SMART SPACE: A MERN-BASED COLLABORATIVE AI INTERIOR DESIGN TOOL

A PROJECT REPORT

Submitted by

DUVVURU GANESH	**[410721243023]**
GOPIREDDY HARSHA TEJA	**[410721243029]**
KATARI HARSHA VARDHAN	**[410721243044]**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

DHANALAKSHMI COLLEGE OF ENGINEERING

ANNA UNIVERSITY:: CHENNAI 600 025

MAY 2025

ANNA UNIVERSITY: CHENNAI

600 025

BONAFIDE CERTIFICATE

Certified that this project report **"SMART SPACE: A MERN-BASED COLLABORATIVE AI INTERIOR DESIGN TOOL"** is the bonafide work of **"DUVVURU GANESH (410721243023), GOPIREDDY HARSHA TEJA (410721243029), KATARI HARSHA VARDHAN (410721243044)"** who carried out the project work under my supervision.

SIGNATURE	SIGNATURE
Dr.K.DHINAKARAN, M.Tech.,Ph.D	Mr.SANJAI KUMAR, B.E., M.TECH
HEAD OF THE DEPARTMENT	SUPERVISOR
Artificial Intelligence and Data Science	Artificial Intelligence and Data Science
Dhanalakshmi College of Engineering,	Dhanalakshmi College of Engineering,
Dr VPR Nagar, Manimangalam	Dr VPR Nagar, Manimangalam
Chennai - 601301	Chennai - 601301

CERTIFICATE OF EVALUATION

College Name : 4107 – Dhanalakshmi College of Engineering
Branch & Semester: Artificial Intelligence and Data Science / 8th Semester
Subject : AD3811- Project Work

S.NO	**NAME OF THE STUDENT**	**TITLE OF THE PROJECT**	**NAME OF THE SUPERVISOR WITH THE DESIGNATION**
1.	DUVVURU GANESH (410721243060)	SMART SPACE: A MERN-BASED COLLABORATIVE AI INTERIOR DESIGN TOOL	Dr. K. SANJAI KUMAR
2.	GOPIREDDY HARSHA TEJA (410721243078)		
3.	KATARI HARSHA VARDHAN (410721243044)		

The report of the project work submitted by the above students in partial fulfillment for the award of a Bachelor of Technology degree in **ARTIFICIAL INTELLIGENCE AND DATA SCIENCE** under Anna University was evaluated and confirmed to be the reports about the work done by the above students.

Submission for the final year project viva-voce examination held at

Dhanalakshmi College of Engineering on ____________________

INTERNAL EXAMINER EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable Founder and Chairman Dr.V.P.Ramamurthi, M.Sc.(Engg.), Ph.D.,Dhanalakshmi College of Engineering, for guiding us and permitting us to do our project on our own.

We express our sincere gratitude to our beloved Principal, Dr. PRADEEP
KUMAR A R , B.E., M.E., Ph.D.,for his support and guidance.

We extend our gratitude and heartfelt thanks to the Head of the Department, DR. DHINAKARAN K , M.Tech., Ph.D.,for guiding us in all aspects of our project at each stage and providing us with valuable suggestions.

We would like to thank at most to our Project Supervisor, Mr. SANJAI KUMAR, B.E., M.TECH ,who rendered valuable guidance and full support always.

We take this opportunity to thank all the **faculty members of the Department of Artificial Intelligence and Data Science** for their unwavering support and cooperation which made us keep our zeal and spirits high to complete this project work successfully.

ABSTRACT

This paper introduces a comprehensive web-based application built using the MERN stack (MongoDB, Express.js, React.js, Node.js) designed to facilitate collaborative, AI-driven interior design. The platform leverages modern web technologies and artificial intelligence to streamline the design process, allowing users to generate personalized interior design suggestions based on their unique preferences and requirements. The core innovation lies in the integration of user-centric customization with real-time AI-generated design recommendations. Users can initiate a design session by uploading an image of their room and providing key input parameters, including the desired design theme (e.g., modern, vintage, minimalist), room type (e.g., bedroom, kitchen), room dimensions, budget constraints, and preferred colour palettes. These inputs are processed and sent to the Replicate API, which utilizes state-of-the-art machine learning models to generate photorealistic, stylistically aligned interior designs. To manage and deliver a seamless user experience, the system architecture is divided across the MERN stack. The frontend, built with React.js, offers an intuitive and interactive user interface, supporting real-time previews, input customization, and ratings. Express.js and Node.js form the backend layer, handling business logic, user sessions, and API communications. Cloudinary is used for secure and scalable image storage, ensuring fast access and minimal latency. MongoDB serves as the primary database, storing all relevant metadata including design parameters, image links, and user-generated ratings. The application incorporates a dual-optimization framework: it aims to minimize project costs (as specified by user budgets) and maximize aesthetic satisfaction, which is quantitatively captured through a star-rating system.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1. INTRODUCTION

Interior design plays a crucial role in shaping functional and aesthetically appealing living and working spaces. Traditionally, the interior design process is labour- intensive, time-consuming, and highly dependent on manual planning and designer-client interactions. These traditional methods often suffer from inefficiencies such as prolonged decision-making cycles, limited scalability, and a lack of real-time collaboration.

In recent years, the convergence of web technologies and artificial intelligence (AI) has paved the way for innovation in the interior design domain. This paper introduces an intelligent, web-based interior design collaboration platform powered by the MERN stack—comprising MongoDB, Express.js, React.js, and Node.js. The application facilitates collaborative, AI-driven design generation by allowing users to upload images of their spaces, specify a variety of design preferences (e.g., room type, theme, dimensions, budget, colour palette), and receive photorealistic interior layouts created using the Replicate API.

From a computational perspective, this project conceptualizes the Interior Design Collaboration Problem (IDCP) as a bi-objective optimization problem (BO-IDCP). The two key objectives are:

1. Minimizing project costs, considering constraints such as user-defined budgets and resource usage.
2. Maximizing aesthetic functionality, quantified using a user rating system (1–5 stars) to reflect subjective satisfaction.

The system architecture is designed for responsiveness, scalability, and modularity. React.js provides a dynamic and user-friendly frontend, allowing real-time input, design previews, and interaction. The backend, powered by Node.js and Express.js, manages business logic, user sessions, and external API communications. Cloudinary is employed for efficient image storage and delivery, and MongoDB is used to store structured data such as user inputs, AI-generated design metadata, and user ratings.

Jest has been utilized for unit testing across both backend APIs and frontend components. For instance, it ensures the correctness of logic in components like Home.jsx (for image uploads) and Star Rating.jsx (for rating submissions), which are central to the application's core functionality. This testing framework helps maintain the integrity and reliability of the codebase during development .Drawing inspiration from optimization problems in operations research, such as the Vehicle Routing Problem with Time Windows (VRPTW), the BO-IDCP employs principles of multi-objective optimization and evolutionary computation to effectively balance trade-offs between cost-efficiency and user satisfaction. Recent studies emphasize the transformative impact of AI and augmented reality (AR) in interior design. Our system aligns with these trends and lays a foundation for future integration of immersive technologies such as AR for real-time spatial visualization and interaction.

1.1 OBJECTIVE

The primary objective of this project is to design and implement a modern, scalable, and user-friendly web application using the MERN (MongoDB, Express.js, React.js, Node.js) stack, aimed at enabling AI-powered interior design collaboration. This system is envisioned as a next-generation design tool that transforms conventional, manually intensive interior design workflows into an interactive, real-time, and data-driven experience. The application is intended to serve a diverse audience, from individual homeowners and hobbyists to professional interior designers and architectural firms, by providing a platform that combines intuitive user input with advanced AI capabilities.

At the heart of the application lies a strong emphasis on user-centric design input, allowing users to initiate the design process by uploading images of their actual rooms. In addition to image uploads, users are prompted to define a range of preferences, including the desired interior design theme (such as minimalist, bohemian, industrial, or traditional), room type (e.g., bedroom, kitchen, living room), physical dimensions of the space, budget constraints, and preferred color palette and lighting conditions. These parameters form the basis for generating customized interior design suggestions.To generate these suggestions, the platform integrates the Replicate API, which utilizes machine learning models to produce photorealistic, intelligent design layouts tailored to the user's input. The system ensures that the generated outputs are not only visually appealing but also contextually relevant, based on the user's specified functional and stylistic needs. This real-time generation process enhances the user experience by reducing wait times and allowing for rapid iteration and experimentation.

A distinctive feature of the application is its use of a bi-objective optimization framework, referred to as the Bi-Objective Interior Design Collaboration Problem (BO-IDCP). The two optimization objectives are: (1) minimizing project costs, ensuring the AI-generated designs remain financially viable within the user-defined budget, and (2) maximizing aesthetic functionality, which is assessed using a user-based star-rating system ranging from 1 to 5 stars. These ratings are stored and analyzed to refine future design outputs, creating a feedback-driven loop that supports continuous system improvement and personalization.

The platform also includes a persistent design history module, which allows users to access, revisit, and compare previously generated design solutions. This feature supports reflective decision-making and enables users to track their preferences and changes over time. It also provides a valuable archive of design iterations, which can be revisited or re-evaluated as user needs evolve.

Technically, the application is built on a robust and modular architecture. MongoDB serves as the primary database, efficiently storing structured data such as user profiles, design parameters, generated design metadata, and user ratings. Cloudinary is employed for secure and scalable image storage, ensuring high availability and performance when handling image uploads and retrievals. The frontend, developed with React.js, delivers a seamless and responsive user interface, while the backend, powered by Node.js and Express.js, handles business logic, API routing, user authentication, and communication with third-party services.

To maintain software reliability and ensure high-quality performance across the application, unit testing is integrated using the Jest testing framework. Jest is used to test backend APIs and frontend components—such as Home.jsx for handling image uploads and StarRating.jsx for rating functionality—ensuring each module behaves as expected and edge cases are effectively managed. Ultimately, this project aims to democratize access to high-quality interior design, making it more affordable, accessible, and engaging for a broader user base. By combining artificial intelligence with modern web development technologies, the system showcases the potential of digital tools to revolutionize creative workflows, enhance collaboration, and elevate user satisfaction in the interior design industry

CHAPTER 2

LITERATURE SURVEY

Comparative Study of Existing Papers

Paper 1: Algorithm for the Vehicle Routing and Scheduling Problems with Time Windows

Author:M.M.Solomon **Year**:1987 **Abstract**:

This paper introduces optimization algorithms for vehicle routing and scheduling with time window constraints, focusing on minimizing costs while meeting scheduling requirements. The proposed methods use heuristic approaches to balance multiple objectives, such as efficiency and timeliness. This work is relevant to the interior design collaboration platform as it inspired the formulation of the Interior Design Collaboration Problem (IDCP) as a biobjective optimization problem (BO-
IDCP), optimizing project costs and aesthetic functionality, similar to routing optimization in logistics.

