ENVIRO-SCAN

AI-POWERED GEOSPATIAL POLLUTION ANALYSIS SYSTEM

Table of Contents

1 Requirements

1.1 Purpose

Air pollution has become a major environmental and public health concern across the world, This lack of source identification limits the ability of authorities and urban planners to take targeted, data-driven actions.

- Air pollution poses a critical threat to environmental and public health worldwide.
- Existing monitoring systems measure pollutant levels but fail to identify the exact pollution sources (industrial, vehicular, agricultural, etc.).
- This gap limits data-driven decision-making by authorities and urban planners.
- Current solutions lack integration of real-time sensor data, weather information, and geospatial analytics for deeper insights.
- Hence, there is a need for an AI-powered system that predicts pollution sources and visualizes them on geospatial maps to enable proactive environmental management.

1.2 Requirement & constraints

A. **Functional Requirements**

Hardware Requirements

- A personal computer or laptop with a minimum 8 GB RAM and Intel i5 processor.
- Internet connectivity for accessing APIs and live environmental data.
- Optional GPU support for faster machine-learning model training.

Software Requirements

- Programming Language: Python 3.x
- Libraries/Frameworks: scikit-learn, pandas, numpy, geopandas, folium, Streamlit
- APIs & Tools: OpenAQ API, OpenWeatherMap API, OpenStreetMap (OSMnx)
- Version Control: Git / GitHub for collaboration and documentation storage
- Operating System: Windows 10 / Linux (Ubuntu)

System Resources

- Cloud or local storage to maintain datasets (CSV / JSON format).
- Access to online map tiles for geospatial visualization.

B. **UI (User Interface) Requirements**

- Interface must be intuitive and easy to navigate for environmental officers and researchers.
- Provide clear input options (city, date range, pollutant type) with labeled buttons and dropdowns.
- Display interactive geospatial heatmaps with zoom and filter controls.
- Use responsive design to ensure compatibility with both desktop and mobile browsers.
- Maintain aesthetic consistency with light color themes, icons, and readable typography.
- Ensure accessibility by providing readable fonts and high-contrast visual elements.

C. **Database Requirements**

- The system must store and retrieve pollution, weather, and geolocation data efficiently.
- Database (local or cloud-based) must be accessible to the AI prediction module.
- Maintain structured tables for:
  - Pollution readings
  - Weather parameters
  - Predicted source types and confidence values
  - User inputs / query logs
- Ensure data integrity and prevent duplication during continuous data collection.

D. **constraints**

1. Time Constraints
   - Project completion targeted by end of current semester .
   - Each module (data collection, model training, visualization, dashboard) to be finished.
2. Technological Constraints
   - Dependent on third-party APIs for data availability.
   - Requires internet access for real-time data fetching and visualization.
3. Legal & Ethical Constraints
   - Must comply with open-data usage policies of APIs.
   - Ensure no personal or location-sensitive information is stored or exposed.
4. Cost Constraints
   - Project to be implemented using free or open-source tools and public APIs.
   - No additional licensing or paid subscription costs involved.
5. Environmental Constraints
   - Real-time API data may vary depending on local weather conditions or network latency.
   - Accuracy may be affected in regions with sparse sensor coverage

1.3 Engineering Standards

A.● IEEE 1008-1987 - Software Unit Testing
○ This describes that the process of creating software is determined by different phases, activities, and tasks, which will be upheld by us while creating our project.

● WCAG 2.1 - Web Content Accessibility Guidelines
    ○ This is a web application, therefore it is necessary to provide equal opportunity

● IEEE 12207-1996 - Software Life Cycle Process
    ○ This lays out a common framework for developing and managing software projects.
    ○ https://standards.ieee.org/standard/12207-1996.html

● IEEE 16085-2020 - Risk Management
    ○ Provides guidance on risk management associated with software projects
    . ○ https://standards.ieee.org/standard/16085-2020.html

1.4 Intended Users and Uses

Intended Users

- Environmental Agencies: For monitoring and identifying pollution sources.
- Urban Planners & Researchers: For analyzing spatial pollution patterns.
- Government Policy Makers: For informed environmental policy decisions.
- Public / NGOs: For awareness and advocacy of pollution control.

Intended Uses

- Detect and classify major pollution sources (industrial, vehicular, agricultural).
- Display real-time pollution heatmaps and alerts.
- Generate analytical reports for decision-making and planning.
- Support environmental studies through data-driven insights.

2 Project Plan

2.1 Project Management/Tracking Procedures

A hybrid approach combining Waterfall and Agile methodologies will be adopted for the EnviroScan project.

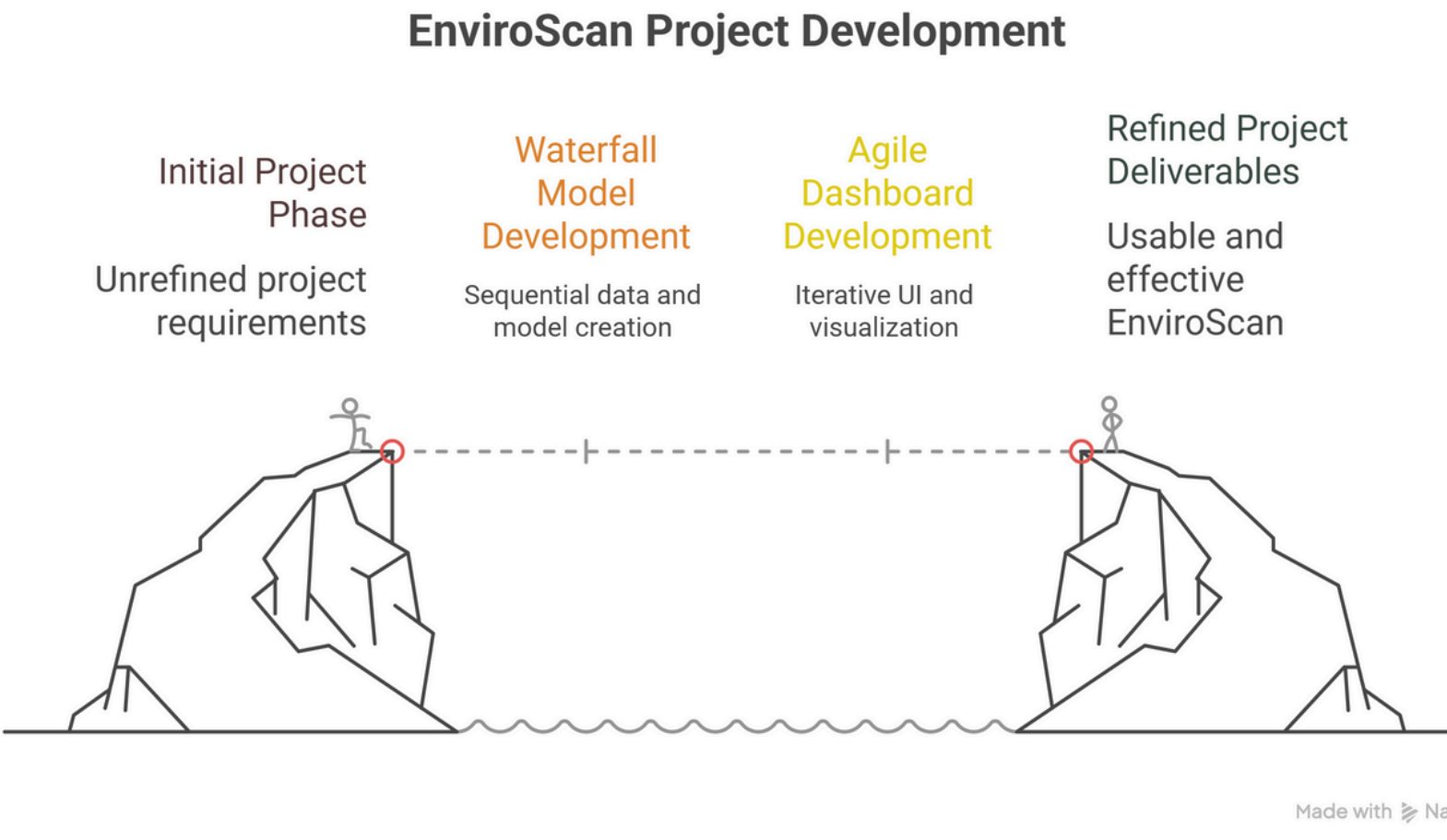
Each major stage of development has a defined goal and deliverable while allowing iterative refinement of AI models and UI components.

