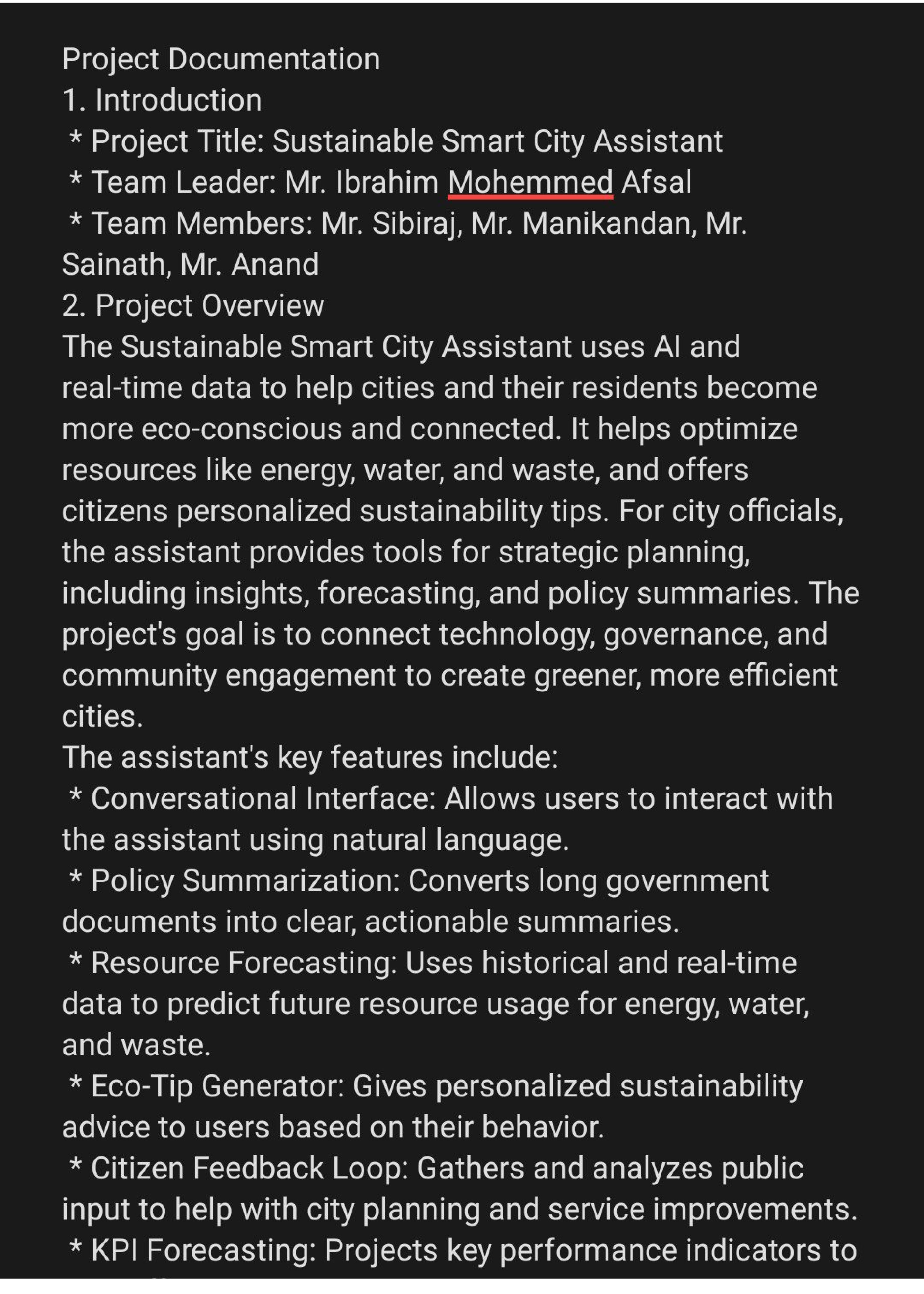


Project Documentation

1. Introduction

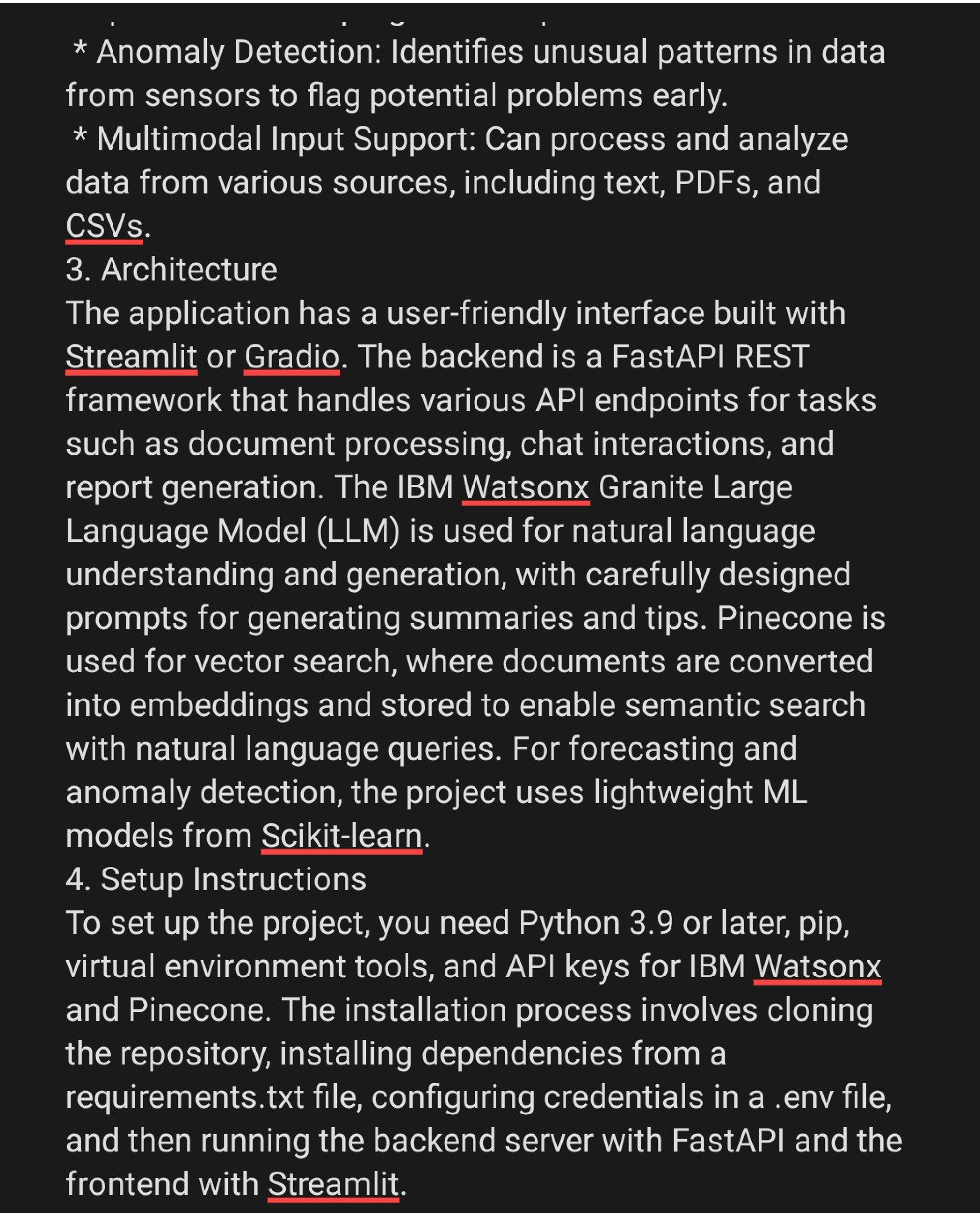
* Project Title: Sustainable Smart City Assistant
* Team Leader: Mr. Ibrahim Mohemmed Afsal
* Team Members: Mr. Sibiraj, Mr. Manikandan, Mr. Sainath, Mr. Anand

2. Project Overview

The Sustainable Smart City Assistant uses AI and real-time data to help cities and their residents become more eco-conscious and connected. It helps optimize resources like energy, water, and waste, and offers citizens personalized sustainability tips. For city officials, the assistant provides tools for strategic planning, including insights, forecasting, and policy summaries. The project's goal is to connect technology, governance, and community engagement to create greener, more efficient cities.

The assistant's key features include:

* Conversational Interface: Allows users to interact with the assistant using natural language.
* Policy Summarization: Converts long government documents into clear, actionable summaries.
* Resource Forecasting: Uses historical and real-time data to predict future resource usage for energy, water, and waste.
* Eco-Tip Generator: Gives personalized sustainability advice to users based on their behavior.
* Citizen Feedback Loop: Gathers and analyzes public input to help with city planning and service improvements.
* KPI Forecasting: Projects key performance indicators to