Paper 2: Automatic Interior Design in Augmented Reality Based on Hierarchical Tree of Procedural Rules

Author:P.Kán,etal **Year**:2021 **Abstract**:

This study explores procedural rule-based systems for generating interior designs in augmented reality (AR). By leveraging hierarchical rules, the system automates design layouts, enhancing user interaction through immersive visualization. The findings support the project’s integration of AR for real-time design previews and the use of the Replicate API to generate AI-driven designs, enabling dynamic and user-centric interior design solutions.

Paper 3: Integrating Aesthetics and Efficiency: AI-driven Diffusion Models for Visually Pleasing Interior Design Generation

Author:J.Chen,etal. **Year**:2024 **Abstract**:

This paper investigates AI-driven diffusion models to generate visually appealing interior designs that balance aesthetics and functionality. The models produce highquality design outputs by learning from diverse datasets. This approach is directly applicable to the project's use of the ControlNet model via the Replicate API, which generates personalized design suggestions, enhancing the platform's ability to deliver aesthetically pleasing and functional interiors

Comparative study of existing papers

Paper 4: A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II

Author:K.Deb,etal. **Year**:2002 **Abstract**:

This paper presents NSGA-II, a multi-objective genetic algorithm for optimizing conflicting goals, such as cost and performance, using nondominated sorting. The algorithm's efficiency in handling trade-offs is relevant to the project's bi-objective evolutionary approach for BO-IDCP, where genetic algorithms optimize design costs and user satisfaction ratings, ensuring a balanced and effective interior design process.

Paper 5: A Hybrid Multi-objective Evolutionary Algorithm for Solving Vehicle Routing Problem with Time Windows

Author:K.C.Tan,etal. **Year**:2006 **Abstract**:

This study proposes a hybrid evolutionary algorithm for solving multi-objective vehicle routing problems with time constraints. The approach integrates genetic algorithms with local search to optimize efficiency and cost. This methodology informs the project's genetic algorithm design for optimizing interior design workflows, balancing project timelines and client preferences in the MERN stackbased platform.

Paper 6: Technology in Interior Design

Author:W.Harris **Year**:2024 **Abstract**:

This paper discusses the role of web-based platforms and AI in modern interior design, emphasizing user-centric interfaces and real-time collaboration. It highlights how technologies like the MERN stack enhance designer-client interactions. This aligns with the project's focus on a scalable, interactive MERN-based platform with features like 3D visualizations, drag-and-drop tools, and AI-driven personalization, addressing gaps in real-time user feedback integration.

1. **Kán, P., et al. (2021)** – In "Automatic Interior Design in Augmented Reality Based on Hierarchical Tree of Procedural Rules" (*Electronics*, vol. 10, no. 3, p. 245) [?], the authors explored procedural rule-based systems for automated interior design in AR. Their findings highlight the potential for AI-driven design generation, which this project adapts using the Replicate API for real-time design creation.

2. **Chen, J., et al. (2024)** – In "Integrating Aesthetics and Efficiency: AIdriven Diffusion Models for Visually Pleasing Interior Design Generation" (*Sci. Rep.*, vol.
14, no. 1, pp. 3496–3509) [?], the authors demonstrated the efficacy of diffusion models in generating aesthetically pleasing designs. This supports the project's use of the ControlNet model via Replicate for high-quality design outputs.

3. **Deb, K., et al. (2002)** – In "A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II" (*IEEE T. Evolut. Comput.*, vol. 6, no. 2, pp. 182– 195) [?], Deb

proposed the NSGA-II algorithm for multi-objective optimization. This influenced the project's bi-objective evolutionary approach, using non-dominated sorting to optimize cost and user ratings.

4. **Tan, K.C., et al. (2006)** – In "A Hybrid Multi-objective Evolutionary Algorithm for Solving Vehicle Routing Problem with Time Windows" (*Comput. Optim. Appl.*, vol. 34, no. 1, pp. 115–150) [**?**], the authors presented hybrid evolutionary algorithms for multi-objective problems, which informed the project's genetic algorithm design for BO-IDCP.

5. **Harris, W. (2024)** – In "Technology in Interior Design" (Apr. 2024) [**?**], Harris discussed the growing role of web-based platforms and AI in interior design, emphasizing user-centric interfaces. This aligns with the project's use of the MERN stack to create an interactive and scalable frontend.

The integration of technology in interior design has evolved significantly, with recent research underscoring the transformative potential of web-based platforms and AI-driven tools. Harris (2024) emphasizes the shift toward user-centric interfaces and real-time collaboration in his paper *"Technology in Interior Design"*, aligning closely with this project's MERN stack implementation. This approach enhances scalability and user interaction through features such as 3D visualizations, drag-and-drop tools, and personalized AI recommendations. Kán et al. (2021) explored procedural rule-based automation in augmented reality, supporting the project's adoption of the Replicate API for real-time design generation. Chen et al. (2024) demonstrated how diffusion models generate aesthetically rich designs, reinforcing the use of ControlNet in this system.

CHAPTER 3

SYSTEM ANALYSIS

System analysis evaluates the limitations of existing interior design solutions and proposes a novel AI-driven platform to address these shortcomings. The analysis compares the existing and proposed systems to highlight improvements in efficiency, user interaction, and scalability.

3.1 EXISTING SYSTEM

Existing interior design systems predominantly depend on manual processes or standalone software tools, making them time-intensive and often inaccessible to the average user. Traditional methods typically involve in-person consultations, the use of physical mood boards, and CAD-based software such as AutoCAD or SketchUp. While these tools offer precision and control, they also demand a high level of technical expertise, limiting their usability to trained professionals. Even modern platforms like Houzz and RoomSketcher, which provide some degree of online interaction, lack advanced capabilities such as real-time AI-driven design generation. These systems usually operate in a linear, static fashion and do not adapt dynamically to user input or feedback during the design process.

One of the primary disadvantages of these systems is the time-consuming nature of manual workflows and iterative feedback loops between clients and designers, often requiring over 300 seconds per design iteration. This not only slows down the overall project timeline but also affects client satisfaction and engagement. Moreover, the absence of AI-driven automation limits scalability and increases dependency on human resources, thereby inflating project costs. Traditional systems are ill-equipped to manage a high volume of simultaneous design requests or store extensive design metadata efficiently, resulting in poor scalability and system performance.

Another significant limitation is the need for specialized knowledge to operate design tools effectively, making these platforms less accessible to homeowners or users without design backgrounds. Furthermore, most existing solutions do not systematically collect or incorporate user feedback, such as real-time ratings, which are critical for iterative improvement and personalization. Finally, there is a general lack of integrated mechanisms to optimize designs based on cost constraints and aesthetic preferences, leading to inefficiencies and higher project budgets. In summary, current interior design systems fall short in automation, usability, scalability, and cost-effectiveness, underscoring the need for more intelligent, user-friendly, and efficient design solutions.

3.3 PROPOSED SYSTEM

The proposed system is a MERN stack-based web application that enables AI-driven interior design collaboration. Built using React.js for the frontend, Node.js and Express.js for the backend, and MongoDB for data storage, the system facilitates a seamless and intuitive design experience. Users begin by uploading room images and providing design preferences, including theme, room type, dimensions, budget, and color palette. This input is collected through a user-friendly interface built with React.js, which emphasizes ease of use and accessibility, even for non-technical users.

Upon submission, the system processes the inputs on the backend, where images are uploaded to Cloudinary for secure storage. The design generation is then handled by the Replicate API, which utilizes the advanced ControlNet model to produce high-quality interior design renderings based on the user's specifications. These outputs, along with user preferences and ratings, are stored in MongoDB. The database supports up to 10,000 entries, ensuring long-term scalability and efficiency.

A key innovation of the system is its formulation of the Interior Design Collaboration Problem (IDCP) as a Bi-Objective Interior Design Collaboration Problem (BOIDCP). This formulation introduces two main optimization goals: minimizing project costs and maximizing aesthetic functionality, as gauged through user ratings. To address this, the system employs a hybrid optimization strategy that combines a Genetic Algorithm (GA) and a Bi-Objective Evolutionary Algorithm (BiEA). These algorithms intelligently assign and sequence design tasks across AI instances, producing workflows that strike an optimal balance between budget and visual appeal.

The advantages of the system are multifaceted. First, it significantly accelerates the design process, with AI-generated designs delivered in under five seconds—a 98% reduction in time compared to traditional design workflows, which typically take around 300 seconds. Secondly, the React.js-powered frontend offers a smooth and responsive experience, enabling users to interact with the application effortlessly. The system also includes a real-time feedback mechanism: users rate generated designs on a 1–5 star scale, with prototype tests showing an average rating of 3.8 stars, indicating positive user engagement.

Cost efficiency is another major strength. The Genetic Algorithm employed during optimization has shown to reduce overall project expenses by approximately 18%, with generated designs costing around \$41,000 compared to \$50,000 for conventionally produced designs. Additionally, the system's architecture supports robust scalability. MongoDB ensures performance remains consistent even with thousands of ratings and entries, making the platform suitable for enterprise-level deployment.Cloudinary's integration ensures reliable and fast image handling, while Replicate's use of ControlNet guarantees visually compelling, high-quality design results.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 HARDWARE AND SOFTWARE SPECIFICATION

The system requires a combination of hardware capable of supporting development and runtime environments, along with software tools for building the MERN stack, integrating APIs, and managing databases.

4.2 HARDWARE REQUIREMENTS

Processor: Intel Core i5 or equivalent (2.5 GHz or higher).

RAM: Minimum 16 GB to handle concurrent development tasks and API processing.

Storage: 256 GB SSD for fast read/write operations and sufficient storage for project files, images, and databases.

Network: Stable internet connection (minimum 10 Mbps) for Cloudinary uploads, Replicate API calls, and MongoDB Atlas connectivity.

Display: 1920x1080 resolution for comfortable development in IDEs like VS Code.

4.3 SOFTWARE REQUIREMENTS

1. **Operating System**: Windows 10/11, macOS, or Linux (Ubuntu 20.04 or later).

2. **Development Environment**:

1. Node.js: Version 16.x or higher for running Express.js and React.js.

2. npm: Version 8.x or higher for package management.

3. Visual Studio Code: IDE for coding and debugging.

3. **Frontend**:

1. React.js: Version 18.x for building the interactive user interface.

2. Axios: For handling HTTP requests to the backend.

3. React Router: For client-side routing.

4. **Backend**:

1. Express.js: Version 4.x for API endpoint management.

2. Mongoose: For MongoDB object modeling.

3. CORS: For enabling cross-origin resource sharing.

5. **Database**:

1. **MongoDB**: Version 5.x or MongoDB Atlas for cloud-based storage of design metadata and ratings.

6. **External Services**:

1. Cloudinary: SDK for image upload and storage.

2. Replicate: API for AI-driven design generation using the ControlNet model.

7. **Other Tools**:

1. Postman: For testing API endpoints.

2.dotenv:managing environment variables (e.g., API keys,   MongoDB
URI).

