A

Project Report

On

AQI PREDICTOR

Submitted for the partial fulfillment of the requirement for the

Degree

of

Bachelor of Technology

in

Computer Science Engineering (Data Science)

Submitted by:

Ayush Sharma	**(2000681540012)**
Dhairya Rajput	**(2000681540014)**
Ankit	**(2000681540007)**
Sandeep Kumar	**(2000681540035)**

Under the Supervision of

Mr. Suraj Bhatnagar

(Professor Department of CSE (Data Science))

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

MEERUT INSTITUTE OF ENGINEERING & TECHNOLOGY, MEERUT

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

(Batch: 2020-2024)

DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature :

Name : Ayush Sharma
Roll No. : 2000681540012
Date :

Signature :
Name : Dhairya Rajput
Roll No. : 2000681540014
Date :

Signature :

Name : Ankit
Roll No. : 2000681540007
Date :

Signature :

Name : Sandeep Kumar
Roll No. : 2000681540035
Date :

Department of CSE – Data Science

Certificate

This is to certify that project/mini project report titled – *AQI Predictor* submitted by *Ayush Sharma (2000681540012), Dhairya Rajput (2000681540014), Ankit (2000681540007), Sandeep Kumar (2000681540035)* has been carried out under the guidance of Mr. Suraj Bhatnagar of Professor, Department of CSE - Data Science, Meerut Institute of Engineering and Technology, Meerut. This project report is approved for Project in 8th semester in CSE – DATA SCEINCE from Meerut Institute of Engineering and Technology, Meerut.

Mr. Suraj Bhatnagar
Project Supervisor

Mr. Rohit Aggarwal
HoD – CSE (Data Science)

Date :

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B.Tech. Final Year. We owe special debt of gratitude to our guide Prof. Suraj Bhatnagar, Department of CSE (Data Science), Meerut Institute of Engineering and Technology, Meerut for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Signature:

Name : Ayush Sharma
Roll No. : 2000681540012
Date :

Signature:

Name : Dhairya Rajput
Roll No. : 2000681540014
Date :

Signature:

Name : Ankit
Roll No. : 2000681540007
Date :

Signature:

Name : Sandeep Kumar
Roll No. : 2000681540035
Date :

ABSTRACT

This project focuses on the development of an Air Quality Index (AQI) predictor for Meerut city, addressing the escalating concerns about deteriorating air quality. With the objective of enhancing public health and environmental sustainability, the system utilizes historical air quality data to forecast AQI values, empowering stakeholders with actionable insights.

The development process integrates several advanced machine learning algorithms, including Linear Regression, Lasso Regression, Decision Tree Regression, Random Forest Regression, and XGBoost Regression. These algorithms are chosen for their ability to model complex relationships and provide accurate predictions based on historical data. The project begins with web scraping techniques to gather extensive historical air quality data, encompassing key pollutants such as PM2.5, PM10, NO2, SO2, CO, and O3, along with meteorological factors like temperature, humidity, and wind speed.

Collected data undergoes comprehensive preprocessing, which includes cleaning, normalization, and feature engineering to ensure consistency and accuracy. The preprocessed data is then used to train and validate the machine learning models. Various metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared are employed to evaluate the performance of the models. Cross-validation techniques further ensure the robustness and generalization capability of the models.

To enhance user engagement and facilitate decision-making, the project incorporates interactive visualizations using Tableau. These visualizations present AQI trends, patterns, and predictions in an accessible and comprehensible manner, aiding residents, local authorities, and businesses in understanding and responding to air quality issues effectively.

The experimental results demonstrate the system's effectiveness in accurately forecasting air quality levels and its computational efficiency. By providing timely and precise AQI predictions, the system enables stakeholders to implement proactive measures to mitigate air pollution, thereby fostering a healthier and more sustainable urban environment in Meerut city. Continuous monitoring, evaluation, and refinement of the system will be essential to maintain its relevance and effectiveness in addressing ongoing air quality challenges.

TABLE OF CONTENT

Page No.

LIST OF TABLES

LIST OF FIGURES

CHAPTER-1
INTRODUCTION

1.1. INTRODUCTION

The escalating concerns about deteriorating air quality in urban areas necessitate the development of robust systems to monitor and predict air quality levels. This project aims to develop an Air Quality Index (AQI) predictor for Meerut city, utilizing historical data and advanced machine learning algorithms. By providing accurate and timely predictions of AQI, the system seeks to empower residents, local authorities, and businesses to take proactive measures to mitigate air pollution and promote a healthier, more sustainable urban environment. Air pollution has become a critical issue worldwide, with urban areas experiencing significant challenges due to industrial activities, vehicular emissions, and other sources of pollutants. In response to this, various cities and organizations have implemented AQI monitoring systems to inform the public about air quality and health risks. However, traditional AQI monitoring often lacks predictive capabilities, which are essential for proactive intervention and planning.

In recent years, advancements in data science and machine learning have enabled the development of more sophisticated AQI prediction models. Researchers have explored various approaches, including statistical methods, neural networks, and hybrid models, to enhance the accuracy and reliability of air quality forecasts. This project builds on these advancements, focusing on the specific context of Meerut city, which faces significant air quality challenges due to rapid urbanization and industrialization. By leveraging state-of-the-art technologies and algorithms, the AQI predictor aims to provide a valuable tool for air quality management, supporting informed decision-making and proactive measures to improve air quality and public health in Meerut city.

1.2. SCOPE

The scope of the Air Quality Index (AQI) predictor project for Meerut city encompasses the comprehensive collection of historical air quality data, including key pollutants and meteorological factors. The project involves data preprocessing to ensure accuracy and consistency, and the implementation of various machine learning algorithms such as Linear Regression, Lasso Regression, Decision Tree Regression, Random Forest Regression, and XGBoost Regression to develop predictive models. The system integrates these models into a user-friendly web interface, supplemented by interactive Tableau dashboards, to provide real-time AQI forecasts and visualizations. The aim is to empower residents, local authorities, and

businesses with actionable insights to mitigate air pollution, supported by continuous evaluation and refinement to maintain system accuracy and reliability.

1.3. SOFTWARE DEVELOPMENT METHODOLOGY

The development of the Air Quality Index (AQI) predictor for Meerut city employs the Agile software development methodology. Agile is chosen for its flexibility, iterative progress, and ability to adapt to changing requirements through continuous stakeholder collaboration. The project begins with thorough requirement gathering and analysis to define clear objectives and deliverables. Data collection involves extensive historical air quality data acquisition through web scraping and APIs, followed by meticulous preprocessing to ensure accuracy. Multiple machine learning algorithms, including Linear Regression, Lasso Regression, Decision Tree Regression, Random Forest Regression, and XGBoost Regression, are implemented, trained, and validated to improve prediction accuracy. The system integrates these models into a user-friendly web application, enhanced by interactive Tableau dashboards for accessible AQI visualizations. Rigorous testing, including unit, integration, and user acceptance testing, ensures seamless functionality. The project is deployed for public access and undergoes continuous monitoring, regular updates, and refinement based on user feedback and technological advancements, maintaining the system's reliability and relevance in addressing Meerut's air quality challenges.

The following illustration is a representation of the different phases of the Agile Model.

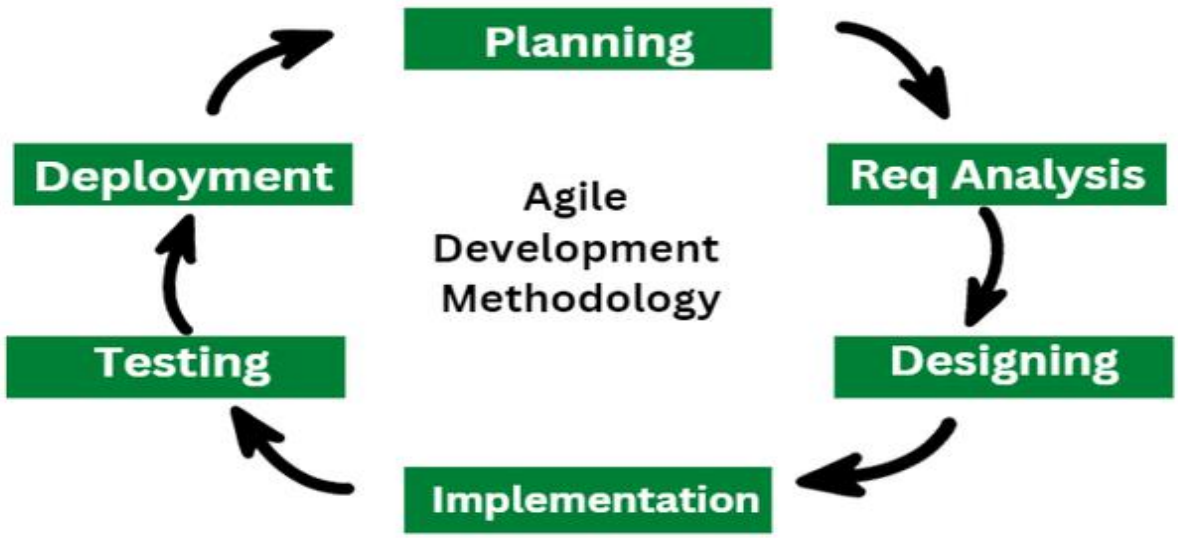

Fig.1.3.1: Agile Development Methodology

The sequential phases in Agile model are –

1.Planning:

The planning phase involves creating a detailed project plan with timelines, milestones, and resource allocation. Tasks are divided into sprints, and a prioritized backlog is developed. This ensures the project remains on schedule and meets stakeholders' expectations.

2.Requirement Analysis:

The needs of the project's stakeholders are carefully considered and recorded at this phase, including both functional and non-functional requirements. A thorough grasp of the project's goals, limitations, and scope is attained through workshops, questionnaires, and interviews, setting the stage for further stages.

