AI-Based Interior Design Generator

Intelligent interior redesign and visualization powered by Generative AI

and advanced spatial modelling

Project Overview:

The AI Interior Design Studio is an intelligent web application that leverages cutting-edge artificial intelligence to transform living spaces through advanced computer vision and generative AI. This platform allows homeowners, interior designers, and real estate professionals to upload room photos and instantly generate multiple design variations in different styles while preserving the original architectural structure.

The system bridges the gap between professional interior design and accessibility by using Stable Diffusion with ControlNet for structure-preserving transformations, Google Gemini for intelligent image analysis, and advanced computer vision for room element detection. This enables users to visualize design concepts instantly without requiring technical expertise or expensive software.

Scenario 1: Home Renovation Planning

Homeowners planning renovations often struggle to visualize how their space will look after changes, leading to uncertainty and costly redesign decisions. The AI Interior Design Studio eliminates this problem by allowing users to upload a photo of their current room and instantly explore multiple interior design variations. The system intelligently analyses the image using computer vision, identifies room structure (walls, windows, doors, ceiling, flooring), and applies different design aesthetics without altering the architectural layout. Users can preview how their room would look in various popular styles such as minimalist, bohemian, Scandinavian, industrial, or luxury modern simply by selecting their preferred style from the interface.

With this real-time visualization, homeowners gain clarity and confidence before investing in paint, furniture, or décor. Instead of relying on imagination or static mood boards, they can compare photorealistic outputs side by side, experiment with colour palettes and lighting settings, and refine ideas collaboratively with contractors or interior designers. This helps users make informed decisions, reduces redesign costs, and speeds up the renovation planning process. In essence, the platform empowers users to experience their space before it physically exists.

Scenario 2: Real Estate and Property Staging

In the real estate sector, visual presentation plays a critical role in capturing buyer interest and increasing property value. Many listings fail to attract attention simply because the rooms are empty, outdated, or poorly furnished. Using the AI Interior Design Studio, real estate agents and property stagers can transform plain property photos into stylish, fully furnished interiors within seconds without heavy spending on physical staging or renting furniture. By uploading photos of a property, the AI intelligently adds décor, furniture, lighting, and texture enhancements, keeping room structure intact while showcasing its full visual potential.

These AI-generated visualizations help clients emotionally connect with the space and envision how they would live in it. Well-staged images increase engagement significantly improve the chances of faster sales or rentals. Instead of showing buyers an empty room, the system enables agents to present multiple themed visualizations such as modern luxury, cozy minimalism, or urban depending on the target market. This makes property marketing more compelling, cost-efficient, and innovative, turning every listing into an immersive experience rather than just a photograph.

Architecture Overview:

The AI-Based Interior Design Generator is built on a modular multi-modal architecture that combines advanced computer vision, generative AI, and real-time interaction. The process begins by preprocessing the uploaded image to standardize input quality. Using PyTorch-based deep learning models, the system performs semantic segmentation (Mask2Former) to detect room components such as walls, floors, and windows, while MiDaS depth estimation creates a depth map to understand spatial geometry. These structural insights, along with user-selected design preferences (interior style, lighting, room type, and colour scheme), are captured through Streamlit's frontend interface.

In the core generative stage, Stable Diffusion with ControlNet uses the depth and segmentation maps to redesign the interior while preserving the original architecture. The generated output is refined through an OpenCV-based post-processing module that enhances clarity, texture, and colour balance. Finally, Google Gemini generates a natural language description of the redesigned space, enabling convenient use for presentations or client proposals. This architecture enables an end-to-end workflow from raw room image to a polished, photorealistic design concept with descriptive insights.

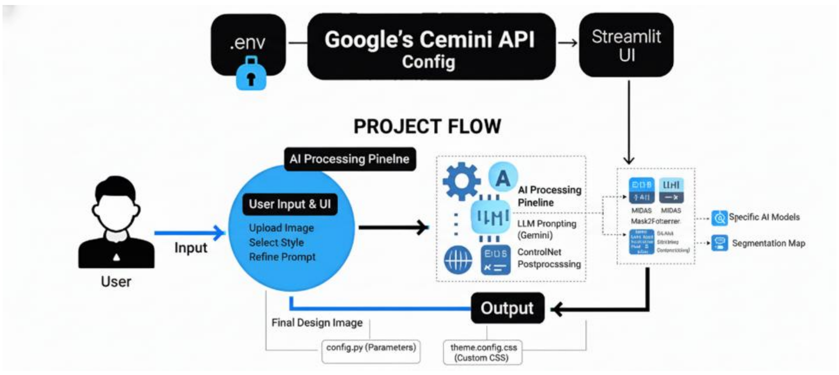


Figure 1: System architecture diagram

Core Technologies:

- **Streamlit**: Frontend framework for building the interactive web interface, managing image uploads, and displaying AI-generated design transformations in real-time.
- **Stable Diffusion with ControlNet**: Advanced generative AI pipeline for structure-preserving image generation that maintains room architecture while transforming interior elements.
- **Google Gemini 2.0 Flash**: Multimodal AI model used for intelligent image analysis and generating detailed descriptions of transformed spaces.
- **PyTorch**: Deep learning framework powering computer vision models for depth estimation and semantic segmentation.
- **Transformers Library**: Pre-trained models for depth estimation (MiDaS) and universal segmentation (Mask2Former) for room analysis.
- **OpenCV**: Computer vision library for image enhancement and post-processing operations.
- **PIL (Python Imaging Library)**: Image processing and manipulation for preprocessing and format handling.

Component-Wise Architecture:

Component	**Description**
User Interface (Streamlit)	Interactive web interface with tabbed navigation for image upload, design configuration, results viewing, and project information
Image Preprocessing Engine	Handles image loading, resizing, and format standardization for consistent processing across the pipeline
Depth Mapping Module	Utilizes MiDaS depth estimation to create 3D spatial understanding of room geometry and proportions
Semantic Segmentation System	Employs Mask2Former for detecting and classifying room elements (walls, floors, ceilings, windows, doors)
ControlNet Pipeline	Stable Diffusion integration with depth control for structure-preserving interior design transformations
AI Enhancement Engine	Post-processing module that applies detail enhancement and quality improvements to generated images

Style Configuration System	Manages design preferences including styles, colour schemes, room types, and lighting options
Image Description Generator	Google Gemini integration for generating detailed, natural language descriptions of transformed spaces
Session Management	Maintains user state across interactions including uploaded images, processing results, and generated designs
Help & Documentation Section	Displays project overview, key features, and quick usage guidance within the application interface.

Pre-requisites:

1. **Python Environment Setup:** The system requires Python 3.9 with comprehensive AI/ML library support. Create and activate a dedicated virtual environment for dependency management.

   Official Documentation: https://www.python.org/downloads/

2. **Streamlit Installation & Configuration**: Streamlit powers the interactive web interface used to upload room images and generate interior design transformations.

   Docs: https://docs.streamlit.io/library/get-started/installation
   Tutorial: https://www.youtube.com/watch?v=JwSS70SZdyM

3. **PyTorch + Deep Learning Dependencies**: Install PyTorch with the appropriate CUDA configuration for GPU acceleration.

   PyTorch Installation Guide: https://pytorch.org/get-started/locally/
   CUDA Toolkit (optional for GPU support): https://developer.nvidia.com/cuda-toolkit

4. **Hugging Face Transformers & Diffusers**
   Required to load:

   - Stable Diffusion models (image generation)
   - ControlNet (structure-guided image generation)

   Transformers Docs: https://huggingface.co/docs/transformers/installation
   Diffusers Docs: https://huggingface.co/docs/diffusers/installation

5. **Computer Vision Libraries**: OpenCV and Pillow, these libraries are responsible for image enhancement, segmentation overlay generation, depth-map preprocessing, and maintaining structural accuracy before passing the image to the generative model.

6. **Google Gemini AI Integration**: For generating image captions and design descriptions using Gemini AI.

   Gemini API Docs: https://ai.google.dev/gemini-api/docs
   Google AI Studio: https://aistudio.google.com/

7. **Dependency Installation**: All core dependencies can be installed using a single command:

   ```
   pip install streamlit torch transformers diffusers opencv-python pillow matplotlib google-generativeai
   ```

8. **Development Tools (Recommended IDEs)**

   Visual Studio Code: https://code.visualstudio.com/

   PyCharm (Community/Professional): https://www.jetbrains.com/pycharm/download/

9. **Hardware Requirements**

   - Minimum: 8GB RAM, 10GB free storage
   - Recommended: 16GB+ RAM, NVIDIA GPU with 8GB+ VRAM, 20GB free storage
   - Storage: Additional space required for model caching (~5-10GB)

Project Flow:

1. AI Model Infrastructure Initialization

- **Activity 1.1:** Obtain Google Gemini API Key and store securely in .env.
- **Activity 1.2:** Generate Hugging Face token and enable access to required models (ControlNet, Stable Diffusion, Segmentation, Depth).
- **Activity 1.3:** Create virtual environment, install required dependencies (streamlit, torch, diffusers, etc.), validate GPU, and set up project folder structure.

2. Core Image Processing Pipeline Development

- **Activity 2.1:** Implement semantic segmentation using Mask2Former to detect room elements (walls, windows, floor, furniture).
- **Activity 2.2:** Generate depth maps using MiDaS model for spatial & 3D structural understanding.
- **Activity 2.3:** Build preprocessing module to enhance images, standardize formats, and optimize session-level caching.
- **Activity 2.4:** Combine segmentation + depth data to create room structure analysis output.

3. AI-Powered Design Generation Engine

- **Activity 3.1:** Configure Stable Diffusion + ControlNet pipeline for structure-preserving generation.
- **Activity 3.2:** Implement dynamic prompt engineering with design style + color + lighting customization.
- **Activity 3.3:** Support multiple image variants and apply enhancement pipeline for high-quality output.

4. Streamlit UI Integration & Application Logic

- **Activity 4.1:** Implement session management and backend logic connection.
- **Activity 4.2:** Develop multi-tab UI (Home, Input, Generate Design, Download, About).

- **Activity 4.3:** Enable image upload, segmentation & depth visualization, and preview.
- **Activity 4.4:** Build design preference selection (style, color scheme, lighting, room type).
- **Activity 4.5:** Display generated images with download/export options and integrate styling/CSS.

5. Testing, Optimization & Deployment

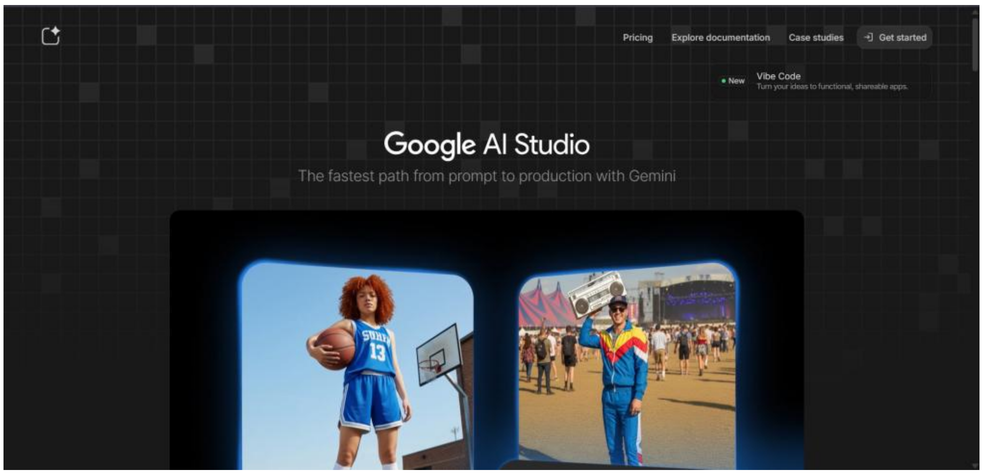
- **Activity 5.1:** Verify UI navigation flow (Home → Input → Output) and test interface usability.
- **Activity 5.2:** Test pipeline using real room images and confirm segmentation/depth → generation accuracy.
- **Activity 5.3:** Optimize generation speed, add progress indicators, handle errors, and validate GPU utilization.

MILESTONE 1: AI Model Infrastructure Initialization

In this foundational stage, the objective is to establish and configure the core AI model infrastructure required to power the AI Interior Design Studio. The platform relies on Google's Gemini 2.0 Flash model for intelligent image analysis and description generation, along with Hugging Face models for computer vision tasks including depth estimation, semantic segmentation, and Stable Diffusion with ControlNet for structure-preserving design generation. This milestone ensures secure API key management, proper model authentication, and validation of all AI service connectivity.

Activity 1.1: Obtain Google Gemini API Key

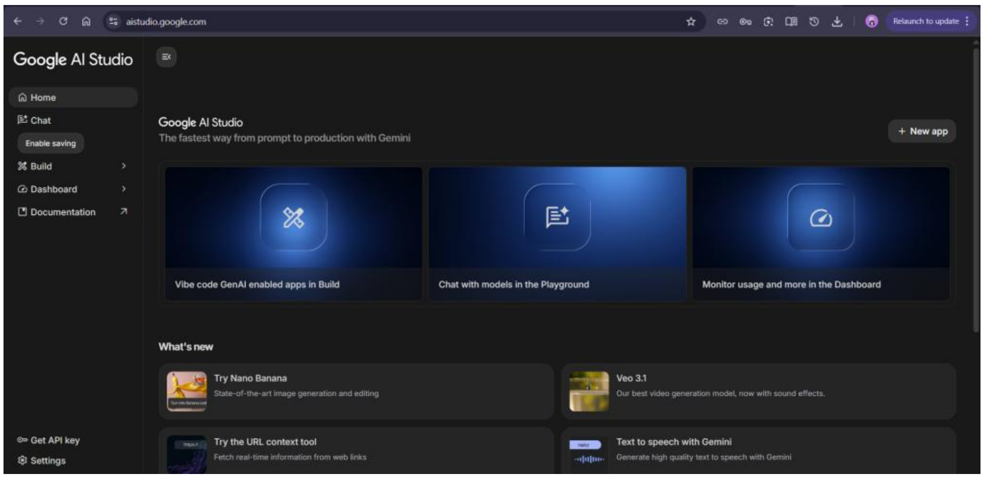
- Sign in or create an account on the official Google AI Studio platform https://aistudio.google.com/



Pricing
Explore documentation
Case studies
Get started
New
Vibe Code
Turn your ideas to functional, shareable apps.
Google AI Studio
The fastest path from prompt to production with Gemini

Figure 2: Google AI Studio Console

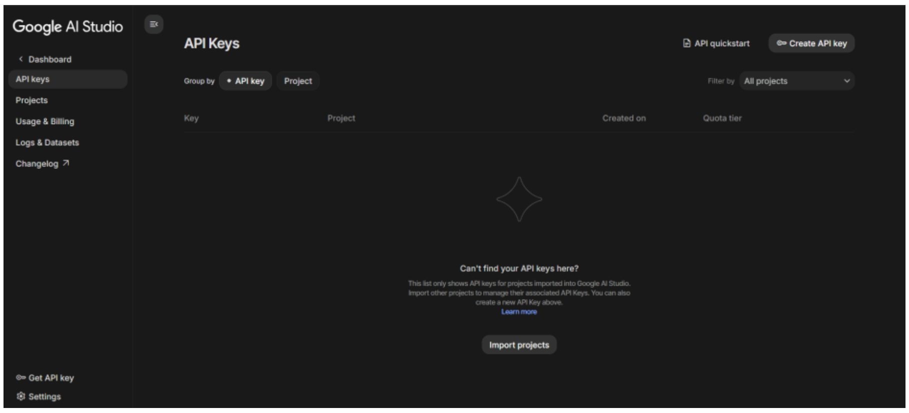
- Clicking on "Get Started" that's leads to this Google AI Dashboard:



aistudio.google.com
Relaunch to update
Google AI Studio
Home
Chat
Enable saving
Build
Dashboard
Documentation
Get API key
Settings
Google AI Studio
The fastest way from prompt to production with Gemini
New app
Vibe code GenAI enabled apps in Build
Chat with models in the Playground
Monitor usage and more in the Dashboard
What's new
Try Nano Banana
State-of-the-art image generation and editing
Veo 3.1
Our best video generation model, now with sound effects.
Try the URL context tool
Fetch real-time information from web links
Text to speech with Gemini
Generate high quality text to speech with Gemini

Figure 3: Google AI Studio Dashboard

- Navigate to the Get API Keys section under your account dashboard.



