Next, the prepare_export(format, content) function processes the previewed content into the format selected by the user—whether it's PDF, PNG, HTML, or JSON—by structuring or transforming the content appropriately. Following this, the export_to_pdf(html_content) function uses PDF generation libraries to convert web previews or documents into printable PDFs with layout accuracy.

Similarly, the export_as_image(canvas_data) function captures on-screen visualizations and converts them into downloadable image formats like PNG or JPEG. When the export involves code files or entire web projects, the package_code_files(file_list) function collects all necessary files, including assets and dependencies, and compresses them into a ZIP archive using packaging libraries.

The download_file(file, file_type) function then enables users to download the final output through the browser or backend API, handling file paths, MIME types, and error handling. For managing version history or repeated export tasks, the save_export_log(user_id, export_type) function records details of each export action, helping in both debugging and audit tracking.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

The AI-Powered Engineering Tool represents a transformative step toward modernizing the way engineering tasks are approached and executed. In a conventional environment, engineers and students rely on multiple disconnected tools to perform calculations, create technical documentation, or visualize engineering concepts—often resulting in inefficiency, increased workload, and limited accessibility. This project addresses these challenges by consolidating essential engineering functions into one intelligent platform powered by artificial intelligence.

Through natural language processing, users can pose technical queries and receive accurate responses in real time, eliminating the need to memorize formulas or toggle between various applications. The platform's web-based design ensures that users can access its features without installing complex software, thereby democratizing engineering knowledge for learners and professionals in resource-constrained environments. The system significantly reduces manual work, enhances accuracy, and promotes self-learning by generating context-aware content such as step-by-step problem solutions, summaries, and formatted reports.

This not only improves efficiency but also makes engineering workflows more inclusive, especially for students, educators, and professionals with limited access to expensive software or advanced technical skills. Overall, the project fulfills its objective of providing a smart, efficient, and accessible engineering assistant, thereby enhancing the way engineering tasks are approached and completed.

7.2 FUTURE ENHANCEMENT

The current version of the AI-Powered Engineering Tool delivers a strong foundation, several future enhancements can elevate its functionality, usability, and reach. First, the integration of generative image models or diagram-drawing APIs can allow users to generate technical diagrams and engineering schematics directly from text, saving time and aiding visual understanding. Support for AI-generated code snippets in languages like Python, MATLAB, or C++ would assist users in implementing simulations or automation tasks more efficiently. The addition of cloud-based data storage, multi-user collaboration features, and version control can support academic group projects and industrial teamwork. Developing a responsive mobile application can further improve accessibility, allowing users to use the platform anytime, anywhere—even in lab settings or during fieldwork.

More advanced techniques incorporating speech-to-text input and multilingual support would make the tool more inclusive, enabling users from diverse linguistic and educational backgrounds to interact with it naturally. Integration with Learning Management Systems (LMS), GitHub, Google Drive, and document-sharing platforms can further extend its utility in educational and research institutions. Lastly, a personalized dashboard that tracks learning progress, offers practice questions, and recommends relevant content based on user behavior would transform the platform into an intelligent, adaptive learning environment. With these enhancements, the tool can evolve from a helpful assistant into a comprehensive engineering ecosystem tailored to the evolving demands of education, research, and industry.

APPENDIX A

SOURCE CODE

FRONTEND

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>AI Website Generator</title>
    <style>
      body {
        font-family: Arial, sans-serif;
        background: #121212;
        color: #eee;
        padding: 2rem;
        max-width: 700px;
        margin: auto;
      }
      h1 {
        color: #ff3b3b;
        text-align: center;
```

```
}
form {
  background: #222;
  padding: 1.5rem;
  border-radius: 8px;
  margin-bottom: 1.5rem;
}
textarea,
input[type="file"] {
  width: 100%;
  margin-bottom: 1rem;
  padding: 0.5rem;
  border-radius: 4px;
  border: none;
  font-size: 1rem;
}
button {
  background: #ff3b3b;
  color: white;
  border: none;
  padding: 0.75rem 1.5rem;
  font-size: 1rem;
```

