AEROTRACK-Real Time Air Quality Monitoring

PYTHON ESSENTIALS

[VITyarthi]

SLOT:B11+B12+B13

By

Ms.JIGYASA

Reg No. :25BCY10166

Submitted to:

Ms. Murugeswari K.

TABLE OF CONTENTS

S NO.	TOPICS
1	INTRODUCTION
2	PROBLEM STATEMENT
3	FUNCTIONAL REQUIREMENTS
4	NON FUNCTIONAL REQUIREMENTS
5	SYSTEM ARCHITECTURE
6	DESIGN DIAGRAMS
7	DESIGN DECISIONS AND RATIONALE
8	IMPLEMENTATION DETAILS
9	SCREENSHOTS/RESULTS
10	TESTING APPROACH
11	CHALLENGES FACED
12	LEARNING&KEY TAKEAWAYS
13	FUTURE ENHANCEMENTS
14	REFERENCES

INTRODUCTION

AeroTrack is a Python-based real-time air quality monitoring system designed to provide instant AQI (Air Quality Index) and pollutant-level information for any location. It uses the OpenWeatherMap API to fetch environmental data and displays it in a structured and easy-to-understand console format.

The aim is to help users become aware of the environmental conditions around them, enabling better health and lifestyle decisions.

PROBLEM STATEMENT

Air pollution is increasing rapidly, and many individuals do not have easy access to real-time air quality data. Existing solutions may be complex, paid, or require installation.

There is a need for a simple, accessible, and lightweight tool that provides:

Real-time AQI

Pollutant concentration levels

Health-based interpretation

Easy location-based monitoring

AeroTrack addresses this by offering a CLI-based program that generates up-to-date air quality reports with minimal effort.

FUNCTIONAL REQUIREMENTS

Functional requirements describe what the system must do.

1 Input Requirements

The system must accept a place name from the user.

2 API Interaction

The system must call the Geocoding API to convert place names to coordinates.

The system must call the Air Pollution API to fetch AQI and pollutant data.

3 Output Requirements

Must display:

AQI (1–5)

Detailed pollutant concentrations (CO, NO_2, O_3, etc.)

Latitude and longitude

AQI category interpretation (Good / Fair / Poor etc.)

4 Error Handling

Must detect invalid place names.

Must handle unavailable data or API issues.

Must show meaningful error messages.

NON FUNCTIONAL REQUIREMENTS

1 Performance

API responses must be processed within 2–4 seconds.

Lightweight execution: minimal memory usage.

2 Usability

Console output must be clean, readable, and formatted.

Users should not require technical knowledge.

3 Reliability

Should respond correctly even when pollutant data is missing.

Should validate API response formats.

4 Scalability

Can be extended for multiple cities, historical results, or dashboards.

5 Security

API keys must be stored securely (environment variables preferred).

6 Portability

Should run on any OS with Python installed.

SYSTEM ARCHITECTURE

This project follows a three-layer architecture:

1. **Presentation Layer**

Accepts user input (place name)

Displays formatted air quality report

2. **Business Logic Layer**

get_coordinates(place_name)

fetch_air_quality(place_name)

interpret_aqi(aqi)

3. **Data Access Layer**

Calls external APIs using requests

Parses JSON response

Data Flow:

User → Program → Geocoding API → Coordinates → AQI API → Report Output

DESIGN DIAGRAMS

1 Use Case Diagram

Actors:User

Use Cases:

- Enter place name
- View AQI report
- See pollutant details
- Receive interpretation
- Handle errors

User → [Enter Location] → System → [Fetch & Display AQI]

User → [Request Pollutant Details]

2 Workflow Diagram

Start
↓
Input Place Name
↓
Call Geocoding API
↓
Valid Coordinates? → No → Display Error → End
↓ Yes
Call Air Pollution API
↓
Receive AQI & Pollutants
↓
Interpret AQI
↓
Display Report
↓
End

