WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Virtual Interior Designer: A Client-Side Web Application for Real-Time Furniture Visualization

Dr Ramya B N[1], Abhishek Kumar A[2], Chethan TD[3], Darshan S[4], Dayananda J[5]

[1]Associate Professor, Department of Computer Science, Jyothy Institute of Technology, Bengaluru, India
[2]Department of Computer Science, Jyothy Institute of Technology, Bengaluru, India
[3]Department of Computer Science, Jyothy Institute of Technology, Bengaluru, India
[4]Department of Computer Science, Jyothy Institute of Technology, Bengaluru, India
[5]Department of Computer Science, Jyothy Institute of Technology, Bengaluru, India

A B S T R A C T

*Interior design planning often involves guesswork or expensive professional software. This paper introduces "Virtual Interior Designer," a lightweight web application that enables users to visualize furniture placement and wall color changes directly in their browser. Unlike traditional CAD tools that require heavy installation, this project utilizes **React.js** and **Konva.js** to render a multi-layered 2D canvas. Users can upload a room photograph, drag-and-drop furniture assets, and apply virtual paint using polygon mapping. To assist decision-making without high API costs, the system features a rule-based "Simulated AI" suggestion engine. The application operates entirely on the client side, ensuring privacy and zero-latency performance, and is deployed via Vercel.*

Keywords: **React.js, Konva.js, Web Application, Client-Side Rendering, Interior Design, Virtual Visualization.**

1. INTRODUCTION

Homeowners often struggle to visualize how new furniture or paint colors will look in their existing space before making a purchase. Traditional solutions like CAD software are too complex for average users, while mobile AR apps often require high-end hardware and large downloads.

This project aims to bridge that gap by providing a **browser-based tool** that is accessible on any device. The "Virtual Interior Designer" allows users to upload a static image of their room and interactively overlay 2D assets. By focusing on **2D canvas manipulation** rather than complex 3D rendering engines, this project utilizes high-performance JavaScript libraries to provide a fast, "good enough" visualization that helps users make quick decisions about layout and color schemes without a steep learning curve.

2. METHODOLOGY

The system is built as a Single Page Application (SPA) using a modern frontend stack. It does not rely on a persistent backend database; instead, all logic executes within the user's browser to ensure speed and privacy.

2.1 Frontend Architecture

The core application is built using **React.js** bundled with **Vite** for optimized build performance. **Tailwind CSS** is utilized for a responsive user interface that adapts to different screen sizes. State management is handled via React Hooks (useState, useEffect), which track the coordinates, rotation, and dimensions of every object on the screen.

2.2 Interactive Canvas Engine

The critical functionality is powered by **Konva.js** (via react-konva). The application creates a multi-layered HTML5 Canvas:

- **Layer 1 (Background):** Renders the user's uploaded room image. We implemented a custom scaling algorithm to fit images of any aspect ratio into the workspace without distortion.
- **Layer 2 (Paint):** Allows users to draw polygons over walls. These polygons are filled with semi-transparent colors to simulate paint overlay.

- **Layer 3 (Furniture):** Renders furniture assets (PNGs with transparent backgrounds). A **Transformer** component allows users to drag, resize, and rotate these assets by 360 degrees.

2.3 Simulated AI Feature

To assist users with design choices, the system includes a recommendation engine. Due to the high computational cost of real-time image analysis APIs, this project implements a Simulated AI approach. A pre-defined dataset of 50+ logic-based design rules is queried using a randomization algorithm with a setTimeout delay to mimic server-side processing, providing instant, relevant design tips.

