

A Mini Project Report On

THEME - BASED INTERIOR DESIGNER

Submitted in Partial fulfillment of the requirements for the award of the Degree of

Bachelor of Technology

In

Department of Computer Science and Engineering

By

Susanna Bhukya	22241A05D9
Harshini Kanne	22241A05F6
Namala Praneetha Reddy	22241A05G8
Apoorva Satakolla	23245A0520

Under the Esteemed guidance of

Dr. G. S. Bapiraju

Professor



Department of Computer Science and Engineering

GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Autonomous)

Bachupally, Kukatpally, Hyderabad, Telangana, India, 500090

2024-2025



GOKARAJU RANGARAJU
INSTITUTE OF ENGINEERING AND TECHNOLOGY
(Autonomous)

CERTIFICATE

This is to certify that the Mini Project entitled “**Theme - Based Interior Kits**” is submitted by **Susanna Bhukya(22241A05D9), Harshini Kanne(22241A05F6), Namala Praneetha Reddy(22241A05G8), Apoorva Satakolla(23245A0520)**, Partial fulfillment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in Computer Science and Engineering during the academic year **2024-2025**.

INTERNAL GUIDE

Dr. G. S. Bapiraju
Professor

HEAD OF THE DEPARTMENT

Dr. B. SANKARA BABU
Professor

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

Many people helped us directly and indirectly to complete our project successfully. We would like to take this opportunity to thank one and all. First, we wish to express our deep gratitude to our internal guide **Dr. G. S. Bapiraju, Professor**, Department of CSE for his support in the completion of our project report. We wish to express our honest and sincere thanks to **S Ritwika and D Pavithra** for coordinating in conducting the project reviews. We express our gratitude to **Dr. B. Sankara Babu, HOD**, department of CSE for providing resources, and to the principal **Dr. J. Praveen** for providing the facilities to complete our Mini Project. We would like to thank all our faculty and friends for their help and constructive criticism during the project completion phase. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

Susanna Bhukya 22241A05D9

Harshini Kanne 22241A05F6

Namala Praneetha Reddy 22241A05G8

Apoorva Satakolla 23245A0520

DECLARATION

We hereby declare that the Mini Project entitled “**Theme - Based Interior Designer**” is the work done during the period from 16th Jan 2025 to 13th May 2025 and is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering from **Gokaraju Rangaraju Institute of Engineering and Technology**. The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

Susanna Bhukya	22241A05D9
Harshini Kanne	22241A05F6
Namala Praneetha Reddy	22241A05G8
Apoorva Satakolla	23245A0520

	Table of Contents	
Chapter	TITLE	Page No
	Abstract	1
1	Introduction	2
2	Literature Survey	3
3	System Requirements	6
	3.1 Software Requirements	6
	3.2 Hardware Requirements	9
	3.3 Methodology & Data Set	10
4	Proposed Approach, Modules Description and UML Diagrams	13
	4.1 Modules	13
	4.2 UML Diagrams	15
5	Implementation, Experimental Results &Test Cases	20
6	Conclusion and Future Scope	47
7	References	49
	Appendix i) Full Paper-Publication Proof ii) Snapshot Of the Result iii) Optional (Like Software Installation /Dependencies/ pseudo code)	

	LIST OF FIGURES	
Fig No	Fig Title	Page No
4.2.1	Use Case Diagram	16
4.2.2	Class Diagram	18
4.2.3	Sequence Diagram	19
5.2.1	Register Page	41
5.2.2	Login Page	41
5.2.3	Select a Room Page	42
5.2.4	Furn AI Image Page	42
5.2.5	Dining Room Image Page	43
5.2.6	Hall Image Page	43
5.2.7	Living Room Image Page	44
5.2.8	Study Room Image Page	44

	LIST OF TABLES	
Table No	Table Name	Page No
5.3	Test Cases	45

ABSTRACT

Interior design frequently presents difficult problems for those with little time, money, or knowledge. "Theme-Based Interior Designer," our project, presents an AI-powered way to streamline and personalize the interior design process. Users can upload a picture of an empty room, enter its measurements, and specify their aesthetic preferences thanks to the system's integration of artificial intelligence (AI) and machine learning (ML). The system responds by providing intelligent, well-thought-out design suggestions that include suggestions for furniture placement, color palettes, lighting configurations, and decor.

The software creates aesthetically pleasing and space-efficient layouts by dynamically analyzing user inputs and room variables. It continuously changes by utilizing machine learning algorithms to learn from user feedback and new design trends. Over time, this leads to recommendations that are more precise and tailored to the individual. The approach, which is intended to be both cost-effective and user-friendly, opens up professional-caliber design to a wider audience.

With applications in interior design consulting, real estate staging, and home remodeling, "Theme-Based Interior Kits" enable users—homeowners, designers, and sellers—to easily convert spaces into visually pleasing, useful spaces.

CHAPTER 1

INTRODUCTION

Enhancing the functionality and elegance of a living or working environment is mostly dependent on interior design. However, a lack of skill, time constraints, or financial limitations make it difficult for many people to design interior spaces effectively. Additionally, traditional interior design services can be expensive and time-consuming, which prevents a wide range of people from using them. Our project, "Theme-Based Interior Designer," offers a creative way to address such issues by using machine learning (ML) and artificial intelligence (AI) technology to personalize and expedite the interior design process.

With the help of this clever technology, customers may enter the size of their empty space, upload a photo of it, and choose their preferred layout. In exchange, it offers AI-generated recommendations for the ideal placement of furniture, color schemes, lighting configurations, and decorative accents based on the user's preferences and available space. Using machine learning algorithms that analyze user feedback and changing design trends, the system continuously modifies and enhances its suggestions. In general, StyleGAN is used for style transfer in existing interior design recommender systems, with a special emphasis on improving the room's textures, colors, and visible appearance. The absence of precise control over furniture placement in those systems, however, usually leads to improbable arrangements that deviate from the actual spatial arrangement. Additionally, their ability to create realistic and space-aware interior designs is limited because they only work with 2D images and are unable to remember depth or room proportions.

In order to overcome those constraints, our suggested method combines StyleGAN with Stable Diffusion, creating a hybrid version that not only enhances style transfer but also ensures sensible and practical furniture placement within the available area. The AI system understands the room layout in three dimensions by integrating ControlNet for depth estimation. This enables it to produce useful furniture configurations that perfectly fit the available area, avoiding problems like object overlap or incorrect scale.

CHAPTER 2

LITERATURE SURVEY

G. Balram, P. Sneha, and P. Sahithi's 2025 paper "**AI Powered Interior Design: A Generative Approach to Room Styling Based on Image Inputs**" describes a design technology system that makes use of the Replicate API in conjunction with a contemporary tech stack that includes Next.js, Tailwind CSS, Neon Database, and Drizzle Studio. The system provides generative design recommendations based on user selections and room photos. Although it works well for basic applications, it has limits when it comes to handling complex room layouts, needs to be optimized for increased computational efficiency, and only supports a limited range of established design styles. Notwithstanding these drawbacks, it provides a solid basis for web-based interactive design platforms, particularly for novices and homeowners seeking quick fixes.

Yiyan Fan, Yang Zhou, and Zheng Yuan's 2024 paper "**Interior Design Evaluation Based on Deep Learning: A Multi-Modal Fusion Evaluation Mechanism**" focuses on assessing interior designs through transformer-based 3D question answering (3DQA) and 3D point cloud processing. Sentiment analysis is used to evaluate user perception. The ABSA, ScanNet, and ScanQA datasets are used in the work. However, the lack of real-world and user-based evaluation, high computing cost, and lack of dataset transparency all impede its practical deployment. Although the method is new and important from an academic standpoint, hardware and data constraints prevent it from being commercially viable.

H. D. Shonali Romeshika, D. C. Deeghayu, and Hashini Herath's paper "**Home Interior Design Suggestion System using Deep Learning**" (2024) suggests a design suggestion system that makes use of ControlNet, YOLOv8, DeepFill, Stable Diffusion, and DenseNet121. It processes 475 photos especially for YOLOv8 and more than 1000 photographs per class. Despite using deep learning and image segmentation techniques to provide respectable results, the model's efficacy is mostly dependent on the accuracy of the segmentation. Furthermore, the system does not fully incorporate user preferences, which restricts the degree of personalization. Nevertheless, it demonstrates how object detection and inpainting may be combined to create visually harmonious environments.

Ch. Adharsh, A. Ravalika, A. Sumanth, Ch. Bharath, and Dr. Madhavi Pingili's **"AI Based Interior Designing App"** (2024) presents an AI-powered system that uses StyleGAN to produce realistic interior designs. The model evaluates image quality and coherence using SSIM, MSE, and perceptual loss (VGG) as evaluation measures. Reproducibility is limited because no specific dataset used for training is mentioned, despite the datasets being cited from ISI Web of Knowledge, Science Direct, and Springer. Furthermore, the system struggles with spatial realism, lacks a well-defined ethical AI framework, and occasionally generates unsuitable room layouts. It also has trouble integrating into professional design workflows. These drawbacks limit its use in situations requiring accuracy in the actual world.

According to Guohui Fan and Chen Guo's **"Personalized Recommendation Algorithm of Interior Design Style Based on Local Social Network"** (2023), the Apriori algorithm and 3D model data are used to create a recommendation system. With a processing time of 1.1 minutes and an accuracy of 82%, the model uses user behavior data and the 3D-FRONT dataset to customize design recommendations. The method lacks mechanisms for real-time preference adjustments and has large computing costs, despite its promise for customisation. To improve responsiveness and lower latency, it also requires further optimization. Because of this, it is less appropriate for dynamic applications that require quick integration of user feedback.

A. Rayyan Ul Haq and B. Mudasir Hanif Shaikh's 2023 paper **"Interior Design 2.0: Style Transfer by Style Mixing of Real Images Using StyleGANs"** investigates the application of StyleGANs for style mixing and transfer in interior images using the LSUN Bedroom dataset, which has more than 300,000 photos. With an emphasis on visual quality, the method assesses output using VGG perceptual loss, MSE, and SSIM. But following style mixing, the encoder's performance is rather poor, frequently leading to misplaced items. Moreover, users cannot regulate the degree of style transfer, which results in irregular or undesirable changes. Despite being unique, the system's accuracy and lack of interactivity make it difficult to use in professional settings.

Hyun Jeong, Young Chae Kim, and Seunghyun Cha's 2023 paper "**Gen AI and Interior Design Representation: Applying Design Styles Using Fine-Tuned Models**" investigates the use of fine-tuned Stable Diffusion models to produce interior designs that are stylistically appropriate. In order to improve style adherence, the authors optimize hyperparameters using a proprietary dataset made up of interior design photos and language prompts that are exclusive to a certain style. Even though it produces visually striking images, the model has trouble with complex or hybrid styles and needs to be manually adjusted for reliable outcomes. Additionally, the model's scalability in larger real-world applications is impacted by the dataset's restricted size, which limits the variety of styles it can generalize to.

K. Rithish's "**Minimalism in Interior Design**" (2022) offers a theoretical investigation of minimalist interior design, emphasizing design components including lighting, material selection, and space partition. The famous Farnsworth House serves as the case study for the examination, which provides insights into the practical and aesthetic elements of minimalist architecture. However, the study's scalability and reproducibility are constrained by the lack of datasets or empirical evaluation. It is less appropriate for computational modeling or integration into AI-driven systems due to its sole focus on residential environments and lack of quantitative validation.

An earlier version of the research presented in Paper 7 may be found in "**INTERIOR DESIGN 2.0: Style Transfer by Style Mixing of Real Images Using StyleGANs**" (2020) by A. Rayyan Ul Haq, B. Mudasir Hanif Shaikh, and C. Syed Sameer Nadeem. This version uses the LSUN Bedroom dataset to investigate style transfer using StyleGAN's style mixing technique. Real picture encoding frequently results in mismatches, and deep style mixing frequently generates distortions, such as twisted or dislocated objects, despite the encoder architecture's ability to partially translate real-world images into the GAN's latent space. Practical deployment is hampered by these technical constraints, especially in situations where output control and fidelity are necessary.

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 Software Requirements

The Theme-Based Interior Kits device is advanced to generate intelligent, visually appealing, and personalised indoors designs through integrating cutting-edge internet technology with Python-primarily based totally deep mastering models. Users can add pics in their rooms, enter dimensions, and pick out layout choices via a easy and interactive frontend constructed the usage of HTML, CSS, JavaScript, and the Canvas API. On the backend, Flask manages conversation among the interface and AI models, at the same time as effective frameworks like PyTorch and Hugging Face Transformers (the usage of Segformer and DPT) cope with intensity estimation and semantic segmentation. This mixture allows the device to supply real-time, superb layout guidelines which might be each spatially correct and tailor-made to person choices.

3.1.1 Python (Backend):

The backend of the Theme-Based Interior Kits gadget is evolved the use of Python, serving because the middle engine that handles AI-primarily based totally layout logic, version execution, and statistics processing. Python`s wealthy atmosphere of libraries and frameworks makes it distinctly appropriate for responsibilities concerning laptop vision, device learning, and net API development.

Image Preprocessing: Room pictures uploaded through customers are processed the usage of OpenCV and incorporated canvas gear to beautify photo readability and put together them for version inference. This consists of resizing, layout conversion, and optimization for AI fashions.

Depth Estimation: The machine leverages the DPT (Dense Prediction Transformer) version from Hugging Face Transformers to research intensity from an unmarried room photo. This permits the machine to estimate spatial shape and help sensible item scaling and location withinside the 3-D space.