Waterfall Approach – Data Collection and Model Development

- Clearly defined stages: data collection, cleaning, feature engineering, model training, and evaluation.
- Sequential execution ensures each module is verified before moving to the next.
- Limited client interaction required during model design; stakeholders are updated after milestone completion.

Agile Approach – Visualization and Dashboard Development

- User interface and visualization modules will be developed iteratively.
- Stakeholders (faculty, reviewers) can provide feedback at each iteration.
- Continuous integration allows incremental updates to the dashboard for usability improvement.



2.2 Task Decomposition

The EnviroScan system will be divided into four module sub-projects and one continuous documentation phase.
Each sub-project will include research, design, implementation, and testing components.

1)Module :- Data Collection and Integration

- Research and identify data sources .
- Extract environmental, weather, and location-based data through APIs.

2)Module:- Data Processing and Feature Engineering

- Clean and preprocess raw data by removing duplicates and handling missing values.
- Create proximity-based and temporal features (distance to road/industry, time, season).
- Normalize datasets and prepare them for model training.

3)Module:- AI Model Development

- Select suitable algorithms (Random Forest, XGBoost).
- Train and test models on labeled pollution-source datasets.
- Evaluate model performance using accuracy, precision, recall, and F1-score.
- Export trained model for dashboard integration.

4)Module:- Visualization and Dashboard Development

- Design interactive dashboard using Streamlit.
- Integrate geospatial visualization with Folium and GeoPandas.
- Add real-time pollution heatmaps, alerts, and reporting features.

5)Module:- Documentation and Testing

- Maintain weekly progress logs and reports.
- Conduct functional and integration testing.

.

EnviroScan System Analysis

A structured breakdown of the EnviroScan system encompassing its sub-projects and documentation phase.

Data Collection and Integration 📊

This initial phase focuses on collecting and integrating relevant environmental, weather, and geolocation data.

- Research sources like government databases, weather stations, and public APIs for environmental data.
- Extract structured and unstructured data through API calls and web scraping, ensuring adherence to data regulations.
- Consolidate data from various sources, making it ready for preprocessing and analysis.

Data Processing and Feature Engineering 🛠️

After data collection, this phase prepares and enriches data to improve model performance.

- Remove duplicates and handle missing values using data imputation techniques.
- Create meaningful features like distance to environmental hazards, seasons, and time-based metrics.
- Normalize datasets and standardize inputs to increase consistency across data types.

AI Model Development 🤖

Develop and train AI models to analyze environmental data and predict pollution levels.

- Select relevant algorithms like Random Forest and XGBoost based on requirements and data characteristics.
- Train machine learning models using labeled datasets for pollution-source classification.
- Evaluate the performance of models using metrics like accuracy, precision, recall, and F1-score.

Visualization and Dashboard Development 📉

Create an intuitive dashboard for displaying environment and pollution data in real-time.

- Design a functional user interface using Streamlit for interactive user experience.
- Incorporate geospatial visual tools like Folium and GeoPandas for mapping data on a geographical plane.
- Implement features like real-time pollution heatmaps, push alerts, and customized reporting for monitoring.

