**A Minor Project Report on**

AI-Powered-Emission-Intelligence-System

**in partial fulfillment for the award of the degree**

***of***

**BACHELOR OF TECHNOLOGY**

**In**

**COMPUTER SCIENCE & ENGINEERING**

**Session ( 2022 – 2026)**

****

**Submitted to:- Submitted by:-**

**Prof.Sapan Kumar Jain**

**Shrikant kushwaha Shivam kushwaha Siddharth Jain**

**0206CS221196**

**0206CS221182**

**0206CS221201**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GYAN GANGA INSTITUTE OF TECHNOLOGY & SCIENCES**

**JABALPUR (M.P.) (2025)**

**RAJIV GANDHI PRODYOGIKI VISHWAVIDYALAYA,**

**BHOPAL (M.P.)**

Table of Contents

1. [Introduction 3](#_bookmark0)
   1. [Purpose of the Project 3](#_bookmark1)
   2. [Project and Product Overview 3](#_bookmark2)
   3. [Intended Audience 3](#_bookmark3)
   4. [Team Architecture 4](#_bookmark4)
   5. [Overall Description 4](#_bookmark5)
   6. [Product Perspective 4](#_bookmark6)
2. [Problem Statement 5](#_bookmark7)
   1. [Business Requirements 5](#_bookmark8)
      1. [Entry Points 5](#_bookmark9)
      2. [Processing Modes 5](#_bookmark10)
      3. [Reports Generated 5](#_bookmark11)
   2. [System Requirements 5](#_bookmark12)
3. [PROJECT UNDERSTANDING DOCUMENT 7](#_bookmark13)
   1. [Purpose of Project 7](#_bookmark14)
   2. [Objective 7](#_bookmark15)
   3. [MIS Reports 7](#_bookmark16)
4. [TIMELINE 8](#_bookmark17)
5. [REQUIREMENTS 9](#_bookmark18)
   1. [SPECIFIC REQUIREMENTS 9](#_bookmark19)
      1. [Hardware Interface 10](#_bookmark20)
      2. [Software Interface 10](#_bookmark21)
   2. [NON-FUNCTIONAL REQUIREMENTS 10](#_bookmark22)
   3. [SOFTWARE SYSTEM ATTRIBUTES 10](#_bookmark23)
6. [DESIGN TECHNIQUES 11](#_bookmark24)
   1. [Frontend 11](#_bookmark25)
   2. [Backend 11](#_bookmark26)
   3. [AI & Language Processing 11](#_bookmark27)
   4. [Structured Data Output 11](#_bookmark28)
   5. [Geospatial Intelligence Engine 11](#_bookmark29)
   6. [Database & Storage 11](#_bookmark30)
7. [TIER ARCHITECTURE 12](#_bookmark31)
   1. [Data Layer 12](#_bookmark32)
   2. [Processing Layer 12](#_bookmark33)
   3. [Application Layer 12](#_bookmark34)
8. [SOFTWARE PROCESS MODEL 14](#_bookmark35)
   1. [Why Not Evolutionary Models? 14](#_bookmark36)
   2. [Why Not the Waterfall Model? 14](#_bookmark37)
   3. [Why the Incremental RAD Model? 14](#_bookmark38)
   4. [Observation 14](#_bookmark39)
   5. [Determining Project Feasibility 14](#_bookmark40)
9. [DESIGN 16](#_bookmark41)
   * 1. [USE CASE DIAGRAM 16](#_bookmark42)
     2. [Class diagram 17](#_bookmark43)
     3. [Chabot Class diagram 17](#_bookmark44)
   1. [Object diagram 18](#_bookmark45)
   2. [Sequence diagram 19](#_bookmark46)
10. [expected outcome 21](#_bookmark47)
    1. [Home Page 21](#_bookmark48)
    2. [Data Upload Section 21](#_bookmark49)
    3. [Emission Data Plotted in Map 21](#_bookmark50)
    4. [EcoVision Ai Chatbot 22](#_bookmark51)
11. [Conclusion 23](#_bookmark52)

## Introduction

Environmental monitoring is essential for understanding the ecological impact of industrialization. However, emissions data is often fragmented, buried in unstructured formats like PDFs, and difficult to process at scale.

**Generative AI-Powered Emission Intelligence** presents a novel multi-modal solution that combines generative AI, geospatial analysis, and advanced natural language processing to automate the extraction, structuring, and visualization of emissions reports from multiple Indian cities.

The system integrates Google's **Gemini API**, **LangChain agents**, **semantic embeddings**, and **CLIP- based image understanding** to build an end-to-end pipeline—from PDF parsing to interactive, chat- based environmental insights. By transforming semi-structured reports into interactive emissions maps, the solution enables real-time exploration and analysis in an accessible, user-friendly format.

##### Purpose of the Project

The primary goals of this project are to:

* Automatically extract structured emissions data from complex PDF documents using generative AI.
* Visualize city-level emissions across India using geospatial tools and vector search.
* Enable semantic querying of city-specific data via LangChain agents, powered by Gemini.
* Interpret emissions maps through multi-modal reasoning with OpenCLIP.
* Provide a natural-language chatbot interface (Gradio) for non-technical stakeholders to interact with and understand the data.

This project demonstrates how advanced LLM-based systems can revolutionize environmental intelligence by making it real-time, visual, and intuitive.

### Project and Product Overview

This capstone project addresses the challenge of fragmented environmental data by creating a robust AI-driven pipeline capable of:

* **Parsing PDFs** using LangChain’s PyPDFLoader.
* **Extracting structured emissions data** using Google Gemini 2.0 Flash and converting it into JSON.
* **Geospatial resolution** by mapping cities to coordinates through semantic search (FAISS).
* **Interactive visualization** via Folium-based emission maps.
* **Map interpretation** using OpenCLIP for visual reasoning.
* **Conversational exploration** using a LangChain + Gradio-powered chatbot.

The result is an AI-powered assistant that can answer environment-related queries, interpret emissions maps.

##### Intended Audience

This solution is designed to benefit a wide range of users:

* Environmental analysts and researchers seeking scalable tools for emissions data analysis.
* Government bodies and regulatory agencies monitoring city-level pollution and trends.
* Data scientists and AI students exploring multi-modal agent workflows.
* Capstone reviewers and academic mentors assessing real-world LLM applications.
* Developers and engineers building intelligent systems involving documents, maps, and conversation interfaces.