Semantic Segmentation: The Segformer version is used to phase key regions withinside the

room along with walls, floors, windows, and furniture. This permits context-conscious layout generation, making sure that decor factors are located logically withinside the room format.

Design Generation and Integration: Based at the intensity map, segmentation output, and consumer preferences, the backend generates a visible format the usage of rule-primarily based totally good judgment or connects with generative fashions for visible enhancement. Final designs are rendered and organized for frontend show or download.

Web API Handling: Flask serves because the internet framework that connects the frontend and backend. It handles HTTP requests, routes consumer inputs (along with pictures and layout preferences), and returns generated layout results. It additionally manages server-facet good judgment and document handling.

3.1.2 HTML, CSS AND JavaScript (Fronddend Development):

The frontend of the Theme-Based Interior Kits device is designed to offer an interactive and user-pleasant interface that permits customers to enter information and examine generated designs seamlessly. It is constructed the use of HTML, CSS, JavaScript, and greater with equipment just like the Canvas API and html2canvas for real-time layout rendering and photo capturing.

HTML (HyperText Markup Language): HTML the structural basis of the application`s interface. It defines the format of key factors including enter paperwork, document add components, headers, buttons, and choice fields. It lets in customers to go into room measurements, add pics, and select aesthetic options like colour schemes or layout themes. The device guarantees clean consumer navigation via a step-by-step shape shape that courses customers via the whole layout process.

CSS (Cascading Style Sheets): CSS complements the visible aesthetics and responsiveness of the application. It guarantees a steady and attractive layout the usage of well-based colour palettes, typography, spacing, and layouts. Advanced CSS capabilities like Flexbox and CSS Grid are used to create a responsive layout that adapts to diverse display screen sizes - whether or not on desktops, tablets, or smartphones. Custom topic assist additionally permits the UI to conform visually primarily based totally at the consumer`s decided on indoors style.

JavaScript: JavaScript drives the interactivity of the device. It plays client-facet validation of consumer inputs - making sure that uploaded documents are legitimate pics and that measurements are appropriately entered. It additionally manages asynchronous communicate with the backend the usage of Fetch API or AJAX, permitting shape submissions and updates with out web page reloads. Additionally, the mixing of the Canvas API and html2canvas lets in for taking pictures generated designs as pics, permitting customers to preview, download, or percentage their custom designed layouts in real-time.

3.1.3 MySQL (Database Management):

The Theme-Based Interior Kits gadget employs MySQL as its Relational Database Management System (RDBMS) to successfully store, retrieve, and manipulate established statistics generated at some stage in the person interplay and AI layout processes. MySQL's reliability, scalability, and seamless integration with Python-primarily based totally frameworks make it an excellent preference for each improvement and deployment environments. MySQL performs an essential position in organizing and retaining the subsequent center datasets:

User Information: The gadget shops important person info along with names, e mail addresses, and securely hashed passwords. This allows person authentication, consultation management, and customized get admission to to stored layout histories and alternatives.

Room Data: When a person uploads a room picture and inputs dimensions, these statistics is saved withinside the database. Each access consists of the picture record path, width and top of the room, and elective attributes like lights type, ground material, or window positions-all of which affect AI-pushed layout generation.

Style Preferences: The database logs person-decided on layout factors inclusive of colour schemes, fixtures styles, and typical themes (e.g., modern, traditional, minimalist). These alternatives are consultation-precise and make contributions to producing tailor-made outputs, in addition to permitting continuity if a person returns later.

Generated Design Outputs: Once a layout is generated with the aid of using the AI, the gadget statistics metadata along with the output picture path, timestamp of generation, and related layout tags or parameters. This lets in customers to view, download, or revisit their preceding designs conveniently.

3.2 Hardware Requirements

The Theme-Based Interior Kits device wishes each light-weight client-aspect gadgets for person interplay and effective server-aspect assets for AI version execution. In addition to facilitating the advent and implementation of system studying fashions, the hardware ought to allow customers to add images and retrieve created designs with ease.

Server Hardware: The significant processing unit is in rate of web website hosting the AI fashions and offering cease customers with the indoors layout device through an internet interface.

CPU: For the green dealing with of a couple of person requests, photograph processing operations, and backend logic, a multi-center CPU is required.

RAM: Stores records and commands quickly for quicker get admission to via way of means of the CPU.

Storage (SSD): Non-risky garage tool used for storing device documents, databases and different records.

Interface for Networks: To assure non-stop touch among customers and the server, a reliable and speedy community interface card (NIC) is necessary, specifically whilst importing massive photograph documents and turning in created outputs.

Client Devices: Common internet-enabled gadgets may be utilized by cease customers, which include actual property agents, designers, and homeowners, to get admission to the application: Laptops and computing device computers: To take a look at high-decision layout outputs and post room images.

- **Tablets and smartphones:** To have interaction with the UI whilst at the move and hastily preview options. Modern browsers like Chrome, Edge, or Firefox ought to be set up on those gadgets a good way to absolutely make use of the frontend capabilities of the device.
- **Laptops and desktop computers:** To take a look at high-decision layout outputs and post room images.

3.3 Methodology and Data Set

3.3.1 Methodology

The Theme-Based Interior Kits machine follows a hybrid, AI-pushed technique that mixes semantic segmentation, intensity estimation, text-to-photo generation, and consumer choice customization to supply intelligent, practical, and visually compelling indoors layout recommendations. The pipeline is designed to provide a continuing consumer enjoy whilst making sure that every layout is spatially correct and tailor-made to character tastes. Below is a step-with the aid of using-step breakdown of the process:

3.3.1.1. Image Upload and Room Information Input

The consumer starts with the aid of using importing an image of an empty room and coming into the corresponding room dimensions (period and width). This facts paperwork the inspiration for the AI to apprehend the bodily format of the space.

3.3.1.2. Preference Collection

Users are precipitated to pick out their desired aesthetic alternatives consisting of shadeation schemes, room styles (e.g., minimalist, modern, classic), and layout elements. These possibilities are saved and used to manual the AI version in producing personalised layout outputs.

3.3.1.3. Semantic Segmentation with Segformer

The uploaded photo is processed thru the Segformer version (through Hugging Face Transformers) to carry out semantic segmentation. This allows the machine understand and isolate numerous additives withinside the room consisting of walls, floors, windows, and furniture, supplying contextual awareness.

3.3.1.4. Depth Estimation with DPT

To apprehend the room`s spatial intensity, the DPT (Dense Prediction Transformer) version is applied. It converts the 2D photo right into a intensity map, permitting the machine to generate 3D-conscious designs with correct item placement and scaling.

3.3.1.5. Design Generation the use of Stable Diffusion v1.5 (RunwayML)

With each the segmented photo and intensity map available, the machine makes use of RunwayML's stable-diffusion-v1.5 version to generate practical and style-unique indoors layout visuals primarily based totally on consumer-decided on issues and room context. The end result is a high-resolution, photorealistic layout output.

3.3.1.6. AI Output Visualization and User Interaction

The generated layout is lower back to the frontend, in which it is miles exhibited to the consumer the use of net technologies. The interface lets in customers to view, download, or decorate their designs in actual time. Feedback from consumer interactions may be logged to fine-song destiny version outputs and enhance personalization.

3.3.2 Data Sets

The Theme-Based Interior Kits gadget makes use of a well-prepared set of datasets to control person interactions, room data, fashion choices, AI-generated designs, and gadget hobby logs. These datasets together allow personalised experiences, guide AI version education or improvement, and make certain sturdy gadget overall performance.

3.3.2.1. User Information: This dataset shops information of people the use of the platform, which include homeowners, indoors designers, and actual property agents. Each person is assigned a completely unique ID to music their moves throughout classes. Stored attributes encompass complete name, electronic mail address, encrypted password, and role (e.g., admin or person). The dataset additionally statistics person hobby consisting of login timestamps, fashion choices, and formerly generated designs. This data helps steady authentication and permits the gadget to provide a customized layout revel in primarily based totally on beyond conduct.

3.3.2.2. Room Information: This dataset captures all important information about the person's room in which indoors layout pointers are to be applied. When a photograph is uploaded, the gadget statistics the report course, room dimensions (width and height), and assigns a completely unique room ID. Optional attributes consisting of floors type, window placement, and lighting fixtures choices (herbal or artificial) will also be included. Each room access is related to a selected person, permitting correct spatial evaluation and applicable layout technology primarily based totally at the bodily context.

3.3.2.3 Style Preferences Data: This dataset logs the classy choices decided on through customers for the duration of a session. These encompass indoors layout patterns like modern, minimalist, bohemian, or traditional, in addition to unique shadeation palettes, furnishings types, format configurations, and ornamental choices. These established records allow the AI version generate designs which can be aligned with person expectancies and permits the gadget to preserve consistency throughout classes through remembering preceding choices.

3.3.2.4. Design Output Data: After producing a layout the use of AI models (e.g., Segformer, DPT, Stable Diffusion), the gadget shops the output on this dataset. Each layout is related to a completely unique output ID and is related to the corresponding person and room. Metadata consisting of the report course of the generated photograph, inference duration, and the AI version model used are recorded. Feedback-associated records like person comments, down load counts, and advised upgrades also are stored, permitting iterative refinement of destiny outputs.

3.3.2.5. System Logs: The logging dataset guarantees operational transparency and gadget integrity through recording all most important frontend and backend events. Each log access carries timestamps and outlines of moves like report uploads, API calls, login attempts, and blunders messages. It additionally consists of version overall performance information and gadget reaction times. These logs are helpful for actual-time monitoring, debugging, and figuring out person conduct trends, that can manual in addition improvement and optimization.

CHAPTER 4

PROPOSED APPROACH, MODULES DESCRIPTION AND UML DIAGRAMS

Proposed Approach

The Theme-Based Interior Designer adopts a hybrid AI-pushed technique to generate realistic, customized, and spatially conscious indoors layout recommendations. The center structure combines text-to-photograph era, semantic segmentation, and intensity estimation to recognize room context and bring first rate visible designs aligned with person preferences. To attain this, the machine makes use of RunwayML's Stable Diffusion v1.5 version to generate photorealistic indoors designs primarily based totally on person-decided on subject matters and fashion prompts. For spatial understanding, the DPT (Dense Prediction Transformer) version is used to estimate intensity from 2D room images, permitting correct belief of room dimensions and spatial relationships. Simultaneously, Segformer, a semantic segmentation version from Hugging Face Transformers, identifies and separates key structural and purposeful factors of the room consisting of walls, floors, furnishings, and windows.

4.1 Modules

The Theme-Based Interior Kits gadget is split into some of interconnected modules, every of that is in rate of coping with a sure utility function. Together, those modules manage enter data, permit consumer interaction, and convey visually attractive and spatially accurate AI-primarily based totally indoors layout outputs. Scalability, ease of maintenance, and improved overall performance throughout all gadget additives are assured through the modular framework.

4.1.1 User Authentication and Management Module

This module handles consumer registration, login, consultation management, and profile facts storage. It makes use of stable encryption strategies including password hashing to make sure secure consumer authentication. Once authenticated, customers benefit get entry to to capabilities like importing room photos, deciding on alternatives, and viewing or downloading formerly generated designs. User-associated information, including email, consultation history, and stored alternatives, is saved in a MySQL database for personalised reports and interest tracking.

4.1.2 Room Upload and Processing Module

Users can add photos of empty rooms and enter the corresponding dimensions (duration and width) thru this module. The uploaded photo undergoes validation and is preprocessed for first-class the use of OpenCV (for layout checks, resizing, and optimization). This module guarantees the backend gets clean, established visible enter, that's important for correct segmentation, intensity estimation, and AI-primarily based totally layout technology.

4.1.3 Style Preferences Selection Module

This module permits customers to personalize their layout necessities with the aid of using deciding on shadeation palettes, layout issues (e.g., modern, minimalist, bohemian), lighting fixtures alternatives, and furnishings patterns. These choices are recorded and connected to the consumer and room ID, forming a key enter for the AI layout technology module. The personalised alternatives manual the visible technology process, making sure outputs align with the consumer`s aesthetic vision.

4.1.4 Design Generation Module

This is the center intelligence layer of the gadget. Segformer is used for semantic segmentation to perceive room additives like walls, floors, and furnishings. DPT is carried out for monocular intensity estimation, supporting the gadget apprehend the three-D shape of the room. Based at the segmented regions, intensity map, and consumer-decided on alternatives, Stable Diffusion v1.5 (thru RunwayML) is hired to generate a photorealistic indoors layout photo that suits the given context. This module guarantees that generated designs aren't simplest visually attractive however additionally recognize spatial constraints for practical item placement and scaling.

4.1.5 Download and Design Enhancement Module

Once the layout is generated, this module shows the output at the frontend the use of JavaScript and the Canvas API. Users can have interaction with the layout, make fundamental adjustments, and use html2canvas to down load the rendered format as a photo. Options for reinforcing or re-producing the layout with altered issues or patterns offer customers with flexibility and manipulate over the very last output.

4.2 UML Diagrams

4.2.1 Use Case Diagram

Actors

User: Positioned on the left side of the diagram, depicted as a stick figure, the User represents the primary customer who interacts with the system to generate and enhance interior designs based on personal preferences.

Admin: Displayed on the right side of the diagram, the Admin manages the backend and ensures system-wide consistency, such as updating themes and maintaining the database.

Use Cases

For User:

Upload Room Picture: The User uploads a photo of their room, which serves as the base image for interior design generation.

1. Take the Room Dimensions: The system allows the User to input room measurements, helping create accurate layout designs.