Key	Project	Created on	Quota tier

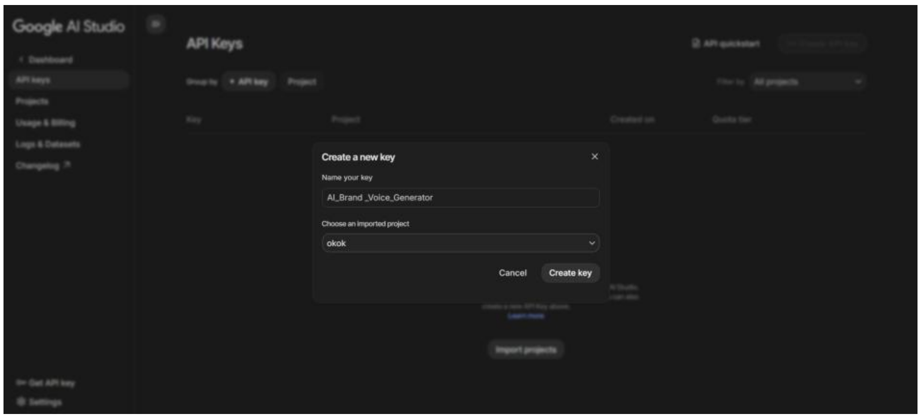
Can't find your API keys here?

This list only shows API keys for projects imported into Google AI Studio. Import other projects to manage their associated API Keys. You can also create a new API Key above.

Learn more

Figure 4: API Key Management Console

- Click on **"Create API Key"** to generate a new key for your project.
- Name the key appropriately (e.g., AI_Interior_Design_Studio) for easy identification.



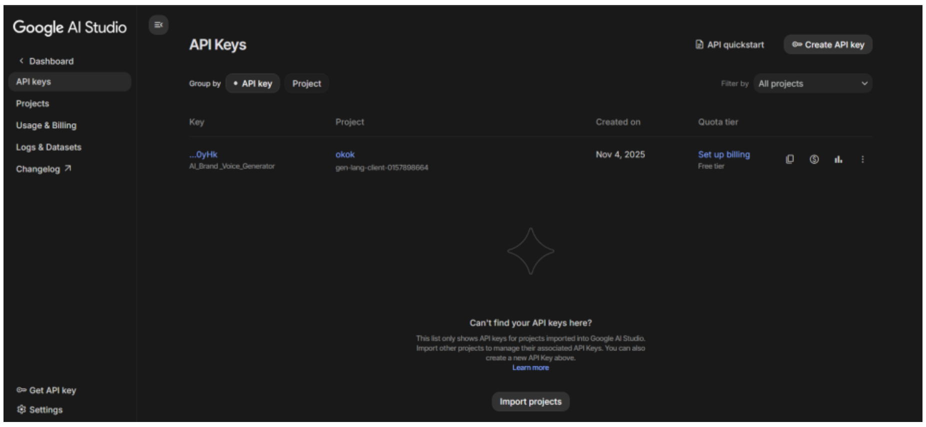
Create a new key

Name your key: AI_Brand _Voice_Generator

Choose an imported project: okok

Figure 5: Creating API Key

- Copy and securely store the generated API key for later configuration in your .env file.



Key	Project	Created on	Quota tier
...0yHk AI_Brand _Voice_Generator	okok gen-lang-client-0157898664	Nov 4, 2025	Set up billing Free tier

Can't find your API keys here?

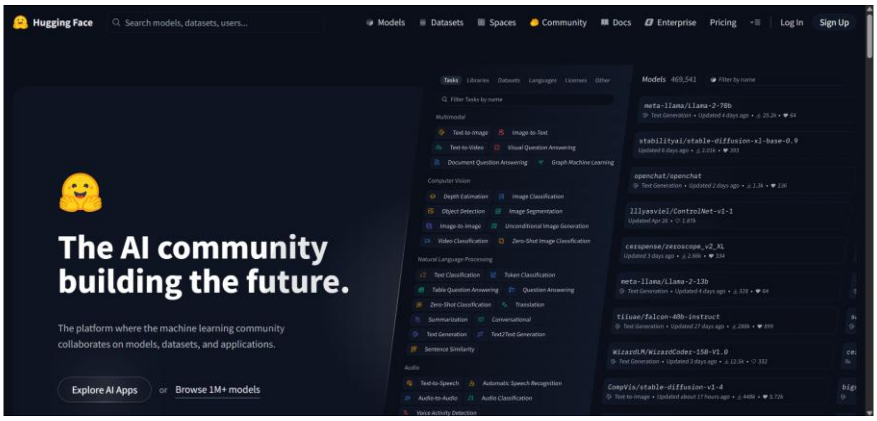
This list only shows API keys for projects imported into Google AI Studio. Import other projects to manage their associated API Keys. You can also create a new API Key above.

Learn more

Figure 6: Showing the API Key and Project name on Google Console

Activity 1.2: Set Up Hugging Face Token and Model Access

- Create account or sign in to Hugging Face platform https://huggingface.co/

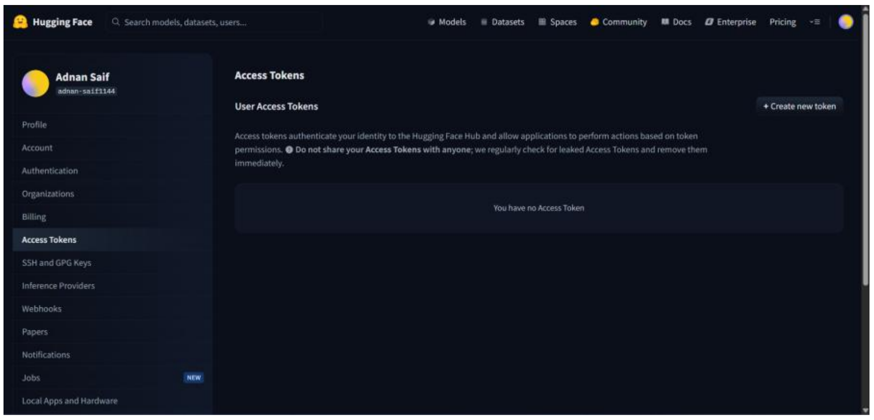


The AI community building the future.

The platform where the machine learning community collaborates on models, datasets, and applications.

Figure 7: Hugging Face home page

- Navigate to Settings → Access Tokens in your account dashboard



Access Tokens

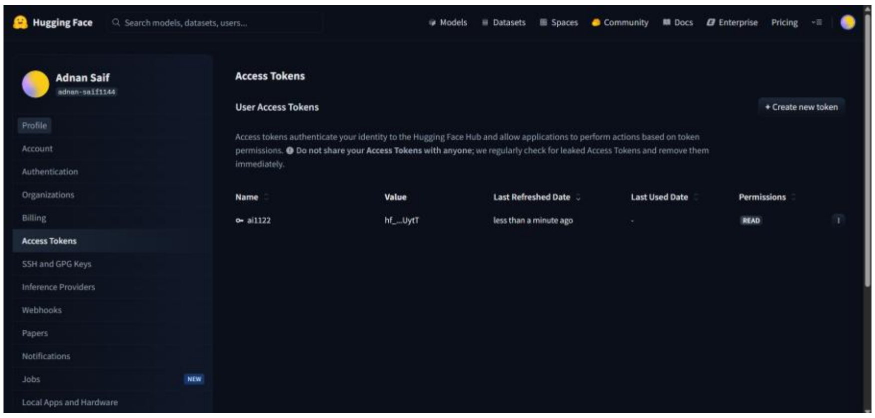
User Access Tokens

Access tokens authenticate your identity to the Hugging Face Hub and allow applications to perform actions based on token permissions. **Do not share your Access Tokens with anyone**; we regularly check for leaked Access Tokens and remove them immediately.

You have no Access Token

Figure 8: Creating Access Token

- Generate a new token with appropriate permissions for model access



Access Tokens

User Access Tokens

Access tokens authenticate your identity to the Hugging Face Hub and allow applications to perform actions based on token permissions. **Do not share your Access Tokens with anyone**; we regularly check for leaked Access Tokens and remove them immediately.

Name	Value	Last Refreshed Date	Last Used Date	Permissions
ai1122	hf_...UytT	less than a minute ago	-	READ

Figure 9: Hugging Face Token

- Configure token for accessing required models:
    - ➢ Stable Diffusion v1.5 (runwayml/stable-diffusion-v1-5)
    - ➢ ControlNet Depth (lllyasviel/sd-controlnet-depth)
    - ➢ Mask2Former Segmentation (facebook/mask2former-swin-large-ade-semantic)
    - ➢ MiDaS Depth Estimation (Intel/dpt-large)

Activity 1.3: Environment Setup and Dependency Configuration

- Create a dedicated Python virtual environment for the project and activating the environment.

```
python -m venv AIIDG
```

```
AIIDG\Scripts\activate
```

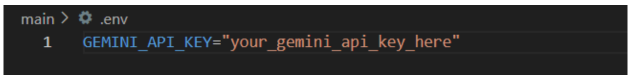
- Install core Python dependencies for the AI pipeline:

```
pip install streamlit torch torchvision transformers diffusers accelerate pillow opencv-python python-dotenv
```

- Alternatively, the same dependencies are documented inside a requirements.txt file, allowing easy installation using:

```
pip install -r requirements.txt
```

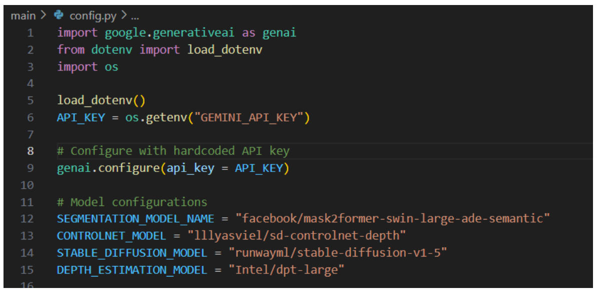
- Configure environment variables in .env file:



```
GEMINI_API_KEY="your_gemini_api_key_here"
```

Figure 10: Loading the Gemini key

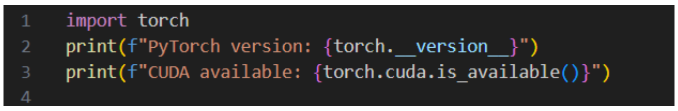
- Implement secure configuration management in config.py:



```python
import google.generativeai as genai
from dotenv import load_dotenv
import os

load_dotenv()
API_KEY = os.getenv("GEMINI_API_KEY")

# Configure with hardcoded API key
genai.configure(api_key = API_KEY)

# Model configurations
SEGMENTATION_MODEL_NAME = "facebook/mask2former-swin-large-ade-semantic"
CONTROLNET_MODEL = "lllyasviel/sd-controlnet-depth"
STABLE_DIFFUSION_MODEL = "runwayml/stable-diffusion-v1-5"
DEPTH_ESTIMATION_MODEL = "Intel/dpt-large"
```

Figure 11: Configuring the models in application

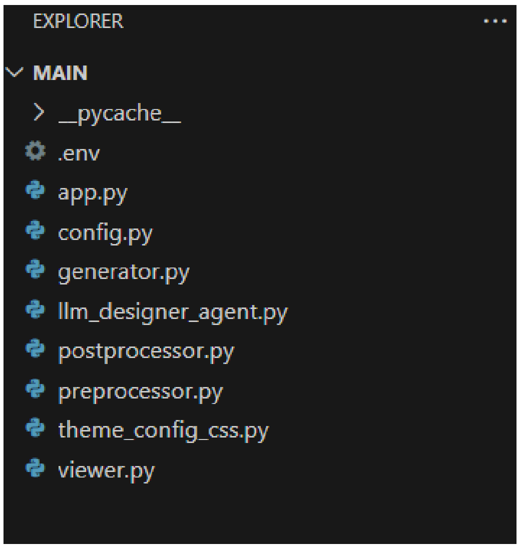
- Validate PyTorch installation and GPU availability:



```
import torch
print(f"PyTorch version: {torch.__version__}")
print(f"CUDA available: {torch.cuda.is_available()}")
```

Figure 12: Checking installation

- Create the project file structure for modular development:



```
EXPLORER
MAIN
  __pycache__
  .env
  app.py
  config.py
  generator.py
  llm_designer_agent.py
  postprocessor.py
  preprocessor.py
  theme_config_css.py
  viewer.py
```

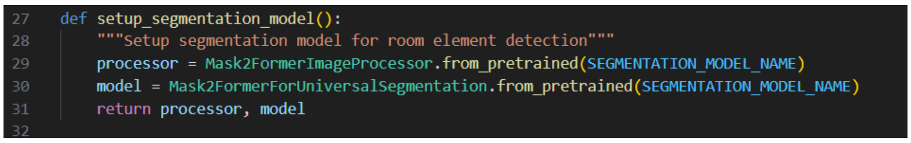
Figure 13: Folder structure

MILESTONE 2: Core Image Processing Pipeline Development

This milestone focuses on building the sophisticated computer vision backbone that enables the AI to understand room geometry, architectural elements, and spatial relationships. The system employs advanced segmentation and depth estimation techniques to create a comprehensive analysis of uploaded room images, forming the foundation for structure-preserving design transformations.

Activity 2.1: Semantic Segmentation Implementation

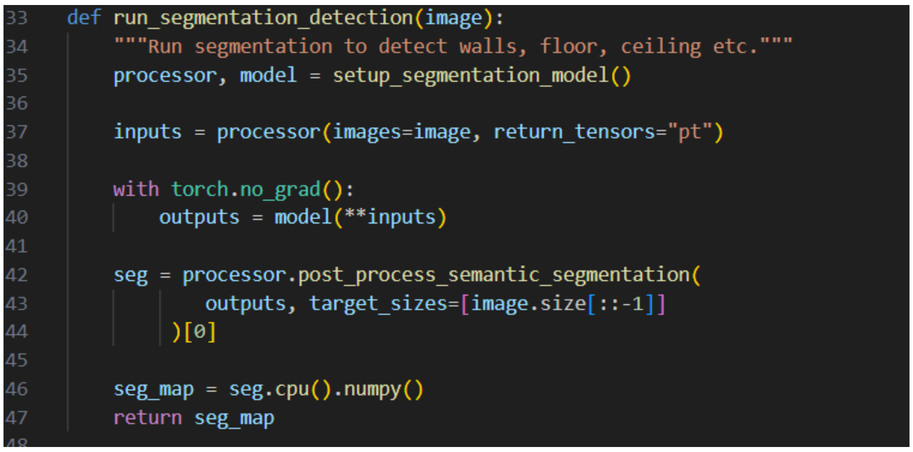
- Implement room element detection using Mask2Former model



```python
def setup_segmentation_model():
    """Setup segmentation model for room element detection"""
    processor = Mask2FormerImageProcessor.from_pretrained(SEGMENTATION_MODEL_NAME)
    model = Mask2FormerForUniversalSegmentation.from_pretrained(SEGMENTATION_MODEL_NAME)
    return processor, model
```

Figure 14: Segmentation model

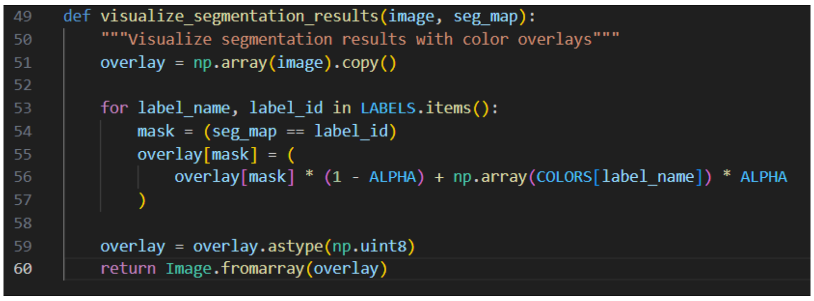
- Configure segmentation for interior spaces



```python
def run_segmentation_detection(image):
    """Run segmentation to detect walls, floor, ceiling etc."""
    processor, model = setup_segmentation_model()

    inputs = processor(images=image, return_tensors="pt")

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

    seg = processor.post_process_semantic_segmentation(
            outputs, target_sizes=[image.size[::-1]]
        )[0]

    seg_map = seg.cpu().numpy()
    return seg_map
```

Figure 15: Segmentation of image

- Develop visualization system with color-coded overlays:

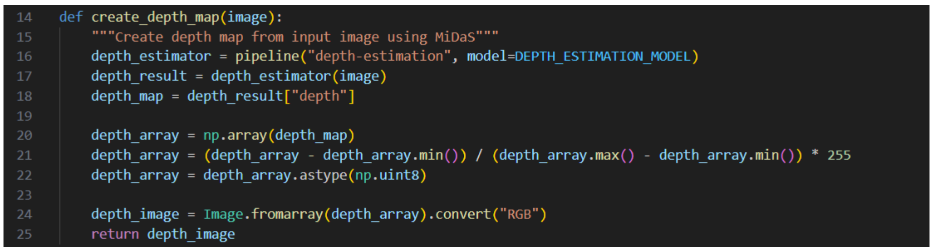


```python
def visualize_segmentation_results(image, seg_map):
    """Visualize segmentation results with color overlays"""
    overlay = np.array(image).copy()

    for label_name, label_id in LABELS.items():
        mask = (seg_map == label_id)
        overlay[mask] = (
            overlay[mask] * (1 - ALPHA) + np.array(COLORS[label_name]) * ALPHA
        )

    overlay = overlay.astype(np.uint8)
    return Image.fromarray(overlay)
```

Figure 16: Visualization of segmentation image

Activity 2.2: 3D Depth Mapping System

- Implement MiDaS depth estimation pipeline



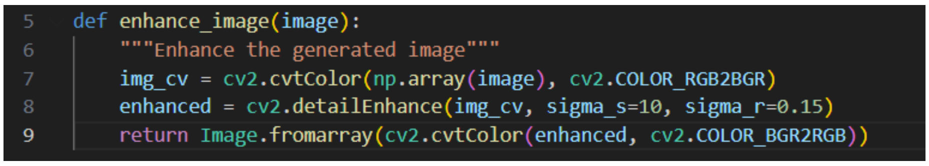
```python
def create_depth_map(image):
    """Create depth map from input image using MiDaS"""
    depth_estimator = pipeline("depth-estimation", model=DEPTH_ESTIMATION_MODEL)
    depth_result = depth_estimator(image)
    depth_map = depth_result["depth"]

    depth_array = np.array(depth_map)
    depth_array = (depth_array - depth_array.min()) / (depth_array.max() - depth_array.min()) * 255
    depth_array = depth_array.astype(np.uint8)

    depth_image = Image.fromarray(depth_array).convert("RGB")
    return depth_image
```

Figure 17: Depth with ControlNet Model

- Develop depth-aware room analysis for spatial understanding
- Create depth normalization techniques for consistent ControlNet conditioning

Activity 2.3: Image Preprocessing and Optimization

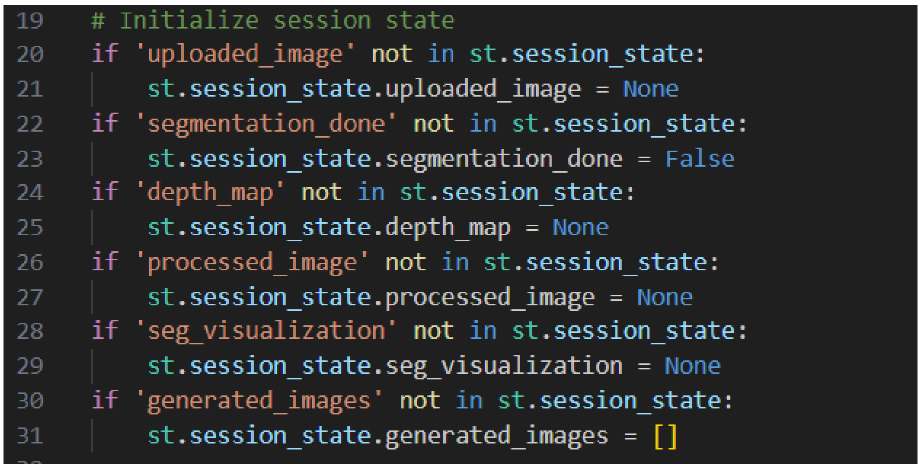
- Implement image standardization pipeline:



```python
def enhance_image(image):
    """Enhance the generated image"""
    img_cv = cv2.cvtColor(np.array(image), cv2.COLOR_RGB2BGR)
    enhanced = cv2.detailEnhance(img_cv, sigma_s=10, sigma_r=0.15)
    return Image.fromarray(cv2.cvtColor(enhanced, cv2.COLOR_BGR2RGB))
```

Figure 18: Enhance Image Function

- Optimize image loading and caching for Streamlit session management



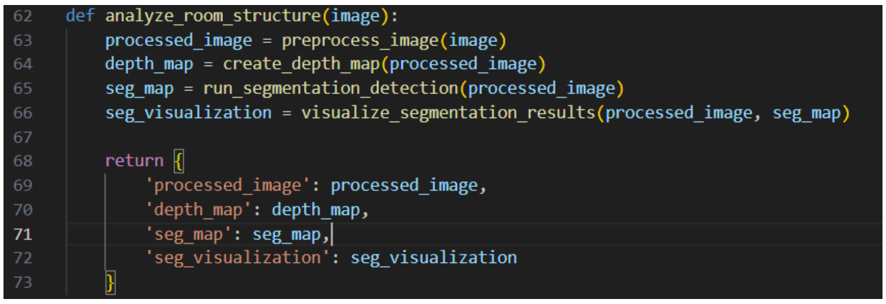
```python
# Initialize session state
if 'uploaded_image' not in st.session_state:
    st.session_state.uploaded_image = None
if 'segmentation_done' not in st.session_state:
    st.session_state.segmentation_done = False
if 'depth_map' not in st.session_state:
    st.session_state.depth_map = None
if 'processed_image' not in st.session_state:
    st.session_state.processed_image = None
if 'seg_visualization' not in st.session_state:
    st.session_state.seg_visualization = None
if 'generated_images' not in st.session_state:
    st.session_state.generated_images = []
```

Figure 19: Sessions States

- Implement memory-efficient processing for large room images

Activity 2.4: Room Structure Analysis Engine

- Create comprehensive room analysis combining segmentation and depth data:



```python
def analyze_room_structure(image):
    processed_image = preprocess_image(image)
    depth_map = create_depth_map(processed_image)
    seg_map = run_segmentation_detection(processed_image)
    seg_visualization = visualize_segmentation_results(processed_image, seg_map)

    return {
        'processed_image': processed_image,
        'depth_map': depth_map,
        'seg_map': seg_map,
        'seg_visualization': seg_visualization
    }
```

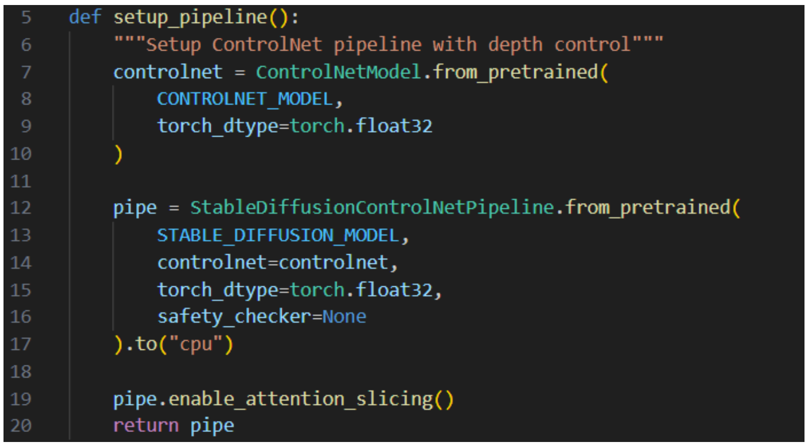
- Implement architectural element preservation logic for design generation

MILESTONE 3: AI-Powered Design Generation Engine

This milestone implements the core generative AI capabilities that transform analyzed room structures into stunning interior designs. The system leverages Stable Diffusion with ControlNet to ensure architectural integrity while applying diverse design styles, color schemes, and furniture arrangements based on user preferences.

Activity 3.1: ControlNet Pipeline Configuration

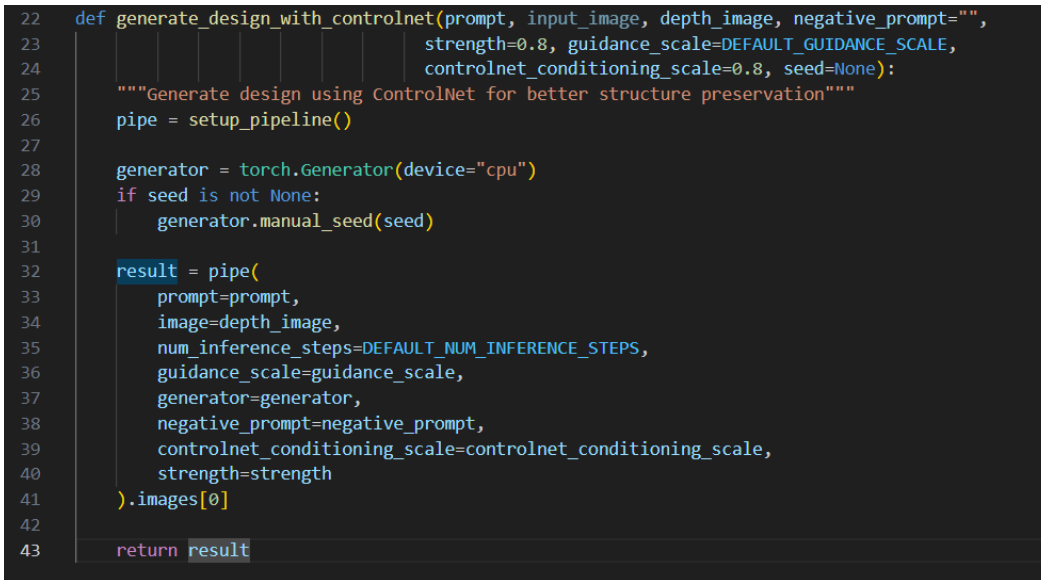
- Implement Stable Diffusion with ControlNet integration:



```python
def setup_pipeline():
    """Setup ControlNet pipeline with depth control"""
    controlnet = ControlNetModel.from_pretrained(
        CONTROLNET_MODEL,
        torch_dtype=torch.float32
    )

    pipe = StableDiffusionControlNetPipeline.from_pretrained(
        STABLE_DIFFUSION_MODEL,
        controlnet=controlnet,
        torch_dtype=torch.float32,
        safety_checker=None
    ).to("cpu")

    pipe.enable_attention_slicing()
    return pipe
```

Figure 20: Pipeline for Stable Diffuser & ControlNet model

- Generating the image with the pipeline setup:

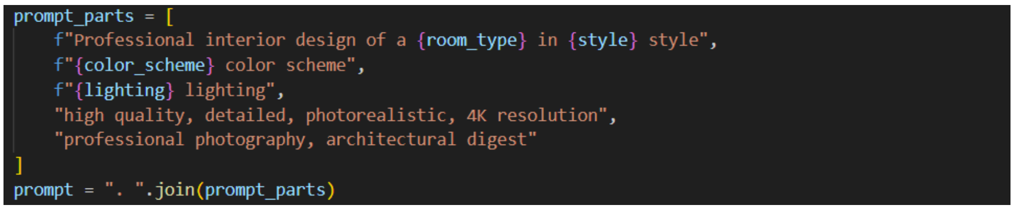


```python
def generate_design_with_controlnet(prompt, input_image, depth_image, negative_prompt="",
                                    strength=0.8, guidance_scale=DEFAULT_GUIDANCE_SCALE,
                                    controlnet_conditioning_scale=0.8, seed=None):
    """Generate design using ControlNet for better structure preservation"""
    pipe = setup_pipeline()

    generator = torch.Generator(device="cpu")
    if seed is not None:
        generator.manual_seed(seed)

    result = pipe(
        prompt=prompt,
        image=depth_image,
        num_inference_steps=DEFAULT_NUM_INFERENCE_STEPS,
        guidance_scale=guidance_scale,
        generator=generator,
        negative_prompt=negative_prompt,
        controlnet_conditioning_scale=controlnet_conditioning_scale,
        strength=strength
    ).images[0]

    return result
```

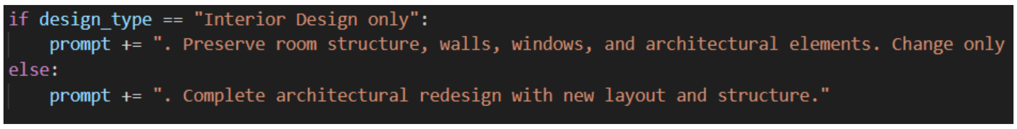
Figure 21: Generating Image with pipeline

Activity 3.2: Prompt Engineering for Interior Design

- Develop dynamic prompt generation system:



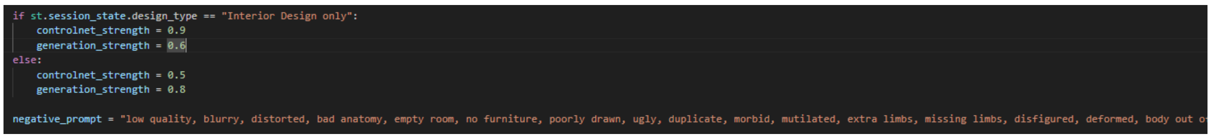
```python
prompt_parts = [
    f"Professional interior design of a {room_type} in {style} style",
    f"{color_scheme} color scheme",
    f"{lighting} lighting",
    "high quality, detailed, photorealistic, 4K resolution",
    "professional photography, architectural digest"
]
prompt = ". ".join(prompt_parts)
```



```python
if design_type == "Interior Design only":
    prompt += ". Preserve room structure, walls, windows, and architectural elements. Change only
else:
    prompt += ". Complete architectural redesign with new layout and structure."
```

Figure 22: Dynamic Prompting

- Implement negative prompt system to avoid common artifacts:

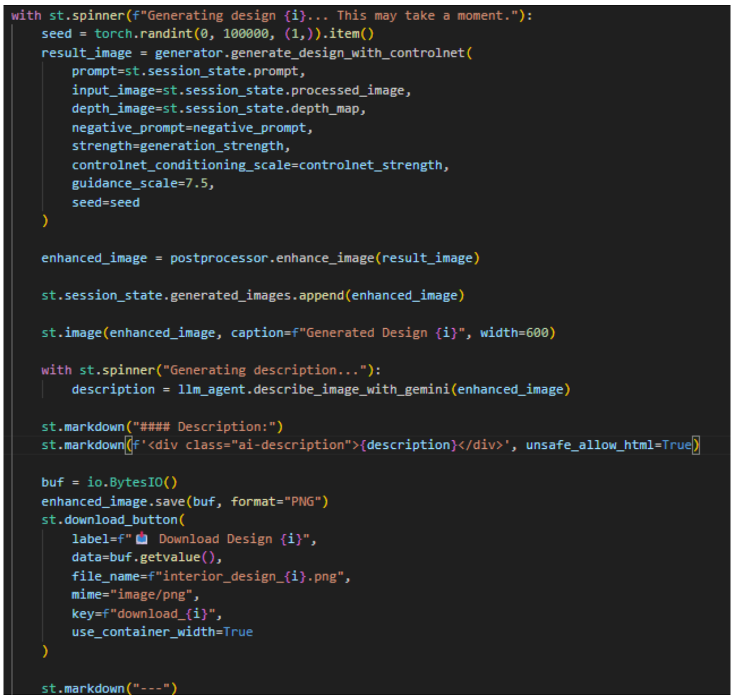


```python
if st.session_state.design_type == "Interior Design only":
    controlnet_strength = 0.9
    generation_strength = 0.6
else:
    controlnet_strength = 0.5
    generation_strength = 0.8

negative_prompt = "low quality, blurry, distorted, bad anatomy, empty room, no furniture, poorly drawn, ugly, duplicate, morbid, mutilated, extra limbs, missing limbs, disfigured, deformed, body out o
```

