AI Enhanced Software Development Lifecycle

Project Documentation

1. Introduction :

Project title :Ai Enhanced Software Development Lifecycle.

- Team member : GUBERAN M
- Team member : NAVANEESWAR N
- Team member : MOHAMMED IMRAN D

2. Project Overview:

- **Purpose:**. **AI Enhanced Software Development Lifecycle**, aims to empower city residents and officials by using AI and real-time data to create a more eco-conscious and connected urban environment. It helps to optimize resources like energy, water, and waste, and provides personalized eco-tips to encourage sustainable behaviors. For city officials, it's a decision-making tool that provides insights, forecasting capabilities, and summaries of complex policies. The project's goal is to connect technology, governance, and community to build more efficient, resilient, and greener cities

Features:

- **Conversational Interface**: Allows natural language interaction for citizens and officials to ask questions and get guidance.
- **Policy Summarization**: Converts long government documents into clear, actionable summaries.
- **Resource Forecasting**: Uses historical and real-time data to predict future usage of energy, water, and waste.
- **Eco-Tip Generator**: Recommends daily actions to help users reduce their environmental impact.
- **Citizen Feedback Loop**: Gathers and analyzes public input for city planning.
- **KPI Forecasting**: Projects key performance indicators for strategic planning.
- **Anomaly Detection**: Acts as an early warning system by identifying unusual patterns in data.
- **Multimodal Input Support**: Handles various data types, including text, PDFs, and CSVs.
- **User-friendly Interface**: An intuitive dashboard built with Streamlit or Gradio UI.
    - .

3. Architecture:

- **Frontend (Streamlit):** The frontend is an interactive web UI with multiple pages for dashboards, file uploads, a chat interface, feedback forms, and report viewers. It uses the Streamlit-option-menu library for sidebar navigation, and each page is modularized for scalability.
- **Backend (FastAPI):** This serves as the REST framework for API endpoints that handle document processing, chat, eco-tip generation, and more. It is optimized for asynchronous performance and easy Swagger integration.
- **LLM Integration (IBM Watsonx Granite):** The project uses Granite LLM models from IBM Watsonx for natural language understanding and generation. Prompts are specifically designed to produce summaries, reports, and sustainability tips.
- **Vector Search (Pinecone):** Uploaded policy documents are converted into embeddings using Sentence Transformers and stored in Pinecone. Semantic search is enabled via cosine similarity, letting users search documents using natural language queries.
- **ML Modules (Forecasting and Anomaly Detection):** Lightweight ML models from Scikit-learn are used for forecasting and anomaly detection. Time-series data is parsed, modeled, and visualized using pandas and matplotlib.

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 and configure a

    .env file with credentials.

  - Run the backend server using FastAPI.
  - Launch the frontend via Streamlit.
  - Upload data and interact with the modules.

5. Folder Structure:

- app/ - Contains all FastAPI backend logic, including routers, models, and integration modules.
- app/api/ - Subdirectory for modular API routes like chat, feedback, and document vectorization.
- ui/ - Contains frontend components for Streamlit pages and form UIs.
- smart_dashboard.py - The entry script for the main Streamlit dashboard.
- granite_llm.py - Handles all communication with the IBM Watsonx Granite model.
- document_embedder.py - Converts documents to embeddings and stores them in Pinecone.
- kpi_file_forecaster.py - Forecasts future trends for energy/water using regression.
- anomaly_file_checker.py - Flags unusual values in uploaded KPI data.
- report_generator.py - Constructs AI-generated sustainability reports.

6. Running the Application:

- To start the project, launch the FastAPI server and then run the Streamlit dashboard.
- Navigate through the pages using the sidebar.
- Users can upload documents or CSVs, interact with the chat assistant, and view outputs like reports, summaries, and predictions.
- All interactions are real-time, with the frontend dynamically updating via backend APIs.