2. Select the Preferences: The User chooses their preferred themes, styles, and colors to guide the design generation process.

3. Generate Design: Based on uploaded images, dimensions, and selected preferences, the system uses ML or AI to generate a customized interior design.

4. Enhance the Design: The User has the ability to tweak or refine the generated design, allowing for more personalized results.

5. Download the Design: Once satisfied, the User can download the final design output for reference or implementation.

For Admin:

1. Update Theme Design: The Admin is responsible for adding new themes, updating existing ones, or modifying theme logic to keep the system fresh and relevant.

2. Manage Database: The Admin ensures proper management of backend databases, which may include storing room images, design templates, themes, and user data.

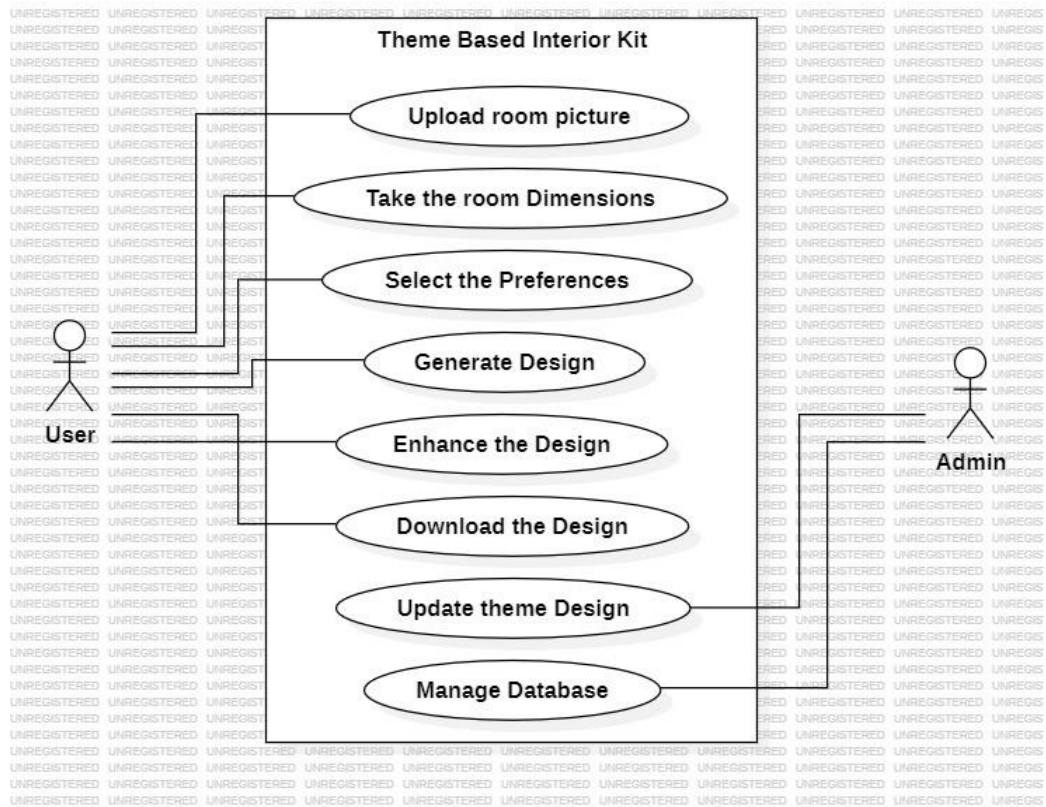


Fig 4.2.1: Use Case Diagram

4.2.2 Class Diagram

The system has six main classes: User, Admin, Room, Generate Design, Furniture, and Database. These are the core entities involved in generating and managing personalized interior designs.

- **Users:** This class represents the individual who accesses the system to generate personalized room designs.

Users can upload images, input room dimensions, and select design preferences.

- **Admin:** Admins maintain the system's backend by managing furniture databases and updating available themes.

They have access to system controls that ensure proper functioning and content updates.

- **Room:** Represents the user's room, capturing its dimensions and the uploaded photo.

This data is essential for tailoring the AI-generated design to the specific space.

- **Generate Design:** This class handles the processing of user preferences, images, and room specifications to generate an optimized interior layout.

It integrates AI/ML algorithms to suggest layouts, color palettes, and furniture placements.

- **Furniture:** Contains information about the design elements available for recommendation, such as furniture items, their types, dimensions, and style.

These elements are used in design suggestions to fill the room aesthetically and functionally.

- **Database:** Acts as the central repository for all project-related data including user profiles, room information, furniture items, and design results.

It ensures efficient storage and retrieval to support the design generation process.

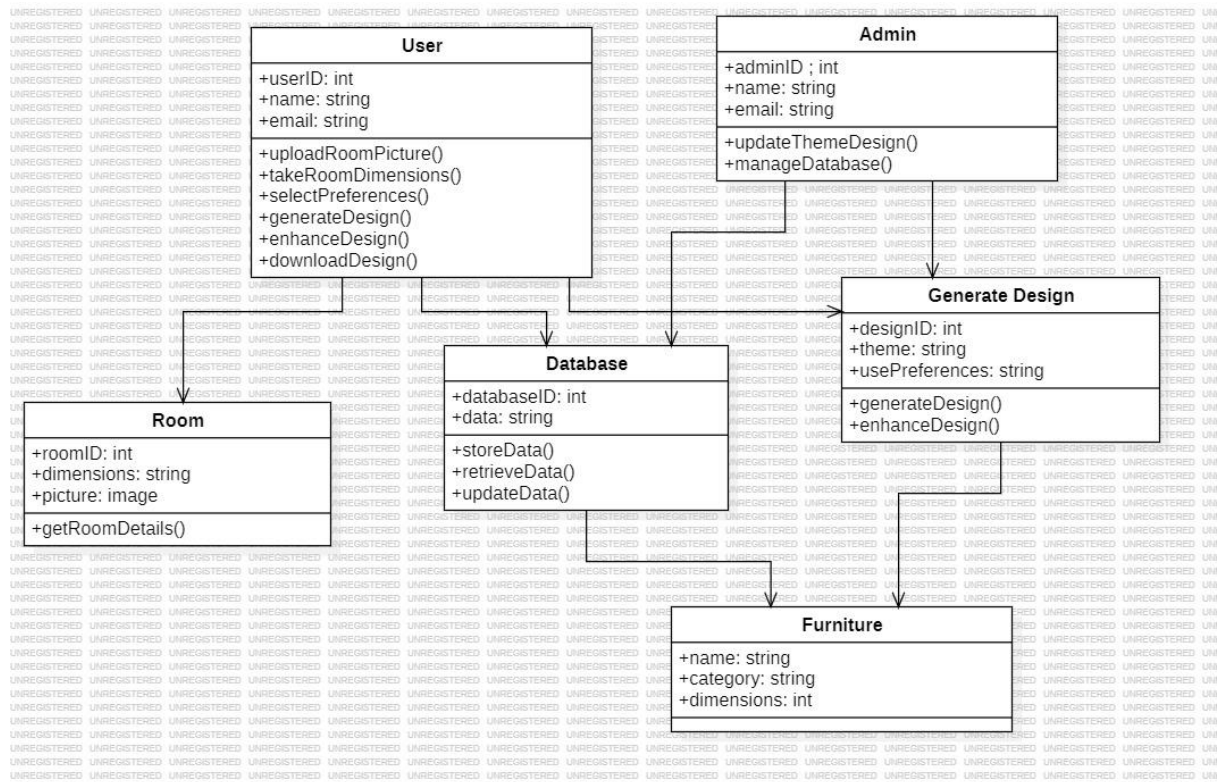


Fig 4.2.2: Class Diagram

4.2.3 Sequence Diagram

The sequence diagram illustrates the interaction between various components of the Theme-Based Interior Kit system to generate a customized interior design. The process starts with the User uploading a room image and entering the room dimensions. After that, the user selects their design preferences, which are then forwarded to the Design Generator for processing.

The Design Generator processes the inputs based on the image and preferences and generates a design. This processed data is returned and also sent to the Admin for storage. The Admin confirms and stores the design data in the Database, ensuring that the customized output is securely saved. Once the design is stored, the output is provided to the user.

The user has the option to enhance the design further or update the theme. Any updates made by the user are handled by the Admin and stored again in the Database. The final design is then made available for download, and the Admin continues to manage the database operations in the background.

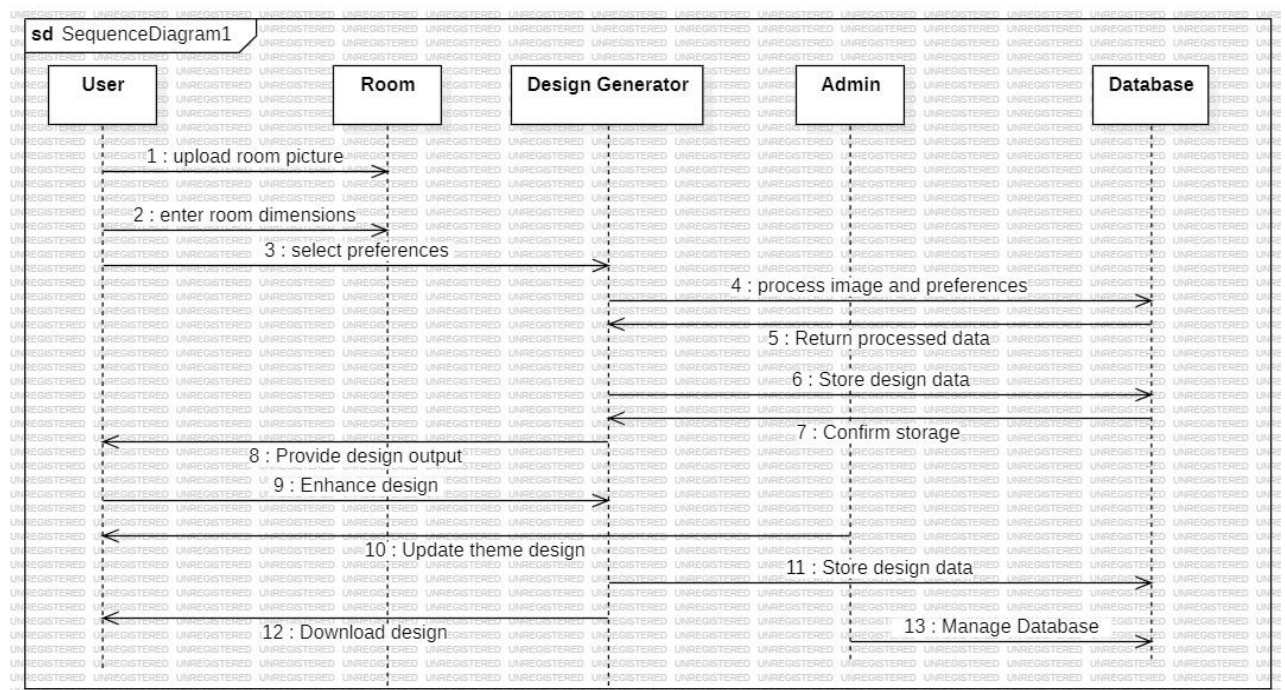


Fig 4.2.3: Sequence Diagram

CHAPTER 5

IMPLEMENTATION, EXPERIMENTAL RESULTS & TEST CASES

5.1 Implementation

5.1.1 Frontend code

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Login - Interior Kits</title>
  <style>
    @import url('https://fonts.googleapis.com/css2?family=Playfair+Display:wght@700&family=Raleway:wght@400;600&display=swap');

    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      font-family: 'Raleway', sans-serif;
      display: flex;
      align-items: center;
      justify-content: center;
      height: 100vh;
      background: url("{ url_for('static', filename='image/i1.jpeg') }") no-repeat center center;
      background-size: cover;
      position: relative;
    }

    body::before {
      content: "";
      position: absolute;
      top: 0;
      left: 0;
      width: 100%;
      height: 100%;
      background: rgba(0, 0, 0, 0.6);
      z-index: 1;
    }

    .content {
      display: flex;
      width: 80%;
      max-width: 1100px;
      justify-content: space-between;
      align-items: center;
    }

    h2 {
      font-size: 28px;
      margin-bottom: 20px;
      color: white;
      font-weight: 600;
    }

    .input-group {
      position: relative;
      margin: 10px 0;
    }

    input {
      width: 100%;
      padding: 12px;
      border: none;
      border-radius: 8px;
      background: rgba(255, 255, 255, 0.2);
      color: white;
      font-size: 16px;
      font-weight: bold;
      outline: none;
      transition: 0.3s;
    }

    input::placeholder {
      color: rgba(255, 255, 255, 0.7);
    }

    input:focus {
      background: rgba(255, 255, 255, 0.3);
      border: 2px solid rgba(255, 255, 255, 0.5);
    }

    button {
      width: 100%;
      padding: 12px;
      border: none;
      border-radius: 8px;
      background: linear-gradient(to right, #ffcc00, #ff8800);
    }
  </style>
</html>
```

```

button:hover {
  transform: scale(1.05);
  background: linear-gradient(to right, #ff8800, #ffcc00);
}

.message {
  color: #ff4d4d;
  font-size: 14px;
  margin-bottom: 10px;
}

.register-link {
  margin-top: 15px;
  font-size: 14px;
  color: rgba(255, 255, 255, 0.7);
}

.register-link a {
  color: #ffbb33;
  text-decoration: none;
  font-weight: bold;
  transition: 0.3s;
}

.register-link a:hover {
  color: rgb(28, 28, 180);
  text-decoration: underline;
}

