

Full Stack Project Documentation – Sustainable Smart City Assistant AI

1. Introduction

Project Title: Sustainable Smart City Assistant AI using IBM granite LLM

Team Members:

- Sai Prashanth (Frontend + Voice Interface)
- Abhilash (AI Model & Integration)
- Devendra (Backend & Database)
- Sai Madhan (Deployment & Testing)

2. Project Overview

Purpose:

To empower citizens and smart city administrators with a conversational AI that provides personalized sustainability recommendations, real-time alerts, service access, and predictive insights.

Features:

1. **CityUpdates:**

- * User enters city name → AI returns local updates (e.g., events, temperature, news)
- * Prompt includes local context and formatting expectations (JSON-style)

2. **ChatAssistant:**

- * Free-form input from user → General AI response
- * Can answer questions like "Where's the nearest water park in this city?"

3. **EcoTips:**

- * No input needed → Returns a unique eco-friendly suggestion with emoji

- * Example: "Use refillable water bottles instead of plastic!"

4.**KPIForecast:**

- * Input: A series of numerical KPI data

- * Output: Forecasted next 3 values, structured in JSON

5.**GrammarCorrection:**

- * Input: Paragraph or sentence

- * Output: Corrected version, returned by model

6.**LiveWeatherReport:**

- * Input: City name

- * Output: AI-generated weather description + temperature (e.g., "Sunny, 30°C")

7.**PolicyCreator:**

- * Input: Keyword + existing policy content

- * Output: AI-summarized policy draft

8.**FeedbackSubmission:**

- * Input: User name + message

- * Output: Stores in SQLite and confirms submission

3. Architecture

Frontend:	Streamlit (styled with custom CSS, responsive cards)
Backend:	Python (FastAPI), SQLite (for feedback storage)
AI or `Model	IBM Granite-3.3-2b.assist
Deployment:	Streamlit Cloud frontend + FastAPI via ngrok
External APIs:	None used explicitly (weather and updates generated via LLM)

4. Setup Instructions

1. Clone the Repository

git clone <https://github.com/prashAnTH630490/Smart-City-assistant.git>

cd Smart-City-assistant/backend

2. Create & Activate Virtual Environment

python -m venv venv

venv\Scripts\activate **# On Windows**

or

source venv/bin/activate

On Linux/Mac


3. Install Dependencies

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

4. Run FastAPI Server

```
uvicorn main:app --reload
```

Access it via:

 <http://localhost:8000/docs> (Swagger UI)

5. Folder Structure

|— main.py # FastAPI backend

|— requirements.txt # Backend Python dependencies

|— .env # Optional environment variables

|— .gitignore # Ignore venv, pycache, etc.

|— README.md # Backend-specific documentation

|— app.py # Main Streamlit application

|— README.md # Frontend-specific documentation

|— render.yaml # Deployment config for Render (backend)

6. Running the Application

Run FastAPI Server & Streamlit server

uvicorn main:app --reload

streamlit run app.py

7. API Documentation

Endpoints:

/chat	AI chat assistant via IBM Granite
/text-correction	Fixes grammar & spelling
/climate-update	Real-time weather and air quality
/place-recommendation	City travel & tourism suggestions
/smartcity-data	Urban data: crime, oxygen, traffic, etc.
/policy-generator	Draft city policies with AI

Example Request:

```
```json
POST /api/chatbot
{ "query": "What is the air quality near me?" }
Response: { "response": "AQI in your zone is Moderate (AQI 72)" }
```
```

8. Authentication

JWT token-based auth

- Login, signup endpoints
- Access roles: citizen, admin
- Middleware: verifyToken, isAdmin

