Smart SDLC Change File

# Requirement Analysis

* + Purpose: Empower cities with AI-driven Sustainable Smart City Assistant.
  + Stakeholders: Citizens, City Officials, Researchers.
  + FunctionalRequirements:ConversationalInterface,PolicySummarization,Forecasting, Eco-Tip Generator, Feedback Loop, Anomaly Detection, KPI Forecasting.
  + Non-Functional Requirements: Scalability, Security, Usability, Real-time Processing.

**TEAM LEADER:** VIGNESHVARAN P

**TEAM MEMBER:**THOMAS EDWARD Y

**TEAM MEMBER:**UPPILIRAJ G

**TEAM MEMBER:**VISAKAN R

**TEAM MEMBER:**SURYA S

# System Design

* + Frontend: Streamlit for dashboards and chat.
  + Backend: FastAPI for API services.
  + LLM: IBM Watsonx Granite for NLP tasks.
  + Vector DB: Pinecone for semantic search.

# Implementation

* + Python 3.9+, FastAPI, Streamlit, Scikit-learn, Pinecone, IBM Watsonx Granite.
  + Folder structure: app/, ui/, scripts.

# API Development

* + Endpoints: /chat/ask, /upload-doc, /search-docs, /get-eco-tips, /submit-feedback.

# Testing

* + Unit tests for prompt functions and utilities.
  + API testing via Swagger/Postman.
  + Manual validation and edge case handling.

# Deployment

* + Steps: Clone repo, install dependencies, configure .env, run FastAPI and Streamlit.
  + Cloud-ready deployment with IBM Cloud and Pinecone.

# Authentication

* + Security via JWT, OAuth2, and role-based access.

# Maintenance

* + Continuous monitoring with anomaly detection and citizen feedback loop.

# Future Enhancements

* + User sessions, history tracking, advanced analytics, scalability improvements.

# Conclusion

* + This SDLC approach ensures a structured, scalable, and secure Smart City Assistant.

# Smart SDLC Change File - 10 Key Points (Summary)

* + 1. Requirement Analysis: Define purpose, stakeholders, requirements.
  + 2. System Design: Architecture (Frontend, Backend, LLM, DB).
  + 3. Module Design: Embedder, Forecaster, Anomaly Checker.
  + 4. Implementation: Python, FastAPI, Streamlit, Watsonx, Pinecone.
  + 5. API Development: Chat, Upload, Search, Eco-Tips, Feedback.
  + 6. Testing: Unit tests, API validation, manual checks.
  + 7. Deployment: Setup repo, environment, services.
  + 8. Authentication: JWT, OAuth2, role-based security.
  + 9. Maintenance: Monitoring, anomaly detection, feedback.
  + 10.Future Enhancements: Sessions, analytics, scalability.