2.4 System Data Flow

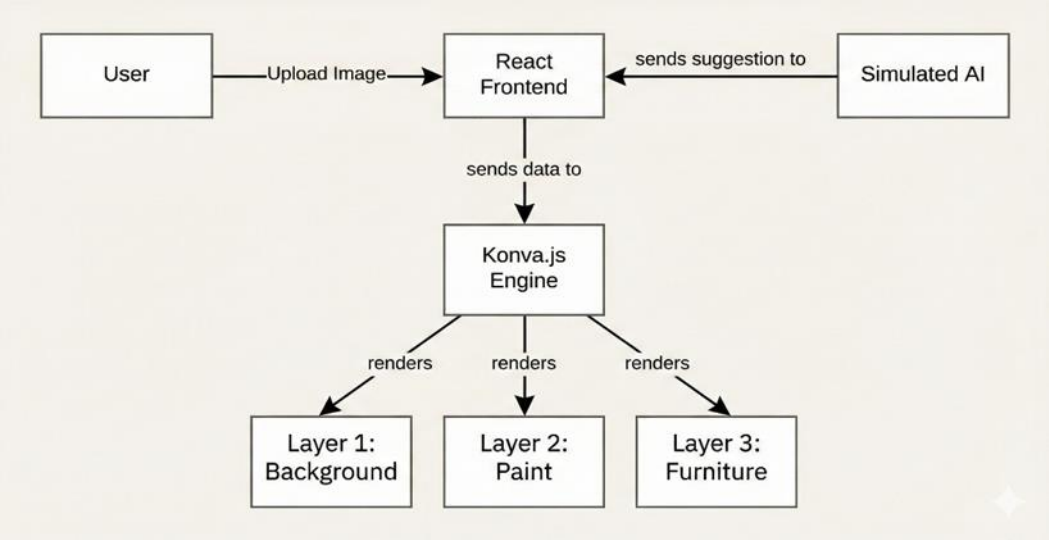

Fig. 1: System Architecture and Data Flow Diagram

Figure 1 illustrates the architectural flow of the Virtual Interior Designer. The system operates entirely on the client side to ensure low latency. The user interacts with the **React Frontend**, which manages the application state. When an image is uploaded, it is processed into a local blob URL. The **Konva.js Engine** then acts as the core renderer, taking inputs from the user (drag coordinates, paint polygons) and updating the HTML5 Canvas in real-time. The **Simulated AI Module** runs asynchronously, generating design logic without blocking the main thread, and feeding recommendations back to the UI.

3.RESULTS AND DISCUSSIONS

The final application successfully performs all intended tasks with zero backend latency. The user interface allows for seamless uploading of images and manipulation of assets.

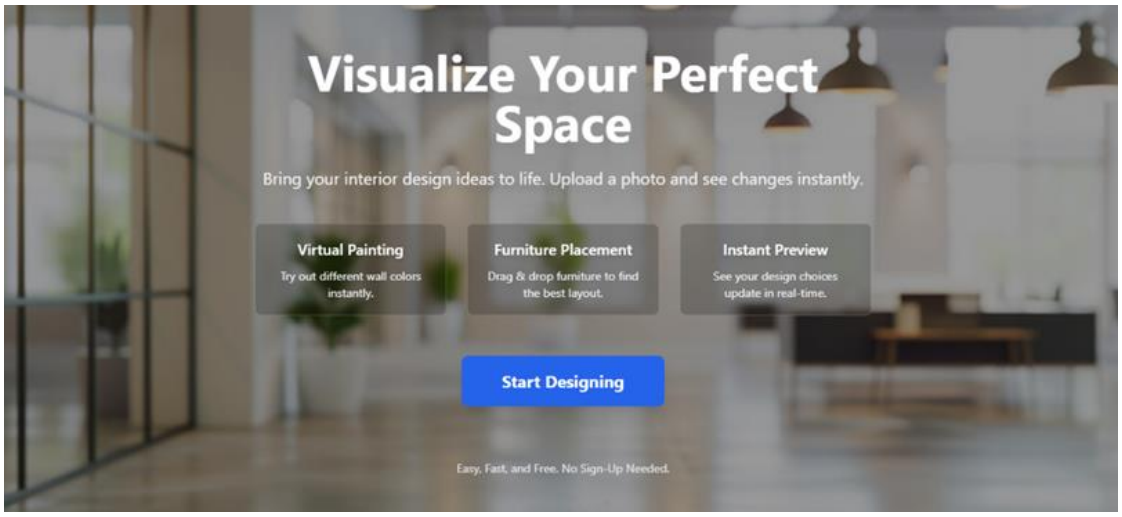

Fig. 2 - The Home Page showing the "Start Designing" interface.