7. API Documentation:

- The backend APIs include:
    - POST /chat/ask - Accepts a user query and returns an AI-generated message.
    - POST /upload-doc - Uploads and embeds documents in Pinecone.
    - GET /search-docs - Returns semantically similar policies to a user query.
    - GET /get-eco-tips - Provides sustainability tips on selected topics.
    - POST /submit-feedback - Stores citizen feedback.
- Each endpoint is documented and tested in Swagger UI.

8. Authentication:

- For demonstration purposes, this version of the project runs in an open environment.

- Secure deployments can include:
  - Token-based authentication (JWT or API keys).
  - OAuth2 with IBM Cloud credentials.
  - Role-based access for different user types (admin, citizen, researcher).
- Future enhancements will include user sessions and history tracking.

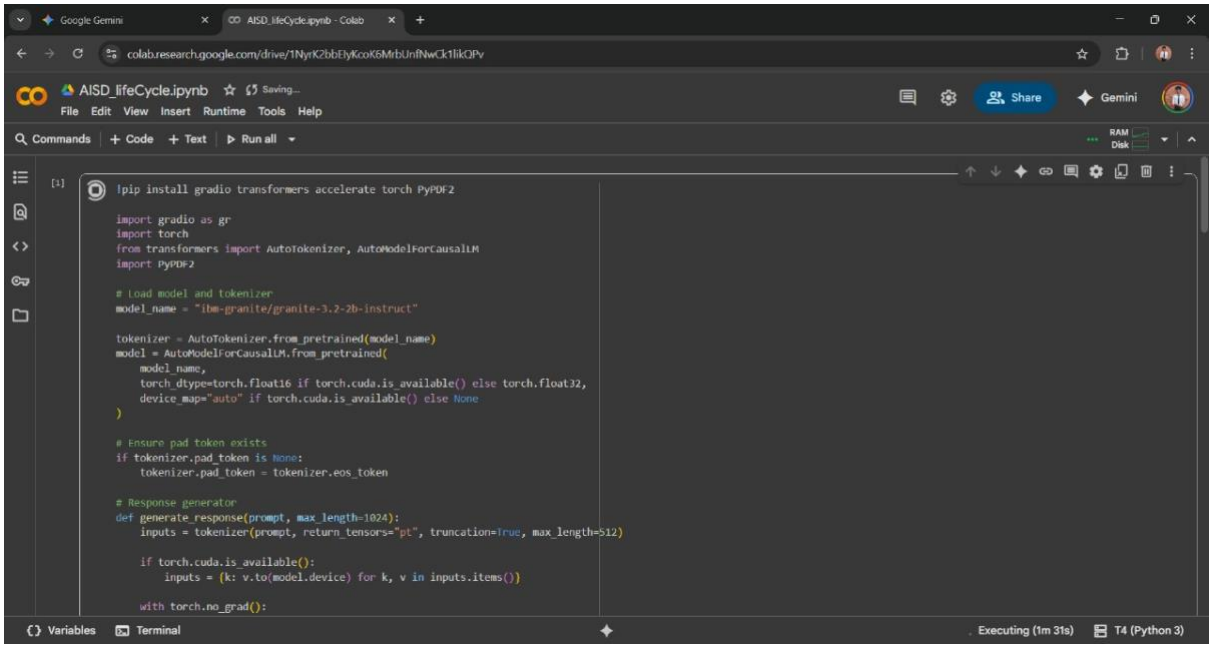
9. User Interface:

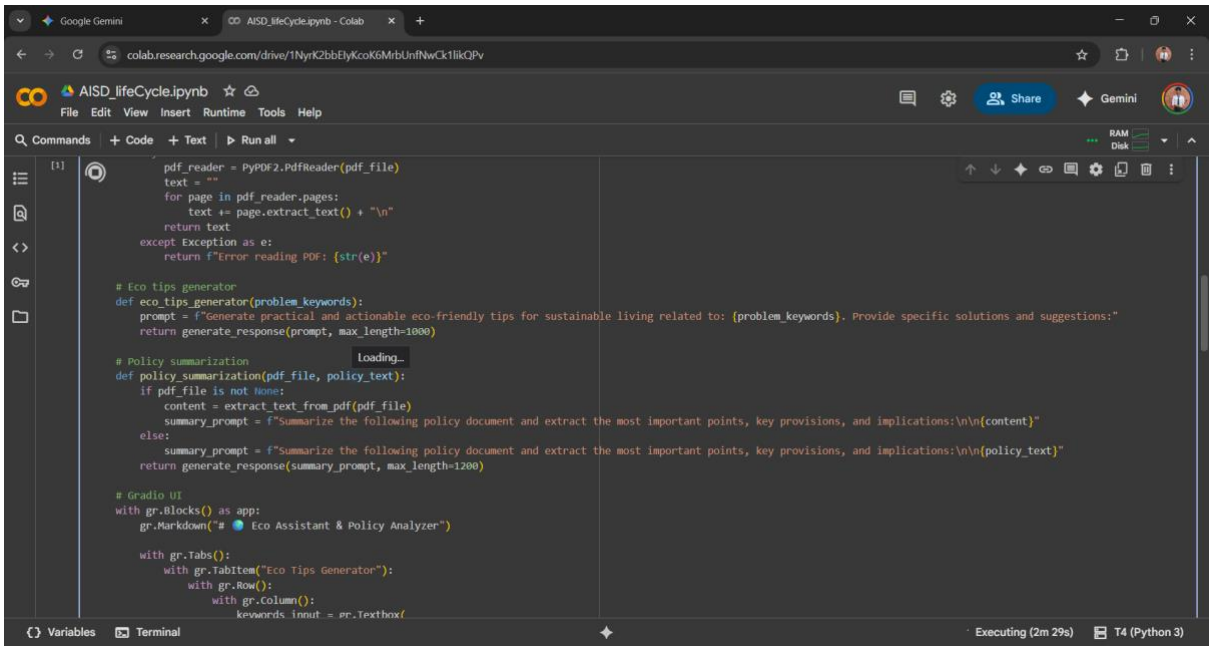
- The interface is minimalist and designed for accessibility for non-technical users.
- Key elements include:
  - A sidebar for navigation.
  - KPI visualizations with summary cards.
  - Tabbed layouts for chat, eco tips, and forecasting.
  - Real-time form handling.
  - PDF report download capability.

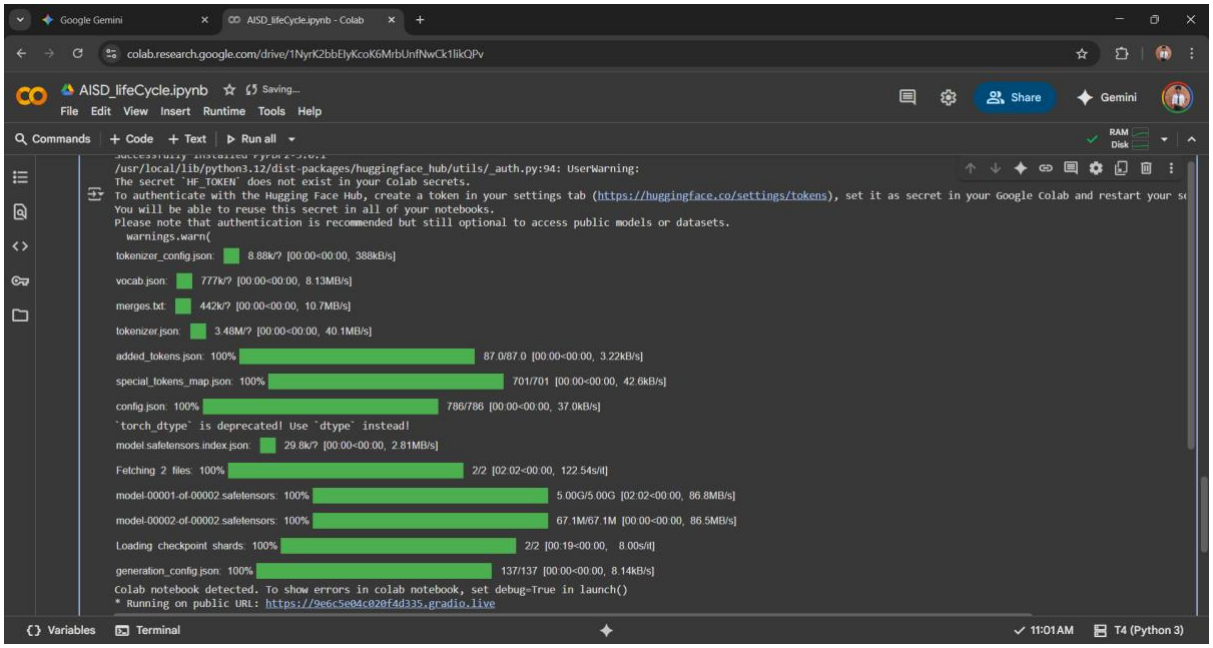
10. Testing:

- Testing was conducted in several phases:
  - **Unit Testing:** For prompt engineering functions and utility scripts.
  - **API Testing:** Done via Swagger UI, Postman, and test scripts.
  - **Manual Testing:** To validate file uploads, chat responses, and output consistency.
  - **Edge Case Handling:** To address malformed inputs, large files, and invalid API keys.
- Each function was validated to ensure reliability in both offline and API-connected modes.

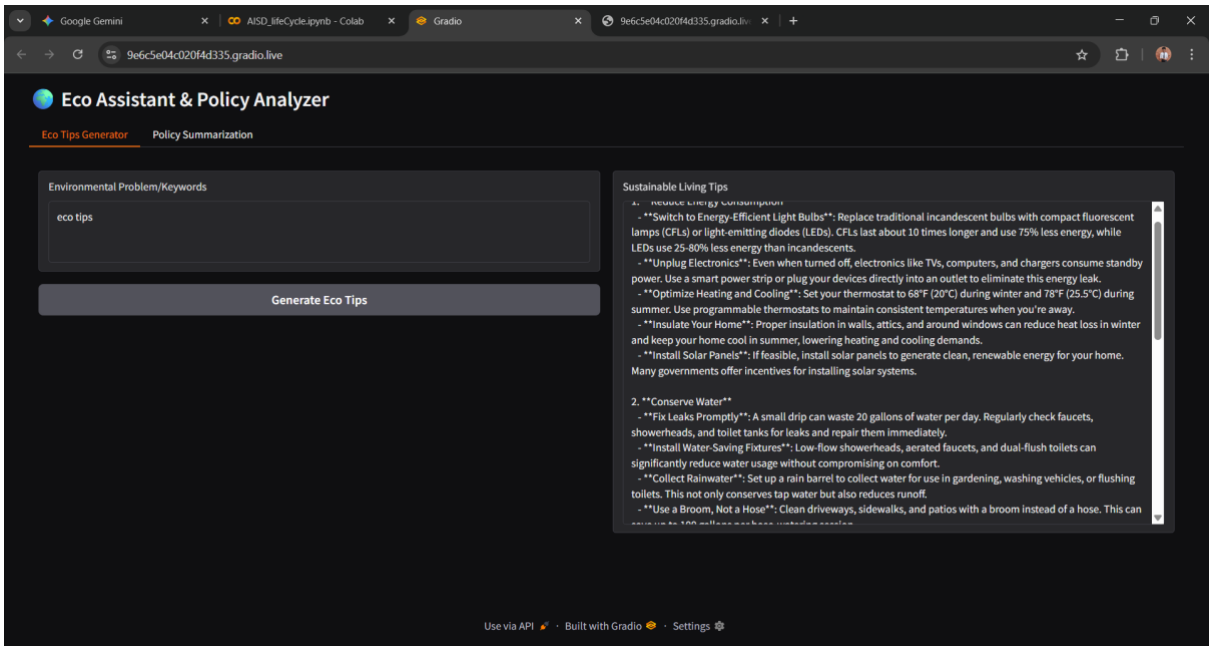
11.Source Code Screenshots:

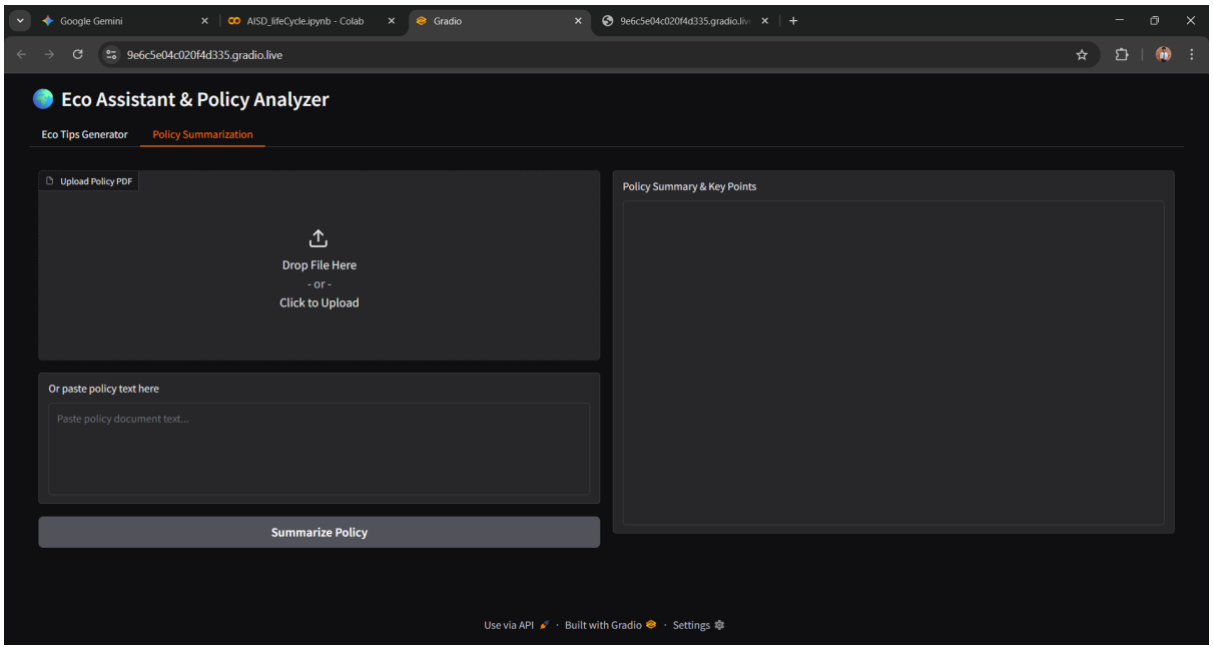

Google Gemini
AISD_lifeCycle.ipynb - Colab
colab.research.google.com/drive/1NyrK2bbElyKcoK6MrbUnfNwCk1ikQPv
AISD_lifeCycle.ipynb
Saving...
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
[1]
!pip install gradio transformers accelerate torch PyPDF2
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM
import PyPDF2
# Load model and tokenizer
model_name = "ibm-granite/granite-3.2-2b-instruct"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModelForCausalLM.from_pretrained(
model_name,
torch_dtype=torch.float16 if torch.cuda.is_available() else torch.float32,
device_map="auto" if torch.cuda.is_available() else None
)
# Ensure pad token exists
if tokenizer.pad_token is None:
tokenizer.pad_token = tokenizer.eos_token
# Response generator
def generate_response(prompt, max_length=1024):
inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)
if torch.cuda.is_available():
inputs = {k: v.to(model.device) for k, v in inputs.items()}
with torch.no_grad():
Variables
Terminal
Executing (1m 31s)
T4 (Python 3)