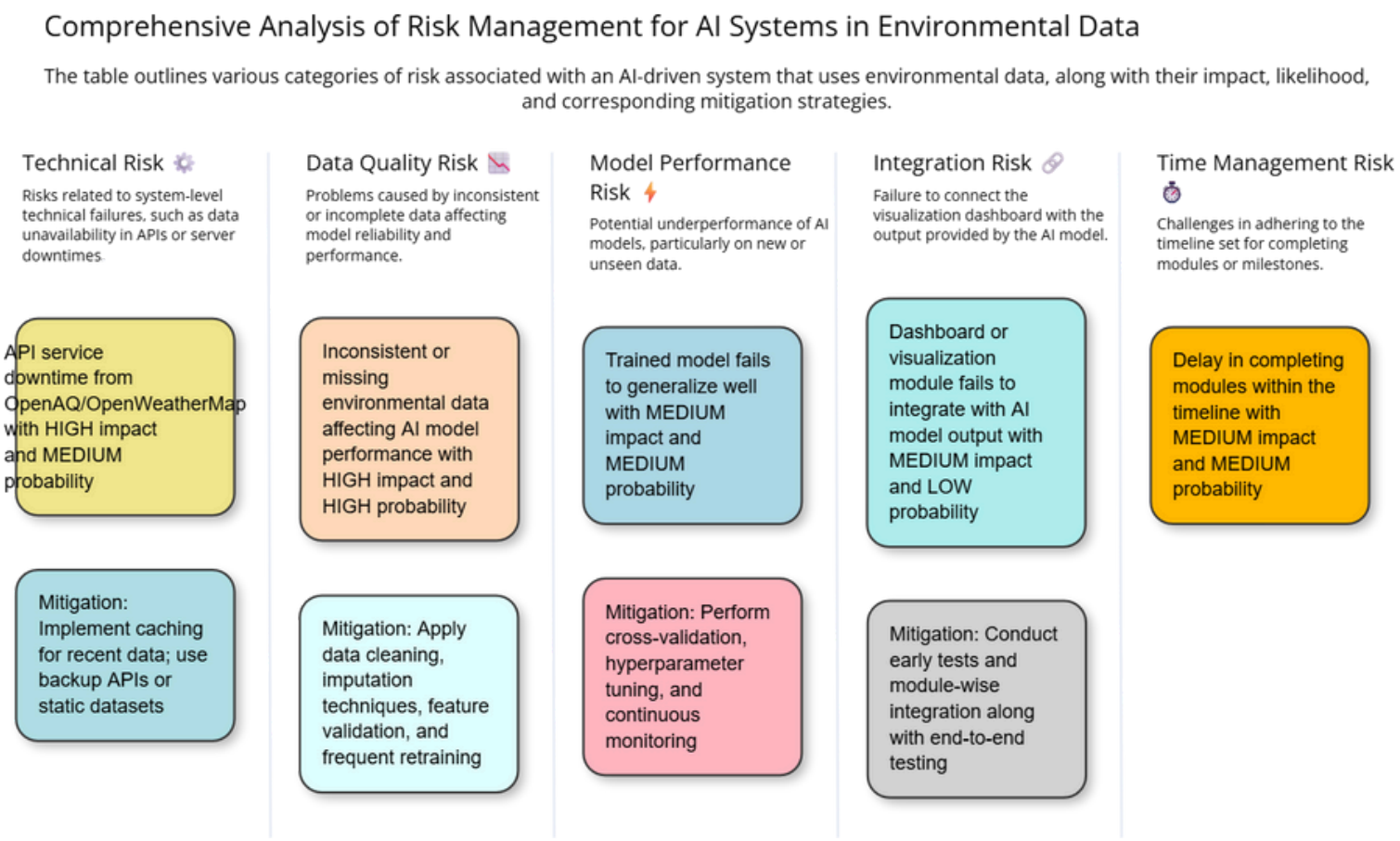
Documentation and Testing 📜

A continuous phase to track development and ensure product quality.

- Maintain weekly reports to document progress, issues encountered, and resolutions.
- Conduct extensive functional testing to validate every component of the solution.
- Perform thorough integration testing to ensure smooth interaction between all system components.

2.3 Risks and Risk Management

Every software project involves uncertainties that may affect its progress, accuracy, or delivery. The following table identifies the potential risks in the EnviroScan project and outlines the corresponding mitigation strategies.



Overall Risk Management Strategy:-

- **Continuous Monitoring**: Maintain a risk register and review progress weekly.
- **Early Testing**: Validate every component before integration to detect issues early.
- **Version Control:** Use Git for rollback in case of system or data failure.
- **Backup Planning:** Keep local copies of datasets, trained models, and documentation.
- **Team Coordination**: Regular progress meetings and Trello updates to track mitigation actions.
- **Quality Assurance**: Conduct both functional and non-functional testing at each stage.

3 DESIGN

3.1 Design Context

3.1.1 Broader Context

The goal of this project is to design an AI-powered web-based system that identifies and analyzes sources of air pollution using geospatial analytics and real-time environmental data. The system aims to assist government agencies, environmental researchers, and policy makers in understanding pollution patterns, locating emission sources, and taking targeted mitigation actions.

Currently, most air quality monitoring platforms only display pollutant concentration levels without identifying the source type or origin. This project bridges that gap by integrating AI-driven source prediction, weather data correlation, and geospatial visualization into a single interactive dashboard.

EnviroScan promotes data-driven environmental planning and supports global sustainability efforts by enhancing pollution intelligence and enabling proactive decision-making.

Area Description Examples

Area	Description	Examples
Public Health, Safety, and Welfare	The project supports public welfare by providing authorities and citizens with actionable insights about pollution hotspots and causes. This helps reduce health risks associated with poor air quality and improves urban livability.	- Identifies and alerts high-risk pollution zones. - Assists in implementing timely pollution control measures. - Reduces long-term exposure-related illnesses.
Global, Cultural, and Social	Air pollution is a global issue transcending borders and cultures. The system's data-driven insights can be adapted to any region, supporting sustainable policies and global climate initiatives.	- Promotes environmental awareness among citizens. - Encourages collaboration between local and international environmental bodies. - Aligns with UN Sustainable Development Goals (SDG 3 & 13).
Environmental	The project directly contributes to environmental protection and sustainability. By identifying pollution sources, it supports emission reduction and cleaner ecosystems.	- Reduces unnecessary industrial emissions through targeted monitoring. - Promotes eco-conscious decision-making by authorities. - Uses digital tools instead of physical survey methods, minimizing environmental footprint.
Economic	The system offers a **cost-effective**, **scalable**, and **open-data-based** solution for pollution analysis. It minimizes costs associated with manual surveys and resource-heavy monitoring.	- Uses free/open APIs and tools. - Low operational and maintenance cost. - Supports efficient urban resource management and policy planning.

3.1.2 User Needs

Government Environmental Agencies

- Need a platform that provides real-time identification of pollution sources to plan effective mitigation measures.
- Require visual pollution maps to monitor and prioritize high-risk regions.
- Need access to analytical reports summarizing pollutant trends and sources for policy enforcement.

Researchers and Environmental Scientists

- Require accurate data aggregation from multiple sources (pollution, weather, geolocation).
- Need an AI-assisted analytical platform to study spatial pollution patterns and environmental correlations.
- Desire an exportable dataset or visualization for publication and research purposes.

Urban Planners and Policymakers

- Need actionable insights to plan infrastructure, industrial zones, and traffic routes with minimal pollution impact.
- Require predictive insights to evaluate the effect of policy changes or environmental regulations.

General Public / NGOs

- Need a user-friendly dashboard to view pollution hotspots in their city.
- Require alerts and reports when pollution levels exceed safe thresholds.
- Desire accessible and transparent environmental information to promote awareness.

System Administrators / Developers

- Need the ability to manage API connections, update datasets, and monitor model accuracy.
- Require secure storage of data and model files.
- Desire simple deployment and maintenance workflows using cloud or local servers.

3.1.3 Prior Work/ Solutions

From the analysis of existing environmental monitoring platforms, several solutions provide pollution measurement but not source identification or AI-driven analysis.

Existing Tools:

- AirVisual (IQAir): Displays real-time pollutant concentrations but lacks pollution source detection or cause analysis.
- World Air Quality Index (WAQI): Offers global air quality data but only focuses on sensor readings and AQI visualization.
- OpenAQ Platform: Provides open-source pollution datasets but requires separate analytical tools for insight generation.

Gap Identified:

- Most tools stop at measurement — none provide intelligent source classification (industrial, vehicular, agricultural, etc.)
  using machine learning and geospatial analytics.

Unique Feature of EnviroScan:

- Combines pollution + weather + spatial data into a unified model.
- Uses AI algorithms to detect likely pollution sources.
- Visualizes results on an interactive, filterable geospatial map.
- Provides real-time alerts and policy-relevant insights to support decision-making.

EnviroScan thus bridges the analytical and visual intelligence gap between data collection and actionable environmental insights.

3.1.4 Technical Complexity

Interdependent Modules

- Data Dependency: Real-time collection from multiple APIs (OpenAQ, OpenWeatherMap, OSMnx) introduces synchronization challenges.
- Module Interaction: Integration between data preprocessing, AI prediction model, and geospatial visualization requires seamless data flow.
- Dynamic Updates: Continuous refresh of data and map visualization demands efficient caching and data management.

Technical Interfaces

- API Integration: Connect to multiple APIs with different data formats and authentication methods.
- Geospatial Visualization: Integration of Folium, GeoPandas, and Streamlit to render live maps dynamically.
- Model Deployment: Integration of machine learning models into a web framework for real-time inference.

Computational and Algorithmic Complexity

- The AI model involves multi-dimensional features (pollutant levels, proximity factors, weather data) and requires complex feature engineering.
- Model training and prediction must balance accuracy and processing efficiency to work in near real time.

Data Management Challenges

- Handling large, continuous datasets while ensuring data quality and consistency.
- Maintaining structured storage formats for API responses and historical data analysis.