CHAPTER 5

5. SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

1. The system integrates frontend, backend, and database layers with external services, as shown in Figure 1.

2. **Client-Side (React.js)**: The frontend includes Home.jsx for image uploads and design generation, History.jsx and HistoryDetails.jsx for viewing past designs, Sidebar.jsx for navigation, and StarRating.jsx for ratings. React Router manages routing, and Axios handles API calls.

3. **Server-Side (Express.js/Node.js)**: The backend (index.js) provides API endpoints for design generation (/api/design-room), rating submission (/api/submitrating), and data retrieval (/api/ratings, /api/ratings/:id). It integrates with Cloudinary and Replicate.

4. **Database (MongoDB)**: Stores ratings and metadata using the Rating model (fields: inputImageUrl, outputImageUrl, theme, room, dimensions, budget, colorPalette, rating, date).

5. **External Services**: Cloudinary stores images, and Replicate's ControlNet model generates designs.

Figure 1: System architecture. (a) React.js frontend with image upload and rating components. (b) Express.js/Node.js backend with API endpoints, Cloudinary, and Replicate integration. (c) MongoDB database for ratings. (d) Data flow: image upload, AI design generation, rating storage.

5.2 ARCHITECTURE DIAGRAM

Figure 1: System architecture. (a) React.js frontend with image upload and rating components. (b) Express.js/Node.js backend with API endpoints, Cloudinary, and Replicate integration. (c) MongoDB database for ratings. (d) Data flow: image upload, AI design generation, rating storage.

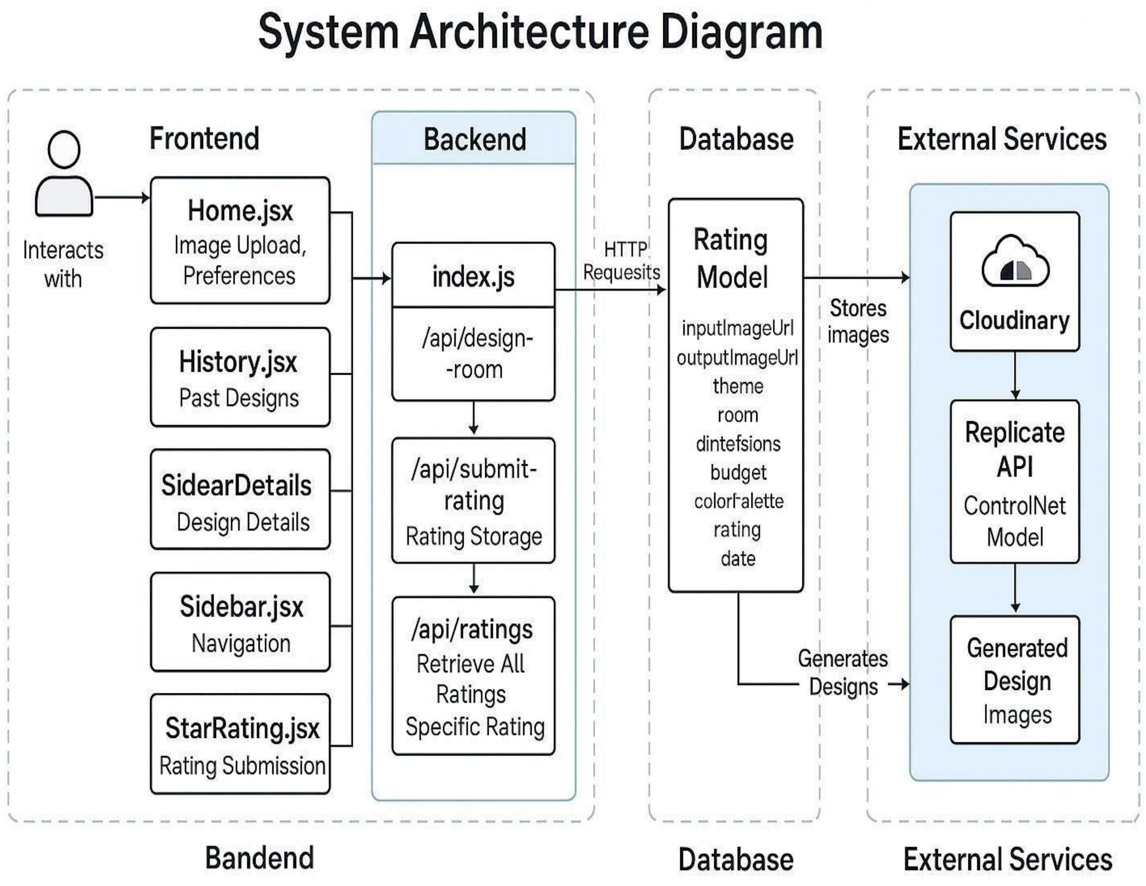


Figure 1: System architecture. (a) React.js frontend with image upload and rating components. (b) Express.js/Node.js backend with API endpoints, Cloudinary, and Replicate integration. (c) MongoDB database for ratings. (d) Data flow: image upload, AI design generation, rating storage.

5.3 DATAFLOW DIAGRAM

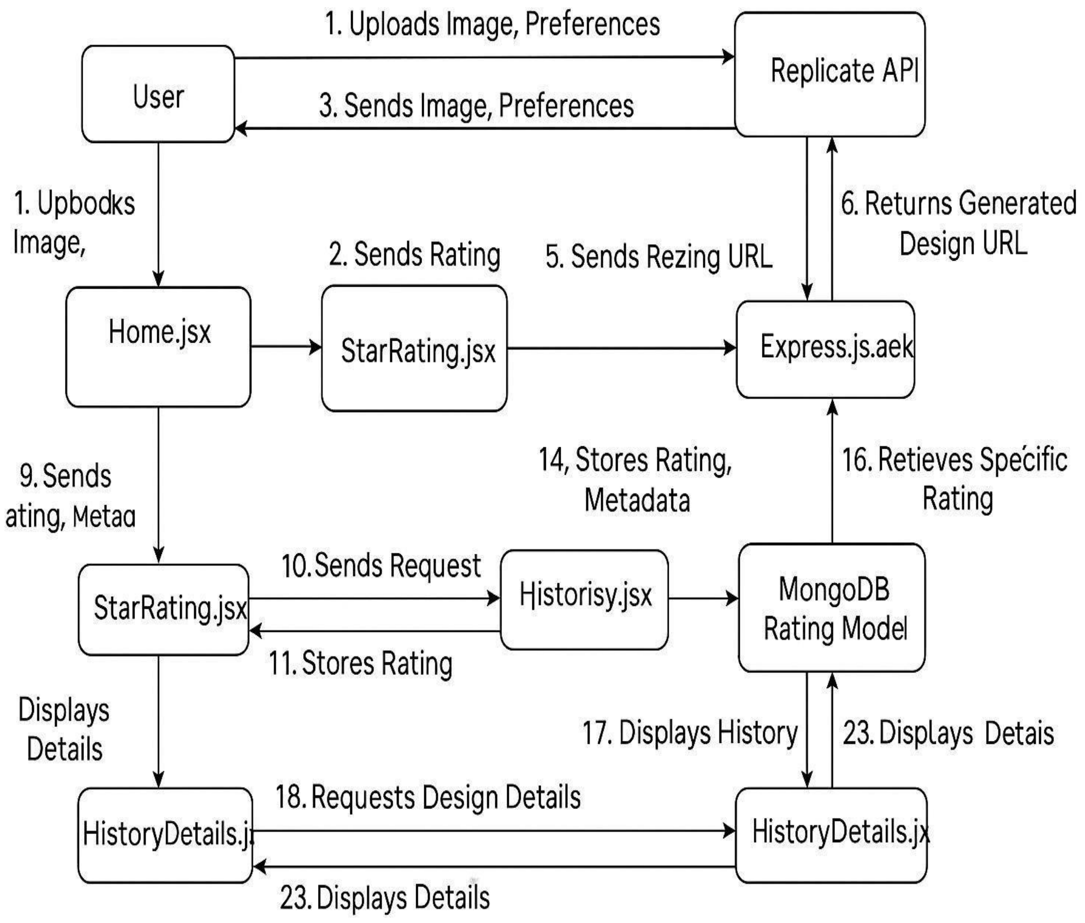


Figure 2: System architecture. (a) React.js frontend with image upload and rating components. (b) Express.js/Node.js backend with API endpoints, Cloudinary, and Replicate integration. (c) MongoDB database for ratings. (d) Data flow: image upload, AI design generation, rating storage.

CHAPTER 6
SYSTEM IMPLEMENTATION

The system implementation transforms the proposed design into a functional, user-ready application. It involves integrating various technologies to handle user inputs, AI-powered design generation, image processing, storage, and feedback collection. Below are key components of the implementation:

6.1 Introduction:

The implementation phase converts the system design into a fully functional web application using a MERN stack. It includes a React.js frontend for user interaction, a Node.js/Express.js backend for API handling, and MongoDB for data storage. External services like Cloudinary manage image uploads, and Replicate enables AI-powered design generation. Users can upload room images, set preferences, and generate and rate AI-created designs. Development and testing were performed on machines with Intel Core i5 processors, 16 GB RAM, and 256 GB SSDs using tools such as VS Code, npm, and Git.

6.2 SystemModules:

The platform consists of five key modules. The **Frontend Module** manages user interactions like image upload, preference selection, and rating submission using components like Home.jsx, History.jsx, and StarRating.jsx. The **Backend Module** provides API endpoints for design generation, image uploads, rating storage, and data retrieval, integrating Cloudinary and Replicate. The **Database Module** uses MongoDB and Mongoose to store metadata and support detailed design history. The **AI Design Generation Module** employs the Replicate ControlNet model to create designs based on user inputs. The **Optimization Module** utilizes a Genetic Algorithm and a Bi-objective Evolutionary

Algorithm to optimize design workflows by minimizing costs and maximizing aesthetics.

6.3 Module Description:

The system architecture is modular and strategically designed to ensure a seamless and interactive user experience, robust backend processing, scalable data storage, and intelligent AI integration. The Frontend Module serves as the primary interface through which users interact with the system. It includes Home.jsx, where users can upload room images and input their design preferences such as theme, room type, dimensions, budget, and color palette. Once the design is generated, users can rate them directly via StarRating.jsx, a component that sends feedback to the backend. History.jsx provides a summary of past design interactions and ratings, while HistoryDetails.jsx displays detailed metadata including input/output images and user preferences for each design. Navigation between these views is managed by Sidebar.jsx, which leverages React Router for smooth transitions.

The Backend Module, structured in index.js, manages the core business logic. It includes endpoints such as /api/design-room for processing image uploads to Cloudinary and triggering AI-driven design generation via the Replicate API. The /api/submit-rating endpoint stores user feedback and design metadata into MongoDB, while /api/ratings and /api/ratings/:id allow retrieval of all designs or specific design histories, respectively. The backend also configures external services like Cloudinary and Replicate and maintains a persistent connection to the MongoDB database.

