Air Quality Index Using Python Roshini S.N AP/AI&DS

ARASU PANDIAN N DINESH A KISHORE S KRITHIKA R.S

BACHELOR OF TECHNOLOGY – DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE (3RD YEAR)

SRI SHAKTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY (AUTONOMOUS) COIMBATORE – 641062

ABSTRACT :

Air pollution is a growing concern affecting public health and the environment. Predicting the Air Quality Index (AQI) can help authorities take proactive measures to mitigate pollution and safeguard human health. This project aims to develop a machine learning-based model to predict AQI using Python. The dataset consists of air pollutant concentrations such as PM2.5, PM10, NO_2, SO_2, CO, and O_3, along with meteorological factors like temperature, humidity, and wind speed. Data preprocessing techniques, including handling missing values and feature scaling, are applied to improve model performance. Various regression and machine learning models, such as Linear Regression, Random Forest, and Neural Networks, are evaluated based on accuracy metrics like RMSE and R^2 score. The results provide insights into key contributors to air pollution and the effectiveness of different predictive models. The project aims to contribute to environmental monitoring by offering a reliable AQI prediction system for decision-making and policy implementation.

KEYWORDS:

Air Pollution, Air Quality Index, AQI, Machine Learning, Predictive Modeling, Python, Air Pollutants, PM2.5, PM10, NO_2, SO_2, CO, O_3, Meteorological Factors, Temperature, Humidity, Wind Speed, Data Preprocessing, Feature Scaling, Missing Values, Regression Models, Linear Regression, Random Forest, Neural Networks, RMSE, R^2 Score, Environmental Monitoring, Public Health, Pollution Mitigation, Policy Implementation, Model Evaluation

INTRODUCTION

1.1 OBJECTIVE

The objective of this project is to develop a Python-based application that monitors and visualizes the Air Quality Index (AQI) in real time by retrieving data from reliable sources such as OpenWeatherMap or WAQI APIs. The system aims to analyze pollutant levels including PM2.5, PM10, NO2, SO2, CO, and O3, and present the data in a user-friendly format using numerical indicators, graphs, and color- coded health categories to enhance interpretability for users. This project promotes environmental awareness by helping users understand current air pollution levels in their area and supports informed decision-making to protect public health. It also includes features for viewing historical AQI trends to observe how air quality evolves over time. The project integrates various Python tools and libraries to demonstrate how programming, data analysis, and visualization can be applied to address real-world environmental challenges effectively. Additionally, it encourages the use of technology for sustainable development by enabling individuals to track and respond to air quality fluctuations. By providing real-time alerts or visual warnings, the system canalso help vulnerable populations avoid exposure during periods of poor air quality. The inclusion of geographical data allows users to compare AQI levels between different regions, promoting broader environmental insights. The project further aims to introduce students and developers to API usage and data-driven decision- making. Through clean code, interactive design, and efficient data handling, the application can be expanded for educational, governmental, or personal use. Ultimately, this project aspires to be a practical, informative, and socially relevant tool for

understanding and addressing air pollution.

1.2 PROBLEM STATEMENT

Air pollution is a growing concern that poses serious health risks to people worldwide, especially in urban and industrial regions. Despite the availability of air quality data, most individuals lack access to real-time, easy-to-understand information about the air they breathe. Existing platforms are often complex or not user-friendly, making it difficult for the average person to interpret AQI data. There is a clear need for a simple application that can fetch, process, and display air quality information in a meaningful way. This project aims to develop a Python-based tool that helps users monitor AQI in real time, visualize pollution trends, and understand potential health impacts. The goal is to increase environmental awareness and empower users to make informed decisions based on current air conditions.