##### Team Architecture

The team comprises individuals with expertise in AI, data science, and full-stack development. Collaborative development was conducted using Gemini APIs, LangChain, Kaggle notebooks, and various open-source tools.

|  |  |
| --- | --- |
| **Name** | **Role** |
| **Prof. Sapan Kumar Jain** | Project Mentor |
| **Shivam Kushwaha (0206CS221182)** | LangChain Integrations, Gemini API |
| **Siddharth Jain(0206CS221201)** | Function Calling, Gradio UI Development |
| **Shrikant Kushwaha(0206CS221196)** | Data Extraction, Map Agent |

Responsibilities were distributed across agent design, vector embedding, visualization, prompt engineering, and interface development.

##### Overall Description

The project follows a comprehensive end-to-end AI development pipeline:

* **Problem Framing**: Emissions data is trapped in unstructured PDF formats.
* **Data Acquisition**: Environmental reports and city coordinates sourced from Kaggle and GeoNames.
* **PDF Parsing**: Performed using LangChain’s PyPDFLoader.
* **Data Structuring**: Gemini generates structured JSON via deterministic, low-temperature prompts.
* **Geo-Matching**: Sentence embeddings and FAISS are used to associate cities with geographic coordinates.
* **Visualization**: Folium and MarkerCluster are used to render interactive emissions maps.
* **Map Interpretation**: CLIP models provide multi-modal reasoning on map content.
* **Conversational Interaction**: LangChain and Gemini enable retrieval-augmented Q&A
* **Frontend**: A Gradio-based chatbot interface allows users to explore data conversationally.

### Product Perspective

The final product is a research prototype that functions as an intelligent, interactive assistant for environmental emissions analysis. Key components include:

* **Generative Language Understanding**: Google Gemini for structured data extraction and Q&A.
* **Semantic Visual Reasoning**: OpenCLIP for interpreting map-based insight
* **Structured Output**: Data available in JSON and CSV formats
* **Dynamic Tool Usage**: LangChain agents and tools for information retrieval.
* **Geospatial Intelligence**: Folium for emissions mapping and FAISS for semantic matching.
* **Accessible Interface**: Gradio chatbot for non-technical users.

## Problem Statement

With the growing impact of climate change, there is an urgent need to make environmental emissions data more accessible, understandable, and usable. Currently, this data is scattered across various sources, often buried in unstructured formats like PDF documents. This lack of standardization makes it difficult for researchers, policymakers, and the general public to derive meaningful insights and take timely action for environmental protection.

The proposed system addresses this challenge by:

* Extracting textual data from emissions-related PDF reports.
* Converting unstructured data into a structured, machine-readable format such as JSON.
* Linking emission data to specific geographical locations using city coordinate datasets.
* Transforming static documents into interactive visualizations for easier analysis and interpretation.

### Business Requirements

#### Entry Points

The system will accept input from two primary sources:

* + - * **PDF Report Input** - Accepts environmental emission reports in PDF format containing unstructured text data.
      * **City Data Input -** Uses the cities500.txt dataset to obtain geographical coordinates for Indian cities, enabling accurate geospatial mapping.

#### Processing Modes

The system will operate in two key processing modes:

* + - * Extracts relevant textual data from PDF reports.
      * Converts the extracted information into structured formats (e.g., JSON) for downstream processing.
      * Enhances structured data with geographical coordinates.
      * Displays enriched data through an interactive map interface to support spatial analysis.

#### Reports Generated

The system will generate the following types of reports:

* + - * **Extraction Report** - Details the content extracted from the uploaded PDF files.
      * **Mapping Report** - Shows which geographical locations were successfully linked with the corresponding emission data.
      * **Analysis Report** - Provides statistical summaries such as emission levels, frequency of pollutant mentions, and other key metrics.

### System Requirements

###### Performance

The system must efficiently process and extract relevant data from PDF documents in a timely manner to ensure responsiveness and scalability

###### Usability

**Input Clarity**: The expected structure and format of the PDF reports should be standardized or clearly defined for accurate parsing.

**Output Clarity**: The structured output (e.g., JSON or CSV) must be easy to understand and suitable for further analysis or visualization tools.

# PROJECT UNDERSTANDING DOCUMENT

### Purpose of Project

The purpose of the “Generative AI-Powered Emission Intelligence” project is to automate the extraction, structuring, and visualization of environmental emissions data from complex and semi- structured PDF reports. It aims to convert fragmented data into a centralized, interactive, and user- friendly format for real-time environmental monitoring. Leveraging cutting-edge generative AI technologies, geospatial tools, and multi-modal reasoning systems, the project seeks to make emissions data accessible and actionable — especially for non-technical users such as policymakers, environmental agencies, and citizens.

### Objective

The main objectives of this project include:

* **Automated Data Extraction:** Use generative AI to parse and extract structured data from unstructured PDFs containing emissions information.
* **Visualization and Mapping:** Employ geospatial libraries to plot and visualize city-level emission data on interactive maps.
* **Semantic Querying:** Utilize LangChain agents and Gemini to enable natural-language querying of emissions data by location, pollutant type, or timeframe.
* **Multi-Modal Interpretation:** Integrate OpenCLIP to interpret emission maps and enable reasoning over images and text.
* **User-Friendly Interface:** Deploy a Gradio-based chatbot interface to facilitate natural-language interaction with the emissions data for all users, regardless of technical background.
* **Scalability:** Ensure the solution can handle multiple city-level reports and scale across different geographic regions in India.

### MIS Reports

The system will generate the following **Management Information System (MIS) reports** for effective decision-making and analysis:

* **City-wise Emission Summary Report:** Displays key emission metrics (e.g., CO2, NOx, SOx levels) for each city over a selected time period.
* **Trend Analysis Report:** Identifies and visualizes temporal trends in emissions across various cities to monitor improvements or deteriorations.
* **Source-wise Emission Distribution:** Breaks down emissions by sources (e.g., industrial, vehicular, power plants) to help target high-emission contributors.
* **Comparative Emission Analysis Report:** Allows comparison of emissions across cities, industries, or states to assess relative environmental performance.
* **Map-Based Interactive Reports:** Provides an intuitive view of emission hotspots on a geospatial map, with the ability to zoom in and explore local insights.