@media (max-width: 768px) {
  .content {
    flex-direction: column;
    text-align: center;
  }

  .branding {
    max-width: 100%;
    margin-bottom: 20px;
  }
}

```

```

<div class="login-box">
  <form method="POST" action="{{ url_for('login') }}">
    <input type="password" name="password" placeholder="🔒 Password required" />
  </div>
  <button type="submit">Login</button>
</form>

```

```

<div class="register-link">
  <p>Not registered?<a href="{{ url_for('register') }}">Create an Account</a></p>
</div>

```

```

<script>
  function handleLogin(e) {
    e.preventDefault(); // Stop form from submitting

    const email = document.getElementById('email').value;
    const password = document.getElementById('password').value;
    const messageDiv = document.querySelector('.message');

    // Retrieve stored password using the email as the key
    const storedPassword = localStorage.getItem(email);

    if (storedPassword && password === storedPassword) {
      window.location.href = "selectroom.html"; // Redirect on successful login
    } else {
      messageDiv.textContent = "Invalid credentials. Try again.";
      messageDiv.style.color = '#ff4d4d';
    }
  }
</script>

<!-- <div class="register-link">
  <p>Not registered?<a href="register.html">Create an Account</a></p>
</div> -->
</div>
</body>
</html>

```

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Register - Interior Kits</title>
  <style>
    @import url('https://fonts.googleapis.com/css2?family=Playfair+Display:wght@700&family=Raleway:wght@400;600&display=swap');

    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      font-family: 'Raleway', sans-serif;
      display: flex;
      align-items: center;
      justify-content: center;
      height: 100vh;
      background: url('{{ url_for('static', filename='image/i1.jpeg') }}') no-repeat center center;

      background-size: cover;
      position: relative;
    }

    body::before {
      content: "";
      position: absolute;
      top: 0;
      left: 0;
      width: 100%;
      height: 100%;
      background: linear-gradient(to top right, transparent 49%, #000000 49%, #000000 51%, #ffffff 51%, #ffffff 53%);
      z-index: 1;
    }

    .content {
      display: flex;
      width: 80%;
      max-width: 1100px;
      justify-content: space-between;
      align-items: center;
    }

    .branding {
      flex: 1;
      color: white;
      text-align: left;
      max-width: 50%;
      font-family: 'Playfair Display', serif;
    }

    .branding h1 {
      font-size: 56px;
      font-weight: 800;
      line-height: 1.3;
      margin-bottom: 15px;
      letter-spacing: 1.5px;
    }

    .branding p {
      font-size: 26px;
      font-weight: 500;
      opacity: 0.9;
    }

    .branding span {
      color: #ffcc00;
    }

    .register-box {
      background: linear-gradient(135deg, rgba(50, 50, 50, 0.8), rgba(20, 20, 20, 0.8));
      padding: 40px;
      border-radius: 12px;
      box-shadow: 0px 4px 15px rgba(0, 0, 0, 0.6);
      width: 360px;
      backdrop-filter: blur(15px);
      border: 2px solid rgba(255, 255, 255, 0.2);
      text-align: center;
    }

    h2 {
      font-size: 28px;
      margin-bottom: 20px;
    }
  </style>
</head>
</html>

```

```

input:focus {
  background: linear-gradient(to right, #fffcc0, #fff880);
  border: 2px solid #fffcc0;
}

button {
  width: 100%;
  padding: 12px;
  border: none;
  border-radius: 8px;
  background: linear-gradient(to right, #fffcc0, #fff880);
  color: white;
  font-size: 18px;
  cursor: pointer;
  font-weight: bold;
  transition: 0.3s;
  box-shadow: 0px 4px 10px #fffcc0;
}

button:hover {
  transform: scale(1.05);
  background: linear-gradient(to right, #fff880, #fffcc0);
}

.message {
  color: #ff4d4d;
  font-size: 14px;
  margin-bottom: 10px;
}

.login-link {
  margin-top: 15px;
  font-size: 14px;
  color: #fffcc0;
}

.login-link a {
  color: #ffbb33;
  text-decoration: none;
  font-weight: bold;
}

```

```

</style>
</head>
<body>
  <div class="content">
    <div class="branding">
      <h1>Transform Your Home with <span>Interior Kits</span></h1>
      <p>The accessible and simple way of interior design. Start your journey with us.</p>
    </div>

    <div class="register-box">
      <h2>Register</h2>

      {% with messages = get_flashed_messages() %}
      {% if messages %}
        <div class="message">{{ messages[0] }}</div>
      {% endif %}
      {% endif %}

      <form action="{{ url_for('register') }}" method="POST">
        <div class="input-group">
          <input type="email" name="email" placeholder="&#x2709; Email" required />
        </div>
        <div class="input-group">
          <input type="password" name="password" placeholder="&#x1F512; Password" required />
        </div>
        <div class="input-group">
          <input type="password" name="confirm_password" placeholder="&#x1F512; Confirm Password" required />
        </div>
        <button type="submit">Register</button>
      </form>

      <div class="login-link">
        <p>Already have an account? <a href="{{ url_for('login') }}">Login</a></p>
      </div>
    </div>
  </body>
</html>

```

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Select a Room</title>
  <style>
    @import url('https://fonts.googleapis.com/css2?family=Playfair+Display:wght@700&family=Raleway:wght@400;600&display=swap');

    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      font-family: 'Raleway', sans-serif;
      background: url("{ url_for('static', filename='image/i2.jpeg') }") no-repeat center center;
      background-size: cover;
      display: flex;
      flex-direction: column;
      justify-content: center;
      align-items: center;
      height: 100vh;
      position: relative;
      overflow: hidden;
    }

    body::before {
      content: "";
      position: absolute;
      top: 0; left: 0;
      width: 100%; height: 100%;
      background: linear-gradient(to right, #ffcc00, #ff8800);
      z-index: 1;
    }

    h2 {
      font-size: 40px;
      font-weight: bold;
      color: white;
      text-shadow: 2px 2px 10px rgba(255, 255, 255, 0.7);
      position: relative;
    }

    .button-container {
      display: grid;
      grid-template-columns: repeat(3, 1fr); /* 3 buttons per row */
      gap: 15px;
      width: 80%;
      max-width: 900px;
      text-align: center;
      position: relative;
      z-index: 2;
    }

    button {
      padding: 15px;
      border: none;
      border-radius: 8px;
      background: linear-gradient(to right, #ffcc00, #ff8800);
      backdrop-filter: blur(12px);
      color: white;
      font-size: 18px;
      font-weight: bold;
      cursor: pointer;
      transition: 0.3s ease-in-out;
      text-transform: uppercase;
      box-shadow: 0px 4px 10px rgba(255, 255, 255, 0.1);
      width: 100%;
      max-width: 220px;
    }

    button:hover {
      background: linear-gradient(to right, #ffcc00, #ff8800);
      transform: scale(1.07);
      color: black;
      box-shadow: 0px 6px 15px rgba(255, 136, 0, 0.5);
    }

    @media (max-width: 768px) {
      .button-container {
        grid-template-columns: repeat(3, 1fr);
      }
    }
  </style>

```



```

        h2 { font-size: 32px; }
    }
</style>
</head>
<body>
    <h2>Select a Room to Design</h2>
    <!-- Replace your form with this -->
    <div class="button-container">
        <button type="button" onclick="handleClick('Bedroom')">Bedroom</button>
        <button type="button" onclick="handleClick('Hall')">Hall</button>
        <button type="button" onclick="handleClick('Living Room')">Living Room</button>
        <button type="button" onclick="handleClick('Dining Room')">Dining Room</button>
        <button type="button" onclick="handleClick('Study Room')">Study Room</button>
        <button type="button" onclick="handleClick('Kids Room')">Kids Room</button>
    </div>

    <script>
        function handleClick(room) {
            window.location.href = "/designer?room_type=" + encodeURIComponent(room);
        }
    </script>

</body>
</html>

```

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Generated Image</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
    <style>
        #imageContainer {
            width: 100%;
            height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
            flex-direction: column;
        }
        #generatedImage {
            max-width: 80%;
            max-height: 80%;
            border: 1px solid #ddd;
            margin-top: 20px;
        }
        #promptInput {
            width: 80%;
            padding: 10px;
            margin-top: 20px;
        }
        #generateButton {
            padding: 10px 20px;
            background-color: #007bff;
            color: white;
            border: none;
            cursor: pointer;
            margin-top: 20px;
        }
    </style>
</head>
<body>
    <div id="imageContainer">
        <input type="text" id="promptInput" placeholder="Enter your prompt" />
        <button id="generateButton" onclick="generateImage()">Generate</button>
        <img id="generatedImage" src="" alt="Generated Image" style="display: none;">
        <a href="{{ url_for('index') }}">Back to Designer</a>
    </div>

```

```

</div>
<script>
  async function generateImage() {
    const promptInput = document.getElementById('promptInput');
    const generatedImage = document.getElementById('generatedImage');
    const prompt = promptInput.value;

    if (!prompt) {
      alert('Please enter a prompt!');
      return;
    }

    const formData = new FormData();
    formData.append('prompt', prompt);

    const response = await fetch('/generate_stable_diffusion_image', {
      method: 'POST',
      body: formData
    });

    if (!response.ok) {
      const error = await response.json();
      alert('Error: ' + error.error);
      return;
    }

    const data = await response.json();
    generatedImage.src = 'data:image/png;base64,' + data.image;
    generatedImage.style.display = 'block';
  }
</script>
</body>
</html>

```

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Room Designer - </title>
  <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}" />
</head>
<body>
  /* Your existing styles */
  body {
    font-family: Arial, sans-serif;
    text-align: center;
    padding: 20px;
  }

  .controls {
    margin: 15px 0;
  }

  canvas {
    border: 1px solid #999;
    margin-top: 20px;
    max-width: 100%;
  }

  #roomDetails,
  #sofaInfo {
    margin-top: 10px;
    font-size: 16px;
  }

  #finalDesignControls {
    margin-top: 20px;
  }

  select, input[type="number"] {
    padding: 4px;
    margin-left: 5px;
  }

  button {
    padding: 6px 12px;
    margin-left: 10px;
    cursor: pointer;
  }

```



```

#suggestions {
  margin-top: 20px;
  padding: 10px;
  border: 1px solid #ddd;
  background-color: #f9f9f9;
  font-family: Arial, sans-serif;
  text-align: left;
}

#suggestions h3 {
  margin-top: 0;
}

#suggestions ul {
  padding-left: 20px;
}

#saveButton {
  display: none;
  margin-top: 10px;
}

#designFinalizedText {
  margin-top: 10px;
  font-weight: bold;
  color: green;
}

#furnailLink {
  margin-top: 20px;
}

#furnailLink button {
  padding: 10px 20px;
  font-size: 16px;
  cursor: pointer;
  background-color: #007bff;
  color: white;
  border: none;
  border-radius: 5px;
}

#furnailLink button:hover {
  background-color: #0056b3;
}

/* New Styles for 2D Layout */
#layoutCanvas {
  border: 1px solid #e5e7eb;
  background-color: #f9fafb;
  cursor: pointer;
  margin-top: 20px;
  max-width: 100%;
  height: auto; /* Maintain aspect ratio */
}

.furniture-button {
  font-family: Arial, sans-serif;
  padding: 6px 12px;
  margin: 5px;
  cursor: pointer;
  background-color: #f0f0f0;
  color: #333;
  border: 1px solid #ccc;
  border-radius: 5px;
  font-size: 12px;
}

.furniture-button:hover {
  background-color: #ddd;
}

.selected-furniture {
  background-color: #4CAF50 !important;
  color: white !important;
  border-color: #4CAF50;
}

</style>
<script src="https://cdnjs.cloudflare.com/ajax/libs/html2canvas/0.4.1/html2canvas.min.js"></script>
</head>
<body>
<h1>🛋️ Interactive Sofa Designer</h1>
<h2>Designing: <strong></strong></h2>

```

```

<div class="controls">
  <label>Choose Room Image: <input type="file" id="uploadRoom" accept="image/*" /></label>
  <label>Choose Sofa Style:
    <select id="sofaStyle">
      <option value="1">Modern</option>
      <option value="2">Minimalist</option>
      <option value="3">Scandinavian</option>
      <option value="4">Traditional</option>
      <option value="5">Mid Century</option>
      <option value="6">Rustic</option>
    </select>
  </label>
  <button onclick="changeSofa()">Change Sofa</button>
</div>

<h2>🎨 Paint the Wall</h2>
<div class="controls" id="paintControls">
  <label for="wallColor">Choose Wall Color:</label>
  <select id="wallColor">
    <option value="initial">Original Color</option>
    <option value="Charcoal">Charcoal</option>
    <option value="Navy Blue">Navy Blue</option>
    <option value="Forest Green">Forest Green</option>
    <option value="Maroon">Maroon</option>
    <option value="Deep Purple">Deep Purple</option>
    <option value="Dark Slate Gray">Dark Slate Gray</option>
    <option value="Indigo">Indigo</option>
    <option value="Dark Olive Green">Dark Olive Green</option>
    <option value="Chocolate">Chocolate</option>
    <option value="Saddle Brown">Saddle Brown</option>
    <option value="Crimson">Crimson</option>
    <option value="Teal">Teal</option>
    <option value="Midnight Blue">Midnight Blue</option>
    <option value="Dark Cyan">Dark Cyan</option>
    <option value="Dark Magenta">Dark Magenta</option>
  </select>
  <button onclick="paintWall()">Paint Wall</button>
</div>

