GreenSpark AI: AI-Enhanced Software Development Lifecycle

Team Leader: RAJA .R
Team Members: RAMJI.S
Team Members: RANJITH.S
Team Members: RAJKUMAR.R

1. Introduction

GreenSpark AI is an eco-focused AI-powered project that leverages IBM Granite models to enhance the Software Development Lifecycle (SDLC). It automates requirement gathering, code generation, testing, deployment, and documentation, while also providing sustainability-focused solutions such as carbon footprint estimation, policy summarization, eco-friendly lifestyle tips, and green technology ideas.

2. Project Overview

Purpose: The purpose of GreenSpark AI is to integrate AI-driven automation with eco-sustainability. It supports developers, researchers, and citizens in reducing environmental impact while demonstrating the power of SmartSDLC. Features: - EcoLife Guide: Generate eco-friendly lifestyle tips. - Policy Snapshot: Summarize environmental policy documents. - Carbon Meter: Estimate and suggest reduction for carbon footprint. - EcoTech Sparks: Recommend sustainable technology ideas.

3. Architecture

Frontend: Gradio-based interactive UI Backend: Python with Hugging Face Transformers and Torch Model: IBM Granite 3.3-2B Instruct (Hugging Face) Deployment: Google Colab with GPU (T4) Version Control: GitHub for hosting and collaboration

4. Setup Instructions

Prerequisites: - Python 3.9 or later - Install libraries: transformers, torch, gradio, pypdf - IBM Granite model access from Hugging Face - GPU-enabled runtime (Google Colab recommended) Steps: 1. Clone the repository 2. Install dependencies 3. Run the Python script 4. Launch the Gradio app 5. Interact with EcoLife, Policy Snapshot, Carbon Meter, and EcoTech modules

5. Folder Structure

app/ – Core functions and logic ui/ – Gradio UI definitions models/ – Model integration with Hugging Face docs/ – Documentation and reports main.py – Entry point to run the project

6. Running the Application

➢ Run the Python script ➢ Launch the Gradio interface in browser ➢ Navigate through tabs: EcoLife, Policy Snapshot, Carbon Meter, EcoTech ➢ Upload PDFs, enter activities, and generate responses

7. Module Documentation

- Eco Tips Generator: Provides actionable eco-friendly suggestions. - Policy Summarization: Extracts and summarizes environmental policies from PDFs or text. - Carbon Footprint Estimator: Estimates monthly CO■ emissions and reduction strategies. - Green Tech Ideas: Suggests innovative sustainable technologies.

8. Authentication

The current version does not include authentication. Future versions may implement role-based access and API key security for extended deployments.

9. User Interface

The UI is built using Gradio with multiple tabs: - EcoLife Guide: Textbox input and AI-generated tips - Policy Snapshot: PDF upload/text input for summaries - Carbon Meter: Lifestyle activity input with footprint estimates - EcoTech Sparks: Sector/industry input for innovation ideas Additional features: Copy-to-clipboard buttons for outputs

10. Testing

Testing includes: - Unit testing (functions and modules) - Manual testing of UI workflows - PDF text extraction validation - Edge case handling (invalid inputs, empty files)

11. Known Issues

- Requires internet access to fetch model - Limited offline support - Performance depends on GPU availability - UI basic styling (can be improved)