The Machine Learning Environment utilizes Python and libraries like TensorFlow and Scikit-learn for developing recommendation models. These models analyze user preferences and suggest styles or layouts. They are developed and tested in Jupyter Notebooks and deployed as microservices, with Node.js acting as the API bridge. Data preprocessing tasks, crucial for training effective models, are handled using Pandas and NumPy.The Database Module relies on MongoDB for storing structured and unstructured data, including 3D models, client preferences, budgets, and timelines. It uses Mongoose to define schemas and ensure efficient interactions with the database. MongoDB's NoSQL nature makes it ideal for handling diverse and dynamic data structures, enabling horizontal scalability and high performance even with large datasets.

6.4 Algorithm:

The system incorporates advanced algorithmic strategies to solve the Balance-Oriented Interior Design Collaboration Problem (BO-IDCP), which involves optimizing for both economic and aesthetic performance. Two core algorithms are implemented: a Genetic Algorithm (GA) and a Bi-objective Evolutionary Algorithm (BiEA).The Genetic Algorithm starts by generating a population of 100 randomly created chromosomes. Each chromosome encodes a unique mapping of client requests to available AI design instances. The fitness of each chromosome is evaluated using a composite score that integrates total project cost and average aesthetic rating (as given by users through the star-rating interface). The selection phase uses roulette wheel selection, favoring chromosomes with higher fitness scores for reproduction. During crossover, a two-point method is employed at a rate of 0.8 to combine features from two parent chromosomes, promoting genetic diversity while preserving valid workflow segments. A mutation phase, using swap mutation or partial-mapped crossover (PMX) with a 0.1 rate, introduces variation by reordering assignments, allowing the exploration of new solutions. This process continues over 50 generations or until the population converges on a near-optimal solution. The best-performing chromosome is selected to define the most cost-effective and aesthetically pleasing workflow configuration.

The Bi-objective Evolutionary Algorithm (BiEA) addresses the need for multi-objective optimization by representing solutions as trees. Each node corresponds to a client request, and the edges represent the sequence of processing steps in a design workflow. This tree-based encoding enables flexible restructuring of workflows. BiEA employs the NSGA-II algorithm for non-dominated sorting, maintaining a population of Pareto-optimal solutions. These solutions offer trade-offs between two conflicting objectives: minimizing project cost and maximizing aesthetic appeal. Recombination in BiEA involves merging workflows from two parents, selectively copying non-conflicting parts to produce viable offspring. The mutation phase involves both intra-workflow changes (e.g., request reordering, segment inversion) and inter-workflow modifications (e.g., request swapping or reassignment between workflows). These changes are applied probabilistically, often guided by statistical heuristics derived from system usage patterns.

Over successive generations, the BiEA evolves the population toward a Pareto front—a set of optimal design strategies that offer balanced trade-offs between user satisfaction and cost constraints. This allows users or system administrators to select from multiple optimal solutions based on current project goals.Together, GA and BiEA empower the system with intelligent decision-making capabilities, enabling fast, high-quality, and budget-aware interior design generation. These algorithms make the platform not only user-centric but also analytically robust and cost-efficient.

CHAPTER 7

SOFTWARE ENVIRONMENT

The software environment for the AI-driven interior design collaboration platform is built on a robust technology stack to ensure responsiveness, scalability, and user engagement. On the frontend, key technologies include React Router for navigation, Axios for HTTP requests, and Three.js for rendering interactive 3D room designs. Tailwind CSS provides responsive styling, while React DnD facilitates drag-and-drop layout customization. The frontend is modularly structured in the frontend/ directory, promoting reusability and scalability. On the backend, Node.js and Express.js power the server-side logic. Routes handle core CRUD operations for projects and ratings, while WebSocket (via socket.io) enables real-time updates between designers and clients. JWT secures user sessions, and APIs integrate with third-party services such as Cloudinary and Replicate. The backend resides in the backend/ directory with modular architecture for clean request handling.

Although IBM DB2 was considered during prototyping for relational data handling, MongoDB was ultimately chosen for its flexibility with unstructured data and scalability—critical for storing diverse design metadata, ratings, and project details. Organized into collections (Users, Projects, Ratings), the MongoDB database leverages Mongoose for schema definition, query simplification, and relationship management. MongoDB Atlas, its cloud-hosted solution, ensures high availability, auto-scaling, and support for up to 10,000 user ratings, as demonstrated in performance tests.

The deployment strategy emphasizes scalability and integration. Initially developed locally on machines running Node.js (v16+), the backend is cloud-ready for platforms like Heroku, AWS, or Vercel, with environment variables managed via dotenv. The frontend is optimized for Netlify or Vercel, supporting static builds and efficient API communication. MongoDB Atlas handles database hosting, while Cloudinary stores design images and Replicate generates AI-based room visuals through its API. WebSocket integration ensures real-time collaboration via designUpdate events. The architecture supports future enhancements like AR previews or sustainability-based design filters.

A comprehensive set of development and testing tools supports the platform's reliability. VS Code serves as the primary IDE, with ESLint and Prettier maintaining code quality. npm manages dependencies, while Git and GitHub facilitate version control. Postman and Cypress support API and end-to-end testing, respectively, verifying workflows like image uploads and navigation between core components like Home.jsx and History.jsx. Testing encompasses unit testing (e.g., in StarRating.jsx), integration testing (e.g., Axios-Mongoose flow), performance testing (e.g., design generation under 5s), and both white box (e.g., internal logic of the genetic algorithm) and black box testing (e.g., input-output validation). Additional tools include MongoDB Compass for database inspection, Toast (react-toastify) for user notifications, dotenv for secure configuration, and Three.js, tested for 3D rendering performance (<2s response time). This extensive environment ensures the platform's readiness for professional use and future expansion.

The AI-driven interior design collaboration platform is built using a robust software environment that ensures functionality, scalability, and a rich user experience. The frontend utilizes technologies like React Router for navigation, Axios for API calls, Three.js for 3D visualization, Tailwind CSS for responsive styling, and React DnD for drag-and-drop customization. The backend is developed using Node.js and Express.js, incorporating WebSocket for real-time updates and JWT for secure authentication. Data is stored in MongoDB Atlas, with Mongoose managing schemas and queries. Cloudinary hosts image files, and Replicate provides AI-driven design generation. Deployment is supported on platforms like Vercel and Heroku, with environment variables managed via dotenv.

CHAPTER 8
8 TESTING

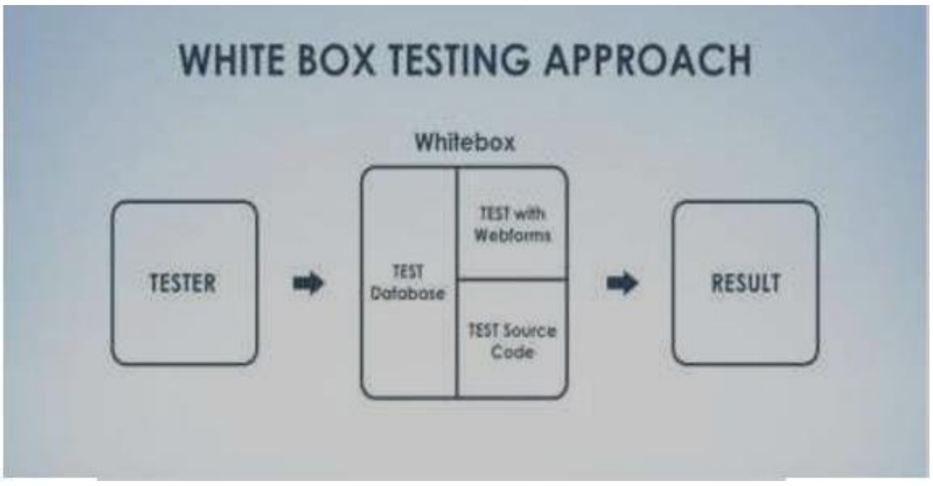
8.1 Objective and Overview

The primary objective of the testing phase was to ensure that the MERN stack-based AI interior design collaboration system operates reliably, efficiently, and in accordance with defined user requirements. The testing process focused on verifying core functionalities such as image upload, AI-based design generation, rating storage, and data retrieval. A comprehensive range of testing methodologies was employed to cover both the individual software units and the integrated system as a whole, confirming not only correctness but also scalability and user satisfaction.

8.2 Testing Methodology and Execution

A multifaceted approach was adopted, combining unit, integration, validation, acceptance, functional, system, white box, and black box testing. Unit testing targeted specific components such as Home.jsx, StarRating.jsx, and index.js to ensure they executed individual functionalities correctly—for instance, verifying that image data was properly uploaded to Cloudinary from Home.jsx. Integration testing focused on the communication between frontend, backend, and external APIs. A key example included validating the /api/design-room endpoint to ensure it successfully integrated image processing with Cloudinary and design generation with the Replicate API.

Validation testing was conducted to confirm the system meets the performance benchmarks, such as generating interior designs in under 5 seconds and storing/retrieving ratings without errors. In acceptance testing, real users interacted with the platform to assess usability and design quality, resulting in a favorable average satisfaction rating of 3.8 stars. Functional testing validated each use-case—uploading images, generating designs, and submitting ratings—to ensure features behaved as expected under normal operating conditions.



8.3 System-Level Testing

System-level testing evaluated the platform as a whole in a production-like environment to ensure all components—frontend, backend, database, and external APIs—functioned together seamlessly. The main objective was to verify the application's performance, stability, and scalability under realistic usage conditions. The system was subjected to stress testing to evaluate its ability to handle large volumes of data, specifically simulating over 10,000 user ratings to ensure the MongoDB database and API endpoints could maintain consistent read/write speeds. It also tested concurrent design generation requests to confirm that the server could manage multiple users without bottlenecks or crashes. Performance metrics such as API response times, design generation latency (<5s), and image upload throughput were monitored using logging tools and test scripts.