**Custom Query Reports:** Enables stakeholders to request custom reports using natural language queries.

# TIMELINE

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr. No** | **Project Phase** | **Duration (Working Days)** | **No. of Members** | **Category** | **Total Person- Days** |
| 1. | Requirement Analysis & Project Planning | 4 Days | 3 | Research and Coordination | 12 |
| 2. | Environment Setup and API Key Integration | 3 Days | 2 | System Setup and Access Management | 6 |
| 3. | Emission Report  Collection & PDF Parsing | 5 Days | 2 | Document Loading and Cleanup | 10 |
| 4. | Data Structuring via Gemini (LLM) | 5 Days | 2 | Prompt Engineering  and LLM Output Parsing | 10 |
| 5. | CSV Export and DataFrame  Preparation | 2 Days | 3 | Data Formatting and Conversion | 2 |
| 6. | Semantic Embedding of Cities using MiniLM | 3 Days | 2 | Embedding and Vector Representation | 6 |
| 7. | FAISS Index Creation and Query Integration | 3 Days | 2 | Vector Search System | 6 |
| 8. | Emission-to-City Mapping and  Coordinate Linking | 3 Days | 2 | Geo Intelligence Matching | 6 |
| 9. | Folium Map  Visualization with  MarkerCluster | 4 Days | 2 | Interactive Mapping | 8 |
| 10. | Screenshot Capture using Selenium  Automation | 2 Days | 1 | UI Automation for Image Acquisition | 2 |
| 11. | Map Interpretation using OpenCLIP | 3 Days | 2 | Multimodal Reasoning (Image-to-  Text) | 6 |
| 12. | LangChain Agent Tooling and Integration | 5 Days | 2 | LLM Agent  Configuration | 10 |
| 13. | Gradio Chatbot Interface Development | 3 Days | 2 | Front-end UI and LangChain Pipeline | 6 |
| 14. | System Integration and End-to-End Testing | 4 Days | 2 | QA and Error Handling | 8 |
| 15. | Final Report, Diagrams, and Documentation | 5 Days | 3 | Report Compilation and Formatting | 15 |

# REQUIREMENTS

### SPECIFIC REQUIREMENTS

* External Interface Requirements

The system is designed to handle **all data input and output operations** independently. Interfaces are detailed below:

###### PDF Report Input Interface

* **Description:** Ingests environmental emission reports in PDF format.
* **Source/Input:** PDF files (unstructured text).
* **Destination/Output:** Extracted text content.
* **Ranges:** Varying number of pages and entries.
* **Accuracy & Tolerances:** High accuracy required; tolerant to varied formatting.
* **Units of Measure:** Not applicable.
* **Timing:** At script start.
* **Data Format:** .pdf
* **Storage Mechanism:** Local file system.

###### City Data Input Interface

* **Description:** Provides geographical coordinates for cities.
* **Source/Input:** cities500.txt (GeoNames dataset).
* **Destination/Output:** City name to latitude/longitude mappings.
* **Ranges:** Thousands of city records.
* **Accuracy & Tolerances:** Dependent on dataset quality.
* **Units of Measure:** Latitude, Longitude (degrees).
* **Timing:** During geospatial enrichment.
* **Data Format:** Tab-separated values.
* **Storage Mechanism:** Local file system.

###### Structured Data Output Interface

* **Description:** Outputs cleaned and structured emission data.
* **Source/Input:** Processed data from PDFs.
* **Destination/Output:** Structured CSV file.
* **Ranges:** Varying number of rows and columns based on data structure.
* **Accuracy & Tolerances:** Based on extraction accuracy.
* **Units of Measure:** As per PDF (e.g., tonnes, ppm).
* **Timing:** Post data structuring.
* **Data Format:** .csv
* **Storage Mechanism:** Local file system.

###### Map Visualization Output Interface

* **Description:** Outputs interactive map visualizing emission data.
* **Source/Input:** Structured data with coordinates.
* **Destination/Output:** Web-based interactive map.
* **Ranges:** Markers per location with data.
* **Accuracy & Tolerances:** Based on geolocation and data precision
* **Units of Measure:** Latitude, Longitude.
* **Timing:** During visualization phase.
* **Data Format:** .html
* **Storage Mechanism:** Local file system.

#### Hardware Interface

Designed for use on a standard computing setup. Recommended specifications:

* + - * **Storage:** 500 GB HDD
      * **Memory:** 8 GB RAM
      * **Processor:** Intel Core i5 or equivalent
      * **Input Devices:** Keyboard, Mouse
      * **Output Device:** Monitor

#### Software Interface

The system relies on the following software stack:

* + - * Operating System:
      * Windows 10 or Linux

###### Programming Language:

* + - * Python 3

###### Libraries:

* + - * **PyPDF2** – For PDF text extraction
      * **Pandas** – For data structuring and manipulation
      * **Selenium** – For browser automation (if required)
      * **Folium** – For interactive map generation
      * **Sentence Transformers, FAISS, CLIP** – For semantic similarity, embedding search.
  1. NON-FUNCTIONAL REQUIREMENTS

|  |  |
| --- | --- |
| **Category** | **Description** |
| **Performance** | Must process data with minimal delay and respond quickly. |
| **User Friendly** | Simple, intuitive interface for users with varying technical skills. |
| **Flexibility** | Easy to modify features, prompts, or integrate tools. |
| **Extensibility** | Should support future upgrades like voice input, multilingual support. |
| **Portability** | Compatible with multiple platforms (desktop, web, mobile). |
| **Reusability** | Core modules (e.g., extraction logic, mapping engine) should be reusable. |

* 1. SOFTWARE SYSTEM ATTRIBUTES

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| **Reliability** | System should deliver accurate, consistent outputs across runs. |
| **Availability** | Should be accessible 24/7 with minimal downtime. |
| **Maintainability** | Must be easy to update, patch, or add new features. |

# DESIGN TECHNIQUES

### Frontend

###### User Interaction Layer (Gradio + HTML/CSS + JavaScript)

* **Gradio** is used to build a responsive, chatbot-style interface for real-time interaction.
* Provides a **conversational UI** that supports natural language input and dynamic response display.
* Designed to be **accessible and intuitive** for both technical and non-technical users.
* **HTML/CSS and JavaScript** enhance responsiveness, design quality, and enable additional interactive features (e.g., filters, download buttons).

### Backend

###### Processing & Intelligence Layer (Python + LangChain + FastAPI)

* **Python** powers the core logic, processing pipelines, and machine learning components.
* **LangChain** enables dynamic workflows, multi-step reasoning, and integration with external tools.
* **FastAPI** serves as the backend framework, exposing RESTful APIs, handling routing, and managing real-time communication between frontend and backend.
* Also supports secure **authentication and session management**.

### AI & Language Processing

###### (Gemini + NLP Models)

* Leverages **generative language models** (e.g., Gemini) to interpret user queries in natural language.
* Extracts key information from **unstructured PDF emission reports**.
* Delivers **context-aware, real-time responses** to user queries.