Figure 23: Negative Prompting

Activity 3.4: Multi-Variant Generation System

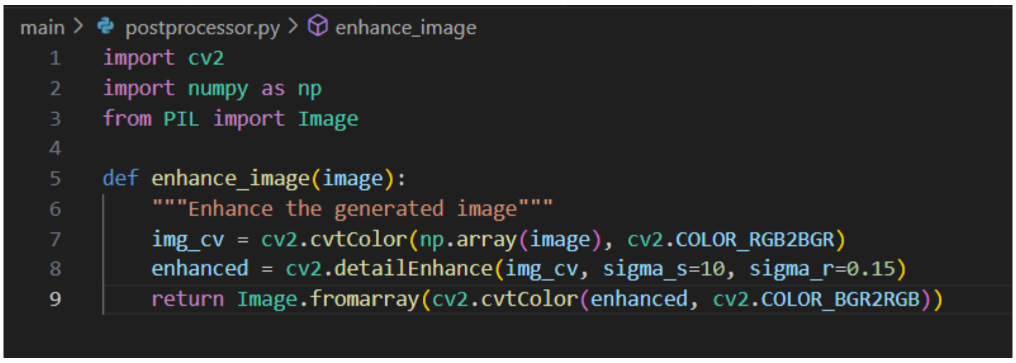
- Implement batch generation with seed control:



```python
with st.spinner(f"Generating design {i}... This may take a moment."):
    seed = torch.randint(0, 100000, (1,)).item()
    result_image = generator.generate_design_with_controlnet(
        prompt=st.session_state.prompt,
        input_image=st.session_state.processed_image,
        depth_image=st.session_state.depth_map,
        negative_prompt=negative_prompt,
        strength=generation_strength,
        controlnet_conditioning_scale=controlnet_strength,
        guidance_scale=7.5,
        seed=seed
    )

    enhanced_image = postprocessor.enhance_image(result_image)

    st.session_state.generated_images.append(enhanced_image)

    st.image(enhanced_image, caption=f"Generated Design {i}", width=600)

    with st.spinner("Generating description..."):
        description = llm_agent.describe_image_with_gemini(enhanced_image)

    st.markdown("#### Description:")
    st.markdown(f'<div class="ai-description">{description}</div>', unsafe_allow_html=True)

    buf = io.BytesIO()
    enhanced_image.save(buf, format="PNG")
    st.download_button(
        label=f"📥 Download Design {i}",
        data=buf.getvalue(),
        file_name=f"interior_design_{i}.png",
        mime="image/png",
        key=f"download_{i}",
        use_container_width=True
    )

    st.markdown("---")
```

Figure 24: Generating images

- Develop quality enhancement pipeline:


main > postprocessor.py > enhance_image

```python
import cv2
import numpy as np
from PIL import Image

def enhance_image(image):
    """Enhance the generated image"""
    img_cv = cv2.cvtColor(np.array(image), cv2.COLOR_RGB2BGR)
    enhanced = cv2.detailEnhance(img_cv, sigma_s=10, sigma_r=0.15)
    return Image.fromarray(cv2.cvtColor(enhanced, cv2.COLOR_BGR2RGB))
```

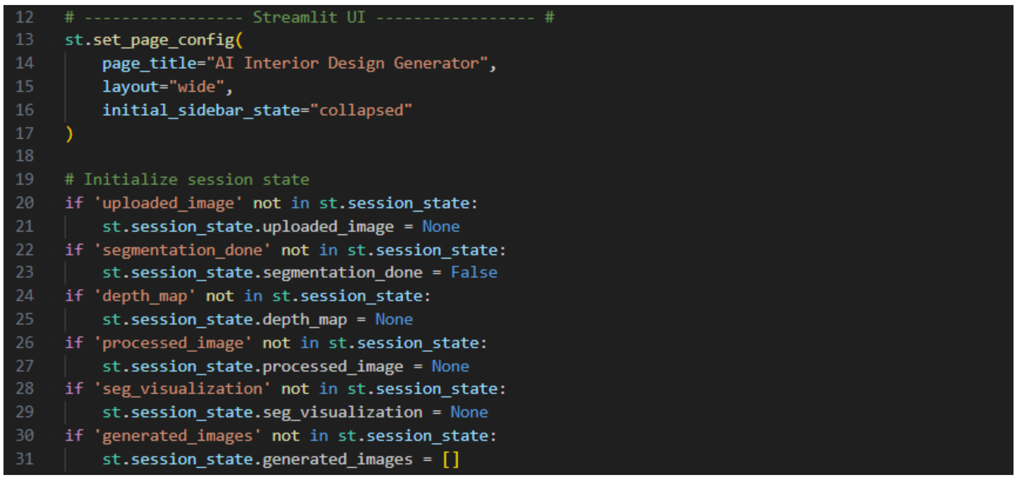
Figure 25: Enhance image function

MILESTONE 4: Streamlit UI Integration and Application Logic

This milestone focuses on creating an intuitive, professional user interface that seamlessly connects all AI components into a cohesive application. The Streamlit-based interface enables users to upload room images, configure design preferences, and view AI-generated transformations with real-time feedback and interactive controls.

Activity 4.1: Application Architecture and Session Management

- Implement comprehensive session state management:

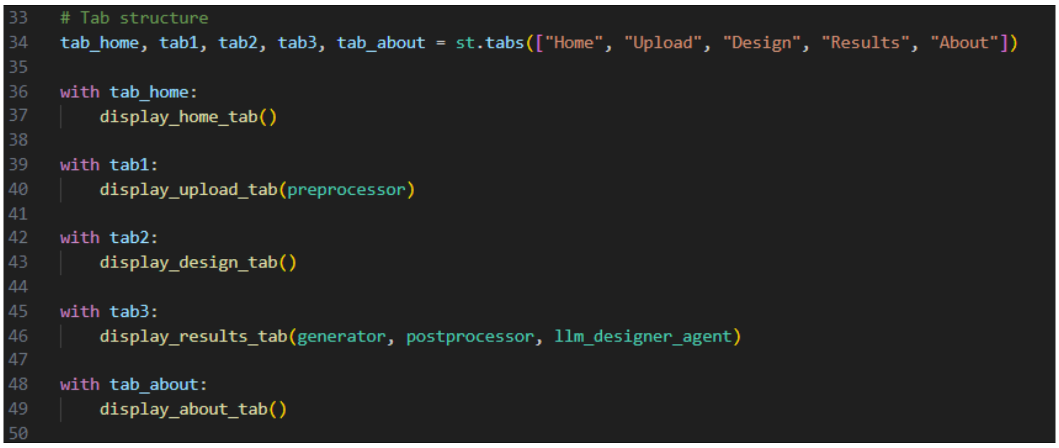


```python
# ---------------- Streamlit UI ---------------- #
st.set_page_config(
    page_title="AI Interior Design Generator",
    layout="wide",
    initial_sidebar_state="collapsed"
)

# Initialize session state
if 'uploaded_image' not in st.session_state:
    st.session_state.uploaded_image = None
if 'segmentation_done' not in st.session_state:
    st.session_state.segmentation_done = False
if 'depth_map' not in st.session_state:
    st.session_state.depth_map = None
if 'processed_image' not in st.session_state:
    st.session_state.processed_image = None
if 'seg_visualization' not in st.session_state:
    st.session_state.seg_visualization = None
if 'generated_images' not in st.session_state:
    st.session_state.generated_images = []
```

Figure 26: Session State

Activity 4.2: Multi-Tab Interface Design

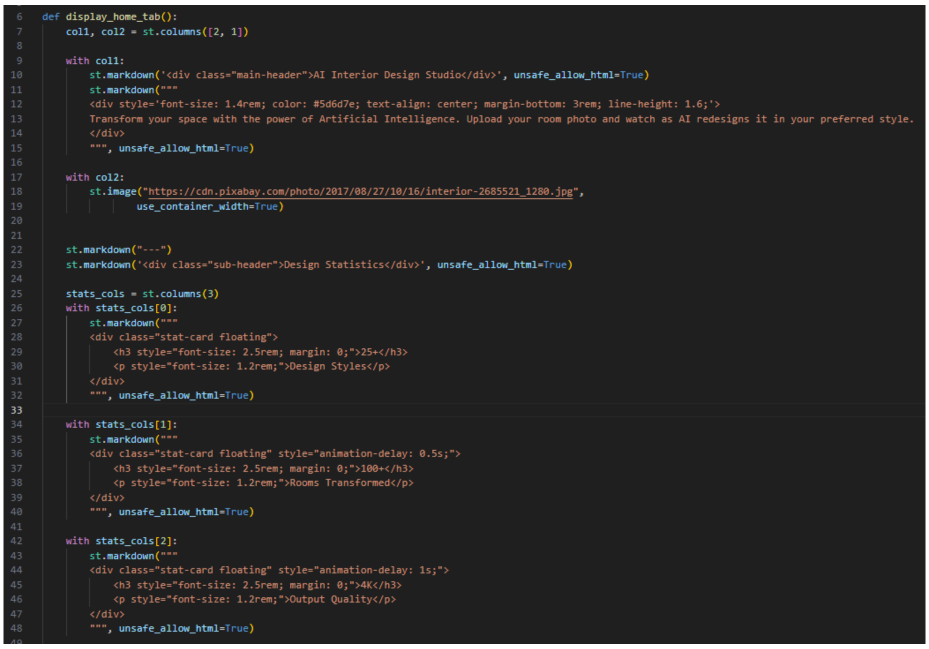
- Implement five-tab navigation structure:



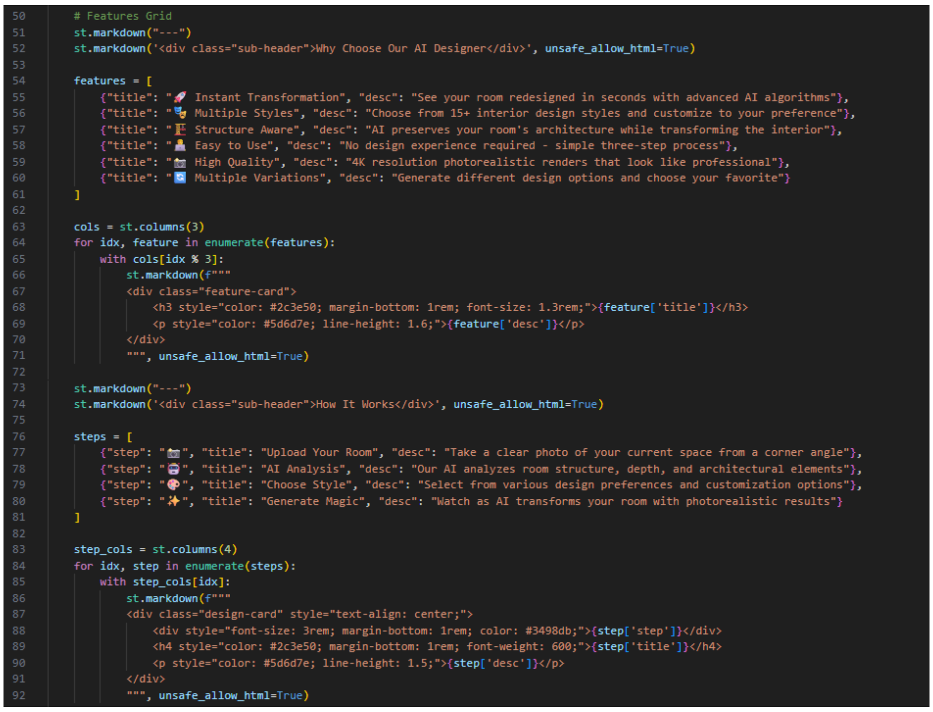
```python
# Tab structure
tab_home, tab1, tab2, tab3, tab_about = st.tabs(["Home", "Upload", "Design", "Results", "About"])

with tab_home:
    display_home_tab()

with tab1:
    display_upload_tab(preprocessor)

with tab2:
    display_design_tab()

with tab3:
    display_results_tab(generator, postprocessor, llm_designer_agent)

with tab_about:
    display_about_tab()

```

Figure 27: Tabs

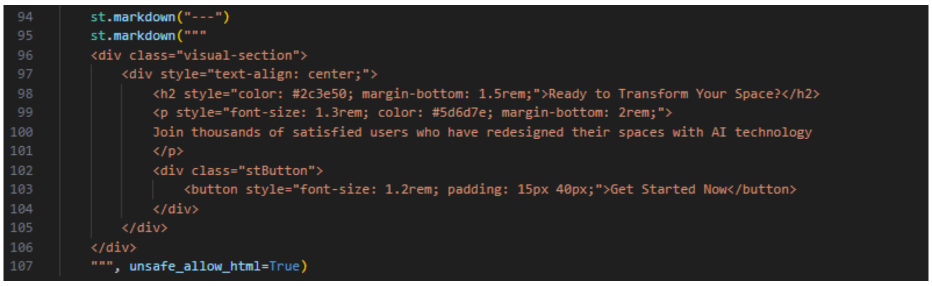
- Develop Home tab with project showcase:



```python
def display_home_tab():
    col1, col2 = st.columns([2, 1])

    with col1:
        st.markdown('<div class="main-header">AI Interior Design Studio</div>', unsafe_allow_html=True)
        st.markdown("""
        <div style='font-size: 1.4rem; color: #5d6d7e; text-align: center; margin-bottom: 3rem; line-height: 1.6;'>
        Transform your space with the power of Artificial Intelligence. Upload your room photo and watch as AI redesigns it in your preferred style.
        </div>
        """, unsafe_allow_html=True)

    with col2:
        st.image("https://cdn.pixabay.com/photo/2017/08/27/10/16/interior-2685521_1280.jpg",
                    use_container_width=True)


    st.markdown("---")
    st.markdown('<div class="sub-header">Design Statistics</div>', unsafe_allow_html=True)

    stats_cols = st.columns(3)
    with stats_cols[0]:
        st.markdown("""
        <div class="stat-card floating">
            <h3 style="font-size: 2.5rem; margin: 0;">25+</h3>
            <p style="font-size: 1.2rem;">Design Styles</p>
        </div>
        """, unsafe_allow_html=True)

    with stats_cols[1]:
        st.markdown("""
        <div class="stat-card floating" style="animation-delay: 0.5s;">
            <h3 style="font-size: 2.5rem; margin: 0;">100+</h3>
            <p style="font-size: 1.2rem;">Rooms Transformed</p>
        </div>
        """, unsafe_allow_html=True)

    with stats_cols[2]:
        st.markdown("""
        <div class="stat-card floating" style="animation-delay: 1s;">
            <h3 style="font-size: 2.5rem; margin: 0;">4K</h3>
            <p style="font-size: 1.2rem;">Output Quality</p>
        </div>
        """, unsafe_allow_html=True)
```



```python
    # Features Grid
    st.markdown("---")
    st.markdown('<div class="sub-header">Why Choose Our AI Designer</div>', unsafe_allow_html=True)

    features = [
        {"title": "🚀 Instant Transformation", "desc": "See your room redesigned in seconds with advanced AI algorithms"},
        {"title": "🎭 Multiple Styles", "desc": "Choose from 15+ interior design styles and customize to your preference"},
        {"title": "🏗 Structure Aware", "desc": "AI preserves your room's architecture while transforming the interior"},
        {"title": "👩‍💻 Easy to Use", "desc": "No design experience required - simple three-step process"},
        {"title": "📸 High Quality", "desc": "4K resolution photorealistic renders that look like professional"},
        {"title": "🔄 Multiple Variations", "desc": "Generate different design options and choose your favorite"}
    ]

    cols = st.columns(3)
    for idx, feature in enumerate(features):
        with cols[idx % 3]:
            st.markdown(f"""
            <div class="feature-card">
                <h3 style="color: #2c3e50; margin-bottom: 1rem; font-size: 1.3rem;">{feature['title']}</h3>
                <p style="color: #5d6d7e; line-height: 1.6;">{feature['desc']}</p>
            </div>
            """, unsafe_allow_html=True)

    st.markdown("---")
    st.markdown('<div class="sub-header">How It Works</div>', unsafe_allow_html=True)

    steps = [
        {"step": "📸", "title": "Upload Your Room", "desc": "Take a clear photo of your current space from a corner angle"},
        {"step": "🤖", "title": "AI Analysis", "desc": "Our AI analyzes room structure, depth, and architectural elements"},
        {"step": "🎨", "title": "Choose Style", "desc": "Select from various design preferences and customization options"},
        {"step": "✨", "title": "Generate Magic", "desc": "Watch as AI transforms your room with photorealistic results"}
    ]

    step_cols = st.columns(4)
    for idx, step in enumerate(steps):
        with step_cols[idx]:
            st.markdown(f"""
            <div class="design-card" style="text-align: center;">
                <div style="font-size: 3rem; margin-bottom: 1rem; color: #3498db;">{step['step']}</div>
                <h4 style="color: #2c3e50; margin-bottom: 1rem; font-weight: 600;">{step['title']}</h4>
                <p style="color: #5d6d7e; line-height: 1.5;">{step['desc']}</p>
            </div>
            """, unsafe_allow_html=True)
```