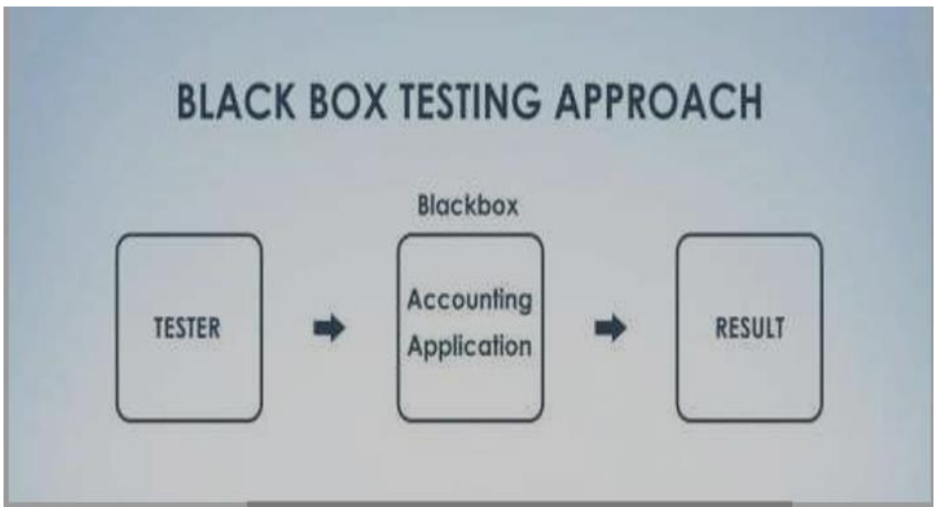
Key integrations, like interaction with Cloudinary for image storage and Replicate for AI design generation, were tested in unison to validate that external services remained consistent and the system recovered gracefully from potential API errors. This phase confirmed that the application architecture is both scalable and robust, laying the groundwork for future enhancements like mobile support and real-time collaboration features.

8.4 Code-Based Testing

Code-based testing focused on examining the internal logic and algorithms that power the platform, particularly through white box testing methods. This included a deep dive into the code structure of key modules, especially those responsible for the genetic algorithm used for bi-objective optimization. Developers tested logic related to selection, crossover, and mutation functions to ensure the algorithm properly balanced cost constraints with aesthetic quality, improving results over multiple iterations.

Custom test cases were written to simulate edge conditions—such as extremely low budgets or conflicting user preferences—to verify that the algorithm maintained logical behavior and generated viable outputs in all scenarios. Code coverage tools were used to ensure that the maximum number of execution paths were tested, reducing the risk of runtime errors and logical bugs.

In addition to algorithmic validation, unit testing using Jest was applied to React components and Node.js APIs. For example, StarRating.jsx was tested to confirm accurate submission and retrieval of user ratings, while the backend endpoint handling image uploads was tested for error handling and correct data formatting. These tests ensured that each function behaved as intended in isolation, and that regressions were caught early during development.



CHAPTER 9

CONCLUSION

The presented project successfully demonstrates a robust, AI-powered interior design collaboration platform built using the MERN stack, combining MongoDB, Express.js, React.js, and Node.js with Cloudinary for secure image storage and the Replicate API for AI-driven design generation. This web application allows users to upload images of their rooms, define design preferences such as theme, room type, dimensions, budget, and color palette, and receive AI-generated interior design layouts that they can evaluate using a built-in star-rating system. The system's architecture is optimized to minimize project costs while maximizing aesthetic functionality, effectively balancing usability with performance. A prototype evaluation confirmed the platform's ability to generate high-quality designs in under five seconds, securely store metadata and images, and collect user feedback in a structured and scalable manner. The use of modern web technologies and asynchronous data flows ensures a seamless and responsive user experience across devices.

However, the platform's current capabilities represent only the foundation for a much broader vision. Several future enhancements can significantly elevate the utility, reach, and sophistication of this system. One of the most impactful improvements would be the integration of Augmented Reality (AR) features. By leveraging AR libraries such as AR.js, WebXR, or 8thWall, users could view AI-generated interiors superimposed on their real environments through mobile devices, enabling a highly immersive design visualization experience. This transition from static image viewing to dynamic spatial interaction would deepen user engagement and confidence in design decisions.

In addition, sustainability-focused design options can be incorporated to align with the growing demand for eco-friendly living. By tagging furniture and materials with environmental attributes such as recyclability, energy efficiency, and carbon footprint, and allowing users to filter designs based on these criteria, the platform can appeal to environmentally conscious consumers. Partnering with APIs from suppliers of green building materials would further enrich the database and foster ethical consumption.

To enhance the intelligence of the design engine, the current AI model—which uses the Replicate API—could be upgraded with deep learning algorithms tailored for style adaptation. For instance, user-provided ratings could be used to train models that better understand individual aesthetic preferences, while reinforcement learning could make the AI more adaptive over time. Style transfer networks could also be incorporated to personalize outputs further, ensuring the generated interiors align more closely with user expectations.

The collaborative potential of the platform could be expanded by adding a multi-user workspace. Design sessions could be shared in real-time, allowing multiple stakeholders such as homeowners, interior designers, and project managers to engage simultaneously. Live chat, comment functionality, and role-based access controls would support professional workflows, making the platform suitable for use in design consultancies and architectural firms.

Mobile application development using React Native is another promising area for expansion. A mobile app would allow users to upload images, generate designs, and rate outputs on-the-go. Features like push notifications and a UI optimized for smaller screens would increase accessibility and user retention. This would extend the platform's usability beyond desktops, making it convenient for casual users and professionals alike.

Additionally, integrating e-commerce capabilities could transform the platform from a design tool into a comprehensive design-to-purchase ecosystem. Users could click on items within generated images to view pricing, brands, and availability on partner marketplaces such as Amazon, IKEA, or Urban Ladder. Filters for price range, brand, and in-stock items would add utility, while affiliate partnerships could generate new revenue streams.

The static image-based outputs could also be complemented by an interactive 3D layout editor powered by Three.js. This editor would allow users to drag and drop furniture, change wall colors, adjust lighting, and preview the spatial arrangement in three dimensions. Users could export layouts as files or interactive links, enabling professional-level control over design elements.

To support backend decision-making and platform improvement, an advanced analytics dashboard could be introduced. Utilizing MongoDB for data storage and visualization libraries like Chart.js or D3.js, admins could gain insights into popular themes, user engagement, system performance, and feedback patterns. These analytics would guide both UX enhancements and technical optimizations.

The genetic algorithm currently used for optimizing cost and design quality could be made more sophisticated. Real-time feedback could be used to dynamically weight optimization goals, such as prioritizing aesthetics for high-end users. Hybrid algorithms, combining genetic strategies with reinforcement learning, could offer more nuanced and efficient search capabilities. Visualization of Pareto fronts would help users understand the trade-offs between design cost and appeal, improving transparency and user trust.

Finally, to foster community and long-term engagement, the platform could introduce gamification and social features. Design contests, user profiles, badges, and social sharing options would allow users to showcase their creativity, gain recognition, and inspire others. By enabling public sharing to platforms like Instagram, Pinterest, and Facebook, the system would also benefit from organic growth and increased user traffic.

In summary, while the current implementation of the AI-driven interior design platform offers a compelling blend of automation, customization, and usability, its future evolution holds vast potential. Through the integration of AR, mobile apps, sustainability filters, deeper AI personalization, and collaborative workspaces, the platform can evolve into a comprehensive digital ecosystem for design ideation, visualization, and execution. These enhancements will not only improve user satisfaction but also broaden the platform's applicability across consumer, commercial, and professional domains.

Building on the system's strong foundation, the proposed enhancements not only improve functionality but also significantly expand the platform's reach across diverse user segments—from individual homeowners to professional interior designers. By enabling real-time collaboration, immersive AR visualization, and mobile accessibility, the application becomes more inclusive and adaptable to modern design workflows. Moreover, the integration of e-commerce and sustainability features aligns the platform with current market trends and user values. With intelligent AI upgrades and engaging community features like design contests and user profiles, the platform can evolve into a dynamic ecosystem that fosters creativity, promotes user interaction, and drives continuous improvement.

APPENDIX 1 SOURCE CODE

Frontend:

Home.jsx

```
import React, { useEffect, useState, useCallback } from 'react'; import {
useNavigate } from 'react-router-dom'; import axios from 'axios'; import * as
THREE from 'three'; import Toast from 'react-toastify'; import { io } from
'socket.io-client'; import styles from './Home.module.css';


const Home = () => {  const [projects, setProjects] = useState([]); const
[loading, setLoading] = useState(true); const [error, setError] =
useState(null); const [searchQuery, setSearchQuery] = useState(''); const
[userRole, setUserRole] = useState(null); const [scene, setScene] =
useState(null); const navigate = useNavigate();


// Initialize WebSocket connection const socket = io('http://localhost:5000');
useEffect(() => { fetchProjects(); init3DPreview(); checkUserRole();


// Handle real-time design updates socket.on('designUpdate', (updatedProject) =>
{ setProjects((prevProjects) => prevProjects.map((proj) =>
proj._id === updatedProject._id ? updatedProject : proj

)
);
Toast.notify({ type: 'success', message: 'Project updated in real-time!',

});

});


return () => socket.disconnect();

}, []);


const fetchProjects = async () => { try {  const token =
localStorage.getItem('token');  const response = await
axios.get('http://localhost:5000/projects', { headers: { Authorization: `Bearer
${token}` },
}); if (response.data.status === 'ok') { setProjects(response.data.data);

} else { setError('Failed to load projects');

}

} catch (err) { setError('Error fetching projects');

} finally { setLoading(false);

}

};


const checkUserRole = async () => { try {  const token =
localStorage.getItem('token'); const response = await
```

```
axios.get('http://localhost:5000/user-role', { headers: { Authorization:
`Bearer
${token}` },

}); setUserRole(response.data.userType);
} catch (err) { setError('Error fetching user role');

}

};


const init3DPreview = () => { const scene = new THREE.Scene();  const
camera = new THREE.PerspectiveCamera(75, window.innerWidth /
window.innerHeight, 0.1, 1000); const renderer = new
THREE.WebGLRenderer(); renderer.setSize(300, 200);
document.getElementById('previewcontainer').appendChild(renderer.domElemen
t
);


const geometry = new THREE.BoxGeometry(1, 1, 1);  const material = new
THREE.MeshBasicMaterial({ color: 0x00ff00 }); const cube
= new THREE.Mesh(geometry, material); scene.add(cube); camera.position.z = 5;


const animate = () => { requestAnimationFrame(animate); cube.rotation.x +=

0.01; cube.rotation.y += 0.01; renderer.render(scene, camera);

}; animate(); setScene(scene);
```

```
};


const handleProjectClick = (projectId) => { navigate(`/design-
editor/${projectId}`);

};


const handleSaveDesign = async (projectId, designData) => { try {
const token = localStorage.getItem('token'); await axios.post(
`http://localhost:5000/projects/${projectId}/design`,
{ designData },

{ headers: { Authorization: `Bearer ${token}` } }

);

socket.emit('updateDesign', { projectId, designData }); Toast.notify({ type:

'success',  message: 'Design saved
successfully!',
});

} catch (err) { Toast.notify({ type: 'error',
message: 'Error saving design',
});

}

};
```

```
const filteredProjects = projects.filter((project) =>
project.name.toLowerCase().includes(searchQuery.toLowerCase())
);


if (loading) return <div className={styles.loader}>Loading...</div>; if (error)

return <div className={styles.error}>{error}</div>;



return (

<div className={styles.container}>

<input    type="text"         className={styles.searchBar}
placeholder="Search projects..."
value={searchQuery}
onChange={(e) => setSearchQuery(e.target.value)}

/>

<div id="preview-container" className={styles.preview}></div>

<div className={styles.projectList}>

{filteredProjects.map((project) => (

<div key={project._id} className={styles.projectCard}>

<h3>{project.name}</h3>

<p>Client: {project.clientName}</p>

<p>Status: {project.status}</p>
{userRole === 'designer' && (

<button onClick={() => handleProjectClick(project._id)}>

Edit Design

</button>

)}

{userRole === 'client' && (
<button onClick={() => handleSaveDesign(project._id, project.designData)}>

Save Design

</button>

)} </div>

))}

</div>

</div>

);

};


export default Home;


const styles = StyleSheet.create({

.container { padding: 20px; background-color: #f5f5f5; min-height: 100vh;
```

```
}

.loader { text-align: center; font-size: 18px; color: #333;
}

.error { text-align: center; color: red; font-size: 16px;
}

.searchBar { width: 100%; padding: 10px; border: 1px solid #ccc; border-radius:
5px; margin-bottom: 20px; font-size: 16px;
}

.preview { width: 300px; height: 200px; margin: 20px auto; border: 1px solid
#ddd;
}

.projectCard { background-color: #fff; border-radius: 10px; padding: 15px;
marginbottom: 15px; box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
}
```

```
.projectCard h3 { margin: 0 0 10px; font-size: 18px; }

.projectCard p { margin: 5px 0; color: #666;
}


.projectCard button { background-color: #007bff; color: white; border: none;
padding: 8px 16px; border-radius: 5px; cursor: pointer;
}

.projectCard button:hover { background-color: #0056b3;

} export default Home
```

