**Civil Engineering Insight Studio**

Problem Statement: Civil engineers often face the challenge of manually describing structures based on images, which can be time-consuming and subjective. Without automated tools, generating detailed descriptions of civil engineering structures, including types, materials, dimensions, construction methods, and notable features, requires significant human effort and expertise. To address this challenge, there is a need for an efficient and reliable tool that can automatically analyze images of civil engineering structures and generate insightful descriptions, enabling engineers to make informed decisions and communicate effectively about their projects.

**Scenario 1: Material Identification in Building Construction**

A construction supervisor overseeing a building project utilizes the Civil Engineering Insight Studio to identify materials used in the building's construction. The supervisor uploads an image of the construction site and prompts the tool to identify construction materials such as concrete, steel, and bricks. The tool analyzes the image, identifies various materials, and provides a comprehensive list with descriptions of each material's type, quantity, and location within the structure.

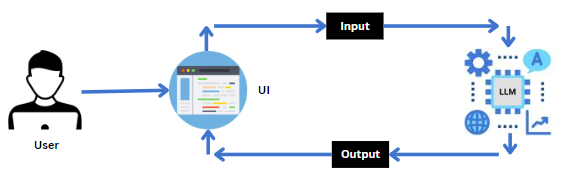
**Scenario 2: Project Progress Documentation for Construction Firm**

A civil engineering firm engaged in documenting ongoing construction projects utilizes the tool to streamline project documentation. Project managers upload images of construction sites and prompt the tool to document project progress, including descriptions of completed and planned structural elements. The tool analyzes the images, generates detailed documentation of completed structural elements, materials used, dimensions, construction methods, and planned phases of work, facilitating effective project management and communication within the firm

**Scenario 3: Structural Analysis of Bridge**

In this scenario, a civil engineer tasked with assessing the structural integrity of a bridge uploads an image of the bridge to the Civil Engineering Insight Studio. The engineer requests an analysis of the bridge's structural components, including beams, columns, and trusses. The tool processes the image, identifies key structural elements, and generates detailed descriptions of each component's material, dimensions, and construction methods. Additionally, the tool highlights any notable features or engineering challenges observed in the structure.

**Architecture**



### Project Flow

* User Input: Users provide descriptions and upload images of civil engineering structures.
* Backend Processing: Input data is analyzed by specialized algorithms.
* Insight Generation: Detailed descriptions and related information are autonomously generated.
* Frontend Display: Generated insights and information  are presented on the user interface.

**To accomplish this, we have to complete all the activities listed below,**

* **Requirements Specification**
  + Create a requirements.txt file to list the required libraries.
  + Install the required libraries
* **Initialization of Google API Key**
  + Generate Google API Key
  + Initialize Google API Key
* **Interfacing with Pre-trained Model**
  + Load the Gemini Pro pre-trained model
  + Implement a function to get gemini response
  + Implement a function to read the Image and set the image format for Gemini Pro model Input
  + Write a prompt for gemini model
* **Model Deployment**
  + Integrate with Web Framework

### Process User Input and Generate Landmark Description

* + Run The Web Application

**Prior Knowledge**

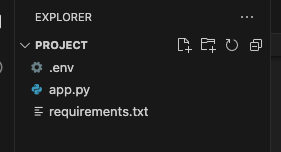
You must have the prior knowledge of the following topics to complete this project.

* Generative AI Concepts
* NLP: <https://www.tutorialspoint.com/natural_language_processing/index.htm>
* Generative AI: <https://en.wikipedia.org/wiki/Generative_artificial_intelligence>
* About Gemini: <https://deepmind.google/technologies/gemini/#introduction>
* Gemini API: <https://ai.google.dev/gemini-api/docs/get-started/python>
* Gemini Demo: <https://colab.research.google.com/github/google/generative-ai-docs/blob/main/site/en/gemini-api/docs/get-started/python.ipynb>

Streamlit: <https://www.geeksforgeeks.org/a-beginners-guide-to-streamlit/>

### Project Structure

Create the Project folder which contains application file as shown below



 ? .env contains the api key for Google Ai studio

? App.py contains the code for running the model using streamlit

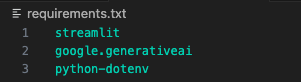
? Requirements.txt has all the requirements (packages and libraries needed to install

before running the project)

**Milestone 1: Requirements Specification**

Specifying the required libraries in the requirements.txt file ensures seamless setup and reproducibility of the project environment, making it easier for others to replicate the development environment

### Activity 1: Create a requirements.txt file to list the required libraries.



* **Streamlit:**A Python library used for creating interactive web applications with simple Python scripts.
* **google.generativeai:**A module providing access to Google's Generative AI models for generating text, images, and other content.
* **python dotenv**:A Python library used for managing environment variables stored in a `.env` file, typically used for configuration settings.