As shown in Fig. 2, the landing page uses fade-in animations to engage the user. Fig. 2 demonstrates the core workspace where a user can upload a living room photo.

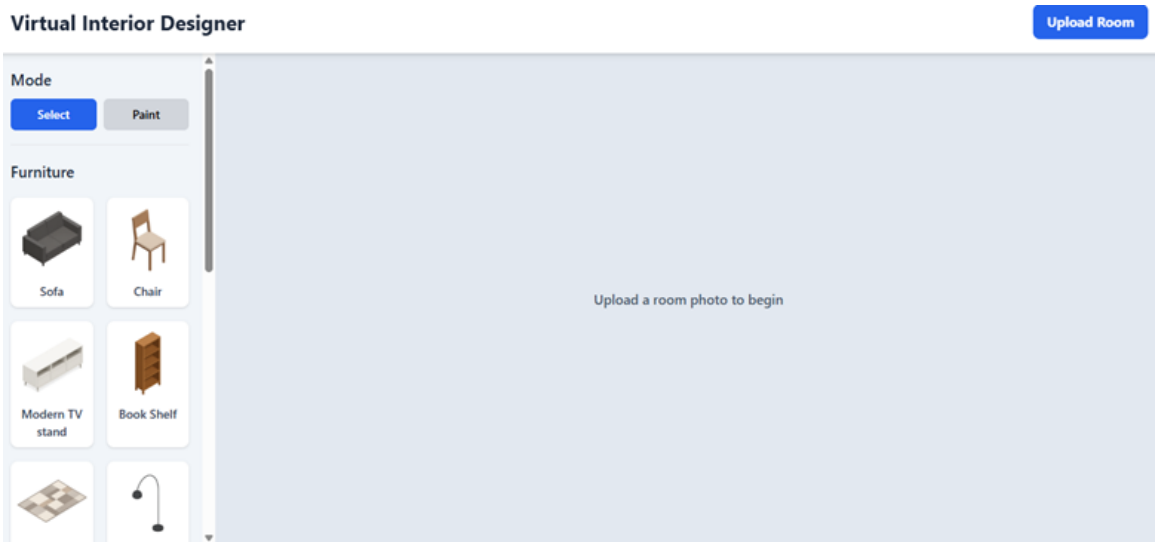

Fig. 3 - The Workspace

Test Case ID	Feature Tested	Input / Action	Expected Outcome	Status
TC-01	Image Upload	User selects a 5MB JPEG file.	Image loads and scales to fit the canvas without distortion.	**Success**
TC-02	Furniture Placement	Click 'Sofa' icon from sidebar.	Sofa asset appears on Layer 3 (top layer).	**Success**
TC-03	Object Manipulation	Drag, resize, and rotate the sofa.	Object follows cursor and transforms smoothly.	**Success**
TC-04	Virtual Painting	Draw a 4-point polygon on a wall.	Polygon closes and fills with semi-transparent color.	**Success**
TC-05	AI Suggestion	Click 'Get Suggestion' button.	System waits 4-7s and displays relevant text tip.	**Success**

4. CONCLUSION

The "Virtual Interior Designer" successfully demonstrates that complex graphical manipulation can be performed entirely on the client-side using modern JavaScript libraries. By eliminating the need for a backend server, the application remains fast, secure, and cost-effective to host. Future enhancements could include integrating a real Generative AI API for image-based suggestions and a Firebase backend to allow users to save and share their project history.

Acknowledgements

We would like to thank our guide, Dr Ramya B N, for their continuous support and guidance throughout this project.

References

[1] Facebook Open Source. (2023). *React Documentation*. Retrieved from https://react.dev/

[2] Lavrenov, A. (2023). *Konva.js API Documentation*. Retrieved from https://konvajs.org/

[3] Tailwind Labs. (2023). *Tailwind CSS Documentation*. Retrieved from https://tailwindcss.com/

[4] Framer. (2023). *Framer Motion Documentation*. Retrieved from https://www.framer.com/motion/

[5] Vercel Inc. (2023). *Vite - Next Generation Frontend Tooling*. Retrieved from https://vitejs.dev/