# Software Requirements Specifications

## Interior Design Advisor
## (Your Smart Designing Assistant)

**Project Code:**

CS-IDA-2025

**Internal Advisor:**

Dr. Faisal Shahzad

**Project Manager:**

Dr. Muhammad Illyas

**Project Team:**

| | |
|---|---|
| Badar Munir (Group Leader) | BSCS51F21S095 |
| Sana Jabbar | BSCS51F22S030 |
| Hafsa Zain | BSCS51F22S001 |

**Submission Date:**

21/10/2025

_____

**Project Manager's Signature**

# Document Information

| Category | Information |
|---|---|
| Customer | Computer Science, UOS |
| Project | Interior Design Advisor |
| Document | Requirement Specifications |
| Document Version | 1.0 |
| Identifier | CS-IDA-2025-RS |
| Status | Draft |
| Author(s) | Badar Munir        BSCS51F21S095<br>Hafsa Hadi        BSCS51F22S001<br>Sana Jabbar        BSCS51F22S030 |
| Approver(s) | Dr. Muhammad Ilyas |
| Issue Date | Oct 17,2025 |
| Document Location | https://github.com/Badarmunir1/capstone-2025/ |
| Distribution | 1. Dr. Faisal Shahzad<br>2. Dr. Muhammad Ilyas |

# Definition of Terms, Acronyms and Abbreviations

| Term | Description |
|---|---|
| ML | Machine Learning |
| CNN | Convolutional Neural Network |
| UI/UX | User Interface / User Experience |
| OpenCV | Open Source Computer Vision Library |
| HEX | Hexadecimal Color Code |
| API | Application Programming Interface |
| RS | Requirement Specification |

# Table of Contents

# 1. Introduction

## 1.1 Purpose of Document

The purpose of this document is to formally define the **Software Requirements Specifications (SRS)** for the Interior Design Advisor project. It outlines the system's intended functionality, performance expectations, constraints, and design considerations.
This document is primarily intended for:

- Project advisors and evaluators for assessment.
- The development team to guide implementation.

  Future maintainers and testers to ensure consistency with requirements.

## 1.2 Project Overview

The **Interior Design Advisor** is a web-app smart assistant that helps users enhance their interior spaces through AI-powered design recommendations. By uploading an image of a room, the system analyzes the visual content using computer vision to extract dominant colors and classify the room style (e.g., Modern, Rustic, Minimalist).

It then suggests suitable furniture, layout, and color themes to match the detected style. The solution leverages lightweight ML models such as **MobileNetV2** and **OpenCV** to perform efficient, real-time image analysis suitable for browser-based applications.

The system targets everyday users seeking personalized design suggestions without needing professional expertise.

## 1.3 Scope

**In Scope:**

- Image  analysis
- Color palette extraction
- Room style classification
- Rule-based recommendation for furniture and layout
- Web-based UI for displaying results

**Out of Scope:**

- AR/VR visualization
- Real-time 3D room reconstruction
- E-commerce integration
- Adaptive learning recommendations

The system will provide a balanced combination of AI-powered insights and user interactivity while maintaining low computational overhead.

# 2. Overall System Description

## 2.1  User Characteristics

**General Users:** Individuals with limited or no design experience who want quick suggestions for interior improvement.

**Students/Researchers:** Those studying computer vision or design-related topics, using the tool for academic purposes.

**Evaluators/Developers:** Project advisors and developers testing functionality and usability

All users are expected to have basic computer literacy, including image uploading and navigating a web interface.

## 2.2 Operating Environment

- **Frontend:** Runs in standard web browsers (Chrome, Firefox, Edge).
- **Backend:** Python (Flask or FastAPI) environment hosted locally or on a cloud server.
- **Database:** SQLite for storing user and recommendation data.
- **Hardware Requirements:**

    ○ Minimum 4GB RAM, Dual-core processor
    ○ GPU optional for local model training

- **Operating System:** Windows, Linux, or macOS

## 2.3 System Constraints

- **Software Constraints:** Dependent on OpenCV, TensorFlow/Keras, Flask, and standard Python libraries.
- **Hardware Constraints:** High-resolution image uploads may increase processing time.
- **User Constraints:** The tool is designed for static room images only; dynamic camera input or 3D scenes are not supported.
- **Legal Constraints:** Uses only open-access datasets for training (e.g., Open Images Dataset).
- **Environmental Constraints:** Requires stable internet for cloud deployment.

# 3. External Interface Requirements

## 3.1 Hardware Interfaces

The system interacts indirectly with hardware through user devices.

- **User-side:** Laptop, desktop, or tablet capable of running a modern web browser.
- **Server-side:** A machine (local or hosted) with sufficient computational power to handle image uploads, model inference, and database queries.

