National Institute of Applied Sciences and Technology

CARTHAGE UNIVERSITY

End of Year Project

Specialty : **Software Engineering**

Enhancing Environmental Data Analysis through Dashboard Visualisation and Infrastructure Optimisation

Presented by

Arwa OUERIEMMI
Karim ELLOUZE
Selma GUEDIDI
Ons OUAHCHI

INSAT Supervisor : **Ms. SFAXI Lilia**
Company Supervisor : **M. Wang Paul S.**

Presented on : **05/06/2024**

JURY

Ms. Lilia Sfaxi (President)
Ms. Sana HAMDI (Reviewer)

Academic Year : 2023/2024

Acknowledgements

We would like to thank all the individuals who contributed to the success of our end-of-year project.

Firstly, we would like to thank Dr.Paul S.Wang, the Chair of the Department of the Computer Science of Morgan State University for providing us with this opportunity and for his guidance throughout the project.

We would also like to thank Ms.Lilia Sfaxi and Mr.Radhouane Chouchane for their patience, availability, and valuable advice and Ms.Sana Hamdi for the time she took for reviewing this work.

We would also like to express our gratitude to the entire team of the Data Science Research Laboratory at Morgan State University.

Table of Contents

List of Figures v

List of Tables vi

Résumé vii

Abstract viii

General Introduction 1

I Context, Problem, and Methodology 3

1 Context . . . 4

2 Problem statement . . . 4

2.1 Project Scope . . . 4

2.2 Challenges . . . 4

3 Current State Analysis . . . 5

4 Contributions . . . 6

4.1 Development of an ETL process for data extraction and integration . . . 6

4.2 Customized Visualization Solution for Air Quality Data . . . 6

4.3 Automated Deployment Pipeline for Dashboard Hosting . . . 7

5 Methodology . . . 7

5.1 Agile Methodology . . . 7

5.1.1 User Stories . . . 7

5.1.2 Product Backlog . . . 8

5.1.3 Sprints . . . 9

5.2 Gantt Chart . . . 9

II Extract-Transform-Load (ETL) 12

1 Architecture . . . 13

2 Data Extraction . . . 13

2.1 Data Sources . . . 13

2.1.1 Pre-compiled data . . . 15

2.1.2 Daily data . . . 15

2.1.3 Comparison . . . 15

2.2 Web scraping . . . 16
3 Data Transformation . . . 17
3.1 Preliminary Data Analysis . . . 17
3.2 Cleaning and Pre-processing . . . 18
3.3 Feature Extraction . . . 19
4 Data Loading . . . 20
4.1 Database Choice . . . 20
4.1.1 Comparison of Database Systems . . . 20
4.1.2 PostgresSQL . . . 20
4.2 Schema Design . . . 21
5 Full Process . . . 21

III Dashbboard Design and Development **25**
1 Dashboard Design . . . 26
1.1 System Requirements . . . 26
1.1.1 Functional Requirements . . . 26
1.1.2 Non-functional Requirements . . . 28
1.2 User Interface Design . . . 28
1.2.1 Design Principles . . . 28
1.2.2 User Flow . . . 29
1.2.3 Website Prototype . . . 30
2 Backend Development . . . 32
2.1 Architecture . . . 32
2.2 API Development . . . 33
3 Frontend Development . . . 34
3.1 Architecture . . . 34
3.2 Feature Implementation . . . 35
3.2.1 Charts . . . 35
3.2.2 Map Integration . . . 38
3.2.3 Timeline Component . . . 39
3.3 API Integration . . . 39
3.4 Client-side Caching . . . 39
4 Testing and Quality Assurance . . . 39
4.1 API Unit Testing . . . 39
4.2 Dashboard Unit Testing . . . 40

4.3 Performance Testing 40

IV Deployment **43**

1 Continuous Integration/Continuous Deployment (CI/CD) 43

2 Deployment Pipeline Setup 45

2.1 Version Control Systems Used 45

2.2 Automated Deployment Tools 45

3 Monitoring and Maintenance 49

3.1 Performance Monitoring and Alerting 49

3.2 Regular Maintenance Tasks 50

4 Scalability and Future Enhancements 51

Conclusion and Perspectives **53**

References **55**

List of Figures

I.1 AQI by year . . . 5
I.2 Pollution Statistics . . . 6
I.3 Project Tasks . . . 9
I.4 Gantt Chart . . . 10

