# AI POWERED REAL TIME ROOM PHOTO INTERIOR DESIGN GENERATOR

***A Project Report submitted in partial fulfillment of the requirements for the award of the degree of***

**BACHELOR OF TECHNOLOGY**

***In***

**COMPUTER SCIENCE & ENGINEERING**

***By***


21B01A0540 – Dayana Devisree

21B01A0564 - Jujjuru Rama Mydhili

21B01A0553 - Gollapalli Hrishitha

21B01A0563 - Jammalapudi Amsika

21B01A0528 - Boppana Monika


***Under the esteemed guidance of***
***G. Ramesh Babu*** **M. Tech (Ph. D)**
**(Assistant Professor)**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**
**SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN(A)**
**(Approved by AICTE, Accredited by NBA & NAAC, Affiliated to JNTU Kakinada)**
**BHIMAVARAM – 534 202**

**2024 – 2025**

# SHRI VISHNU ENGINEERING COLLEGE FOR WOMEN(A)

**(Approved by AICTE, Accredited by NBA & NAAC, Affiliated to JNTU Kakinada)**
**BHIMAVARAM – 534 202**

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING


## CERTIFICATE

*This is to certify that the project entitled "**AI-Powered-Real-Time-Room-Photo-Interior-Design-Generator",** is being submitted by **D. Devisree, J. Rama Mydhili, G. Hrishitha, J. Amsika, B. Monika** bearing the **Regd. No's. 21B01A0540, 21B01A0564, 21B01A0553, 21B01A0563, 21B05A0528** respectively in partial fulfillment of the requirements for the award of the degree of "**Bachelor of Technology** in **Computer Science & Engineering**" is a record of bonafide work carried out by them under my guidance and supervision during the academic year 2024– 2025 and it has been found worthy of acceptance according to the requirements of the university.*

**Internal Guide** **Head of the Department**

**External Examiner**

# ACKNOWLEDGMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant encouragement and guidance has been a source of inspiration throughout the course of this seminar. We take this opportunity to express our gratitude to all those who have helped us in this seminar.

We wish to place our deep sense of gratitude to **Sri. K. V. Vishnu Raju**, Chairman of SVES, for his constant support on each and every progressive work of mine.

We wish to express our sincere thanks to **Dr. P. Srinivasa Raju, Director Students Affairs & Admin for SVES Group of Institutions,** for being a source of inspiration and constant encouragement.

We wish to express our sincere thanks to **Dr. G. Srinivasa Rao**, Principal of SVECW for being a source of inspiration and constant encouragement.

We wish to express our sincere thanks to **Prof P. Venkata Rama Raju,** Vice-Principal of SVECW for being a source of inspirational and constant encouragement.

We wish to place our deep sense of gratitude to **Dr. P. Kiran Sree**, Head of the Department of Computer Science & Engineering for his valuable pieces of advice in completing this seminar successfully.

We are deeply thankful to our project coordinator **G. Ramesh Babu, Assistant Professor**, for his indispensable guidance and unwavering support throughout our project's completion. His expertise and dedication have been invaluable to our success.

We are deeply indebted and sincere thanks to our PRC members PRC members **Dr. P. R. Sudha Rani, Dr. V. V. R. Maheswara Rao, Dr. J. Veeraraghavan, Mr. G. V. S. S. Prasad Raju** for their valuable advice in completing this project successfully.

Our deep sense of gratitude and sincere thanks to **Dr. P. R. Sudha Rani**, Professor for her unflinching devotion and valuable suggestions throughout our project work.

**Project Team:**

1. 21B01A0540-Dayana Devi Sree
2. 21B01A0564-Jujjuru Rama Mydhili
3. 21B01A0553-Gollapalli Hrishitha
4. 21B01A0563-Jammalapudi Amsika
5. 21B05A0528-Boppana Monika

# ABSTRACT

In the era of digital transformation and smart living, interior design is increasingly influenced by artificial intelligence and personalization. This project presents a comprehensive AI- powered interior design recommendation system that assists users in generating customized room layouts and furniture arrangements based on their style preferences and room types. By integrating a user-friendly interface with a powerful backend, the system allows individuals to upload room images, choose desired themes (Minimal, Modern, Classic), and receive AI- generated designs that cater to their aesthetic needs.

The core engine of the platform utilizes the Replicate API model (Jschoormans/comfyui- interior-remodel) to process visual and textual inputs, generating photorealistic design suggestions. A dynamic quiz captures user intent, and a 5-star rating system gathers feedback for continuous improvement. Additionally, a product search API is implemented to extract and display real-world product links that match the items in the generated design, enhancing the practicality of the system. MongoDB is employed to store user history, which includes design data, ratings, and timestamps, ensuring a personalized and persistent experience.

Experimental implementation confirms that the platform delivers relevant and visually appealing design outcomes while enhancing user satisfaction through seamless product discovery. By bridging AI-generated creativity with e-commerce functionality, this project contributes to the development of intelligent home styling platforms and opens new possibilities in retail integration, spatial aesthetics, and real-time personalization in the domain of smart interior design.

# Table of Contents

# Table of Figures

# 1. INTRODUCTION

The world of interior design is evolving rapidly with the advancement of technology, and artificial intelligence is now playing a significant role in enhancing personalization and user experience. Traditionally, interior design required professional guidance, detailed planning, and often high costs, making it less accessible to everyday users. This project aims to democratize interior design by providing an intelligent, web-based solution that allows users to visualize personalized room makeovers effortlessly using a simple, interactive interface and AI-generated outputs.

Our platform enables users to upload a photo of their room and take a short quiz selecting their preferred interior design style—Minimal, Modern, or Classic—and their room type—Bedroom, Living Room, or Kitchen. These inputs are then processed and sent to a pretrained interior design generation model hosted on the Replicate API. The model analyzes the inputs and returns a realistic interior redesign of the uploaded room image, tailored to the user's chosen style and room type.

To make the experience more interactive, the platform includes a 5-star rating system where users can rate the generated output based on their satisfaction. Once the user submits their feedback, the application stores all relevant details—such as the user inputs, generated design, and submission date—in a MongoDB database. This enables users to revisit their design history and developers to gather insights for further enhancement of the platform.

Beneath the design output section, a search API is integrated to recommend real-world products that visually resemble the items in the generated image. These product links help users conveniently explore similar decor or furniture pieces from online retailers. By clicking on any suggested item, users are redirected to the respective retailer's website, making it easy to turn design inspiration into actual purchases.

This streamlined system combines AI-generated creativity with e-commerce usability, allowing users to not only visualize their redesigned space but also discover relevant products to bring that vision to life. Unlike traditional design platforms or manual planning, our approach is fast, intuitive, and requires no technical or design expertise from the user. The end-to-end experience—from quiz to design output, rating, and product exploration—is carefully designed to maximize convenience and satisfaction.

By leveraging existing AI tools through APIs and combining them with a thoughtfully designed frontend and backend, this project highlights the potential of integrating design technology with everyday user needs. It stands as a practical solution for users seeking smart, personalized, and shoppable interior design experiences all accessible from a single web application.

**Project Objectives :**

The primary objective of the Interior Designer AI System is to make personalized interior design accessible to everyone, regardless of their design knowledge or technical expertise. By leveraging artificial intelligence and an intuitive web interface, the platform aims to bridge the gap between professional interior design and everyday users. Users can upload an image of their room, choose their style preferences, and instantly receive realistic design suggestions, eliminating the need for expensive consultations or complex design software.

key objective is to provide a seamless end-to-end user experience by integrating all stages of the interior design process—style selection, visualization, feedback, and product discovery—into a single platform. The system enables users not only to view customized design results but also to interact with them through ratings and browsing recommended furniture products. This all-in-one approach simplifies the journey from concept to implementation, making it easier for users to bring their dream spaces to life.

The project aims to collect and utilize user feedback for continuous improvement. With the built-in 5-star rating system and design history tracking, the platform can learn from user preferences and adjust future design suggestions accordingly. This feedback loop enhances the personalization aspect of the system, helping it deliver more accurate and user-specific recommendations over time.

The project seeks to integrate real-world shopping experiences through AI-generated product recommendations. The system intelligently identifies items in the generated designs and maps them to similar products available online. This functionality transforms the design output from a static visual suggestion into an interactive, shoppable experience, empowering users to easily purchase decor or furniture that matches their redesigned space.

The project emphasizes scalability, responsiveness, and cross-platform compatibility. Whether accessed from a desktop, tablet, or mobile device, the platform is designed to perform efficiently and maintain a consistent user interface. With technologies like React.js, Node.js, and MongoDB working behind the scenes, the system can handle high user traffic, maintain data integrity, and support future feature expansions—ensuring long-term reliability and adaptability as user needs grow.

# 2. SYSTEM ANALYSIS

System analysis plays a foundational role in the software development lifecycle, laying the groundwork for identifying limitations in current methodologies and defining a structured pathway toward a new, optimized system. In this section, we evaluate the limitations of traditional interior design solutions, introduce the proposed AI-driven web platform, and present an in-depth overview of the system's architecture, functionalities, and feasibility—specific to our project, "AI- Based Interior Design Visualization and Product Discovery Using Replicate API."

## 2.1 Existing System

The conventional interior design process, while effective in certain scenarios, presents several limitations in terms of personalization, automation, and user accessibility. Below is a comprehensive breakdown of the existing system and its associated drawbacks:

### Manual Design Process

- Traditional interior design often involves hiring professional designers for in-person consultations.
- This process is time-consuming, expensive, and requires multiple rounds of communication and revisions.
- Users without access to such services are left to rely on static images from magazines or online platforms for inspiration.

### Limited Personalization

- Most online design tools provide predefined templates or generic 3D room layouts.
- These templates are not tailored to the user's actual room dimensions, images, or personal preferences.
- Users cannot visualize how a selected design style (Minimal, Modern, Classic, etc.) would look in their unique space.

### Disconnected Shopping Experience

- Even when users find appealing designs, there is no direct link to purchasable items resembling the furniture or décor shown.
- Users have to search separately for similar products across multiple e-commerce platforms, which is inefficient and frustrating.
- The disconnect between design inspiration and actionable shopping options weakens the user journey.

### High Dependency on Professional Services

Professional interior design remains a luxury service that's inaccessible to many due to:

- High consultation fees.
- Long project timelines.
- Inability to preview realistic designs before committing.