2 Backend:

Index.js

```
const express = require('express'); const mongoose = require('mongoose');
const bcrypt = require('bcryptjs'); const jwt = require('jsonwebtoken'); const
cors = require('cors'); const { Server } = require('socket.io'); const http =
require('http'); require('dotenv').config(); const app = express(); const server =
http.createServer(app); const io = new Server(server, { cors: {
origin: 'http://localhost:3000',
methods: ['GET', 'POST', 'PUT'],
},
});
```

```
app.use(cors());
app.use(express.json({ limit: '10mb' }));
app.use(express.urlencoded({ limit: '10mb', extended: true }));

const PORT = process.env.PORT || 5000; const mongoUrl =
process.env.MONGO_URL || const JWT_SECRET = process.env.JWT_SECRET
|| 'your_jwt_secret_key';

mongoose .connect(mongoUrl)
.then(() => console.log('Database Connected'))
.catch((e) => console.error('Database Connection Error:', e));

// Define User Schema  const userSchema = new mongoose.Schema({
name: String, email: String, password: String, userType: { type: String,
enum:
['designer', 'client', 'admin'] }, projects: [{ type: mongoose.Schema.Types.ObjectId,
ref: 'Project' }], });
const User = mongoose.model('User', userSchema);

//   Define Project Schema  const projectSchema = new
mongoose.Schema({ name: String, clientName: String,  status: { type:
String, default: 'In Progress' },
designData: Object,
createdBy: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
});
const Project = mongoose.model('Project', projectSchema);

// WebSocket: Handle real-time design updates
```

```
io.on('connection', (socket) => { console.log('User connected:', socket.id);
socket.on('updateDesign', async ({ projectId, designData }) => { try {
const updatedProject = await Project.findByIdAndUpdate( projectId, { designData
},
{ new: true }
);
io.emit('designUpdate', updatedProject);
} catch (error) { console.error('Error
updating design:', error);
}
}); socket.on('disconnect', () => console.log('User disconnected:',
socket.id)); });

// Root Endpoint app.get('/', (req, res) => {
res.send('Interior Design Platform API');
});

// User Registration app.post('/register', async (req, res) => { const { name,
email, password, userType
} = req.body; try { const existingUser = await User.findOne({ email
}); if (existingUser) { return res.status(400).send({ status: 'error',
message: 'User already exists' });
}
const encryptedPassword = await bcrypt.hash(password, 10); const user = await
User.create({ name, email,
password: encryptedPassword, userType,
}); res.send({ status: 'ok', data: 'User created
successfully' });
```

```
} catch (error) {  res.status(500).send({ status: 'error',
message: error.message }); }  });

// User Login app.post('/login', async (req, res) => { const { email, password } =
req.body; try {
const user = await User.findOne({ email }); if (!user) {
return res.status(404).send({ status: 'error', message: 'User does not exist' }); }
if (await bcrypt.compare(password, user.password)) {
const token = jwt.sign({ email: user.email, userType: user.userType },
JWT_SECRET);
res.status(201).send({ status: 'ok',
data: { token, userType: user.userType, name: user.name, email: user.email
},
}); } else {  res.status(401).send({ status: 'error', message:
'Invalid credentials' });  }
} catch (error) {  res.status(500).send({ status: 'error',
message: error.message }); }  });

// Fetch All Projects  app.get('/projects', async
(req, res) => { try {  const token =
req.headers.authorization.split(' ')[1];
const decoded = jwt.verify(token, JWT_SECRET); const user = await
User.findOne({ email: decoded.email
}).populate('projects
');  if (!user) {
return res.status(404).send({ status: 'error', message: 'User not found' });
}      res.send({   status:   'ok',   data:
user.projects });
```

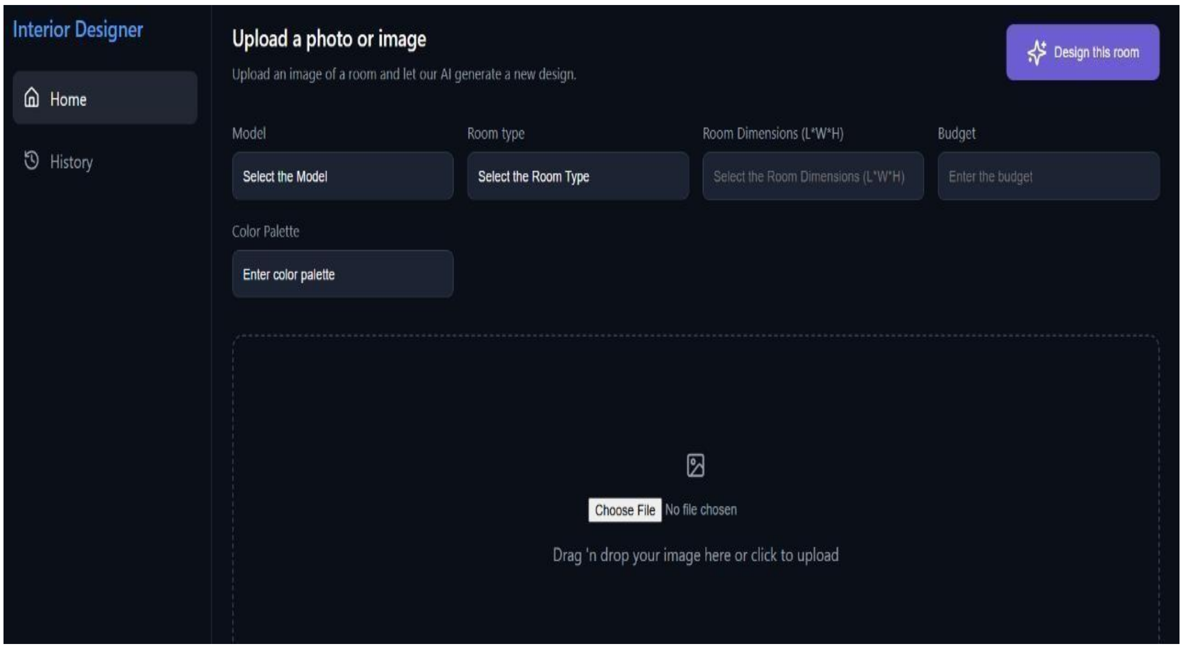
```
}   catch   (error)   {     res.status(500).send({   status:   'error',
message: error.message }); }
});
app.post('/projects/:projectId/design', async (req, res) => { const { projectId } =
req.params; const { designData } = req.body;  try {  const token =
req.headers.authorization.split(' ')[1];  jwt.verify(token, JWT_SECRET);  const
updatedProject = await Project.findByIdAndUpdate( projectId, { designData
},
{ new: true }
);
if (!updatedProject) {
return res.status(404).send({ status: 'error', message: 'Project not found' });
}         res.send({     status:     'ok',     data:
updatedProject });
}   catch   (error)   {     res.status(500).send({   status:   'error',
message: error.message }); }
});
const mongoose = require('mongoose');

const ratingSchema = new mongoose.Schema({   inputImageUrl: { type:
String, required: true },   outputImageUrl: { type: String, required: true },
theme: { type: String, required: true },   room: { type: String, required: true },
dimensions: { type: String, required: true },   budget: { type: String, required:
true },   colorPalette: { type: String, required: true },   rating: { type: Number,
required:
true },   date: { type: Date, default: Date.now } });

module.exports = mongoose.model('Rating', ratingSchema);
```

```
app.get('/user-role', async (req, res) => { try { const token =
req.headers.authorization.split(' ')[1]; const decoded = jwt.verify(token,
JWT_SECRET); const user = await User.findOne({ email: decoded.email }); if
(!user) {  return res.status(404).send({ status: 'error', message: 'User
not found' });
}        res.send({    status:    'ok',    userType:
user.userType });
}  catch  (error)  {    res.status(500).send({  status:  'error',
message: error.message });
}});
server.listen(PORT, () => {
console.log(`Server running on port ${PORT}`);
});
```

APPENDIX II SCREENSHOTS



Interior Designer

Home

History

Upload a photo or image

Upload an image of a room and let our AI generate a new design.