II.1 ETL process architecture . . . 14
II.2 Correlation Matrix for first maximum value, arithmetic mean, and AQI . . . 18
II.3 AQI Levels [1] . . . 19
II.4 Database Schema . . . 22
II.5 Activity Diagram for the ETL Process . . . 23

III.1 Use Case Diagram . . . 28
III.2 User Flow for exploring Air Quality Levels Across the USA . . . 29
III.3 User flow for exploring Air Quality Trends Over Time . . . 29
III.4 User flow for interacting with the Map . . . 30
III.5 Description of website prototype 1 . . . 30
III.6 Description of website prototype 2 . . . 31
III.7 Description of website prototype 3 . . . 31
III.8 Description of website prototype 4 . . . 32
III.9 Backend's Architecture . . . 33
III.10 Frontend's Architecture . . . 34
III.11 Doughnut Chart for Levels of Concern Across the USA . . . 37
III.12 Bar Chart for Air Quality Through the Years . . . 37
III.13 Bar Chart for Air Quality Daily Trends . . . 37
III.14 Bar Chart for Air Quality Seasonal Trends . . . 38
III.15 Bar Chart for Air Quality Hourly Trends . . . 38
III.16 Performance Results . . . 41

IV.1 CI/CD Workflow for the Air Quality Monitoring and Visualization System . . . 45
IV.2 CI/CD Workflow for the Flask server . . . 47
IV.3 CI/CD Workflow for the Angular Application . . . 48
IV.4 CloudWatch dashboard for flask server environment health . . . 49
IV.5 CloudWatch dashboard for aqi-dashboard bucket metrics . . . 50

List of Tables

I.1 User Stories 8
I.2 Project Tasks and Details 8
I.3 Sprint Schedule and Deliverables 9

II.1 Comparison of Data Quality 16
II.2 Web Scraping Methodologies 17

III.1 Service Descriptions 35

Résumé

La pollution de l'air est un problème environnemental et de santé publique majeur à l'échelle mondiale. La surveillance et la compréhension des données sur la qualité de l'air sont essentielles pour remédier à ce problème. Cependant, la complexité et le volume des données rendent difficile leur analyse et leur interprétation de manière efficace. Les visualisations sont des outils puissants pour relever ce défi.

L'objectif de ce projet est de développer un tableau de bord complet à partir des données extraites de l'Agence de protection de l'environnement des États-Unis, permettant aux utilisateurs de visualiser l'évolution la qualité de l'air au fil des ans dans les différents États et comtés des États-Unis. L'accent sera spécifiquement mis sur la concentration de différents polluants, en utilisant des cartes et divers graphiques.

Abstract

Air pollution is a major global environmental and public health issue. Monitoring and understanding air quality data are essential to address this problem. However, the complexity and volume of data make it difficult to analyze and interpret them effectively. Visualizations are powerful tools for tackling this challenge.

The aim of this project is to develop a comprehensive dashboard using data extracted from the United States Environmental Protection Agency, allowing users to visualize the evolution of air quality over the years in different states and counties across the United States. The focus will specifically be on the concentration of various pollutants, using maps and various graphs.

General Introduction

Air pollution, driven by various harmful suspended substances, poses an urgent threat to both the environment and public health. Emissions from sources like vehicles, industries, and agricultural activities are primary contributors to this atmospheric pollution. These emissions produce pollution indicators such as fine particles, sulfur dioxide, nitrogen oxides, and volatile organic compounds, which adversely affect ecosystems and human health. This pollution is associated with increased respiratory diseases, allergies, and cardiovascular issues among exposed individuals, while also contributing to climate change and declining air quality.

To combat this air pollution, implementing a monitoring system to collect these pollution indicators is essential. However, the challenge lies in the volume of available data making the extraction of meaningful metrics and insights to inform decision-making processes difficult.

This project is part of a collaboration with Morgan State University titled "Enhancing Environmental Data Analysis through Dashboard Visualization and Infrastructure Optimization." Its primary aim is to enhance the analysis and visualization of air quality data across various counties and states in the USA to provide actionable insights for decision-making. This involves developing a comprehensive dashboard to efficiently represent air quality data and aid in understanding air quality trends and patterns.