Security and Accessibility

- Secure handling of API keys and datasets.
- Ensuring cross-platform dashboard access (desktop and mobile) with responsive design

Environmental Monitoring Mind Map

A structured analysis highlighting user needs and technical solutions demonstrated in the environmental monitoring mind map.

Data Analysis

Empower researchers with multi-source aggregated data and advanced analytical tools.

- AI-driven insights for spatial pollution correlations.
- Exportable visualizations suitable for scientific publications.
- Integration of pollution, geolocation, and weather data.

Infrastructure Planning Insights

Provide policymakers with predictive analytics for informed urban planning decisions.

- Policy evaluation to predict pollution impact from changes.
- Recommendations for environmentally conscious infrastructure projects.
- Data-driven strategic zoning for industrial and traffic planning.

Pollution Source Detection

Use machine learning to pinpoint and classify pollution sources effectively.

- Detection of vehicle emissions as a major pollutant.
- Analysis of agricultural practices influencing airborne pollutants.
- Identification of factory and industrial outputs contributing to air quality decline.

Secure System Management

Provide developers with efficient tools for system security and deployment.

- API integration to manage authentication and data formatting.
- Cloud and local deployment to streamline development workflows.
- Encrypted storage for safeguarding datasets and models.

3.2 Design Exploration

3.2.1 Design Decisions

We will need to make several design choices for the development of the EnviroScan system.

The project involves multiple components such as data processing, AI model development, and geospatial visualization.

Key decisions include:

- Frontend: Select a framework for dashboard and visualization → Streamlit
- Backend: Select a framework for server-side operations and data management → Flask (Python)
- Machine Learning: Choose algorithms for pollution source prediction → Random Forest / XGBoost
- Database: Choose format and structure for storing collected and processed data → SQLite / PostgreSQL
- Architecture: Decide on overall system architecture and API integration flow → Modular layered architecture
- Visualization: Select a mapping library for geospatial rendering → Folium / GeoPandas
- Communication: Define interaction between AI model, backend API, and visualization modules.

3.2.1 Design Decisions

We will need to make Among the key design decisions, selecting the most appropriate framework and visualization technology is both critical and complex.

The decision was guided by criteria such as ease of use, community support, data-science compatibility, and visualization quality.

The following options were initially considered for different components of the project:

Component	Options Considered	Factors for Evaluation
Frontend Framework	Streamlit, Dash, Flask (UI layer), React	Ease of integration with Python ML backend, lightweight, rapid prototyping
Backend Framework	Flask, FastAPI, Django	API flexibility, learning curve, and deployment simplicity
Mapping Library	Folium, Kepler.gl, Leaflet.js, Plotly	Interactive visualization, geospatial support, and integration with Python
Machine Learning Algorithms	Random Forest, XGBoost, Decision Tree, SVM	Accuracy, interpretability, and computational efficiency
Database	SQLite, PostgreSQL, MongoDB	Ease of setup, storage size, geospatial data handling capabilities

- After evaluating pros and cons, **Streamlit + Flask** chosen as the development base due to their simplicity, flexibility, and native compatibility with Python's data-science ecosystem.

3.2.3 Decision-Making and Trade-Off

Framework / Tool	Pros	Cons
Streamlit	- Very easy to build dashboards directly from Python scripts.- Supports live data and model interaction.- Quick UI deployment with minimal frontend code.- Compatible with Pandas, Matplotlib, Folium, etc.	- Limited in customization compared to full web frameworks.- Not suitable for highly complex frontends.
Flask	- Lightweight and flexible.- Easy to create REST APIs for ML model interaction.- Strong community support.- Simple deployment process.	- Requires manual management of routes and APIs.- Not as feature-rich as Django for large-scale systems.
Random Forest	- High accuracy with limited parameter tuning.- Handles nonlinear data well.- Interpretable feature importance.	- Slower on large datasets.- May require optimization for real-time inference.
XGBoost	- Superior performance and speed.- Excellent generalization and robustness.- Highly tunable parameters.	- More complex to implement and fine-tune.- Higher computational cost for training.
Folium	- Integrates seamlessly with Python and Jupyter/Streamlit.- Interactive map visualization with heatmaps and markers.- Based on open-source Leaflet.js.	- Limited customization of map style.- Heavy for large datasets; may affect rendering speed.
PostgreSQL (with PostGIS)	- Excellent support for geospatial data.- Open source and scalable.- Reliable query performance.	- Slightly higher learning curve for PostGIS setup.
SQLite	- Simple setup, no server installation needed.- Ideal for prototype and local testing.	- Not suited for large concurrent data or multi-user access.

3.3 Proposed Decisions

3.3.1 Design Visual and Description

The proposed EnviroScan system will collect, process, and analyze environmental data to identify pollution sources and visualize results on an interactive geospatial dashboard.

The workflow begins with the data collection module, which gathers air quality data (e.g., PM2.5, CO, NO_2 levels) and weather information (wind direction, temperature, humidity) from public APIs such as OpenAQ and OpenWeatherMap.

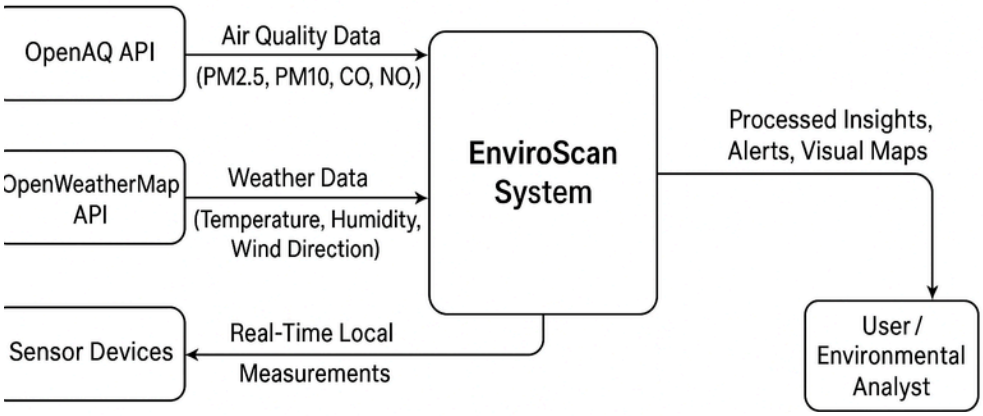
This data is then stored in the project database and passed to the AI-based Source Identification Module, which classifies the likely source of pollution (industrial, vehicular, biomass burning, or natural).

The final results are displayed on a Streamlit-powered dashboard, where users can view pollution intensity, sources, and geospatial spread through interactive maps built with Folium and GeoPandas.

