**CAD Visualizer Project Documentation**

**Table of Contents**

1. **Introduction**
2. **Project Overview**
3. **Requirements**
4. **Setup Instructions**
5. **Folder Structure**
6. **Code Explanation**
   * **Flask App (app.py)**
   * **HTML Templates**
   * **CSS Styling (styles.css)**
   * **JavaScript Functionality (scripts.js)**
7. **How to Use the App**
8. **Screenshots**
9. **Conclusion**

**1. Introduction**

**The CAD Visualizer is a web application that allows users to upload CAD models (e.g., .stl, .obj files) and generate photorealistic images of the models in customizable environments, lighting, and angles. The app uses Flask for the backend, Replicate API for image generation, and HTML/CSS/JavaScript for the frontend.**

**2. Project Overview**

* **Frontend: HTML, CSS, JavaScript**
* **Backend: Flask (Python)**
* **API: Replicate API (Stable Diffusion)**
* **Features:**
  + **Upload CAD models.**
  + **Customize environment, lighting, and angle.**
  + **Generate photorealistic images.**
  + **Display images as landscape wallpapers.**
  + **User-friendly interface with animations.**

**3. Requirements**

**Software Requirements**

* **Python 3.7 or higher**
* **Flask (pip install flask)**
* **Replicate Python Client (pip install replicate)**
* **Pillow (pip install pillow)**
* **Requests (pip install requests)**

**API Key**

* **Sign up for a Replicate API key at**[**Replicate**](https://replicate.com/)**.**
* **Save the API key in a file named api\_key.txt in the project root directory.**

**4. Setup Instructions**

1. **Clone the Repository:**

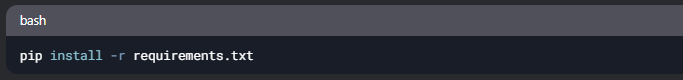
**bash**

**Copy**

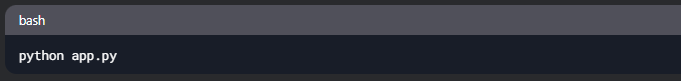
**git clone https://github.com/yourusername/cad-visualizer.git**

**cd cad-visualizer**

1. **Install Dependencies:**

****

1. **Add API Key:**
   * **Create a file named api\_key.txt in the project root.**
   * **Paste your Replicate API key into the file.**
2. **Run the App:**

****

1. **Access the App:  
   Open your browser and navigate to http://127.0.0.1:5000.**

**5. Folder Structure**

**A screenshot of a computer

AI-generated content may be incorrect.**

**6. Code Explanation**

**6.1 Flask App (app.py)**

**The Flask app handles file uploads, API calls, and image generation.**

**Python**

**1. Flask Module**

**Flask is a lightweight web framework for Python. It is used to create the backend of the application, handle routes, and render HTML templates.**

**Explanation**

* **Flask: The main class used to create the Flask application.**
* **request: Used to handle incoming HTTP requests (e.g., file uploads, form data).**
* **render\_template: Renders HTML templates with dynamic data.**
* **send\_file: Sends files (e.g., images) as responses.**
* **redirect: Redirects the user to a different route.**
* **url\_for: Generates URLs for routes.**

**2. OS Module**

**The os module is used to interact with the operating system, such as creating directories and handling file paths.**

**Explanation**

* **os.makedirs(): Creates directories (e.g., uploads/ and static/generated/).**
* **os.path.join(): Joins file paths in a platform-independent way.**
* **os.path.exists(): Checks if a file or directory exists.**

**3. Replicate Module**

**The replicate module is used to interact with the Replicate API, which generates photorealistic images using the Stable Diffusion model.**

**Explanation**

* **replicate.run(): Calls the Replicate API to generate images based on a prompt.**
* **Input Parameters:**
  + **prompt: A text description of the image to generate.**
  + **width and height: The dimensions of the generated image (e.g., 1920x1080 for 16:9 aspect ratio).**