To achieve these objectives, the project is divided into four major parts:

- **Data Collection and Scraping** The initial phase involves collecting and scraping air quality data from various sources using tools like Selenium. This allows us to compile a comprehensive dataset for analysis.
- **Data Analysis** The collected air quality data was analysed to identify significant relationships and patterns. Python was used for this analysis due to its powerful libraries and tools for data processing and statistical analysis.
- **Dashboard Development** Based on the analysis, the most relevant air quality data to visualize is determined. This involves deciding on key metrics and designing charts and an interactive map that effectively convey the insights. The result is a user-friendly dashboard, enabling users to explore the air quality data and gain insights easily.
- **CI/CD Pipeline** The final step is the creation of a fully automated CI/CD pipeline that builds, test and deploys both the front and back-end of the dashboard.

Part I

Context, Problem, and Methodology

Chapter I

Context, Problem, and Methodology

Summary

1 Context **4**

2 Problem statement **4**

2.1 Project Scope 4

2.2 Challenges 4

3 Current State Analysis **5**

4 Contributions **6**

4.1 Development of an ETL process for data extraction and integration . 6

4.2 Customized Visualization Solution for Air Quality Data 6

4.3 Automated Deployment Pipeline for Dashboard Hosting 7

5 Methodology **7**

5.1 Agile Methodology 7

5.2 Gantt Chart 9

Introduction

The following section delves into the context, problem statement, and methodology employed in the project. This comprehensive overview provides insights into the environmental data analysis domain, outlines the challenges addressed, and elucidates the methodologies utilized to tackle these challenges effectively.

1 Context

This work originated from and is presided by Dr.Paul S. Wang, the Chair of the Department of Computer Science of Morgan State University. Morgan State University, renowned for its dedication to pioneering research and innovation, has been selected to lead this initiative focused on air quality analysis.

2 Problem statement

2.1 Project Scope

The primary scope of this project is to enhance the analysis and visualization of large volumes of environmental data to support informed decision-making by the U.S. Department of Health and Human Services. This involves developing methodologies for efficient data collection, integration, and visualization to provide actionable insights for policymakers.

2.2 Challenges

The primary challenge addressed by this project is the effective analysis and visualization of large volumes of environmental data. Although substantial data are available, extracting meaningful metrics and insights from these data to inform decision making in the US Department of Health and Human Services presents significant difficulties. Specifically, the project aims to overcome the following issues:

- **Data Overload**: The sheer volume of environmental data can be overwhelming, making it difficult to identify relevant patterns and trends.
- **Data Integration**: Combining data from multiple sources in a coherent and consistent manner.
- **Monthly Analysis**: Providing monthly updates and insights to support timely decision-making.
- **User Accessibility**: Ensuring that the data and insights are accessible and understandable to a broad audience, including policymakers and the general public.

By addressing these challenges, the project aims to enhance the capability of the U.S. Department of Health and Human Services to make informed decisions based on comprehensive environmental data analysis. This will ultimately contribute to better environmental management and policy development, promoting environmental justice and public health.

3 Current State Analysis

In the first stages of the project, the previous team's efforts were concentrated on extracting data from various sources, necessitating the implementation of specific tools and methodologies to streamline the process.

Tools:

The team utilized a blend of scrapy and selenium for scraping purposes, leveraging scrapy for certain datasets and resorting to selenium to overcome anti-scraping measures prevalent on websites primarily rendering HTML through JavaScript.

Methodologies:

Two primary approaches were adopted: traditional scraping, involving data extraction from HTML tables, and downloading data if provided by the website. The team followed a systematic process for most datasets, which included rendering the website using Selenium, parsing the dataset or initiating file downloads, managing downloaded files within dataset-specific folders, extracting files if necessary, converting formats to CSV, utilizing the Redivis API to create datasets, and uploading files as Redivis tables.

Additionally, a dashboard was created using Plotly, although it was not utilized in production due to the preference for a more interactive solution involving JavaScript as shown in the figures I.1 and I.2.