### Structured Data Output

###### (JSON & CSV Generator)

* Cleans and organizes extracted data, converting it into **structured formats** such as JSON and CSV.
* Supports use in **dashboards, reporting tools**, and external data pipelines.
* Allows easy **export** for stakeholder analysis and regulatory compliance.

### Geospatial Intelligence Engine

###### (Folium + FAISS + OpenStreetMap)

* **Folium:** Generates interactive maps for geospatial visualization of emission data.
* **FAISS:** Performs fast similarity search and clustering within large spatial datasets.
* **OpenStreetMap (OSM):** Provides high-resolution maps for accurate environmental data overlay.

### Database & Storage

###### (SQL Database + File System)

* Structured emission data, user sessions, and extracted content are stored in a **SQL database**

(e.g., MySQL).

* Handles metadata for uploaded PDFs, user queries, and analysis results using relational tables.
* File system stores actual PDF files and generated visualizations; their references are saved in the database.

# TIER ARCHITECTURE

The Generative AI-Powered Emission Intelligence system is built on a **three-tier architecture** to ensure **modularity, scalability, and maintainability**. Each tier performs distinct roles in data acquisition, processing, and user interaction, enabling seamless updates and future expansion.

### Data Layer

Handles acquisition, storage, and organization of raw and structured data.

###### Key Components:

* **PDF Documents:** Raw emission reports are parsed and cleaned, then converted into structured formats (CSV/JSON).
* **Geospatial Data:** City names with latitude/longitude are stored in GeoJSON or geospatial databases for efficient mapping.
* **Environmental Datasets:** CO₂ levels and other emissions statistics are validated and formatted

for AI analysis.

### Processing Layer

Performs data transformation, AI integration, and geospatial analysis.

###### Key Components:

* **PDF Extraction:** Tools like LangChain’s PyPDFLoader parse and structure textual data.
* **Gemini API:** Uses generative AI to summarize reports and answer queries based on cleaned data.
* **Semantic Search (FAISS):** Enables fast, vector-based similarity search for cities and related content.
* **Geospatial Mapping:** Tools like Folium and MarkerCluster display emission locations on maps. Selenium and OpenCLIP help analyze visual map data.

### Application Layer

User-facing tier offering real-time interaction and visualization.

###### Key Components:

* **Gradio Interface:** Chat-based UI for user queries and map interaction.
* **LangChain Agent:** Core logic that routes user inputs to appropriate tools and returns relevant responses.
* **Multi-Modal Insights (OpenCLIP):** Analyzes map screenshots and provides context-aware insights.
* **Real-time Interaction:** Supports conversational memory and dynamic responses.

###### Summary Table

|  |  |  |
| --- | --- | --- |
| **Layer** | **Components** | **Key Functions** |
| **Data Layer** | PDF files, Geospatial data, Emissions  datasets | Data ingestion, parsing, and  storage |
| **Processing**  **Layer** | Gemini API, FAISS, LangChain,  Folium, OpenCLIP | AI inference, semantic search, map  generation |

|  |  |  |
| --- | --- | --- |
| **Application**  **Layer** | Gradio UI, LangChain agent,  OpenCLIP | User interaction, chatbot, visual  insights |

**Interaction Between Layers**

* **Data Flow:** Emission data is collected in the Data Layer → processed via AI/geospatial tools in the Processing Layer → visualized and presented in the Application Layer.
* **User Queries:** Processed via LangChain Agent; visual queries are interpreted using OpenCLIP.
* **Modular Design:** Each layer operates independently, allowing seamless upgrades and better fault isolation.

###### Key Advantages

* **Modularity:** Easily extend or modify any tier without affecting others.
* **Scalability:** Efficient handling of large datasets via FAISS and AI tools.
* **Separation of Concerns:** Simplifies debugging and enables clean integration of data, logic, and interface.

# SOFTWARE PROCESS MODEL

### Why Not Evolutionary Models?

Evolutionary models are most suitable when the project requirements are ambiguous or expected to evolve over time. However, in this project, the system requirements are **clearly defined and stable** from the outset. Thus, employing an evolutionary approach would introduce unnecessary complexity and inefficiency.

### Why Not the Waterfall Model?

While the Waterfall model supports projects with fixed requirements, it presents several limitations:

* **No parallel development:** All phases follow a strict sequence, limiting team collaboration across modules.
* **Delayed testing and feedback:** Usability issues are often discovered late in the process. Given our **tight deadlines** and need for **modular and parallel development**, the Waterfall model is not an optimal choice.

### Why the Incremental RAD Model?

The **Incremental Rapid Application Development (RAD)** model is the most suitable approach for this project due to the following reasons:

* **Well-defined requirements** are available from the beginning.
* A **short timeline (3 months)** necessitates rapid development cycles.
* The project consists of independent, functional **modules** such as:
* Content extraction and summarization
* Natural language query processing
* Geospatial visualization
* Allows for **parallel development** by multiple teams.
* Supports **early user feedback**, ensuring continuous usability improvements.
* Makes use of **modern tools and frameworks** to accelerate development and deployment.

### Observation

The proposed Generative AI system will **automate manual and repetitive tasks**, significantly improving:

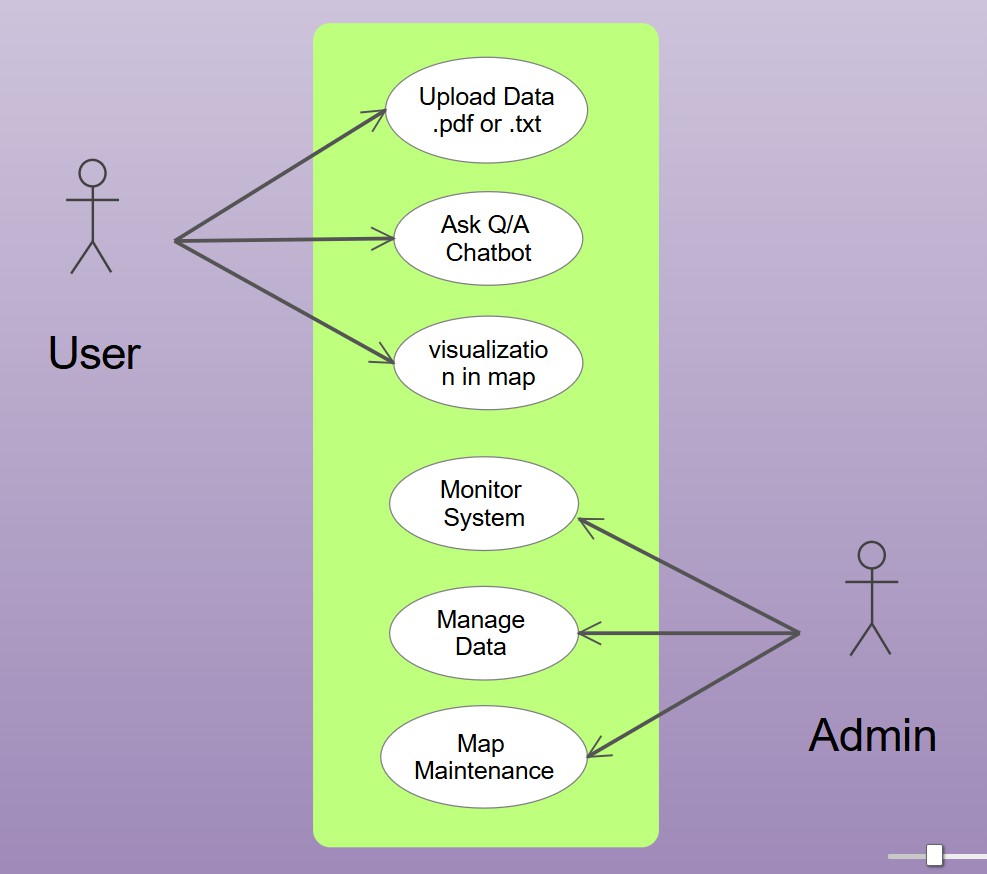
* **Accuracy** of data extraction and presentation
* **Time efficiency** in accessing environmental data
* **Cost-effectiveness** through long-term operational savings
* **User experience** via intelligent, structured content interaction

### Determining Project Feasibility

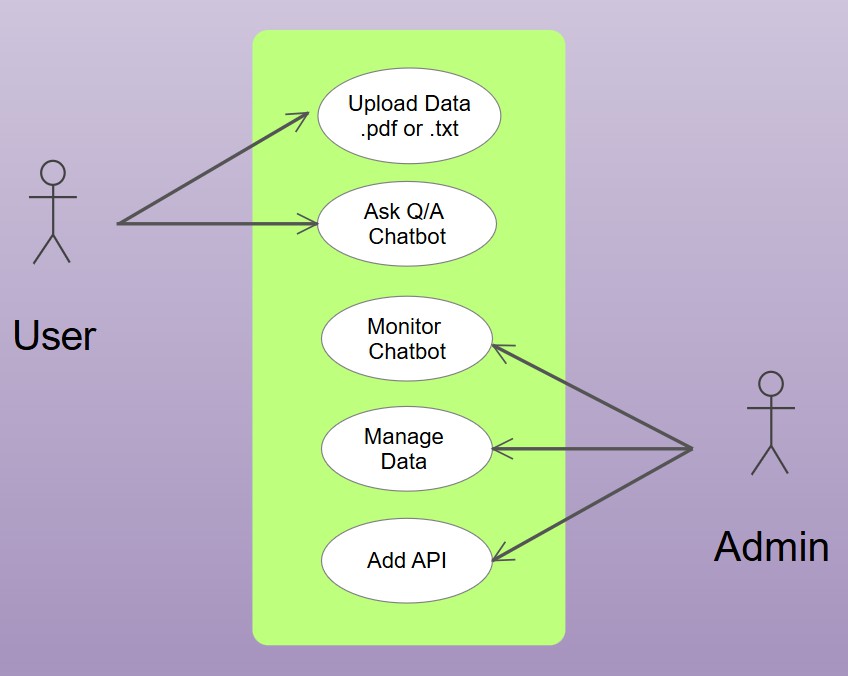
* The investment in system development will result in long-term savings by **eliminating labor- intensive manual tasks** and enhancing productivity, making the solution economically viable.
* The required **software tools, APIs, and hardware infrastructure** are readily available and widely supported, ensuring smooth technical execution.
* The **user-friendly Gradio interface**, combined with natural language interaction, ensures **ease of use** and smooth adoption by target users, replacing existing inefficient workflows.