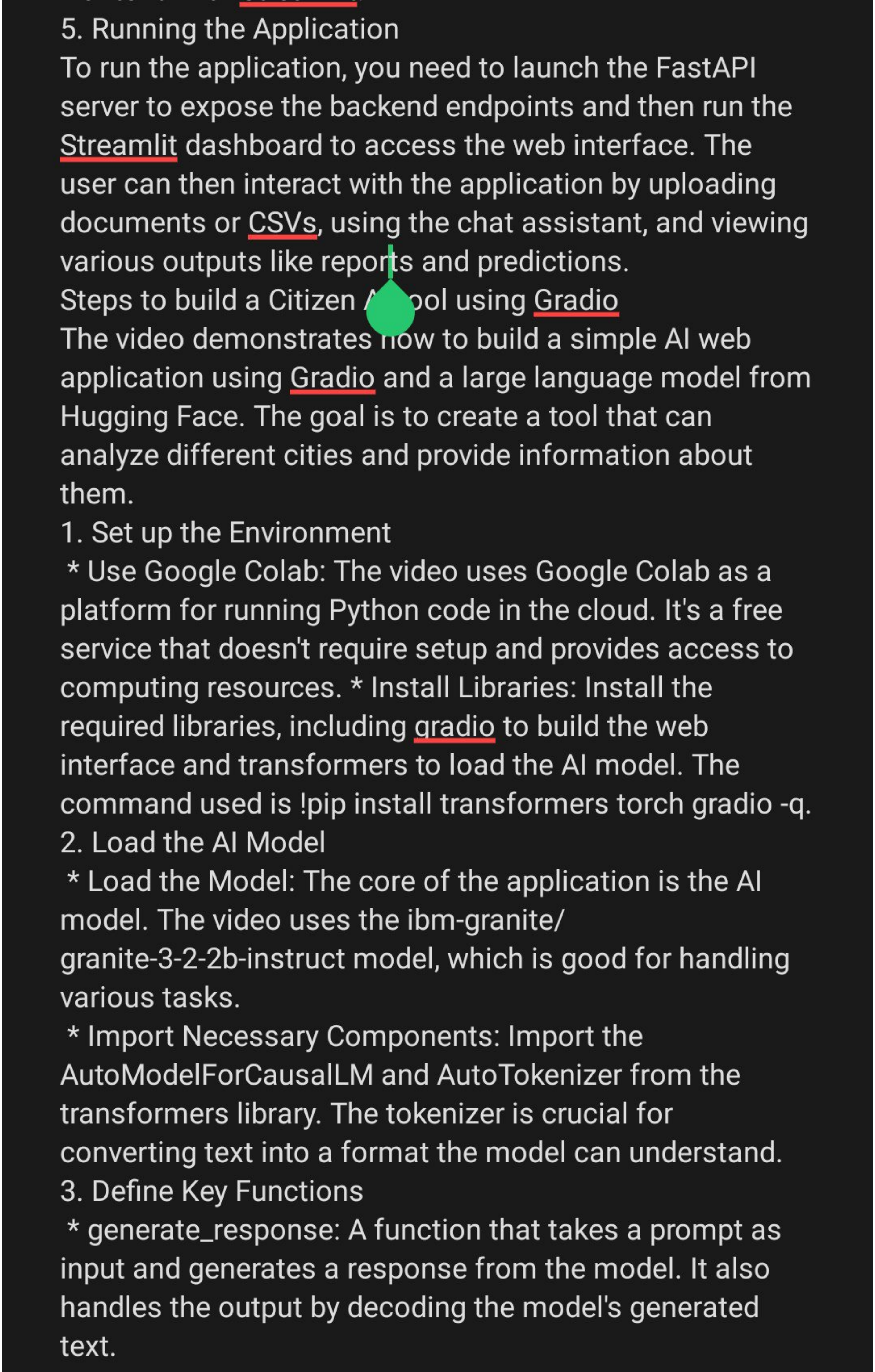
* Anomaly Detection: Identifies unusual patterns in data from sensors to flag potential problems early.
* Multimodal Input Support: Can process and analyze data from various sources, including text, PDFs, and CSVs.

3. Architecture

The application has a user-friendly interface built with Streamlit or Gradio. The backend is a FastAPI REST framework that handles various API endpoints for tasks such as document processing, chat interactions, and report generation. The IBM Watsonx Granite Large Language Model (LLM) is used for natural language understanding and generation, with carefully designed prompts for generating summaries and tips. Pinecone is used for vector search, where documents are converted into embeddings and stored to enable semantic search with natural language queries. For forecasting and anomaly detection, the project uses lightweight ML models from Scikit-learn.

4. Setup Instructions

To set up the project, you need Python 3.9 or later, pip, virtual environment tools, and API keys for IBM Watsonx and Pinecone. The installation process involves cloning the repository, installing dependencies from a requirements.txt file, configuring credentials in a .env file, and then running the backend server with FastAPI and the frontend with Streamlit.



5. Running the Application

To run the application, you need to launch the FastAPI server to expose the backend endpoints and then run the Streamlit dashboard to access the web interface. The user can then interact with the application by uploading documents or CSVs, using the chat assistant, and viewing various outputs like reports and predictions.

Steps to build a Citizen AI Tool using Gradio

The video demonstrates how to build a simple AI web application using Gradio and a large language model from Hugging Face. The goal is to create a tool that can analyze different cities and provide information about them.

1. Set up the Environment

* Use Google Colab: The video uses Google Colab as a platform for running Python code in the cloud. It's a free service that doesn't require setup and provides access to computing resources. * Install Libraries: Install the required libraries, including gradio to build the web interface and transformers to load the AI model. The command used is !pip install transformers torch gradio -q.

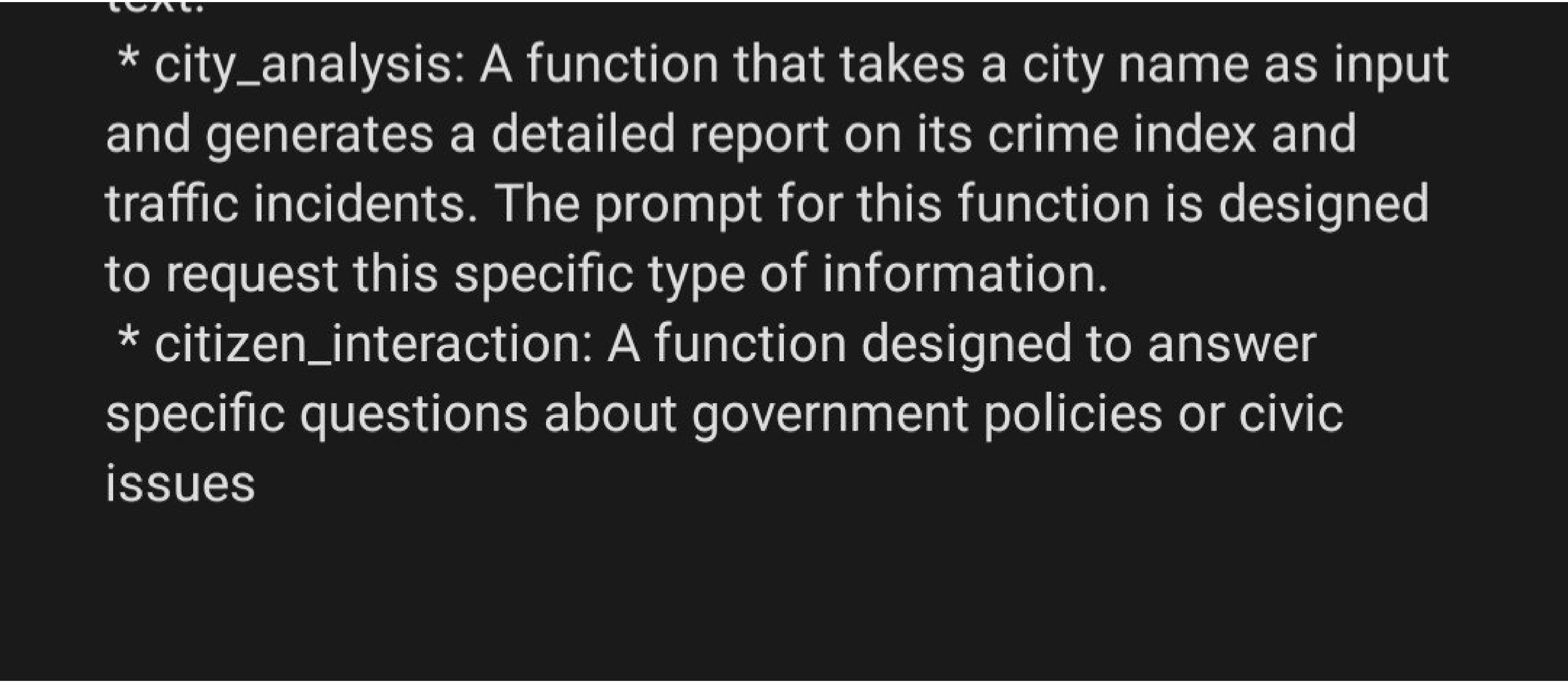
2. Load the AI Model

* Load the Model: The core of the application is the AI model. The video uses the ibm-granite/granite-3-2-2b-instruct model, which is good for handling various tasks.

* Import Necessary Components: Import the AutoModelForCausalLM and AutoTokenizer from the transformers library. The tokenizer is crucial for converting text into a format the model can understand.

3. Define Key Functions

* generate_response: A function that takes a prompt as input and generates a response from the model. It also handles the output by decoding the model's generated text.



text:

* city_analysis: A function that takes a city name as input and generates a detailed report on its crime index and traffic incidents. The prompt for this function is designed to request this specific type of information.
* citizen_interaction: A function designed to answer specific questions about government policies or civic issues