## 2.2 Proposed System

The proposed system is a web-based platform that enables users to upload an image of their room and select style preferences to receive a fully AI-generated interior design. The model behind this functionality is a pretrained model hosted on Replicate API, which generates styled redesigns of real-world rooms. In addition, the platform features a user feedback system, a searchable product recommendation engine, and data storage via MongoDB to retain user history.

## Key Features:

1. **User Quiz Input:**

- Room Type: Bedroom, Living Room, or Kitchen
- Design Style: Minimal, Modern, or Classic

2. **Image Upload:**

- Users upload a photo of their real room.
- This photo is used as input to the design model.

3. **Design Generation (Replicate API):**

- A request is sent to the Replicate API using user inputs.
- The 'Jschoormans/comfyui-interior-remodel' model returns an AI-generated image.

### 4. Rating & Submission:

- A 5-star rating component allows users to give feedback on the generated output.
- Clicking the Submit button saves:
  - User-selected style and room type
  - Uploaded image and AI-generated image
  - Date and rating
  - All saved to MongoDB for future retrieval via the History Page.

### 5. Search API (Product Discovery):

- Automatically identifies similar-looking furniture or decor items from the generated image.
- Product links are fetched via a search API.
- Users can click these links to be redirected to the appropriate retailer's site (e.g., Amazon, Flipkart).

### 6. History Page:

- Displays a timeline view of previous room uploads and generated outputs along with ratings and product links.

## 2.3 Feasibility Study

This section evaluates the technical, operational, and economic viability of our project: AI-Powered Interior Design Generator with Product Discovery.

### a) Technical Feasibility

- **Frontend (React.js):**

  Delivers an interactive and responsive UI for quizzes, image uploads, and display design.

- **Backend (Node.js + Express):**

  Handles user requests, API integration, and database communication efficiently.

- **Database (MongoDB Atlas):**

  Stores user inputs, generated design history, and ratings in a cloud-hosted NoSQL Database.

- **External Services:**
  - **Replicate API**: Generates designs using pretrained AI models.
  - **Search API**: Displays purchasable product links from the generated image.
  - **Cloudinary**: Manages image uploads and fast content delivery.

### b) Operational Feasibility

Designed for users without technical or design expertise:

- User-Friendly Interface: Simple workflow from quiz to image generation and product discovery.
- Automated System: Requires minimal input; all design generation and product recommendations are handled by the backend.
- Accessibility: Makes professional design experience accessible to everyday users in seconds.

### c) Economic Feasibility

The project is cost-efficient for small-scale deployment and holds strong monetization potential:

- Low Infrastructure Costs:
  - Free tiers of Replicate, MongoDB, and Cloudinary cover basic usage.
  - Affordable deployment on platforms like Vercel or Render.
- Revenue Opportunities:
  - Monetization through affiliate links for suggested products.
  - Display product links.

# 3. SYSTEM REQUIREMENTS

## 3.1 Software Requirements

**Programming Languages:**

- JavaScript (Primary language used for both frontend and backend)
- HTML/CSS (For UI structure and styling)

**Frameworks & Libraries:**

- **Frontend**: React.js (for dynamic UI and state management)
- **Backend**: Node.js with Express.js (for routing, API communication, and server-side logic)
- **Database**: MongoDB Atlas (cloud-hosted NoSQL database)
- **Image Handling**: Cloudinary SDK (for image upload and retrieval)
- **Utility Libraries**: Axios (HTTP requests), Mongoose (MongoDB object modeling), Fuzzy String Matching (for product similarity)

### External APIs:

- **Replicate API** – To generate interior design images using pretrained AI models.
- **Product Search API** – For displaying purchase links to real items matching the design.
- **Cloudinary API** – For efficient media storage and delivery.

## 3.2 Hardware Requirements

**Minimum Configuration (for development):**

- Processor: Intel i5 or equivalent multi-core processor
- Memory: 8 GB RAM (16 GB recommended for smoother performance)
- Storage: SSD with at least 256 GB (for handling image data and project files)

## 3.3 Functional Requirements

### 1. Quiz-Based Style & Room Selection

- Users select room type (e.g., bedroom, kitchen, living room) and preferred style (minimalist, modern, classic).
- Optional dropdown for selecting furniture items specific to the room type.

### 2. Image Upload

- Users upload a photo of their room.
- The image is processed and sent to the Replicate API for interior design generation.

### 3. Design Generation

- System integrates with the Replicate API to generate styled interior designs based on user inputs.
- Generated images are displayed to users along with a rating system.

### 4. History and Ratings

- Users can view previously generated designs in a history page.
- Allows submission of star ratings and feedback for each design.

### 5. Product Discovery Integration

- Below each generated image, a product search API shows visually similar purchasable items.
- Clicking a product redirects users to the corresponding e-commerce site.

### 6. Cloud-Based Storage

- Uploaded room images and generated designs are stored on Cloudinary.
- MongoDB stores user quiz inputs, image URLs, generation timestamps, and ratings.

### 7. Budget-Aware Suggestions (Planned)

- Budget entered during quiz will be used to filter product suggestions within the specified price range.

# 4. SYSTEM DESIGN

System design is a critical phase in the software development lifecycle where the conceptualization from earlier stages begins to take concrete form. It involves the creation of a detailed blueprint that outlines the architecture, components, modules, and interactions within the system. This phase not only guides the development team but also serves as a communication tool, ensuring a shared understanding of the system's structure and functionality among stakeholders.

## 4.1 Introduction to Software Design

Software design is one of the most critical phases in the software development life cycle (SDLC), serving as the blueprint that shapes the foundation of the entire system. It involves translating user needs and functional requirements into structured technical components, establishing how the system will be built, interact, and evolve. A robust design not only enhances clarity for the development team but also ensures scalability, maintainability, performance, and user Satisfaction.

In the context of our project, "Interior Designer AI Web Application" software design plays a central role in integrating various technologies to deliver an intelligent, user-friendly platform. The goal is to empower users to generate AI-based room designs from their uploaded images and receive product recommendations—without requiring technical expertise or professional design help.

The software design of this system takes into account the following key priorities:

- Modularity: Ensures each component (UI, backend logic, external APIs, database, etc.) functions independent
- Real-time interaction: Enables instant generation and display of design outputs, product links, and ratings to maintain a smooth user experience.
- Scalability: Accommodates increasing user traffic, image requests, and design histories without performance degradation.
- Cloud-first architecture: Utilize cloud services like MongoDB Atlas and Cloudinary for reliable data management and image delivery.
- API-Centric communication: Promotes seamless interaction between frontend, backend, and third-party services such as Replicate and product search APIs.
- Maintainability: Designed using standard frameworks and practices that make it easy to update components (e.g., swap APIs or upgrade UI modules).

This system is not just a visual generator—it's an intelligent design assistant. Users begin by taking a short quiz that personalizes the experience, then upload a photo of their space. Behind the scenes, the software coordinates image processing using a pretrained AI model via the Replicate API, stores and manages generated results, and even recommends furniture and decor items through product APIs. This comprehensive flow is backed by thoughtful architectural choices that ensure each feature works reliably and intuitively.

Software design is a foundational step in the software development lifecycle that transitions conceptual requirements into a structured technical blueprint. It defines how the various components of the application—both visible to users and behind the scenes—will interact, function, and scale. This phase ensures that the system is not only functional and reliable but also maintainable and extensible for future enhancements.

In our project, the Interior Designer AI Web Application, the software design plays a crucial role in shaping a seamless integration of interactive frontend features, intelligent backend processes, and powerful third-party AI services. The system is expected to accept user inputs via a quiz and image upload, utilize AI to generate customized room designs, retrieve relevant product links, and allow users to save, rate, and revisit their design history.

To support this user-centric journey, the design must accommodate real-time data handling, secure API communication, efficient media storage, and intuitive UI feedback mechanisms. Special attention is given to ensuring modularity—each subsystem (e.g., image generation, product discovery, database storage) operates independently while contributing to a cohesive, responsive user experience.

This design not only addresses the current functionality but also lays the groundwork for future scalability, such as introducing user accounts, analytics, or monetization strategies. It balances performance, usability, and maintainability—key attributes in delivering a successful AI-powered digital product.

## 4.2 Software Design in Our Project:

The software design of the "Interior Designer AI Web Application" is centered around modularity, scalability, and user-friendliness. The application enables users to upload images of their rooms, take a design preference quiz, and receive AI-generated interior design outputs along with curated product suggestions. The system architecture includes several integrated components such as frontend UI, backend logic, database, external APIs (e.g., Replicate, Cloudinary), and image and product management pipelines.

### a) Overall System Architecture

The architecture follows a client-server model and is composed of the following layers:

- **Frontend Layer**:

  Developed using React.js and TypeScript, this layer handles user interactions, file uploads, quiz rendering, and the display of design results and product suggestions. It communicates with the backend via RESTful APIs.

- **Backend Layer:**

  Built using Node.js and Express.js, the backend processes user inputs, manages API calls to Replicate and search endpoints, stores image metadata and history in MongoDB, and handles image uploads via Cloudinary.

- **Database Layer**:

  MongoDB Atlas is used as a cloud-based document database for storing:

  - User design history
  - Quiz responses and preferences
  - Links to uploaded and generated images
  - Product recommendation metadata

- **External APIs:**

  Replicate API: Invokes pretrained AI models that generate room design outputs based on uploaded images and style inputs.

  Cloudinary API: Handles secure image uploads, transformations, and delivery.

  Product Search API (if integrated): Fetches relevant decor/furniture items based on design themes or colors detected in AI outputs.

b) **Component-wise Breakdown**

- **User Interface Design (React.js)**
  - Home Page: Introduction to the tool and navigation options.
  - Quiz Component: Collects user preferences (e.g., style, color, mood).
  - Upload Component: Allows users to upload an image of their room.
  - Results Page: Displays generated design image and associated product suggestions.
  - History Page: (Optional) Allows users to revisit previous generations.
- **Design Generation Engine (Backend)**
  - Receives user inputs and quiz responses.
  - Packages request data and sends it to the Replicate API.
  - Receives the AI-generated design image.
  - Uploads the image to Cloudinary and stores the URL in MongoDB.
  - Optionally analyzes the image for dominant colors or styles to search for matching products.
- **Product Recommendation Engine**
  - Matches the generated design output with relevant decor or furniture using:
  - Keyword tags from the quiz
  - Visual features detected from the design image (e.g., color palette)
  - Uses a product search API or curated database to retrieve item listings.
- **Image and Data Management**
  - All user uploads and generated images are stored in Cloudinary for optimized delivery.
  - Metadata (e.g., timestamps, associated quiz data, API links) is stored in MongoDB.
- **Notifications and Feedback**
  - Future scope includes integrating email notifications or design sharing features.
  - Users may rate designs for feedback collection.