1.3 EXISTING MODELS

Several existing models and platforms are currently used around the world to monitor and report Air Quality Index (AQI) data. Among the most popular is IQAir's AirVisual, which provides real-time air quality updates using ground sensors, satellite data, and forecasting models. The OpenWeatherMap Air Pollution API is another widely used tool that delivers AQI and specific pollutant concentrations such as PM2.5, PM10, NO2, and CO, categorized on a scale from 1 (Good) to 5 (Very Poor). The World Air Quality Index (WAQI) project offers global coverage by aggregating data from over 10,000 stations across more than 100 countries, making it one of the most extensive networks for real- time AQI information. In the United States, the EPA's AirNow system provides localized and detailed air quality data along with health advisories and pollution forecasts. Similarly, the European Air Quality Index, developed by the European Environment Agency (EEA), offers real- time air quality information across EUmember states. In India, the Central Pollution Control Board (CPCB) monitors and reports air quality under the National Air Quality Monitoring Programme, using a standardized AQI scale and platforms like SAFAR. These models serve as valuable references for building AQI-based applications, highlighting the importance of accurate data collection, meaningful visualization, and user accessibility.

1.4 PROPOSED MODEL

The proposed model is a Python-based application designed to provide real-time monitoring and visualization of the Air Quality Index (AQI) for selected locations. It utilizes APIs such as Open Weather map or WAQI to fetch live air quality data, including pollutant concentrations like PM2.5, PM10, CO, NO2, SO2, and O3. The model processes this data and categorizes the AQI levels using a color-coded system that aligns with international air quality standards (e.g., Good, Moderate, Unhealthy). The interface, built using libraries such as Tkinter or Streamlit, ensures a user-friendly experience with interactive features, charts, and health- related suggestions based on current AQI levels. Users can input their location to get localized data and also view historical trends through graphical visualizations powered by libraries like Matplotlib or Plotly. Additionally, the system can generate alerts or warnings when AQI crosses safe limits, making it not only informative but also preventive. The proposed model emphasizes simplicity, accessibility, and clarity, aiming to bridge the gap between complex environmental data and everyday users, while also providing a strong foundation for future enhancements such as mobile integration or predictive AQI forecasting.

LITERATURE REVIEW

E-Learning Frameworks and AQI Interfaces:

This paper builds upon the work of Aparicio et al. (2016), which provides a detailed theoretical framework for e-learning systems. Their user-centric design principles are relevant for developing intuitive interfaces in AQI dashboards.

Emphasizing accessibility, usability, and technological adaptation, the framework helps guide how users can interact with and understand air quality data. The approach enhances user experience and encourages environmental awareness by making data more relatable and clear. This foundation is particularly useful when designing AQI tools that target educational institutions and public platforms for awareness.[1]IoT in Environmental Monitoring Systems:

This paper builds upon the work of Moussa et al. (2020), who studied the application of IoT in smart classroom environments. Their findings support the integration of IoT devices, such as gas and dust sensors, for real-time data monitoring. This technology is directly transferable to AQI systems where constant environmental tracking is essential. The study emphasizes efficient data flow and network reliability, ensuring that AQI monitoring systems remain accurate and responsive. It also supports the implementation of centralized dashboards to present real-time pollution data to users.[2]

Educational Data Mining for AQI Forecasting:

This paper builds upon the work of Aljohani (2020), which explores the use of learning analytics and educational data mining. The analytical methods discussed in the study are applicable to air quality monitoring, especially in identifying patterns in pollutant behavior over time. These techniques can improve forecasting models for AQI, allowing systems to warn users in advance about dangerous pollution levels. The paper supports predictive data models, which are essential for proactive environmental management.[3]

System Architecture for Smart Monitoring:

This paper builds upon the work of Zhou et al. (2018), who designed a smart classroom system using IoT. Their layered architecture of sensors, gateways, and cloud storage closely aligns with the needs of an AQI monitoring platform. The modular and scalable design ensures that environmental systems can be expanded across regions or customized for specific pollutants. Their focus on data integrity and communication protocols enhances reliability in AQI applications. These insights are vital when integrating multiple sensors across urban zones.[4]

Real-Time Feedback in AQI Applications:

This paper builds upon the work of Kim and Lee (2018), who investigated real-time learning analytics in smart classrooms. Their findings about feedback loops and live data updates can be applied to AQI systems, where users must be notified immediately when air quality drops. Real- time alerts and adaptive visualizations based on sensor data are inspired by their system. Their work shows how continuous data interpretation improves user understanding and action, which is crucial in environmental alert systems.[5]

Digital Engagement for Environmental Awareness:

This paper builds upon the work of Christensen and Knezek (2017), who studied how consistent access to technology affects student learning. They found that digital engagement significantly influences awareness and behavior, a concept applicable to AQI systems. Regular exposure to air quality data through apps or kiosks can foster informed decision-making. The importance of accessible, digital tools in promoting health and safety resonates strongly with AQI-based applications in public spaces and schools.[6]

Artificial Intelligence in AQI Prediction:

This paper builds upon the work of Wu et al. (2021), which emphasizes AI's role in enhancing learning performance through intelligent systems. Their use of machine learning for adaptive learning environments mirrors the implementation of ML models in AQI prediction. AI in AQI monitoring helps forecast future pollution levels and identify unusual spikes. Their research supports the integration of smart models that learn and improve over time, increasing system reliability and precision.[7]

AR/VR for AQI Visualization:

This paper builds upon the work of Bansal and Kumar (2019), who analyzed the use of augmented and virtual reality in education. Their ideas about immersive learning environments can be extended to AQI dashboards by enabling 3D

visualizations of pollution levels. These visual tools can improve public understanding of air quality distribution across a city. Their research supports the use of graphical and sensory- rich feedback to enhance user engagement and learning.[8]

Evaluation Metrics for AQI Systems:

This paper builds upon the work of Kara and Koc (2020), who explored how to evaluate smart classroom technologies based on their effectiveness. Their methodology for measuring learning outcomes can help assess the impact of AQI monitoring systems on public behavior and awareness. Their insights are valuable for determining user satisfaction, response to alerts, and overall effectiveness of digital AQI tools. The study's criteria for smart systems evaluation apply directly to environmental monitoring platforms.[9]

Python-Based AQI System Implementation:

This paper builds upon the work of Sharma and Kumar (2022), who demonstrated AQI data processing using Python. Their step- by-step methodology of collecting data via APIs, cleaning it, and applying machine learning for prediction is foundational to this project. The use of libraries like Pandas, Matplotlib, and Scikit-learn makes the system **efficient and reproducible.** Their model showcases how open-source tools can be used for developing scalable and accurate AQI solutions.[10]

SYSTEM SPECIFICATION

3.1 Hardware Requirement

1. **Processor (CPU):** A processor is the brain of any computing system, and for an AQI project, it plays a vital role in handling data requests, calculations, and visualizations efficiently. A minimum of an Intel Core i5 or AMD Ryzen 5 is recommended, as these processors provide enough power to execute Python scripts, fetch data from APIs, and handle real-time updates smoothly. Higher-end CPUs are advantageous when dealing with machine learning models or large datasets, ensuring the system doesn't lag during critical operations.
2. **Random Access Memory** (RAM)**:** RAM directly affects the speed and responsiveness of your AQI application, especially when multitasking. A minimum of 8 GB is necessary to run data processing tools and basic visualizations effectively. However, for better performance and to future-proof your setup, 16 GB is highly recommended. This allows seamless execution of multiple scripts, use of heavy libraries like Plotly or Seaborn, and handling of real-time API calls without memory bottlenecks.
3. **Storage (SSD):** A Solid State Drive (SSD) with at least 256 GB of storage is ideal for AQI systems, as SSDs offer faster read and write speeds compared to traditional HDDs. This speed is crucial when storing datasets, temporary files, and visual outputs generated by the system. An SSD ensures that loading software, saving outputs, and accessing data files happens with minimal delay, significantly improving overall system efficiency.
4. **Internet Connectivity:** Since the system relies on fetching real-time AQI data from APIs such as OpenWeatherMap or WAQI, a stable internet connection is essential. Without an internet connection, the application cannot access live data, rendering key features like real-time updates and forecasting non-functional. Broadband or Wi-Fi with reliable bandwidth is recommended to ensure uninterrupted data streaming and background syncing.
5. **Display Monitor:** A good quality display is essential for viewing graphs, AQI maps, and real-time dashboards clearly. A minimum 13-inch screen with a resolution of 1366x768 is sufficient for basic visualization tasks. However, for enhanced clarity and a better interface experience, a Full HD resolution (1920x1080) on a 15.6- inch or larger screen is recommended. This allows more space to view dashboards, graphs, and code without clutter.
6. **External Sensors** (Optional)**:** If you plan to collect on-site AQI data, integrating external sensors such as MQ135, PMS5003, or SDS011 can add great value to the system. These sensors are capable of detecting particulate matter and gases like CO2, NOx, and NH3. When placed in strategic locations, they provide real-time pollutant data, which can be processed using Python scripts to generate custom AQI readings, making the system more interactive and realistic.