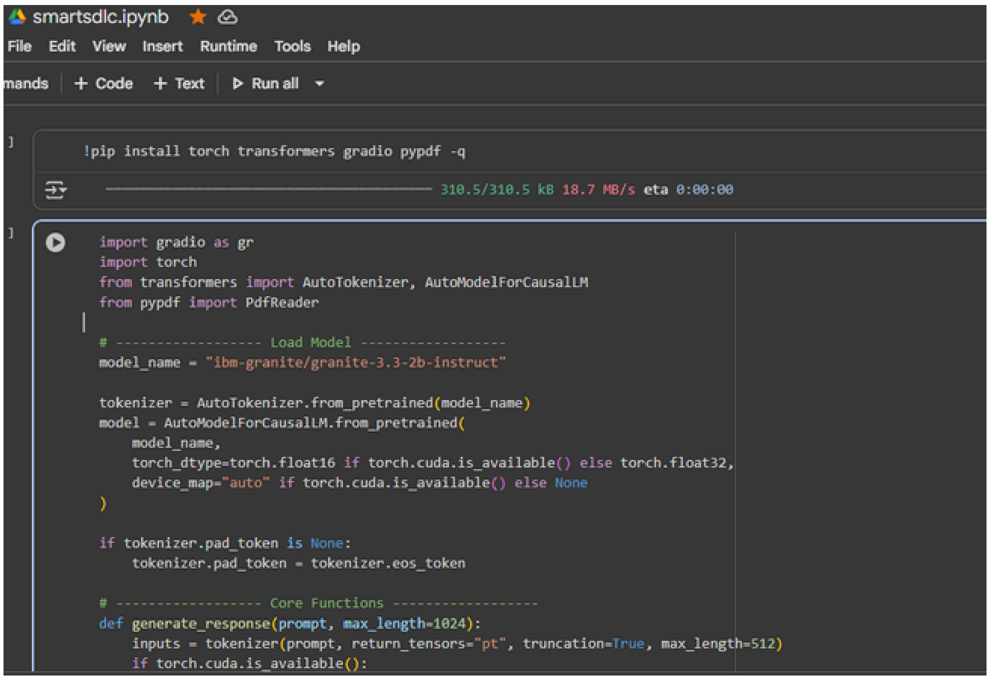
12. Future Enhancements

- Mobile app version - Multi-language support - Voice-based eco-tips - IoT integration for real-time monitoring - Enhanced dashboards for visualization

13. Screenshots & Outputs

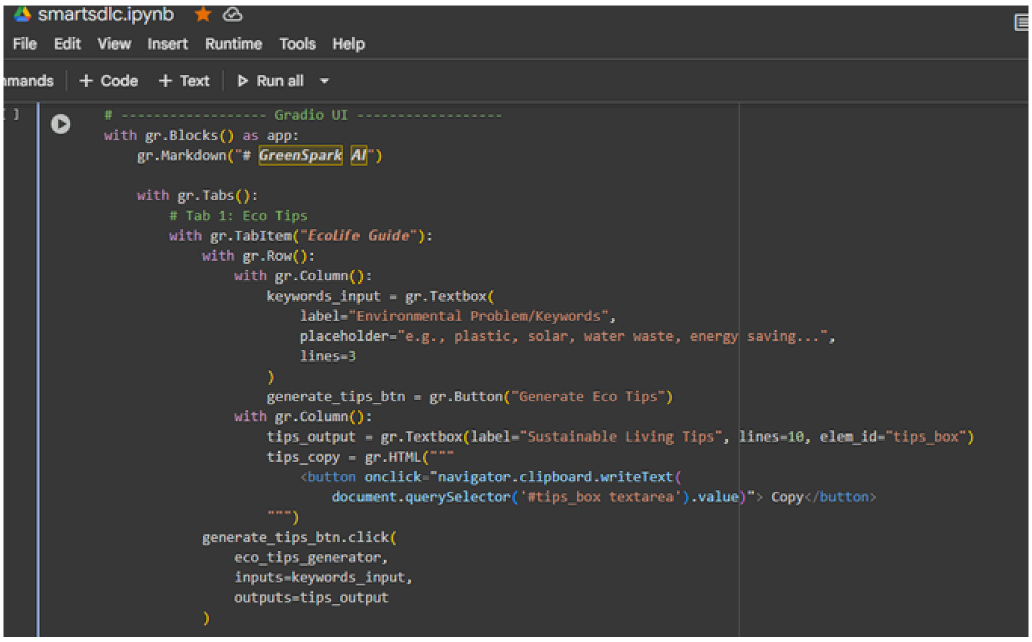
[Insert screenshots of Gradio interface, sample outputs, and code execution here]

Program


smartsdlc.ipynb
File Edit View Insert Runtime Tools Help
mands + Code + Text Run all

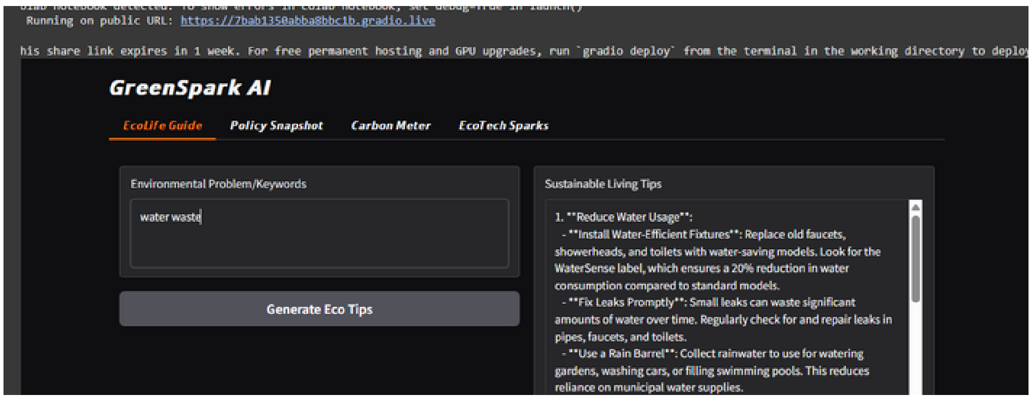
```
!pip install torch transformers gradio pypdf -q

   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 310.5/310.5 kB 18.7 MB/s eta 0:00:00
```

```python
import gradio as gr
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM
from pypdf import PdfReader

# ------------------ Load Model ------------------
model_name = "ibm-granite/granite-3.3-2b-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
)

if tokenizer.pad_token is None:
    tokenizer.pad_token = tokenizer.eos_token

# ------------------ Core Functions ------------------
def generate_response(prompt, max_length=1024):
    inputs = tokenizer(prompt, return_tensors="pt", truncation=True, max_length=512)
    if torch.cuda.is_available():
```


smartsdlc.ipynb
File Edit View Insert Runtime Tools Help
mands + Code + Text ▷ Run all

```python
# ----------------- Gradio UI -----------------
with gr.Blocks() as app:
    gr.Markdown("# GreenSpark AI")

    with gr.Tabs():
        # Tab 1: Eco Tips
        with gr.TabItem("EcoLife Guide"):
            with gr.Row():
                with gr.Column():
                    keywords_input = gr.Textbox(
                        label="Environmental Problem/Keywords",
                        placeholder="e.g., plastic, solar, water waste, energy saving...",
                        lines=3
                    )
                    generate_tips_btn = gr.Button("Generate Eco Tips")
                with gr.Column():
                    tips_output = gr.Textbox(label="Sustainable Living Tips", lines=10, elem_id="tips_box")
                    tips_copy = gr.HTML("""
                        <button onclick="navigator.clipboard.writeText(
                            document.querySelector('#tips_box textarea').value)"> Copy</button>
                    """)
            generate_tips_btn.click(
                eco_tips_generator,
                inputs=keywords_input,
                outputs=tips_output
            )
```

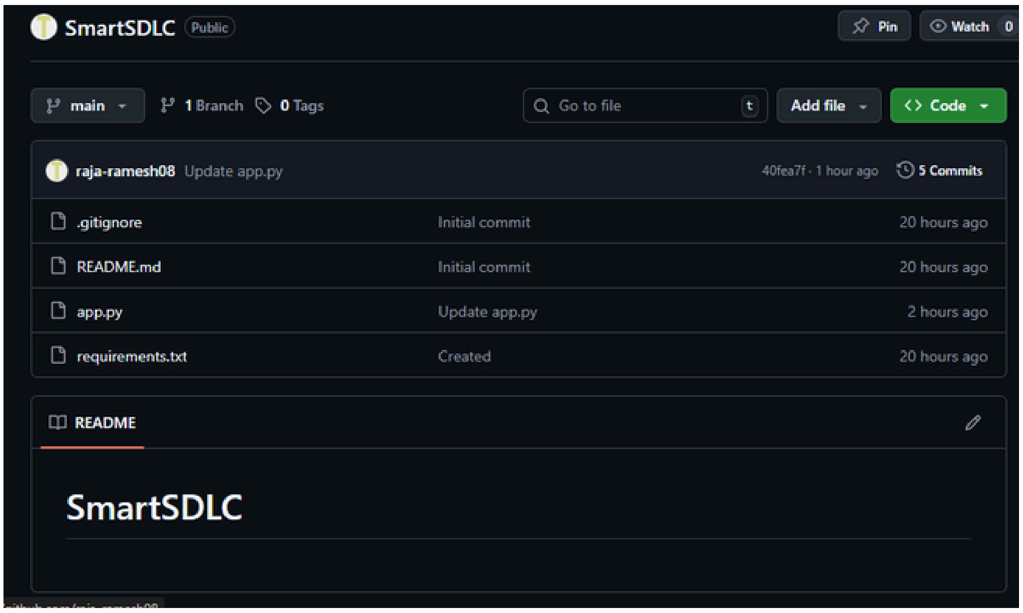
Output



```
Running on public URL: https://7bab1350abba8bbc1b.gradio.live

his share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy
```

```
1. **Reduce Water Usage**:
  - **Install Water-Efficient Fixtures**: Replace old faucets,
showerheads, and toilets with water-saving models. Look for the
WaterSense label, which ensures a 20% reduction in water
consumption compared to standard models.
  - **Fix Leaks Promptly**: Small leaks can waste significant
amounts of water over time. Regularly check for and repair leaks in
pipes, faucets, and toilets.
  - **Use a Rain Barrel**: Collect rainwater to use for watering
gardens, washing cars, or filling swimming pools. This reduces
reliance on municipal water supplies.
```

GitHub



raja-rameshO8 Update app.py 40fea7f · 1 hour ago 5 Commits

.gitignore	Initial commit	20 hours ago
README.md	Initial commit	20 hours ago
app.py	Update app.py	2 hours ago
requirements.txt	Created	20 hours ago

README

SmartSDLC