Sustainable Smart City Assistant

1. Introduction

- Project Title: Sustainable Smart City Assistant
- Team Members:
  - Mounika Loya : UI Developer & AI Integration
  - Medimpudi Raj Vardhan : Research & Feature Implementation
  - Mathi Pashvitha : Data Handling & Backend Support
  - Meghavath Sreevalli : Testing, Debugging

2. Project Overview

Purpose:

The main purpose of this project is to automate common tasks that city administrators and citizens encounter — such as understanding complex documents, reporting issues, and interpreting large datasets. The Smart City AI Assistant bridges the gap between AI technology and real-world governance, helping cities operate more efficiently.

Key Features:

- **Document Summarization (PDFs)**
  Enables users to upload lengthy policy documents in PDF format and automatically generates concise summaries. This helps policymakers and citizens quickly understand key information without reading the entire document.
- **Multilingual Translation (Hindi, Telugu, Tamil, Bengali)**
  Supports translation of AI-generated answers or summaries into multiple regional languages. This breaks language barriers and makes information accessible to diverse communities within the smart city.
- **Voice-based Question Handling**
  Allows users to interact with the assistant using voice commands. The system converts speech to text, processes the query through the AI model, and provides relevant responses, making the tool accessible to users who prefer voice over typing.

- **KPI Forecasting using Machine Learning**
  Provides trend analysis and forecasting for city-related key performance indicators (KPIs) like air quality, electricity consumption, or traffic data. This helps city officials predict future scenarios and make informed decisions.
- **Anomaly Detection in City Data**
  Detects irregularities or sudden spikes in data that might indicate problems such as pollution surges, system failures, or unusual activity. This feature enables quick identification of issues requiring immediate attention.
- **Simple and Interactive Web Interface (Streamlit-based)**
  Built with Streamlit, the application offers a clean, intuitive, and interactive web-based interface. Users can easily navigate between modules without technical knowledge, ensuring a smooth user experience.
- **Lightweight Deployment; Runs Without GPU Requirements**
  Designed to be resource-efficient, the assistant runs seamlessly on CPU environments without the need for heavy GPU setups. This makes it suitable for local use, lightweight servers, and educational or demo purposes.

3. Architecture

Frontend:

- Built using Streamlit, offering an intuitive, responsive web UI.
- Provides easy navigation via a sidebar with modules: Policy Assistant, Citizen Tools, and City Analytics.

Backend:

- Uses AI models and machine learning algorithms:
  - **Summarizer:** Hugging Face model google/flan-t5-base
  - **Embeddings:** Sentence Transformers (all-MiniLM-L6-v2)
  - **Translation:** googletrans
  - **Voice Recognition:** speech_recognition
  - **Forecasting:** Linear Regression from scikit-learn
  - **Anomaly Detection:** Isolation Forest

Database:

- No persistent database in this version.

- Works entirely on user-uploaded files (PDFs for summarization and CSVs for KPI analysis).
- Temporary session-based data handling.

4. Setup Instructions

Prerequisites:

- Python 3.x
- Required Python libraries:
  - streamlit
  - transformers
  - sentence-transformers
  - scikit-learn
  - pandas
  - PyPDF2
  - googletrans==4.0.0-rc1
  - speechrecognition
  - python-dotenv (for future environment variables)

Installation Steps:

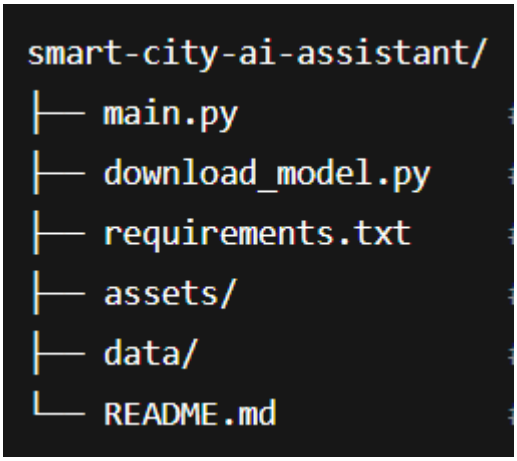
```
git clone https://github.com/pashvitha/smart-city-ai-assistant.git

cd smart-city-ai-assistant

pip install -r requirements.txt

streamlit run main.py
```