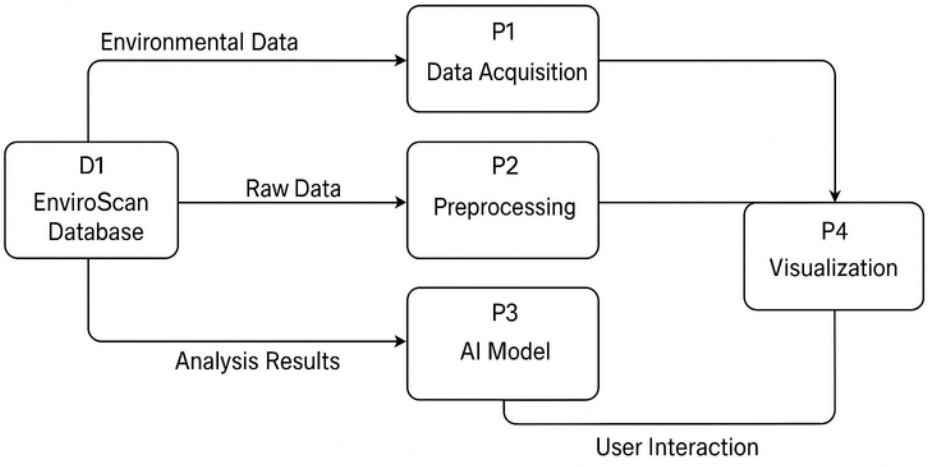
High-level Data Flow:

1. Data Acquisition (APIs, Sensors, External Datasets)
2. Preprocessing and Cleaning
3. AI Model Prediction (Source Identification)
4. Geospatial Visualization (Map Rendering)
5. User Interaction (Reports, Filters, and Alerts)

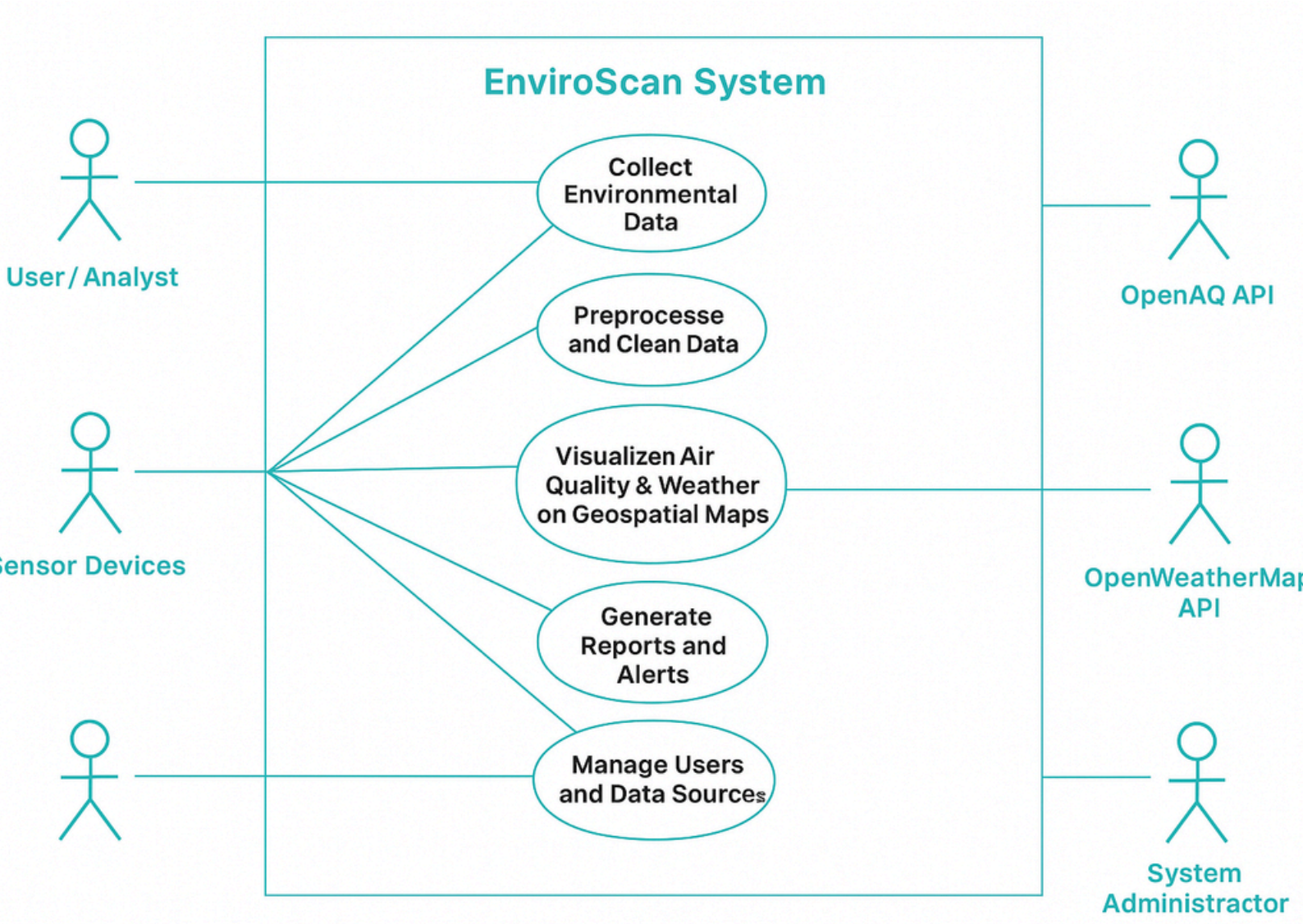
Level 0
Context Diagram



3.3.3
Level 1 Data Plow Diagram



USE CASE DIAGRAM



3.4 Design Analysis

So far, the proposed design of the EnviroScan system has performed as intended during the early stages of development and testing.

Initial data collection and preprocessing modules have been successfully implemented using OpenAQ and OpenWeatherMap APIs, confirming stable data acquisition and integration workflows.

Preliminary tests of the AI pollution source classifier (using Random Forest and XGBoost) show promising results with consistent prediction accuracy on sample datasets.

The integration between the Flask backend, AI model, and Streamlit dashboard is working smoothly, enabling real-time visualization of pollution hotspots and their probable sources.

Ongoing design validation focuses on:

- Testing the accuracy and reliability of pollution source predictions under varying environmental conditions.
- Optimizing geospatial rendering performance for large data inputs.
- Enhancing database storage structure for efficient query handling and future scalability

3.5 Design Plan

Our design plan is structured around three primary modules:

1. Frontend (User Interface and Visualization)
2. Backend (Data Management and API Integration)
3. AI & Geospatial Analytics Engine (Core Intelligence Module)

The following subsections and diagrams outline the structure and functionality of each component.

3.6.1 Frontend

UI Diagram

(Here you can insert a simple wireframe or block diagram showing your Streamlit dashboard — e.g., sidebar filters, map area, charts.)

The frontend of EnviroScan is developed using Streamlit, designed for simplicity, interactivity, and cross-platform accessibility.

It provides users (government agencies, researchers, and the public) with clear, visual, and analytical insights into pollution patterns and sources.