### Activity 2: Install the required libraries.

20.png

* Open the terminal.
* Run the command: pip install r requirements.txt
* This command installs all the libraries listed in the requirements.txt file

**Milestone 2: Initialization of Google API Key**

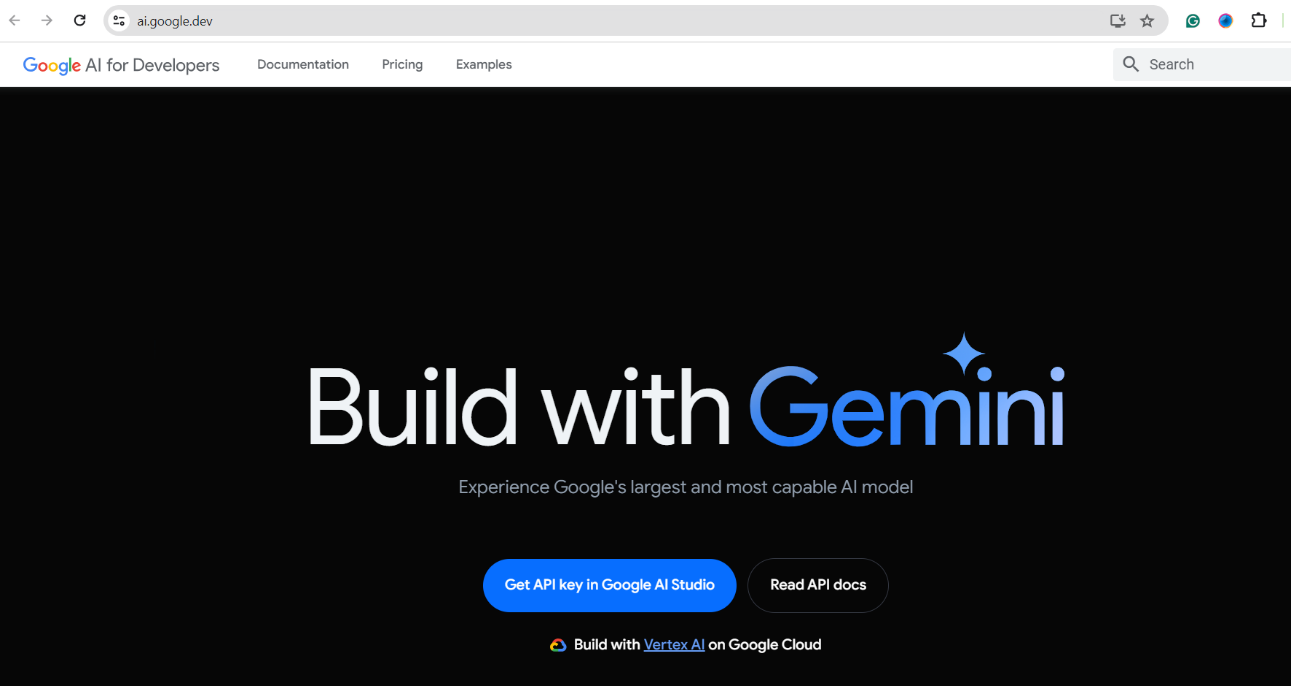
The Google API key is a secure access token provided by Google, enabling developers to authenticate and interact with various Google APIs. It acts as a form of identification, allowing users to access specific Google services and resources. This key plays a crucial role in authorizing and securing API requests, ensuring that only authorized users can access and utilize Google's services.

### Activity 1: Generate Google API Key

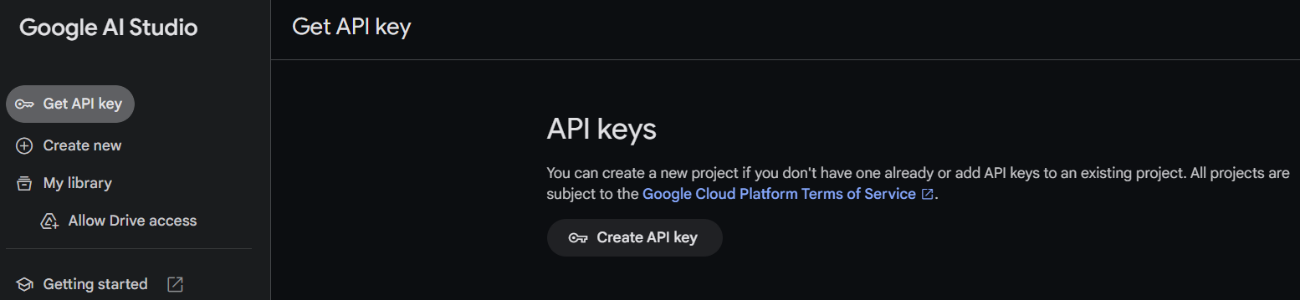
Click the provided link to access the following webpage.