5. Folder Structure:



1) main.py

- This is the main entry point of the project.
- It contains the Streamlit code for the user interface and handles all three modules:
    - Policy Assistant (Summarization)
    - Citizen Tools (Voice, Translation, Query Answering)
    - City Analytics (Forecasting & Anomaly Detection)
- Manages user inputs, interactions, and displays outputs.

2) **download_model.py**

- Responsible for downloading and loading the Hugging Face FLAN-T5 model used for summarization.
- Initializes the NLP pipeline needed for text generation tasks.
- Keeps model loading separate from UI logic for cleaner code organization.

3.**requirements.txt**

- Lists all the Python libraries required to run the project, including:
    - streamlit, transformers, sentence-transformers, scikit-learn, pandas, PyPDF2, googletrans, speechrecognition, python-dotenv, and torch.
- You can install all dependencies using:

  pip install -r requirements.txt

4. assets/

- Stores static files like:
    - Images (e.g., app screenshots, icons, banners)
    - Logos or branding materials
- Useful for documentation, the README file, or enhancing the app's UI.

5.data/

Holds sample datasets such as:

- PDF files for testing the summarization feature.
- CSV files for KPI forecasting and anomaly detection.

- Useful for testing or demonstration purposes without requiring users to upload their own files.

6.README.md

- Markdown file that serves as the landing page on GitHub.
- Contains detailed project information including:
    - Project Description
    - Features List
    - Installation Instructions
    - Usage Guide
    - Screenshots or Demo Links
    - Future Enhancements or Contribution Guidelines

6. Running the Application

Command to Run:

streamlit run main.py

Access:

- The app will automatically open in your browser at http://localhost:8501 after running the above command.

How it Works:

- The user uploads PDFs or CSV files depending on the module.
- The backend models process the input (summarization, translation, forecasting).
- The output is displayed directly in the Streamlit web interface.

7. API Documentation

This application is interface-driven and does not expose REST APIs in the current version. However, backend functions handle:

- **Summarization** :

  Processes extracted text from PDF documents and generates concise summaries using the Hugging Face FLAN-T5 NLP model. This helps users quickly understand lengthy policies or reports.

- **Translation** :

  Converts AI-generated answers, summaries, or user inputs into selected languages **(**Hindi, Telugu, Tamil, Bengali**)** using the googletrans API, enhancing accessibility for regional language speakers.

- **Voice input processing** :

  Captures user voice via microphone and converts speech into text using the speech_recognition library. The converted text is then processed by the AI to generate relevant responses.

- **Data forecasting** :

  Uses Linear Regression to analyze CSV files containing time-based KPI data and predicts future values, helping city administrators monitor trends like pollution or energy usage.

- **Anomaly detection** :

  Employs the Isolation Forest algorithm to detect unusual spikes or drops in city data from CSV files. It identifies outliers that may indicate errors, emergencies, or irregularities.

8. Authentication

- Currently, there is no authentication as this is a local app intended for demo/testing.
- In future versions, authentication can be added to secure access, including user accounts, role-based permissions, and token/session-based security.

9. User Interface

Policy Assistant

This module allows users to upload a PDF document, extract its text, and generate a summarized version using the AI model. The interface includes a file uploader, a text preview box for extracted content, and a "Summarize" button. The output summary is displayed neatly below for easy reading. This module simplifies the process of understanding lengthy policy documents.

Citizen Tools

The Citizen Tools module is designed for interactive engagement. Users can enter questions via text or use voice input, which the app converts into text. It can generate eco-friendly tips, answer queries, and provide translations of the output into regional languages like Hindi, Telugu, Tamil, and Bengali. The interface includes input boxes, a microphone button, and dropdowns for language selection, ensuring an inclusive and accessible experience.