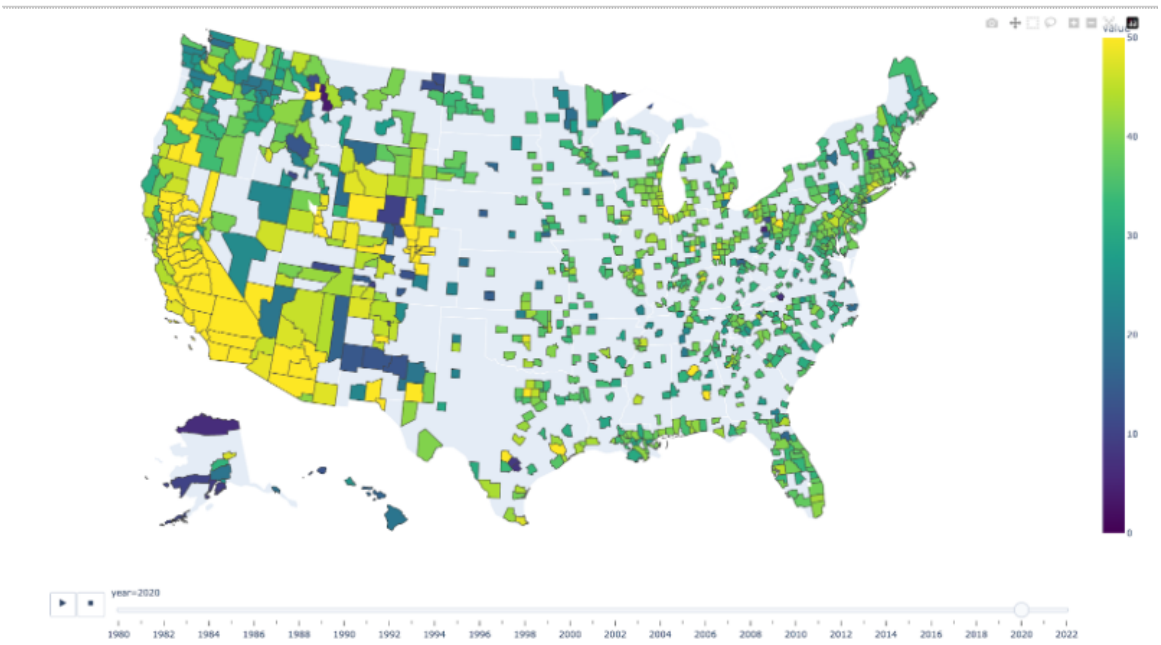

Figure I.1 – AQI by year

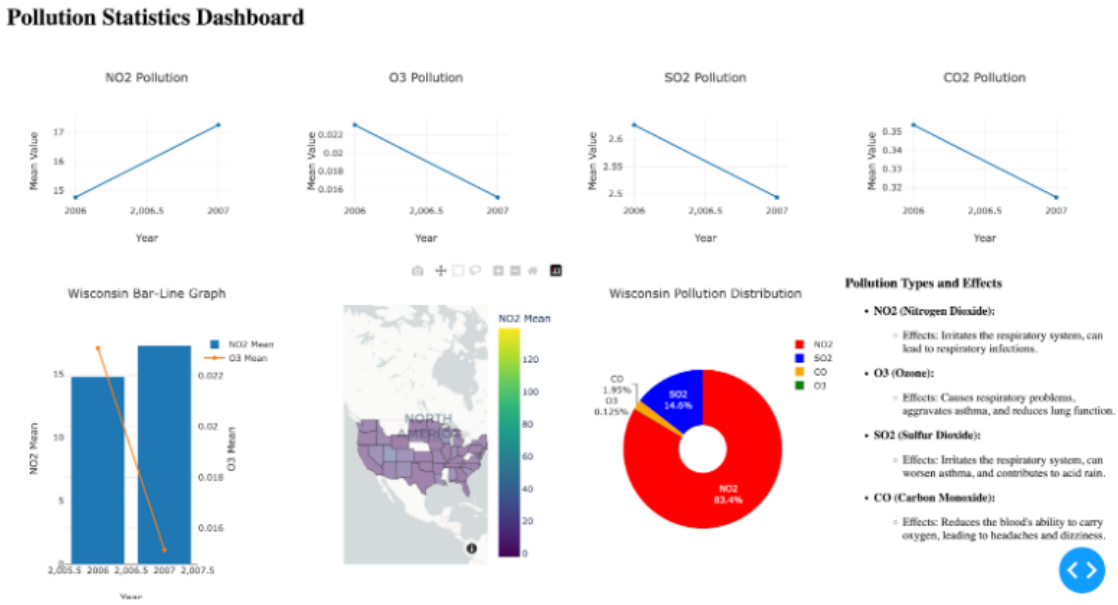

Figure I.2 – Pollution Statistics

4 Contributions

This work was divided into three parts. The first part focused on data acquisition and integration. The second phase involved designing and developing a dashboard to visualize the data. The final part pertained to the development of a CI/CD pipeline for deploying the dashboard.