Overview of the Dashboard Interface:

- Home Page:
- Displays an overview of air quality index (AQI) trends and quick links to regional pollution reports.
- Geospatial Map View:
- Interactive map showing pollution concentration and detected sources (industrial, vehicular, etc.) using color-coded layers.
- Users can zoom, filter by pollutant type, or view historical data overlays.
- Source Analysis Panel:

- Displays real-time predictions of pollution sources, accompanied by model confidence levels and comparison charts.
- Analytics and Reports Page:
- Summarizes pollutant variations, correlations with weather factors, and regional comparisons over selected time frames.
- Admin Controls (restricted access):
- Allows authorized personnel to upload datasets, trigger model retraining, and manage API configurations.
- UI Goals:
- Maintain a clean, minimal, and informative layout suitable for both technical and non-technical users.
- Ensure full responsiveness on desktops, tablets, and mobile devices.
- Offer intuitive controls for map filters, pollutant selections, and timeline navigation

3.6 Design Plan

So 3.6 Design Plan

Our design plan is structured around three primary modules:

1. Frontend (User Interface and Visualization)
2. Backend (Data Management and API Integration)
3. AI & Geospatial Analytics Engine (Core Intelligence Module)

The following subsections and diagrams outline the structure and functionality of each component.

3.6.1 Frontend

UI Diagram

(Here you can insert a simple wireframe or block diagram showing your Streamlit dashboard — e.g., sidebar filters, map area, charts.)

The frontend of EnviroScan is developed using Streamlit, designed for simplicity, interactivity, and cross-platform accessibility.

It provides users (government agencies, researchers, and the public) with clear, visual, and analytical insights into pollution patterns and sources.

Overview of the Dashboard Interface:

- Home Page:
- Displays an overview of air quality index (AQI) trends and quick links to regional pollution reports.
- Geospatial Map View:
- Interactive map showing pollution concentration and detected sources (industrial, vehicular, etc.) using color-coded layers.
- Users can zoom, filter by pollutant type, or view historical data overlays.
- Source Analysis Panel:
- Displays real-time predictions of pollution sources, accompanied by model confidence levels and comparison charts.

- Analytics and Reports Page:
- Summarizes pollutant variations, correlations with weather factors, and regional comparisons over selected time frames.
- Admin Controls (restricted access):
- Allows authorized personnel to upload datasets, trigger model retraining, and manage API configurations.

UI Goals:

- Maintain a clean, minimal, and informative layout suitable for both technical and non-technical users.
- Ensure full responsiveness on desktops, tablets, and mobile devices.
- Offer intuitive controls for map filters, pollutant selections, and timeline navigation.

3.6.2 Backend

The backend is implemented using Flask, which handles:

- API integration for fetching live pollution and weather data.
- Database communication (SQLite for prototype, PostgreSQL for production).
- Model communication, sending data inputs to the AI engine and returning predictions to the frontend.

Backend Functions:

- Schedule API data fetches at fixed intervals.
- Clean, validate, and store environmental readings.
- Provide RESTful endpoints for the frontend dashboard to request updated results.
- Manage user sessions and role-based access (e.g., admin vs. public viewer).

Key Backend Diagram (conceptual):

[External APIs] --> [Flask Server] --> [Database] --> [AI Engine] --> [Visualization Layer]

3.6.3 AI & Geospatial Analytics Engine

This is the core intelligence of the EnviroScan system.

It combines machine learning and spatial data modeling to classify and visualize pollution sources.

Core Steps:

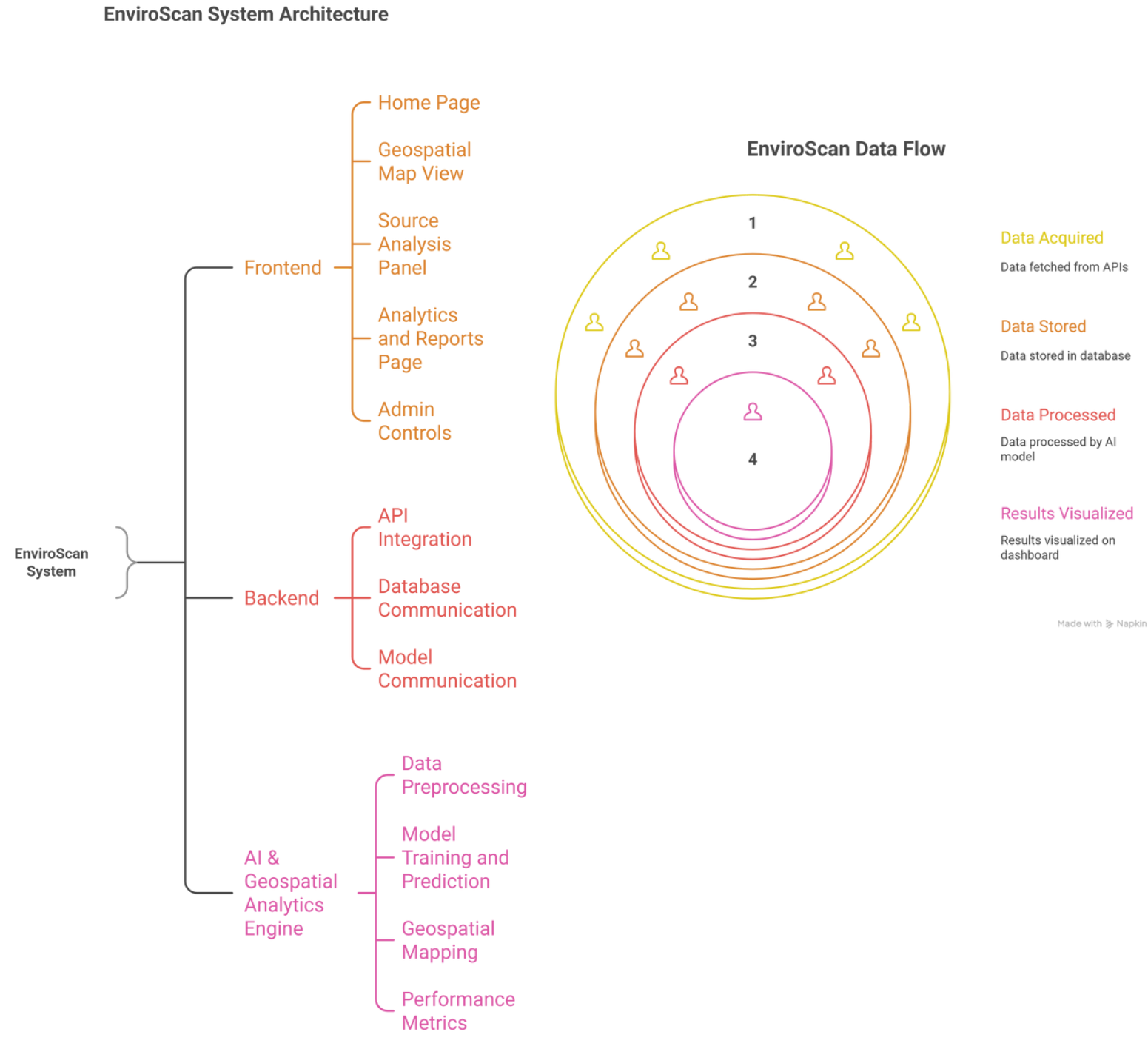
1. Data Preprocessing:
2. Normalize pollution and meteorological data, remove missing values, and encode categorical attributes.
3. Model Training and Prediction:
4. Train models (Random Forest, XGBoost) using labeled datasets to predict pollution source categories.
5. Geospatial Mapping:
6. Link predictions with geospatial coordinates using GeoPandas and render results via Folium heatmaps.
7. Performance Metrics:
8. Evaluate model accuracy, precision, recall, and F1-score to ensure reliable source identification.

