# SmartSDLC-AI-Ehanced software development lifecycle

Project Title: SmartSDLC-AI-Enhanced  
Project Documentation  
  
1. Introduction  
• Project title: SmartSDLC-AI-Enhanced  
• Team member: shamera M

Team member:sindhu P

Team member:sathya S

Team member:soundariya S  
  
**2. Project Overview**Purpose:  
The purpose of a Sustainable Smart City Assistant is to empower cities and their residents to thrive in a more eco-conscious and connected urban environment. By leveraging AI and real-time data, the assistant helps optimize essential resources like energy, water, and waste, while also guiding sustainable behaviors among citizens through personalized tips and services. For city officials, it serves as a decision-making partner—offering clear insights, forecasting tools, and summarizations of complex policies to support strategic planning. Ultimately, this assistant bridges technology, governance, and community engagement to foster greener cities that are more efficient, inclusive, and resilient.  
 **Features:**- Conversational Interface (Natural language interaction)  
- Policy Summarization (Simplified policy understanding)  
- Resource Forecasting (Predictive analytics)  
- Eco-Tip Generator (Personalized sustainability advice)  
- Citizen Feedback Loop (Community engagement)  
- KPI Forecasting (Strategic planning support)  
- Anomaly Detection (Early warning system)  
- Multimodal Input Support (Flexible data handling)  
- Streamlit or Gradio UI (User-friendly interface)  
  
**3. Architecture**Frontend (Streamlit): Interactive dashboards, file uploads, chat interface, and feedback forms.  
Backend (FastAPI): Handles document processing, chat, eco tips, report creation, and vector embedding.  
LLM Integration (IBM Watsonx Granite): Natural language understanding and generation.  
Vector Search (Pinecone): Stores and searches policy document embeddings.  
ML Modules: Forecasting and anomaly detection.  
  
**4. Setup Instructions**Prerequisites:  
- Python 3.9 or later  
- pip and virtual environment tools  
- API keys for IBM Watsonx and Pinecone  
- Internet access for cloud services  
  
Installation Process:  
- Clone the repository  
- Install dependencies from requirements.txt  
- Create a .env file and configure credentials  
- Run the backend server using FastAPI  
- Launch the frontend via Streamlit  
- Upload data and interact with the modules  
  
**5. Folder Structure**app/ – Backend logic (routers, models, integrations)  
app/api/ – API routes for chat, feedback, report, document vectorization  
ui/ – Frontend components for Streamlit  
smart\_dashboard.py – Launches main dashboard  
granite\_llm.py – IBM Watsonx Granite communication  
document\_embedder.py – Document embeddings for Pinecone  
kpi\_file\_forecaster.py – Forecasts KPI trends  
anomaly\_file\_checker.py – Detects anomalies  
report\_generator.py – Generates sustainability reports  
 **6. Running the Application**- Launch FastAPI server to expose backend endpoints  
- Run Streamlit dashboard  
- Navigate through pages via sidebar  
- Upload documents, interact with chat, and view outputs like reports and forecasts  
  
**7. API Documentation**Endpoints:  
- POST /chat/ask  
- POST /upload-doc  
- GET /search-docs  
- GET /get-eco-tips  
- POST /submit-feedback  
  
**8. Authentication**Supports:  
- Token-based authentication (JWT or API keys)  
- OAuth2 with IBM Cloud credentials  
- Role-based access (admin, citizen, researcher)  
  
**9. User Interface**Minimalist design with sidebar navigation, KPI visualizations, chat, eco tips, forecasting tabs, and PDF download.  
  
10. Testing  
Unit testing, API testing, manual testing, and edge case handling.

**screenshot:**  
