SmartSDLC – AI-Enhanced Software Development Lifecycle

Sustainable Smart City Assistant Using

IBM Granite LLM

Team Leader: M. Navin kumar

Team Members: S. Santhosh, A. Jayan Raman, A. Vasanth kumar

1. Introduction

Project Title: SmartSDLC – AI-Enhanced Software Development Lifecycle

This project leverages the capabilities of IBM Granite LLM to build an AI -powered assistant designed for sustainable smart cities. It aims to improve urban management, enhance citizen services, and promote sustainability by using advanced AI -driven insig hts.

2. Project Overview

Purpose:

The Sustainable Smart City Assistant project is intended to provide intelligent, real -time solutions to urban challenges such as traffic management, waste disposal, energy optimization, and citizen engagement. By integra ting with IoT devices, environmental sensors, and city infrastructure, the system ensures efficient resource utilization and sustainable urban growth.

Features:

- AI-powered decision support for smart governance
- Real -time traffic and transport optimiz ation
- Smart energy monitoring and predictive consumption analysis
- Waste management insights through data -driven predictions
- Citizen engagement chatbot for queries and service requests
- Integration with IoT devices and city -wide sensor networks
- Secure and responsible data handling with role -based access

3. Architecture

The architecture of the Smart City Assistant is built on a layered framework:

- Data Layer: Collects data from IoT sensors, cameras, environmental monitors,

and government records.

- Preprocessing Layer: Cleans and transforms raw data for AI analysis.

- AI/LLM Layer: Uses IBM Granite LLM for natural language understanding,

predictions, and smart recommendations.

- API/Service Layer: Provides secure endpoints for city applications and dashboards.

- Application Layer: Dashboards for administrators, mobile apps for citizens,

and control panels for city planners.

- Security & Compliance Layer: Ensures encryption, authentication, and compliance

with smart city regulations.

- Moni toring Layer: Continuous feedback and improvements for system performance.

4. Setup Instructions

- Install Python, AI libraries, and configure IBM Granite LLM API access.
- Set up IoT/sensor data pipelines for real -time input.
- Configure APIs with Flask/F astAPI for backend integration.
- Build front -end dashboards with React or mobile apps for citizen interaction.
- Containerize with Docker for deployment across city servers.
- Use environment variables and secure authentication for data safety.

5. Folder Structure

- data/ : raw, processed, and external smart city datasets
- notebooks/ : Jupyter notebooks for experiments and analysis
- src/ : core source code (data_preprocessing, models, prediction, nlp, utils)
- api/ : backend APIs for city services
- mode ls/ : trained AI and LLM models

- tests/ : unit and integration testing

- configs/ : environment and system configuration

- logs/ : runtime and system logs

- Dockerfile : containerization instructions

- requirements.txt : dependencies

- README.md : project overview

- .env : environment variables and API keys

6. Running the Application

- Preprocess sensor and city datasets using preprocessing scripts.

- Train or fine- tune AI/LLM models with available datasets.

- Launch backend APIs with Flask/FastAPI to serv e predictions.

- Start dashboards or mobile frontends for citizens and administrators.

- Deploy using Docker containers for scalability across servers.

- Test APIs and monitor logs to ensure system reliability.

7. API Documentation

The API provides endpoints for:

- Traffic and transport optimization requests

- Energy usage analysis and predictions

- Waste management recommendations

- Citizen chatbot queries and responses

- IoT sensor data integration

APIs support authentication with keys or tokens and retu rn structured JSON responses.

8. Authentication

The system uses OAuth2.0/JWT for authentication with role -based access.

- Admins: Access to dashboards and configurations.

- Citizens: Chatbot queries and service requests.

- Operators: IoT device and infrastructure monitoring.

All communications are secured with HTTPS encryption.

9. User Interface

The UI includes:

- City administrator dashboards with insights and visualizations.
- Citizen chatbot accessible via web and mobile apps.
- Real -time maps for traff ic and energy monitoring.
- Notifications and alerts for city services.

The interface is responsive, user -friendly, and multilingual.

10. Testing

Testing includes:

- Unit testing for preprocessing and API logic.
- Integration testing with IoT and city data sources.
- Performance testing for scalability across servers.
- Security testing for vulnerabilities.
- User acceptance testing with city officials and citizens.

Conclusion

The Sustainable Smart City Assistant project demonstrates how IBM Granite LLM

can be harnessed to create a sustainable, efficient, and citizen- friendly urban ecosystem.

By combining IoT data, AI -driven predictions, and user -centric interfaces, the system

offers solutions for energy efficiency, waste management, traffic optimizati on,

and citizen engagement. This project sets the foundation for future AI -powered smart cities.

THANK YOU