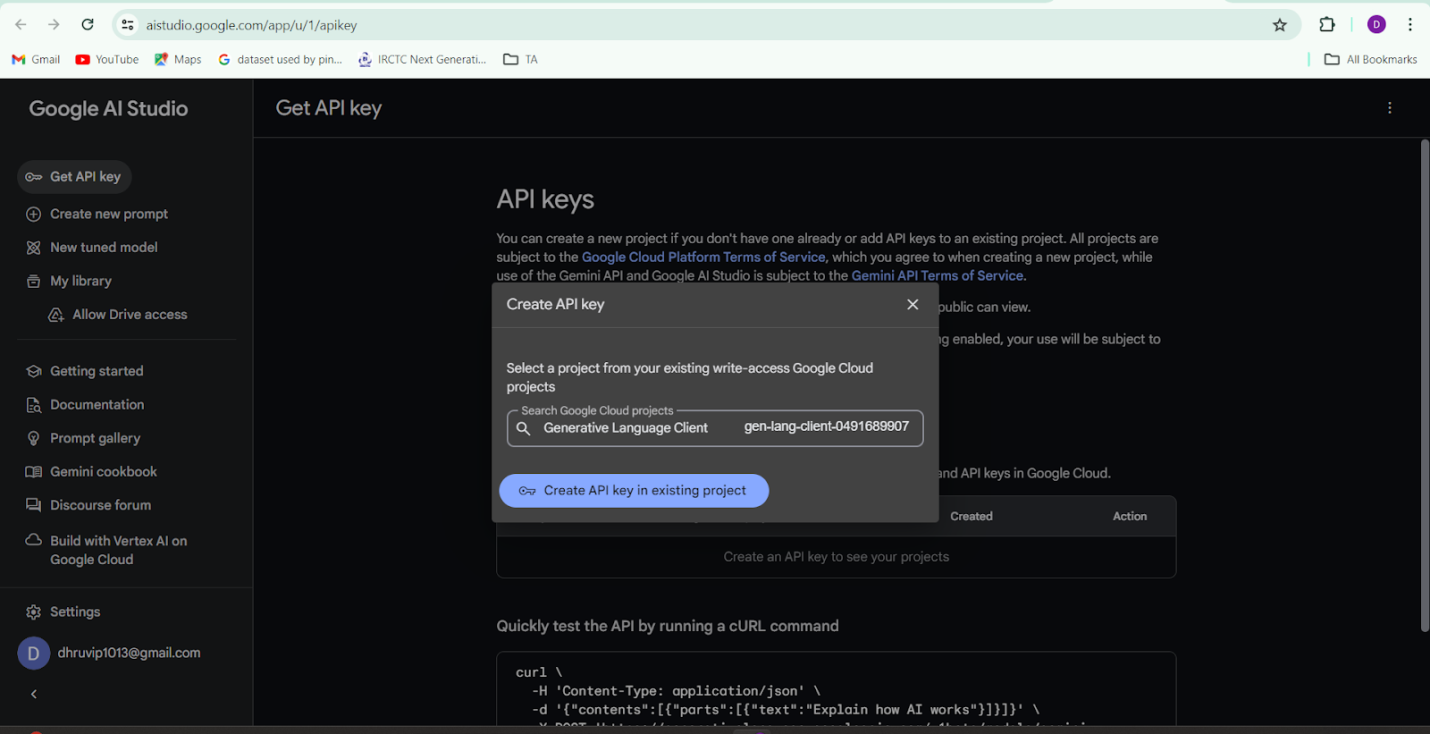
Link: <https://ai.google.dev/gemini-api/docs/api-key>

After signing in to your account, navigate to the 'Get an API Key' option. Clicking on this option will redirect you to another webpage as shown below



Next, click on 'Create API Key' and choose the generative language client as the project. Then, select 'Create API key in existing project'.





Copy the newly generated API key as it is required for loading the Gemini Pro pre-trained model.

### Activity 2: Initialize Google API Key

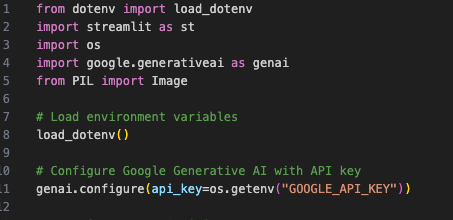
5.png

* Create a .env file and define a variable named GOOGLE\_API\_KEY.
* Assign the copied Google API key to this variable.
* Paste the API key obtained from the previous steps here.

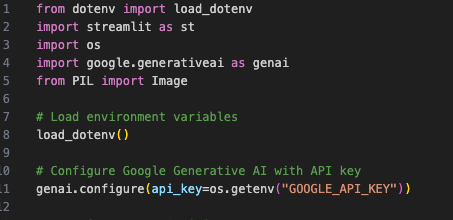
### Milestone 3: Interfacing with Pre-trained Model

To interface with the pre-trained model, we'll start by creating an app.py file, which will contain both the model and Streamlit UI code.

### Activity 1: Load the Gemini Pro API

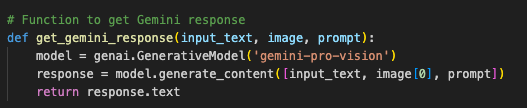


* `dotenv`: Loads environment variables from a `.env` file into the environment.
* `streamlit`: Creates interactive web applications using Python scripts.
* `os`: Provides a way to interact with the operating system, including accessing environment variables and file paths.
* `google.generative ai`: Grants access to Google's Generative AI models for content generation.
* `PIL (Python Imaging Library)`: Allows opening, manipulating, and saving various image file formats



* Load Environment Variables: Utilizes dotenv to load environment variables, ensuring sensitive data like API keys are securely managed.
* Configure Google Generative AI: Uses the configured API key from the environment variables to set up and authenticate with Google's Generative AI service, enabling content generation functionalities.