    ○ Minimum: 8 GB RAM, 2 GHz processor
    ○ Optional GPU for training and faster inference

- **Peripheral Devices:** Users may use webcams or cameras to capture room images but this hardware is not directly controlled by the system.

## 3.2 Software Interfaces

*The system depends on multiple software libraries and frameworks:*

| Component | Description |
|---|---|
| *Operating System* | *Windows / Linux / macOS* |

| | |
|---|---|
| *Backend Framework* | *Flask or FastAPI (Python)* |
| *Frontend Framework* | *ReactJS / Streamlit* |
| *Machine Learning Library* | *TensorFlow / Keras* |
| *Computer Vision Library* | *OpenCV* |
| *Database* | *SQLite* |
| *Visualization API* | *Matplotlib or ColorThief for color palette rendering* |
| *Web Server* | *Localhost or any HTTP-compatible web server (e.g., Nginx)* |

*The system interfaces between these software components to perform data flow from user image input → ML processing → output recommendation.*

## 3.3 Communications Interfaces

● **Client-Server Communication:**

    ○ RESTful APIs (HTTP/HTTPS) used for communication between frontend and backend.
    ○ JSON used as the standard data exchange format.

● **Network Protocols:**

    ○ TCP/IP for client-server data transmission.
    ○ HTTPS for secure data exchange when deployed online.

● **Security:**

    ○ SSL encryption for communication (if deployed publicly).
    ○ No sensitive personal data will be transmitted.

# 4. Functional Requirements

The system's functional requirements are grouped into modules:

| Module | Functional Requirement |
|---|---|
| *User Interface* | *Allow users to upload a room image.- Display extracted color palette, style, and design suggestions.- Provide clean, responsive, and intuitive UI.* |
| *Image Preprocessing* | *Resize and normalize uploaded images.- Convert RGB to other color spaces (e.g., HSV) using OpenCV.- Handle invalid or corrupted image formats gracefully.* |
| *Color Palette Extraction* | *Use K-Means clustering to detect dominant colors.- Display color HEX codes and sample swatches to users.* |

| | |
|---|---|
| ***Style Classification*** | *Use pretrained MobileNetV2 model to classify images into style categories (Modern, Rustic, Minimalist, Classic).- Return style label and confidence score.* |
| ***Recommendation Engine*** | *Match style and color results with rule-based database.- Suggest suitable furniture and layout ideas.* |
| ***Database Operations*** | *Log user interactions (image name, detected style, timestamp).- Store feedback from user testing phase.* |
| ***Testing Module*** | *Allow evaluation using test dataset.- Record accuracy, confusion matrix, and processing time.* |

# 5. Non-Functional Requirements

## 5.1 Performance Requirements

- The system should process and return design recommendations within **5–8 seconds** per image.
- Style classification model accuracy target: **≥85%**.
- Application should support **simultaneous use by 5–10 users** on a local or hosted server.
- The interface should remain responsive under normal load.

## 5.2 Safety Requirements

- The system must prevent upload of malicious or executable files.
- The application should restrict excessive image size (>5MB) to prevent crashes.
- Regular data backups should be maintained for test and feedback data.

## 5.3 Security Requirements

- Implement **input validation** to prevent code injection or malicious uploads.
- Use **HTTPS** when deployed online to secure data communication.
- Maintain user anonymity; no personal or location data stored.
- System access is limited to authorized project users during the development phase.

## 5.4 User Documentation

The following documentation will be provided with the system:

1. **User Manual** – step-by-step guide on how to upload an image and interpret results.
2. **Installation Guide** – setup instructions for local or server-based deployment.
3. **Technical Documentation** – for developers including API endpoints and model configuration.
4. **Testing Report** – showing performance metrics and feedback results.

# 6. Assumptions and Dependencies

- It is assumed that all users have access to a stable internet and a web browser.
- The machine learning model depends on publicly available datasets such as the **Open Images Dataset**.
- The system assumes well-lit, clear room images for accurate analysis.
- Future extension (e.g., e-commerce integration) may depend on APIs from external vendors (e.g., IKEA, Wayfair).
- Deployment timeline depends on completion of dataset preparation and model fine-tuning.

# 7. References

| Ref. No. | Document Title | Date of Release/ Publication | Document Source |
|---|---|---|---|
| [1] | Project Proposal – Interior Design Advisor | Oct 5, 2025 | https://github.com/Badarmunir1/capstone-2025 |
| [2] | OpenCV Documentation | 2024 | https://docs.opencv.org |
| [3] | MobileNetV2: Efficient Convolutional Neural Networks for Mobile Vision Applications | 2017 | https://arxiv.org/abs/1704.04861 |
| [4] | Adobe Color Theory Guide | 2023 | https://color.adobe.com |
| [5] | Open Images Dataset | 2024 | https://storage.googleapis.com/openimages/web/index.html |