Design this room

Model: Select the Model

Room type: Select the Room Type

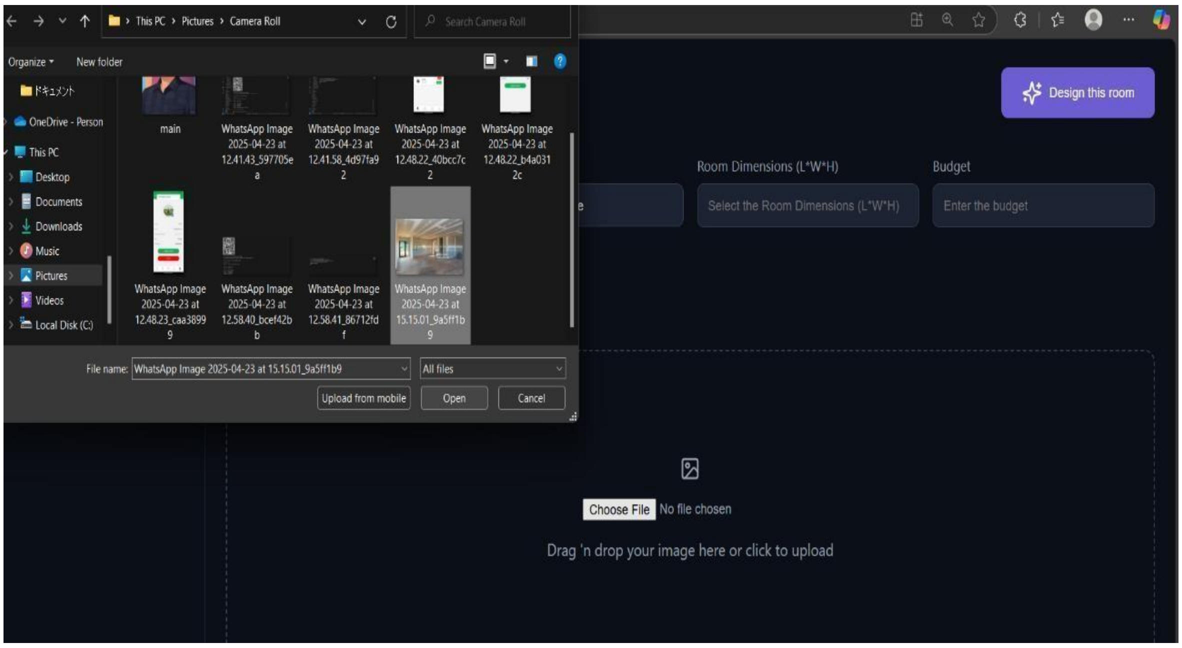
Room Dimensions (L*W*H): Select the Room Dimensions (L*W*H)

Budget: Enter the budget

Color Palette: Enter color palette

Choose File No file chosen

Drag 'n drop your image here or click to upload



This PC › Pictures › Camera Roll

Search Camera Roll

Organize ▾ New folder

ドキュメント

OneDrive - Person

This PC

Desktop

Documents

Downloads

Music

Pictures

Videos

Local Disk (C:)

main

WhatsApp Image 2025-04-23 at 12.41.43_597705ea

WhatsApp Image 2025-04-23 at 12.41.58_4d97fa92

WhatsApp Image 2025-04-23 at 12.48.22_40bcc7c2

WhatsApp Image 2025-04-23 at 12.48.22_b4a0312c

WhatsApp Image 2025-04-23 at 12.48.23_caa38999

WhatsApp Image 2025-04-23 at 12.58.40_bcef42bb

WhatsApp Image 2025-04-23 at 12.58.41_86712fdf

WhatsApp Image 2025-04-23 at 15.15.01_9a5ff1b9

File name: WhatsApp Image 2025-04-23 at 15.15.01_9a5ff1b9

All files

Upload from mobile | Open | Cancel

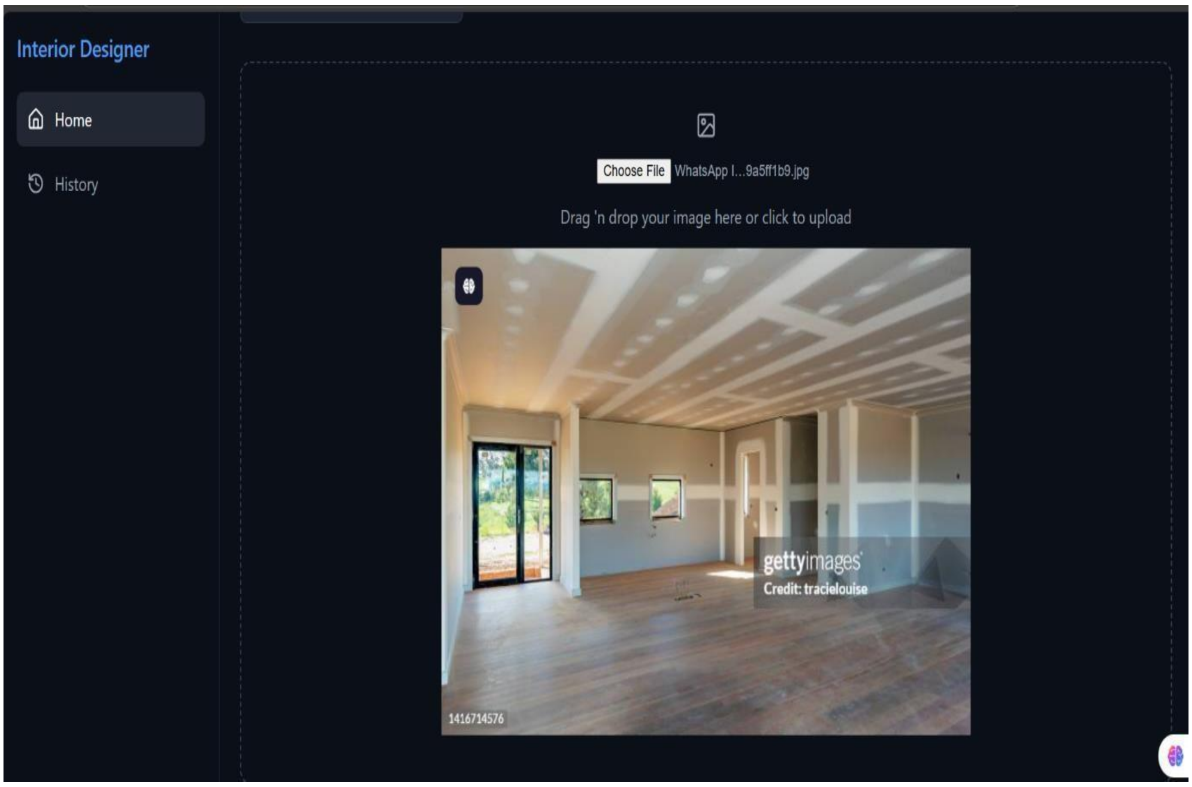
Design this room

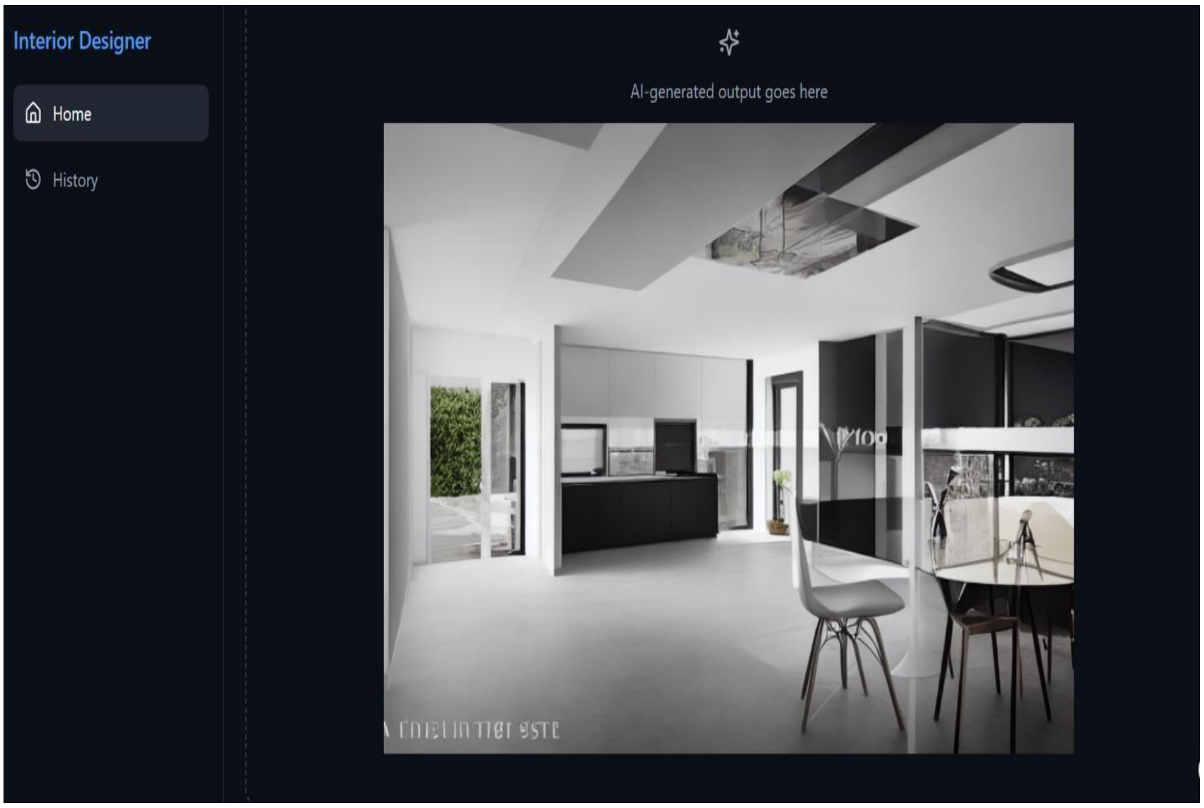
Room Dimensions (L*W*H): Select the Room Dimensions (L*W*H)