3 Sequence Diagram

User → Program: Enter place name

Program → Geocoding API: Request coordinates

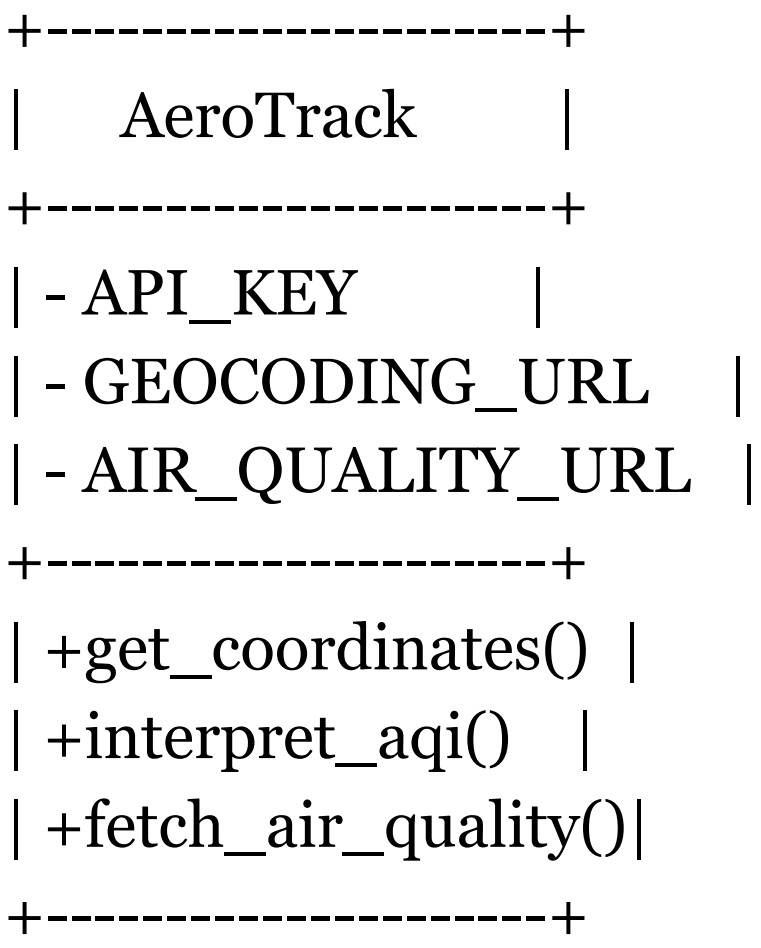
Geocoding API → Program: Return lat/lon

Program → AQI API: Request air quality data

AQI API → Program: Return AQI & components

Program → User: Display formatted report

4 Class / Component Diagram



5 ER Diagram

(Not applicable — no local storage/database used)

Design Decisions & Rationale

DECISIONS	RATIONALE
CLI-based program	Simple and lightweight; no GUI dependency
OpenWeatherMap API	Free, reliable, global coverage
requests library	Clean and easy HTTP handling
AQI Interpretation dictionary	Faster lookup and cleaner code
Separation of functions	Improves maintainability and readability.