**c) Data Flow Diagram (High-Level)**

- User → React Frontend
- Quiz Answers & Image → Backend (Node.js)
- Replicate API → Generate Design Image
- Upload to Cloudinary

- Store metadata in MongoDB
- Analyze image → Fetch products
- Send all data back to Frontend → Display to User

### d) Key Design Principles Followed

- **Modularity:** Each component (quiz, upload, design generation, product recommendation) operates independently and can be updated or replaced.
- **Separation of Concerns:** Frontend manages user experience, backend handles logic and data, while external APIs process media and generate designs.
- **Scalability:** Designed to integrate additional design styles, models, or APIs without overhauling core structure.
- **Fault Tolerance:** API response errors are handled gracefully, and users are informed in case of failure.

### e) Technologies Used

**Frontend**:

- React.js
- TypeScript
- Tailwind CSS / Styled Components

**Backend:**

- Node.js
- Express.js

**Database:**

- MongoDB Atlas

**Cloud Services:**

- Cloudinary (image storage & transformation)
- Replicate API (AI-generated design images)

With this design, the "Interior Designer AI Web Application" delivers a powerful and intuitive experience, making personalized interior design accessible, intelligent, and enjoyable for all users.

## 4.3 UML Diagrams

Use-oriented techniques are widely used in software requirement analysis and design. Use cases and usage scenarios facilitate system understanding and provide a common language for communication. This paper presents a scenario-based modeling technique and discusses its applications. In this model, scenarios are organized hierarchically and they capture the system functionality at various abstraction levels including scenario groups, scenarios, and sub-scenarios.

Combining scenarios or sub-scenarios can form complex scenarios. Data are also separately identified, organized, and attached to scenarios. This scenario model can be used to cross check with the UML model. It can also direct systematic scenario-based testing including test case generation, test coverage analysis with respect to requirements, and functional regression testing.

An object contains both data and methods that control the data. The data represents the state of the object. A class describes an object and they also form a hierarchy to model the real-world system. The hierarchy is represented as inheritance and the classes can also be associated in different ways as per the requirement. Objects are the real-world entities that exist around us and the basic concepts such as abstraction, encapsulation, inheritance, and polymorphism all can be represented using UML.UML is powerful enough to represent all the concepts that exist in object - oriented analysis and design.

### The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts
- Be independent of programming languages and development processes.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of the OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.

### a) Class diagram:

The Class Diagram for our Interior Design AI Recommendation System outlines the structural blueprint of the application by showcasing key classes, their attributes, and interactions. The central User class maintains the identity and login details of a person using the platform. This class is associated with one or many DesignRequest objects, which encapsulate the design image uploads and preferences submitted by the user for AI processing. Each design request links to a DesignOutput, which includes the generated AI design image, style type, and any associated recommendations.

The DesignRequest class is also connected to the QuizResponse class. This represents responses to style-related questions, which the system uses to personalize design results. These quiz answers are important inputs in determining the final style applied to a room. The DesignOutput class includes an aggregation relationship with the Recommendation class, which generates product suggestions or styling ideas based on the AI's interpretation of the room image and preferences.

At the core of system functionality is the APIHandler class, which handles interactions with external services like the Replicate API (for design generation) and product recommendation APIs. This class includes methods to send user data and images to third-party AI models and retrieve the results. The ImageStorage class is used for saving and managing design images through Cloudinary. It manages both uploaded images and generated images, handling URL references and image transformations.

The relationships shown in the class diagram reflect an object-oriented structure with separation of concerns, making the system modular and scalable. Each class is responsible for a specific domain of functionality—image handling, design generation, and recommendations—enabling maintainability and ease of testing. This structure allows future improvements, such as supporting additional AI models or more personalized recommendation engines, to be integrated smoothly into the existing framework.

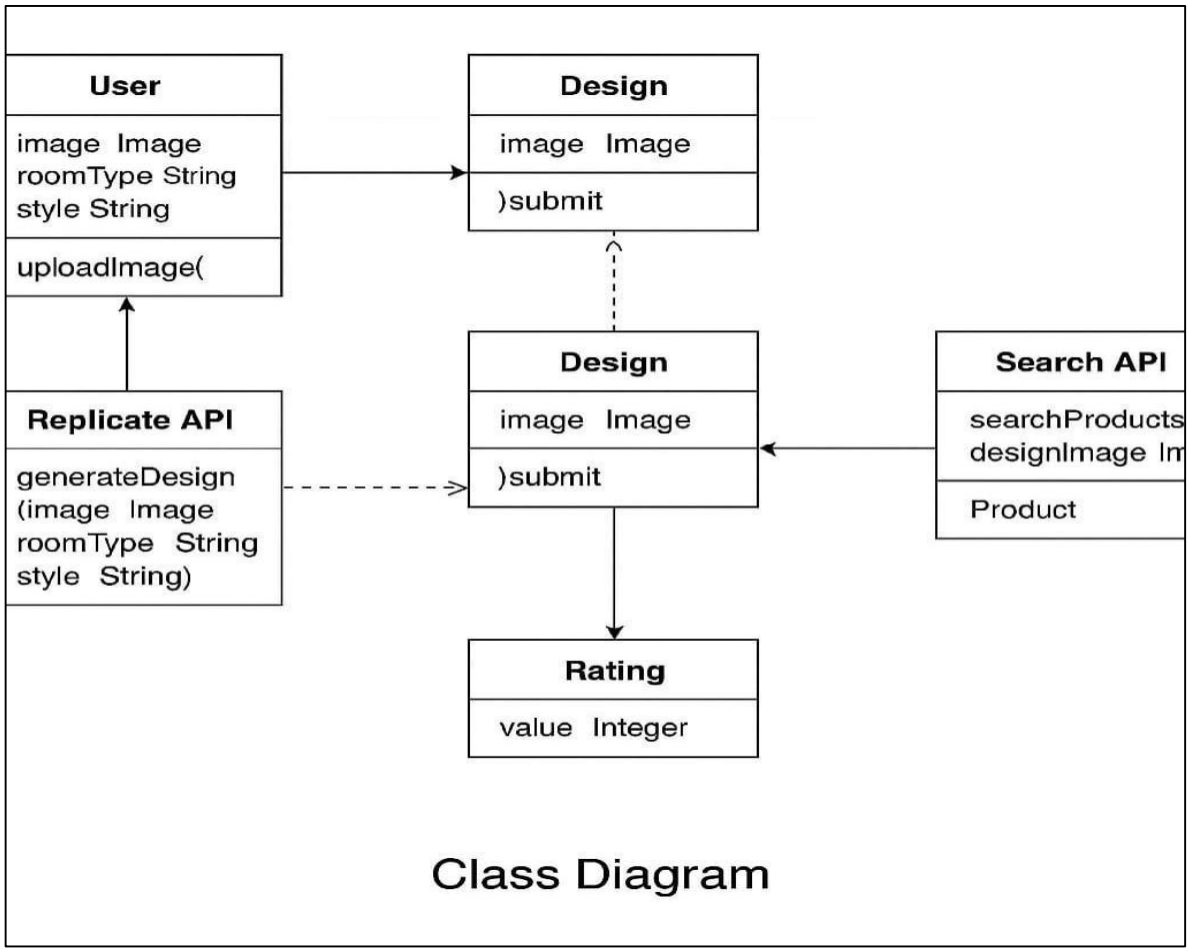



*Fig:4.3.1. Class Diagram*

## b) Object Diagram

Object diagrams are instances of class diagrams that showcase a snapshot of the system at a particular point in time. In this case, the object diagram visually represents actual instances of key objects within the AI-driven Interior Designer system, such as a specific user interacting with the design generator, quiz module, and recommendation engine. It emphasizes how data is passed between modules and how real- time objects interconnect to complete the design workflow.

The object relationships are marked through links between instances. For instance: User is associated with :Design Request via a composition link, highlighting that each request is tied directly to a user. Similarly,: Design Request is linked to :Quiz Response and :Design Output to complete the feedback loop between user input, AI generation, and response handling. These links help developers visualize how data flows through the system in real time.

The diagram further includes : Recommendation and : APIHandler objects. : Recommendation stores AI- inferred product suggestions like "Wooden Nightstand" or "Neutral Curtains", while :APIHandler shows its role in bridging between the AI model (Replicate API) and the frontend. This depiction clarifies how backend operations are tied to user-facing features, ensuring coherence between what users see and the underlying system mechanics