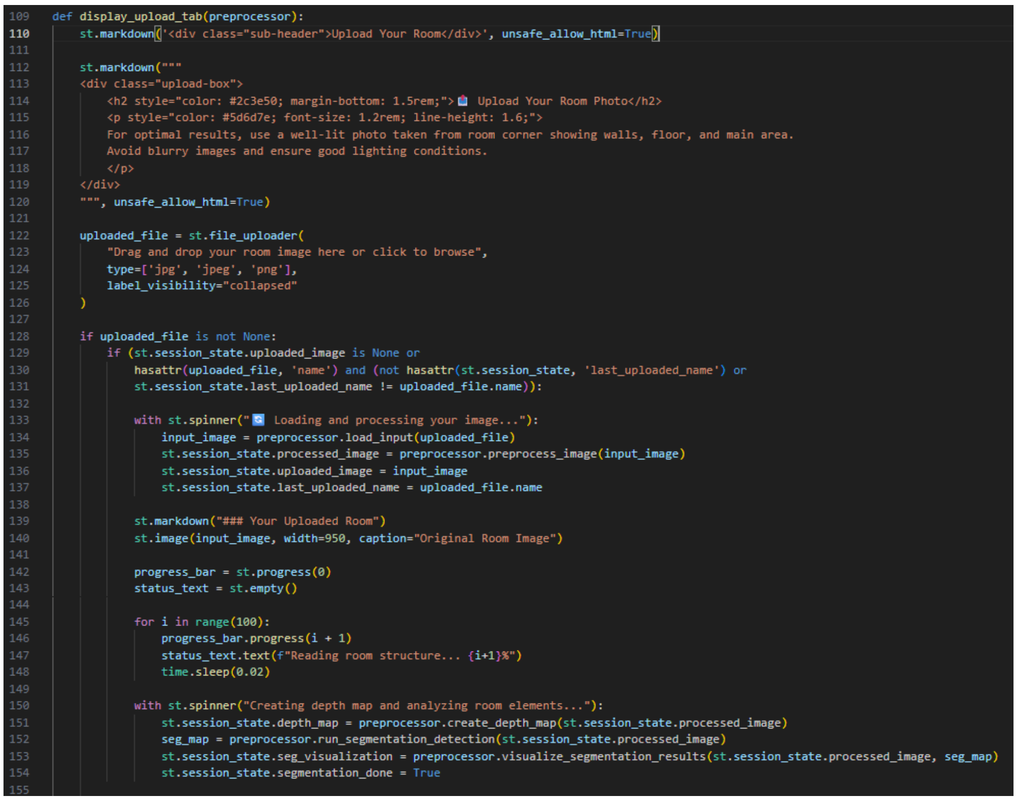


```python
    st.markdown("---")
    st.markdown("""
    <div class="visual-section">
        <div style="text-align: center;">
            <h2 style="color: #2c3e50; margin-bottom: 1.5rem;">Ready to Transform Your Space?</h2>
            <p style="font-size: 1.3rem; color: #5d6d7e; margin-bottom: 2rem;">
            Join thousands of satisfied users who have redesigned their spaces with AI technology
            </p>
            <div class="stButton">
                <button style="font-size: 1.2rem; padding: 15px 40px;">Get Started Now</button>
            </div>
        </div>
    </div>
    """, unsafe_allow_html=True)
```

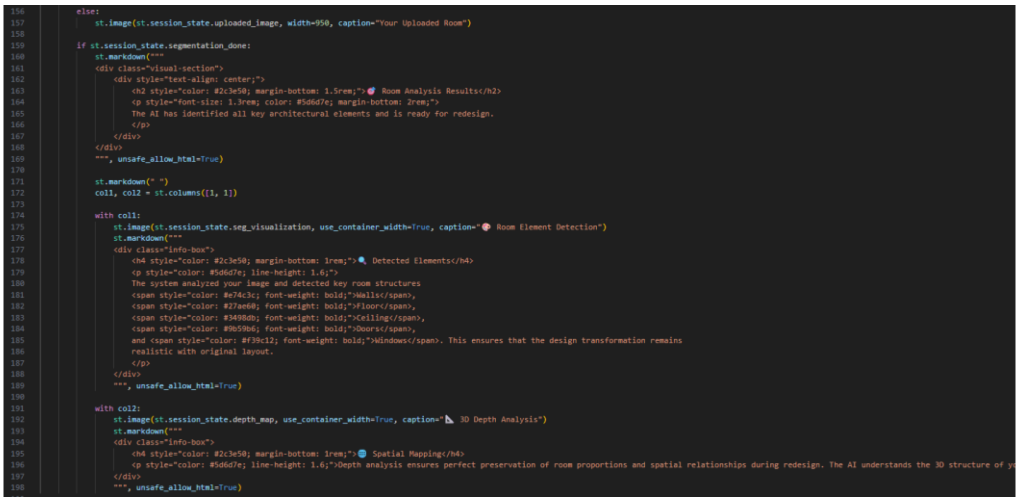
Figure 28: Home Tab

Activity 4.3: Image Upload and Processing Workflow

- Develop the Input Tab for User Interaction.



```python
def display_upload_tab(preprocessor):
    st.markdown('<div class="sub-header">Upload Your Room</div>', unsafe_allow_html=True)

    st.markdown("""
    <div class="upload-box">
        <h2 style="color: #2c3e50; margin-bottom: 1.5rem;">📤 Upload Your Room Photo</h2>
        <p style="color: #5d6d7e; font-size: 1.2rem; line-height: 1.6;">
        For optimal results, use a well-lit photo taken from room corner showing walls, floor, and main area.
        Avoid blurry images and ensure good lighting conditions.
        </p>
    </div>
    """, unsafe_allow_html=True)

    uploaded_file = st.file_uploader(
        "Drag and drop your room image here or click to browse",
        type=['jpg', 'jpeg', 'png'],
        label_visibility="collapsed"
    )

    if uploaded_file is not None:
        if (st.session_state.uploaded_image is None or
            hasattr(uploaded_file, 'name') and (not hasattr(st.session_state, 'last_uploaded_name') or
            st.session_state.last_uploaded_name != uploaded_file.name)):

            with st.spinner("🔄 Loading and processing your image..."):
                input_image = preprocessor.load_input(uploaded_file)
                st.session_state.processed_image = preprocessor.preprocess_image(input_image)
                st.session_state.uploaded_image = input_image
                st.session_state.last_uploaded_name = uploaded_file.name

            st.markdown("### Your Uploaded Room")
            st.image(input_image, width=950, caption="Original Room Image")

            progress_bar = st.progress(0)
            status_text = st.empty()

            for i in range(100):
                progress_bar.progress(i + 1)
                status_text.text(f"Reading room structure... {i+1}%")
                time.sleep(0.02)

            with st.spinner("Creating depth map and analyzing room elements..."):
                st.session_state.depth_map = preprocessor.create_depth_map(st.session_state.processed_image)
                seg_map = preprocessor.run_segmentation_detection(st.session_state.processed_image)
                st.session_state.seg_visualization = preprocessor.visualize_segmentation_results(st.session_state.processed_image, seg_map)
                st.session_state.segmentation_done = True

```



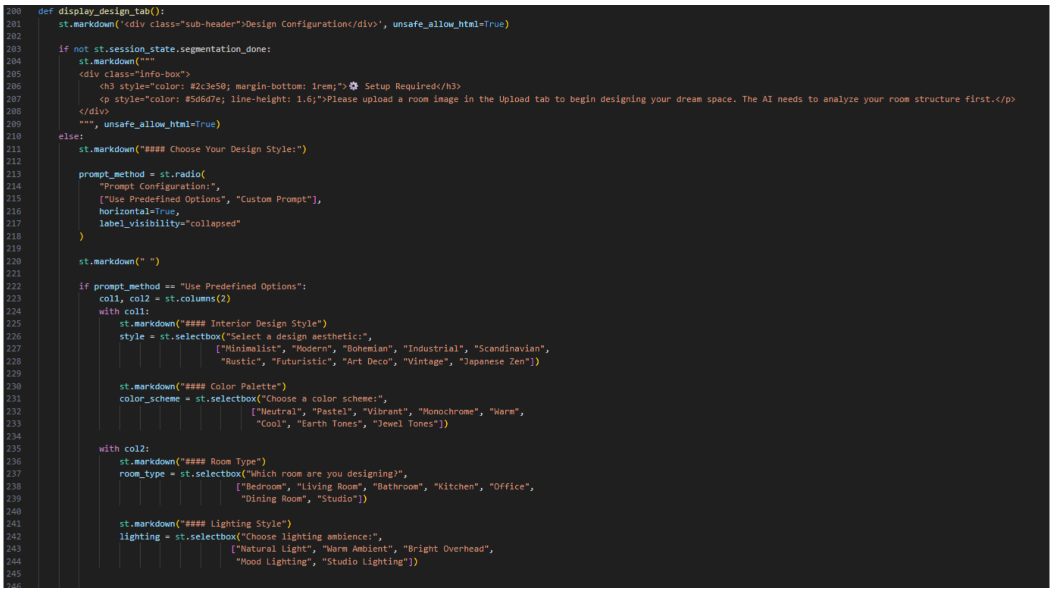
```python
        else:
            st.image(st.session_state.uploaded_image, width=950, caption="Your Uploaded Room")

        if st.session_state.segmentation_done:
            st.markdown("""
            <div class="visual-section">
                <div style="text-align: center;">
                    <h2 style="color: #2c3e50; margin-bottom: 1.5rem;">🎯 Room Analysis Results</h2>
                    <p style="font-size: 1.3rem; color: #5d6d7e; margin-bottom: 2rem;">
                    The AI has identified all key architectural elements and is ready for redesign.
                    </p>
                </div>
            </div>
            """, unsafe_allow_html=True)

            st.markdown(" ")
            col1, col2 = st.columns([1, 1])

            with col1:
                st.image(st.session_state.seg_visualization, use_container_width=True, caption="🎨 Room Element Detection")
                st.markdown("""
                <div class="info-box">
                    <h4 style="color: #2c3e50; margin-bottom: 1rem;">🔍 Detected Elements</h4>
                    <p style="color: #5d6d7e; line-height: 1.6;">
                    The system analyzed your image and detected key room structures
                    <span style="color: #e74c3c; font-weight: bold;">Walls</span>,
                    <span style="color: #27ae60; font-weight: bold;">Floor</span>,
                    <span style="color: #3498db; font-weight: bold;">Ceiling</span>,
                    <span style="color: #9b59b6; font-weight: bold;">Doors</span>,
                    and <span style="color: #f39c12; font-weight: bold;">Windows</span>. This ensures that the design transformation remains
                    realistic with original layout.
                    </p>
                </div>
                """, unsafe_allow_html=True)

            with col2:
                st.image(st.session_state.depth_map, use_container_width=True, caption="📐 3D Depth Analysis")
                st.markdown("""
                <div class="info-box">
                    <h4 style="color: #2c3e50; margin-bottom: 1rem;">🌐 Spatial Mapping</h4>
                    <p style="color: #5d6d7e; line-height: 1.6;">Depth analysis ensures perfect preservation of room proportions and spatial relationships during redesign. The AI understands the 3D structure of yo
                </div>
                """, unsafe_allow_html=True)
```

Figure 29: Input Tab

- Implement drag-and-drop upload system.
- Create real-time processing feedback with progress indicators.

Activity 4.4: Design Configuration Interface

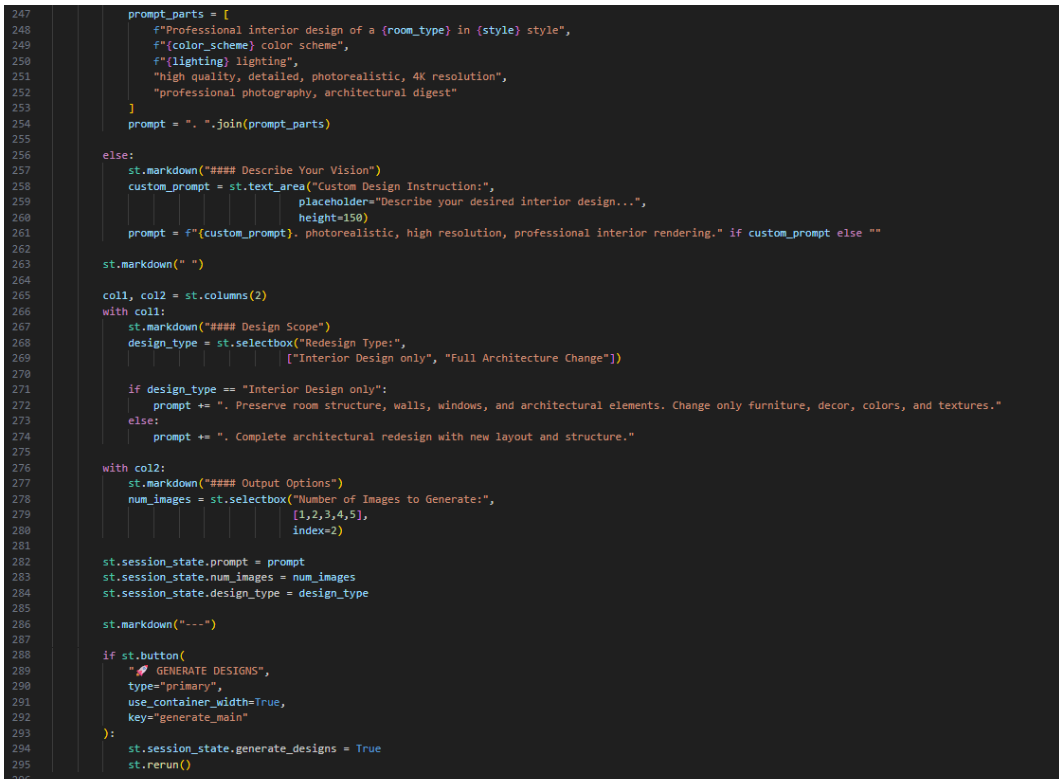
- Implement design preference selection system:



```python
def display_design_tab():
    st.markdown('<div class="sub-header">Design Configuration</div>', unsafe_allow_html=True)

    if not st.session_state.segmentation_done:
        st.markdown("""
        <div class="info-box">
            <h3 style="color: #2c3e50; margin-bottom: 1rem;">⚙️ Setup Required</h3>
            <p style="color: #5d6d7e; line-height: 1.6;">Please upload a room image in the Upload tab to begin designing your dream space. The AI needs to analyze your room structure first.</p>
        </div>
        """, unsafe_allow_html=True)
    else:
        st.markdown("#### Choose Your Design Style:")

        prompt_method = st.radio(
            "Prompt Configuration:",
            ["Use Predefined Options", "Custom Prompt"],
            horizontal=True,
            label_visibility="collapsed"
        )

        st.markdown(" ")

        if prompt_method == "Use Predefined Options":
            col1, col2 = st.columns(2)
            with col1:
                st.markdown("#### Interior Design Style")
                style = st.selectbox("Select a design aesthetic:",
                                    ["Minimalist", "Modern", "Bohemian", "Industrial", "Scandinavian",
                                     "Rustic", "Futuristic", "Art Deco", "Vintage", "Japanese Zen"])

                st.markdown("#### Color Palette")
                color_scheme = st.selectbox("Choose a color scheme:",
                                           ["Neutral", "Pastel", "Vibrant", "Monochrome", "Warm",
                                            "Cool", "Earth Tones", "Jewel Tones"])

            with col2:
                st.markdown("#### Room Type")
                room_type = st.selectbox("Which room are you designing?",
                                        ["Bedroom", "Living Room", "Bathroom", "Kitchen", "Office",
                                         "Dining Room", "Studio"])

                st.markdown("#### Lighting Style")
                lighting = st.selectbox("Choose lighting ambience:",
                                       ["Natural Light", "Warm Ambient", "Bright Overhead",
                                        "Mood Lighting", "Studio Lighting"])

```

Figure 30: Option Selection

- Develop custom prompt input for advanced users:

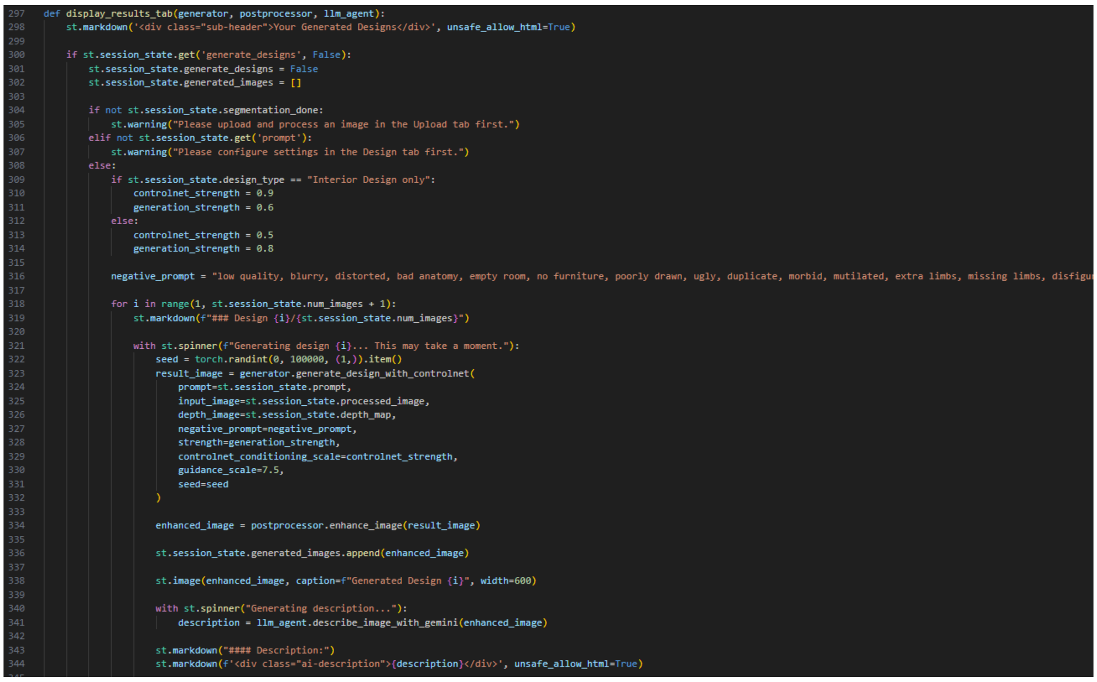


```python
            prompt_parts = [
                f"Professional interior design of a {room_type} in {style} style",
                f"{color_scheme} color scheme",
                f"{lighting} lighting",
                "high quality, detailed, photorealistic, 4K resolution",
                "professional photography, architectural digest"
            ]
            prompt = ". ".join(prompt_parts)

        else:
            st.markdown("#### Describe Your Vision")
            custom_prompt = st.text_area("Custom Design Instruction:",
                                       placeholder="Describe your desired interior design...",
                                       height=150)
            prompt = f"{custom_prompt}. photorealistic, high resolution, professional interior rendering." if custom_prompt else ""

        st.markdown(" ")

        col1, col2 = st.columns(2)
        with col1:
            st.markdown("#### Design Scope")
            design_type = st.selectbox("Redesign Type:",
                                     ["Interior Design only", "Full Architecture Change"])

            if design_type == "Interior Design only":
                prompt += ". Preserve room structure, walls, windows, and architectural elements. Change only furniture, decor, colors, and textures."
            else:
                prompt += ". Complete architectural redesign with new layout and structure."

        with col2:
            st.markdown("#### Output Options")
            num_images = st.selectbox("Number of Images to Generate:",
                                    [1,2,3,4,5],
                                    index=2)

        st.session_state.prompt = prompt
        st.session_state.num_images = num_images
        st.session_state.design_type = design_type

        st.markdown("---")

        if st.button(
            "🚀 GENERATE DESIGNS",
            type="primary",
            use_container_width=True,
            key="generate_main"
        ):
            st.session_state.generate_designs = True
            st.rerun()
```

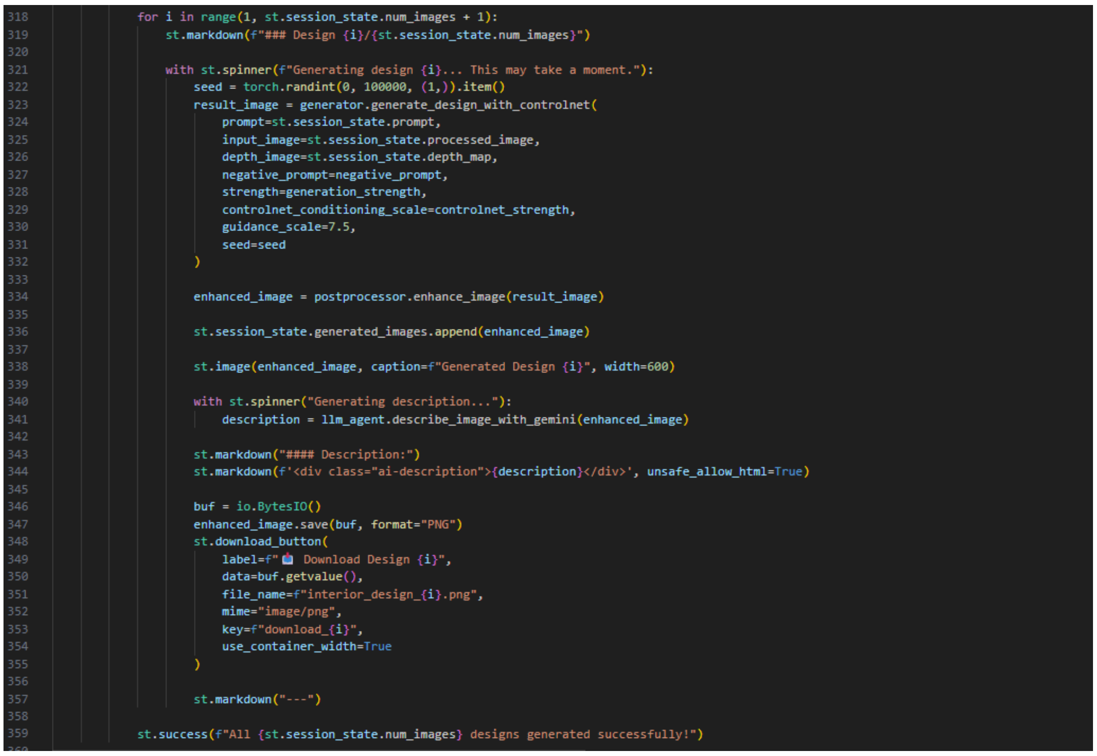
Figure 31: Custom Prompting

Activity 4.5: Results Display and Export System

- Implement generated images gallery:



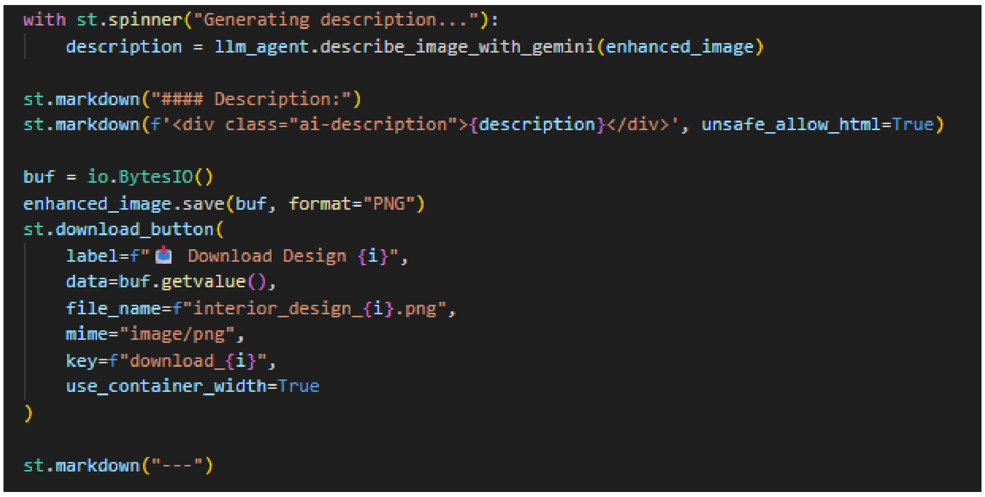
```python
def display_results_tab(generator, postprocessor, llm_agent):
    st.markdown('<div class="sub-header">Your Generated Designs</div>', unsafe_allow_html=True)

    if st.session_state.get('generate_designs', False):
        st.session_state.generate_designs = False
        st.session_state.generated_images = []

        if not st.session_state.segmentation_done:
            st.warning("Please upload and process an image in the Upload tab first.")
        elif not st.session_state.get('prompt'):
            st.warning("Please configure settings in the Design tab first.")
        else:
            if st.session_state.design_type == "Interior Design only":
                controlnet_strength = 0.9
                generation_strength = 0.6
            else:
                controlnet_strength = 0.5
                generation_strength = 0.8

            negative_prompt = "low quality, blurry, distorted, bad anatomy, empty room, no furniture, poorly drawn, ugly, duplicate, morbid, mutilated, extra limbs, missing limbs, disfigu

            for i in range(1, st.session_state.num_images + 1):
                st.markdown(f"### Design {i}/{st.session_state.num_images}")

                with st.spinner(f"Generating design {i}... This may take a moment."):
                    seed = torch.randint(0, 100000, (1,)).item()
                    result_image = generator.generate_design_with_controlnet(
                        prompt=st.session_state.prompt,
                        input_image=st.session_state.processed_image,
                        depth_image=st.session_state.depth_map,
                        negative_prompt=negative_prompt,
                        strength=generation_strength,
                        controlnet_conditioning_scale=controlnet_strength,
                        guidance_scale=7.5,
                        seed=seed
                    )

                    enhanced_image = postprocessor.enhance_image(result_image)

                    st.session_state.generated_images.append(enhanced_image)

                    st.image(enhanced_image, caption=f"Generated Design {i}", width=600)

                    with st.spinner("Generating description..."):
                        description = llm_agent.describe_image_with_gemini(enhanced_image)

                    st.markdown("#### Description:")
                    st.markdown(f'<div class="ai-description">{description}</div>', unsafe_allow_html=True)
```



```python
            for i in range(1, st.session_state.num_images + 1):
                st.markdown(f"### Design {i}/{st.session_state.num_images}")

                with st.spinner(f"Generating design {i}... This may take a moment."):
                    seed = torch.randint(0, 100000, (1,)).item()
                    result_image = generator.generate_design_with_controlnet(
                        prompt=st.session_state.prompt,
                        input_image=st.session_state.processed_image,
                        depth_image=st.session_state.depth_map,
                        negative_prompt=negative_prompt,
                        strength=generation_strength,
                        controlnet_conditioning_scale=controlnet_strength,
                        guidance_scale=7.5,
                        seed=seed
                    )

                    enhanced_image = postprocessor.enhance_image(result_image)

                    st.session_state.generated_images.append(enhanced_image)

                    st.image(enhanced_image, caption=f"Generated Design {i}", width=600)

                    with st.spinner("Generating description..."):
                        description = llm_agent.describe_image_with_gemini(enhanced_image)

                    st.markdown("#### Description:")
                    st.markdown(f'<div class="ai-description">{description}</div>', unsafe_allow_html=True)

                    buf = io.BytesIO()
                    enhanced_image.save(buf, format="PNG")
                    st.download_button(
                        label=f"📥 Download Design {i}",
                        data=buf.getvalue(),
                        file_name=f"interior_design_{i}.png",
                        mime="image/png",
                        key=f"download_{i}",
                        use_container_width=True
                    )

                    st.markdown("---")

            st.success(f"All {st.session_state.num_images} designs generated successfully!")
```

Figure 32: Output Section

- Create download functionality for generated designs:

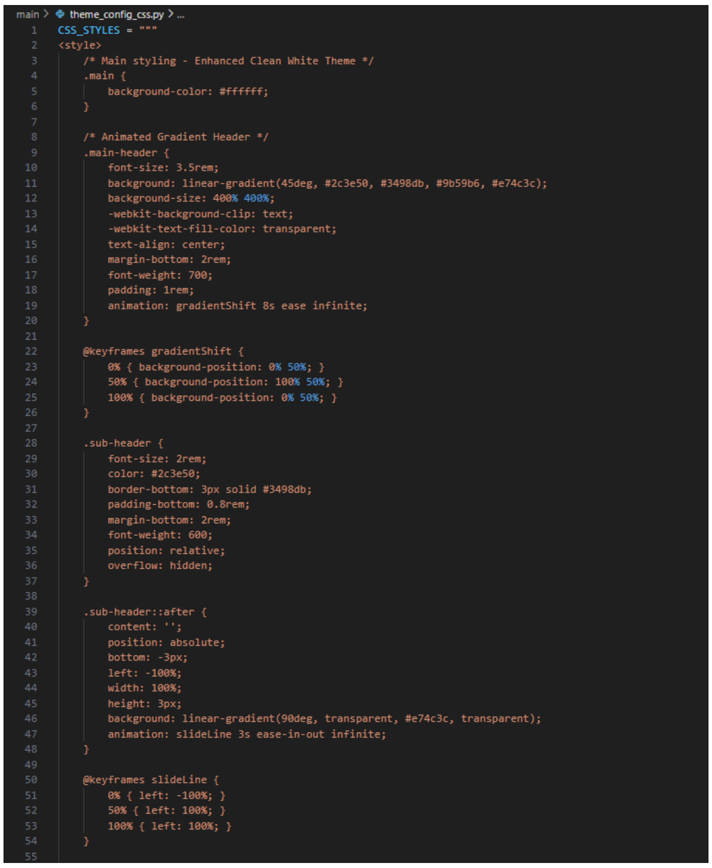


```python
with st.spinner("Generating description..."):
    description = llm_agent.describe_image_with_gemini(enhanced_image)

st.markdown("#### Description:")
st.markdown(f'<div class="ai-description">{description}</div>', unsafe_allow_html=True)

buf = io.BytesIO()
enhanced_image.save(buf, format="PNG")
st.download_button(
    label=f"📥 Download Design {i}",
    data=buf.getvalue(),
    file_name=f"interior_design_{i}.png",
    mime="image/png",
    key=f"download_{i}",
    use_container_width=True
)

st.markdown("---")
```

Figure 33: Download Button

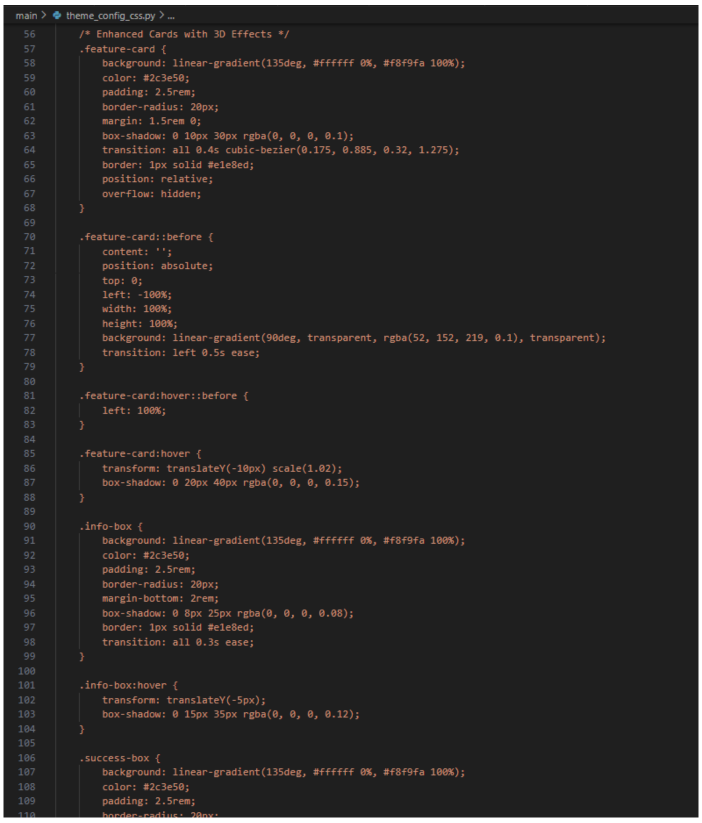
Activity 4.6: Professional UI Styling and Theming

- Implement comprehensive CSS styling system:



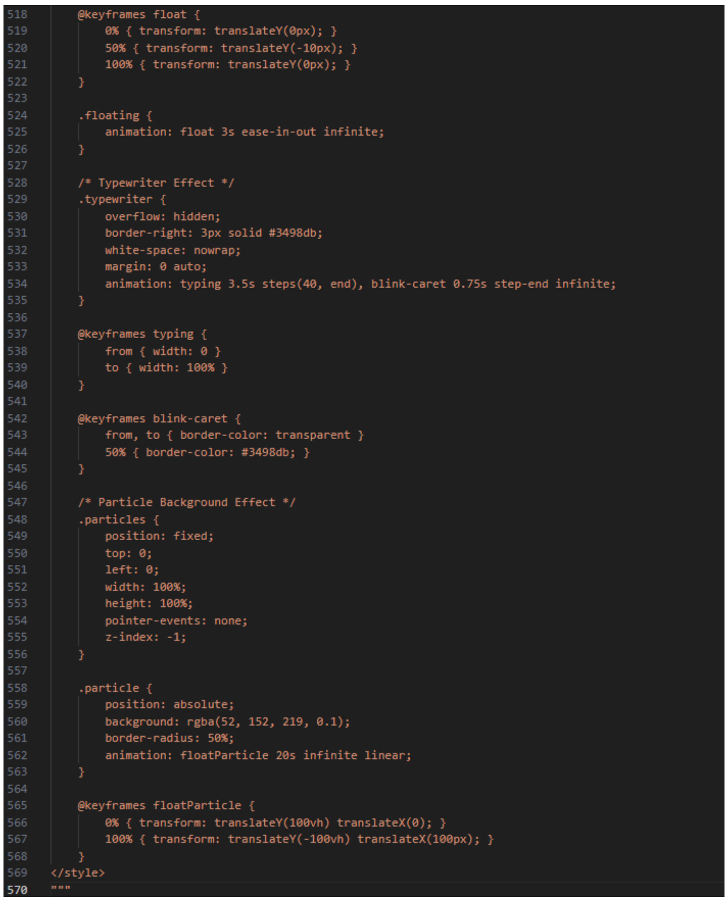
main > theme_config_css.py > ...

```
CSS_STYLES = """
<style>
    /* Main styling - Enhanced Clean White Theme */
    .main {
        background-color: #ffffff;
    }

    /* Animated Gradient Header */
    .main-header {
        font-size: 3.5rem;
        background: linear-gradient(45deg, #2c3e50, #3498db, #9b59b6, #e74c3c);
        background-size: 400% 400%;
        -webkit-background-clip: text;
        -webkit-text-fill-color: transparent;
        text-align: center;
        margin-bottom: 2rem;
        font-weight: 700;
        padding: 1rem;
        animation: gradientShift 8s ease infinite;
    }

    @keyframes gradientShift {
        0% { background-position: 0% 50%; }
        50% { background-position: 100% 50%; }
        100% { background-position: 0% 50%; }
    }

    .sub-header {
        font-size: 2rem;
        color: #2c3e50;
        border-bottom: 3px solid #3498db;
        padding-bottom: 0.8rem;
        margin-bottom: 2rem;
        font-weight: 600;
        position: relative;
        overflow: hidden;
    }

    .sub-header::after {
        content: '';
        position: absolute;
        bottom: -3px;
        left: -100%;
        width: 100%;
        height: 3px;
        background: linear-gradient(90deg, transparent, #e74c3c, transparent);
        animation: slideLine 3s ease-in-out infinite;
    }

    @keyframes slideLine {
        0% { left: -100%; }
        50% { left: 100%; }
        100% { left: 100%; }
    }

```



main > theme_config_css.py > ...

```
    /* Enhanced Cards with 3D Effects */
    .feature-card {
        background: linear-gradient(135deg, #ffffff 0%, #f8f9fa 100%);
        color: #2c3e50;
        padding: 2.5rem;
        border-radius: 20px;
        margin: 1.5rem 0;
        box-shadow: 0 10px 30px rgba(0, 0, 0, 0.1);
        transition: all 0.4s cubic-bezier(0.175, 0.885, 0.32, 1.275);
        border: 1px solid #e1e8ed;
        position: relative;
        overflow: hidden;
    }

    .feature-card::before {
        content: '';
        position: absolute;
        top: 0;
        left: -100%;
        width: 100%;
        height: 100%;
        background: linear-gradient(90deg, transparent, rgba(52, 152, 219, 0.1), transparent);
        transition: left 0.5s ease;
    }

    .feature-card:hover::before {
        left: 100%;
    }

    .feature-card:hover {
        transform: translateY(-10px) scale(1.02);
        box-shadow: 0 20px 40px rgba(0, 0, 0, 0.15);
    }

    .info-box {
        background: linear-gradient(135deg, #ffffff 0%, #f8f9fa 100%);
        color: #2c3e50;
        padding: 2.5rem;
        border-radius: 20px;
        margin-bottom: 2rem;
        box-shadow: 0 8px 25px rgba(0, 0, 0, 0.08);
        border: 1px solid #e1e8ed;
        transition: all 0.3s ease;
    }

    .info-box:hover {
        transform: translateY(-5px);
        box-shadow: 0 15px 35px rgba(0, 0, 0, 0.12);
    }

    .success-box {
        background: linear-gradient(135deg, #ffffff 0%, #f8f9fa 100%);
        color: #2c3e50;
        padding: 2.5rem;
        border-radius: 20px;
```



```
    @keyframes float {
        0% { transform: translateY(0px); }
        50% { transform: translateY(-10px); }
        100% { transform: translateY(0px); }
    }

    .floating {
        animation: float 3s ease-in-out infinite;
    }

    /* Typewriter Effect */
    .typewriter {
        overflow: hidden;
        border-right: 3px solid #3498db;
        white-space: nowrap;
        margin: 0 auto;
        animation: typing 3.5s steps(40, end), blink-caret 0.75s step-end infinite;
    }

    @keyframes typing {
        from { width: 0 }
        to { width: 100% }
    }

    @keyframes blink-caret {
        from, to { border-color: transparent }
        50% { border-color: #3498db; }
    }

    /* Particle Background Effect */
    .particles {
        position: fixed;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
        pointer-events: none;
        z-index: -1;
    }

    .particle {
        position: absolute;
        background: rgba(52, 152, 219, 0.1);
        border-radius: 50%;
        animation: floatParticle 20s infinite linear;
    }

    @keyframes floatParticle {
        0% { transform: translateY(100vh) translateX(0); }
        100% { transform: translateY(-100vh) translateX(100px); }
    }
</style>
"""
```

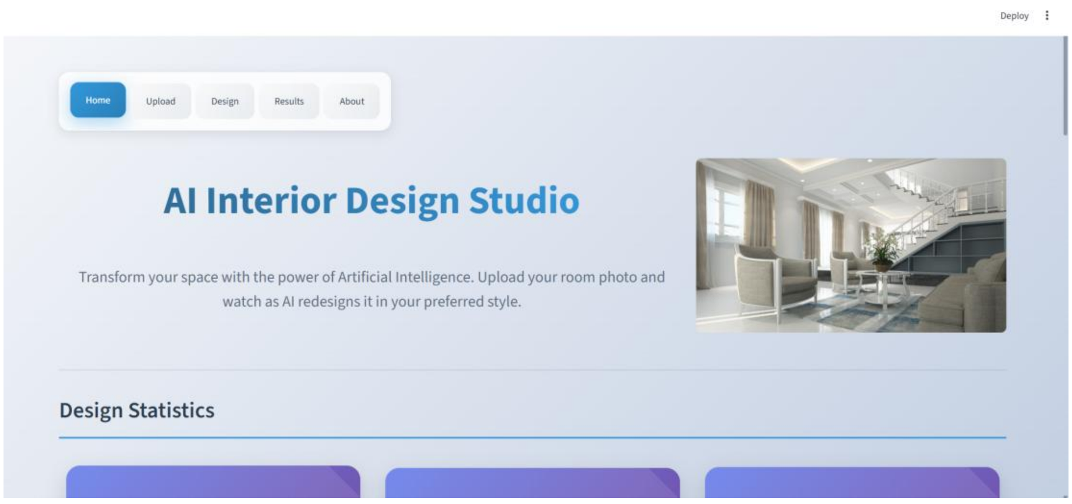
Figure 34: Custom CSS

MILESTONE 5: Testing, Optimization and Deployment

This final milestone ensures the AI Interior Design Studio delivers reliable, high-quality performance across diverse room types and design requirements. The focus is on validating output quality, optimizing generation speed, and preparing the application for production deployment with robust error handling and user experience refinements.

Activity 5.1: UI Layout & Multi-Page Navigation Setup

- Created Home Page with project overview and navigation button

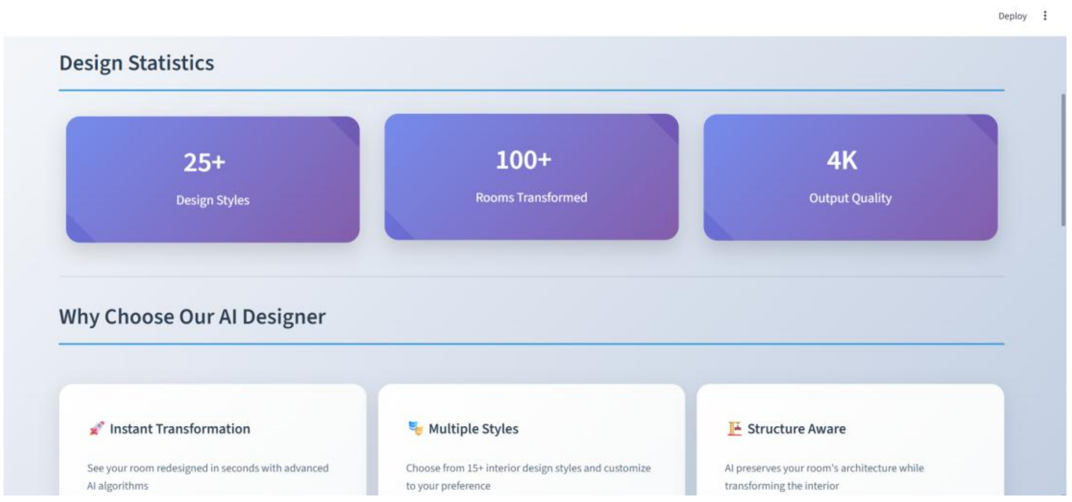


AI Interior Design Studio

Transform your space with the power of Artificial Intelligence. Upload your room photo and watch as AI redesigns it in your preferred style.

Design Statistics

Figure 35: Home Page Preview 1



Design Statistics

25+ Design Styles

100+ Rooms Transformed

4K Output Quality

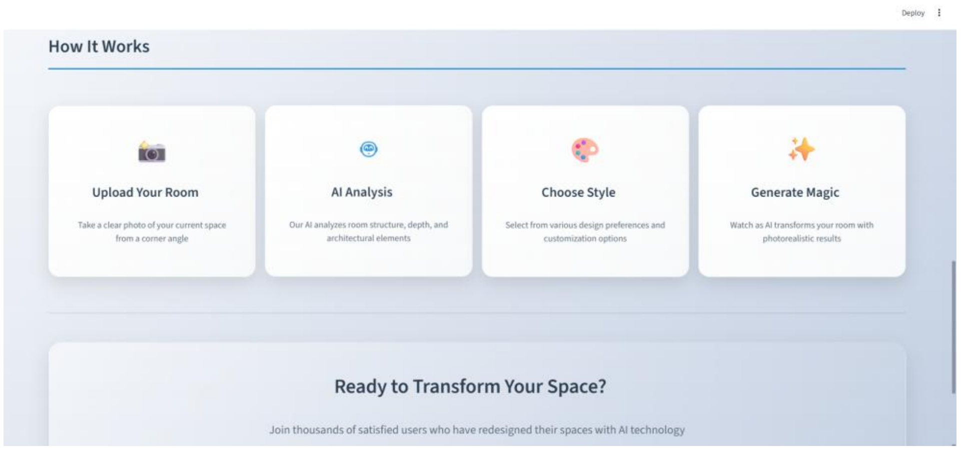
Why Choose Our AI Designer

Instant Transformation
See your room redesigned in seconds with advanced AI algorithms

Multiple Styles
Choose from 15+ interior design styles and customize to your preference

Structure Aware
AI preserves your room's architecture while transforming the interior

Figure 36: Home Page Preview 2



How It Works

Upload Your Room

Take a clear photo of your current space from a corner angle

AI Analysis

Our AI analyzes room structure, depth, and architectural elements

Choose Style

Select from various design preferences and customization options

Generate Magic

Watch as AI transforms your room with photorealistic results

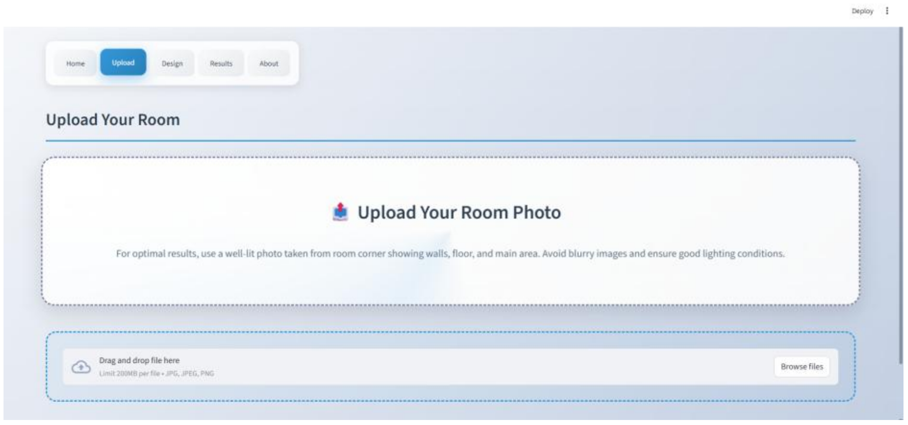
Ready to Transform Your Space?

Join thousands of satisfied users who have redesigned their spaces with AI technology

Figure 37: Home Page Preview 3

- Developed Input Page containing:

  – Image upload section



Upload Your Room

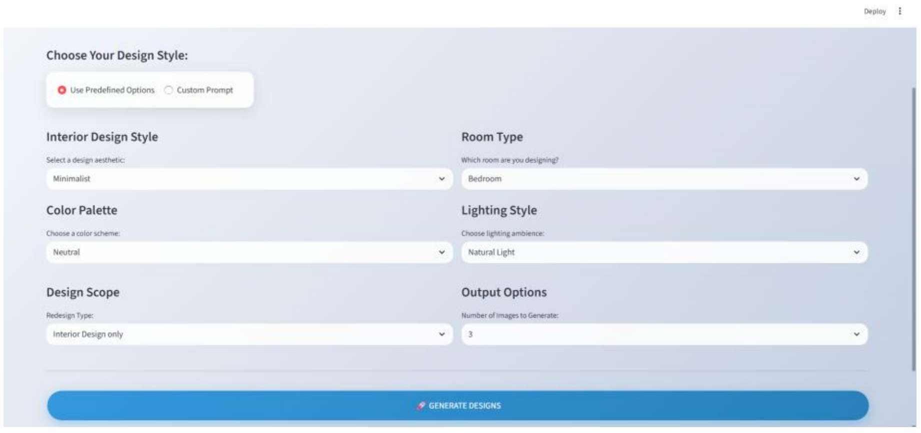
Upload Your Room Photo

For optimal results, use a well-lit photo taken from room corner showing walls, floor, and main area. Avoid blurry images and ensure good lighting conditions.

Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG

Figure 38: Image uploader

  – Dropdown options for room type, design style, colour scheme and lighting preference



Choose Your Design Style:

[✓ Use Predefined Options] Custom Prompt

Interior Design Style
Select a design aesthetic: Minimalist

Room Type
Which room are you designing? Bedroom

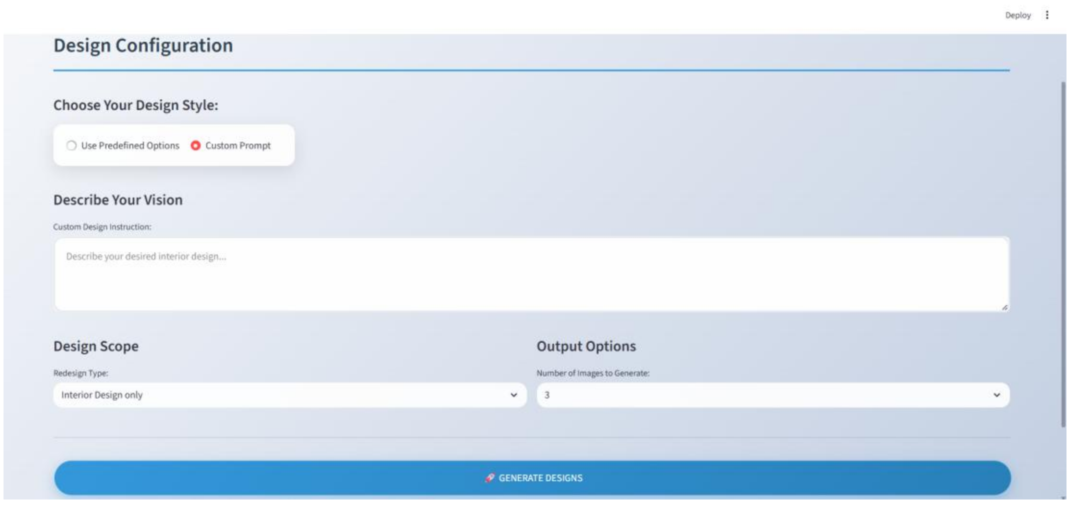
Color Palette
Choose a color scheme: Neutral

Lighting Style
Choose lighting ambience: Natural Light

Design Scope
Redesign Type: Interior Design only

Output Options
Number of Images to Generate: 3

Figure 39: Drop-down Option Selector



Deploy

Design Configuration

Choose Your Design Style:

Use Predefined Options [✓ Custom Prompt]

Describe Your Vision

Custom Design Instruction:

Describe your desired interior design...