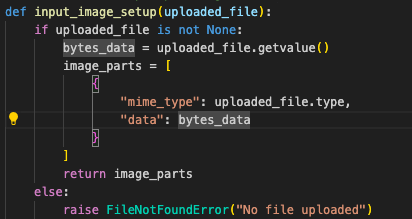
### Activity 2: Implement a function to get gemini response



* Function Definition: The function `get\_gemini\_response` is defined to accept three parameters: `input\_text`, `image`, and `prompt`.
* Model Initialization: Inside the function, a new instance of the Gemini AI model is created using `genai.GenerativeModel('gemini-pro-vision')`.
* Generate Content Call: The function then calls `model.generate\_content` to generate content.
* Input Parameters: The `generate\_content` method is given a list containing `input\_text`, the first item from the `image` list, and `prompt`.
* Content Generation: The AI model processes these inputs to generate a detailed description of the civil engineering structure in the image.
* Extracting Text: The response from the AI model includes various data; the function specifically extracts the text part of this response.
* Return Statement: The extracted text is returned by the function.
* Purpose: This function facilitates the interaction with the Gemini AI model to generate detailed descriptions of civil engineering structures based on input text, an image, and a prompt, aiding civil engineers in analyzing and documenting structural details.

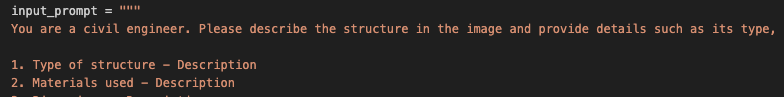
### Activity 3: Implement a function to read the Image and set the image format for Gemini Pro model Input

**Activity 3: Implement a function to read the Image and set the image format for Gemini Pro model Input**



* Function Definition: The function `input\_image\_setup` is defined to take a single parameter, `uploaded\_file`.
* Check for File: The function first checks if `uploaded\_file` is not `None`, ensuring that a file has been uploaded.
* Retrieve File Data: If a file is uploaded, it retrieves the file's binary data using `uploaded\_file.getvalue()`.
* Prepare Image Parts: It then creates a list called `image\_parts` containing a dictionary.
* Dictionary Content: The dictionary includes the MIME type of the uploaded file (`uploaded\_file.type`) and the binary data (`bytes\_data`).
* Return Image Parts: The function returns the `image\_parts` list, which is formatted for further processing.
* Handle No File: If no file is uploaded, the function raises a `FileNotFoundError` with the message "No file uploaded".

### Activity 4: Write a prompt for gemini model

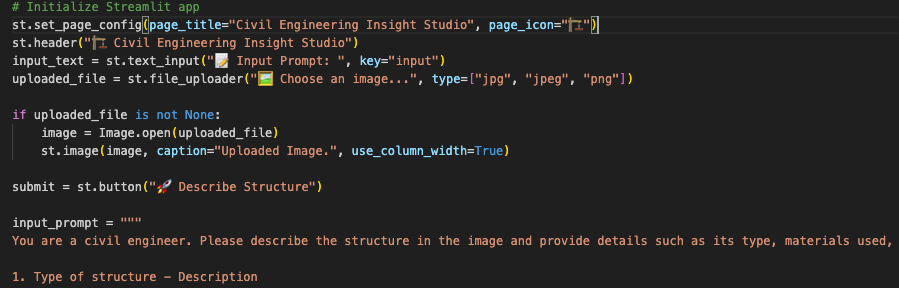


The variable input\_prompt is a single-line string designed as a prompt for a captioning AI model. It instructs the model to analyze an image .Additionally, the model is to provide a detailed breakdown of each item with its respective count

### Milestone 4: Model Deployment

This milestone sets up a web application for the Civil Engineering Structure Description App. It allows users to input a text prompt and upload an image of a civil engineering structure. Upon uploading an image, the app displays the image and provides a button for users to submit their input. The app is designed to describe civil engineering structures based on user input, using the Google Generative AI model. It provides detailed descriptions of the structure, including type, materials used, dimensions, construction methods, and notable features or engineering challenges

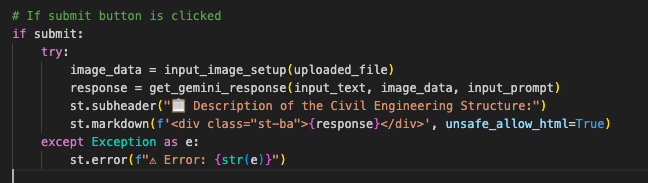
### Activity 1:Integrate with Web Framework