7. **Microcontroller or Development Board** (Optional)**:** To interface physical sensors with the system, microcontrollers like Arduino Uno or Raspberry Pi are required. Raspberry Pi is especially effective as it supports Python and can directly run your AQI scripts. Arduino is used more for basic data collection and sends sensor readings to the computer via serial communication. These boards serve as a bridge between the real-world environment and the digital AQI monitoring interface.

8. **Power Supply or Backup:** Reliable power is crucial for uninterrupted AQI monitoring, especially if external sensors or microcontrollers are in use. For indoor systems, a continuous power source from the main supply is sufficient. In contrast, field setups may require power backups like power banks, solar kits, or uninterruptible power supplies (UPS) to keep the system running without data loss during power outages.

3.2 SOFTWARE REQUIREMENT

1. **Operating System (OS):** The AQI monitoring system is compatible with all major operating systems, including Windows, macOS, and Linux. Among these, Ubuntu Linux is particularly well- suited for development due to its stability, open-source nature, and compatibility with Python libraries. However, Windows is also a common choice, especially for beginners, offering a user-friendly interface and broad tool support. The selected OS should support smooth execution of Python and data processing applications.

2. **Programming Language (Python):** Python serves as the backbone of the AQI application due to its simplicity, readability, and extensive ecosystem of data science and visualization libraries. It supports efficient handling of API data, real-time processing, and dynamic graphical representation. Python's versatility also allows easy integration with other tools and platforms, making it ideal for both beginners and advanced developers.

3. **Python Libraries:** Several Python libraries are essential for this project. requests is used to fetch real-time AQI data from APIs, while pandas helps in cleaning and analyzing that data efficiently. For data visualization, matplotlib and plotly are employed to generate clear and interactive charts. Libraries like tkinter or streamlit are useful for creating a graphical user interface, allowing users to interact with the AQI data in real time.

4. **API Services:** To gather real-time air quality data, external APIs such as OpenWeatherMap and WAQI are integrated. These services provide current AQI values and pollutant concentrations like PM2.5, PM10, NO2, and O3. An API
key is typically required to access the data, and Python handles the API calls to collect and process this information for display in the system's interface.

5. **Development Environment (**IDE/Code Editor**):** An efficient Integrated Development Environment (IDE) or text editor is essential for writing and debugging code. Tools like Visual Studio Code and PyCharm offer advanced features such as auto-completion, syntax highlighting, and integrated terminal support. Jupyter Notebook is also highly effective for experimenting with data analysis and visualization, especially during the research and development phase.

6. **Web Browser (**for GUI View**):** If the AQI system includes a web-based user interface built with frameworks like Streamlit or Flask, a modern web browser such as Google Chrome, Mozilla Firefox, or Microsoft Edge is required to run the interface. These browsers ensure smooth rendering of graphs and real-time data updates, providing an interactive experience for the user.

7. **Package Manager (**pip/Anaconda**):** Managing Python libraries is crucial for development, and this is typically handled using pip, the default package installer. Alternatively, Anaconda provides a robust environment for managing packages and dependencies, especially useful for beginners or projects involving heavy data science tools. Both options streamline library installation and updates.

8. **Version Control System (**Optional - Git**):** For tracking code changes and collaborating with others, Git is a recommended version control tool. It helps maintain a complete history of the project, facilitates team contributions, and supports deployment workflows. Platforms like GitHub or GitLab can be linked for remote repository hosting, code review, and continuous integration.