3.Design:

After requirement analysis, the system's architectural foundation and software architecture are carefully created and documented during the design phase. The structure, elements, interfaces, and interactions of the system are described in both high-level and detailed designs. The scalability, maintainability, and general performance of the system are significantly impacted by decisions taken during this phase.

4.Implementation:

After the design specifications are agreed, the project moves onto the implementation phase, where talented developers code and develop the design blueprints to make them come to life. Software components are developed, tested, and merged in accordance with coding standards and best practices to build the entire system, guaranteeing smooth functionality and coherence.

5.Testing:

After the program is put into use, it is put through a thorough testing process that includes unit, integration, system, and acceptance testing to ensure that it is reliable and functional at all levels. Defects and discrepancies are found, recorded, and fixed through rigorous testing to guarantee the software fulfills stakeholder expectations and quality requirements.

6.Deployment:

The program moves on to the deployment phase, when it is installed and set up for usage in the production environment, following successful testing and approval. Meticulous preparation and implementation reduce downtime, while thorough training and documentation promote user acceptance and competence, guaranteeing a seamless shift to operational state.

Advantages of Agile Model

- Working through Pair programming produces well-written compact programs which have fewer errors as compared to programmers working alone.
- It reduces the total development time of the whole project.
- Agile development emphasizes face-to-face communication among team members, leading to better collaboration and understanding of project goals.
- Customer representatives get the idea of updated software products after each iteration. So, it is easy for him to change any requirement if needed.

Disadvantages of Agile Model

- The lack of formal documents creates confusion and important decisions taken during different phases can be misinterpreted at any time by different team members.
- It is not suitable for handling complex dependencies.
- The agile model depends highly on customer interactions so if the customer is not clear, then the development team can be driven in the wrong direction.

Methodology

Step 1: Literature review, and technology related paper study

Step 2: Identification of the research gaps

Step 3: Designing of the system

Step 4: Code the system

Step 5: Testing the system

Step 1: Literature Review and Technology Related Paper Study

The study begins with a thorough evaluation of the literature and an analysis of technology-related studies that concentrate on machine learning-based user segmentation in e-commerce. This stage comprises examining scholarly journals, industry reports, and research articles to get insights into the most recent approaches, software, and uses of user segmentation in the e-commerce space. A particular focus is on research that use evaluation measures like the silhouette score and clustering methods like DBSCAN and K-means to measure the quality of the clustering. The project intends to build a thorough understanding of the landscape by synthesizing current literature, providing a strong basis for further development phases.

Step 2: Identification of Research Gaps

The study project moves forward to identify research gaps and areas ripe for improvement in current approaches, drawing on insights obtained from the literature assessment. The project's goal is to identify particular difficulties, restrictions, or untapped potential in customer segmentation for e-commerce through careful investigation. A special emphasis is on examining the suitability and usefulness of assessment measures like the silhouette score in assessing clustering quality, as well as the efficacy of clustering algorithms like K-means and DBSCAN in effectively segmenting users. The project intends to provide new perspectives to the subject and creates a foundation for innovation by identifying research needs.

Step 3: Designing of the System

The project moves on to the design phase with a more nuanced understanding of current approaches and identified research gaps. Here, the research gap analysis and literature study provide insights that inform the system architecture and design. During the design phase, the system's scope, essential features and functionalities, workflow, and interactions between its many components are all outlined. A special focus is placed on integrating suitable evaluation measures, such as the silhouette score, with clustering methods, such as DBSCAN and K-means, in the system architecture. The goal is to create a strong and efficient user segmentation system for e-commerce that fills in the identified research gaps and accomplishes the project's goals.

Step 4: Coding the System

The project proceeds to the implementation phase when the system design is complete, during which the system is created and coded in accordance with the design standards. During this stage, code must be written to preprocess data, apply clustering techniques like K-means and DBSCAN, show the results of clustering, and compute evaluation metrics like the silhouette score. The coding process follows best practices and coding standards to guarantee the system's scalability, maintainability, and dependability.

Step 5: Testing the System

The project carries out thorough testing after system implementation to confirm the produced system's functionality, performance, and dependability. To find and fix any flaws or problems in the system, a variety of testing procedures are carried out, such as system, integration, and unit testing. In particular, the quality of the clustering results using assessment metrics like the silhouette score is evaluated, and the efficacy of clustering algorithms like K-means and DBSCAN in effectively segmenting users is evaluated. The project's goal is to guarantee the efficacy and quality of the created user segmentation system for e-commerce through extensive testing. It is end-to-end testing where the testing environment is similar to the production environment.

Step	Description
Literature Review	Conduct a thorough review of existing literature, research papers, and related projects
Research Gap Analysis	Identify gaps in existing knowledge and areas where the project can contribute or address challenges
System Design	Design the overall architecture of the system, including modules, components, and data flow
Implementation	Code the system according to the design specifications, incorporating algorithms and functionalities
Testing	Perform unit testing, integration testing, and system testing to ensure the correctness of the system
Deployment	Deploy the system in a production environment, making it available for real-world usage

Table.1.3.1: Methodology

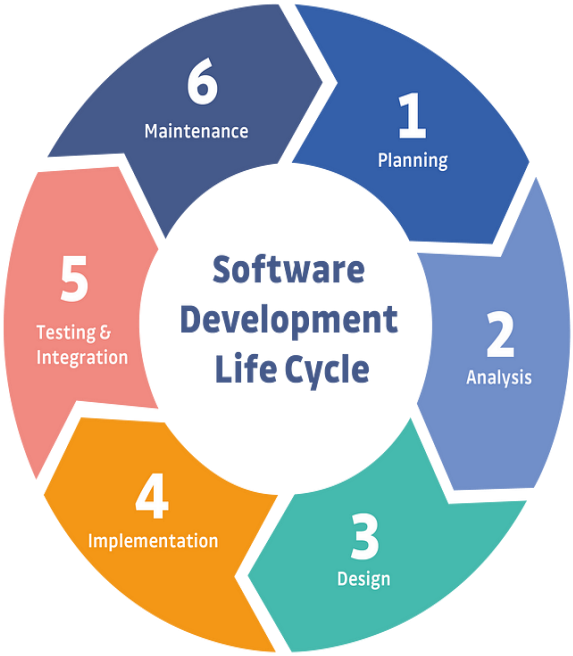

Fig 1.3.2: Software Development Life Cycle (SDLC) Phase

CHAPTER-2
EFFORT AND COST ESTIMATION

Effort and cost estimation are critical aspects of project planning, allowing for the allocation of resources and budgeting to ensure successful project execution. In the context of this project, effort and cost estimation will be conducted based on various factors including project scope, complexity, team size, and available resources.

The following steps outline the process for effort and cost estimation:

Project Scope Definition: The scope of the AQI predictor project for Meerut city encompasses defining the features and functionalities that will be implemented, along with the project's boundaries and goals. Key aspects include collecting historical air quality data, developing machine learning models for AQI prediction, creating a user-friendly web interface, and integrating interactive Tableau dashboards.

Estimation Techniques: Appropriate estimation techniques such as expert judgment, analogous estimation, parametric estimation, and three-point estimation are employed to estimate the amount of work required for each task. Factors like team experience, project complexity, and historical data are considered to ensure accurate work estimates.

Resource Allocation: Human resources, including team members and stakeholders, are assigned to specific tasks and activities based on their qualifications, availability, and experience. Ensuring the team composition is balanced and aligns with project requirements is critical. Analysts, data engineers, machine learning engineers, web developers, QA testers, and DevOps engineers are among the key personnel involved.

Cost Estimation: Once effort estimates are established, the cost of each task in terms of resources is calculated. This includes labor costs, equipment expenditures, software licenses, and overhead. The total estimated cost covers personnel, tools, software, training, and miscellaneous expenses.

Risk Assessment: Identifying and evaluating potential risks and uncertainties that could impact project effort and budget projections is crucial. Mitigation plans are developed for identified risks, and a contingency fund is included in the project budget to handle unexpected circumstances.

CHAPTER-3

SOFTWARE REQUIREMENTS SPECIFICATION

3.1. INTRODUCTION

This chapter outlines the Software Requirements Specification (SRS) for the Air Quality Index (AQI) predictor for Meerut city, a project aimed at addressing the growing concerns about air pollution and its adverse effects on public health and the environment. The AQI predictor system is designed to provide accurate and timely predictions of air quality based on historical data, empowering residents, local authorities, and businesses to make informed decisions and take proactive measures to improve air quality.

The SRS document is a comprehensive guide that details the requirements for the development and implementation of the AQI predictor system. It serves as a blueprint for the project, ensuring that all stakeholders have a clear understanding of the system's scope, objectives, and technical requirements. By providing a structured and detailed description of the system, the SRS facilitates effective communication among project team members and stakeholders, helping to ensure that the final product meets user needs and expectations.

3.2. INTENDED AUDIENCE AND READING SUGGESTIONS

The intended audience for this Software Requirements Specification (SRS) document includes various stakeholders who will be involved in or affected by the AQI predictor system for Meerut city. These stakeholders include project stakeholders such as local authorities, environmental agencies, and community representatives who need to understand the project’s scope, objectives, and deliverables. Developers, including software engineers, data scientists, and machine learning specialists, will use this document to gain a detailed understanding of the technical requirements, system architecture, and functional specifications necessary for the development and implementation of the system. QA testers will design and execute test plans based on the document to ensure the system meets all specified requirements. End-users, such as residents, business owners, and health professionals, will use the AQI predictor system to monitor air quality and make informed decisions, while researchers interested in air quality monitoring, environmental science, or machine learning applications will find insights into the methodology and implementation details.