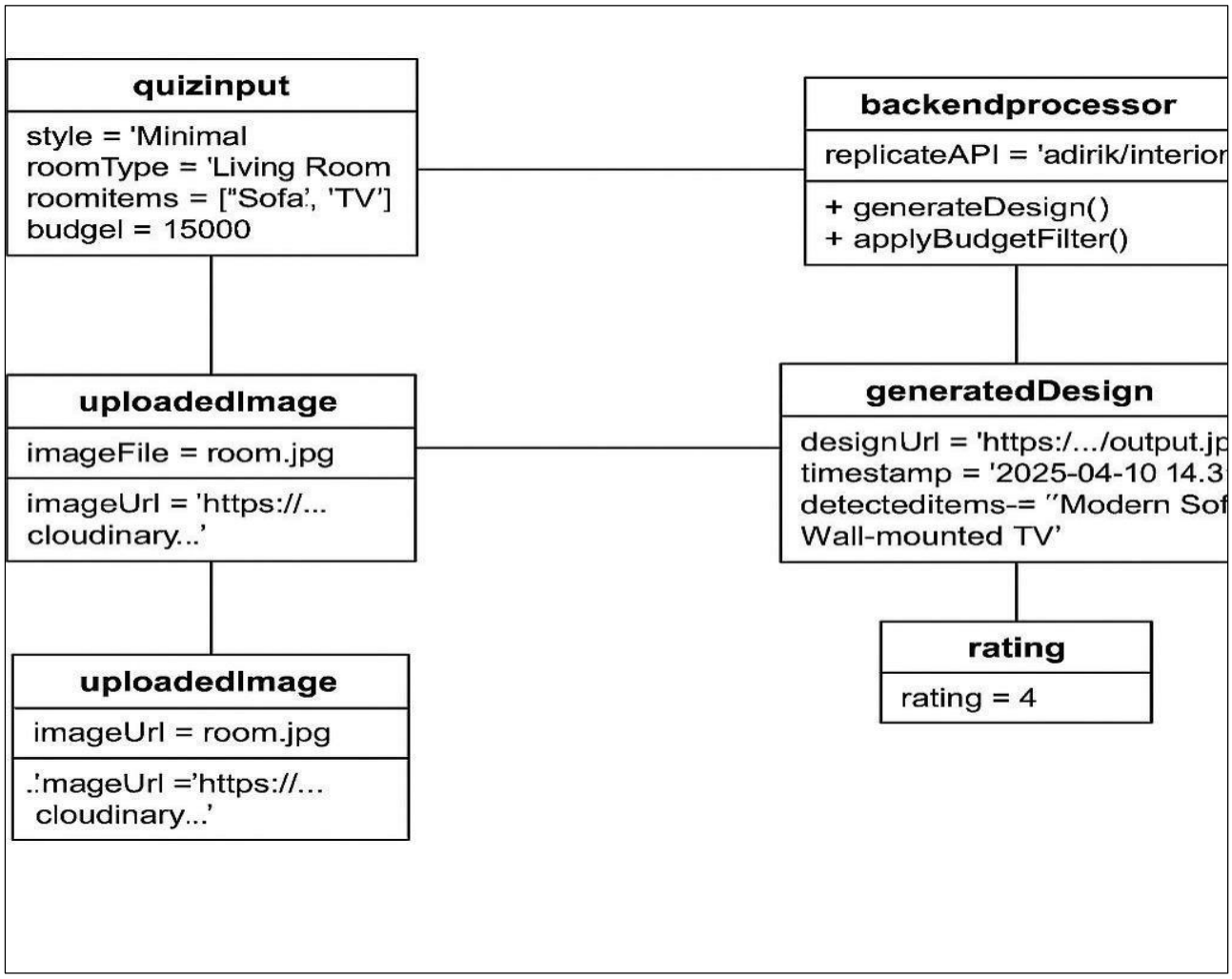



*Fig.4.3.2. Object Diagram*

## c) Component Diagram

The component diagram presents the high-level architecture of the system in terms of modular components and their interactions. It's essential for illustrating how different parts of the system are divided logically and deployed independently.

Key components in this AI interior design system include the Frontend UI (React/Next.js), Backend Server (Node.js), AI Generator (Replicate API), Search API Handler, Cloudinary Image Storage, MongoDB Database, and Quiz Processing Module. Each is encapsulated and connects through defined interfaces.

The frontend component is responsible for collecting user input, including room images and quiz responses. It then communicates with the backend component using RESTful APIs. The backend routes requests to the AI Generator for processing and invokes the Search API for relevant products.

Components such as Cloudinary and MongoDB are integrated with the backend to manage images and user history. The diagram also shows dependencies and communication flows using directional arrows to demonstrate service invocation patterns.

This component view supports scalability and modular development, helping teams work independently on separate services, and making future upgrades easier without disturbing the overall system.

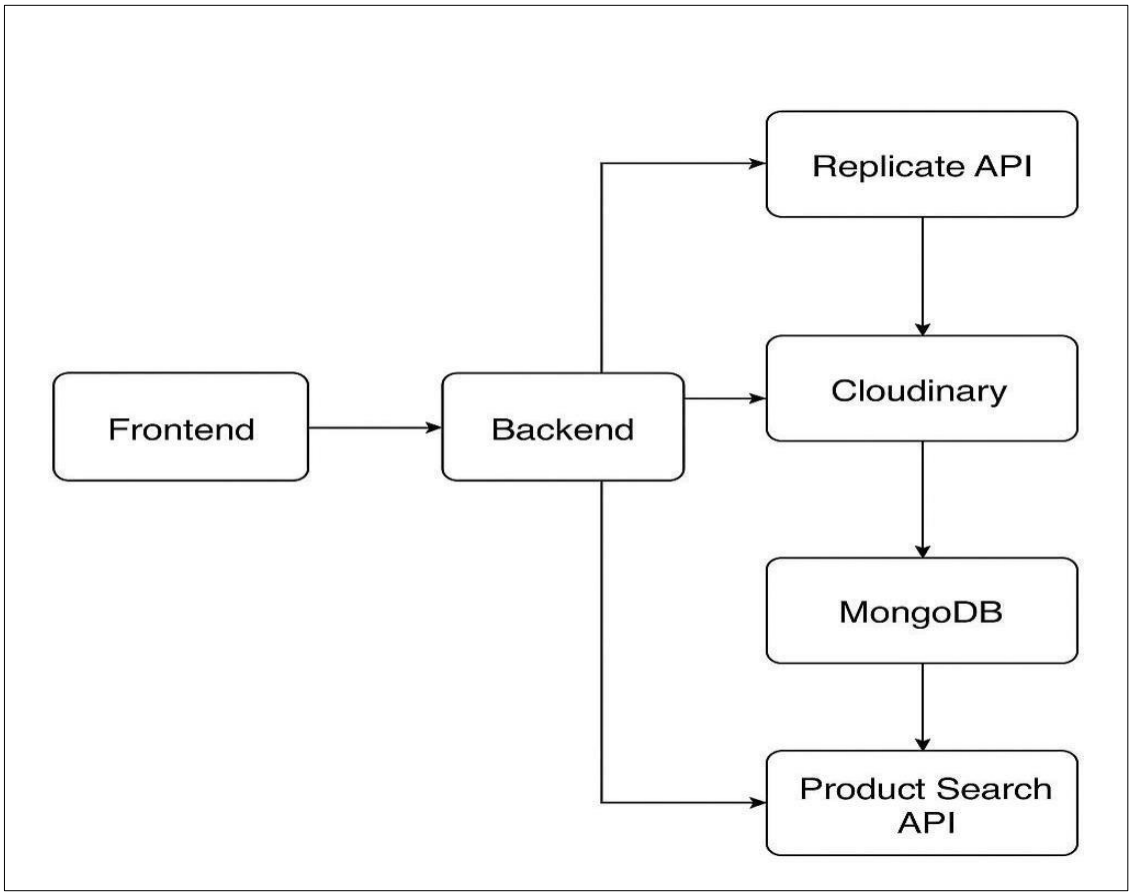



*Fig.4.3.3. Component Diagram*

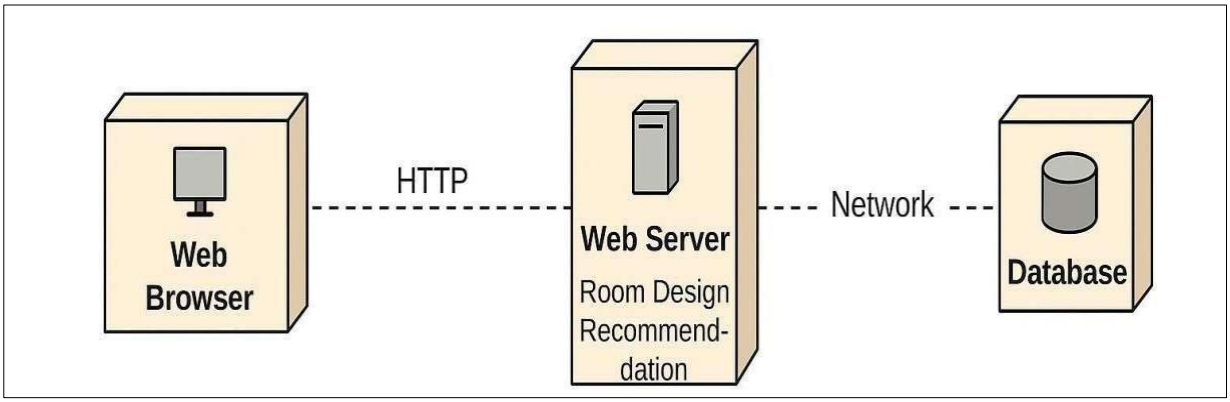
### d) Deployment Diagram

The deployment diagram maps the physical deployment of software artifacts onto hardware nodes. It is useful for understanding where and how different system components will be hosted and interact in a live environment.

In this project, the deployment diagram includes nodes such as User Device (browser), Frontend Server, Backend Server, Cloud Storage Server (Cloudinary), Database Server (MongoDB Atlas), and External Services (Replicate API and Product Search APIs). These nodes are connected through network relationships.

User devices communicate with the frontend via HTTPS. The frontend and backend are deployed on separate cloud environments (e.g., Vercel and Render). Backend communicates with the database for storing user data and with Cloudinary for handling image assets. AI image generation is outsourced to the Replicate API, while product searches are handled via external APIs. These external services are treated as separate deployment nodes interacting over secure internet connections.

This diagram helps identify potential bottlenecks, optimize network calls, and ensure proper deployment strategies for performance and cost-efficiency.




*4.3.4. Deployment Diagram*

### e) Use Case Diagram

The use case diagram identifies the functional requirements of the system from the user's perspective. It depicts the primary actors and their interactions with system functionalities.

Actors include User and Admin. Key use cases for the User include Upload Room Image, Take Style Quiz, Generate Design, View Recommendations, and Save History. The Admin use case includes Monitor Usage and Manage API Tokens.

Relationships between use cases are shown through include and extend relationships. For example, Generate Design includes Analyze Quiz and Generate Image. View Recommendations extends from Generate Design.

This diagram helps stakeholders understand the scope of the system and ensures that all user needs are addressed in the development process.

It also aids in organizing features into modular stories during agile development and can be directly mapped to frontend and backend tasks.