Google Gemini
AISD_lifeCycle.ipynb - Colab
colab.research.google.com/drive/1NyrK2bbElyKcoK6MrbUnfNwCk1ikQPv
AISD_lifeCycle.ipynb
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Run all
RAM
Disk
[1]
pdf_reader = PyPDF2.PdfReader(pdf_file)
text = ""
for page in pdf_reader.pages:
text += page.extract_text() + "\n"
return text
except Exception as e:
return f"Error reading PDF: {str(e)}"
# Eco tips generator
def eco_tips_generator(problem_keywords):
prompt = f"Generate practical and actionable eco-friendly tips for sustainable living related to: {problem_keywords}. Provide specific solutions and suggestions:"
return generate_response(prompt, max_length=1000)
Loading...
# Policy summarization
def policy_summarization(pdf_file, policy_text):
if pdf_file is not None:
content = extract_text_from_pdf(pdf_file)
summary_prompt = f"Summarize the following policy document and extract the most important points, key provisions, and implications:\n\n{content}"
else:
summary_prompt = f"Summarize the following policy document and extract the most important points, key provisions, and implications:\n\n{policy_text}"
return generate_response(summary_prompt, max_length=1200)
Gradio UI
with gr.Blocks() as app:
gr.Markdown("# Eco Assistant & Policy Analyzer")
with gr.Tabs():
with gr.TabItem("Eco Tips Generator"):
with gr.Row():
with gr.Column():
keywords_input = gr.Textbox(
Variables
Terminal
Executing (2m 29s)
T4 (Python 3)



12.Source Output:


Google Gemini
AISD_lifeCycle.ipynb - Colab
Gradio
9e6c5e04c020f4d335.gradio.live
9e6c5e04c020f4d335.gradio.live
Eco Assistant & Policy Analyzer
Eco Tips Generator
Policy Summarization
Environmental Problem/Keywords
eco tips
Generate Eco Tips
Sustainable Living Tips
- **Switch to Energy-Efficient Light Bulbs**: Replace traditional incandescent bulbs with compact fluorescent lamps (CFLs) or light-emitting diodes (LEDs). CFLs last about 10 times longer and use 75% less energy, while LEDs use 25-80% less energy than incandescents.
- **Unplug Electronics**: Even when turned off, electronics like TVs, computers, and chargers consume standby power. Use a smart power strip or plug your devices directly into an outlet to eliminate this energy leak.
- **Optimize Heating and Cooling**: Set your thermostat to 68°F (20°C) during winter and 78°F (25.5°C) during summer. Use programmable thermostats to maintain consistent temperatures when you're away.
- **Insulate Your Home**: Proper insulation in walls, attics, and around windows can reduce heat loss in winter and keep your home cool in summer, lowering heating and cooling demands.
- **Install Solar Panels**: If feasible, install solar panels to generate clean, renewable energy for your home. Many governments offer incentives for installing solar systems.
2. **Conserve Water**
- **Fix Leaks Promptly**: A small drip can waste 20 gallons of water per day. Regularly check faucets, showerheads, and toilet tanks for leaks and repair them immediately.
- **Install Water-Saving Fixtures**: Low-flow showerheads, aerated faucets, and dual-flush toilets can significantly reduce water usage without compromising on comfort.
- **Collect Rainwater**: Set up a rain barrel to collect water for use in gardening, washing vehicles, or flushing toilets. This not only conserves tap water but also reduces runoff.
- **Use a Broom, Not a Hose**: Clean driveways, sidewalks, and patios with a broom instead of a hose. This can
Use via API · Built with Gradio · Settings


Google Gemini
AISD_lifeCycle.ipynb - Colab
Gradio
9e6c5e04c020f4d335.gradio.live
9e6c5e04c020f4d335.gradio.live
Eco Assistant & Policy Analyzer
Eco Tips Generator
Policy Summarization
Upload Policy PDF
Drop File Here
- or -
Click to Upload
Or paste policy text here
Paste policy document text...
Summarize Policy
Policy Summary & Key Points
Use via API · Built with Gradio · Settings

13. Future Enhancements:

Planned future enhancements include:

- **User Sessions and History Tracking**: To provide a more personalized experience for users.
- **Enhanced Security**: Implementing token-based authentication, OAuth2, and role-based access for secure deployments