**4. Pillow Module**

**The Pillow module (PIL) is used for image processing. In this project, it is used to create and manipulate images.**

**Explanation**

* **Image.new(): Creates a new image (used in the /generate\_image route).**
* **Image.save(): Saves an image to a file or BytesIO object.**

**5. IO Module**

**The io module is used to handle byte streams, such as storing images in memory before sending them as responses.**

**Explanation**

* **io.BytesIO(): Creates an in-memory binary stream to store image data.**
* **img\_io.seek(0): Resets the stream position to the beginning.**

**6. Requests Module**

**The requests module is used to make HTTP requests, such as downloading the generated image from the Replicate API.**

**Explanation**

* **requests.get(): Sends a GET request to download the image from a URL.**
* **response.content: Contains the binary data of the downloaded image.**

**7. Logging Module**

**The logging module is used to log messages for debugging and monitoring.**

**Explanation**

* **logging.basicConfig(): Configures the logging level (e.g., logging.DEBUG).**
* **logging.debug(): Logs debug messages.**
* **logging.error(): Logs error messages.**

**8. Flask App Initialization**

**The Flask app is initialized and configured with upload and generated folders.**

**Explanation**

* **UPLOAD\_FOLDER: Directory where uploaded CAD files are stored.**
* **GENERATED\_FOLDER: Directory where generated images are saved.**

**9. Reading the API Key**

**The Replicate API key is read from a file (api\_key.txt) and set as an environment variable.**

**Explanation**

* **read\_api\_key(): Reads the API key from api\_key.txt.**
* **os.environ[]: Sets the API key as an environment variable for the Replicate client.**

**10. Routes**

**Home Route (/)**

**Renders the homepage (index.html).**

**File Upload Route (/upload)**

**Handles file uploads and saves the file to the uploads/ folder.**

**Image Generation Route (/generate)**

**Generates an image using the Replicate API and renders the results page (results.html).**

**Image Download Function**

**Downloads the generated image from the Replicate API and saves it to the static/generated/ folder.**

**11. Running the App**

**The app is run in debug mode for development.**

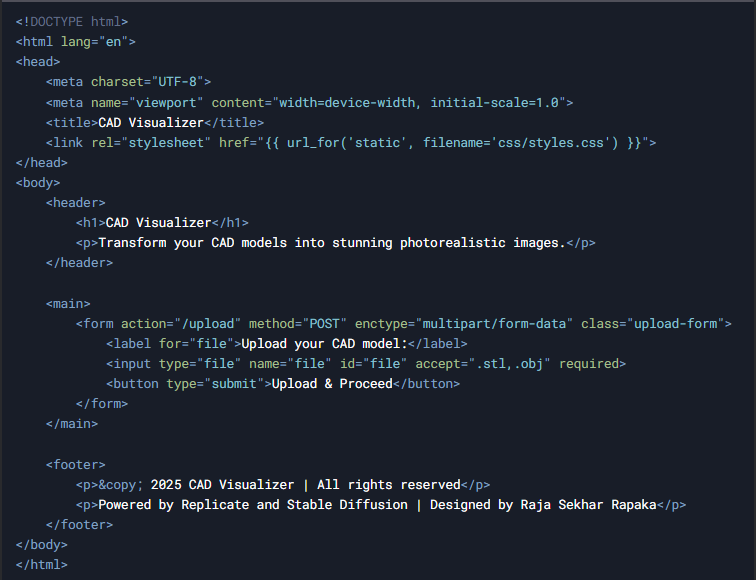
**A black and white text on a black background

AI-generated content may be incorrect.**

**6.2 HTML Templates**

**index.html**

**The homepage where users upload CAD models.**

****

**6.3 CSS Styling (styles.css)**

**Styles for the app.**

**Css**

**1. General Styles**

**These styles apply to the entire application.**

**Code Example**

**A computer screen with white text

AI-generated content may be incorrect.**

**Explanation**

* **font-family: Sets the font to Arial or a sans-serif fallback.**
* **margin and padding: Removes default spacing around the body.**
* **background: Sets a light gray background color.**
* **color: Sets the default text color to dark gray.**
* **display: flex: Uses Flexbox to create a column layout.**
* **min-height: 100vh: Ensures the body takes up at least the full viewport height.**