<canvas id="canvas" width="800" height="600"></canvas>

<div id="roomDetails"></div>

<div class="controls">
  <label for="layoutCanvas">Room Layout (Click to add furniture):</label>
  <canvas id="layoutCanvas" width="400" height="300"></canvas>
</div>

<div class="controls">
  <label for="furnitureSelect">Add Basic Furniture:</label>
  <div class="flex space-x-2 flex-wrap" id="furnitureSelect">
    <button data-furniture="bed" class="furniture-button">Bed</button>
    <button data-furniture="table" class="furniture-button">Table</button>
    <button data-furniture="chair" class="furniture-button">Chair</button>
    <button data-furniture="bookshelf" class="furniture-button">Bookshelf</button>
    <button data-furniture="tv_stand" class="furniture-button">TV Stand</button>
  </div>
</div>

<div id="finalDesignControls">
  <button onclick="finalizeDesign()" id="finalizeBtn">Finalize Design</button>
  <div id="designFinalizedText"></div>
  <button id="saveButton" onclick="saveDesign()">Save Design</button>
</div>

<div id="suggestions">
  <h3>Suggested Furniture</h3>
  <ul id="furnitureSuggestions"></ul>
</div>
<input type="hidden" id="roomType" value="">

<div id="furnailink">
  <a href="{{ url_for('text_to_image') }}">
    <button>Want to explore more styles? Try our FurnAI</button>
  </a>
</div>

<script src="{{ url_for('static', filename='script.js') }}"></script>
</body>
</html>

```

5.1.2 Backend Code

```
const canvas = document.getElementById('canvas');
const ctx = canvas.getContext('2d');

const sofaInfo = document.getElementById('sofaInfo');
const roomDetails = document.getElementById('roomDetails');
const finalizeBtn = document.getElementById('finalizeBtn');
const saveButton = document.getElementById('saveButton');
const furnitureSuggestionsDiv = document.getElementById('furnitureSuggestions');
const layoutCanvas = document.getElementById('layoutCanvas');
const layoutCtx = layoutCanvas.getContext('2d');
const upload = document.getElementById('uploadRoom');

let sofaImage = new Image();
sofaImage.src = "/static/sofa1.png";

let roomScale = { x: 1, y: 1 };
let backgroundImage = null;
let originalImageDataURL = null;

let sofaRotationDeg = 0;
let isDraggingHandle = false;
let isDesignFinalized = false;

let room = {
  width: 0,
  length: 0,
  height: 0,
  type: 'hall'
};

let sofa = {
  realWidth: 1.8,
  realHeight: 1.0,
  realX: 1.0,
  realY: 1.0,
  pixelX: 0,
  pixelY: 0,
  pixelWidth: 0,
  pixelHeight: 0,
  dragging: false,
  offsetX: 0,
  offsetY: 0
};

let table = null;
let chairs = [];

// New function to add a table
function addTable() {
  if (!room.width || !room.length) {
    alert("Please set room dimensions first.");
    return;
  }

  const tableWidth = 1.2;
  const tableLength = 0.8;
  const tableX = room.width / 2 - tableWidth / 2;
  const tableY = room.length / 2 - tableLength / 2;

  table = {
    realWidth: tableWidth,
    realLength: tableLength,
    realX: tableX,
    realY: tableY,
    pixelX: tableX * roomScale.x,
    pixelY: tableY * roomScale.y,
    pixelWidth: tableWidth * roomScale.x,
    pixelLength: tableLength * roomScale.y
  };

  draw();
  update2DLayout();
}

// New function to add chairs
function addChairs() {
  if (!room.width || !room.length) {
    alert("Please set room dimensions first.");
    return;
  }

  const chairWidth = 0.5;
  const chairLength = 0.5;
  const chairPositions = [
    { x: room.width / 2 - 1.5 * chairWidth, y: room.length / 2 },
    { x: room.width / 2 + 0.5 * chairWidth, y: room.length / 2 },
    { x: room.width / 2 - 0.5 * chairWidth, y: room.length / 2 - 1.5 * chairLength },
    { x: room.width / 2 + 0.5 * chairWidth, y: room.length / 2 - 1.5 * chairLength }
  ];
}
```

```

        chairs = chairPositions.map(pos => ({
            realX: pos.x,
            realY: pos.y,
            realWidth: chairWidth,
            realLength: chairLength,
            pixelX: pos.x * roomScale.x,
            pixelY: pos.y * roomScale.y,
            pixelWidth: chairWidth * roomScale.x,
            pixelLength: chairLength * roomScale.y
        }));

        draw();
        update2DLayout();
    }

function estimateHeight(type, width, length) {
    const area = width * length;
    if (type === 'hall') return area > 25 ? 3.2 : 3.0;
    if (type === 'dining') return area > 20 ? 2.8 : 2.6;
    if (type === 'bedroom') return area > 16 ? 2.6 : 2.4;
    return 2.5;
}

function applyRoomDimensions() {
    const type = document.getElementById('roomType').value;
    const width = parseFloat(document.getElementById('roomWidth').value);
    const length = parseFloat(document.getElementById('roomLength').value);

    if (!width || !length) {
        alert("Enter valid width and length.");
        return;
    }

    const height = estimateHeight(type, width, length);
    room = { width, length, height, type };

    // Reset table and chairs when room dimensions change
    table = null;
    chairs = [];

    if (type === 'hall') {
        sofa.realWidth = 2.2;
        sofa.realHeight = 1.1;
        sofa.realX = 1.0;
        sofa.realY = 1.0;
    } else if (type === 'dining') {
        sofa.realWidth = 1.8;
        sofa.realHeight = 1.0;
        sofa.realX = 0.5;
        sofa.realY = 1.2;
    } else if (type === 'bedroom') {
        sofa.realWidth = 1.5;
        sofa.realHeight = 0.9;
        sofa.realX = 0.5;
        sofa.realY = 0.5;
    }

    if (backgroundImage) {
        roomScale.x = canvas.width / width;
        roomScale.y = canvas.height / length;
        const adjustedHeight = sofa.realHeight * (room.height / 2.7);
        sofa.pixelX = sofa.realX * roomScale.x;
        sofa.pixelY = sofa.realY * roomScale.y;
        sofa.pixelWidth = sofa.realWidth * roomScale.x;
        sofa.pixelHeight = adjustedHeight * roomScale.y;
    } else {
        canvas.width = 600;
        canvas.height = 400;
        roomScale.x = canvas.width / width;
        roomScale.y = canvas.height / length;
    }

    draw();
    updateSofaInfo();
    updateRoomInfo();
    generateFurnitureSuggestions();
    update2DLayout();
}

upload.addEventListener('change', (e) => {
    const reader = new FileReader();

```

```

function draw() {
  ctx.clearRect(0, 0, canvas.width, canvas.height);

  // Draw background if available
  if (backgroundImage) {
    ctx.drawImage(backgroundImage, 0, 0, canvas.width, canvas.height);
  } else {
    ctx.fillStyle = "#f0f0f0";
    ctx.fillRect(0, 0, canvas.width, canvas.height);
  }

  // Draw sofa
  if (sofa.pixelWidth && sofa.pixelHeight) {
    ctx.save();
    ctx.translate(sofa.pixelX + sofa.pixelWidth / 2, sofa.pixelY + sofa.pixelHeight / 2);
    ctx.rotate((sofa.rotationDeg * Math.PI) / 180);
    ctx.drawImage(sofaImage, -sofa.pixelWidth / 2, -sofa.pixelHeight / 2, sofa.pixelWidth, sofa.pixelHeight);
    ctx.restore();
  }

  // Draw table
  if (table) {
    ctx.fillStyle = 'rgba(150, 100, 50, 0.8)';
    ctx.fillRect(table.pixelX, table.pixelY, table.pixelWidth, table.pixelHeight);
    ctx.fillStyle = 'white';
    ctx.font = '16px Arial';
    ctx.textAlign = 'center';
    ctx.fillText('Table', table.pixelX + table.pixelWidth / 2, table.pixelY + table.pixelHeight / 2);
  }

  // Draw chairs
  chairs.forEach(chair => {
    ctx.fillStyle = 'rgba(100, 100, 100, 0.8)';
    ctx.fillRect(chair.pixelX, chair.pixelY, chair.pixelWidth, chair.pixelHeight);
    ctx.fillStyle = 'white';
    ctx.font = '12px Arial';
    ctx.textAlign = 'center';
    ctx.fillText('Chair', chair.pixelX + chair.pixelWidth / 2, chair.pixelY + chair.pixelHeight / 2);
  });

  drawHandles();
}

width: item.width,
height: item.height,
count: count
});
suggestionsString += <li>${item.name} - ${count}  $\times$  ${item.width}m  $\times$  ${item.height}m</li>;
remainingArea -= count * itemArea;
}

furnitureSuggestionsDiv.innerHTML = suggestionsString || '<li>No other furniture fits in the remaining space.</li>';
}

function finalizeDesign() {
  if (!backgroundImage) {
    alert("Please upload a room image first.");
    return;
  }

  if (!sofa.pixelWidth || !sofa.pixelHeight) {
    alert("Please place the sofa in the room first.");
    return;
  }

  isDesignFinalized = true;
  alert("Design finalized! You can now save the design.");
  showSaveButton();
}

function showSaveButton() {
  saveButton.style.display = 'inline-block';
}

function saveDesign() {
  if (!isDesignFinalized) {
    alert("Please finalize the design before saving.");
    return;
  }

  const dataURL = canvas.toDataURL();
  const a = document.createElement('a');
  a.href = dataURL;

```

```

const roomType = room.type;
let furnitureCatalog = [];

if (roomType === 'hall') {
  furnitureCatalog = [
    { name: 'TV Unit', width: 1.5, height: 0.4 },
    { name: 'Coffee Table', width: 1.0, height: 0.8 },
    { name: 'Bookshelf', width: 0.8, height: 0.3 },
    { name: 'Accent Chair', width: 0.8, height: 0.8 },
    { name: 'Side Table', width: 0.5, height: 0.5 },
    { name: 'Bean Bag', width: 0.7, height: 0.7 }
  ];
} else if (roomType === 'dining') {
  furnitureCatalog = [
    { name: 'Sideboard', width: 1.5, height: 0.5 },
    { name: 'Cabinet', width: 1.2, height: 0.5 },
    { name: 'Bar Cart', width: 0.8, height: 0.4 },
    { name: 'Console Table', width: 1.2, height: 0.4 },
    { name: 'Storage Shelf', width: 1.0, height: 0.3 }
  ];
} else if (roomType === 'bedroom') {
  furnitureCatalog = [
    { name: 'Wardrobe', width: 1.5, height: 0.6 },
    { name: 'Nightstand', width: 0.6, height: 0.4 },
    { name: 'Desk', width: 1.2, height: 0.6 },
    { name: 'Chair', width: 0.5, height: 0.5 },
    { name: 'Bookshelf', width: 0.8, height: 0.3 },
    { name: 'Dresser', width: 1.2, height: 0.5 }
  ];
}

furnitureCatalog.sort((a, b) => (a.width * a.height) - (b.width * b.height));

const suggestions = [];
let suggestionsString = '';

for (let item of furnitureCatalog) {
  const itemArea = item.width * item.height;
  const count = Math.floor(remainingArea / itemArea);
  if (count > 0) {
    suggestions.push({
      name: item.name,
    });
  }
}

canvas.addEventListener('mousemove', (e) => {
  const rect = canvas.getBoundingClientRect();
  const mouseX = e.clientX - rect.left;
  const mouseY = e.clientY - rect.top;
  const size = 10;
  const handleX = sofa.pixelX + sofa.pixelWidth - size / 2;
  const handleY = sofa.pixelY + sofa.pixelHeight - size / 2;

  // Cursor feedback
  if (
    mouseX >= handleX && mouseX <= handleX + size &&
    mouseY >= handleY && mouseY <= handleY + size
  ) {
    canvas.style.cursor = 'nwse-resize';
  } else if (
    mouseX > sofa.pixelX &&
    mouseX < sofa.pixelX + sofa.pixelWidth &&
    mouseY > sofa.pixelY &&
    mouseY < sofa.pixelY + sofa.pixelHeight
  ) {
    canvas.style.cursor = 'move';
  } else {
    canvas.style.cursor = 'default';
  }

  if (sofa.dragging) {
    sofa.pixelX = mouseX - sofa.offsetX;
    sofa.pixelY = mouseY - sofa.offsetY;
    draw();
    updateSofaInfo();
  } else if (isDraggingHandle) {
    sofa.pixelWidth = Math.max(20, mouseX - sofa.pixelX);
    sofa.pixelHeight = Math.max(20, mouseY - sofa.pixelY);
    draw();
    updateSofaInfo();
  }
});

canvas.addEventListener('mouseup', () => {
  sofa.dragging = false;
  isDraggingHandle = false;
});

```

```

document.addEventListener('mouseup', () => {
  clearInterval(rotationInterval);
});

function paintWall() {
  if (!originalImageDataURL) {
    alert("Please upload a room image first.");
    return;
  }
  const selectedColor = document.getElementById('wallColor').value;

  fetch(originalImageDataURL)
    .then(res => res.blob())
    .then(blob => {
      const formData = new FormData();
      formData.append('image', blob, 'original_room_image.png');
      formData.append('color', selectedColor);

      fetch('/paint_wall', {
        method: 'POST',
        body: formData
      })
        .then(response => response.blob())
        .then(imageBlob => {
          const imageUrl = URL.createObjectURL(imageBlob);
          backgroundImage.src = imageUrl;
          backgroundImage.onload = () => {
            canvas.width = backgroundImage.width;
            canvas.height = backgroundImage.height;
            draw();
          };
        })
        .catch(error => {
          console.error('Error painting wall:', error);
          alert('Failed to paint wall.');
```