Output Example:

- Pollution map showing regions in red (industrial), yellow (vehicular), and green (natural sources).
- Bar charts showing pollutant contribution per source type.

3.6.4 Integration and Data Flow

- The system follows a modular integration pipeline:
- Data acquired from APIs →
- Stored in database →
- Processed by AI model →
- Results visualized on Streamlit dashboard.
- Each component communicates via internal Flask APIs, ensuring modularity and scalability



4. Testing

The testing strategy for EnviroScan follows the principle of "test early and test often."
Since the project involves multiple modules — data collection, AI model prediction, backend API, and visualization — testing will be performed at every stage to ensure accuracy, reliability, and smooth integration.
We will use a combination of Unit Testing, Integration Testing, System Testing, Regression Testing, Acceptance Testing, and Security Testing to verify the correctness of each module and the overall system.
By performing continuous testing during each development phase, we aim to identify and correct defects early, ensuring that the system meets functional, performance, and usability requirements.
Unique Challenges
Some key testing challenges for this project include:

- AI Model Validation: Ensuring that pollution source predictions are accurate and scientifically valid.
- Geospatial Data Integration: Testing the compatibility and accuracy of geospatial data mapping with predicted results.
- Virtual Testing Environment: Since all components are software-based (no hardware sensors), the entire testing process will be performed virtually using simulated or real-time API data.
- Module Coordination: Verifying smooth communication between the backend API, AI engine, and frontend visualization interface

.

4.1 Unit Testing

The smallest testable units of the EnviroScan system will be data preprocessing functions, machine learning model components, and API utility modules.
Each unit will be tested with multiple input scenarios to confirm that outputs match expected results.
Unit Testing Focus Areas:

- Data Cleaning Functions: Ensure missing or invalid data are handled correctly.
- Feature Extraction Methods: Verify that environmental and weather parameters are accurately combined.
- Model Prediction Functions: Validate AI model outputs with known datasets.
- API Handlers: Confirm data is fetched and parsed correctly from OpenAQ and OpenWeatherMap APIs.

Tools and Frameworks Used:

- PyTest / Unittest → For testing data processing and model functions.
- Mock Testing (unittest.mock) → To simulate API responses and ensure backend stability.
- Coverage.py → To track how much of the code is tested and ensure completeness.

Each function will be tested with both valid and invalid datasets to confirm system robustness against data inconsistencies.

.

4.2 Interface Testing

Since EnviroScan is a web-based dashboard, interface testing is crucial to ensure that the user experience, responsiveness, and data presentation meet expectations.

The focus will be on functionality, usability, and aesthetics.

Functional Interface Testing:

- Verify data from APIs and database displays correctly on the dashboard.
- Ensure filters (pollutant type, time range, source category) work as expected.
- Validate all interactive map features (zoom, region click, tooltip details) operate smoothly.
- Confirm dynamic charts and reports update in real time as new data is fetched.

Aesthetic and Usability Testing:

- Assess visual consistency, map clarity, and font readability.
- Evaluate user workflow — easy navigation between analytics pages and maps.
- Gather periodic user feedback from trial testers (students and environmental users) to refine UI experience.

Testing Tools and Methods:

- Manual Testing: During each new dashboard feature implementation, developers perform step-by-step UI checks.
- Automated Testing:
  - Selenium – to automate UI tests for buttons, navigation, and form elements.
  - Streamlit Testing Framework (pytest-streamlit) – to validate component rendering.
- User Feedback Testing: Early trial versions will be shared for real-world usability evaluation.

4.3 Integration Testing

The EnviroScan project has three critical integration paths that must function seamlessly for accurate data collection, model execution, and visualization.

These integrations ensure smooth communication between the database, backend API, machine learning engine, and frontend dashboard.

4.3.1 Database to Server Communication

This is one of the most essential integration steps in EnviroScan.

For reliable operation, the Flask backend must successfully communicate with the database (SQLite during prototype, PostgreSQL for production).

Testing Objectives:

- Verify that the server establishes a stable database connection.
- Ensure correct creation and retrieval of pollution and weather data records.
- Validate database schema and ensure consistency in data fields (timestamp, location, pollutant type, source label).

Testing Method:

- Use Postman or Flask routes to manually insert and retrieve dummy pollution data.
- Confirm that data is properly stored and retrievable via SQL queries.
- Test with both valid and invalid data entries to check error handling.

4.3.2 Web Application to Database/Server Communication (Frontend to Backend)

The Streamlit frontend communicates with the Flask backend via RESTful APIs to display pollution data, analytics, and predictions.

This integration ensures that all user actions — such as filtering pollutants, refreshing maps, or viewing source analysis — retrieve accurate data from the backend.

Testing Objectives:

- Verify all API endpoints are functional and return correct JSON responses.
- Ensure visual components (maps, charts, and tables) display updated results dynamically.
- Confirm backend processing time is acceptable and UI updates are responsive.

Testing Method:

- Perform manual and automated API request tests using Postman and pytest-flask.
- Validate data integrity from database to UI by comparing API output and displayed map data.
- Check edge cases such as invalid coordinates or missing data responses.

4.3.3 AI Model Integration with Backend

The AI Pollution Source Classifier is a core component that must integrate seamlessly with the backend.

The backend provides processed environmental data to the model, receives predictions, and stores results in the database.

Testing Objectives:

- Verify correct data transfer between the backend and AI engine.
- Confirm prediction results are returned in the expected format (source type, confidence score).
- Ensure model outputs are stored correctly in the database and visualized on the map.

Testing Method:

- Use mock datasets to simulate backend requests to the model.
- Log model input-output pairs to validate the integrity of predictions.
- Compare predicted results with test datasets to check accuracy and data flow consistency.

4.4 System Testing

System testing will be performed after all unit, interface, and integrat
ion testing is completed successfully.

The goal is to verify that the complete EnviroScan system works as intended and meets all user and functional requirements.

Testing Approach:

- Use a combination of real-time API data and dummy datasets to simulate diverse conditions.
- Verify data collection, processing, prediction, and visualization workflows end-to-end.

Key Scenarios to Test:

- Fetching live pollution data from APIs and storing it in the database.
- Running the AI model to identify likely pollution sources for different regions.
- Displaying source predictions accurately on the interactive map.
- Generating regional and temporal pollution trend reports.
- Testing admin operations like dataset uploads, model retraining, or manual data refresh.

Expected Outcome:

Each system operation should execute correctly and consistently, with accurate pollutant visualization and reliable source predictions displayed on the dashboard.

4.5 Regression Testing

Regression testing ensures that any new update, feature, or bug fix does not affect previously stable functionalities of the system.

Given that EnviroScan relies heavily on multiple interconnected modules, regression testing is essential to maintain long-term system integrity.

Testing Strategy:

- Retest all previously passed unit, integration, and interface tests after each new code update.
- Re-run core workflows like API fetching, data storage, model execution, and dashboard display after modifications.
- Automate regression tests using PyTest scripts to save time and maintain consistency.

Critical Regression Areas:

- API response and data parsing reliability.
- Model prediction consistency after retraining or code changes.
- Database schema compatibility with existing data.
- Frontend–backend synchronization during map and chart updates.

Goal:

To ensure that all previous functionalities remain intact and unaffected by future updates, maintaining a stable, accurate, and responsive EnviroScan system.