```
    border-radius: 4px;
    cursor: pointer;
   }
   button:hover {
    background: #e03333;
   }
   #status {
    margin-bottom: 1rem;
    font-style: italic;
    min-height: 20px;
   }
   iframe {
    width: 100%;
    height: 400px;
    border: 1px solid #444;
    border-radius: 8px;
    background: white;
   }
  </style>
</head>
<body>
  <h1>AI POWERED ENGINEERING TOOL</h1>
```

```
<form id="generateForm">
  <label for="prompt">Website Description (min 10 characters):</label>
  <textarea
    id="prompt"
    name="prompt"
    minlength="10"
    required
    placeholder="Describe your website..."
  ></textarea>
  <label for="image">Upload an image (optional):</label>
  <input type="file" id="image" name="image" accept="image/*" />
  <button type="submit">Generate Preview</button>
</form>
<div id="status"></div>
<iframe
  id="previewFrame"
  sandbox="allow-scripts allow-same-origin"
></iframe>
<script>
  const form = document.getElementById("generateForm");
  const statusDiv = document.getElementById("status");
```

```
const iframe = document.getElementById("previewFrame");
form.addEventListener("submit", async (e) => {
  e.preventDefault();
  statusDiv.textContent = "Generating preview... please wait.";
  const formData = new FormData(form);
  try {
    const response = await fetch("/preview", {
      method: "POST",
      body: formData,
    });
    if (!response.ok) {
      const errorData = await response.json();
      throw new Error(errorData.error || "Failed to generate preview");
    }
    const data = await response.json();
    // Load the HTML into iframe by writing into its document
    const doc = iframe.contentWindow.document;
    doc.open();
    doc.write(data.previewHTML);
    doc.close();
    statusDiv.textContent = "Preview generated below!";
  } catch (err) {
```

```
        statusDiv.textContent = "Error: " + err.message;
        iframe.srcdoc = ""; // clear iframe on error
      }
    });
  </script>
 </body>
</html>
```

BACKEND

```
const express = require("express");
const fs = require("fs");
const path = require("path");
const axios = require("axios");
const dotenv = require("dotenv");
const multer = require("multer");
const archiver = require("archiver");
dotenv.config();

const app = express();
const port = 3000;
// Multer setup
const storage = multer.diskStorage({
  destination: (req, file, cb) => cb(null, "uploads/"),
```

```
  filename: (req, file, cb) => {
    const uniqueSuffix = Date.now() + "-" + Math.round(Math.random() * 1e9);
    const ext = path.extname(file.originalname);
    cb(null, file.fieldname + "-" + uniqueSuffix + ext);
  },
});
const upload = multer({
  storage,
  limits: { fileSize: 5 * 1024 * 1024 },
  fileFilter: (req, file, cb) => {
    if (!file.mimetype.startsWith("image/"))
      return cb(new Error("Only image files allowed!"));
    cb(null, true);
  },
});
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(express.static(path.join(__dirname, "public")));
app.use("/uploads", express.static(path.join(__dirname, "uploads")));
const TOGETHER_API_URL = "https://api.together.xyz/v1/chat/completions";
const MODEL = "mistralai/Mixtral-8x7B-Instruct-v0.1";
```

```
// Helper: parse code blocks more flexibly
function extractCodeBlock(content, lang) {
  const regex = new RegExp("```" + lang + "\\s*([\\s\\S]*?)```", "i");
  const match = content.match(regex);
  if (match && match[1]) return match[1].trim();
  return null;
}
// POST /generate - returns ZIP file with generated site files
app.post("/generate", upload.single("image"), async (req, res) => {
  const prompt = req.body.prompt;
  if (!prompt || prompt.trim().length < 10) {
    return res
      .status(400)
      .json({ error: "Prompt must be at least 10 characters." });
  }
  let imageUrl = "";
  if (req.file) {
    imageUrl = `${req.protocol}://${req.get("host")}/uploads/${
      req.file.filename
    }`;
  }
  // Strong prompt asking for code blocks with exact format
```

