Project Documentation: Sustainable Smart City Assistant

# 1. Introduction

• Project Title: Sustainable Smart City Assistant

• Team Members:

- Mullamuri Tirupathi Naidu– Frontend Developer (Streamlit UI)  
 - P . Viswasree Akhilesh – Backend Developer (FastAPI & AI Integration)  
 - Pecheti Tejassu & Mule Lakshmi – Testing & Documentation

# 2. Project Overview

• Purpose:  
 To provide citizens with an AI-powered assistant that offers eco-friendly suggestions, smart city tips, reports, and query responses through a chat interface.

• Features:  
 - Chat-based assistant for sustainability queries  
 - Eco-tips module for water, energy, waste management  
 - Report generator for summaries and KPIs  
 - Admin dashboard (optional)

# 3. Architecture

• Frontend (Streamlit):  
 Built using Streamlit to provide a minimal, fast, and responsive UI hosted via local/Streamlit Cloud.

• Backend (FastAPI):  
 Modular API endpoints (/chat, /tips, /report) for processing input, calling Hugging Face models, and formatting responses.

• Database:  
 - Lite version: Local JSON/CSV logs for queries  
 - Extended version: MongoDB or Firebase for persistent logs

# 4. Setup Instructions

• Prerequisites:  
 - Python 3.10+  
 - pip  
 - Internet (for Hugging Face API)  
 - MongoDB (optional)

• Installation:

# Clone the repository  
[Akhileshp123/Sustainable-Smart-City-Assistant-Using-IBM-Granite-LLM: 🌆 Sustainable Smart City Assistant An AI-powered assistant that provides eco-tips, policy summaries, KPI insights, and citizen feedback handling using IBM Granite LLM. Built with FastAPI and Streamlit, this tool helps drive smarter, greener urban solutions.](https://github.com/Akhileshp123/Sustainable-Smart-City-Assistant-Using-IBM-Granite-LLM)# Install backend dependencies  
pip install -r app/requirements.txt  
  
# Set up environment variables  
export HF\_API\_KEY=your\_huggingface\_key

# 5. Folder Structure

• Client (Streamlit UI):

frontend/  
├── dashboard.py  
├── components/  
│ ├── chat\_box.py  
│ ├── report\_viewer.py

• Server (FastAPI Backend):

app/  
├── main.py  
├── api/  
│ ├── chat\_router.py  
│ ├── tips\_router.py  
│ └── report\_router.py  
├── services/  
│ └── huggingface\_service.py

# 6. Running the Application

• Frontend:  
 streamlit run frontend/dashboard.py

• Backend:  
 uvicorn app.main:app --reload

# 7. API Documentation

Accessible via Swagger UI:  
http://127.0.0.1:8000/docs

- GET /tips – Get eco suggestions  
- POST /chat – Submit a query  
- POST /report – Generate summary reports

# 8. Authentication

Currently, this is a public demo.

# 9. User Interface

Clean, tab-based layout (Chat, Eco Tips, Report)  
Screenshot Example:  
- [Insert Screenshot of Chat Interface]

# 10. Testing

• Tools Used: pytest, Postman, Streamlit debug console

• Strategy: Unit tests for service layer, manual UI tests for integration.

# 11. Screenshots or Demo12. Known Issues

- Hugging Face API latency may vary  
- Some queries may return generic answers depending on model context of ibm-granite/granite-3.3-8b-instruct

# 13. Future Enhancements

- User login/authentication  
- Voice input integration  
- Feedback and rating system  
- Admin analytics dashboard  
- Mobile responsiveness