9. User Interface

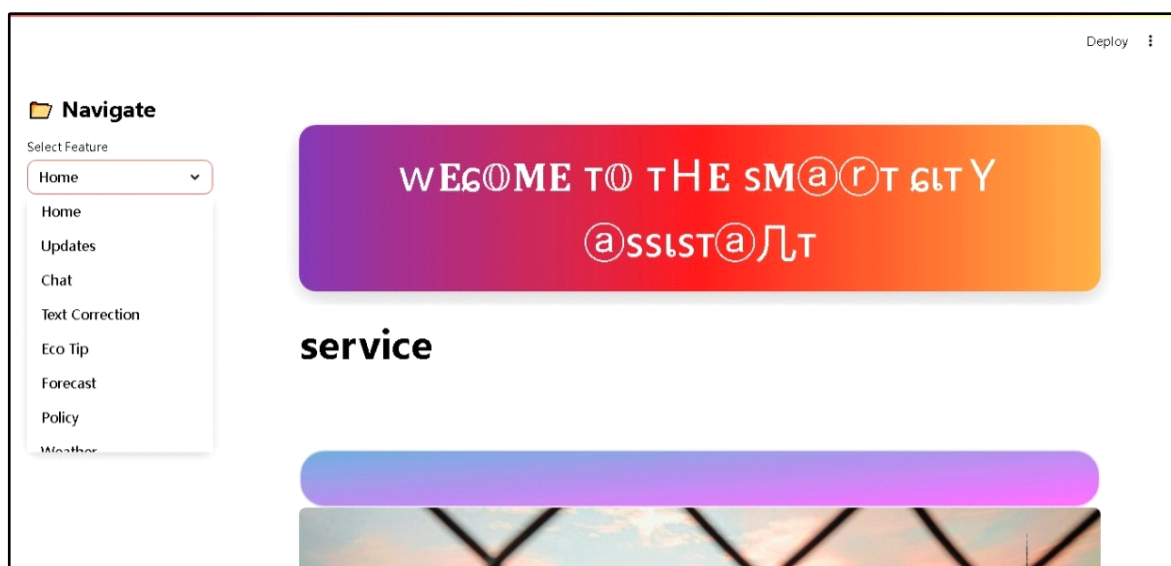
- Home dashboard with metrics
- Chatbot panel (text + voice input)
- Admin panel with usage and feedback stats
- city updates
- KPI forecast
- live weather
- eco tips
-

10. Testing

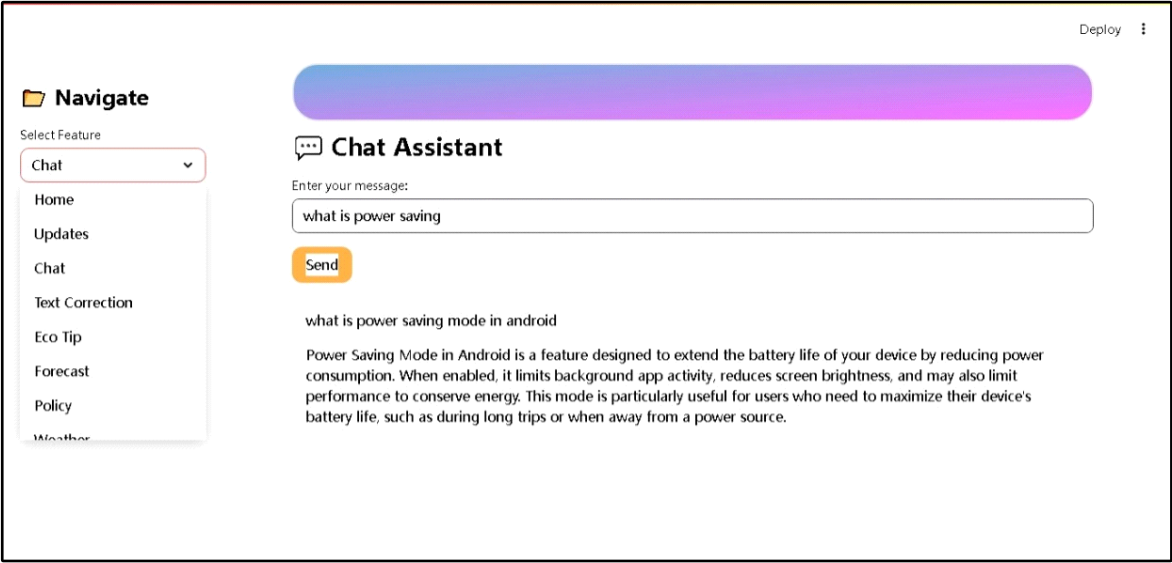
- Unit Testing: Jest (FastAPI), Rest Testing Library
- Integration: Postman for API tests
- Load Testing: Apache JMeter
- Manual UI/UX walkthroughs