For effective understanding, it is recommended to start with the introduction section to gain an overview of the project, its objectives, and the importance of accurate AQI predictions for Meerut city. Next, proceed to the general architecture section to understand the overall structure and components of the AQI predictor system, including details on data collection, preprocessing, machine learning models, web interface, and visualization tools. The functional requirements section provides a detailed description of the system's functionalities, such as data collection, preprocessing, model training, prediction output, visualization, user interface, and notification system. The non-functional requirements section outlines the quality attributes of the system, including performance, scalability, reliability, usability, and security.

Reviewing the feasibility study section helps assess the operational, technical, and economic feasibility of the project. The system requirements study section details the software and hardware requirements necessary for the system's implementation. Finally, the user requirement document section provides visual representations of the system's functionality through use-case diagrams and activity diagrams, and the system design section delves into the technical design, including data flow diagrams, sequence diagrams, and class diagrams. By following these reading suggestions, each stakeholder can efficiently navigate the SRS document and focus on the sections most relevant to their roles and interests, ensuring a clear understanding of the system's requirements and design.

3.3. GENERAL ARCHITECTURE OF SOFTWARE

The general architecture of the AQI predictor system for Meerut city is designed to provide a robust, scalable, and user-friendly platform for predicting air quality. The architecture encompasses various components that work together seamlessly to collect, process, analyse, and present air quality data. The following sections outline the main components of the system and their interactions.

Data Collection Layer

The data collection layer is responsible for gathering historical and real-time air quality data from various sources. This includes:

- **Web Scraping Module:** Extracts air quality data from government websites, environmental monitoring agencies, and other online sources. It ensures that the data is collected at regular intervals and is up-to-date.
- **API Integration:** Connects to APIs provided by environmental agencies to fetch real-time air quality data. This ensures the system has access to the latest information.

Data Preprocessing Layer

Once the data is collected, it needs to be cleaned and processed to ensure it is suitable for analysis. The data preprocessing layer includes:

- **Data Cleaning:** Handles missing values, outliers, and inconsistent data to ensure the quality of the dataset.
- **Data Transformation:** Converts raw data into a structured format, normalizing and scaling the data as required by the machine learning models.
- **Feature Engineering:** Extracts relevant features from the raw data, such as pollutant levels, weather conditions, and other environmental factors.

Machine Learning Layer

The machine learning layer is the core of the AQI predictor system, responsible for building and training models that can accurately predict air quality. This layer includes:

- **Model Selection:** Involves choosing the appropriate machine learning algorithms based on the nature of the data and the prediction goals. Algorithms used include Linear Regression, Lasso Regression, Decision Tree Regression, Random Forest Regression, and XGBoost Regression.
- **Model Training:** Trains the selected models on historical air quality data. The training process involves splitting the data into training and validation sets, tuning hyperparameters, and evaluating model performance.
- **Model Evaluation:** Assesses the accuracy and reliability of the trained models using metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R^2) score. The model with the highest accuracy and lowest error rates is selected for deployment.

Prediction and Visualization Layer

This layer provides the interface for users to interact with the AQI predictor system and visualize the results. It includes:

- **Prediction Module:** Uses the trained machine learning model to predict future air quality based on current and historical data.
- **Web Interface:** A user-friendly web application where users can input data, view predictions, and access historical air quality information. The interface includes features such as data visualization, interactive charts, and AQI alerts.
- **Visualization Tools**: Utilizes tools like Tableau to create interactive dashboards that display air quality trends, pollutant levels, and other relevant data insights.

Database Layer

The database layer manages the storage and retrieval of air quality data and model predictions. It includes:

- **Relational Database:** Stores structured data, including historical air quality records, model parameters, and user information.
- **Data Warehouse:** Aggregates large volumes of historical and real-time data, supporting advanced analytics and reporting.

Integration and Deployment Layer

This layer ensures that the AQI predictor system is deployed effectively and integrates seamlessly with other systems. It includes:

- **Continuous Integration/Continuous Deployment (CI/CD):** Automates the process of deploying new updates and features to the system, ensuring that it remains up-to-date and functional.
- **API Services:** Provides endpoints for other applications to access the prediction results and historical data, enabling integration with third-party systems.

Security and Monitoring Layer

To ensure the system's integrity and reliability, the security and monitoring layer includes:

- **Security Protocols:** Implements measures to protect data from unauthorized access, ensuring data privacy and compliance with regulations.
- **Monitoring Tools:** Continuously monitors system performance, detecting and resolving issues to maintain optimal functionality.

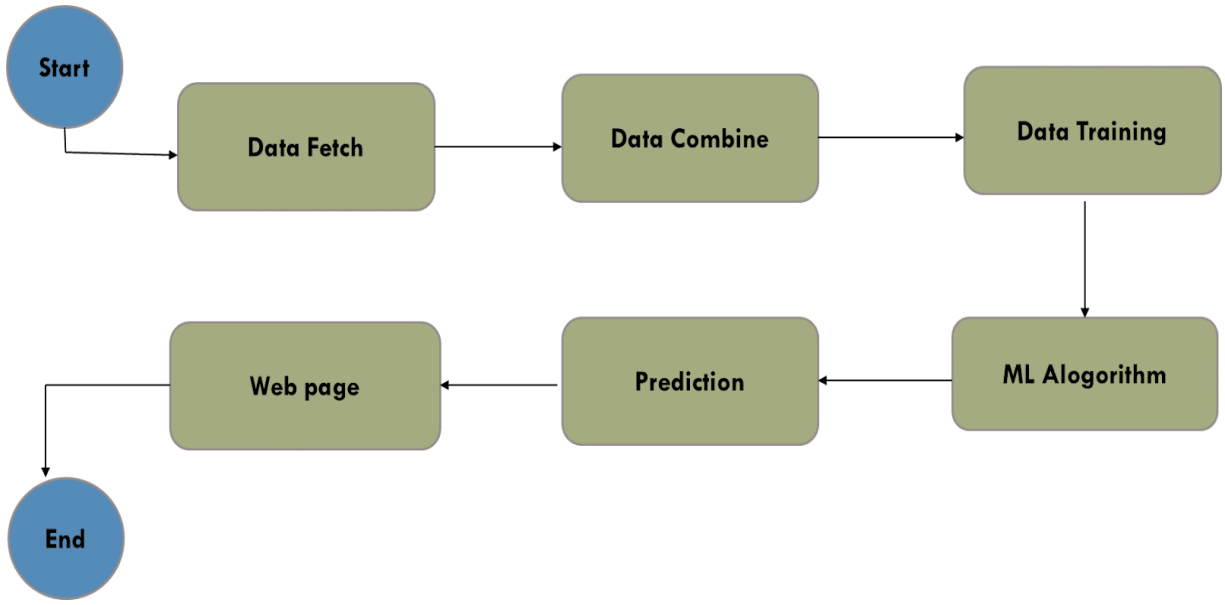

Fig.3.3.1 General Architecture of AQI Predictor

3.4. REQUIREMENT SPECIFICATION

3.4.1. FUNCTIONAL REQUIREMENTS

Functional requirements delineate the specific features, functionalities, and behaviours of the system. For the AQI Predictor project, functional requirements include:

- **Data Collection:** The system must automatically collect air quality data from various sources, including government websites, environmental monitoring agencies, and APIs. Ensuring data accuracy and integrity through validation processes is crucial for maintaining the reliability of the collected data.
- **Data Preprocessing:** The system must clean the collected data to handle missing values, outliers, and inconsistencies. This involves tasks such as removing incomplete records, imputing missing values, and using statistical techniques or machine learning algorithms to detect and eliminate outliers, which can distort the results of clustering.
- **Implementing Machine Learning Algorithms:** After preprocessing the data, machine learning algorithms are employed to analyze and predict air quality. The selection of algorithms, including Linear Regression, Lasso Regression, Decision Tree Regression, Random Forest Regression, and XGBoost Regression, depends on the data characteristics and prediction objectives. Each algorithm has its strengths, such as handling complex relationships in data or minimizing prediction errors.

- **Providing Interactive Visualization Tools:** Effective exploration and interpretation of predictions are facilitated by interactive visualization tools. The system should provide real-time visualizations through dashboards or graphical user interfaces, allowing users to interact with the data by zooming in on specific areas, filtering out unnecessary data points, and examining individual clusters in detail. Visualization methods like scatter plots, dendrograms, heatmaps, and parallel coordinates help depict multidimensional data effectively.
- **Calculating Evaluation Metrics:** Evaluation metrics are necessary to assess the accuracy and reliability of the machine learning models. Metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R^2) score help compare different algorithms and parameter combinations systematically. Visual presentations of these metrics aid in data-driven decision-making and facilitate stakeholder interpretation.
- **Generating Comprehensive Reports and Summaries:** The final stage involves generating detailed reports and summaries to communicate the most important findings and insights gained from the analysis. Reports should provide descriptive information, visuals, and interpretations of air quality trends, highlighting significant patterns and potential areas of concern. These reports guide stakeholders in making informed decisions regarding public health, regulatory measures, and individual actions to improve air quality.

3.4.2.NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements define the quality attributes and constraints of the system, ensuring that it meets user expectations and industry standards. Non-functional requirements for the AQI Predictor project include:

- **Performance**: It requires that the system be able to manage massive amounts of data effectively. It should be able to process user data quickly and respond to queries and analysis requests in real-time or almost in real-time. By ensuring that users encounter as little delay as possible while getting segmentation data or using visualization tools, this improves user happiness and productivity in general.

- **Scalability:** It is another essential non-functional requirement, requiring the system to expand with increasing user loads and data quantities in a seamless manner. The system should be able to scale vertically or horizontally to handle changes in the user base or data complexity without seeing a noticeable decline in performance. This guarantees that the

system will continue to function well and be reachable even in the face of high usage, fostering ongoing innovation and corporate expansion.

- **Reliability:** It is essential for guaranteeing continuous operation and preserving the integrity of the data. In order to survive malfunctions or interruptions without sacrificing functioning, the system must be strong and resilient. To reduce downtime and provide reliable access to segmentation services, strategies including fault tolerance, redundancy, and automated recovery methods are to be put in place. By placing a high priority on dependability, the system builds user and stakeholder confidence in its abilities and outcomes.