4.6 Acceptance Testing

Acceptance testing will be derived from the system's use-cases to verify that all functional design requirements have been met. Each use-case will be marked "Passed Acceptance Testing" when the expected behavior is demonstrated with real or simulated data. Key stakeholders (environmental agency reps, researchers, and a faculty reviewer) will be invited to perform acceptance tests at major milestones and provide feedback for adjustments.

Example Use-Cases and Acceptance Criteria

Environmental Agency (Admin)

- Mass upload sensor / station metadata via CSV.
- Acceptance criterion: CSV accepted, records created, and visible in admin panel.
- Configure alert thresholds and notification channels.
- Acceptance criterion: Alerts trigger when thresholds breached and notifications are delivered.
- Trigger model retraining with new labeled data.
- Acceptance criterion: Retraining job runs and new model version appears in admin log.

Researcher / Analyst

- Query and export filtered historical dataset (by pollutant, date-range, geofence).
- Acceptance criterion: Exported CSV matches filters and column schema.
- Request source-prediction for a selected region/time window.
- Acceptance criterion: Dashboard shows predicted source categories with confidence scores and downloadable report.

Public / NGO User

- View live pollution heatmap for a city and receive alerts.
- Acceptance criterion: Map loads, hotspots match test dataset, alert received when simulated threshold exceeded.

System Admin / Developer

- Confirm API health and data ingestion pipeline status.
- Acceptance criterion: Health endpoints respond; ingestion logs show successful fetches.

Procedure

- Acceptance tests run at Milestones 3 and 5 (after Model Training & after Dashboard Integration).
- Stakeholders perform scripted scenarios; testers log pass/fail with evidence (screenshots, logs).
- All failed items are triaged, prioritized, fixed, and re-tested until acceptance.

4.7 Security Testing

Security testing ensures EnviroScan's web interface, APIs, and data storage are protected from misuse, data leakage, and attacks. Tests will be performed periodically during development and before major releases.

4.7.1 Injection & Cross-Site Scripting (XSS)

Risk: Malicious payloads submitted via inputs or query parameters that expose or corrupt data.

Mitigation: Input validation/whitelisting, use of parameterized queries (ORM or prepared statements), and sanitization on output.

Tests:

- Manual and automated attempts to inject SQL-like payloads into forms and API inputs.
- XSS payload tests in map popups and user-visible text fields.
- Tools: sqlmap (for controlled testing), Burp Suite / OWASP ZAP.

4.7.2 Brute-Force & Authentication Attacks

Risk: Unauthorized account takeover via repeated login attempts or token brute-force.

Mitigation: Rate-limiting, account lockout after configurable failed attempts, enforce strong passwords / MFA for admin users, secure session cookies.

Tests:

- Repeated login attempts to confirm lockout behavior.
- Session fixation and session expiration tests.

4.7.3 Role Management & URL/Endpoint Manipulation

Risk: Users accessing admin pages or restricted endpoints by editing URLs or API parameters.

Mitigation: Server-side authorization checks for every endpoint; role-based access control (RBAC).

Tests:

- Login as low-privilege user and attempt to access admin endpoints; assert 403/401 responses.
- Tamper with request parameters to validate server rejects unauthorized actions.

4.7.4 API Key & Secret Exposure

Risk: Accidental leakage of API keys in code or public repositories.

Mitigation: Store keys in environment variables / secret manager; never commit secrets; rotate keys regularly.

Tests:

- Scan repo for secrets before release (git-secrets or truffleHog).
- Confirm deployment uses environment-based config.

4.7.5 Data Privacy & Compliance

Risk: Exposure of location-sensitive or personally identifying data.

Mitigation: Avoid storing personally identifiable information (PII); anonymize/stage datasets; follow API usage licenses and data terms.

Tests:

- Privacy audit of stored schemas and logs.
- Confirm redaction/anonymization on exports when required.

4.8 Results (Test Reporting & Traceability)

At the time of this document draft, full test suites are being prepared and initial unit/integration tests have been executed on sample datasets. Results will be recorded in a Test Report Log that includes:

- Test ID / Use Case ID
- Test Description
- Input Data / Environment (API live vs simulated)
- Steps Performed
- Expected Result
- Actual Result
- Status: Pass / Fail / Blocked
- Evidence: Screenshots, API responses, logs, or model metrics
- Assigned To / Fix ETA

Metrics & Acceptance Criteria to be Reported

- Unit test coverage: Target ≥ 80% for core processing modules.
- Model performance: Target baseline accuracy / F1 specific to the chosen dataset (report exact values after training).
- API uptime (during tests): Target ≥ 99% for critical endpoints.
- Frontend responsiveness: Map load and filter update within X seconds (define during performance testing).
- Security checks: Zero critical vulnerabilities in pre-release scan.

Traceability

Each requirement in Section 1.2 / SRS will be linked to one or more test cases to provide traceability (Requirement → Test Case → Result). The traceability matrix will be included in Appendix B of the final report.

5 Appendix

Appendix A-1: Team Contract

Team Name: EnviroScan Development Team

Project Title: EnviroScan – Smart Environmental Monitoring System

Name	Role	Responsibilities
Prince Kumar	Project Manager / Developer	Oversees the project schedule, assigns tasks, ensures communication among members, manages resources.
[Member 2 Name]	Software Analyst	Gathers and analyzes requirements, interacts with stakeholders, defines system scope.
[Member 3 Name]	Designer	Creates UI/UX design and system architecture diagrams.
[Member 4 Name]	Developer	Implements backend and frontend functionalities.
[Member 5 Name]	Tester	Conducts test cases, ensures software quality and performance.

Required Skill Sets for the Project

- Programming (Python, HTML, CSS, JavaScript)
- Database Management (MySQL)
- Data Analysis and Visualization
- Project Documentation and Presentation
- Team Collaboration and Communication

Team Rules and Expectations

1. Each member must attend team meetings and complete assigned work on time.
2. Communication should be clear and professional (via WhatsApp/Google Meet).
3. All members must contribute equally to documentation and development.
4. In case of conflict, the issue will be discussed openly within the team.
5. Deadlines must be respected to ensure smooth project progression.

5 Appendix A-2

Survey Responses

Purpose of the Survey

To collect user feedback and environmental data needs to guide the design of the EnviroScan system.

Target Audience

- Local residents
- Environmental enthusiasts
- Municipal officers / pollution control staff

Sample Survey Questions

1. Are you aware of local air and water pollution levels in your area?
2. How frequently would you like to receive pollution alerts?
3. Which environmental factors concern you the most? (Air Quality / Water Quality / Noise Pollution)
4. Do you prefer mobile or web access for viewing pollution data?
5. Would you like the system to suggest preventive measures?

Summary of Responses

Question	Majority Response
Awareness about pollution	85% respondents were somewhat aware
Preferred update frequency	70% preferred daily updates
Most concerning factor	60% chose Air Pollution
Platform preference	55% preferred Mobile access
Interest in recommendations	90% said Yes

Key Insights

- Users expect real-time and accurate pollution data.
- Mobile notifications are highly preferred.
- Preventive suggestions significantly increase system value.
- Survey confirms strong community interest in EnviroScan for daily use.