Implementation Details

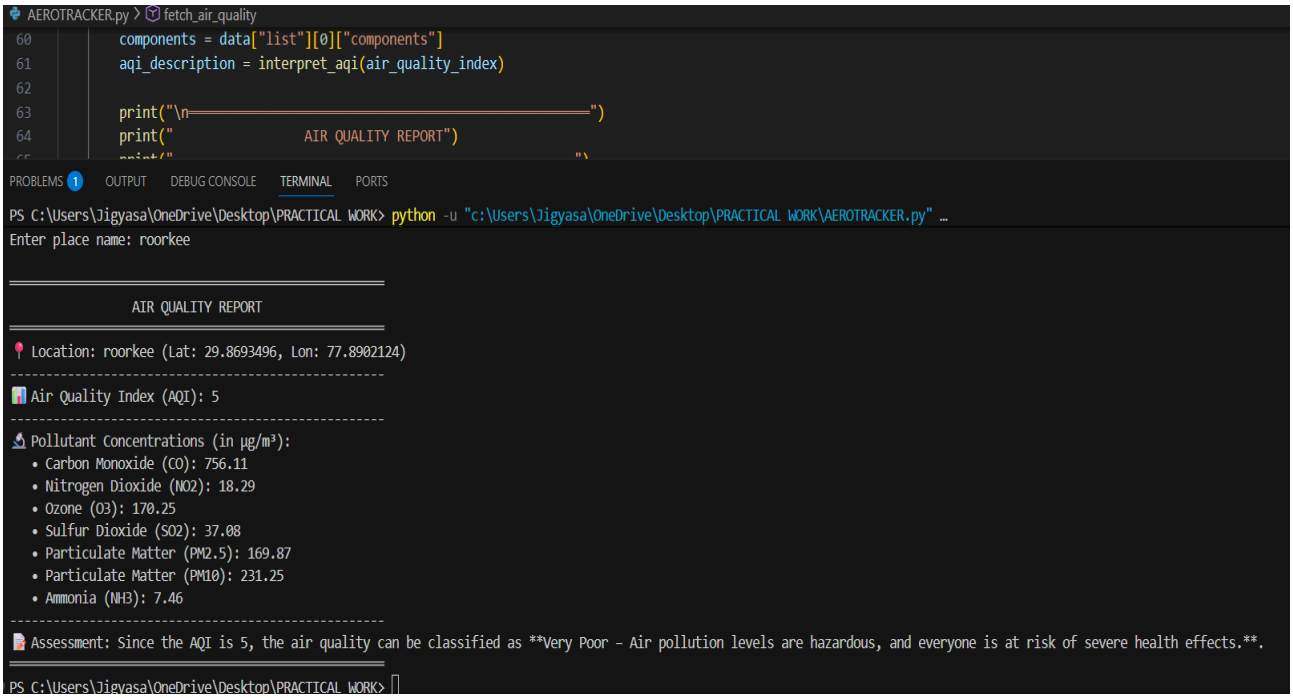
1.Implemented in Python 3.11

2.Three main functions:
get_coordinates() → Fetch latitude & longitude
fetch_air_quality() → Retrieve AQI + pollutants
interpret_aqi() → Convert AQI to readable text

3.Uses JSON parsing, conditional checks, and formatted printing.

4.Error handling ensures smooth user experience.

SCREENSHOT/OUTPUTS



TESTING APPROACH

Test Types:

1. **Functional Testing**
   - Input valid/invalid place names
   - Process API data correctly
2. **Negative Testing**
   - Network failure
   - Incorrect API key
   - Empty responses
3. **Boundary Testing**
   - Cities with multiple words (“New York”)
   - Small villages
   - Locations with missing AQI data
4. **Usability Testing**
   - Readability and clarity of console output

Challenges Faced

- Handling wrong or misspelled location names
- Dealing with inconsistent API responses
- Managing API key security
- Ensuring user-friendly formatting
- Internet dependency during testing

Learnings & Key Takeaways

- Understanding REST APIs & HTTP GET requests
- Working with real-time JSON data
- Error handling and input validation
- Importance of readable UI/UX even in CLI programs
- Basics of environmental pollutant categories
- Structured program design using functions

FUTURE ENHANCEMENTS

- Add historical AQI trend graphs
- Create a web or mobile UI
- Add database to store past results
- Provide alert notifications on poor air quality
- Map visualization using libraries like Folium
- Multi-city comparison analytics
- Support for offline caching

REFERENCES

In order to work on this project websites are referred by me during the various phases of development of the project.

1) www.youtube.com

2) www.python.com

OTHER THAN THE MENTIONED THING I HAVE ALSO SEEKED HELP AND INFORMATION FROM MY TEACHERS WHO MADE ME UNDERSTAND EACH AND EVERY DETAIL OF THE PROJECT.