- **Security:** It is an essential prerequisite for protecting user information and upholding privacy. The system ought to follow industry standards for data security, putting strong safeguards in place to keep private data safe from hacking or illegal access. This entails strict access rules, encryption of data while it's in transit and at rest, and frequent security audits to find and fix any weaknesses. By putting security first, the system complies with industry standards and legal requirements while maintaining user confidentiality and confidence.

- **Usability:** For the AQI Predictor project, it is an essential non-functional need that highlights the significance of interface and user experience design. Users should be able to access, study, and analyze segmentation data with ease thanks to the system's intuitive user interface, which makes interaction and navigation effortless.

3.5. FEASIBILITY STUDY

3.5.1. OPERATIONAL FEASIBILITY

The operational feasibility of the AQI predictor system involves a comprehensive assessment of the system's compatibility with the existing operational processes in Meerut's environmental monitoring and public health sectors. This includes evaluating user acceptance of the prediction model among various stakeholders, such as local authorities, residents, and businesses. The necessary training for staff members to effectively use and maintain the system must be identified and implemented. Additionally, the readiness of different municipal departments to integrate the new technology into their operations must be assessed. Stakeholder feedback is crucial for

gaining support and ensuring successful deployment and adoption of the AQI predictor system. Effective communication channels should be established to address concerns and provide continuous updates on system performance and improvements.

3.5.2.TECHNICAL FEASIBILITY

Technical feasibility focuses on determining whether the proposed AQI predictor system can be implemented using the available technological resources and infrastructure in Meerut. This assessment includes ensuring the system meets stringent performance requirements for accurate and timely AQI predictions. Scalability is a key consideration, as the system must handle large volumes of air quality data collected from multiple sources. Compatibility with existing hardware and software systems within the city's IT infrastructure must be evaluated to facilitate seamless integration. Security and privacy concerns must be addressed to protect sensitive environmental and health data, ensuring compliance with legal regulations. A thorough technical feasibility analysis helps manage implementation risks and ensures the system's robust performance and reliability.

3.5.3.ECONOMIC FEASIBILITY

Economic feasibility involves evaluating the cost-effectiveness of developing and maintaining the AQI predictor system over its lifecycle. This includes estimating initial development costs, such as software development, hardware procurement, and data acquisition expenses. Ongoing operational costs, including maintenance, updates, and staff training, must also be considered. The project team should conduct a detailed cost-benefit analysis to determine the potential return on investment (ROI). Benefits such as improved public health outcomes, enhanced regulatory compliance, and more effective pollution control measures should be quantified. Long-term sustainability factors, such as potential cost savings from proactive air quality management and revenue generation through data-driven services, should also be considered. By thoroughly evaluating the project's economic feasibility, stakeholders can make informed decisions about resource allocation and investment, ensuring the efficient use of financial resources and maximizing the project's impact on air quality management in Meerut.

3.6. SYSTEM REQUIREMENTS STUDY

3.6.1. SOFTWARE REQUIREMENTS

The software requirements for the AQI predictor system encompass various tools, libraries, and platforms necessary to build, deploy, and maintain the system effectively. Key software requirements include:

- **Programming Languages:** Python is the primary programming language used for developing machine learning models and data processing scripts due to its extensive libraries and ease of use.
- **Machine Learning Libraries:** Libraries such as scikit-learn for implementing algorithms like Linear Regression, Lasso Regression, Decision Tree Regression, Random Forest Regression, and XGBoost are essential. Additional libraries such as TensorFlow or PyTorch may be used for advanced modeling.
- **Data Processing and Analysis Tools:** Pandas and NumPy are crucial for data manipulation and analysis, while Seaborn and Matplotlib are used for data visualization.
- **Web Scraping Tools:** Beautiful Soup and Scrapy are used for extracting data from web sources to ensure comprehensive data collection.
- **Database Management Systems:** SQL-based databases like PostgreSQL or NoSQL databases like MongoDB are necessary for storing and managing large volumes of air quality data.
- **Visualization Tools:** Tableau or similar visualization platforms are used to create interactive dashboards that display air quality trends and insights in an easily understandable format.
- **Web Development Frameworks:** Frameworks like Flask or Django are used to develop the web interface of the system, allowing users to interact with the prediction model and view results.
- **Version Control Systems:** Git is essential for version control, enabling collaborative development and management of code changes.
- **Integrated Development Environment (IDE):** IDEs such as PyCharm or Jupyter Notebook are used for writing and testing code.

3.6.2. HARDWARE REQUIREMENTS

The hardware requirements for the AQI predictor system are essential to support the computational needs of data processing, model training, and deployment. Key hardware requirements include:

- **Computing Power:** High-performance servers or cloud-based virtual machines (VMs) with multiple cores and high-speed processors (such as Intel Xeon or AMD EPYC) are required to handle large datasets and perform complex computations.
- **Memory (RAM):** Sufficient memory is crucial for handling large datasets and running machine learning algorithms efficiently. A minimum of 32GB of RAM is recommended for data processing and model training tasks.
- **Storage:** Large storage capacity is needed to store raw and processed air quality data, model parameters, and historical predictions. Solid State Drives (SSDs) with capacities of 1TB or more are recommended for faster data access and retrieval.
- **Graphics Processing Unit (GPU):** For advanced machine learning models, especially those involving deep learning, GPUs like NVIDIA Tesla or RTX series can significantly speed up training times and improve performance.
- **Networking Equipment:** Reliable and high-speed internet connectivity is necessary for real-time data collection from web sources and APIs. Networking equipment should support seamless data transfer and communication between different system components.
- **Backup Solutions:** Regular backups are essential to prevent data loss and ensure system reliability. Backup solutions should include external hard drives, cloud storage services, or network-attached storage (NAS) systems.
- **Redundancy and Failover Systems:** To maintain system availability and reliability, redundancy and failover systems should be in place. This includes redundant servers, power supplies, and network connections to ensure continuous operation in case of hardware failure.

3.7. USER REQUIREMENT DOCUMENT (URD)

The User Requirement Document (URD) outlines the requirements from the perspective of the end users of the AQI predictor system. It includes detailed descriptions of user interactions, system functionalities, and user goals to ensure the system meets their needs effectively.

3.7.1. USE-CASE DIAGRAM

A use-case diagram provides a visual representation of the various interactions users will have with the AQI predictor system. It helps identify the system's functionalities and the relationships between different actors and use cases. Below is an example description of key use cases:

Actors:

- **Resident:** A citizen of Meerut who wants to check the air quality index for personal health and safety.
- **Local Authority:** Government officials responsible for monitoring and regulating air quality.
- **Business Owner:** Business operators who need air quality data for operational decisions.
- **System Administrator:** The individual responsible for maintaining and managing the AQI predictor system.

Use Cases:

- **Check Current AQI:** Residents, local authorities, and business owners can view the current air quality index for specific locations in Meerut.
- **View Historical Data:** Users can access historical air quality data to analyze trends and patterns over time.
- **Receive Alerts:** Users can subscribe to notifications and receive alerts when the AQI reaches hazardous levels.
- **Generate Reports:** Local authorities and business owners can generate detailed reports based on historical and current AQI data for regulatory compliance and operational planning.
- **Manage System:** System administrators can update, maintain, and monitor the system to ensure it functions correctly.

3.7.2. ACTIVITY DIAGRAM

An activity diagram for the AQI predictor system outlines the sequential flow of activities for key user interactions, ensuring clarity in the process and user experience.

- **Check Current AQI:** The process begins with the user logging into the system using their credentials. Once logged in, the user selects the desired location for which they want to view the air quality index. The system then fetches the latest AQI data for the selected location from the database. After retrieving the data, the system displays the current AQI on the user interface. At this point, users have the option to interact further with the data, such as viewing detailed information on specific pollutants or switching to another location to check its AQI. Finally, once the user has finished their session, they log out of the system.
- **Generate Reports:** This activity starts similarly with the user logging into the system. Local authorities or business owners then select the parameters for their report, including

the date range, location, and type of report needed. The system proceeds to fetch the relevant historical AQI data based on these parameters. Once the data is retrieved, the system processes it and generates a detailed report that includes various charts and analysis. Users can then download or print the report for their records or further use. The process concludes with the user logging out of the system.

3.8. SYSTEM DESIGN

3.8.1. INTRODUCTION

The system design phase outlines the architecture and components necessary to build the AQI predictor system. This includes defining the data flow, sequences of interactions, and the structure of the system. A well-thought-out design ensures the system is scalable, maintainable, and meets the requirements outlined in the user requirement document.

3.8.2. DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) illustrates how data moves through the AQI predictor system, showing the inputs, processes, and outputs.

Level 0: Context Diagram

- **External Entities:** Residents, Local Authorities, Business Owners, Data Sources (e.g., weather stations, environmental monitoring agencies)
- **System:** AQI Predictor
- **Data Flows:** Requests for AQI data, delivery of AQI data, notifications, and alerts

Level 1: Detailed Data Flow

- **Processes:**
  - **User Interface Management:** Handles user login, location selection, and display of AQI data.
  - **Data Retrieval:** Fetches current and historical AQI data from the database.
  - **Data Processing:** Cleans and processes raw data for analysis and reporting.
  - **Notification System:** Manages user subscriptions and sends alerts.
  - **Report Generation:** Creates detailed reports based on user-selected parameters.
- **Data Stores:**
  - **User Data Store:** Stores user credentials, preferences, and subscription details.
  - **AQI Data Store:** Maintains current and historical AQI data.