* Initialize Streamlit App with configured page title and icon.
* Header Definition: The code starts by setting the header of the Streamlit application to "Civil Engineering Insight Studio" using st.header(). This sets the title for the web app.
* Text Input Field: The st.text\_input() function creates an input field where users can type a text prompt. The prompt is labeled " Input Prompt: " and is assigned the key "input" for easy reference.
* File Uploader: The st.file\_uploader() function creates an interface for users to upload an image file. The accepted file types are specified as "jpg", "jpeg", and "png". The label for this uploader is "Choose an image...".
* Image Display: If a user uploads a file, the code reads the image using Image.open(uploaded\_file). The uploaded image is then displayed in the web app with the caption "Uploaded Image." and is set to use the column width for display using st.image().
* Submit Button: The st.button() function creates a button labeled " Describe Structure". When this button is clicked, the code proceeds to the next steps.
* Input Prompt: The input\_prompt variable contains a detailed prompt for the AI model. It instructs the model to describe the structure in the image and provide various details such as type, materials used, dimensions, construction methods, and notable features or engineering challenges.
* Button Click Handling: When the "Describe Structure" button is clicked, the code checks if an image file has been uploaded. If so, it processes the image and uses it along with the input text and prompt to generate a detailed description of the civil engineering structure.

### Activity 2: Process User Input and Generate Landmark Description

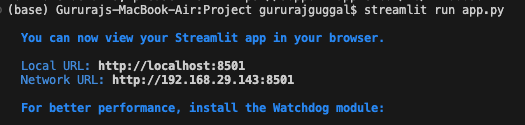
This activity involves processing user input and generating a description of the landmark based on the provided information.



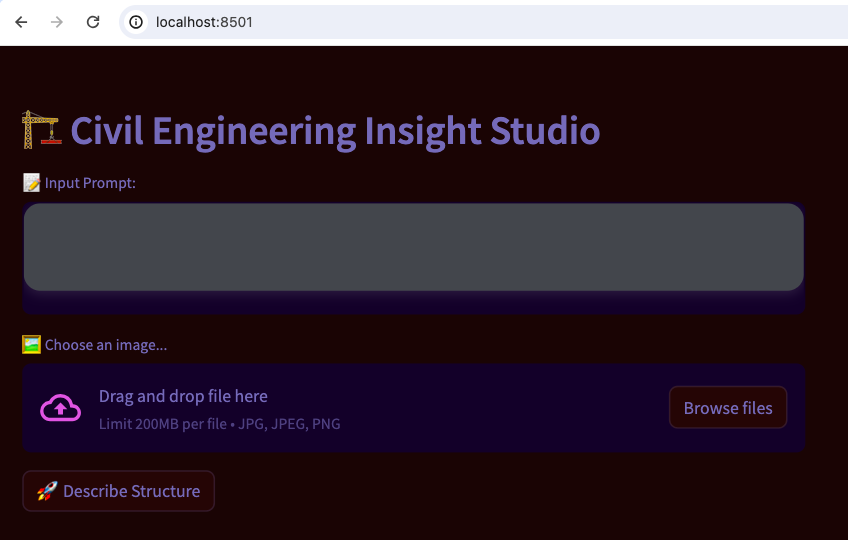
* This code is part of a web application designed for civil engineering projects.
* Users can input text and upload an image of a structure.
* The app processes this data to create a description of the structure.
* The description is then displayed on the webpage.
* If any errors occur during this process, a warning message is shown.
* Overall, the code facilitates easy understanding and analysis of civil engineering structures through user input and automated description generation.

**Activity 3:Run the Web Application**

* Open the anaconda prompt from the start menu
* Navigate to the folder where your Python script is.
* Now type “streamlit run app.py” command
* Navigate to the localhost where you can view your web page