**2. Header Styles**

**Styles for the header section, which includes the app title and description.**

**Code Example**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Explanation**

* **background: Applies a gradient background (purple to blue).**
* **color: Sets the text color to white.**
* **padding: Adds spacing inside the header.**
* **text-align: Centers the text horizontally.**
* **font-size: Sets the size of the heading and paragraph text.**
* **margin: Adjusts spacing around the heading and paragraph.**

**3. Form Styles**

**Styles for the file upload and image generation forms.**

**Code Example**

**A screen shot of a computer program

AI-generated content may be incorrect.**

**Explanation**

* **background: Sets a white background for the forms.**
* **padding: Adds spacing inside the forms.**
* **border-radius: Rounds the corners of the forms.**
* **box-shadow: Adds a subtle shadow for depth.**
* **max-width: Limits the width of the forms.**
* **margin: Centers the forms horizontally.**
* **animation: Applies a fade-in animation.**
* **label: Styles form labels (bold and block-level).**
* **input[type="file"], select, button: Styles file inputs, dropdowns, and buttons.**
* **button:hover: Changes the button color on hover.**

**4. Image Container Styles**

**Styles for the container that displays the generated image.**

**Code Example**

**A screen shot of a computer

AI-generated content may be incorrect.**

**Explanation**

* **width and height: Makes the container full-screen.**
* **display: flex: Centers the image horizontally and vertically.**
* **overflow: hidden: Hides any overflow from the image.**
* **background: Sets a black background for contrast.**
* **object-fit: cover: Ensures the image covers the container without distortion.**
* **animation: Applies a fade-in animation.**

**5. Footer Styles**

**Styles for the footer section, which includes copyright and attribution information.**

**Code Example**

**A black rectangular object with a white border

AI-generated content may be incorrect.**

**Explanation**

* **text-align: Centers the text horizontally.**
* **padding: Adds spacing inside the footer.**
* **background: Sets a dark gray background.**
* **color: Sets the text color to white.**

**6. Animations**

**Defines keyframe animations for fade-in effects.**

**Code Example**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Explanation**

* **@keyframes fadeIn: Defines a fade-in animation.**
  + **from: Starts with 0 opacity and a slight vertical offset.**
  + **to: Ends with full opacity and no offset.**

**7. How to Use Button and Instructions Box**

**Styles for the "How to Use" button and instructions box.**

**Code Example**

**A screen shot of a computer

AI-generated content may be incorrect.**

**A screen shot of a computer

AI-generated content may be incorrect.**

**A screen shot of a computer

AI-generated content may be incorrect.**

**A computer screen shot of a computer code

AI-generated content may be incorrect.**

**Explanation**

* **.how-to-use-container: Positions the button and instructions box.**
* **#how-to-use-btn: Styles the "How to Use" button.**
* **.instructions-box: Styles the instructions box (hidden by default).**
* **@keyframes slideDown: Defines a slide-down animation for the instructions box.**

**A black and white text on a black background

AI-generated content may be incorrect.**

**6.4 JavaScript Functionality (scripts.js)**

**Handles the "How to Use" instructions box.**

**Javascript**

****

**7. How to Use the App**

1. **Upload a CAD Model:**
   * **Go to the homepage and upload a .stl , .obj, .dwg, .dxf file.**
2. **Customize Settings:**
   * **Select the environment, lighting, and angle.**
3. **Generate Image:**
   * **Click "Generate Image" to create a photorealistic render.**
4. **View Results:**
   * **The generated image will be displayed as a landscape wallpaper.**