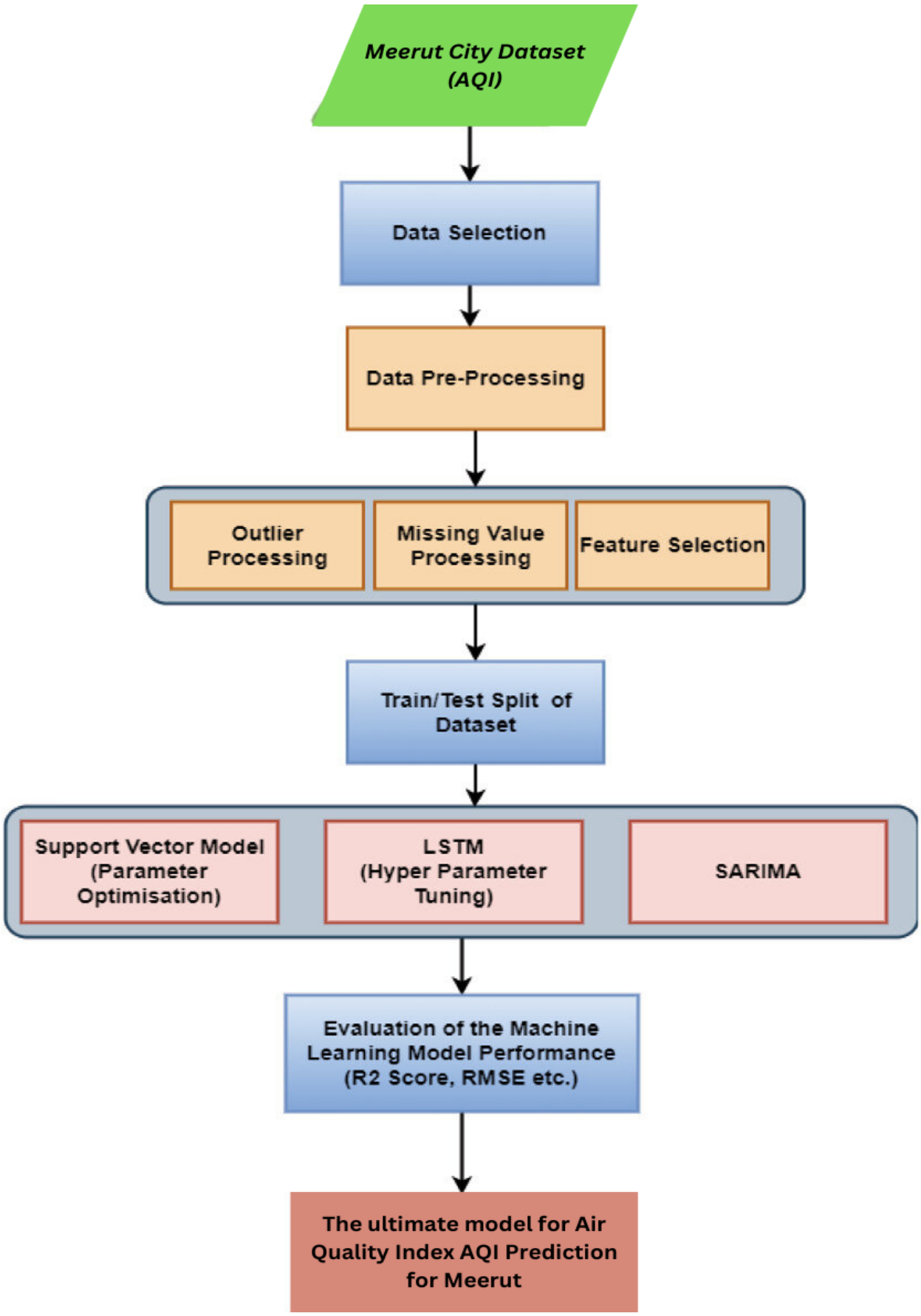

Fig.3.8.2.1 DFD of AQI Predictor of Meerut City

3.8.3. SEQUENCE DIAGRAM

The sequence diagram for the AQI predictor system outlines the interactions between users and the system components during key activities. It ensures that the order of operations and the flow of data are clearly understood, facilitating smooth implementation and user experience.

Check Current AQI: The process begins with the user initiating the login sequence. Upon entering their credentials, the system verifies the user's identity through the authentication mechanism, ensuring secure access. Once authenticated, the user proceeds to select a location for which they wish to view the current air quality index (AQI). The system then accesses the AQI Data Store to fetch the most up-to-date AQI data for the selected location. This involves retrieving relevant data, such as pollutant levels and overall AQI value, which is then displayed on the user interface. Users can interact with the displayed data by delving deeper into specific pollutants or switching to view AQI data for different locations. This interactivity allows users to gain detailed insights into the air quality in various areas. The sequence concludes when the user logs out, securely ending their session.

Generate Reports: For the report generation sequence, the user begins by logging into the system. After successful authentication, the user specifies the parameters for the report, including the date range, location, and type of report required. The system responds by querying the AQI Data Store for historical AQI data that matches the specified parameters. This data retrieval step is critical for ensuring that the report is comprehensive and accurate. Once the data is gathered, the system processes it to generate a detailed report. This report includes various visualizations and analyses, helping users to understand trends and patterns in air quality over the selected period. The system then presents the report to the user, who can choose to download or print it for further use. The report generation process concludes with the user logging out, ensuring that their session is securely terminated.

3.8.4. CLASS DIAGRAM

The class diagram for the AQI predictor system represents the system's structural components and their relationships, defining the attributes and methods of each class, which collectively form the backbone of the system's functionality.

User Class:

This class encapsulates user-related attributes and behaviors. Attributes include userID, name, email, and password, while methods involve login(), logout(), and viewCurrentAQI().

The login() and logout() methods handle authentication, ensuring that users can securely access and exit the system. The viewCurrentAQI() method allows users to request the current air quality information for a specific location.

Location Class:

This class stores information about various locations for which AQI data is available. Attributes include locationID, cityName, and coordinates. Methods in this class, such as getLocationDetails() and updateLocation(), facilitate retrieving and managing location-specific data. The getLocationDetails() method fetches location information, while the updateLocation() method allows for modifications to existing location data.

AQIDataStore Class:

This class is crucial for managing the AQI data. It includes attributes such as dataID, timestamp, pollutantLevels, and AQIValue. Methods like fetchCurrentAQI(), fetchHistoricalAQI(), and updateAQIData() are essential for data retrieval and updates. The fetchCurrentAQI() method retrieves the latest AQI data, fetchHistoricalAQI() gathers past AQI data for trend analysis, and updateAQIData() ensures the data store is always current with the latest air quality readings.

Report Class:

This class handles the generation and management of reports. Attributes include reportID, reportType, dateRange, and generatedBy (linking to the User class). Methods such as generateReport() and exportReport() are designed to create and distribute detailed reports. The generateReport() method processes the requested data and compiles it into a coherent report, while the exportReport() method allows users to download or print the report.

VisualizationTool Class:

This class provides functionalities for data visualization. Attributes include visualizationID, visualizationType, and dataPoints. Methods such as displayGraph() and customizeView() are included to offer interactive visualizations of AQI data. The displayGraph() method renders graphical representations of the data, and customizeView() allows users to adjust the visualization settings for better analysis.

Authentication Class:

This class manages user authentication processes. Attributes include sessionID and authenticationStatus. Methods like validateUser() and endSession() ensure secure login and

logout procedures. The validateUser() method verifies user credentials against stored data, and endSession() terminates the user's session securely.

SystemSettings Class:

This class allows for the configuration of system parameters. Attributes include settingsID, parameterName, and parameterValue. Methods such as updateSetting() and getSetting() help manage the system’s operational settings. The updateSetting() method modifies system parameters as needed, while the getSetting() method retrieves current settings for review or adjustment.

CHAPTER- 4

SCREENSHOTS

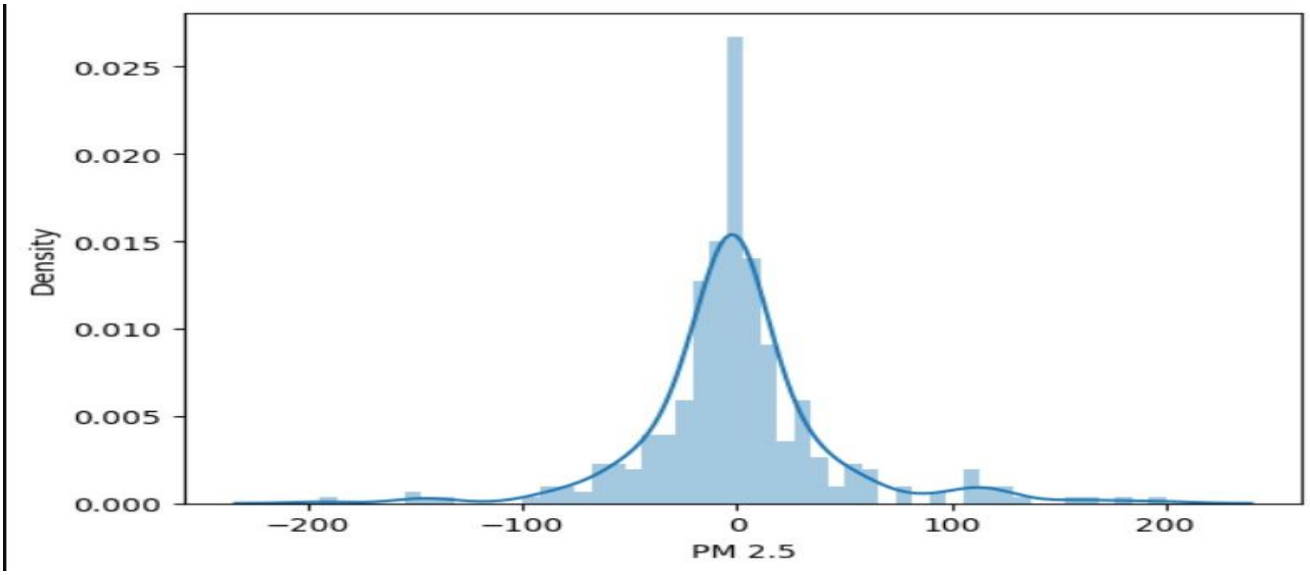

Fig.4.1: Relationship between density and PM2.5 levels in the context of air quality, providing valuable insights into variations of particulate matter concentrations with differing atmospheric densities