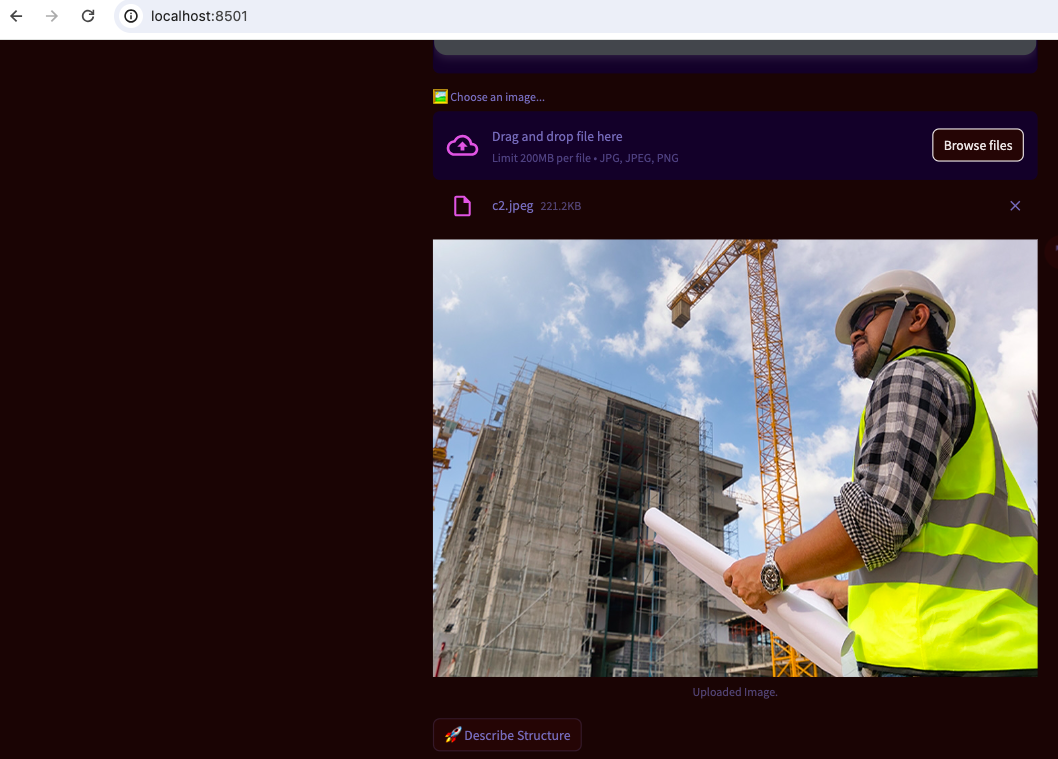


Now, the application will open in the web browser

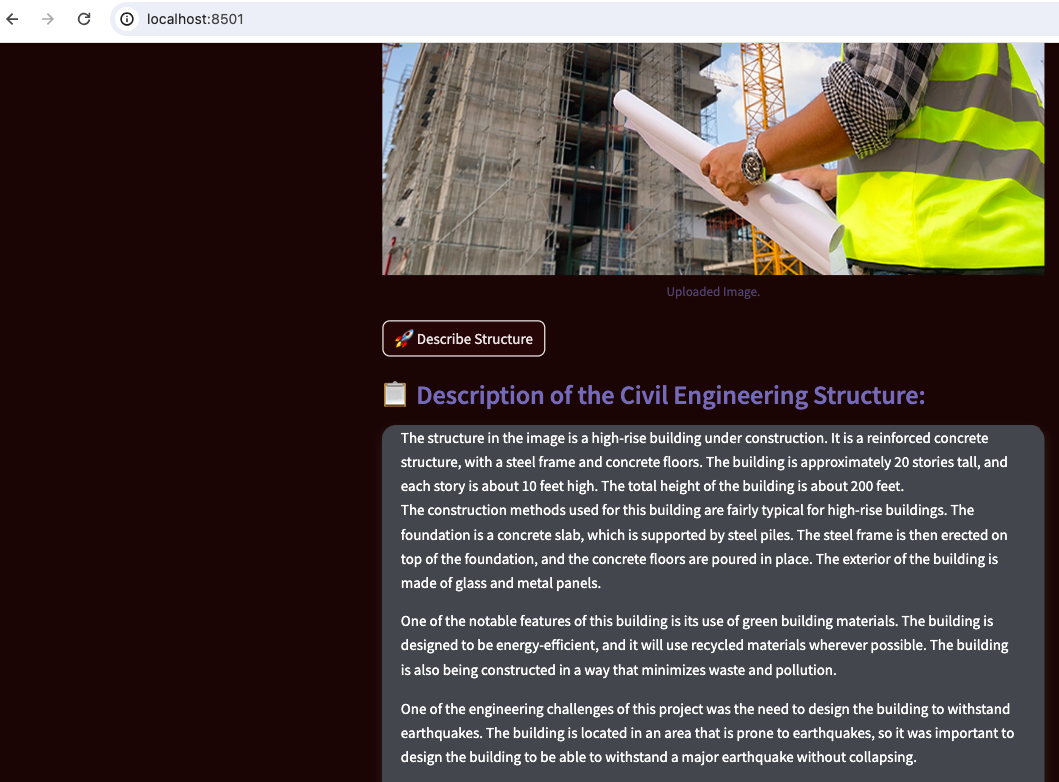


Input1:

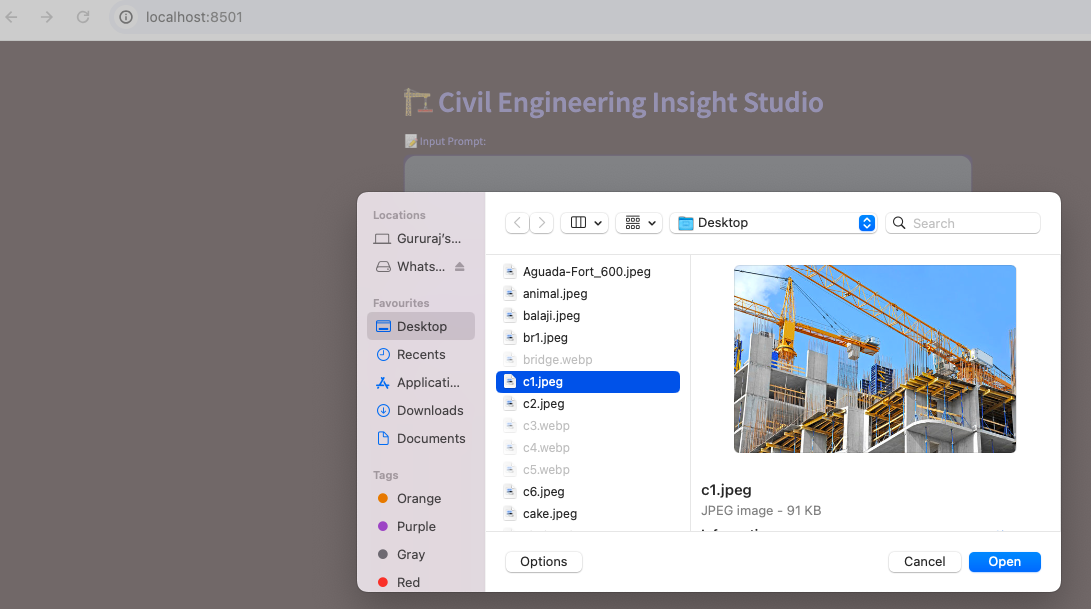


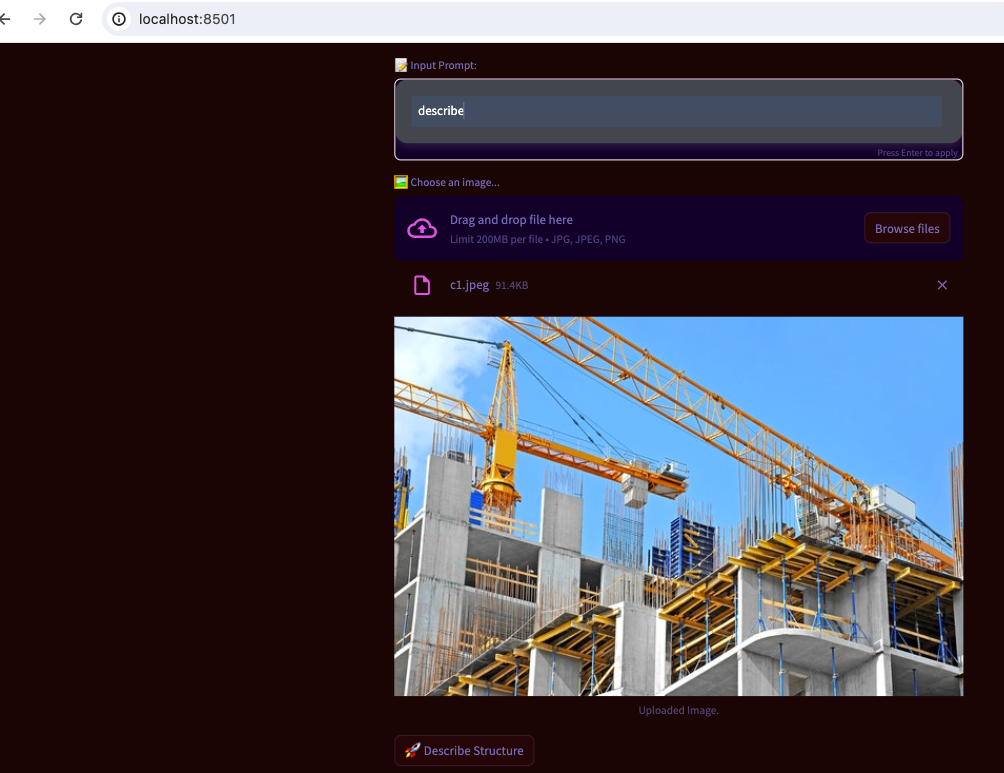


Output:

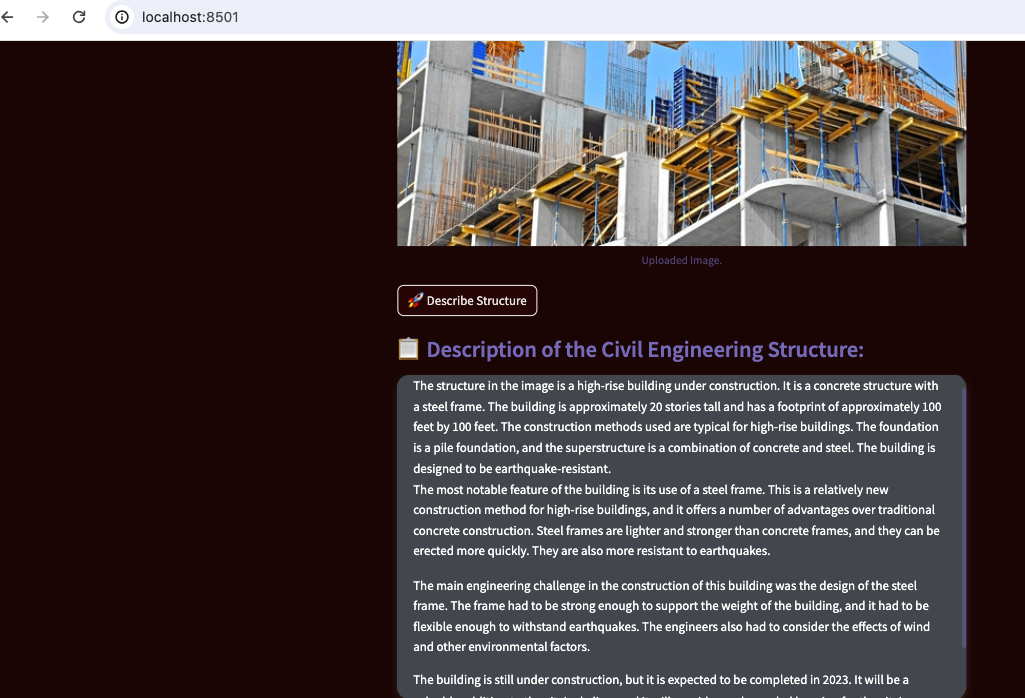


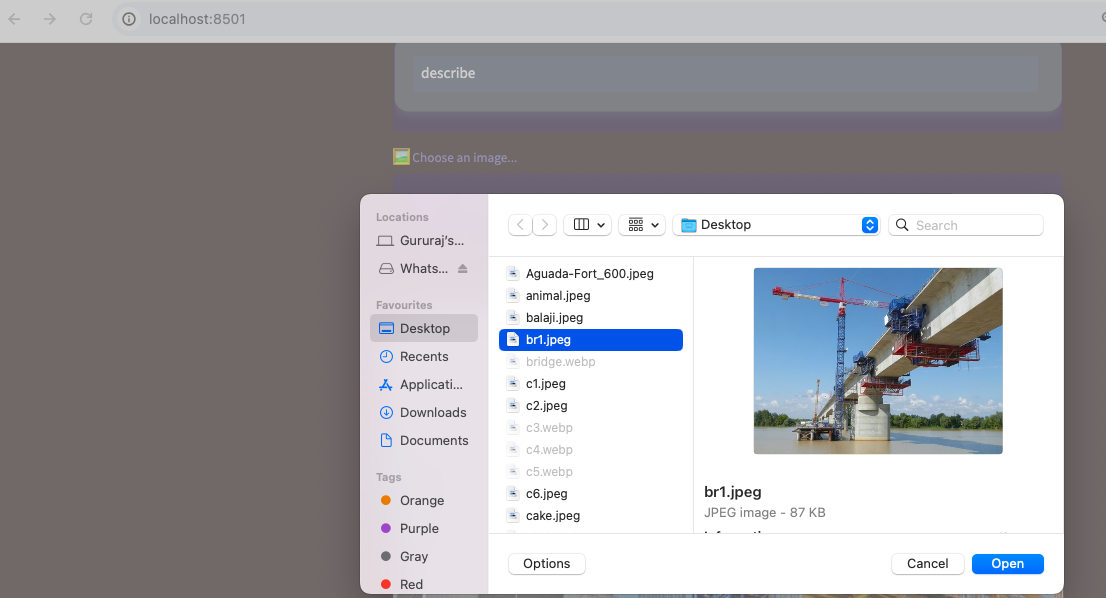
Input 2:

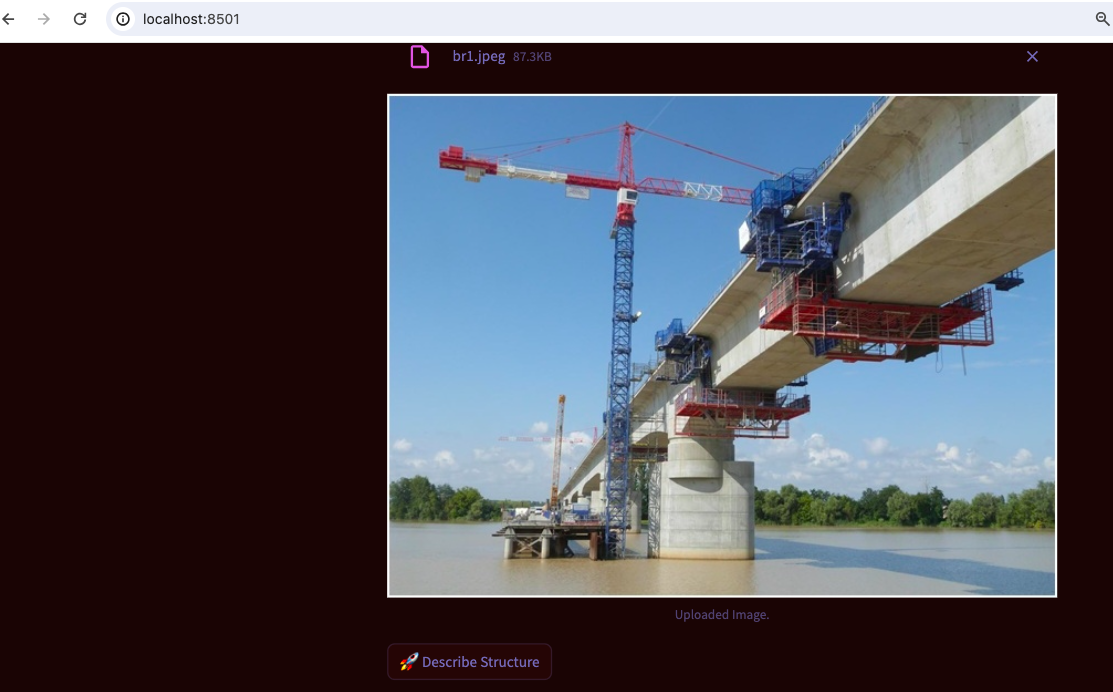




Output:



Input:3 3:



Output:

