# PROJECT REPORT

# Sustainable Smart City Assistant
## Using IBM Granite LLM

**YEAR** **: 2025 – 2026**

COLLEGE NAME : **K.C.S KASI NADAR COLLEGE OF ARTS & SCIENCE**

CODE :

DEPARTMENT : **COMPUTER SCIENCE**

PROGRAM : **B.SC**

SEMESTER :**VI**

PROJECT SUBMITTED TO: UNIVERSITY OF MADRAS / NAAN MUDALVAN

COURSE NAME :


**TEAM LEADER: LAKSHMAN.K**

**MEMBERS:**

1. SACHIN.L
2. RITHICK KUMAR.D
3. LOKESH.B

**GUIDED BY: MRS.R.PADMADEVI**

**SPOC NAME: DR.K. LATHAKAMESWARI**

# Introduction

This program is a Gradio-based AI assistant designed to support users in two meaningful ways: promoting sustainable living and simplifying the understanding of complex policy documents. It combines natural language processing (NLP) with an interactive web interface, making advanced AI tools accessible to everyday users without requiring programming knowledge.

The first functionality, the Eco Tips Generator, focuses on addressing environmental challenges by generating practical, eco-friendly lifestyle tips. Users can input specific keywords related to environmental problems, such as “plastic waste,” “solar energy,” or “water conservation.” The system then leverages a pre-trained language model to produce actionable suggestions tailored to the given keywords. For example, if a user inputs “plastic,” the tool might recommend reducing singleuse plastics, switching to reusable alternatives, or promoting recycling habits. This feature encourages sustainable practices at an individual level while raising awareness about global environmental concerns.

The second functionality, the Policy Summarization Tool, is aimed at simplifying policy documents, which are often lengthy, technical, and difficult for non-experts to interpret. Users can either upload a policy document in PDF format or directly paste policy text into the interface. The system processes the document and generates a concise summary highlighting the most important points, provisions, and implications. This is particularly valuable for students, researchers, and professionals who need to quickly grasp the essence of lengthy policies without reading through the entire text. By breaking down complex material into digestible insights, the tool helps users save time and enhances their understanding of critical issues.

The application is built on a technology stack that includes Hugging Face Transformers for the underlying AI model, PyTorch for efficient model execution with GPU support, PyPDF2 for extracting text from PDF files, and Gradio for creating an intuitive, browser-based interface. The chosen model, IBM Granite 3.2-2B Instruct, is a causal language model fine-tuned for text generation tasks, enabling it to generate contextually relevant tips and summaries.

Overall, this program demonstrates how AI can be applied to solve real-world problems by combining environmental sustainability and policy analysis in a single tool. Its simple design, flexible inputs, and practical outputs make it useful for both individuals seeking eco-friendly lifestyle guidance and professionals who need efficient ways to analyze policy documents.

# Features of Sustainable Smart City

The development of this program relies on a combination of powerful Python libraries and a pretrained language model, each serving a specific purpose in ensuring efficiency, accuracy, and userfriendliness. By integrating these technologies, the application is able to perform natural language generation, process PDF documents, and provide an intuitive web interface for end users. .

# Python Libraries

# 1.Transformers

The transformers library, developed by Hugging Face, is the core component of this program. It provides access to state-of-the-art pre-trained language models, including the IBM Granite model used here. With the help of this library, the program can tokenize input text, generate meaningful responses, and fine-tune model behavior for specific tasks such as eco tips generation and policy summarization**.**

# 2. Torch (PyTorch)

PyTorch is a widely used deep learning framework that enables efficient model execution. It supports both CPU and GPU computation, ensuring that the program can run on different hardware setups. When a GPU is available, PyTorch allows for faster processing by utilizing float16 precision, making response generation quicker and more efficient.

# 3. PyPDF2

The PyPDF2 library is employed for reading and extracting text from PDF documents. Since policy documents are often shared in PDF format, this library ensures that users can upload files and have the text content automatically extracted. This step is crucial for enabling the policy summarization functionality.

# 4. Gradio

Gradio is used to create the interactive, browser-based user interface. It simplifies deployment by allowing users to interact with the application through textboxes, file uploads, and buttons, without needing to install additional software or write code. Gradio also supports sharing the app via a public URL, making it easily accessible.

# 5. io

The io library is included to handle file-like objects where necessary. While not central to the core functionality, it provides support for smooth file handling and data processing. intervention.

# Model

The application uses ibm-granite/granite-3.2-2b-instruct, a causal language model designed for text generation. This model has been fine-tuned to follow instructions, making it suitable for generating actionable eco tips and producing structured summaries of complex documents. Its adaptability allows the program to handle different types of inputs, from short keywords to lengthy policy texts, while maintaining relevance and clarity in the output.

# Functional Components

The program is structured into several functional components, each handling a specific task that contributes to the overall workflow. These components work together to load the model, process input, extract content, and generate meaningful outputs for the user.

# 1 Model Loading

The first step is loading the IBM Granite model and its corresponding tokenizer. This ensures that the program can interpret user input and generate natural language responses. The implementation is hardware-adaptive: if CUDA (GPU support) is available, the model runs using FP16 precision, which improves speed and reduces memory usage. If no GPU is available, it falls back to CPU execution with standard precision. Additionally, the program checks whether a padding token exists in the tokenizer. If not, it assigns the end-of-sequence token (eos_token) as the padding token, ensuring smooth text processing without errors.

# 2 Text Generation Function (generate_response)

This function is responsible for generating AI-driven text outputs. It accepts a user prompt and tokenizes the input with a maximum length of 512 tokens. The model then generates text with the following configurations:

max_length=1024 to allow for detailed responses,

temperature=0.7 to balance between creativity and coherence,

and do_sample=True to introduce variation in responses.

Finally, the generated output is decoded, and the original prompt is removed to return only the meaningful response.

# 3 PDF Text Extraction (extract_text_from_pdf)

Since many policy documents come in PDF format, this component uses PyPDF2.PdfReader to extract text from uploaded files. It iterates through each page, gathers the text, and combines it into a single string. If the PDF cannot be read, an appropriate error message is returned. This ensures that even lengthy documents can be processed efficiently.

# 4 Eco Tips Generator (eco_tips_generator)

This component is designed to promote sustainable living. It accepts user-provided environmental problem keywords and constructs a prompt that asks the model to generate actionable eco-friendly suggestions. The output is a list of practical lifestyle changes and sustainability tips tailored to the keywords, such as plastic reduction, energy conservation, or water saving.

# 5 Policy Summarization (policy_summarization)

This function enables the program to simplify complex documents. Users can either upload a PDF or paste policy text directly. If a PDF is uploaded, the text is extracted using the earlier component. The program then creates a summarization prompt asking the model to highlight important points, key provisions, and implications. The response is a concise, structured summary that allows users to understand lengthy documents quickly.

# User Interface (Gradio)

The program employs Gradio Blocks to create an interactive and user-friendly interface. Gradio allows machine learning applications to be deployed as simple web apps that can be accessed directly through a browser, removing the need for complex installations or technical expertise. In this program, the interface is organized into two main tabs, each corresponding to one of the core functionalities: the Eco Tips Generator and the Policy Summarization Tool.

# 1. Eco Tips Generator Tab

This tab is designed to encourage sustainable living by generating practical environmental solutions. It contains:

**Input:** A textbox where users can type keywords related to environmental challenges. For instance, terms like “plastic pollution”, “renewable energy”, or “water conservation” can be entered. The input is flexible, allowing users to focus on broad topics or specific concerns.

**Button:** A clickable button labeled “Generate Eco Tips”. Once pressed, it triggers the backend function that constructs a prompt and sends it to the AI model for response generation.

**Output:** A multi-line textbox displays the results. The generated content consists of actionable and eco-friendly lifestyle suggestions tailored to the keywords entered. This ensures that users receive relevant, specific, and practical tips rather than generic advice.

# 2. Policy Summarization Tab

The second tab is dedicated to simplifying complex policy documents. It provides two different input methods for flexibility:

**Input Options**:

**File Upload:** Users can upload a policy document in PDF format. The program extracts text from the file automatically using the PDF extraction function.

**Textbox:** Alternatively, users can paste policy text directly into a textbox. This is helpful for shorter documents or excerpts.

**Button:** A “Summarize Policy” button triggers the summarization process.

**Output:** A multi-line textbox displays the results, which include the most important points, key provisions, and implications of the policy. The summary is concise yet comprehensive, allowing users to quickly grasp essential information without reading the full document.

# Overall Interface Design

The interface is minimalistic and intuitive, with clear labels for each input, button, and output area. By separating functionalities into two tabs, the design avoids clutter and makes navigation straightforward. Users can easily switch between eco tips generation and policy summarization without confusion.

# Application Workflow

The workflow of the application is designed to be straightforward, ensuring that users can easily interact with the system without requiring prior technical knowledge. The process involves launching the app, selecting the desired functionality, generating outputs through the AI model, and displaying results in a clean and user-friendly format.

# 1. Launching the Application

The application begins with the command app.launch(share=True), which initializes the Gradio interface. This creates a public URL that users can access directly through their browser. By enabling sharing, the program allows users to test and interact with the tool without needing to set up a local environment. This makes the application accessible across different devices and platforms.

# 2. Selecting a Tab

Once launched, the interface presents two clearly separated tabs for the program’s functionalities:

## Eco Tips Generator:

Users type in environmental problem keywords, such as “plastic waste,” “deforestation,” or “solar energy.” After clicking the “Generate Eco Tips” button, the input is processed and converted into a prompt for the model.

# 3. Real-Time Model Processing

After receiving input, the program passes it to the IBM Granite model via the text generation function. The model processes the prompt in real-time and generates outputs based on its training and instruction-following capability. For eco tips, it produces actionable sustainability suggestions, while for policy summarization, it extracts the most important points, key provisions, and potential implications.

# 4. Displaying the Results

The generated results are then displayed in multi-line textboxes within the same tab. This design ensures that users immediately see the output without navigating away from the input section. The results are clear, structured, and ready for interpretation or further use.

# Strengths of the Program

One of the primary strengths of this program lies in its ability to combine AI-driven text generation with document summarization, offering users two powerful features within a single application. The Eco Tips Generator empowers individuals to adopt sustainable practices by providing actionable lifestyle suggestions based on simple keyword prompts, while the Policy Summarization Tool simplifies lengthy or complex documents into concise, understandable summaries. Together, these components showcase the practical utility of artificial intelligence in addressing two socially relevant domains: sustainability and policy analysis.

Another notable strength is the user-friendly interface built with Gradio. Instead of requiring technical expertise, users can interact with the system through an intuitive web interface featuring tabs, textboxes, and upload buttons. This accessibility ensures that the tool can be adopted by a wide audience, including students, educators, researchers, and environmentally conscious citizens. By lowering technical barriers, the program demonstrates inclusivity and encourages broader usage.

The program also offers flexibility in input handling. For policy summarization, users are not restricted to a single mode of input; they can either upload PDF documents or paste text directly into the interface. This dual approach increases convenience, ensuring that users can work with whatever resources they have available. In practice, this makes the system adaptable to a variety of real-world scenarios, such as analyzing government policy briefs, NGO reports, or academic papers.

Additionally, the program is designed with hardware adaptability in mind. By detecting whether a GPU is available and switching to FP16 precision for efficient execution, it can significantly improve performance on high-end machines. At the same time, it provides fallback support for CPU-only environments, ensuring that the application remains usable even on devices without specialized hardware. This adaptability broadens the reach of the program across different platforms and user needs.

Finally, the program demonstrates strong real-world relevance. The Eco Tips Generator directly supports sustainability initiatives, encouraging small but impactful changes in daily life. Meanwhile, the Policy Summarization Tool helps individuals, researchers, and organizations save time and

# Limitations

While the program provides useful functionalities for generating eco-friendly lifestyle tips and summarizing policy documents, it also has certain limitations that may affect performance and output quality. These constraints are largely due to the nature of the model, the libraries used, and the hardware available for execution.

# Model Size

The program uses the IBM Granite 2B model, which, while efficient, is relatively small compared to larger language models with tens or hundreds of billions of parameters. As a result, it may struggle when processing very large documents or highly technical policies that require deep contextual understanding. For example, lengthy government policies with specialized legal or economic terminology may not be summarized as accurately or comprehensively as desired. The model's limited size restricts its ability to capture and represent very complex information.

# PDF Parsing

The program relies on PyPDF2 for extracting text from PDF files. While this works well for textbased PDFs, it is not effective for scanned documents or PDFs that primarily contain images. In such cases, the extracted content may be incomplete, garbled, or entirely missing. This limitation means that some policy documents, especially those distributed as scanned copies, cannot be processed without additional tools such as Optical Character Recognition (OCR).

# Performance

Another challenge is performance variability depending on hardware. On systems with a GPU, the program runs efficiently, using float16 precision to speed up text generation. However, on CPU-only systems, response generation can be noticeably slower, especially when processing large inputs or generating lengthy outputs. This delay may reduce usability for users who do not have access to high-performance hardware.

# Accuracy of Outputs

The program's results depend heavily on the AI model's training data and generation capabilities. While it generally produces relevant eco tips and policy summaries, the outputs may sometimes be vague, repetitive, or overly generic. For example, eco-friendly tips might repeat common advice such as reducing plastic use or conserving energy without offering new insights. Similarly, policy summaries may overlook subtle but important details. Users should therefore treat the outputs as supportive suggestions rather than authoritative answers.

# Overall Impact

These limitations highlight areas where the program could be improved, such as integrating more powerful models, adding OCR for scanned documents, and refining summarization techniques. Despite these challenges, the application remains a valuable tool, but users should be aware of its constraints when interpreting the results.

# Possible Improvements

Although the current version of the program provides valuable features for generating eco-friendly tips and summarizing policies, there are several ways it can be enhanced to improve accuracy, accessibility, and overall usability. These improvements would make the application more versatile and suitable for a wider audience.

# Adding OCR for Scanned PDFs

At present, the program relies on PyPDF2, which only works with text-based PDFs. Many official documents, however, are distributed as scanned copies containing images rather than selectable text. By integrating Optical Character Recognition (OCR) technologies such as Tesseract or Google's OCR APIs, the program could extract text from image-based PDFs. This would significantly expand the tool's usefulness for researchers, students, and professionals who often work with scanned policy documents.

# Using Long-Document Summarization Models

The IBM Granite 2B model has limitations when dealing with very large or highly technical documents. To address this, the program could incorporate long-document summarization models like LongT5, BART-large, or LLaMA-based summarizers that are optimized for handling extended contexts. This would allow the tool to process government policies, research reports, or legal documents more effectively, providing users with summaries that capture both detail and nuance.

# Providing Structured Output

Currently, outputs are returned as plain text. While readable, this format may not always be the most efficient for users seeking quick insights. Enhancing the system to generate structured outputs—such as bullet points, headings, and categorized sections—would make results easier to scan and interpret. For example, a policy summary could be broken down into "Key Provisions," "Implications," and "Stakeholder Impact," providing clearer value.

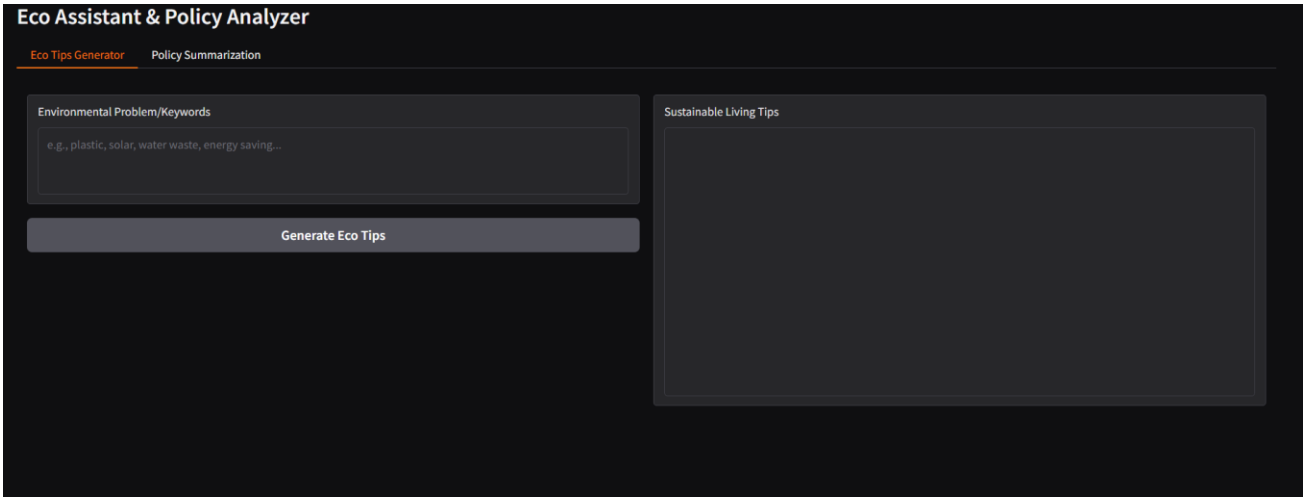
# Downloadable Results

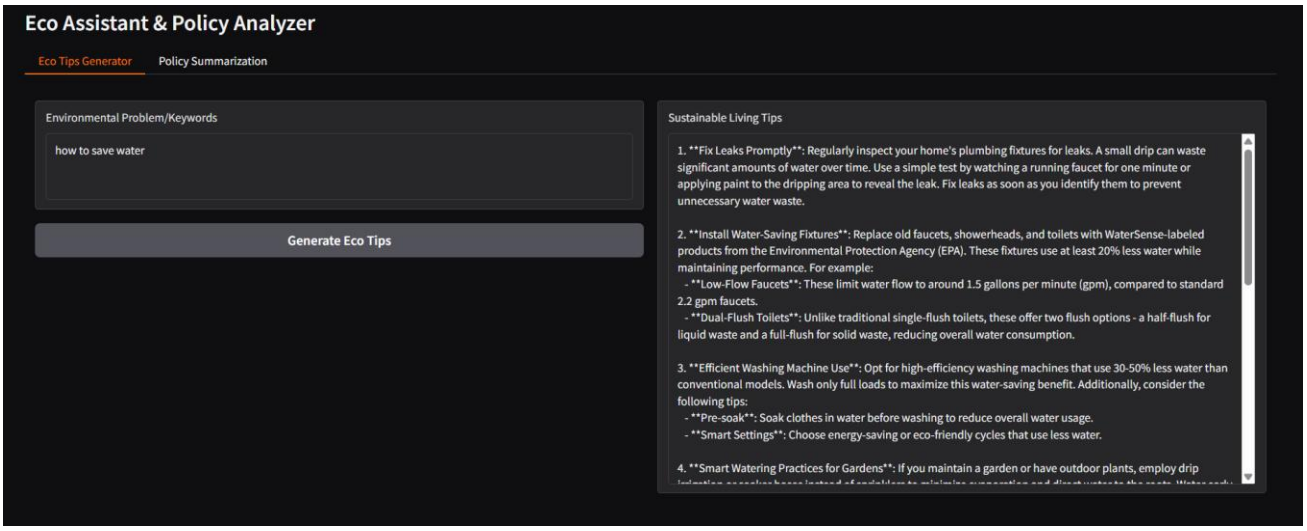
Another improvement would be to allow users to download results in various formats such as text, PDF, or Word documents. This feature would be particularly helpful for students writing reports, researchers compiling references, or professionals preparing presentations. It would also make it easier to store and share results.

# Multilingual Support

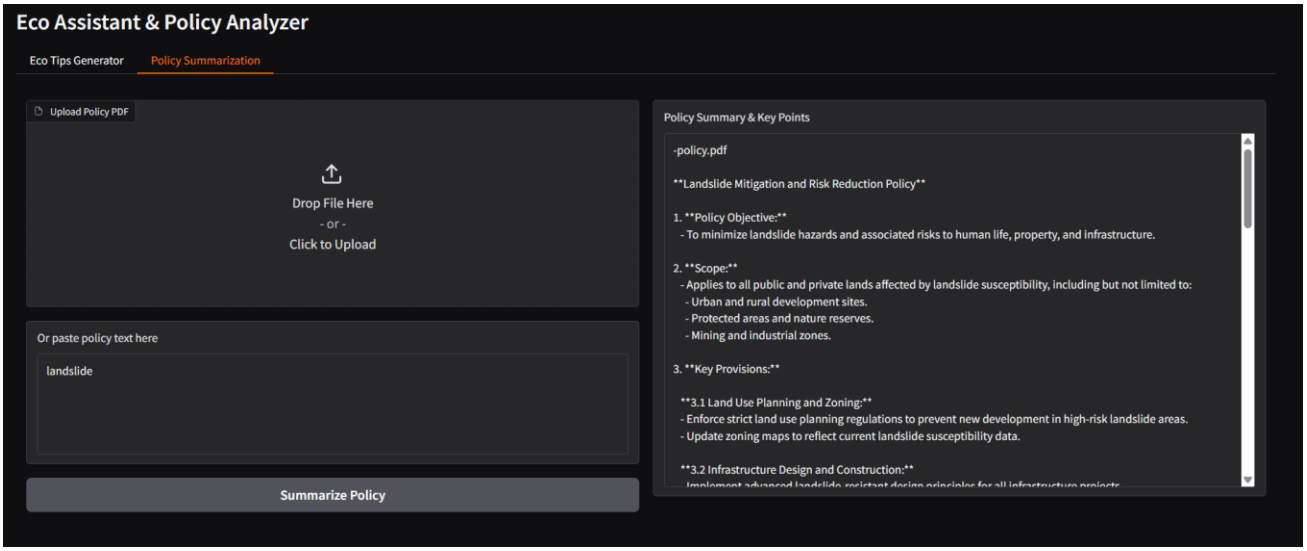
Currently, the system primarily functions in English. Adding multilingual support would greatly enhance its accessibility, enabling users from different regions to generate eco tips or analyze policies in their native languages. Libraries like Hugging Face's multilingual models could be integrated to achieve this.

# Eco Tips Generator




# Policy Summarization



# Conclusion

The program demonstrates the potential of combining artificial intelligence with user-friendly interfaces to solve real-world challenges in both sustainability and policy analysis. As a prototype, it effectively showcases how natural language processing, document handling, and interactive design can work together to create a practical and accessible application. By offering two core features—the Eco Tips Generator and the Policy Summarization Tool—the system addresses important needs: promoting sustainable lifestyle choices and simplifying complex policy documents.

The integration of technologies such as Hugging Face Transformers, PyTorch, PyPDF2, and Gradio provides a strong foundation for the application. Each technology plays a vital role, from enabling advanced text generation to managing PDF extraction and delivering a seamless web-based user experience. The use of the IBM Granite 2B Instruct model ensures that the responses are contextaware, while the Gradio interface makes the tool accessible to a wide range of users, including students, educators, researchers, and environmentally conscious individuals.

However, the current version is not without its limitations. The model's size restricts its ability to process very large or highly technical documents with complete accuracy. PDF parsing issues arise when dealing with scanned or image-based files, and performance may vary depending on hardware availability, with slower response times on CPU-only systems. Additionally, the outputs, while helpful, may sometimes lack depth, structure, or precision.

Despite these challenges, the program's potential for growth is significant. By incorporating features such as OCR for scanned PDFs, long-document summarization models, structured output formatting, downloadable results, and multilingual support, the system can be transformed into a more robust and comprehensive solution. These improvements would not only enhance usability but also broaden the scope of the application, making it valuable across different domains and regions.

In conclusion, this program stands as an effective prototype of an AI-powered Eco Assistant and Policy Analyzer. It demonstrates how artificial intelligence can be leveraged to encourage sustainable practices and make complex information more digestible. With targeted enhancements and scalability improvements, the application has the potential to become a powerful tool for sustainability advocates, policy researchers, educators, and decision-makers worldwide.

The program also offers flexibility in input handling. For policy summarization, users are not restricted to a single mode of input; they can either upload PDF documents or paste text directly into the interface. This dual approach increases convenience, ensuring that users can work with whatever resources they have available. In practice, this makes the system adaptable to a variety of real-world scenarios, such as analyzing government policy briefs, NGO reports, or academic papers.

Additionally, the program is designed with hardware adaptability in mind. By detecting whether a GPU is available and switching to FP16 precision for efficient execution, it can significantly improve performance on high-end machines. At the same time, it provides fallback support for CPU-only environments, ensuring that the application remains usable even on devices without specialized hardware. This adaptability broadens the reach of the program across different platforms and user needs.