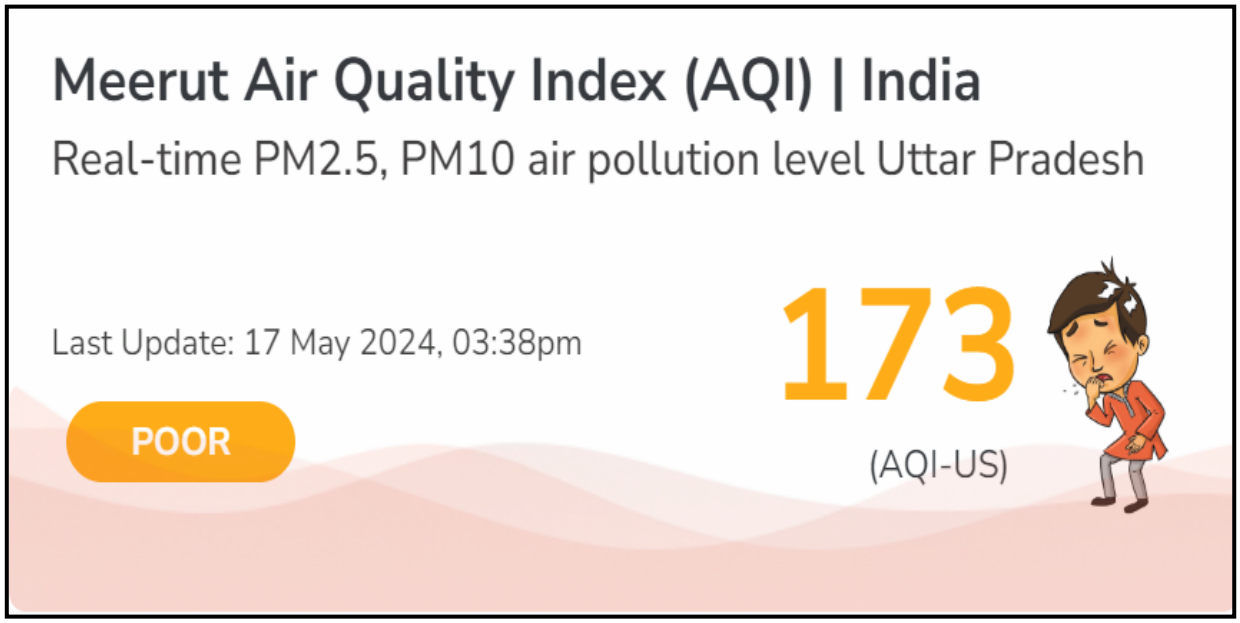

Fig.4.2: AQI of Meerut City

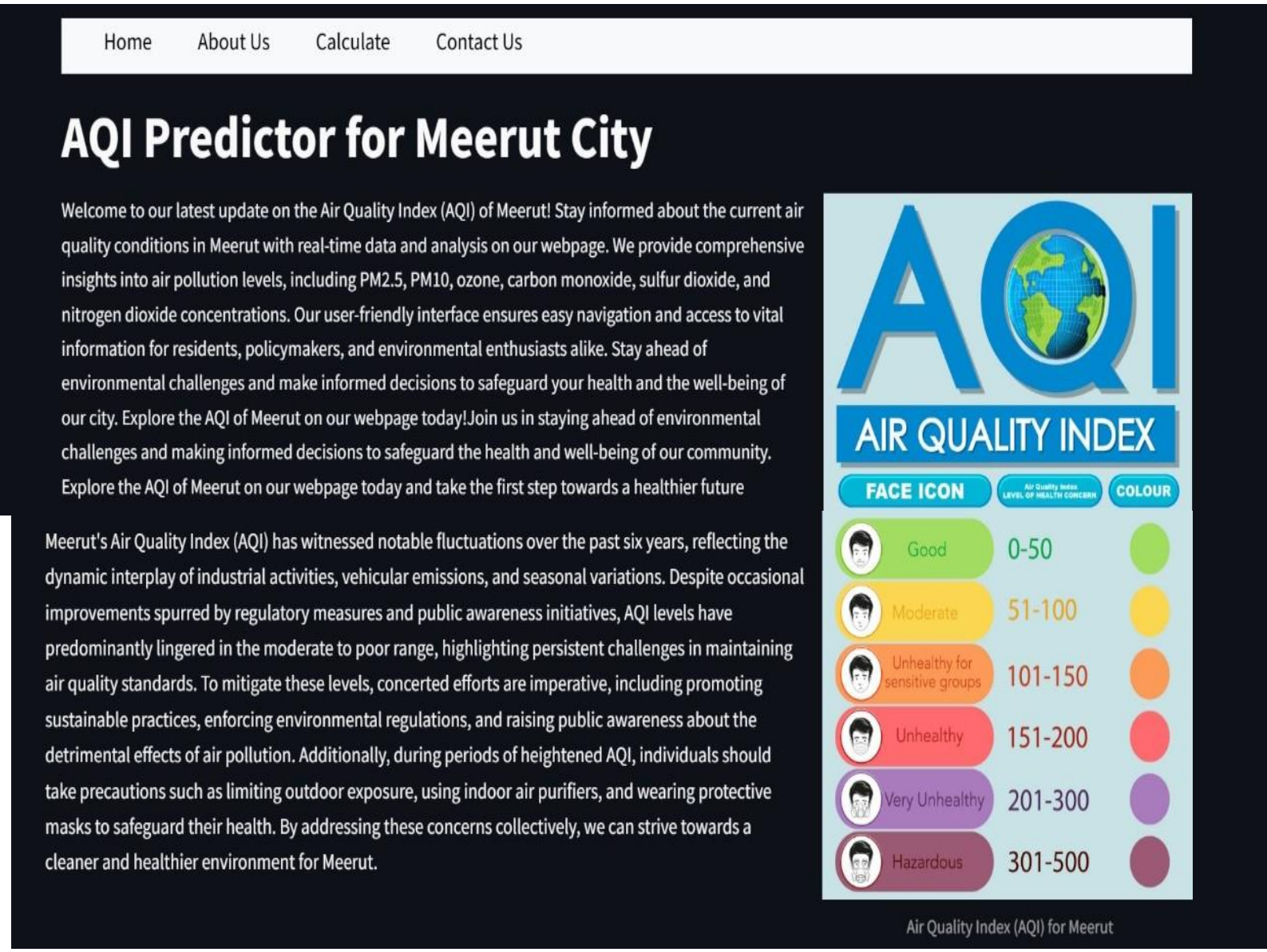

Fig.4.3: Home Page

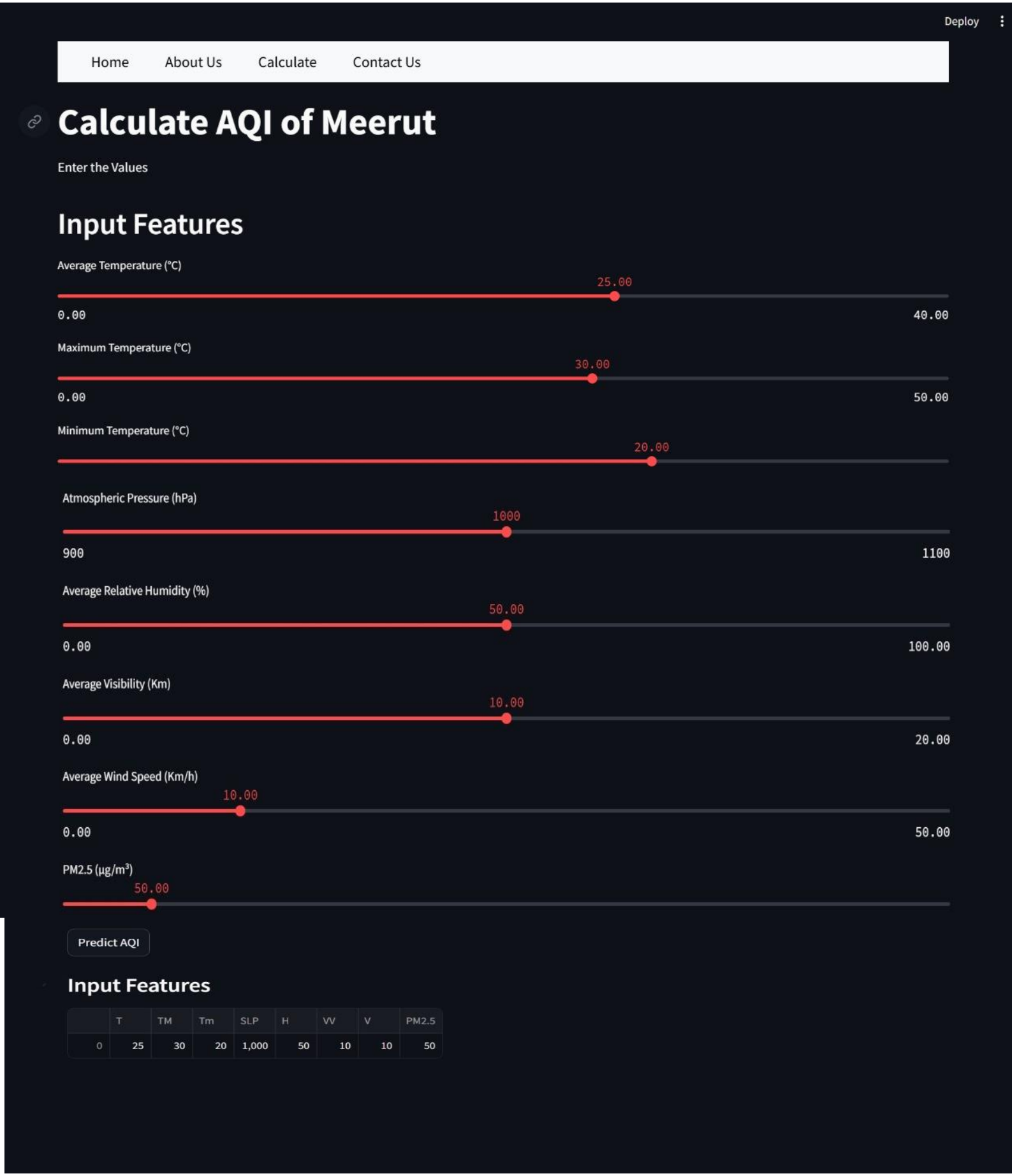

Fig.4.4: Calculate Page

Fig.4.5: Real-Time AQI Prediction Page

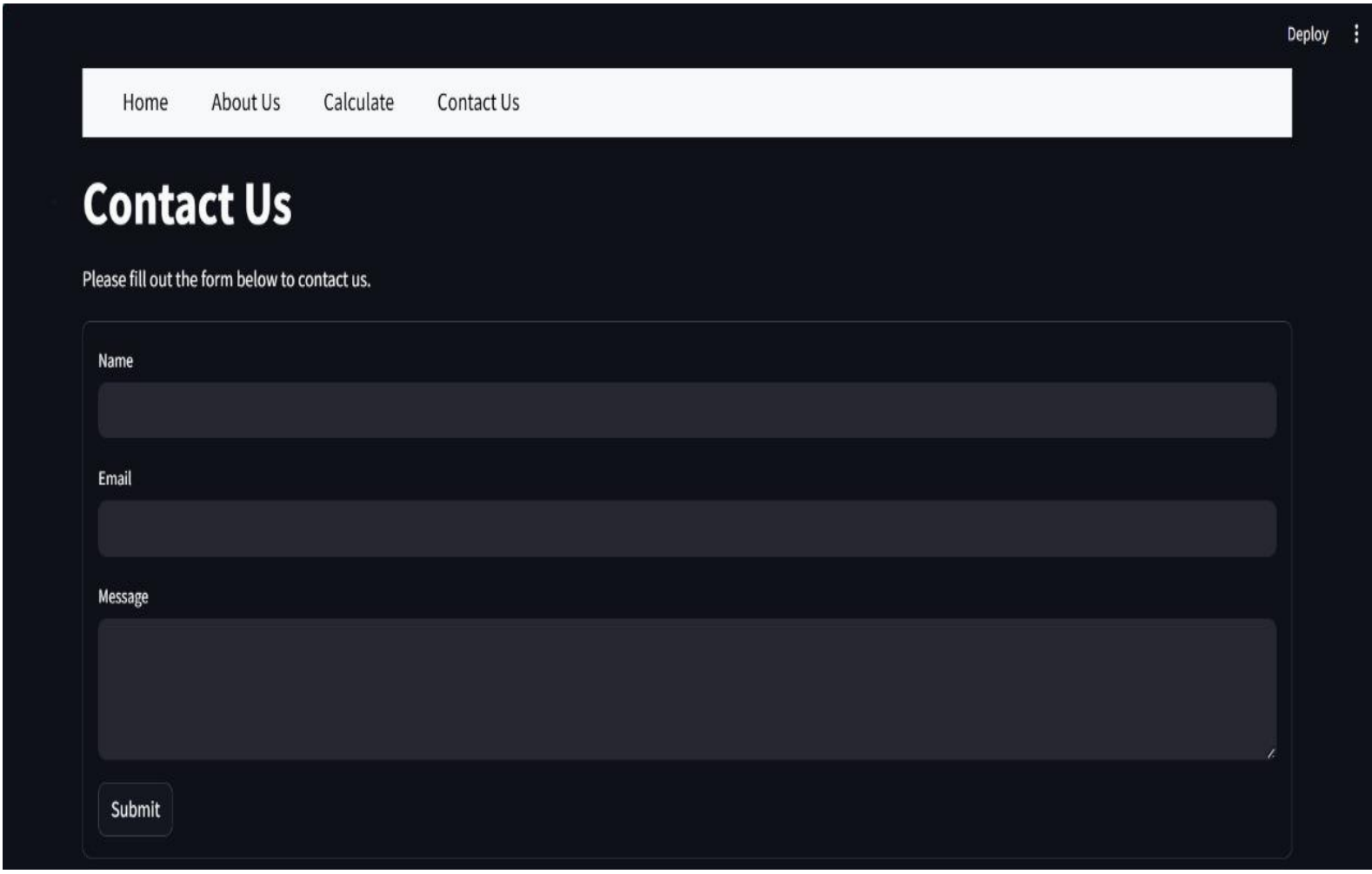

Fig.4.6: Contact Us Page

CHAPTER -5
TECHNOLOGY USED

The development of the AQI predictor system for Meerut city leverages a variety of advanced technologies, software tools, and methodologies. These technologies ensure the system is robust, scalable, and capable of providing accurate air quality predictions.

5.1.Programming Languages:

- **Python:** Python is the primary programming language used for data processing, machine learning model development, and integration tasks. Its extensive libraries and frameworks, such as NumPy, Pandas, and Scikit-learn, make it ideal for handling large datasets and performing complex computations.
- **HTML/CSS/JavaScript:** These are used for developing the web interface. HTML structures the web pages, CSS is used for styling, and JavaScript adds interactivity to the web application, enhancing the user experience.

5.2.Machine Learning Libraries and Frameworks:

- **Scikit-learn:** This Python library is utilized for implementing machine learning algorithms like Linear Regression, Lasso Regression, Decision Tree Regression, Random Forest Regression, and XGBoost. Scikit-learn provides tools for model training, evaluation, and validation.
- **XGBoost:** Specifically used for its gradient boosting capabilities, XGBoost enhances the accuracy of the predictions by handling large-scale datasets efficiently and improving model performance.

5.3.Data Processing and Analysis Tools:

- **Pandas:** Pandas is used for data manipulation and analysis. It provides data structures and functions needed to clean and prepare the AQI data for machine learning models.
- **NumPy:** NumPy is utilized for numerical computations, handling arrays and matrices, which are essential for performing mathematical operations on the data.

5.4.Data Visualization Tools:

- **Tableau:** Tableau is employed to create interactive and insightful dashboards that visualize the AQI data, trends, and predictions. It helps users understand complex data through graphical representations.

- **Matplotlib and Seaborn:** These Python libraries are used for generating static, animated, and interactive visualizations of the data, aiding in exploratory data analysis and reporting.

5.5.Web Scraping Tools:

- **BeautifulSoup:** Used for web scraping tasks to gather historical AQI data from various online sources. BeautifulSoup helps parse HTML and XML documents, allowing for easy extraction of data.
- **Selenium:** Selenium is used for automated web scraping, particularly for dynamic websites where data is loaded through JavaScript. It automates browser interactions to fetch data programmatically.

5.6.Web Development Frameworks:

- **Flask/Django:** These Python web frameworks are considered for developing the backend of the web application. They handle routing, database interactions, and server-side logic.
- **Bootstrap:** A front-end framework used to create responsive and mobile-first web pages. It simplifies the design process with pre-built components and a grid system.