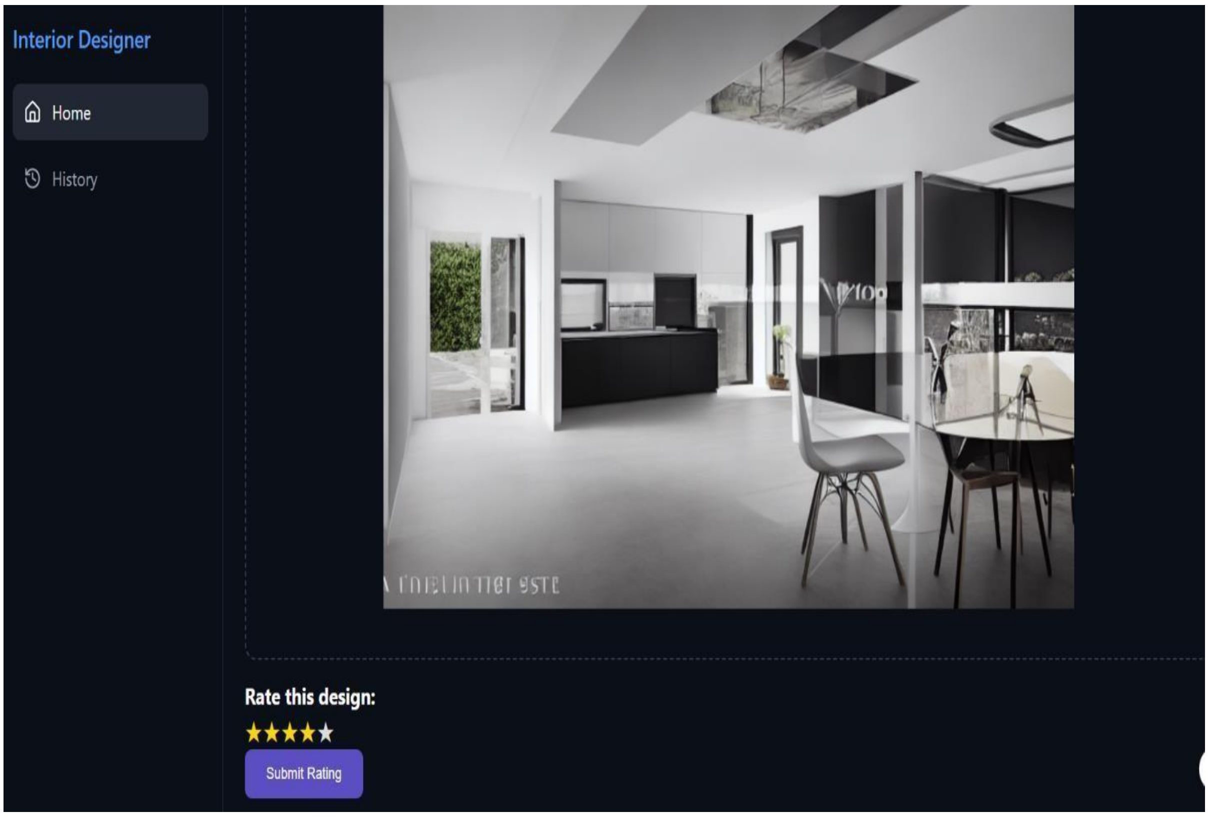
Budget: Enter the budget

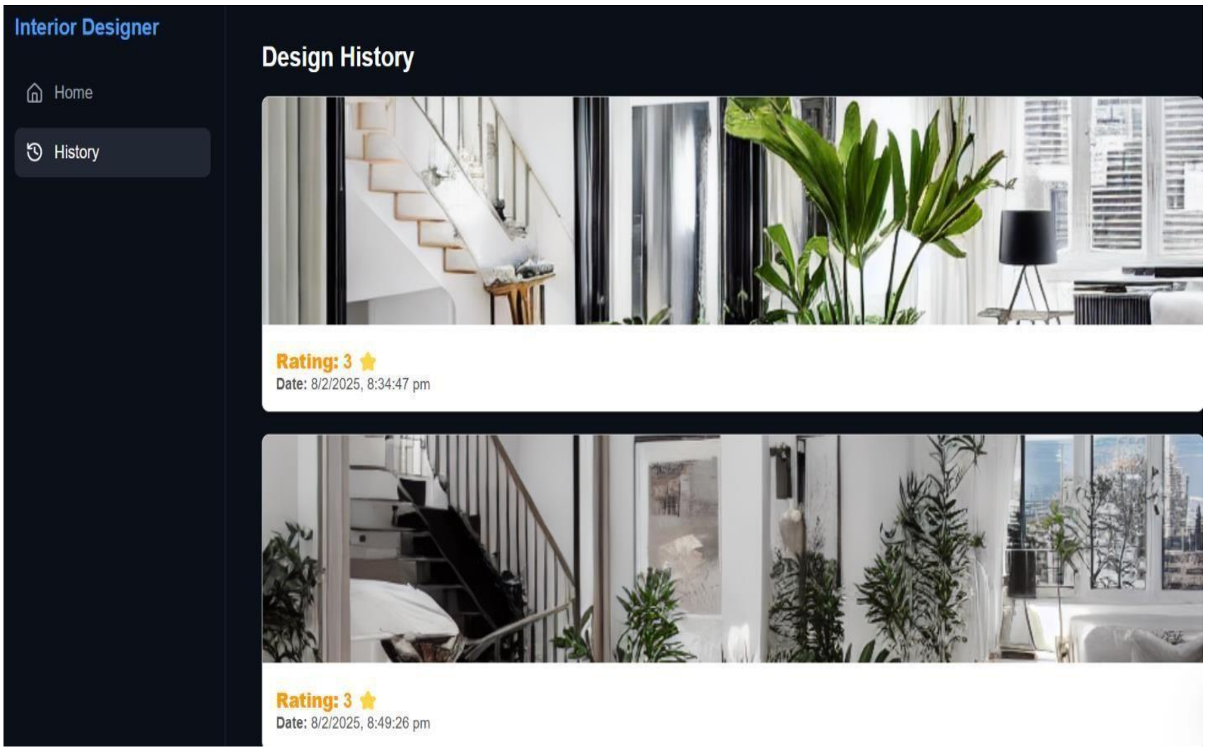
Choose File No file chosen

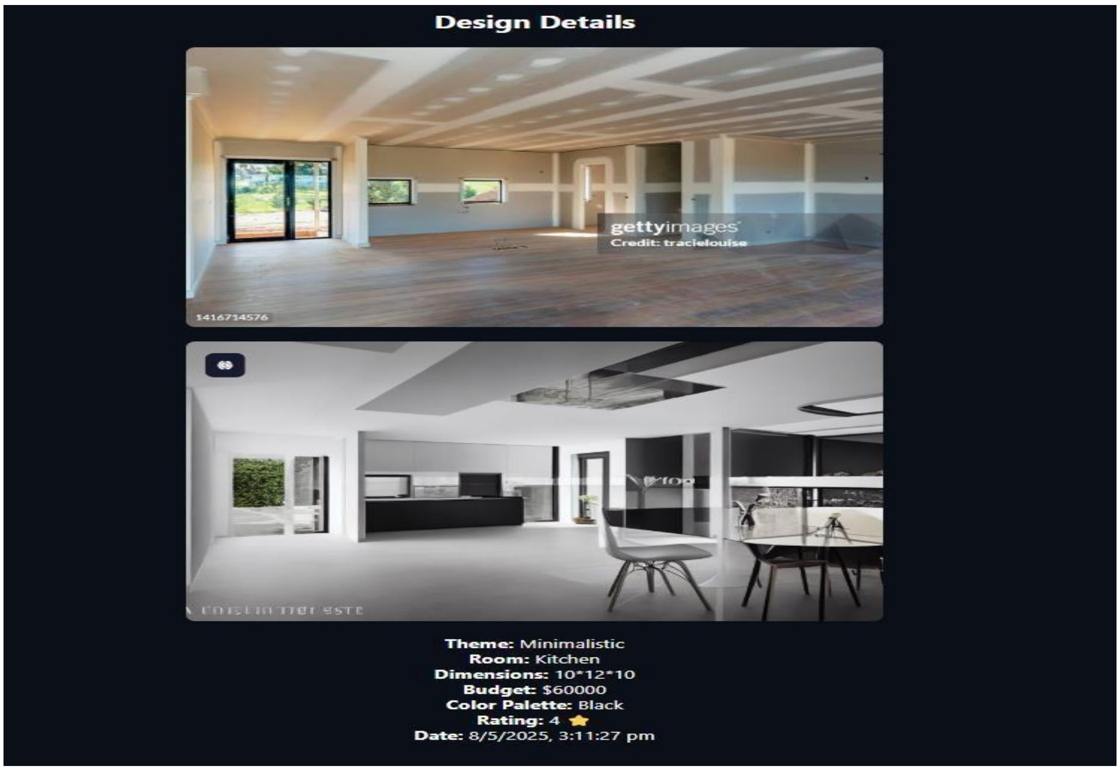
Drag 'n drop your image here or click to upload


Interior Designer
Home
History
Choose File
WhatsApp I...9a5ff1b9.jpg
Drag 'n drop your image here or click to upload
gettyimages
Credit: tracielouise
1416714576


Interior Designer
Home
History
AI-generated output goes here


Interior Designer
Home
History
Rate this design:
★★★★★
Submit Rating


Interior Designer
Home
History
Design History
Rating: 3
Date: 8/2/2025, 8:34:47 pm
Rating: 3
Date: 8/2/2025, 8:49:26 pm



Design Details

Theme: Minimalistic
Room: Kitchen
Dimensions: 10*12*10
Budget: $60000
Color Palette: Black
Rating: 4 ⭐
Date: 8/5/2025, 3:11:27 pm

REFERENCE

1. Akin, O. (1986) Psychology of Architectural Design, Pion Ltd., London.
2. Arora, A. and Aggarwal, S. (2021) ‘Cloudinary Integration for Media Management in Web Applications’, International Journal of Computer Applications, Vol. 178, No. 25, pp. 25-30.
3. Bansal, P., Kaur, A. and Singh, J. (2020) ‘MongoDB: A NoSQL Database for Modern Web Applications’, International Journal of Scientific Research in Computer Science, Engineering and Information Technology, Vol. 6, No. 3, pp. 487–494.
4. Brownlee, J. (2011) Clever Algorithms: Nature-Inspired Programming Recipes, Lulu Press, North Carolina.
5. Choudhary, N., Pandey, R. and Gupta, A. (2018) ‘Performance Comparison of Express and Django for RESTful API Design’, Journal of Web Engineering, Vol. 17, No. 7, pp. 645–658.
6. Delisle, S. and Bouchard, C. (2020) ‘User-centered Design Approach for AI-enabled Interfaces’, International Journal on Interactive Design and Manufacturing, Vol. 14, No. 4, pp. 1127–1138.
7. Goodfellow, I., Bengio, Y. and Courville, A. (2016) Deep Learning, MIT Press, Cambridge, MA.
8. Goldberg, D.E. (1989) Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA.
9. Halvorsrud, R., Kvale, K. and Følstad, A. (2016) ‘Improving Service Quality through Customer Journey Analysis’, Journal of Service Theory and Practice, Vol. 26, No. 6, pp. 840–867.

10. Jain, A.K., Murty, M.N. and Flynn, P.J. (1999) ‘Data Clustering: A Review’, ACM Computing Surveys, Vol. 31, No. 3, pp. 264–323.
11. Johnson, J. (2013) Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Rules, 2nd ed., Morgan Kaufmann, Burlington.
12. Kruchten, P. (2004) The Rational Unified Process: An Introduction, Addison-Wesley, Boston.
13. Kumar, S. and Sharma, R. (2022) ‘Leveraging AI in Interior Design: A Deep Learning Approach’, International Journal of Artificial Intelligence Research, Vol. 16, No. 2, pp. 115–129.
14. Lodato, T. and DiSalvo, C. (2016) ‘Issue-oriented Design’, Design Issues, Vol. 32, No. 1, pp. 3–13.
15. Patel, D., Desai, M. and Patel, N. (2021) ‘React.js Based Frontend Framework for Real-Time Applications’, Journal of Software Engineering and Applications, Vol. 14, No. 5, pp. 215–225.
16. Russell, S. and Norvig, P. (2020) Artificial Intelligence: A Modern Approach, 4th ed., Pearson, London.
17. Sharma, P. and Bhalla, V. (2021) ‘Evaluation of Node.js for Scalable and Event-Driven Web Applications’, International Journal of Computer Applications, Vol. 183, No. 2, pp. 19–24.
18. Szeliski, R. (2010) Computer Vision: Algorithms and Applications, Springer, London.
19. Zhuang, Y. et al. (2020) ‘AI Meets Design: Artificial Intelligence in Architecture and Interior Design’, Frontiers of Architectural Research, Vol. 9, No. 2, pp. 371–382.
20. Zomorodian, A. (2015) Topology for Computing, Cambridge University Press, New York.