```

        });
      });
    }

    canvas.addEventListener('mousedown', (e) => {
      const rect = canvas.getBoundingClientRect();

```

```

function updateRoomInfo() {
  roomDetails.textContent =  Room Type: ${room.type}, Width: ${room.width}m, Length: ${room.length}m, Estimated Height: ${room.height}m;
}

```

```

function changeSofa() {
  const sofaId = document.getElementById('sofaStyle').value;
  const roomType = document.getElementById('roomType').value.toLowerCase();
  let imagePath = '';

```

```

  switch (roomType) {
    case 'hall':
      imagePath = /static/sofa${sofaId}.png;
      break;
    case 'bedroom':
      imagePath = /static/b${sofaId}.png;
      break;
    case 'living room':
      imagePath = /static/l${sofaId}.png;
      break;
    case 'study room':
      imagePath = /static/s${sofaId}.png;
      break;
    case 'kids room':
      imagePath = /static/k${sofaId}.png;
      break;
    case 'dining room':
      imagePath = /static/d${sofaId}.png;
      break;
    default:
      console.warn(Unknown room type: ${roomType}. Using default sofa.);
      imagePath = /static/sofa${sofaId}.png;
      break;
  }

```

```

  sofaImage.src = imagePath;
  sofaImage.onload = draw;
}

```

```

function updateStyleOptions() {
  const sofaStyleSelect = document.getElementById('sofaStyle');
  sofaStyleSelect.innerHTML = '';
  const styles = ['Style 1', 'Style 2', 'Style 3', 'Style 4', 'Style 5', 'Style 6'];

```

```

const layoutTableWidth = table.realWidth * scaleX;
const layoutTableLength = table.reallength * scaleY;
const layoutTableX = table.realX * scaleX;
const layoutTableY = table.realY * scaleY;

layoutCtx.fillStyle = 'rgba(150, 100, 50, 0.8)';
layoutCtx.fillRect(layoutTableX, layoutTableY, layoutTableWidth, layoutTableLength);
layoutCtx.fillStyle = 'white';
layoutCtx.font = '10px Arial';
layoutCtx.textAlign = 'center';
layoutCtx.fillText('Table', layoutTableX + layoutTableWidth / 2, layoutTableY + layoutTableLength / 2);
}

// Draw chairs
chairs.forEach(chair => {
const layoutChairWidth = chair.realWidth * scaleX;
const layoutChairLength = chair.reallength * scaleY;
const layoutChairX = chair.realX * scaleX;
const layoutChairY = chair.realY * scaleY;

layoutCtx.fillStyle = 'rgba(100, 100, 100, 0.8)';
layoutCtx.fillRect(layoutChairX, layoutChairY, layoutChairWidth, layoutChairLength);
layoutCtx.fillStyle = 'white';
layoutCtx.font = '10px Arial';
layoutCtx.textAlign = 'center';
layoutCtx.fillText('Chair', layoutChairX + layoutChairWidth / 2, layoutChairY + layoutChairLength / 2);
});
}

// Event Listeners
finalizeBtn.addEventListener('click', finalizeDesign);
saveButton.addEventListener('click', saveDesign);

// Initial calls
applyRoomDimensions();
updateSofaInfo();
updateRoomInfo();
generateFurnitureSuggestions();
update2DLayout();

```

```

import torch
from transformers import SegformerForSemanticSegmentation, SegformerImageProcessor
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
import cv2

# Paths to model and image
model_dir = "./segformer_model"
image_path = "./room1.jpg"

# Load model and processor
processor = SegformerImageProcessor.from_pretrained(model_dir, local_files_only=True)
model = SegformerForSemanticSegmentation.from_pretrained(model_dir, local_files_only=True)

# Load and preprocess the image
image = Image.open(image_path).convert("RGB")
inputs = processor(images=image, return_tensors="pt")

# Perform inference
with torch.no_grad():
    outputs = model(**inputs)

# Extract predictions
logits = outputs.logits
predicted_segmentation = torch.argmax(logits, dim=1).squeeze(0).cpu().numpy()

# Resize segmentation mask to match original image size
original_width, original_height = image.size
resized_segmentation = np.array(Image.fromarray(predicted_segmentation.astype(np.uint8)).resize(
    (original_width, original_height), resample=Image.NEAREST))

# Identify the wall class (assumed class 0 for 'wall')
wall_class_id = 0
wall_mask = resized_segmentation == wall_class_id

# Convert the image to a NumPy array
image_array = np.array(image)

# Dark wall color options
color_options = {
    "Charcoal": (54, 69, 79),
    "Navy Blue": (0, 0, 128),
    "Forest Green": (34, 139, 34),

```



```

"Maroon": (128, 0, 0),
"Deep Purple": (48, 25, 52),
"Dark Slate Gray": (47, 79, 79),
"Indigo": (75, 0, 130),
"Dark Olive Green": (85, 107, 47),
"Chocolate": (123, 63, 0),
"Saddle Brown": (139, 69, 19),
"Crimson": (220, 20, 60),
"Teal": (0, 128, 128),
"Midnight Blue": (25, 25, 112),
"Dark Cyan": (0, 139, 139),
"Dark Magenta": (139, 0, 139),
}

# Show available color options
print("Available Dark Wall Colors:")
for i, color_name in enumerate(color_options):
    print(f"{i+1}. {color_name}")

# Get user's color choice
while True:
    try:
        choice = int(input("Enter the number of the color you want to use (or 0 to exit): "))
        if choice == 0:
            exit()
        if 1 <= choice <= len(color_options):
            selected_color = list(color_options.values())[choice - 1]
            break
        else:
            print("Invalid choice. Please enter a valid number.")
    except ValueError:
        print("Invalid input. Please enter a number.")

# Use the selected dark color directly (no brightening)
new_color = np.array(selected_color)

# Blending factor
alpha = 0.4

# Ensure wall_mask shape matches image
wall_mask = cv2.resize(wall_mask.astype(np.uint8), (image_array.shape[1], image_array.shape[0]), interpolation=cv2.INTER_NEAREST) == 1

# Blend selected color with wall pixels
for c in range(3):
    image_array[wall_mask, c] = (1 - alpha) * image_array[wall_mask, c] + alpha * new_color[c]

# Apply a subtle blur to simulate paint finish
kernel_size = (5, 5)
blurred_wall = cv2.GaussianBlur(image_array, kernel_size, 0)
image_array[wall_mask] = blurred_wall[wall_mask]

# Convert back to PIL image
output_image = Image.fromarray(image_array)

# Display result
plt.figure(figsize=(10, 10))
plt.imshow(output_image)
plt.title(f"Wall Color Changed to {list(color_options.keys())[choice-1]} (Dark Tone)")
plt.axis("off")
plt.show()

# Save output
output_image.save(f"output_with_{list(color_options.keys())[choice-1].lower().replace(' ', '_')}_wall.jpg")

```

```

from flask import Flask, render_template, request, redirect, url_for, session, send_file, jsonify, flash
from PIL import Image
import numpy as np
import io
import torch
from transformers import SegformerForSemanticSegmentation, SegformerImageProcessor
import cv2
from werkzeug.security import generate_password_hash, check_password_hash
from flask_sqlalchemy import SQLAlchemy
import torch
from diffusers import StableDiffusionPipeline

auth_token = "hf_..." #put your auth token here

app = Flask(__name__)
app.secret_key = 'your_secret_key' # Required for session and flash

# Load Stable Diffusion model (outside the route to load it only once)
device = "cuda" if torch.cuda.is_available() else "cpu"
dtype = torch.float16 if device == "cuda" else torch.float32
pipe = None
try:
    pipe = StableDiffusionPipeline.from_pretrained(
        "runwayml/stable-diffusion-v1-5",
        revision="fp16" if device == "cuda" else None,
        torch_dtype=dtype,
        use_auth_token=auth_token,
    )
    pipe.to(device)
    pipe.safety_checker = lambda images, **kwargs: (images, [False] * len(images))
except Exception as e:
    print(f"Error loading the Stable Diffusion Pipeline: {e}")

@app.route('/')
def index():
    return redirect(url_for('login'))

@app.route('/text_to_image')
def text_to_image():
    return render_template('text_to_image.html')

@app.route('/generate_stable_diffusion_image', methods=['POST'])
def generate_stable_diffusion_image():
    if pipe is None:
        prompt_text = request.form.get('prompt')
        if not prompt_text:
            return jsonify({'error': 'No text prompt provided'}), 400

    try:
        if device == "cuda":
            with torch.autocast("cuda"):
                result = pipe(prompt_text, guidance_scale=7.5, num_inference_steps=25)
        else:
            with torch.no_grad():
                result = pipe(prompt_text, guidance_scale=7.5, num_inference_steps=25)
        image = result.images[0].resize((512, 512))

        # Convert the PIL Image to a base64 string
        import io
        import base64
        buffered = io.BytesIO()
        image.save(buffered, format="PNG")
        img_str = base64.b64encode(buffered.getvalue()).decode()

        return jsonify({'image': img_str})

    except Exception as e:
        return jsonify({'error': f'Error generating image: {e}'}), 500

# Database config
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///users.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)

# Wall colors
color_options = {
    "Charcoal": (54, 69, 79), "Navy Blue": (0, 0, 128), "Forest Green": (34, 139, 34),
    "Maroon": (128, 0, 0), "Deep Purple": (48, 25, 52), "Dark Slate Gray": (47, 79, 79),
    "Indigo": (75, 0, 130), "Dark Olive Green": (85, 107, 47), "Chocolate": (123, 63, 0),
    "Saddle Brown": (139, 69, 19), "Crimson": (220, 20, 60), "Teal": (0, 128, 128),
    "Midnight Blue": (25, 25, 112), "Dark Cyan": (0, 139, 139), "Dark Magenta": (139, 0, 139),
}

```

```

model_dir = "../segformer_model"
try:
    processor = SegFormerImageProcessor.from_pretrained(model_dir, local_files_only=True)
    model = SegFormerForSemanticSegmentation.from_pretrained(model_dir, local_files_only=True)
except Exception as e:
    print(f"Error loading Segformer model: {e}")
    processor = None
    model = None

# User model
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(120), unique=True, nullable=False)
    password = db.Column(db.String(200), nullable=False)

    def __repr__(self):
        return f"<User {self.email}>"

# Initialize DB
with app.app_context():
    db.create_all()

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        user = User.query.filter_by(email=email).first()
        if user and check_password_hash(user.password, password):
            session['username'] = email
            return redirect(url_for('selectroom'))
        else:
            flash('Invalid email or password.', 'danger')
    return render_template('login.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']
        confirm = request.form['confirm_password']

        if password != confirm:
            if User.query.filter_by(email=email).first():
                flash("Email is already registered. Please use a different one.", "danger")
                return render_template("register.html")

            hashed_password = generate_password_hash(password)
            new_user = User(email=email, password=hashed_password)
            try:
                db.session.add(new_user)
                db.session.commit()
                flash("Registration successful! Please log in.", "success")
                return redirect(url_for("login"))
            except Exception as e:
                db.session.rollback()
                flash(f"Error: {str(e)}", "danger")
    return render_template('register.html')

@app.route('/selectroom', methods=['GET', 'POST'])
def selectroom():
    if request.method == 'POST':
        room_type = request.form['room_type']
        return redirect(url_for('designer', room_type=room_type))
    return render_template('selectroom.html')

@app.route('/designer')
def designer():
    room_type = request.args.get('room_type', 'Hall')
    return render_template('index.html', room_type=room_type)

@app.route('/setroom/<room>')
def set_room(room):
    session['room_type'] = room
    return redirect(url_for('designer'))

@app.route('/paint_wall', methods=['POST'])
def paint_wall():
    if 'image' not in request.files:
        return jsonify({'error': 'No image uploaded'}), 400
    image_file = request.files['image']
    color = request.form.get('color')
    if not color:
        return jsonify({'error': 'No color selected'}), 400
    painted_image = paint_wall_on_image(image_file, color)

```



```

if painted_image:
    return send_file(io.BytesIO(painted_image), mimetype='image/png')
    return jsonify({'error': 'Failed to paint wall'}), 500

def paint_wall_on_image(image_file, color_name):
    try:
        image = Image.open(image_file).convert("RGB")
        if processor is None or model is None:
            raise Exception("Segformer model or processor not loaded.")
        inputs = processor(images=image, return_tensors="pt")
        with torch.no_grad():
            outputs = model(**inputs)
        logits = outputs.logits
        predicted_segmentation = torch.argmax(logits, dim=1).squeeze(0).cpu().numpy()
        original_width, original_height = image.size
        resized_segmentation = np.array(Image.fromarray(predicted_segmentation.astype(np.uint8)).resize(
            (original_width, original_height), resample=Image.NEAREST))
        wall_class_id = 0
        wall_mask = resized_segmentation == wall_class_id
        image_array = np.array(image)
        wall_mask = cv2.resize(wall_mask.astype(np.uint8), (image_array.shape[1], image_array.shape[0]), interpolation=cv2.INTER_NEAREST) == 1
        if color_name in color_options:
            new_color = np.array(color_options[color_name])
            alpha = 0.4
            for c in range(3):
                image_array[wall_mask, c] = (1 - alpha) * image_array[wall_mask, c] + alpha * new_color[c]
            blurred = cv2.GaussianBlur(image_array, (5, 5), 0)
            image_array[wall_mask] = blurred[wall_mask]
            output = Image.fromarray(image_array)
            img_byte_arr = io.BytesIO()
            output.save(img_byte_arr, format='PNG')
            return img_byte_arr.getvalue()
    except Exception as e:
        print("Error painting wall:", e)
        return None

```