# DESIGN

##### USE CASE DIAGRAM

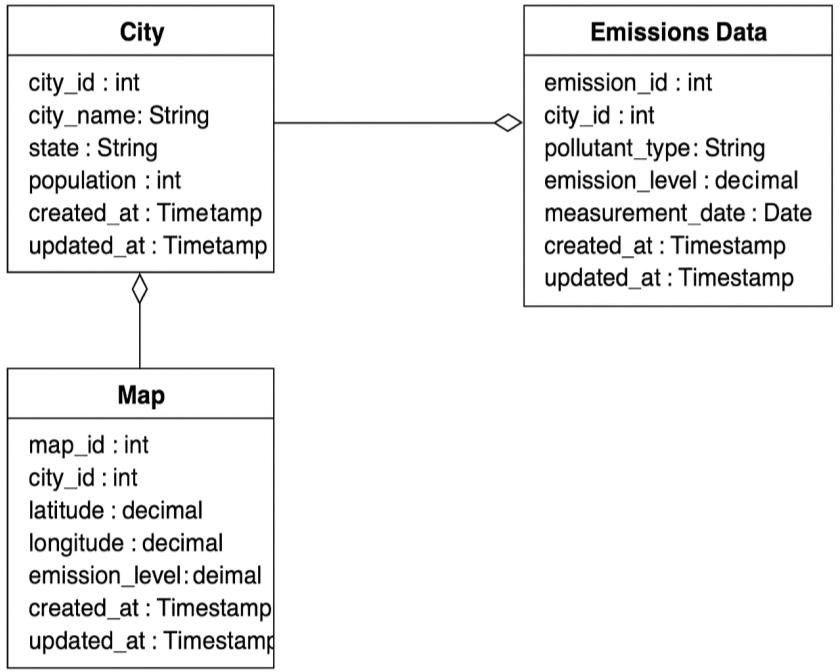


*Fig : 9.1 usecase diagram for ecovision Ai*

**

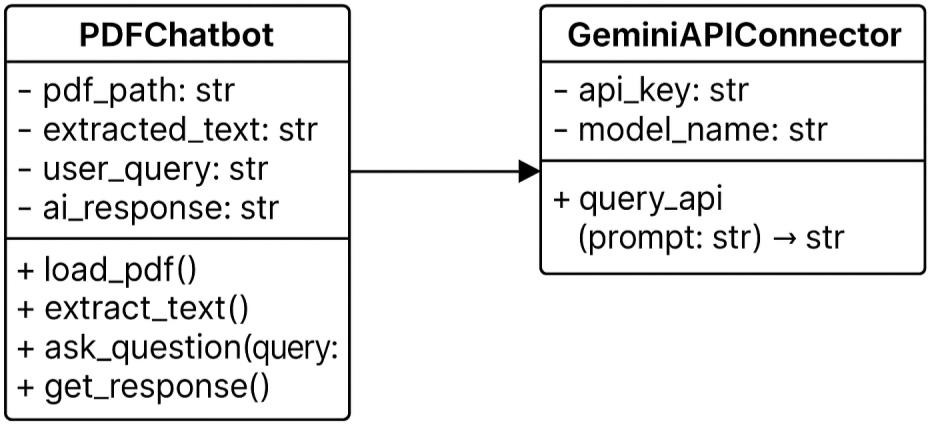
*Fig : 9.2 usecase diagram for ecovision Ai Chatbot for Enviormmnet Emmition*

### Class diagram



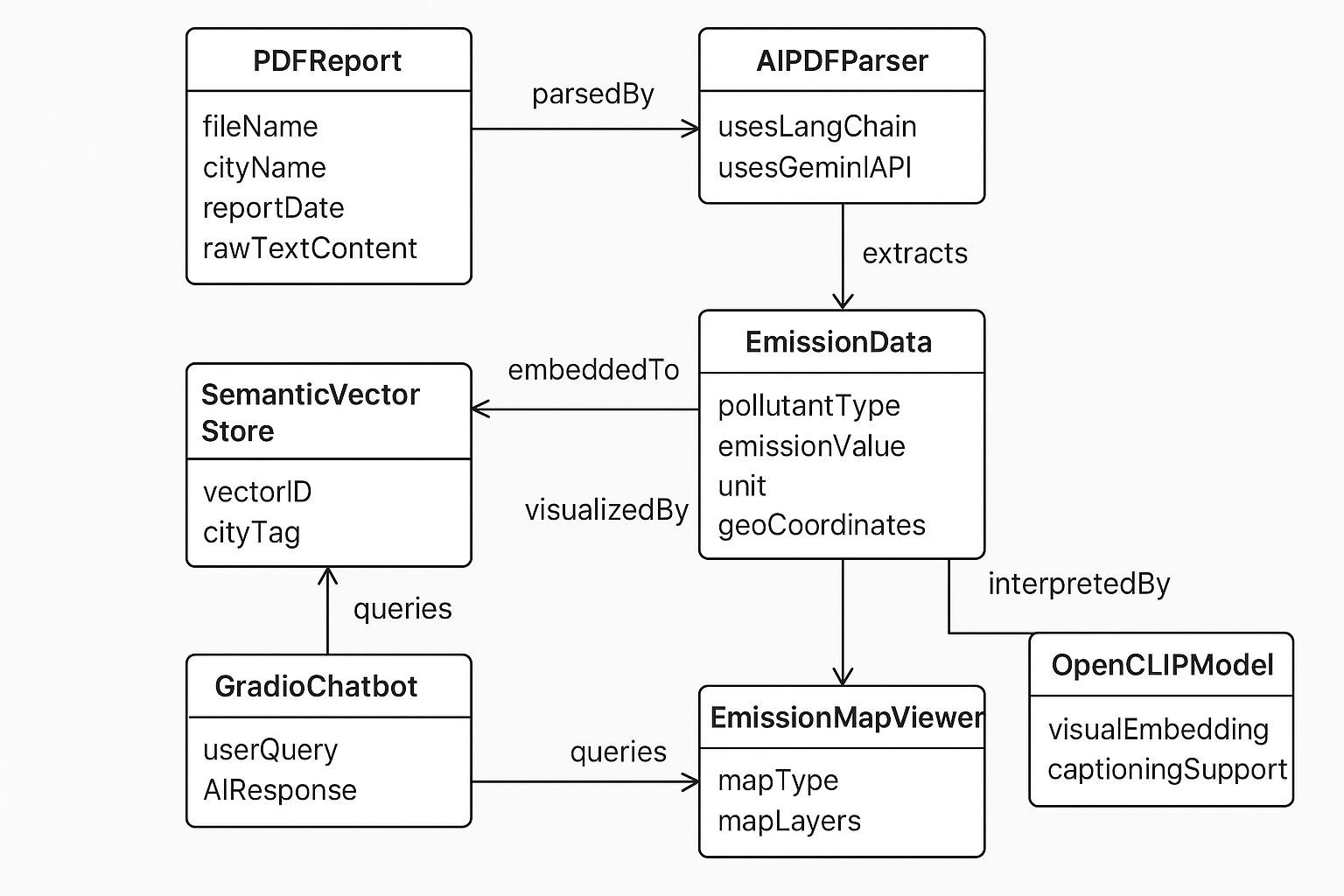
*Fig : 9.2.1 Class diagram for ecovision Ai Chatbot*

### Chatbot Class diagram



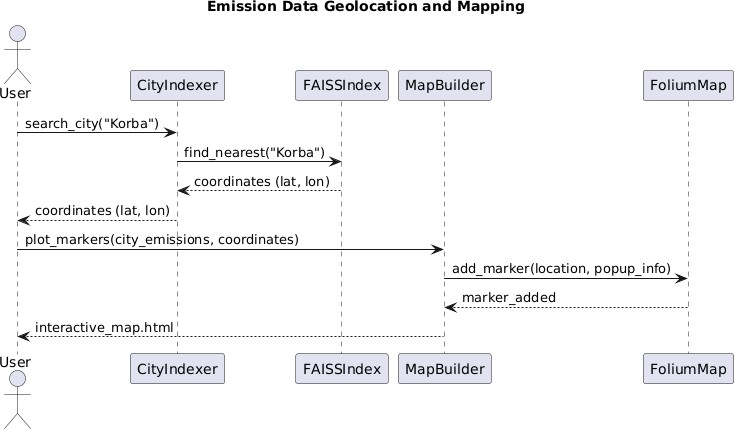
*Fig : 9.2.2 Class diagram for ecovision Ai Chatbot*