Design Scope

Redesign Type:

Interior Design only

Output Options

Number of Images to Generate:

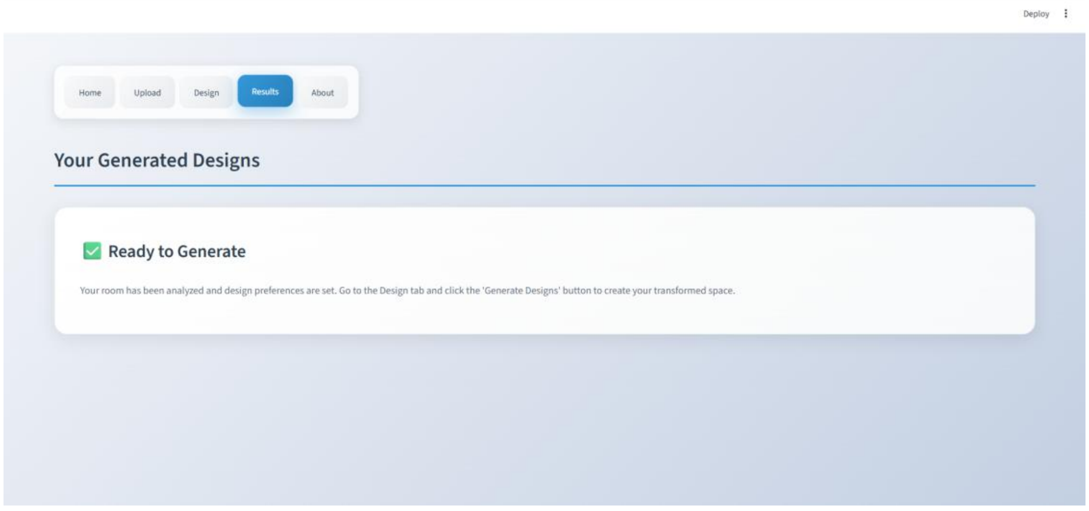
3

GENERATE DESIGNS

Figure 40: Custom Prompting Option

- Developed Output Page to display:

  – Generated design image



Deploy

Home Upload Design Results About

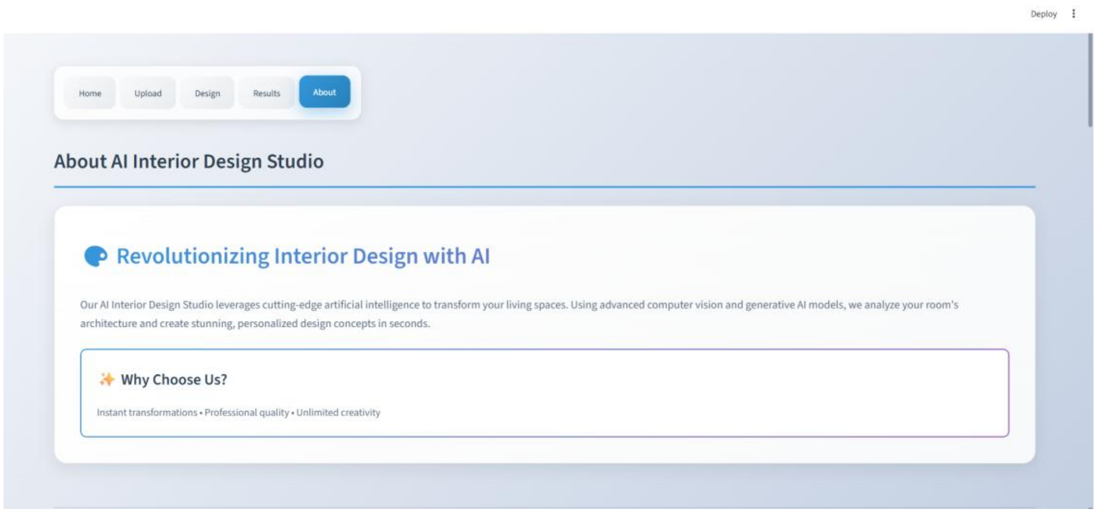
Your Generated Designs

✓ Ready to Generate

Your room has been analyzed and design preferences are set. Go to the Design tab and click the 'Generate Designs' button to create your transformed space.

Figure 41: Generated Image appear here

- About tab for this project



About AI Interior Design Studio

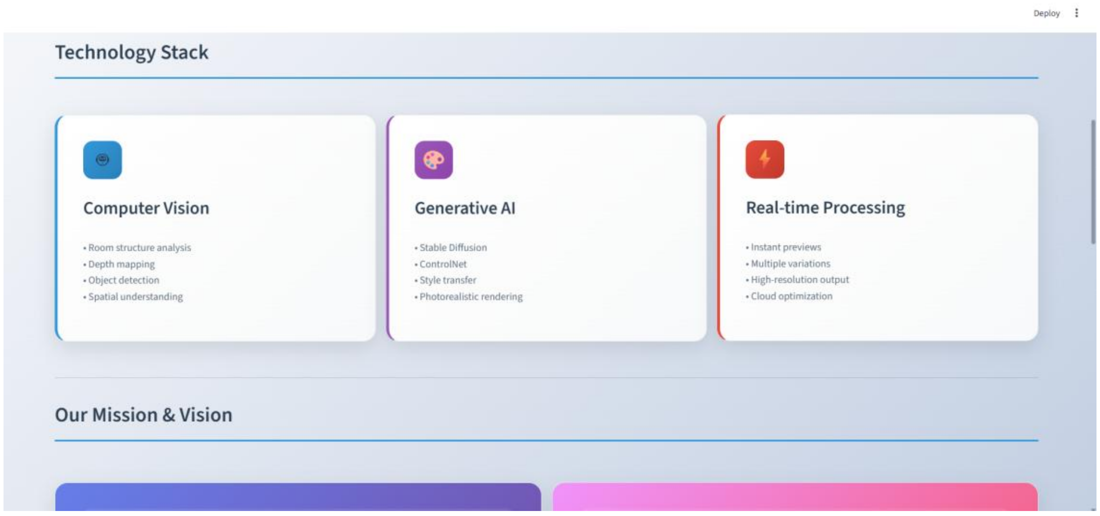
Revolutionizing Interior Design with AI

Our AI Interior Design Studio leverages cutting-edge artificial intelligence to transform your living spaces. Using advanced computer vision and generative AI models, we analyze your room's architecture and create stunning, personalized design concepts in seconds.

Why Choose Us?

Instant transformations • Professional quality • Unlimited creativity

Figure 42: About Tab Preview 1



Technology Stack

Computer Vision
- Room structure analysis
- Depth mapping
- Object detection
- Spatial understanding

Generative AI
- Stable Diffusion
- ControlNet
- Style transfer
- Photorealistic rendering

Real-time Processing
- Instant previews
- Multiple variations
- High-resolution output
- Cloud optimization

Our Mission & Vision

Figure 43: About Section Preview 2

Activity 5.2: Testing with Sample Room Images

- Uploaded a room images (e.g. bedroom, living room, workspace, kitchen)

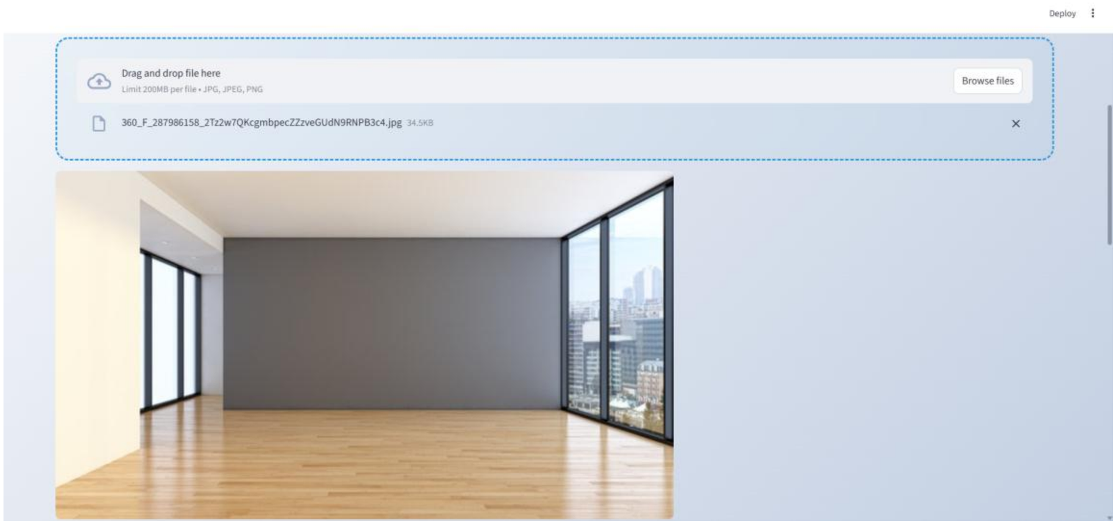
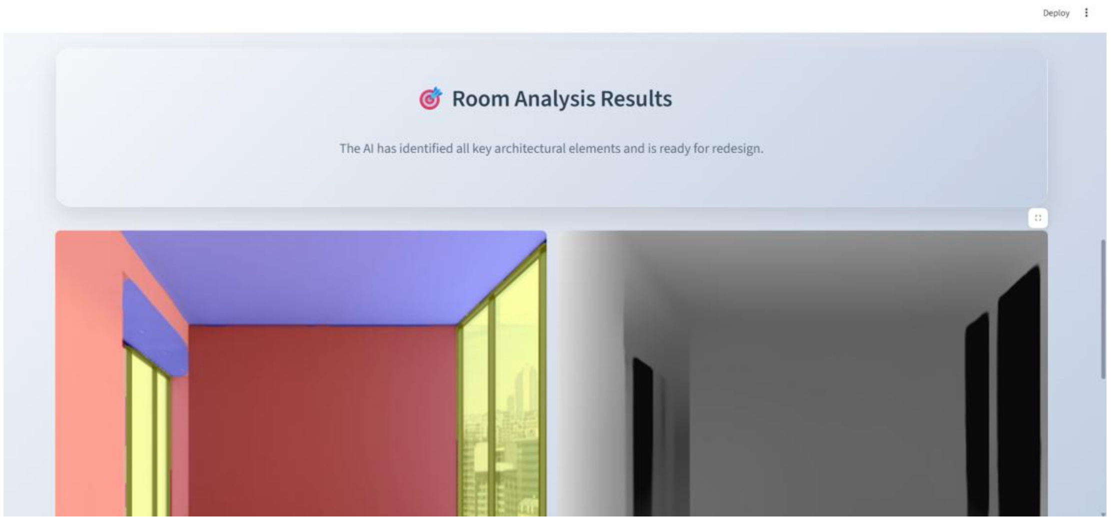

Deploy
Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG
Browse files
360_F_287986158_2Tz2w7QKcgmbpecZZzveGUdN9RNPB3c4.jpg 34.5KB

Figure 44: Uploading room image

- Verified image preview after upload along with the segmentation and depth map

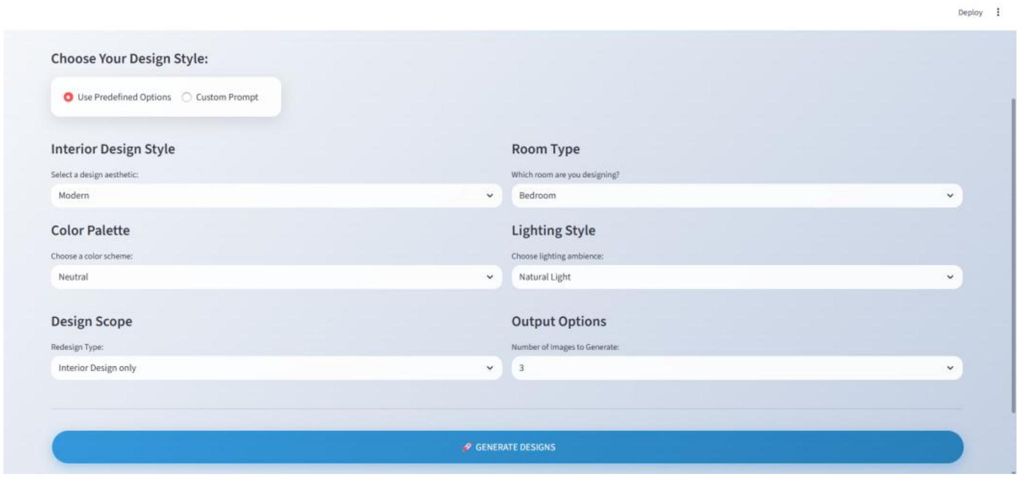


Room Analysis Results

The AI has identified all key architectural elements and is ready for redesign.

Figure 45: Segmentation and Depth Map

- Entered design preferences (style, colour, lighting) and generated AI-based interior design


Deploy

Choose Your Design Style:

[✓ Use Predefined Options] Custom Prompt

Interior Design Style

Select a design aesthetic: Modern

Room Type

Which room are you designing? Bedroom

Color Palette

Choose a color scheme: Neutral

Lighting Style

Choose lighting ambience: Natural Light

Design Scope

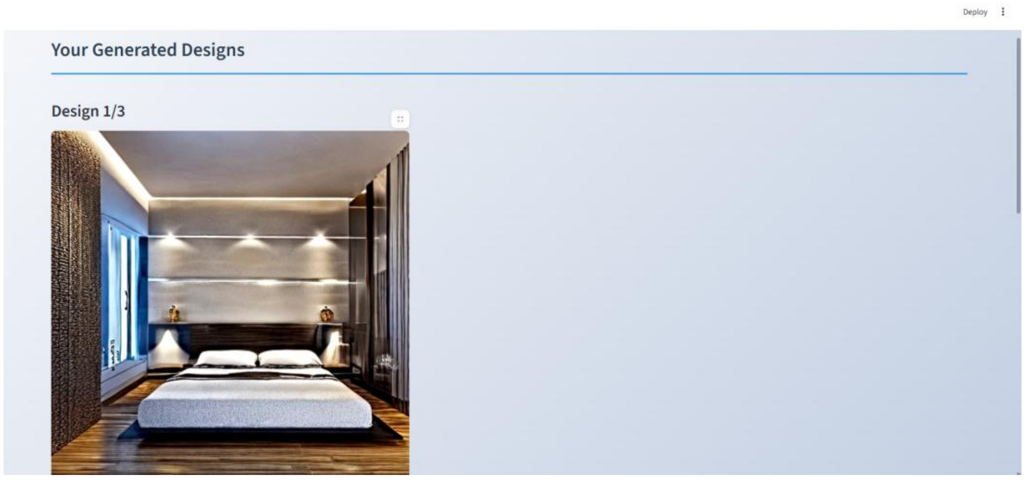
Redesign Type: Interior Design only

Output Options

Number of Images to Generate: 3

Figure 46: Selecting the option for generation of image

- Tested whether Stable Diffusion + ControlNet pipeline follows depth/segmentation correctly and give proper output



Your Generated Designs

Design 1/3

Figure 47: Generated Image Tab

Activity 5.3: Performance Optimization & Error Handling

- Optimized model loading by storing models in session_state to avoid reloading
- Added progress indicators and visual feedback during generation
- Implemented error handling for:
    - Invalid image format
    - Missing user input
    - Connection/model load failure

- Verified GPU availability for faster processing (torch.cuda.is_available())
- Improved UI responsiveness and reduced design generation delay

Conclusion

The AI Interior Design Generator represents a significant advancement in accessible, AI-powered home design technology. By combining state-of-the-art computer vision with generative AI, the platform democratizes professional interior design capabilities, making them available to homeowners, renters, and real estate professionals alike.

The integration of Stable Diffusion with ControlNet ensures that transformations maintain architectural integrity while enabling creative exploration. The sophisticated room analysis pipeline provides users with insights into their space's structure, while the AI-powered generation system offers limitless design possibilities tailored to individual preferences.

Built on a robust technical foundation with Streamlit providing an intuitive interface, the application demonstrates how advanced AI capabilities can be packaged for mainstream accessibility. The attention to user experience, from the engaging homepage to the detailed results presentation, ensures that users feel guided and supported throughout their design journey.

Looking forward, the platform has substantial potential for expansion. Features like 3D room planning, furniture recommendation systems, material and cost estimation, integration with e-commerce platforms for direct purchasing, and collaborative tools for professional designers could further transform how people approach interior design. The underlying technology also shows promise for applications in virtual staging, architectural visualization, and real estate marketing.

In essence, the AI Interior Design Studio bridges the gap between professional design expertise and everyday home improvement, empowering users to reimagine their spaces with confidence and creativity, backed by the transformative power of artificial intelligence.