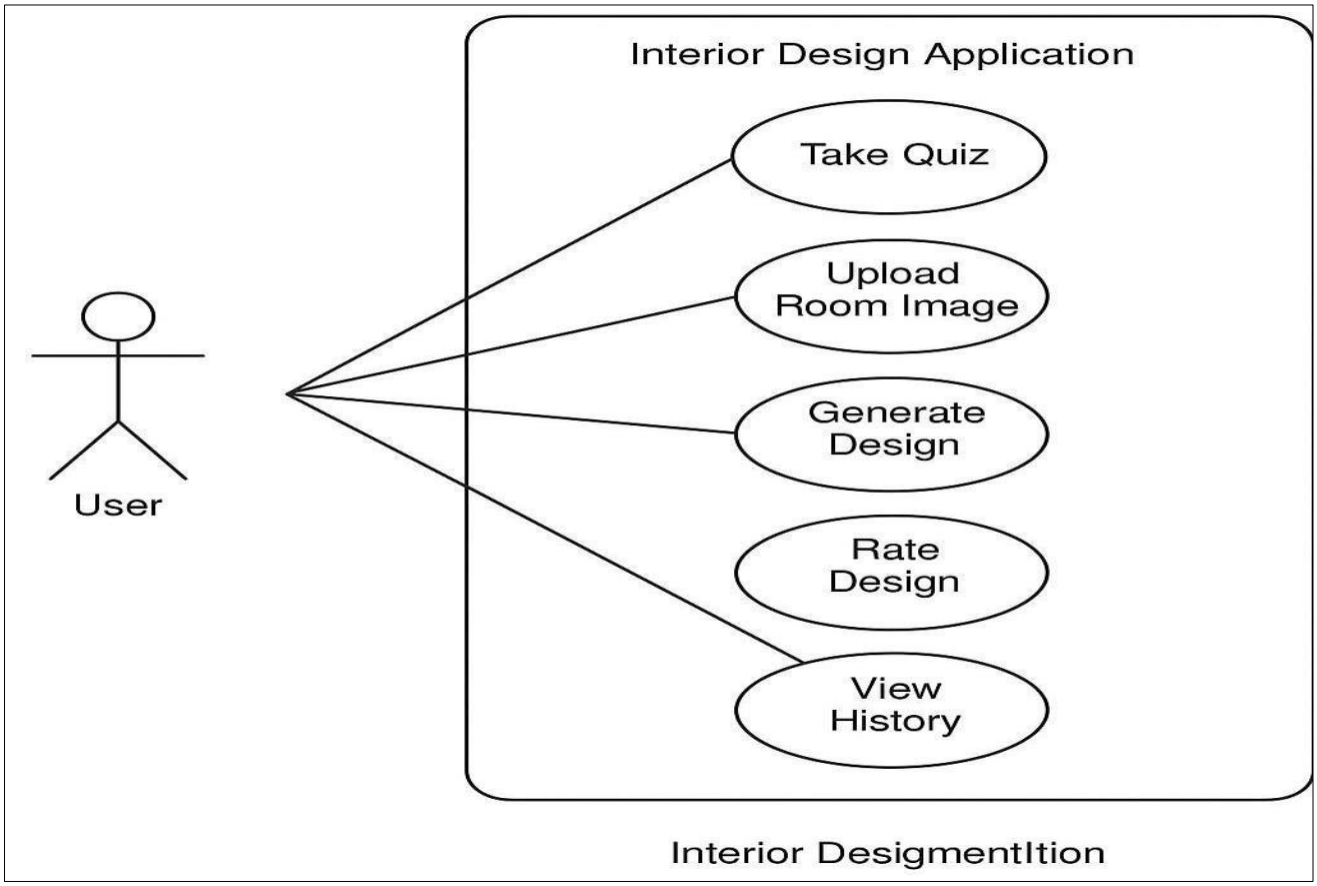



*Fig.4.3.5. Use Case Diagram*

## f) Sequence Diagram

The sequence diagram shows the order of messages exchanged between system components over time during a specific interaction, such as generating a design.

It includes lifelines for User, Frontend, Backend, Replicate API, Product Search API, and Database. The diagram starts with the User submitting an image and quiz preferences on the Frontend.

The Frontend sends this data to the Backend, which invokes the Replicate API to generate the design. After receiving the AI-generated image, the Backend queries the Search API for matching products.

The response is sent back to the Frontend, where the user can view the design and related items. The sequence diagram illustrates time-ordered message calls and returns.

This helps in understanding interaction flows, debugging asynchronous behavior, and optimizing performance bottlenecks.

The collaboration diagram is another interaction diagram that emphasizes object organization rather than the time sequence.

In this system, it shows how User, Frontend, Backend, AI Service, and Product API collaborate to achieve the goal of personalized interior design generation.
Each object is connected by links showing message directions and identifiers like 1:submitImage(), 2:analyzePreferences(), etc., describing the flow of control.

The layout helps to visualize how tightly or loosely objects are coupled and how responsibilities are distributed across modules.
This diagram is useful for reengineering, refactoring, and assessing maintainability of the design.

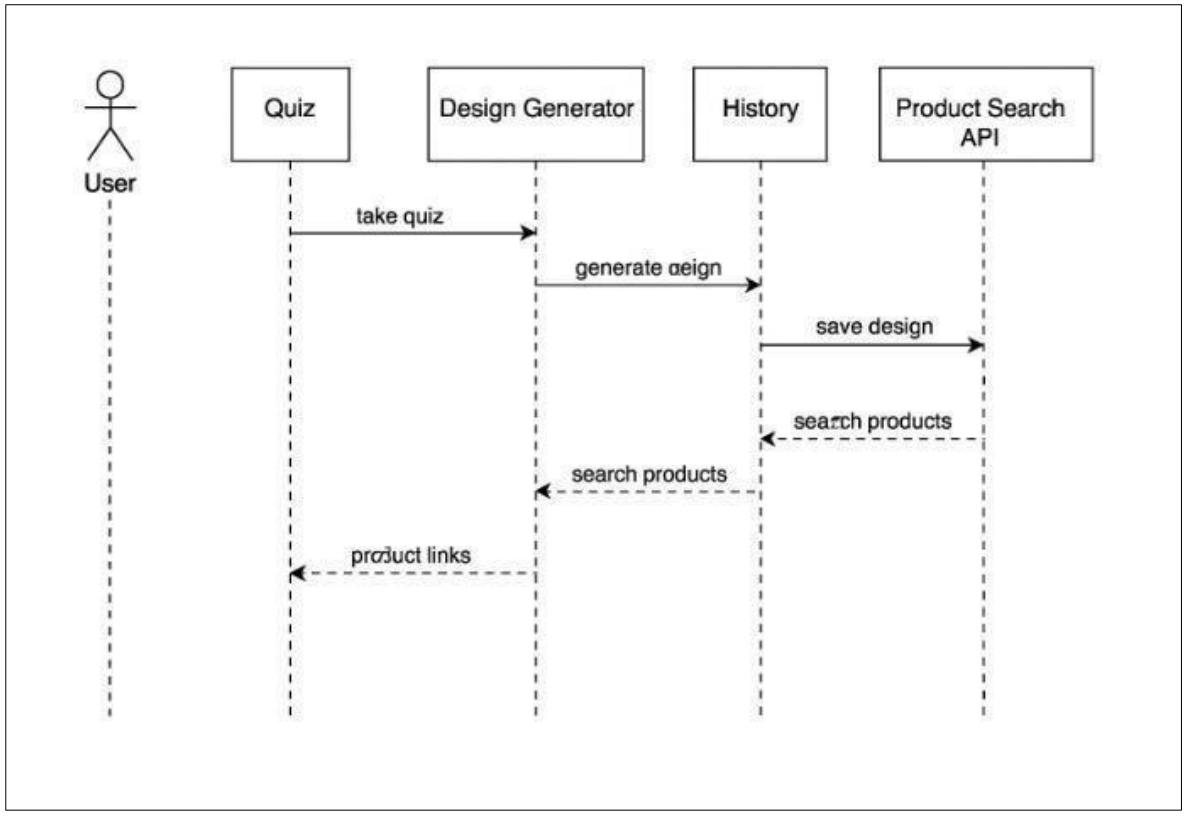



*Fig.4.3.6. Sequence Diagram*

# 5. System Implementation

System implementation is a critical phase in the Software Development Life Cycle (SDLC) where the conceptual design is transformed into a working, deployable system. This phase focuses on developing, integrating, and testing the various components of the AI-powered furniture interior design platform to ensure seamless functionality from frontend to backend.

## 5.1 Frontend Development (React.js)

The frontend of the application is developed using React.js, a powerful JavaScript library for building user interfaces. It enables the creation of a responsive, component-based UI that interacts seamlessly with the backend and external APIs. The main goal is to provide users with an intuitive, aesthetically pleasing, and smooth experience from start to finish.

- **Key Features in Detail:**

**Quiz Interface**

- Purpose: To collect user preferences regarding interior design style and room type.
- Built using controlled components (like dropdown menus and radio buttons).
- Two fields are present:
  - Style Type: Minimal, Modern, Classic
  - Room Type: Bedroom, Living Room, Kitchen
- Form validation ensures both options are selected before proceeding.
- The selected values are stored in the component state and passed to the backend when the user clicks "Generate Design".

*4.3.1. Class Daigram*

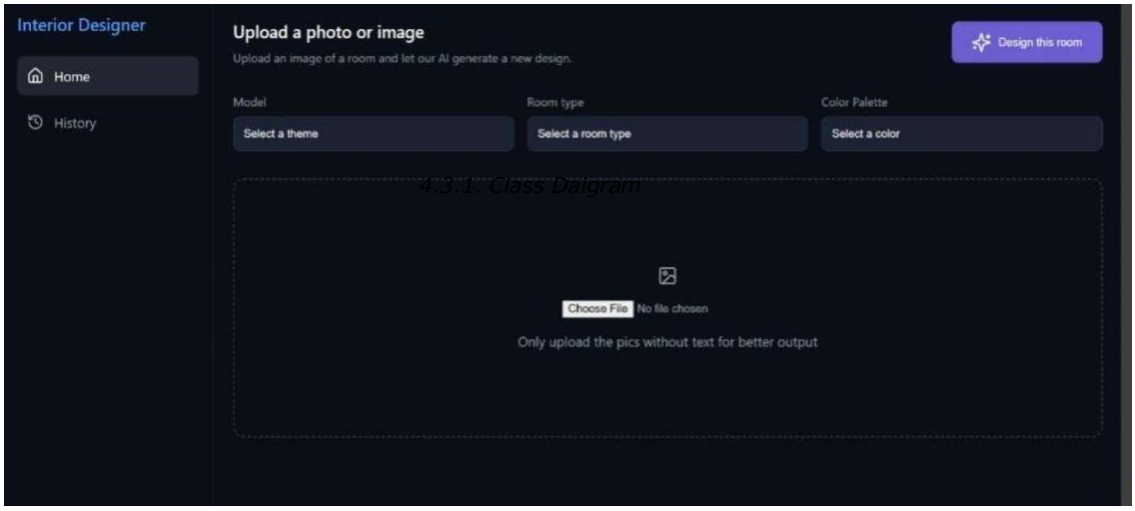



*Fig.5.1.1. Interface Design*

## Image Upload

- Purpose: To allow users to upload an actual photo of their room that needs to be redesigned.
- Implementation:
  - The upload is handled using a file input field.
  - On file selection, the image is previewed on the screen for user confirmation.
- The file is sent to the backend via a POST request where it is uploaded to Cloudinary.
- The resulting Cloudinary URL is stored and passed to the Replicate API for design generation.