```

import tkinter as tk
import customtkinter as ctk

from PIL import Image
from customtkinter import CTkImage
from auth_token import auth_token

import torch
from diffusers import StableDiffusionPipeline

# Create the app window
app = tk.Tk()
app.geometry("532x632")
app.title("Stable Bud")
ctk.set_appearance_mode("dark")

# Prompt input box
prompt = ctk.CTkEntry(master=app, height=40, width=512, text_color="black", fg_color="white")
prompt.place(x=10, y=10)

# Label to show generated image
lmain = ctk.CTkLabel(master=app, height=512, width=512, text="")
lmain.place(x=10, y=110)

# Load model
device = "cuda" if torch.cuda.is_available() else "cpu"
dtype = torch.float16 if device == "cuda" else torch.float32

pipe = StableDiffusionPipeline.from_pretrained(
    "6",
    revision="fp16" if device == "cuda" else None,
    torch_dtype=dtype,
    use_auth_token=auth_token,
)
pipe.to(device)

# Optional: Disable safety checker to speed things up (for personal use only)
pipe.safety_checker = lambda images, **kwargs: (images, [False] * len(images))

# Generate image from prompt
def generate():
    prompt_text = prompt.get()

    if device == "cuda":

```

```

        with torch.autocast("cuda"):
            result = pipe(prompt_text, guidance_scale=7.5, num_inference_steps=25)
    else:
        with torch.no_grad():
            result = pipe(prompt_text, guidance_scale=7.5, num_inference_steps=25)

    image = result.images[0].resize((512, 512))
    ctk_image = CTKImage(light_image=image, dark_image=image, size=(512, 512))

    lmain.configure(image=ctk_image)
    lmain.image = ctk_image

# Button to trigger generation
trigger = ctk.CTkButton(
    master=app,
    height=40, width=120, text_color="white", fg_color="blue", command=generate,
    text="Generate"
)
trigger.place(x=206, y=60)

# Start the app
app.mainloop()

```

```

body {
    font-family: sans-serif;
    background: #f5f5f5;
    text-align: center;
}

.controls {
    margin: 10px 0;
}

canvas {
    border: 2px solid #444;
    margin-top: 20px;
    cursor: move;
}

#sofaInfo, #roomDetails {
    margin-top: 10px;
    font-weight: bold;
}

/* --- New Styles for Paint Wall Section --- */
#paintControls {
    margin-top: 20px;
    padding: 15px;
    border: 1px solid #ccc;
    background-color: #f9f9f9;
    border-radius: 5px;
}

#sofaImage {
    transform-style: preserve-3d;
    backface-visibility: hidden;
}

#paintControls label {
    margin-right: 10px;
}

#paintControls select {
    padding: 8px;
    border: 1px solid #aaa;
    border-radius: 3px;
}

```

```

#paintControls button {
  padding: 8px 15px;
  background-color: #5cb85c;
  color: white;
  border: none;
  border-radius: 3px;
  cursor: pointer;
  font-size: 1em;
  margin-left: 10px;
}

#paintControls button:hover {
  background-color: #4cae4c;
}

/* --- New Styles for Finalize Design Section --- */
#finalDesignControls {
  margin-top: 20px;
  padding: 15px;
  border: 1px solid #ccc;
  background-color: #f9f9f9;
  border-radius: 5px;
}

#finalDesignControls button {
  padding: 10px 20px;
  background-color: #007bff;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  font-size: 1.1em;
}