City Analytics

This module lets users upload CSV files containing time-based KPI data (Date, Value). It displays a line chart of the data, predicts the next KPI value using forecasting, and highlights anomalies detected in the dataset. The interface includes file uploaders, result displays for forecasted values, and data tables for anomalies, providing city administrators with a clear and data-driven view of trends and irregularities.

Common UI Features Across Modules:

- Sidebar for seamless navigation between modules.
- Clear buttons like Summarize, Submit Issue, Ask, and Generate Tips.

- Input fields, file uploaders, language selectors, and output boxes.
- Responsive design with user feedback like success messages and error handling.

10. Testing :

Testing Strategy

The testing strategy focused on validating the functionality, performance, usability, and robustness of the Smart City AI Assistant. The approach included the following types of testing:

1. **Functional Testing**

   Verified that each module works according to the requirements.

   Ensured correct outputs for summarization, translation, voice recognition, forecasting, and anomaly detection.

2. **Performance Testing**

   Measured the response time and efficiency of AI models, file processing, and data handling.

   Checked how the system performs with large PDFs, big CSV datasets, and multiple simultaneous tasks.

3. **Usability Testing**

   Ensured that the Streamlit interface is intuitive, easy to navigate, and user-friendly for both technical and non-technical users.

4. **Error Handling and Validation Testing**

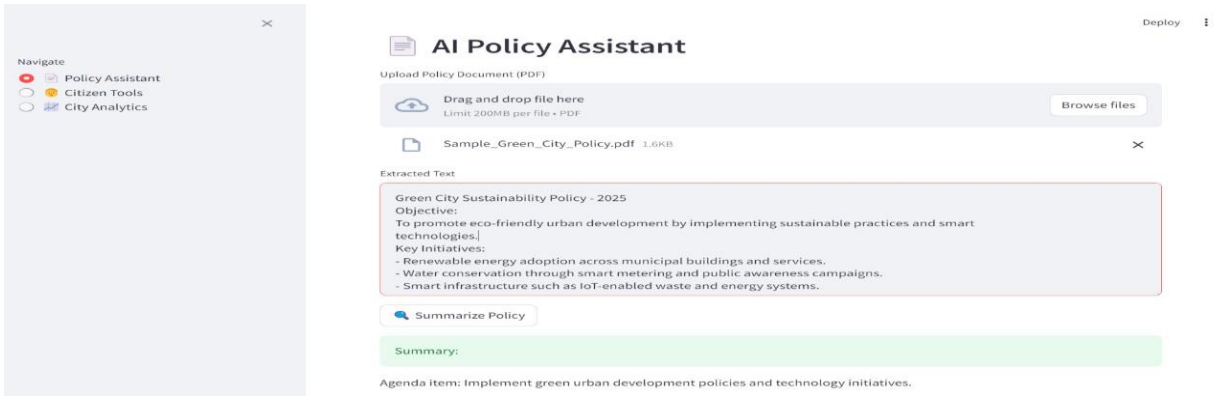
   Tested the system's response to invalid inputs such as unsupported file formats, empty files, or API failures (like translation or voice errors).

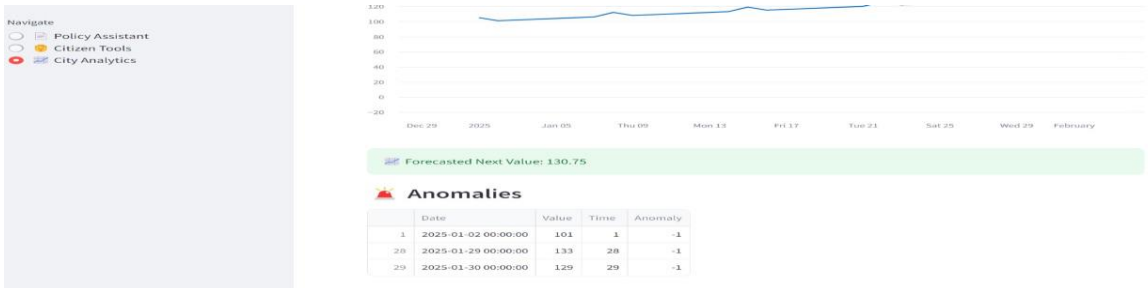
   Confirmed that the app displays proper error messages without crashing.