## Design Display

- Purpose: To show the AI-generated interior design image.
- Implementation:
  - Once the Replicate API returns the transformed image, it is displayed in the frontend using the returned Cloudinary URL.
  - A loading spinner or progress bar is shown while waiting for the result.
  - The UI dynamically replaces the original image upload form with the output design.


*Fig.5.1.2. Generated Output*

## Rating System

- Purpose: To collect user feedback on the quality and appeal of the AI-generated design.
- Implementation:
    - A 5-star rating system is implemented using clickable icons (typically using a package like react-stars or custom SVG icons).
    - The selected rating is stored in the state and submitted along with the design metadata (style, room type, image URL) to the backend.
    - Ratings are used for analyzing user preferences and improving the model in the future.

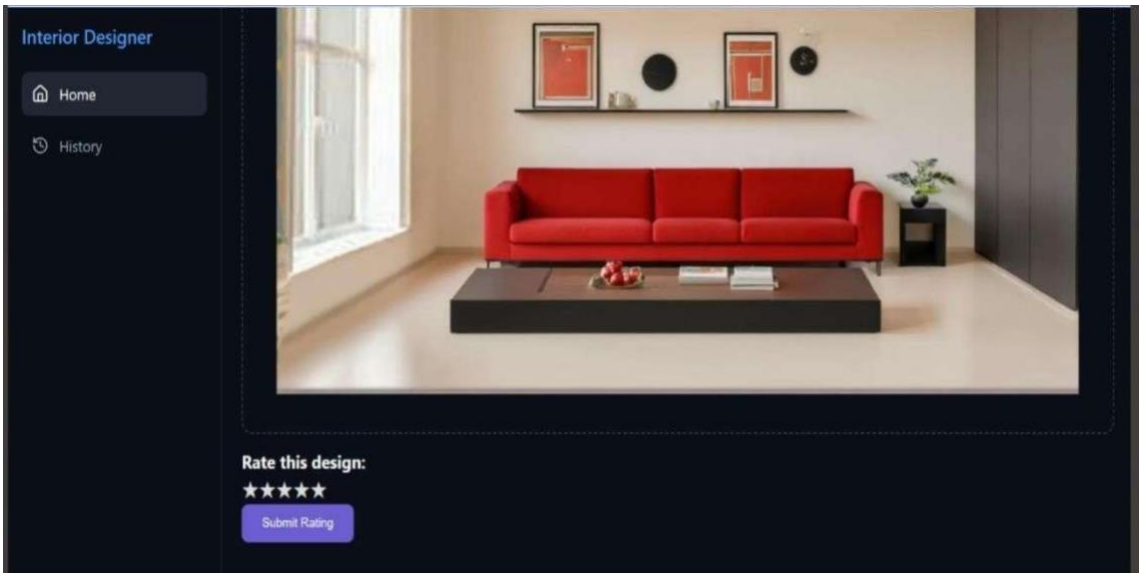



*Fig.5.1.3. Ratings*

## History View

- Purpose: To allow users to view all their previously generated designs along with the ratings and dates.
- Implementation:
    - Data is fetched from the backend using an API that queries MongoDB for entries associated with the user.
    - Each record includes:
        - Design image (Cloudinary link),Style type and room type

- User rating (1 to 5 stars)
- Date of creation

- Entries are displayed in a grid or list view, sorted by most recent.
- This module encourages user re-engagement and serves as a design portfolio.

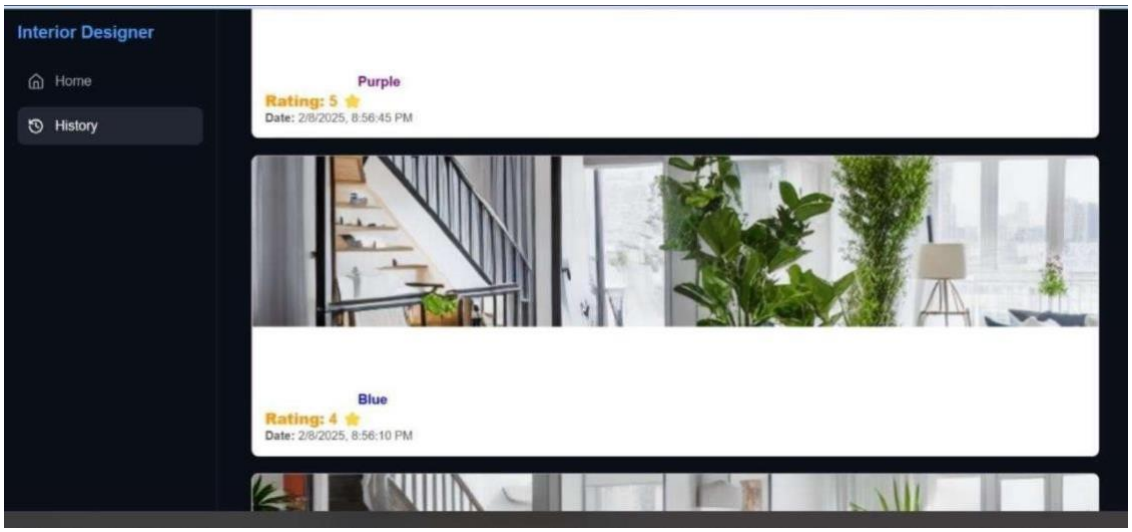



*Fig.5.1.4. History Page*

## 5.2 Backend Development (Node.js & Express.js)

The backend of the AI-powered interior design application is developed using Node.js and Express.js, providing the core logic that connects the frontend interface with cloud services, the AI model, and the database. It ensures smooth data processing, communication with third-party APIs, and persistent storage.

### a) Quiz Submission Handling

When users complete the quiz by selecting a style type (e.g., Minimal, Modern, Classic) and room type (e.g., Bedroom, Living Room, Kitchen), their selections are sent to the backend.

- The backend receives this data and verifies it.
- These preferences are then used as input for generating personalized AI-based interior designs.
- The values are stored along with user session data (if implemented) or paired with the design for history tracking.

### b) Image Upload and Management

- The backend processes the image file and uploads it to Cloudinary, a cloud-based media hosting service.
- Once uploaded, Cloudinary returns a secure image URL.
- This URL is stored in the database and also used as input for the AI model.

### c) AI Design Generation ReplicateAPI

The backend is integrated with Replicate API, which hosts the Jschoormans/comfyui-interior-remodel model used to generate AI designs.

- The backend combines the image URL and user-selected preferences into a single request.
- This request is sent to the Replicate API.
- The API responds with a link to the generated AI design image.
- This output is passed to the frontend and stored in the database for later retrieval.

### d) Rating System and Historu Management

Users can rate the AI-generated design using a 5-star rating system and view a history of their past designs.

- When a user submits a rating, it is sent to the backend.
- The backend stores:
    - The rating,
    - The generated design image URL,
    - The initial quiz inputs (style and room type),
    - The submission timestamp.
- This data is saved in MongoDB, allowing users to:
    - Revisit old designs,
    - Track preferences,
    - Review ratings given.

## 5.3 AI Model Integration (Replicate API)

A core component of the system is the integration with the Replicate API, which provides access to a powerful AI model named Jschoormans/comfyui-interior-remodel. This model enables the application to generate realistic, visually appealing interior design images based on user input. The backend sends required data to the model and retrieves generated outputs seamlessly.

The Jschoormans/comfyui-interior-remodel model specializes in generating photorealistic interior design mockups. It takes an image of a room and stylistic preferences as input and returns an enhanced image that reflects a professionally designed interior space.

The AI model receives the following three types of input:

a) Uploaded Room Image

- Users upload a photo of their existing room (e.g., empty or semi-furnished).
- The backend uploads the image to Cloudinary and retrieves a hosted URL.
- This URL is sent to the Replicate model as the base image for transformation.

b) Selected Room Type

- The user selects a room type from predefined categories (Bedroom, Living Room, Kitchen).
- This choice helps the model understand the functional context of the space, influencing layout and furniture selection.

c) Selected Style Type

- Users choose a style preference such as Minimal, Modern, or Classic.
- The model uses this input to guide its visual design choices, ensuring the output aligns with the chosen aesthetic.

```
app.post("/api/design-room", async (req, res) => {
  const { image, theme, room, colorPalette, dimensions } = req.body;

  const prompt = `A ${room} designed in a ${theme} style with a ${colorPalette} color palette and dimensions of ${dimensions

  const replicateRes = await fetch("https://api.replicate.com/v1/predictions", {
    method: "POST",
    headers: {
      "Authorization": `Token ${process.env.REPLICATE_API_TOKEN}`,
      "Content-Type": "application/json",
    },
    body: JSON.stringify({
      version: "2a360362540e1f6cfe59c9db4aa8aa9059233d40e638aae0cdeb6b41f3d0dcce",
      input: {
        image,
        prompt,
        scale: 7,
        ddim_steps: 20,
        a_prompt: "high quality, detailed, modern interior",
        n_prompt: "low resolution, blurry, distorted",
        seed: Math.floor(Math.random() * 100000),
      },
    }),
```

*Fig.5.3.1. Model Integration*

## 5.4 Database Management (MongoDB):

MongoDB is used as the primary database for storing and managing structured data related to user interactions. It ensures the application can retrieve, update, and maintain user records efficiently.

a) Stored Data Includes:

Quiz Selections:

- Room type (e.g., Bedroom, Kitchen)
- Style type (e.g., Minimal, Modern)
- Uploaded Image & Generated Design Links
- Original room photo (Cloudinary URL)
- AI-generated design (Replicate model output URL)
- User Ratings & History Logs
- Star rating provided by the user
- Timestamp of submission

All data tied together to maintain a clear record for each design session MongoDB enables easy access to past records, enabling personalized user experiences like history tracking and feedback.

```
// MongoDB schema
const ratingSchema = new mongoose.Schema({
  inputImageUrl: String,
  outputImageUrl: String,
  theme: String,
  room: String,
  colorPalette: String,
  rating: Number,
  date: Date,
});

const Rating = mongoose.model("Rating", ratingSchema);

// Connect MongoDB
mongoose.connect(process.env.MONGO_URI)
  .then(() => console.log("MongoDB connected"))
  .catch((err) => console.error("MongoDB connection error:", err));
```

*Fig.5.4.1. MangoDB Connection*

## 5.5 Product Recommendation System:

This module helps bridge the gap between inspiration and action by suggesting real products based on the generated interior design.

- Dynamic Search API

  Based on identified items, the backend performs keyword-based searches using external APIs or databases.
  Results are matched to real, purchasable products from popular e-commerce sites.

- Product Display

  Relevant products are displayed as clickable links directly below the generated design image.
  Clicking a product redirects the user to the vendor's site for purchase or details

```
app.post("/api/search-similar-products", async (req, res) => {
  const { imageUrl } = req.body;

  try {
    console.log("Original Image URL:", imageUrl);

    // Upload to Cloudinary and convert to PNG (corrected format as per original code)
    const uploadRes = await cloudinary.uploader.upload(imageUrl, {
      format: "png",  // This should be 'png' instead of 'jpeg' for a PNG image conversion
    });

    const jpegUrl = uploadRes.secure_url;
    console.log("Cloudinary PNG URL:", jpegUrl);

    // Call SearchAPI with Cloudinary PNG URL
    const url = "https://www.searchapi.io/api/v1/search";
    const params = {
      engine: "google_lens",
      search_type: "products",
      url: jpegUrl,
      country: "in",
```

*Fig.5.5.1. Search API Integration*

# 6. SYSTEM TESTING

## 6.1 Introduction

### Overview of System Testing

System testing is a vital stage in the software development life cycle, focusing on verifying the end-to- end functionality, integration, performance, and security of a complete software system. For the Interior Designer AI System, system testing ensures that all modules—from user interaction on the frontend to AI-driven design generation and product recommendations—work cohesively and reliably.

This application integrates advanced AI, interactive quiz modules, dynamic design generation, and product suggestion features. Testing ensures that the system provides accurate outputs, maintains data consistency, and delivers a seamless experience across devices and user scenarios.

The Interior Designer AI System is a multi-component platform built using React.js (frontend), Node.js & Express.js (backend), MongoDB (database), Cloudinary (image hosting), and Replicate API (AI-based design generation). Therefore, system testing evaluates:

- AI Model Integration: Testing the transformation accuracy of room images into design outputs based on user preferences.
- Quiz & Image Upload Flow: Ensuring correct capture and processing of style and room selections along with image uploads.
- Design Display and Rating System: Verifying the loading, display, and star-based rating of AI- generated designs.
    - Search API & Shopping Links: Checking the retrieval and relevance of product links based on image content.
- History Logging: Ensuring previous designs, timestamps, and feedback are correctly stored and retrieved.

Given the multi-layered architecture, system testing involves validating each module in isolation and in conjunction with other parts to ensure smooth interoperability and consistent user experience.

## Importance of System Testing

System testing is essential to validate the correctness, efficiency, and security of the Interior Designer AI System before deployment. It helps uncover defects and integration issues that may not surface during earlier development stages.

Key benefits of system testing in this project:

- End-to-End Validation: Ensures that all key features—such as quiz responses, AI image transformation, product display, and history logging—function as expected.
- Module Integration Check: Identifies any miscommunication or data mismatch between the frontend, backend, database, and third-party APIs like Replicate and Cloudinary.
- Performance Analysis: Evaluates the responsiveness of the app, especially during AI model execution and image uploads.
- Security Enforcement: Confirms that image data, user history, and feedback are handled securely, especially during upload and retrieval processes.
- Cross-Platform Compatibility: Validates usability and design across various browsers and devices.
- Budget Application Accuracy: Tests whether the budget input (when fully implemented) influences the AI design output and product suggestions.

## Objectives of System Testing

The following are the key objectives of system testing for the Interior Designer AI System:

- **Functional Verification**: Confirm all features—from quiz inputs to AI outputs—operate as expected.
- **Performance Evaluation**: Measure response times, especially for design generation and product link loading.

- **Usability Testing**: Ensure that the interface is user-friendly, consistent, and accessible.
- **Security Validation**: Protect sensitive data such as image uploads and design history from unauthorized access.
- **Compatibility Testing**: Validate system behavior across different devices, browsers, and screen sizes.
- **Budget Feature Validation**: Check if future budget constraints correctly influence design generation and product filtering.

## Scope of System Testing

System testing for the Interior Designer AI System includes the following components:

- **User Interface Testing**: Verifies responsive behavior of the quiz, image upload, and design rating on devices like phones, tablets, and desktops.
- **Backend Functional Testing**: Ensures API endpoints correctly process quiz inputs, image uploads, AI requests, and ratings.
- **Integration Testing:** Confirms seamless communication between the AI model (Replicate API), image storage (Cloudinary), product search, and MongoDB.
- **Security Testing**: Evaluates the encryption and access control mechanisms around sensitive user data and uploads.
- **Load Testing:** Assesses system performance during high traffic situations, such as concurrent image uploads or quiz submissions.
- **Performance Testing:** Measures system latency during image transformation, design retrieval, and search result display.

## 6.2 Testing Methods

### a. Functional Testing

Functional testing ensures that each feature of the Interior Designer AI System works according to its specifications. The goal is to verify that the user experience flows correctly from the quiz and image upload to AI-generated designs, product recommendations, and rating submissions.

**Key Focus Areas:**

- Accurate quiz handling and option storage (style and room type)
- Smooth image upload and storage via Cloudinary
- Correct API calls to Replicate for AI-based design generation
- Proper display of generated images and product suggestions
- Reliable star rating and feedback submission
- Storing design history with metadata

**Test Case Example**:

Uploading a Room Image and Generating a design

**Test Steps:**

- Complete the style and room type quiz.
- Upload a room image.
- Click "Generate Design."
- Observe the output and product suggestions.

**Expected Outcome:**

- Image uploads successfully and is saved to Cloudinary.
- System sends correct API request to Replicate with quiz data and image.
- AI-generated design appears on the page.
- Product suggestions populate below the image based on design elements.
- Rating system works and feedback can be submitted and stored in history.

## b. Performance Testing

Performance testing evaluates how efficiently the system operates under various workloads. It is essential for assessing speed, stability, and resource usage when processing image uploads, generating designs, or retrieving shopping links.

**Key Metrics:**

- Image upload and processing time
- AI design generation latency
- Product suggestion load time
- Frontend rendering speed

**Test Case Example:**

Measuring Image Processing Time

**Test Steps:**

- Upload a 4K-resolution room image.
- Trigger the AI design generation process.

- Use performance tracking tools (e.g., Chrome DevTools or Postman monitors) to measure response time.

**Expected Outcome:**

- Image is uploaded in under 2 seconds.
- AI design generation completes within 3 seconds.
- Entire output (design + suggestions) renders within 5 seconds max.

## c. Usability Testing

Usability testing ensures that users find the platform easy to use, intuitive, and accessible. It includes navigation, readability, responsiveness, and interaction design.

**Test Focus:**

- Smooth flow from quiz to image upload to design generation
- Clear and simple instructions throughout the UI
- Mobile responsiveness and accessibility for different users

**Test Case Example:** First-Time User Journey

**Test Steps:**

- Open the site on a desktop or mobile browser.
- Take the quiz, upload an image, and generate a design.
- Rate the design and view the design history.

**Expected Outcome:**

- No confusion at any step.
- All buttons and forms behave as expected.
- Quiz options and tooltips provide clarity.
- Users can navigate back and forth between steps effortlessly.

## d. Compatibility Testing

Compatibility testing checks that the platform functions properly across different browsers, devices, and screen sizes. This is crucial for a web-based system with visual-heavy features.

**Compatibility Targets:**

- Browsers: Chrome, Firefox, Safari, Microsoft Edge
- Devices: Desktop, Laptop, Tablet, Mobile
- Operating Systems: Windows, macOS, iOS, Android

**Test Case Example:** Cross-Device UI Consistency

**Test Steps:**

- Open the application on different browsers and devices.
- Complete the quiz and upload images on each platform.
- Generate and view designs, and interact with rating system.

**Expected Outcome:**

- Consistent design and functionality across all environments.
- Image preview, generated output, and buttons scale properly on mobile.

- No broken layouts or hidden elements on small screens.

## e. Load Testing

Load testing evaluates how well the system performs when multiple users access and use it simultaneously. It simulates real-world usage patterns to ensure scalability and stability under stress.

**Tools Suggested:**

- Apache JMeter
- Artillery.io
- k6

**Test Case Example**: Simulating Heavy Traffic

**Test Steps:**

- Use a load testing tool to simulate 1000+ users uploading images and generating designs.
- Monitor backend CPU, memory usage, and MongoDB query performance.
- Observe response times and system behavior under load.

**Expected Outcome:**

- Server remains stable without downtime.
- Design generation requests are queued efficiently.
- No image upload or product search failures.
- Performance remains within acceptable thresholds (e.g., <5s for high-load responses).

# 7. CONCLUSION

The AI-Powered Interior Designer System aims to revolutionize the way users approach home design by merging artificial intelligence with modern web development practices. Traditional interior design often requires professional consultation and manual processes that can be time- consuming and expensive. Our solution simplifies and personalizes the design experience, allowing users to receive customized interior design suggestions in seconds by simply uploading a photo of their room.

The system architecture incorporates a combination of powerful technologies including React.js, Next.js, and TypeScript on the frontend, and AI-driven design processing through the Replicate API on the backend. By leveraging modern web development frameworks and AI models, the application delivers an intuitive user interface and intelligent backend logic capable of generating visually compelling and tailored room designs.