5.7.Database Management:

- **SQLite/MySQL:** These relational database management systems store historical AQI data, user information, and other relevant datasets. They provide efficient data retrieval and management capabilities.

CHAPTER- 6
TESTING AND INTEGRATION

6.1.TEST CASE DESCRIPTION

Test case descriptions are essential for the software testing process, providing a comprehensive guide to verify the functionality and performance of the AQI predictor system. Each test case description includes key components to ensure thorough evaluation. A unique Test Case ID helps track and manage each test case. The title offers a brief overview of the test case's purpose, such as "Verify Data Preprocessing Functionality." The description provides detailed objectives and the specific system aspect being tested, like ensuring the data preprocessing module handles missing values, outliers, and data inconsistencies correctly. Prerequisites outline any conditions or requirements needed before execution, such as access to specific datasets or system module configurations. Test data includes the input data used for testing, representing real-world scenarios the system will encounter. Steps to execute offer a detailed, step-by-step guide ensuring accurate test replication by different testers. Expected results define clearly measurable outcomes anticipated from the test, like processed data without missing values and correctly handled outliers. Actual results are the observed outcomes after test execution, determining if the test case passes or fails based on the comparison with expected results. Pass/Fail criteria set the standards for test success, and remarks provide additional notes or observations, including unexpected behaviors or suggestions for improvement. This structured approach ensures a systematic and repeatable testing process, validating all functionalities and promptly addressing any issues in the AQI predictor system.

6.2.TYPES OF TESTING

There are various types of testing are employed to ensure the robustness and reliability of the AQI predictor system:

- **Unit Testing:** This involves testing individual components or functions of the system in isolation to ensure they work correctly. Each module, such as the data preprocessing unit or the machine learning model, is tested independently.
- **Integration Testing:** Integration testing focuses on verifying that the combined modules of the system work together as expected. This ensures that data flows correctly between the frontend, backend, and the machine learning models.

- **System Testing:** System testing involves testing the entire integrated system to ensure it meets the specified requirements. This type of testing is performed in an environment that closely resembles the production environment.
- **Regression Testing:** Regression testing ensures that new code changes do not adversely affect existing functionalities. It involves re-running previously completed tests to confirm that the same results are achieved.
- **User Acceptance Testing**: UAT is performed by the end-users to ensure the system meets their expectations and requirements. This is the final phase of testing before the system goes live.