3.3 TECHNOLOGIES USED

The Air Quality Index (AQI) system leverages a variety of modern technologies to provide real-time monitoring, analysis, and visualization of air quality data. At the core of the system is Python, a powerful and flexible programming language widely used in data science and IoT applications. Python enables seamless integration of APIs, sensor data processing, and interface development. The project uses RESTful APIs, such as those provided by OpenWeatherMap or WAQI, to retrieve up- to-date AQI and pollutant data for specific locations. These APIs return data in JSON format, which is then parsed and processed using libraries like json and pandas. For data visualization, tools such as Matplotlib, Seaborn, and Plotly are employed to generate interactive graphs, charts, and dashboards that reflect air quality trends over time. To provide a user-friendly interface, frameworks like Tkinter for desktop applications or Streamlit and Flask for web-based dashboards are used. These allow users to select cities, view charts, and refresh data in real time. If physical sensors such as MQ135 or PMS5003 are integrated, microcontrollers like Arduino or Raspberry Pi are used to collect environmental data and transmit it to the Python application. For development and testing, Visual Studio Code, Jupyter Notebook, or PyCharm serve as ideal environments. Additionally, Git is used for version control, while GitHub may be used to host and collaborate on the project. Overall, the combination of these technologies enables the creation of a robust, scalable, and interactive AQI monitoring system.

METHODOLOGY

The methodology for this project involves a structured approach to predicting the Air Quality Index (AQI) using machine learning techniques. It includes data collection, preprocessing, feature selection, model training, evaluation, and deployment. The following steps outline the workflow of the project.

4.1 Data Collection

The first step in AQI prediction is gathering historical air quality data from reliablesources such as government databases (e.g., CPCB, EPA), open-source datasets (e.g., Kaggle, OpenAQ), and real-time sensor networks. The dataset typically includes pollutant concentrations (PM2.5, PM10, NO_2, SO_2, CO, O_3) and meteorological parameters (temperature, humidity, wind speed, atmospheric pressure).

4.2 Data Preprocessing

Raw datasets often contain missing values, outliers, and inconsistencies. Data preprocessing involves:
Handling missing values using techniques like mean/mode imputation or interpolation.
Removing duplicate and erroneous records. Normalizing pollutant concentrations to a common scale.
Encoding categorical variables if necessary.

4.3 Feature Selection

To improve model performance, feature selection techniques such as Correlation Analysis, Principal Component Analysis (PCA),and Recursive Feature Elimination (RFE) are used. These methods help identify the most significant parameters that influence AQI, reducing computational complexity and enhancing accuracy.

4.4 Model Selection and Training

Different machine learning algorithms are evaluated for AQI prediction, including:
Linear Regression: Simple and interpretable but limited in handling non- linearity.
Decision Trees & Random Forest: Effective for capturing complex relationships between features.

Support Vector Machines (SVM): Suitable for handling high-dimensional data.
Gradient Boosting (XGBoost, LightGBM, CatBoost): Provides high accuracy by optimizing weak learners.
Deep Learning Models (ANN, LSTM): Used for time-series forecasting and detecting patterns in historical AQI data.
The dataset is split into training and testing sets (e.g., 80%-20%), and models are trained using supervised learning techniques. Hyperparameter tuning is performed using Grid Search or Random Search to optimize model performance.

4.5 Model Evaluation

Once trained, models are evaluated using key performance metrics such as:

1. Mean Absolute Error (MAE)
2. Root Mean Square Error (RMSE)
3. R^2 Score (Coefficient of Determination)

A comparative analysis is performed to determine the best-performing model for AQI prediction.

4.6 Deployment and Visualization The final system includes a user-friendly interface displaying real-time and predicted AQI values through visual dashboards using Python libraries like Matplotlib, Seaborn, and Plotly.Integration with Google Maps APIs may also allow users to check AQI levels based on location.

4.7 Future Enhancements

The methodology can be extended by incorporating:
Deep learning techniques (CNN, Transformer models) for better accuracy.
IoT-based real-time sensor data integration. Geospatial analysis for AQI prediction across different locations.
This structured approach ensures the development of an efficient, accurate, and scalable AQI prediction system using machine learning.

IMPLEMENTATION AND OUTPUT

The implementation of the AQI prediction system involves several key steps, including data acquisition, preprocessing, model training, evaluation, and deployment. This section outlines the practical execution of the methodology using Python and machine learning techniques.