One of the key strengths of the project lies in its personalization engine. Users are guided through a simple quiz to identify their style preferences and functional needs. These preferences, along with the uploaded room image, inform the AI generation process to ensure the suggested designs reflect the user's individual taste. The system's dynamic flow between quiz input, design generation, and output presentation is carefully crafted for seamless interaction.

The integration of external tools like Cloudinary ensures that images are stored, retrieved, and displayed efficiently, preserving quality and speed. This cloud-based image management contributes to a smooth and responsive user experience, essential for applications dealing with visual content. Additionally, components such as text-to-speech and visual cues improve accessibility and engagement across diverse user demographics.

The project design follows a modular and scalable approach. Each system function—image input, quiz handling, AI processing, recommendation generation, and design output—is encapsulated in its own component, enabling independent development, testing, and maintenance. This modularity not only enhances system clarity but also allows for future upgrades, such as integration with e- commerce platforms for product suggestions or the addition of augmented reality previews.

During development, various UML diagrams were employed to model system structure and behavior. The Class Diagram provided a blueprint of the system's core components and their relationships. The Object Diagram demonstrated real-time instance interactions. The Component and Deployment Diagrams helped visualize how system components are logically and physically distributed. Use Case, Sequence, and Activity Diagrams further illustrated the user flow, data processing, and system behavior in detail.

By combining scenario-based analysis with object-oriented design principles, the system was conceptualized and built to be robust, user-friendly, and ready for real-world deployment. The use of UML tools ensured effective communication between team members and stakeholders, offering a shared language to discuss requirements, architecture, and development priorities. It also facilitated early identification of potential risks and system bottlenecks.

Operational feasibility was a priority. The system was designed to require no technical expertise from the user. All technical processes such as AI interaction, image handling, and style recommendation are abstracted behind a clean UI, making the application usable even by non- designers. The seamless end-to-end flow ensures users can go from input to actionable design inspiration in just a few clicks.

Economically, the system uses free-tier services such as MongoDB Atlas, Replicate API, and Cloudinary, which make the platform viable for small-scale or startup-level deployment without incurring heavy infrastructure costs. In future iterations, the platform can be monetized through premium features such as design downloads, affiliate product links, or integration with virtual showrooms.

In conclusion, the AI Interior Designer system stands as a comprehensive and innovative application that bridges the gap between design inspiration and practical implementation. It empowers everyday users to make informed interior design choices using AI-generated visuals tailored to their unique spaces. The project successfully combines technical sophistication with user-centric thinking and sets the stage for future innovations in automated design, smart home personalization, and AI-powered creative assistance.

# 8.Bibliography

[1] Scolere, L. (2023). **"*Connected design learning: Aspiring designers, Pinterest, and social media literacy*"**. Journal of Interior Design, 48(3), 191–206. DOI: 10.1177/10717641231184214

[2] J. R. G. Cuevas et al**., "LayOut Loud: An AI-Powered Augmented Reality and Mobile Application for Room Interior Design and Layout Optimization**", 2024 International Conference on Intelligent Cybernetics Technology & Applications (ICICyTA), pp. 888–894.
DOI: 10.1109/ICICYTA64807.2024.10912928

[3] Abouelela, A., Al-Saud, K., AlAli, R., et al. (2024). **"Towards Greener Futures: AI-Powered Designs for Sustainable, Accessible, and Energy-Smart Urban Furniture to Improve Happiness and Well-being in Public Parks."**
DOI: 10.21203/rs.3.rs-5266205/v1

[4] AlShkipi, O. A., & Zahran, B. (2024). **"Implementation of artificial intelligence in interior design: systematic literature review",** Artificial Intelligence, vol. 4, p. 5.

[5] Almajaibel, M. K. E. A. (2024). **"How far does Artificial Intelligence (AI) evolve in the pursuit of Interior design as alternatives to traditional tools and their impact on the designer's function?"**, International Design Journal, vol. 14, no. 1, pp. 185–203.

[6] He, J. (2024). **"Research and Implementation of Intelligent Interior Design Algorithms Based on Artificial Intelligence and Big Data",** 2024 IEEE 3rd World Conference on Applied Intelligence and Computing (AIC), pp. 655–660.
DOI: 10.1109/AIC61668.2024.10730915

[7] Karan, E., Asgari, S., & Rashidi, A. (2021**). "A markov decision process workflow for automating interior design",** KSCE Journal of Civil Engineering, 25(9), 3199–3212.
DOI: 10.1007/s12205-021-1373-2

[8] Pylypchuk, O. D., Polubok, A. P., & Avdieieva, N. Y. (2022). **"Using artificial intelligence to create a practical tool for interior design incorporating artwork."**

[9] Hussein, G. K. (2023). "**Improving Design Efficiency Using Artificial Intelligence: A Study on the Role of Artificial Intelligence in Streamlining the Interior Design Process",** International Design Journal, vol. 13, no. 5, pp. 255–270.

[10] Yanhua, L. (2024). **"Research on the Application of Artificial Intelligence in Interior Design",** International Journal of Science and Engineering Applications.

# 9. Appendix

# Appendix-A
## Project Repository Details

**Project Title:** AI Powered Real Time Room Photo Interior Design Generator

**Batch:** A9

**Batch Members:**

1. Dayana Devisree (21B01A0540)
2. Jujjuru Rama Mydhili (21B01A0564)
3. Gollapalli Hrishitha (21B0A0553)
4. Jammalapudi Amsika (21B01A0563)
5. Boppana Monika (21B01A0528)

**Department:** Computer Science and Engineering

**Institution:** Shri Vishnu Engineering College for Women

**Guide:** G. Ramesh babu M. Tech (Ph.D)

**Submission Date:** [12/04/2025]

**Project Repository Link**

The complete project files, including source code, documentation, and additional resources, are available at the following GitHub repository:

**GitHub Repository:**

https://github.com/2711-amsi/AI-Powered-Real-Time-Room-Photo-Interior-Design-Generator

**For quick access, scan the QR code below:**

# Appendix B

## Technologies Used – Features and Benefits

### React.js

React.js was used to build the frontend of the application. It is a popular JavaScript library for creating dynamic and interactive user interfaces. React's component-based structure and use of the Virtual DOM enhance UI rendering efficiency. This results in a seamless experience for users navigating through quizzes, uploading images, and viewing design results. The reusable components also ensure the frontend is maintainable and scalable.

### Node.js and Express.js

Node.js, along with the Express.js framework, was implemented for the backend of the system. Node.js offers a non-blocking, event-driven architecture that ensures high performance under load, while Express provides routing and middleware support. Together, they form the backbone of the server-side logic, handling user requests, integrating APIs, and interacting with databases and cloud services.

### MongoDB Atlas

The project uses MongoDB Atlas as a NoSQL cloud database for storing user history, design metadata, feedback, and preferences. Its flexible schema and document-oriented design suit the dynamic nature of user data. MongoDB's scalability and cloud integration make it ideal for real-time applications that expect to grow in scope and complexity.

### Cloudinary

Cloudinary is integrated for managing image uploads and delivery. Users upload photos of their rooms, which are processed and stored in Cloudinary's cloud infrastructure. Its capabilities such as automatic format optimization, CDN delivery, and transformation tools ensure fast and secure handling of image assets, improving application performance and responsiveness.

### Replicate API (Jschoormans/comfyui-interior-remodel model)

The AI functionality of the platform is powered by Replicate's hosted machine learning models. The specific model used, Jschoormans/comfyui-interior-remodel, accepts a room image and a style preference and returns a photorealistic interior redesign. Replicate allows the team to use advanced deep learning models without maintaining complex GPU infrastructure, enabling rapid deployment and iteration.

### Axios

Axios is used as the HTTP client to handle communication between the frontend and backend. It supports asynchronous calls, error handling, and token-based requests. It is lightweight and plays a vital role in ensuring that API requests and responses are handled efficiently.

### Fuzzy Search JavaScript Library

For enhanced product discovery, a fuzzy search algorithm is employed to match AI-generated room elements with real-world products. This method allows approximate string matching, increasing the chances of finding similar items from a product database or external API even when the labels are not exact.

### Tailwind CSS

Tailwind CSS is used to streamline the styling of the frontend interface. It provides utility-first classes that allow rapid development of responsive, clean, and modern UI designs. Tailwind ensures visual consistency across all pages while maintaining design flexibility.

### Mongoose

Mongoose acts as an Object Data Modeling (ODM) library for MongoDB. It simplifies interactions with the database by allowing schema-based modeling and provides built-in validations, making the backend more reliable and structured.

### Streamlit (For Data Visualization - Optional Extension)

Although not part of the core user-facing system, Streamlit is considered for building internal dashboards for developers or admins. Streamlit enables rapid development of interactive dashboards in Python, useful for visualizing usage statistics, system logs, or performance metrics.

## Appendix C

### AI Workflow Description

The AI-powered system begins with user interaction on the frontend where a photo of the room and style preference (Minimal, Modern, or Classic) are submitted via a quiz interface. This data is sent to the backend, where the room image is first uploaded to Cloudinary. The secure image URL and user preferences are then passed to the Jschoormans/comfyui-interior-remodel model hosted on Replicate.

Once the AI model processes the input, it returns a redesigned image that reflects the chosen aesthetic. This image is displayed on the frontend, and the system then uses a fuzzy search and product API to identify decor items that visually match the generated design. These products are linked to online shopping platforms, providing actionable inspiration. The final step allows users to rate the generated design, and all data—including the quiz inputs, image URLs, and ratings—is stored in MongoDB for future reference.

# Appendix D
## Future Scope and Enhancements

In future versions of the system, several enhancements are proposed:

1. **Augmented Reality Integration**: Users could visualize AI-generated designs in their real space using AR, enhancing interactivity.
2. **User Authentication and Profiles**: Personalized dashboards and saved preferences
3. can be introduced through user accounts.
4. **Budget-Based Product Filtering**: Users may input a preferred price range to receive cost-effective suggestions from e-commerce sources.
5. **Multilingual and Voice Interaction**: Expanding accessibility through regional language support and voice-based control.
6. **Model Training with Custom Dataset**: Fine-tuning the AI model with a curated
7. dataset of Indian room styles or localized furniture options.