**8. Screenshots**

**Include screenshots of:**

* **Homepage**

**A screenshot of a computer

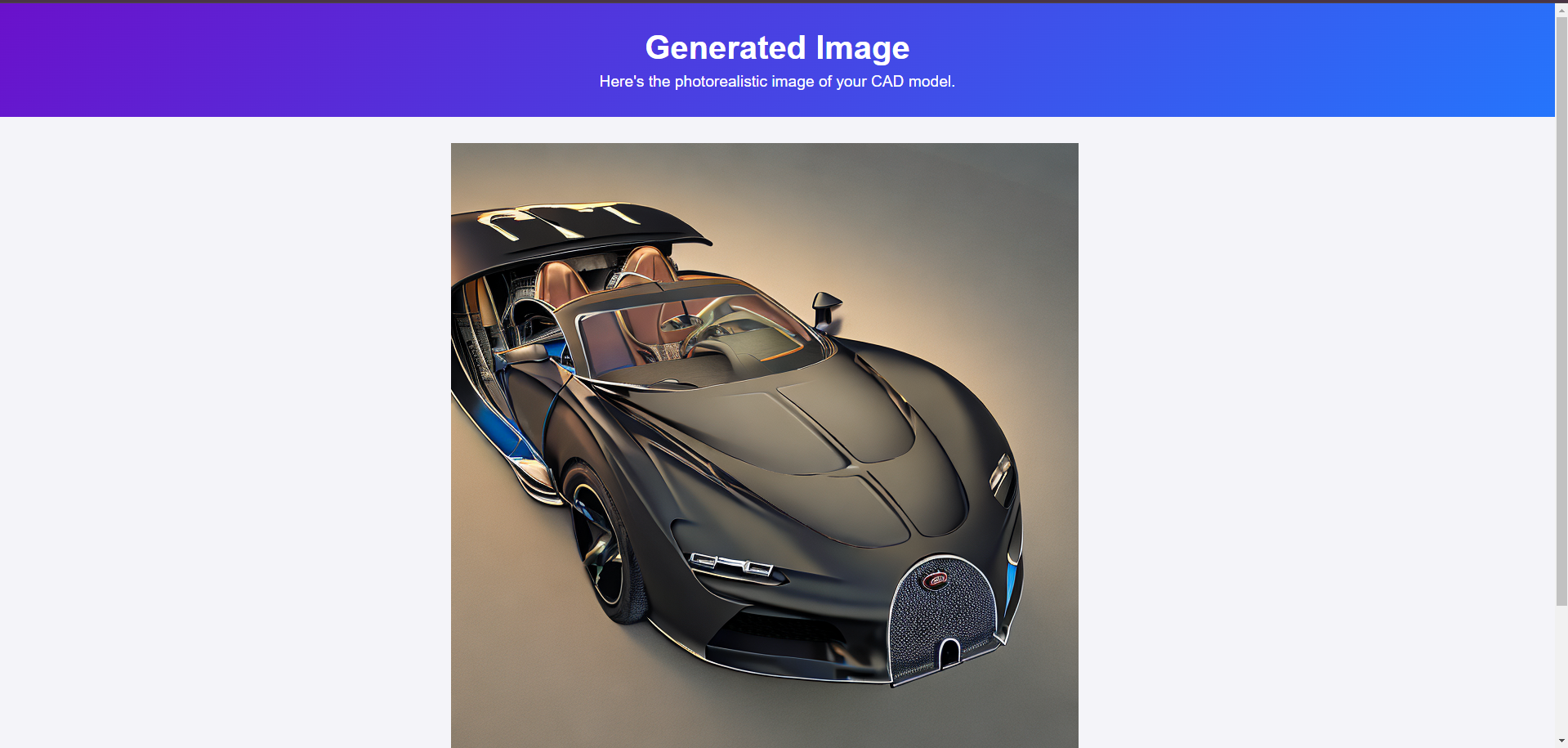
AI-generated content may be incorrect.**

* **Customization page**

**A screenshot of a computer

AI-generated content may be incorrect.**

* **Generated image page**

****

**9. Conclusion**

**The CAD Visualizer is a powerful tool for transforming CAD models into photorealistic images. With its user-friendly interface and customizable settings, it’s perfect for designers, engineers, and hobbyists.**

**Raja Sekhar Rapaka**