```
  const fullPrompt = imageUrl
    ? `${prompt}. Include this image in the design: ${imageUrl}`
    : prompt;
  const finalPrompt = `You are an expert web developer. Based on this description, create a fully responsive HTML5 website with separate files: HTML, CSS, and JavaScript. Output ONLY in the following markdown format (no explanations):
\`\`\`html
<!-- HTML content here -->
\`\`\`
\`\`\`css
/* CSS content here */
\`\`\`
\`\`\`js
// JavaScript content here
\`\`\`
Description: ${fullPrompt}`;
  try {
    const response = await axios.post(
      TOGETHER_API_URL,
      {
        model: MODEL,
        messages: [{ role: "user", content: finalPrompt }],
```

```
    temperature: 0.7,
    max_tokens: 2048,
  },
  {
    headers: {
      Authorization: `Bearer ${process.env.TOGETHER_API_KEY}`,
      "Content-Type": "application/json",
    },
  }
);
const aiResponse = response.data.choices[0].message.content;
// Extract code blocks
const htmlContent = extractCodeBlock(aiResponse, "html");
const cssContent = extractCodeBlock(aiResponse, "css");
const jsContent = extractCodeBlock(aiResponse, "js");
if (!htmlContent || !cssContent || !jsContent) {
  return res.status(500).json({
    error: "Failed to parse AI response into files.",
    aiResponseRaw: aiResponse,
  });
}
```

```
    // Create ZIP archive and stream it as response
    res.setHeader("Content-Type", "application/zip");
    res.setHeader("Content-Disposition", "attachment; filename=website.zip");
    const archive = archiver("zip", { zlib: { level: 9 } });
    archive.on("error", (err) => {
      throw err;
    });
    archive.pipe(res);
    archive.append(htmlContent, { name: "index.html" });
    archive.append(cssContent, { name: "styles.css" });
    archive.append(jsContent, { name: "scripts.js" });
    await archive.finalize();
  } catch (error) {
    console.error("Generation failed:", error.response?.data || error.message);
    res.status(500).json({ error: "Failed to generate website" });
  }
});

// POST /preview - returns combined preview HTML (CSS and JS inline)
app.post("/preview", upload.single("image"), async (req, res) => {
  const prompt = req.body.prompt;
  if (!prompt || prompt.trim().length < 10) {
```

```
    return res
      .status(400)
      .json({ error: "Prompt must be at least 10 characters." });
  }
  let imageUrl = "";
  if (req.file) {
    imageUrl = `${req.protocol}://${req.get("host")}/uploads/${
      req.file.filename
    }`;
  }
  const fullPrompt = imageUrl
    ? `${prompt}. Include this image in the design: ${imageUrl}`
    : prompt;
  const finalPrompt = `You are an expert web developer. Based on this description, create a fully responsive HTML5 website with separate files: HTML, CSS, and JavaScript. Output ONLY in the following markdown format (no explanations):
\`\`\`html
<!-- HTML content here -->
\`\`\`
\`\`\`css
/* CSS content here */
\`\`\`
```

```
\`\`\`js
// JavaScript content here
\`\`\`
Description: ${fullPrompt}`;
  try {
    const response = await axios.post(
      TOGETHER_API_URL,
      {
        model: MODEL,
        messages: [{ role: "user", content: finalPrompt }],
        temperature: 0.7,
        max_tokens: 2048,
      },
      {
        headers: {
          Authorization: `Bearer ${process.env.TOGETHER_API_KEY}`,
          "Content-Type": "application/json",
        },
      }
    );
    const aiResponse = response.data.choices[0].message.content;
```

```
    const htmlContent = extractCodeBlock(aiResponse, "html");
    const cssContent = extractCodeBlock(aiResponse, "css");
    const jsContent = extractCodeBlock(aiResponse, "js");
    if (!htmlContent || !cssContent || !jsContent) {
      return res.status(500).json({
        error: "Failed to parse AI response into files.",
        aiResponseRaw: aiResponse,
      });
    }
    const combinedHTML = `<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<style>${cssContent}</style>
<title>Preview</title>
</head>
<body>
${htmlContent}
<script>${jsContent}</script>
</body>
</html>`;
```