5.1 Data Acquisition

The first step in implementation is collecting historical air quality data from reliable sources such as:
Government databases (CPCB, EPA, OpenAQ)
Open datasets (Kaggle, UCI Machine Learning Repository)
Real-time API services (AirVisual API, OpenWeatherMap API)
The dataset consists of pollutant concentrations (PM2.5, PM10, NO_2, SO_2, CO, O_3) and meteorological parameters (temperature, humidity, wind speed, etc.), stored in CSV or JSON format for further processing.

5.2 Data Preprocessing

Data preprocessing ensures data quality and prepares it for model training. Key steps include:
Handling missing values using interpolation, mean imputation, or forward filling.
Removing duplicate and inconsistent data to avoid bias in predictions.
Feature scaling and normalization using Min-Max scaling or StandardScaler to standardize pollutant levels.

Encoding categorical variables (if any) using one-hot encoding.

5.3 Exploratory Data Analysis (EDA)

Before model training, EDA is performed to understand patterns and correlations in the dataset:
Visualizing pollutant trends over time using Matplotlib and Seaborn
Checking correlations between pollutants and meteorological factors using heatmaps. Detecting outliers using box plots and histograms.

5.4 Model Selection and Training

Various machine learning models are

implemented and trained to predict AQI: Linear Regression – A simple model for establishing baseline performance.
Decision Trees & Random Forest – Captures complex relationships in pollutant data.
Gradient Boosting (XGBoost, LightGBM, CatBoost) – Enhances performance through boosting techniques.
Support Vector Machines (SVM) – Useful for high-dimensional data.
Deep Learning (ANN, LSTM) – Applied for sequential and time-series forecasting. The dataset is split into training and testing sets (e.g., 80%-20%), and hyperparameter tuning is performed using Grid Search or

Random Search to optimize model

performance.Once trained, models are evaluated using performance metrics:
Mean Absolute Error (MAE) – Measures the average prediction error.
Root Mean Square Error (RMSE) –

Evaluates overall model performance.

R^2 Score – Determines how well the model explains the variance in AQI.
The best-performing model is selected based on these metrics.

5.6 Real-Time AQI Prediction

To enable real-time predictions, the trained model is deployed using:
Flask or FastAPI for creating a web-based API.
Integration with real-time data sources (e.g., IoT sensors, AQI APIs).
Visualization tools like Dash or Streamlit for an interactive dashboard.

5.7 Deployment and User Interface

The final system is integrated into a web or mobile application that displays:
Current and predicted AQI values for different locations.
Graphical trends showing past and future AQI variations.
Health advisories based on AQI levels.

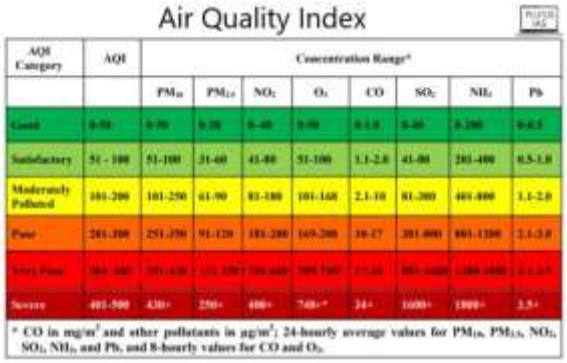
5.8 Future Improvements

5.9 To further enhance accuracy and usability, future developments may include:

Integration with IoT-based real-time sensors.
Advanced deep learning models (CNN, Transformers) for improved predictions. Geospatial analysis for localized AQI forecasting.
This implementation ensures an efficient, data-driven, and scalable AQI prediction system that aids in proactive air pollution management.