5. **Stress Testing (Manual)**

   Checked the system behavior when multiple actions were performed quickly or when large files were uploaded to ensure stability.

11. Screenshots for Demo:





12. Known Issues

- Voice recognition accuracy drops in noisy environments or with unclear pronunciation.
- Translation API (googletrans) occasionally fails if Google's free API rate limits are exceeded or if there are connectivity issues.
- Linear Regression forecasting struggles with highly volatile, seasonal, or non-linear data trends, leading to less accurate predictions.
- PDF summarization fails for scanned documents or image-based PDFs since OCR functionality is not currently implemented.
- Anomaly detection may produce false positives in very small datasets or datasets with natural high variability.
- The application requires a stable internet connection for translation and speech recognition; these features do not work offline.

13. Future Enhancements

- **Add OCR functionality for scanned PDFs**
  To enable the assistant to process image-based PDFs and scanned documents, OCR integration will allow accurate text extraction, expanding the utility for government documents, handwritten forms, and non-digital records.

- **Implement Text-to-Speech (TTS) for audible AI responses**
  Integrating TTS will allow the assistant to speak responses, making it more accessible to visually impaired users, elderly citizens, or users who prefer auditory feedback over reading text.

- **Upgrade forecasting to use Prophet, ARIMA, or deep learning-based models like LSTM**
  Advanced time-series models will improve accuracy for non-linear, seasonal, and complex data patterns. This will make forecasts more reliable for city metrics like pollution, traffic, and energy consumption.

- **Enable cloud deployment for multi-user online access**
  Cloud hosting on platforms like AWS, Azure, or Google Cloud will allow the assistant

to be accessible from anywhere, support concurrent users, ensure data persistence, and scale as needed for large city operations.

- **Integrate authentication for secured usage**
  Implementing user authentication and role-based access will ensure that sensitive data and administrative controls are accessible only to authorized users, improving data security and privacy.

- **Add Pinecone/FAISS for vector-based document search and memory in conversations**
  Enabling vector search will allow the assistant to remember previous queries, retrieve relevant past documents, and provide smarter, context-aware responses. This will be key for building conversational memory.

- **Create a mobile-friendly version or a mobile app**
  Developing a mobile version will make the assistant easily accessible for citizens and administrators on smartphones and tablets, improving convenience and allowing real-time access to features.

- **Real-time notifications and alerts**
  The system will send alerts via SMS, email, or app notifications whenever anomalies are detected in city data, helping administrators respond faster to issues like pollution spikes or equipment failures.

- **Advanced data visualization dashboards**
  Adding interactive charts, graphs, heatmaps, and data filters will help users better understand trends, anomalies, and forecasts, leading to more informed decisions.

- **Integration with IoT and smart sensors**
  Connecting the assistant to real-time IoT devices such as air quality monitors, traffic cameras, or energy meters will allow for instant data analysis and insights, further enhancing smart city operations.

- **Offline support for key features**
  Allowing summarization and KPI analysis to work offline will improve reliability in areas with poor internet connectivity, ensuring uninterrupted functionality.

- **Multi-language chatbot with conversation history**
  Developing an intelligent chatbot with multi-language support and memory of past conversations will enhance the user experience, making interactions more natural and efficient.

- **API development for third-party integration**
  Exposing internal functions (summarization, forecasting, anomaly detection) as APIs will enable integration with other smart city platforms, mobile apps, and government systems.