11. Screenshots / Demo

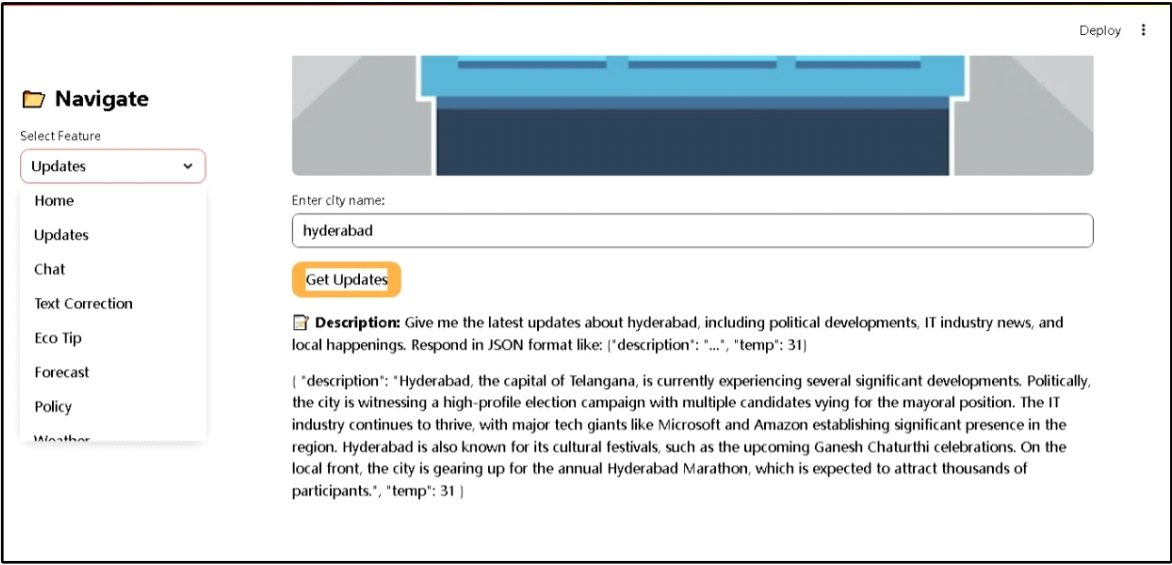
Dashboard UI



Chatbot Panel



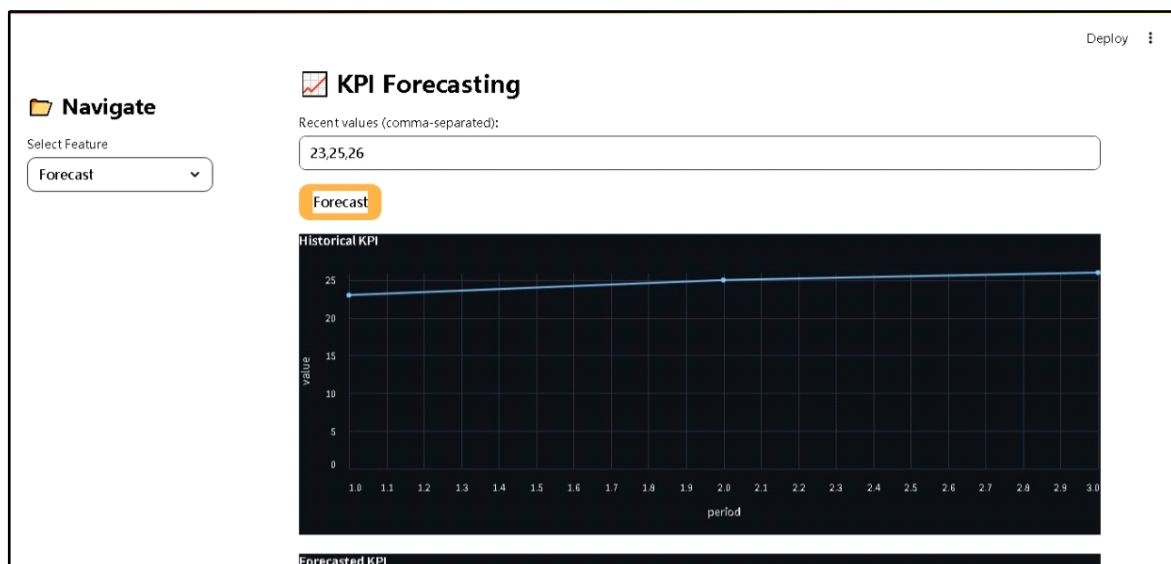
city updates



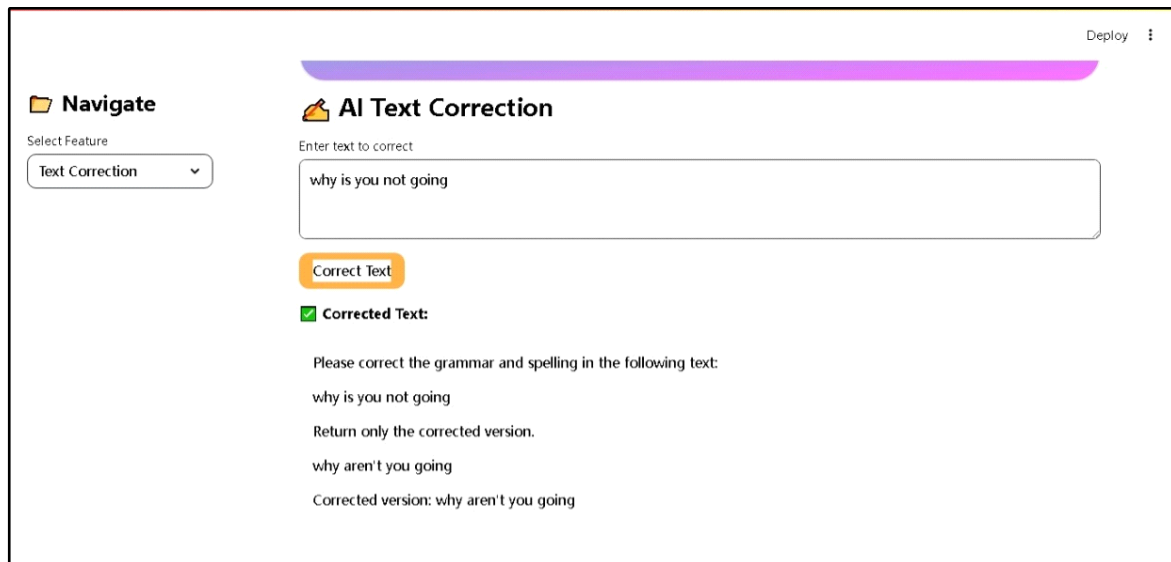
eco tips



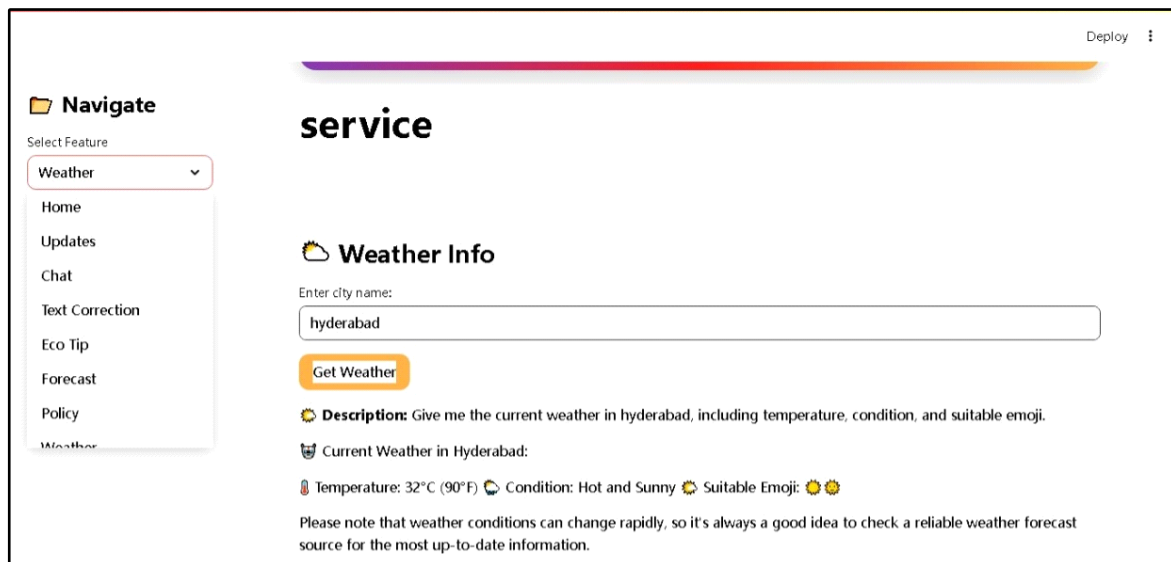
KPI forecast



grammer correction



live weather



policy summery

Deploy

Navigate

Select Feature

Policy

▼

Home

Updates

Chat

Text Correction

Eco Tip

Forecast

Policy

Weather

Policy Summary

Query keyword:

health insurance

Policy text:

it helps to stand still in emergency situations

Summarize

Summarize the following city policy focusing on 'health insurance': it helps to stand still in emergency situations, providing immediate medical attention and support. it ensures that all residents have access to affordable health insurance options. it promotes preventive healthcare measures and regular health screenings. it encourages community health initiatives and partnerships with local healthcare providers. it offers financial assistance for low-income families to cover healthcare costs. it establishes a city-wide health information system for transparent and efficient healthcare management.

The city policy aims to enhance overall community health by prioritizing health insurance accessibility, affordability, and preventive care. It emphasizes the importance of immediate medical support in emergencies, promotes preventive

Feedback Submission

Deploy

Navigate

Select Feature

Feedback

▼

Submit Feedback

Your Name:

prashanth

Your feedback:

this app is greater than ☺

Submit

All Feedback

gopi: super

[WebApp Link] <https://smart-city-assistant-1.onrender.com>

[GitHub Repo] <https://github.com/prashanth630490/Smart-City-assistant>

12. Known Issues

- Some NLP queries misinterpret regional phrasing
- Hugging face API fails occasionally in low-bandwidth zones
- UI under testing for performance

13. Future Enhancements

- Smart Home IoT integration
- Voice alert system for seniors
- Predictive outage alerts