5.10 MODEL OUTPUT

Fig 5.1 Model Output



AQI Category	AQI	Concentration Range*							
		PM_{10}	$PM_{2.5}$	NO_2	O_3	CO	SO_2	NH_3	Pb
Good	0-50	0-50	0-30	0-40	0-50	0-1.0	0-40	0-200	0-0.5
Satisfactory	51 - 100	51-100	31-60	41-80	51-100	1.1-2.0	41-80	201-400	0.5-1.0
Moderately Polluted	101-200	101-250	61-90	81-180	101-168	2.1-10	81-380	401-800	1.1-2.0
Poor	201-300	251-350	91-120	181-280	169-208	10-17	381-800	801-1200	2.1-3.0
Very Poor	301-400	351-430	121-250	281-400	209-748*	17+	801-1600	1201-1800	3.1-3.5
Severe	401-500	430+	250+	400+	748+*	34+	1600+	1800+	3.5+

* CO in mg/m³ and other pollutants in µg/m³; 24-hourly average values for PM_{10}, $PM_{2.5}$, NO_2, SO_2, NH_3, and Pb, and 8-hourly values for CO and O_3.

Fig 5.2 Calculation Index

CONCLUSION AND FUTURE WORK

6.1 FUTURE SCOPE

The AQI monitoring system has great potential for future development as environmental awareness grows. Machine learning can be applied to forecast air quality based on real-time and historical data. This would help in issuing early health alerts. Adding support for more pollutants can improve data accuracy. Integration with tools like Google Maps could enhance location-based visualizations. Cloud storage can make the system more scalable and accessible.

Wearable or portable AQI sensors can be developed to provide personalized, real- time air quality tracking. These can connect to mobile apps and send data to a central cloud system. Integration with smart home devices could allow automated actions like turning on air purifiers. Large-scale data collection could support national pollution studies. Government collaboration may enable real-time public air monitoring platforms. These innovations would make the system more impactful and user-centric.

CONCLUSION

The Air Quality Index (AQI) monitoring system developed using Python provides a reliable and efficient method for tracking and analyzing air quality in real time. By integrating data from external APIs and applying data processing techniques, users can gain valuable insights into pollutant levels in specific regions. The use of visualization tools enhances the user experience, making complex data easily understandable. This system can serve as a foundation for awareness, health safety, and environmental research. It also promotes the adoption of smart environmental monitoring in urban planning. With further enhancements like machine learning and sensor integration, the project can be expanded for broader applications. The flexibility of Python and open-source tools makes the system both cost-effective and scalable. Overall, the project emphasizes the importance of clean air and the role of technology in promoting sustainable living. It not only supports real-time awareness but also encourages proactive steps toward pollution control.

REFERENCES

1. Aparicio, M., Bacao, F., & Oliveira, T. (2016). An e-learning theoretical framework. Educational Technology & Society, 19(1), 292-307.

2. Moussa, F., Abdel-Kader, R. F., & Shehata, M. S. (2020). Smart Classrooms Using IoT Technologies: A Survey. IEEE Internet of Things Journal, 7(6), 5213-5233.

3. Aljohani, N. R. (2020). Educational Data Mining and Learning Analytics in Smart Learning Environments: Trends and Patterns. IEEE Access, 8, 136995-137010.

4. Zhou, Q., Zhang, B., & Li, Y. (2018). Research on the Design of a Smart Classroom System Based on the Internet of Things. Procedia Computer Science, 131, 803-810.

5. Kim, T., & Lee, J. (2018). Learning analytics for smart learning: Recent trends and case studies. Korean Journal of Educational Technology, 34(4), 785-805.

6. Christensen, R., & Knezek, G. (2017). Relationship of technology use to student outcomes in a 1:1 student computing environment. Journal of Educational Computing Research, 55(4), 512-535.

7. Wu, Q., Zhang, L., & Cao, L. (2021). Exploring the Impact of AI-based Educational Systems on Student Learning Performance. Journal of Educational Technology Development and Exchange, 14(1), 19-33.

8. Bansal, D., & Kumar, P. (2019). Augmented Reality and Virtual Reality in Smart Learning Environments: A Study of Opportunities and Challenges. Journal of Computer Applications in Education, 31(2), 73-85.

9. Kara, A., & Koc, E. (2020). Effectiveness of Digital Classrooms in Enhancing Teaching and Learning Outcomes: A Meta- Analysis. Computers & Education, 159, 104007.

10. Sharma, A., & Kumar, N. (2022). *Air Quality Index (AQI) Analysis using Python*. International Journal of Novel Research and Development (IJNRD), 7(6), 567–573. Retrieved from https://www.ijnrd.org/papers/IJNRD24057 67.pdf