### Object diagram

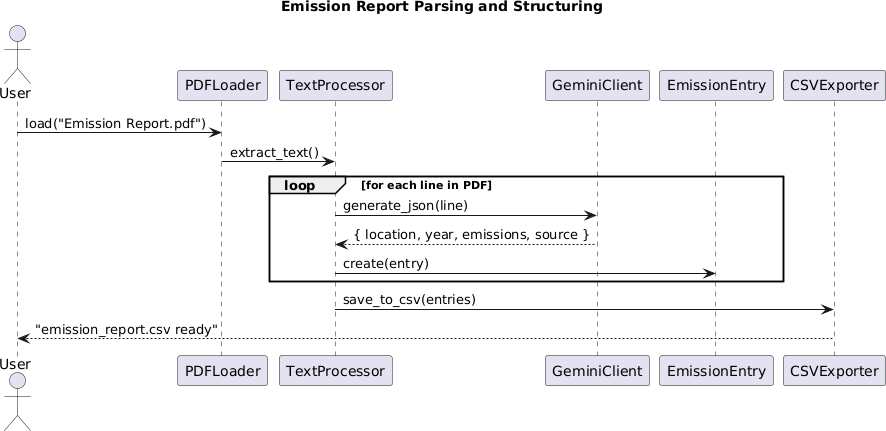


*Fig : 9.3.1 object diagram for ecovision Ai Chatbot*

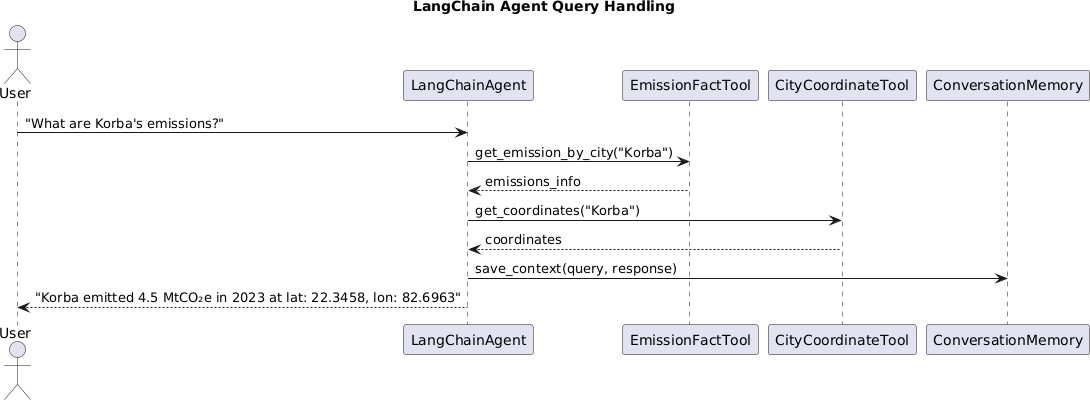
### Sequence diagram



* + 1. *Emission Data Geolocation and mapping*

**

* + 1. *Emission Data Geolocation and mapping*

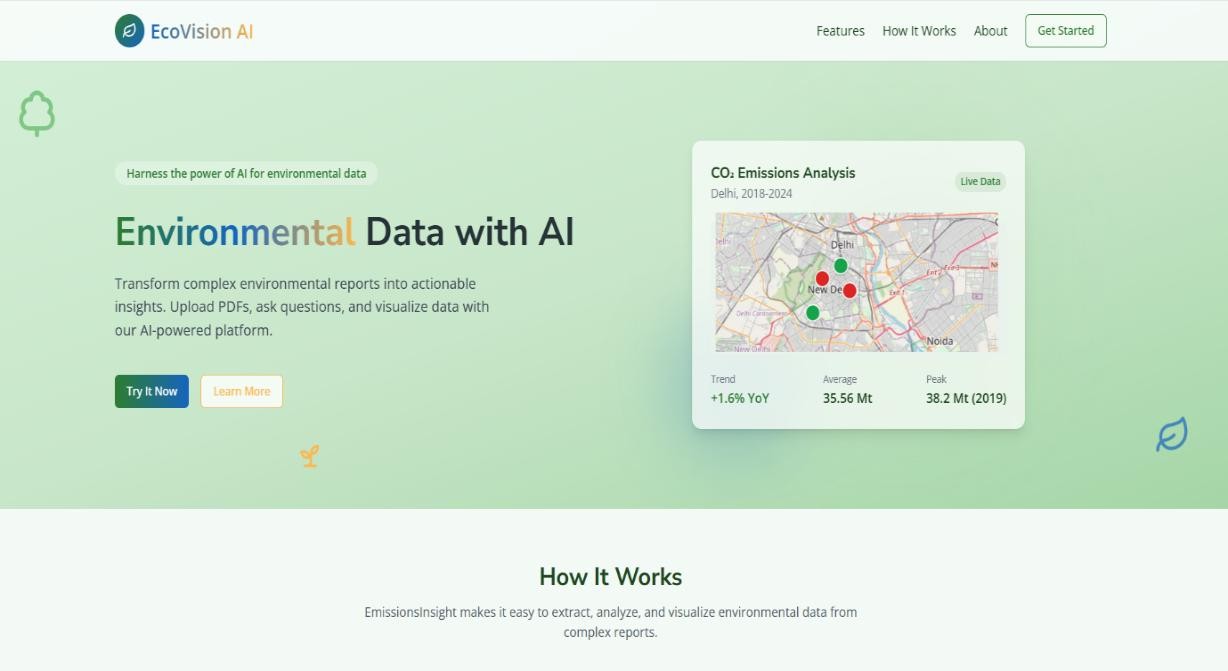


* + 1. *Emission Data Geolocation and mapping*

## expected outcome

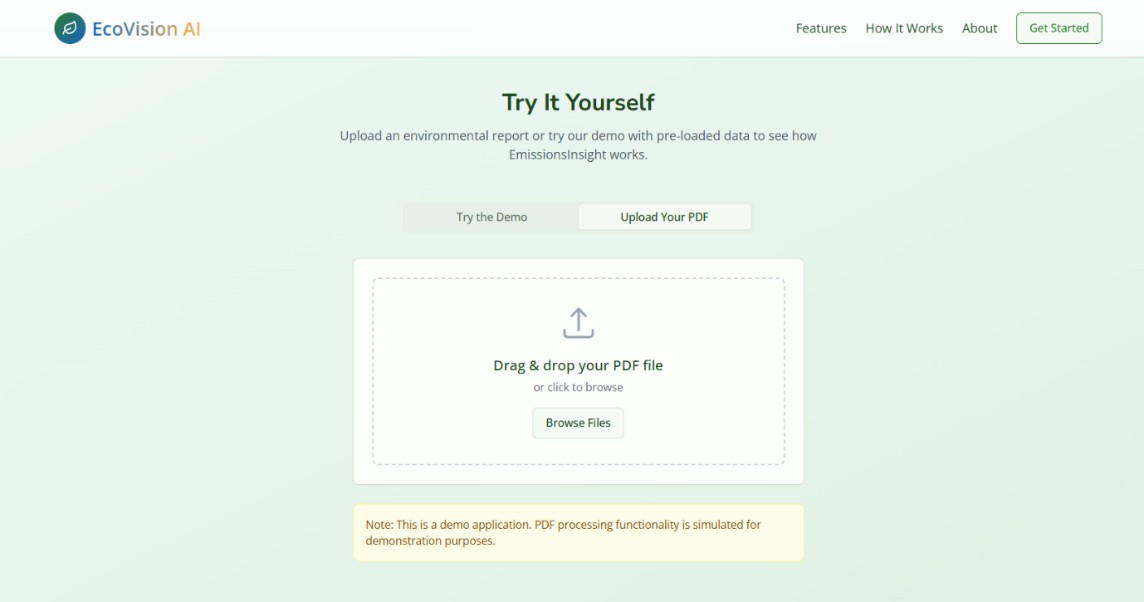
### Home Page

The homepage of EcoVision AI provides a clean interface to explore AI-powered environmental data insights. It includes key features like emissions analysis, navigation links, and quick access buttons like "Try It Now".



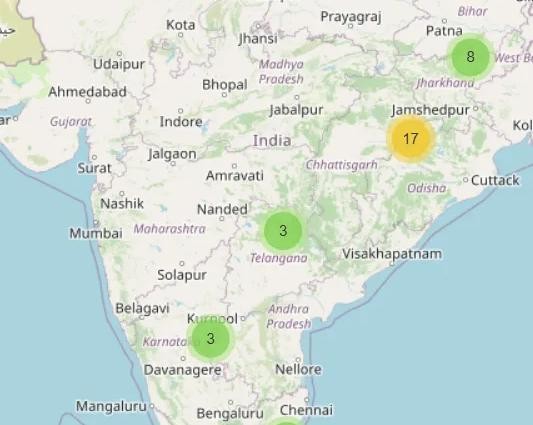
### Data Upload Section

In the upload section, users can upload their own custom data in the form of PDF files.



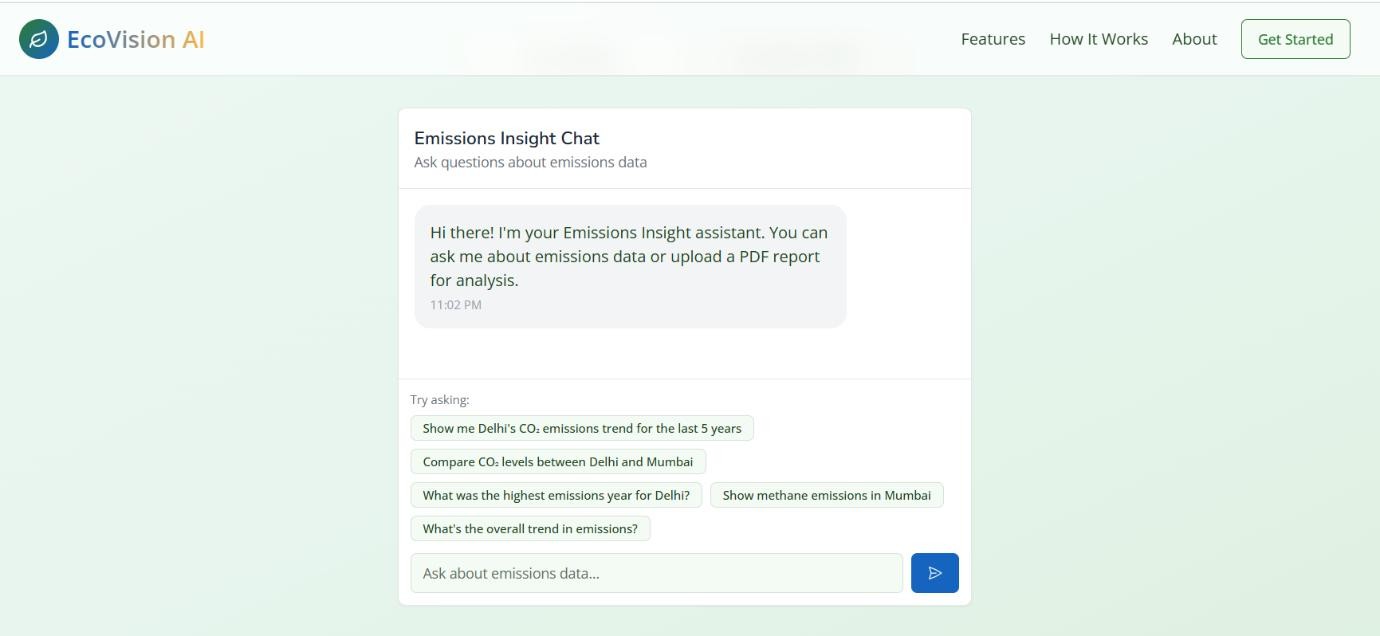
##### Emission Data Plotted in Map

This map displays emission data for specific areas, showing the intensity and distribution of emissions across regions.



### EcoVision Ai Chatbot

In this chatbot, users can ask questions related to emissions, such as how many areas are polluted or what the carbon emissions are in a particular area.



## Conclusion

The Generative AI-Powered Emission Intelligence project presents a groundbreaking step toward modernizing environmental monitoring through advanced technologies. By combining generative AI, geospatial analysis, and multimodal reasoning, the system efficiently transforms complex, unstructured emissions reports into structured, interactive, and visual data representations.

This end-to-end pipeline—from PDF parsing to semantic search and chat-based insights—enables real-time, accessible environmental intelligence. It eliminates the manual effort traditionally required to analyze fragmented data and empowers users, including non-technical stakeholders, to explore and understand emissions trends across Indian cities with ease.

#### Expected Impact

* Informed Decision-Making: Policymakers, environmentalists, and urban planners can make faster and more informed decisions based on real-time, city-level emissions data.
* Environmental Awareness: Citizens gain clearer insights into pollution levels in their regions, leading to greater public awareness and collective responsibility.
* Health Benefits: By identifying high-emission zones, authorities can implement timely actions to reduce pollution exposure, thus improving public health outcomes.
* Social Equity: Communities disproportionately affected by pollution can be identified and prioritized for support, fostering a more equitable approach to environmental justice.
* Transparency & Accountability: Open access to structured environmental data increases transparency and holds industrial and governmental bodies accountable.