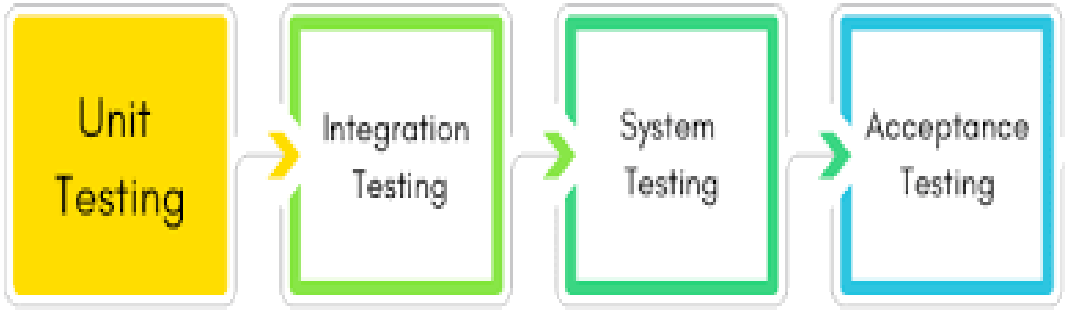

Fig.6.2.1 Types of Testing

6.3.TEST CASES

Test cases for the AQI predictor system are designed to ensure that each component functions correctly and meets the specified requirements. For instance, one test case focuses on data preprocessing, verifying that missing values are correctly handled either through imputation or removal, ensuring the reliability of the dataset. Another test case examines the implementation of the K-Means clustering algorithm, ensuring that it effectively divides the data into distinct clusters based on specified parameters. Additionally, there are test cases for interactive visualization tools, ensuring they display clustering results accurately and allow user interactions like zooming and filtering. The system's ability to calculate evaluation metrics, such as the silhouette score and Davies–Bouldin index, is also tested to ensure the quality of clustering results. Furthermore, test cases for report generation verify that the system produces comprehensive and accurate reports that provide detailed insights and visualizations of the clustered data. These test cases collectively ensure the AQI predictor system is robust, reliable, and provides meaningful insights into air quality patterns.

6.4.FUTURE ENHANCEMENT

Future enhancements to the AQI predictor system could focus on several key areas to improve its functionality and user experience:

- **Real-time Data Integration:** Incorporate real-time data feeds from sensors and weather stations to provide up-to-date AQI predictions.
- **Enhanced Machine Learning Models:** Explore advanced machine learning and deep learning algorithms to improve prediction accuracy and handle more complex data relationships.
- **Mobile Application Development:** Develop a mobile application to provide users with convenient access to AQI predictions and alerts on their smartphones.
- **Geographical Expansion:** Extend the system to cover additional cities and regions, adapting the models to handle different environmental conditions and pollutant sources.
- **User Feedback Mechanism:** Implement a feature for users to provide feedback on the predictions and system performance, using this data to continuously improve the models and interface.
- **Predictive Analytics for Health Impacts:** Integrate health impact predictions based on AQI levels, offering users personalized advice and health recommendations.
- **Visualization and Reporting:** Enhance the visualization capabilities with more interactive and customizable dashboards, and provide detailed reports that can be used by local authorities and policymakers.
- **Integration with IoT Devices:** Integrate the system with Internet of Things (IoT) devices to collect more granular data on air quality from various locations in real-time.

CONCLUSION

The development of the AQI predictor for Meerut city marks a significant step towards addressing the critical issue of air pollution and its impact on public health and the environment. This project leverages historical air quality data and employs advanced machine learning algorithms to forecast the Air Quality Index (AQI), thereby enabling residents, local authorities, and businesses to make informed decisions and take proactive measures to mitigate pollution.

Throughout the project, various methodologies and technologies have been utilized to ensure the system's accuracy and reliability. From data collection through web scraping to data preprocessing, the foundation has been laid for effective machine learning model training. The use of multiple regression models, such as Linear Regression, Lasso Regression, Decision Tree Regression, Random Forest Regression, and XGBoost, has provided a comparative analysis of their performance, with Random Forest Regression emerging as the most accurate with an accuracy rate of 70%.

Interactive visualization tools, created using Tableau, have played a crucial role in making the data and predictions accessible and understandable to non-technical users. These tools facilitate the exploration of historical data and forecasted trends, aiding in the dissemination of vital information to the public and stakeholders.

The project's feasibility studies—operational, technical, and economic—have confirmed its viability within the existing infrastructure and budget constraints, ensuring that it can be sustainably integrated into current systems. Moreover, the comprehensive testing phase, encompassing various test cases, has validated the system's functionality, robustness, and user-friendliness.

In conclusion, the AQI predictor system not only enhances the capability to monitor and forecast air quality but also empowers the community to take necessary actions to improve air quality. By providing timely and accurate predictions, this system supports public awareness campaigns, regulatory decisions, and individual choices aimed at reducing pollution and fostering a healthier environment in Meerut city. Future enhancements may include expanding the dataset, incorporating additional pollutants, and integrating real-time data streams to further improve the system's accuracy and responsiveness.

REFERENCES

1. Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5-32.
2. Bureau of Indian Standards (BIS). (2014). National Ambient Air Quality Standards. Retrieved from [https://cpcb.nic.in/National-Ambient-Air-Quality Standards.php]
3. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794.
4. Commission for Air Quality Management in NCR and Adjoining Areas. (2023). Measures to control air pollution.
5. Environmental Protection Agency (EPA).(2020). Air Quality Index (AQI) Basics.
6. Friedman, J., Hastie, T., & Tibshirani, R. (2010).Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software, 33(1), 1-22.
7. Gupta, M., & Kumar, A. (2017). A review of machine learning techniques for predicting air quality index. Environmental Monitoring and Assessment, 189(5), 249.
8. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.
9. He, K., Huo, H., & Zhang, Q. (2016). Urban air pollution in China: Current status, characteristics, and progress. Annual Review of Energy and the Environment, 31(1), 303-344.
10. Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. Proceedings of the 14th International Joint Conference on Artificial Intelligence, 2(12), 1137-1143.
11. Liu, F., & Sun, Y. (2018). Air quality index prediction based on random forests. Proceedings of the International Conference on Big Data Analytics, 85-92.
12. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.
13. Seinfeld, J. H., & Pandis, S. N. (2016). Atmospheric Chemistry and Physics: From Air Pollution to Climate Change. John Wiley & Sons.
14. Vapnik, V. (1995).** The Nature of Statistical Learning Theory.
15. Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(2), 301-320.

APPENDIX

RESEARCH PROBLEM

The project aims to develop an Air Quality Index (AQI) predictor for Meerut city. The escalating concerns about deteriorating air quality necessitate a robust system that forecasts AQI based on historical data. This predictor will empower residents, local authorities, and businesses to proactively address air quality issues, fostering a healthier and more sustainable urban environment. The system's accuracy and timely predictions will be paramount in guiding public awareness campaigns, regulatory decisions, and individual choices to mitigate the adverse effects of air pollution.

Random forest regression

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import sklearn
import seaborn as sns
df=pd.read_csv('C:\\Users\\aayus\\AQI-Project-master\\AQI-
Project-master\\Data\\Real-Data\\updated_combine.csv')


df.head()
## Check for null values

sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='vir
idis')
df=df.dropna()
X=df.iloc[:,:-1] ## independent features
y=df.iloc[:,-1] ## dependent features
## check null values
X.isnull()
y.isnull()
sns.pairplot(df)
df.corr()
```

Correlation matrix with heatmap

```
import seaborn as sns
#get correlations of each features in dataset
corrmat = df.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
corrmat.index
```

Feature importance

```
from sklearn.ensemble import ExtraTreesRegressor
import matplotlib.pyplot as plt
model = ExtraTreesRegressor()
model.fit(X,y)
X.head()
print(model.feature_importances_)

#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.show()
sns.distplot(y)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

Train test split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
from sklearn.ensemble import RandomForestRegressor
regressor=RandomForestRegressor()
regressor.fit(X_train,y_train)
print("Coefficient of determination R^2 <-- on train set: {}".format(regressor.score(X_train, y_train)))
```

```
print("Coefficient of determination R^2 <-- on test set:
{}".format(regressor.score(X_test, y_test)))
from sklearn.model_selection import cross_val_score
score=cross_val_score(regressor,X,y,cv=5)
score.mean()
prediction=regressor.predict(X_test)
sns.distplot(y_test-prediction)
plt.scatter(y_test,prediction)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
import matplotlib.pyplot as plt

# Assuming y_test and prediction are already defined
plt.scatter(y_test, prediction, c=prediction, cmap='viridis')

# Adding labels to the axes
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')

plt.colorbar(label='Prediction')

plt.show()

RandomForestRegressor()
from sklearn.model_selection import RandomizedSearchCV
n_estimators = [int(x) for x in np.linspace(start = 100, stop
= 1200, num = 12)]
print(n_estimators)
 #Randomized Search CV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop
= 1200, num = 12)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
# max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# Minimum number of samples required at each leaf node
```

```
min_samples_leaf = [1, 2, 5, 10]
# Method of selecting samples for training each tree
# bootstrap = [True, False]

# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}

print(random_grid)
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestRegressor()
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations
rf_random = RandomizedSearchCV(estimator = rf,
param_distributions =
random_grid,scoring='neg_mean_squared_error', n_iter = 100, cv
= 5, verbose=2, random_state=42, n_jobs = 1)
rf_random.fit(X_train,y_train)
rf_random.best_params_
rf_random.best_score_
predictions=rf_random.predict(X_test)
sns.distplot(y_test-predictions)
plt.scatter(y_test,prediction)
from sklearn import metrics
print('MAE:', metrics.mean_absolute_error(y_test, prediction))
print('MSE:', metrics.mean_squared_error(y_test, prediction))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test,
prediction)))
```

Model Result

```
import pickle
# open a file, where you ant to store the data
file = open('random_forest_regression_model.pkl', 'wb')

# dump information to that file
pickle.dump(rf_random, file)
```