```
    res.json({ previewHTML: combinedHTML });
  } catch (error) {
    console.error(
      "Preview generation failed:",
      error.response?.data || error.message
    );
    res.status(500).json({ error: "Failed to generate preview" });
  }
});
app.use((err, req, res, next) => {
  if (err instanceof multer.MulterError)
    return res.status(400).json({ error: err.message });
  else if (err) return res.status(400).json({ error: err.message });
  next();
});
app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}`);
});
```

APPENDIX B

SCREENDHOTS

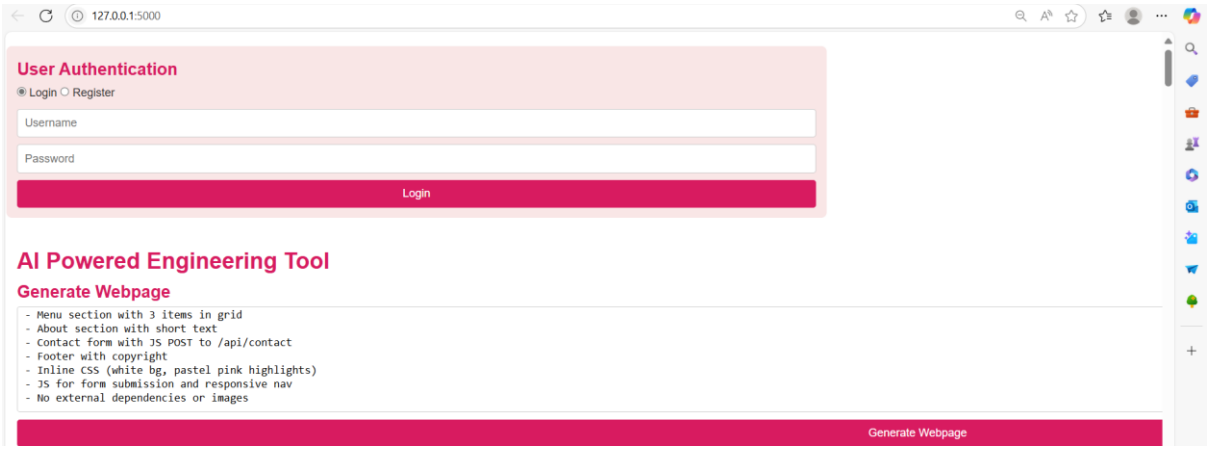

Fig.B.1 User Interface

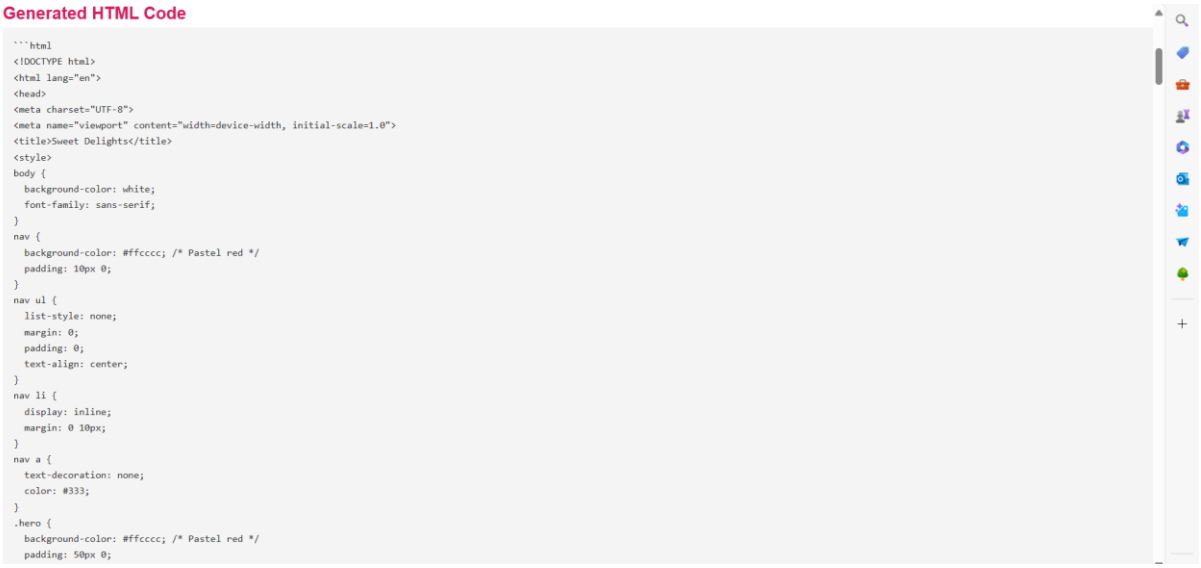

Fig.B.2 Generating Preview

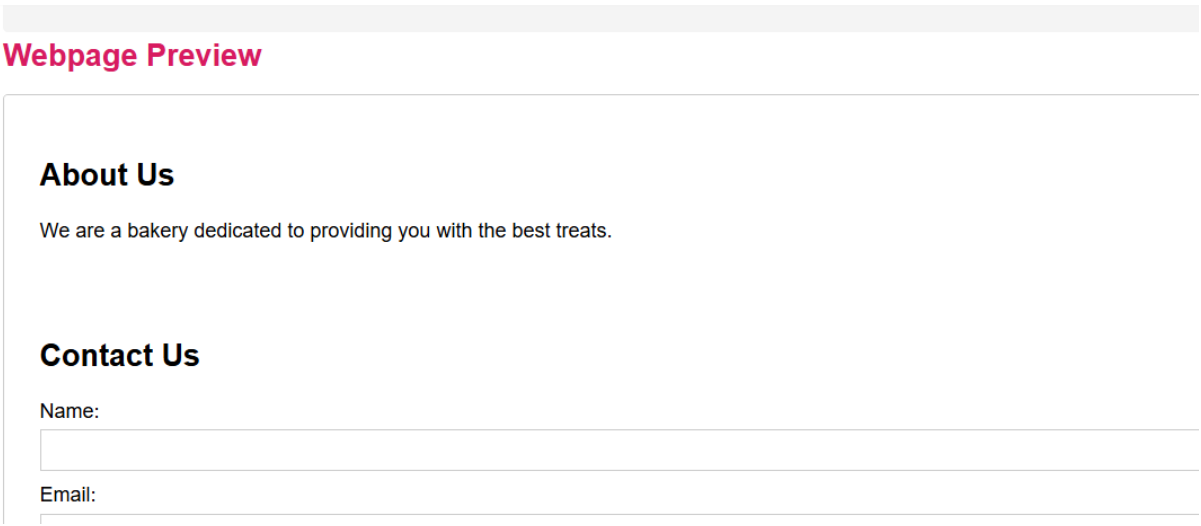

Fig.B.3 Website Preview

REFERENCES

1. Praveen, A., & Sharma, N., "AI-Based Event Management System: Automating Planning and Real-Time Monitoring. International Journal of Emerging Technology and Advanced Engineering" ,2024.

2. Abdallah, A., Bakhit, R., & Hassan, R., "Smart Surveillance System Using AI and IoT for Enhanced Home Security", International Journal of Engineering and Technology, 2024.

3. Zhang, F., Li, S., & Han, K., "Streamlit-Based Applications for AI-Driven Workflow Automation," Journal of Web and Software Tools, 2024.

4. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., & Scialom, T., "LLaMA: Open and Efficient Foundation Language Models" , Meta AI Research, 2024.

5. Hang, Z., Yin, S., & Li, Y.,"A Deep Learning Approach for Automatic Poster Generation in Education Events.", IEEE Access, 2024.

6. Chen, T., Liu, J., & Zhao, R., "Event Promotion with Generative Visual Content: AI-Based Poster Design and Text-to-Image Tools", AI and Society , 2023.

7. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., & Chen, M., "Hierarchical Text-Conditional Image Generation with CLIP Latents", OpenAI Technical Report, 2023.

8. Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., & Salimans, T., "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding", 2023.

9. Wang, Y., Lin, T., & Zhao, H., "Automated Event Scheduling and Management using Machine Learning Techniques", International Journal of Advanced Computer Science, 2022.

10. Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., & Sutskever, I. "Learning Transferable Visual Models from Natural Language Supervision (CLIP)", OpenAI, 2022.