#finalDesignControls button:hover {
  background-color: #0056b3;
}

#finalDesignOutput {
  margin-top: 10px;
  font-weight: bold;
}

```

5.2 Experimental Results

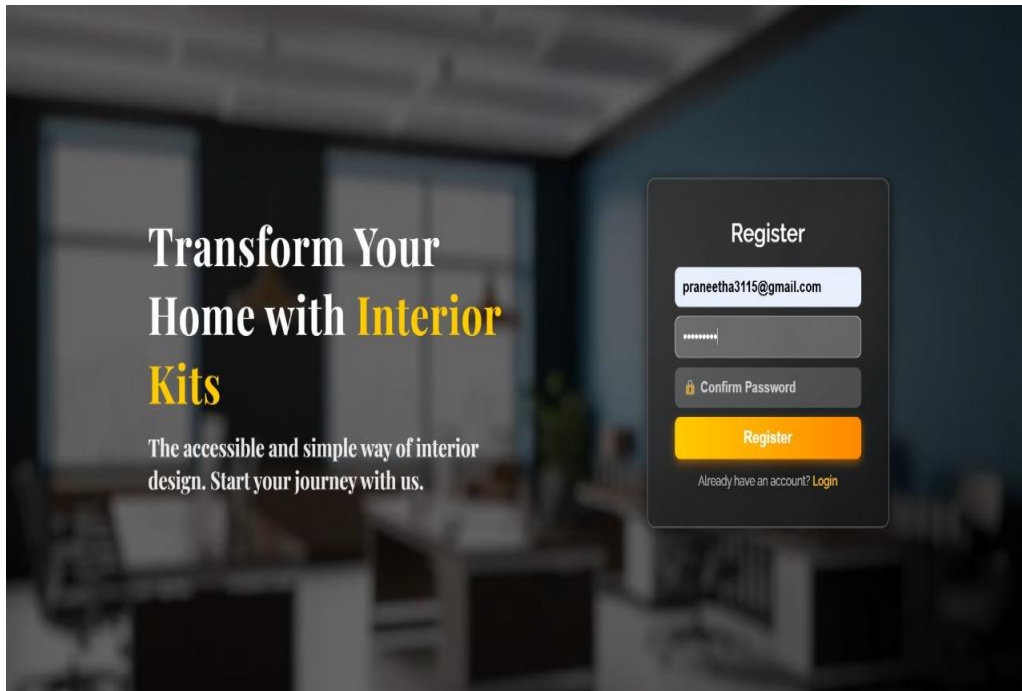


Fig 5.2.1 Register Page

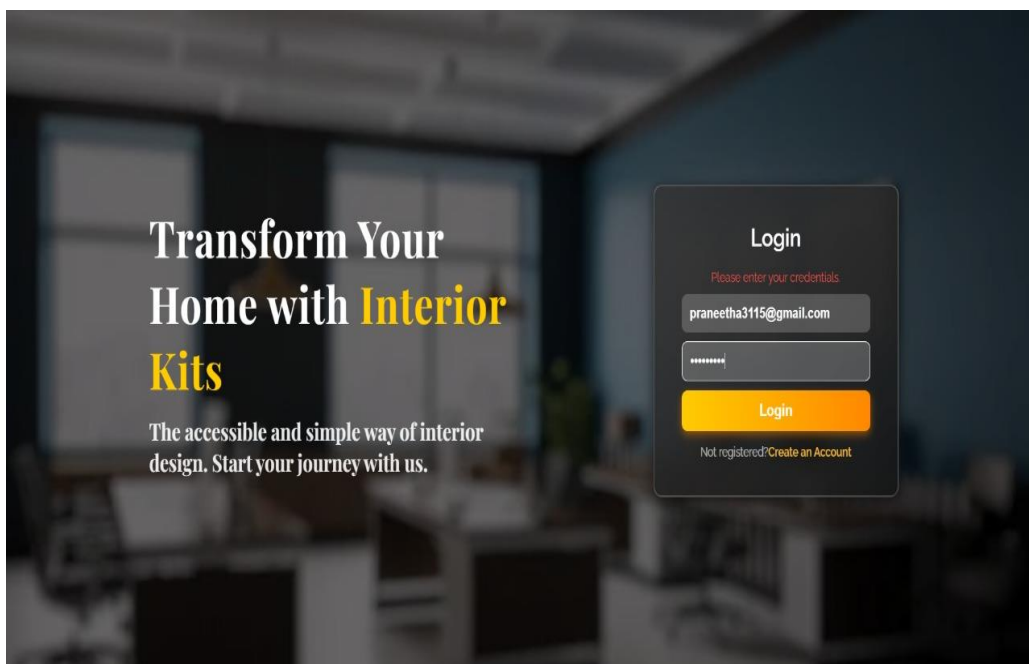


Fig 5.2.2 Login Page

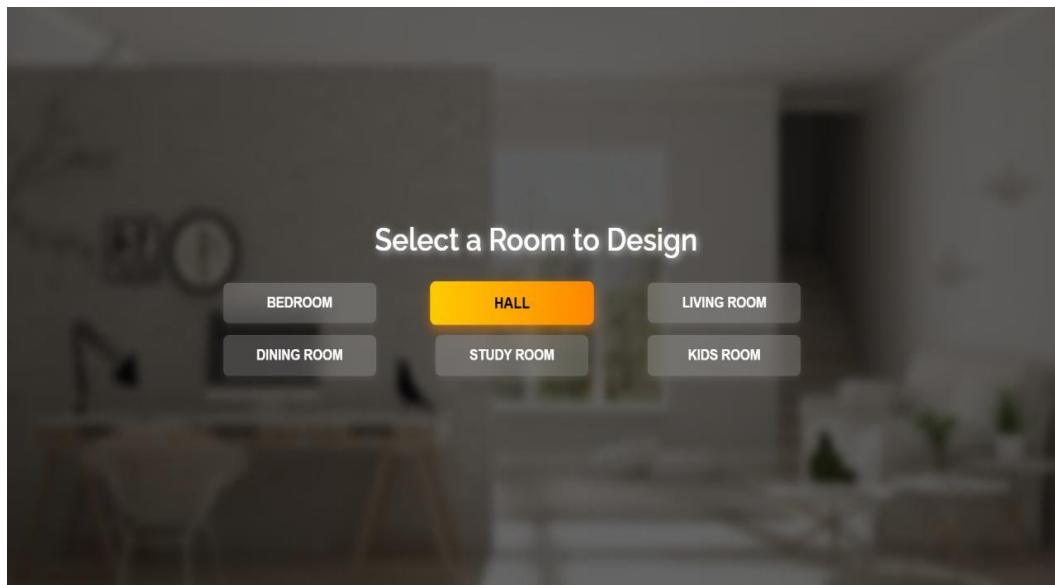


Fig 5.2.3 Select a room Page

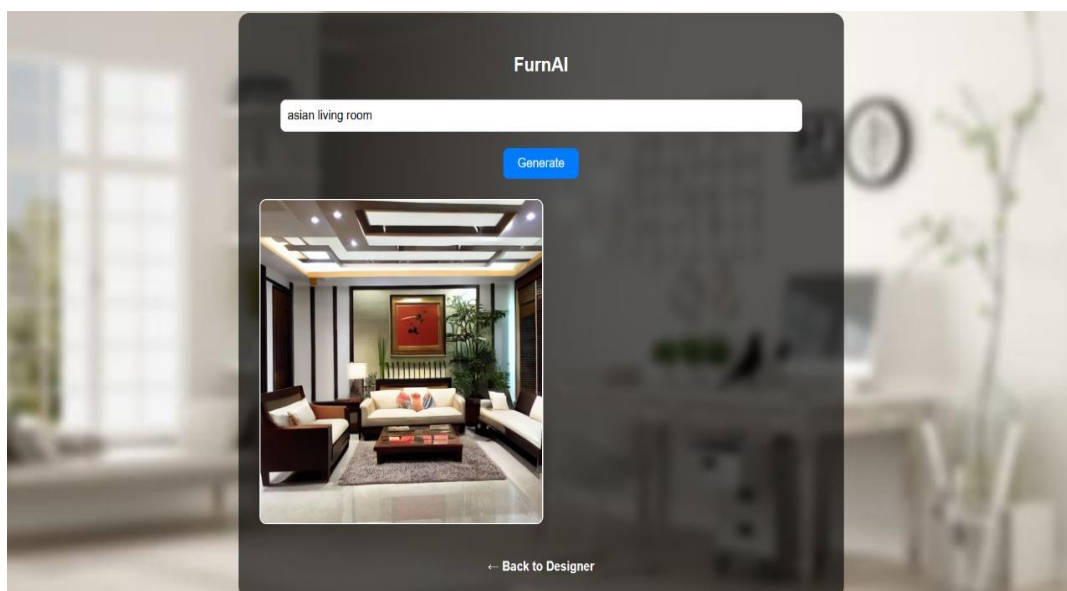


Fig 5.2.4 FurnAI image Page

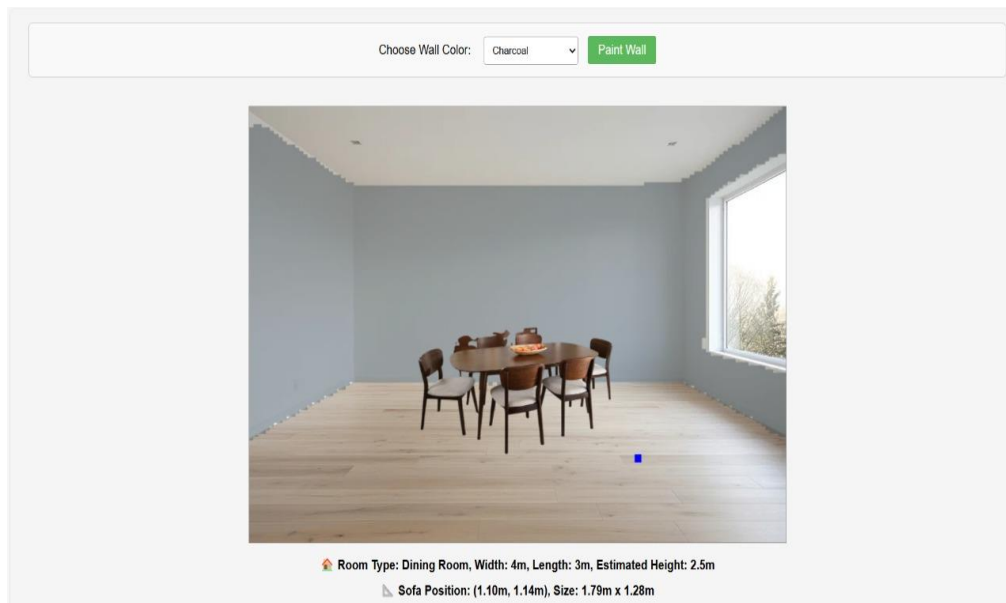


Fig 5.2.5 Dining room image Page

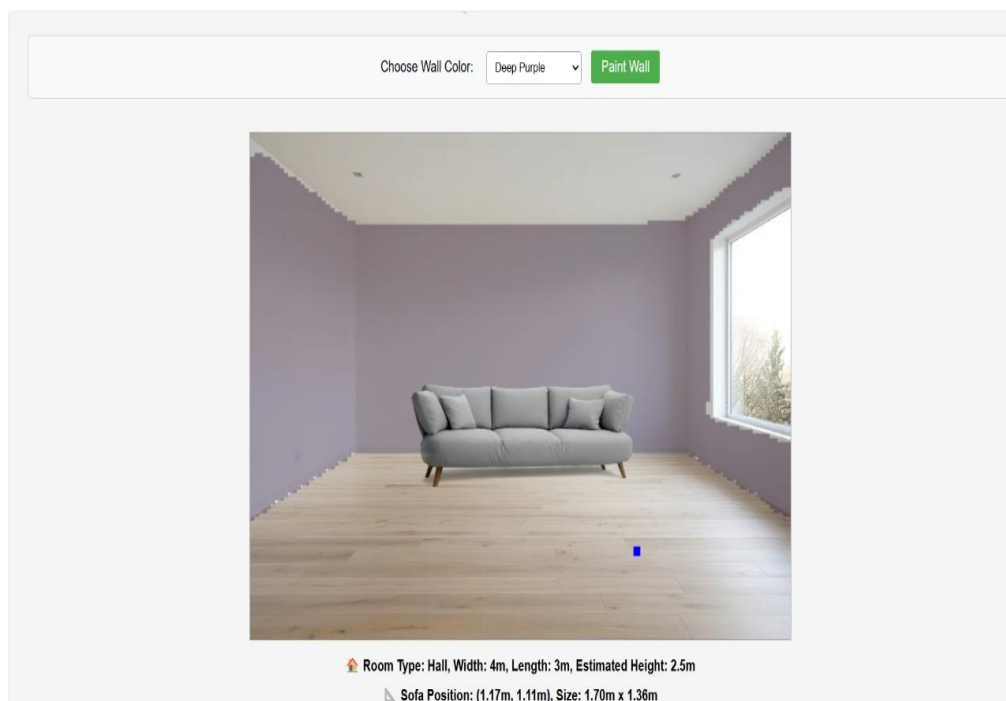


Fig 5.2.6 Hall image Page

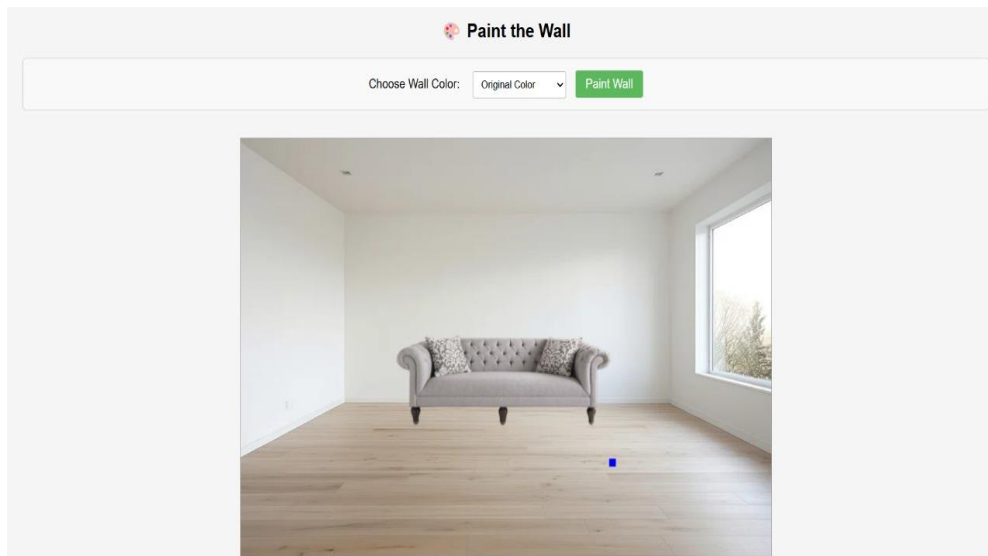


Fig 5.2.7 Living room image Page

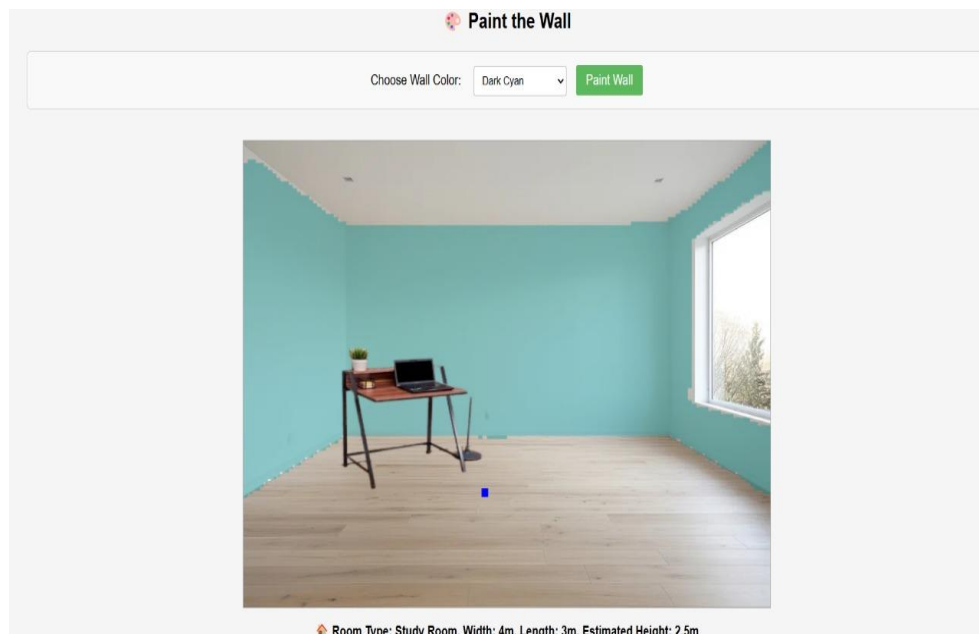


Fig 5.2.8 Study room image Page

5.3.1 Test Cases

Table 5.3: Test Cases

S.no	Test Scenario	Test Case	Expected Results	Status Pass/Fail
1	Login Authentication	Verify that user credentials are stored securely with encryption.	User credentials are stored securely with password hashing.	Pass
2	Password Validation	Ensure password is not less than 6 characters during registration.	If password is < 6 characters, system shows an error and rejects registration.	Pass
3	Room Dimension Validation	Enter invalid (alphabetical) characters in room dimensions field.	System displays an error message allowing only numeric values	Pass
4	Design Generation	Provide incomplete room information (e.g., missing dimensions).	System should not proceed and should ask for complete information.	Fail
5	Image Upload Validation	Upload a .txt file instead of an image.	System should reject the file and only accept JPG/PNG	Pass
6	Image Upload Size Limit	Try uploading an image larger than allowed size (e.g., >5MB).	System should reject large file uploads and display size limit error.	Pass
7	Forgot Password Recovery	Test the forgot password functionality with a registered email.	Password reset link should be sent to the registered email address.	Fail

8	Style Preference Reset	After selecting preferences, click reset/clear button.	All selected preferences should be cleared and reset to default.	Pass
9	Large Room Dimensions	Enter abnormally large dimensions (e.g., 99999 x 99999).	System should restrict unrealistic inputs and show validation error.	Pass
10	Empty Field Validation	Submit form without entering any data (empty fields).	System should display validation errors for required fields.	Pass

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

The Theme-Based Interior Designer system makes use of a mix of superior deep gaining knowledge of fashions and traditional internet technology to offer a smart and beneficial solution to modern indoors layout problems. With a frontend constructed with HTML, CSS, JavaScript, and Canvas API, and a backend pushed with the aid of using Flask, PyTorch, and modern-day imaginative and prescient fashions like Segformer and DPT, the gadget ensures a easy and responsive consumer experience. Interactivity is similarly progressed with the aid of using technology like html2canvas, which permit customers preview and shop their created designs directly from the browser interface.

In assessment to conventional structures that most effective deal with 2D visible upgrades, this challenge makes use of monocular intensity estimation (DPT) and semantic segmentation (Segformer) to deal with spatial comprehension, permitting sensible and accurately scaled furnishings placement inside a given room. The webweb page creates visually attractive and spatially correct AI-sponsored layout layouts with the aid of using permitting customers to post room photos, input dimensions, and pick out their favored layout. Homeowners, designers, and actual property dealers can also additionally now extra without difficulty gain individualized, great indoors layout way to the combination of those fashions with an intuitive interface.

6.2 Future Scope

Augmented reality (AR) integration: Enhance spatial visualization via allowing customers to find out AI-generated interior designs in their actual surroundings via smartphones or AR glasses.

Real-Time User Feedback Loop: Give clients the selection to remark, rate, or regulate the designs which may be produced. This allows the model to be retrained or improved, developing the precision and applicability of subsequent outputs.

Personalized Budget Estimator: Make the tool viable for an entire lot of financial plans via inclusive of a dynamic fee variety estimation tool that determines the overall rate of decided on products and offers rate-effective substitutes.

AI Chatbot Assistant: Create an interactive chatbot that uses natural language processing (NLP) to assist clients with the format approach via answering questions, making recommendations for enhancements, and helping with step-via-step customization.

E-commerce Platform Integration: Make it possible for customers to buy the recommended products immediately from the platform via allowing direct integration with on-line fixtures and decor retailers.

Voice Command Integration: Give clients the cappotential to talk with the tool via inquiring for format changes, navigating the interface, and getting into alternatives the usage of natural language voice commands.

Multi-Language Support: To increase the platform`s global reap and make it usable via clients with diverse linguistic backgrounds, include language localization alternatives.

CHAPTER 7

REFERENCES

- [1] "Home Interior Design Suggestion System using Deep Learning" - H.D. Shonali Romeshika, Dilshan Dasanayaka, D.C. Deeghayu Alwis
https://www.researchgate.net/publication/385535859_Home_Interior_Design_Suggestion_System_using_Deep_Learning
- [2] "Interior Design Evaluation Based on Deep Learning: A Multi-Modal Fusion Evaluation Mechanism" - Yiyan Fan, Yang Zhou, Zheng Yuan
<https://www.mdpi.com/2227-7390/12/10/1560>
- [3] "Dynamic Style Transfer for Interior Design: An IoT-Driven Approach" -Yangyang Li, Li Sun
<https://www.sciencedirect.com/science/article/pii/S1110016824016168>
- [4] "RoomDiffusion: A Specialized Diffusion Model in the Interior Design Industry" -Zhaowei Wang, Ying Hao, Hao Wei
<https://arxiv.org/abs/2409.03198>
- [5] "Interior Design Using Augmented Reality" – Chetan Patil
https://www.researchgate.net/publication/325822199_Interior_Design_Using_Augmented_Reality
- [6] "DiffDesign: Controllable Diffusion with Meta Prior for Efficient Interior Design Generation" – Yuxuan Yang, Jingyao Wang, Tao Geng, Wenwen Qiang, Changwen Zheng, Fuchun Sun
<https://arxiv.org/abs/2411.16301>
- [7] "Application Research of Interior Design Style Migration from the Perspective of Artificial Intelligence" – Yangyang Li, Li Sun
<https://drpress.org/ojs/index.php/ajst/article/view/12272>
- [8] "Augmented Reality Based Interior Designing System" – Aswin Tharayil Santhosh, Anagha S, Willson C Joseph, Godwin Baiju, Aston Raju
<https://www.ijert.org/augmented-reality-based-interior-designing-system>

[9] "AI-Based Interior Designing App" – Ch. Adarsh, A. Ravalika, A. Sumanth, Dr. Madhavi Pingili

<https://ieeexplore.ieee.org/document/10262485>

[10] "Personalized Recommendation Algorithm of Interior Design Style Based on Local Social Network" – Guohui Fan, Chen Guo

<https://xml.jips-k.org/full-text/view?doi=10.3745/JIPS.01.0096>

[11] "Gen AI and Interior Design Representation: Applying Design Styles Using Fine-Tuned Models" – Hyun Jeong, Youngchae Kim, Seunghyun Cha

<https://books.fupress.com/chapter/gen-ai-and-interior-design-representation-applying-design-styles-using-fine-tuned-models/14482>

[12] "Minimalism in Interior Design" – K. Rithish

https://www.academia.edu/72103851/MINIMALISM_IN_INTERIOR_DESIGN