4.1 Development of an ETL process for data extraction and integration

Our project introduces a fully automated ETL process where data extracted from the United States Environmental Protection Agency is first processed before being hosted in an AWS RDS PostgreSQL database.

4.2 Customized Visualization Solution for Air Quality Data

We have designed interactive dashboard for air quality data analysis. These solutions provide stakeholders with access to air pollution concentration data, allowing them to visualize spatial and temporal trends and make informed decisions regarding air quality management and public health interventions.

4.3 Automated Deployment Pipeline for Dashboard Hosting

We've implemented a comprehensive CI/CD pipeline to facilitate the seamless deployment of both front and backend components of our dashboard. This pipeline streamlines the deployment process, ensuring rapid and efficient updates to the dashboard's features and functionalities.

5 Methodology

5.1 Agile Methodology

For this project, a Scrum methodology was chosen. Agile Scrum combines the iterative, incremental philosophy of Agile with the structured, sprint-based framework of Scrum. Agile focuses on developing projects in small, manageable increments, allowing for continuous improvement and flexibility. Scrum, a subset of Agile, breaks down projects into "sprints," which are short, time-boxed periods during which specific tasks are completed [2].

The decision to use Scrum was driven by the need to deliver specific project components quickly and efficiently. Scrum's structure promotes regular progress checks, continuous feedback, and adaptability to changing requirements, making it ideal for dynamic and complex projects like ours. The methodology facilitates clear communication and ensures that the project stays on track through frequent meetings and reviews.

For this project, mandatory meetings were held every Friday at 6:00 PM to review progress, address any issues, and plan for the next sprint. Additionally, optional meetings every Wednesday at 2:00 PM were occasionally arranged for team members to ask questions and seek clarifications. This regular interaction helps maintain alignment, quickly resolve any obstacles, and keep the project moving forward smoothly.

5.1.1 User Stories

The user stories section presents a comprehensive overview of the various roles and their corresponding needs within the project as shown in the table I.1.

Table I.1 – User Stories

User Role	User Story
Environmental Scientist	As an environmental scientist, I want to visualize air quality data over time so that I can identify trends and anomalies.
Researcher	As a researcher, I want access to clean and well-structured air quality data so that I can conduct my studies efficiently.
Data Analyst	As a data analyst, I want to use an interactive dashboard to explore relationships between different air quality indicators.
Public Member	As a member of the public, I want to see current air quality levels in my area so that I can take necessary precautions.

5.1.2 Product Backlog

The product backlog, presented in the table I.2, outlines the prioritized list of tasks and features.

Table I.2 – Project Tasks and Details

Task	Details
Data Collection and cleaning	• Organize scraped data into structured formats (CSV). • Clean and validate data to remove inconsistencies and errors.
Data Analysis	• Analyze data to find patterns and identify significant trends. • Document findings and insights.
Dashboard Development	• Design the dashboard layout and user interface using Angular and implement charts and maps. • Test the dashboard for usability and performance.
Infrastructure Optimization	• Optimize data storage and processing on AWS. • Set up automated CI/CD pipelines for the front and backend of the project and ensure scalability.
Final Review and Documentation	• Prepare comprehensive documentation of the project.

5.1.3 Sprints

The project has been divided into 7 sprints as shown in the table I.3.

Table I.3 – Sprint Schedule and Deliverables

Sprint	Duration	Task	Deliverables
Sprint 1	Week 1-2	Project Comprehension and Initialization	Project plan, initial setup
Sprint 2	Week 3	Data Cleaning	Cleaned and validated datasets
Sprint 3	Week 4-5	Data Analysis	Analysis report, insights
Sprint 4	Week 5-7	Dashboard Development (Part 1)	Basic dashboard with initial visualizations
Sprint 5	Week 7-8	Dashboard Development (Part 2)	Fully functional dashboard
Sprint 6	Week 8-10	Infrastructure Optimization	A deployed dashboard
Sprint 7	Week 11	Final Review and Documentation	Final project report, presentations

5.2 Gantt Chart

A detailed Gantt chart was developed, outlining the various tasks, their start and end dates, and the overall duration of the project, which spans 74 days.Each task is strategically planned to ensure a smooth workflow and timely completion of the project objectives. The figure summarizing these tasks is shown in the figures I.3 and I.4.

Start Date	Start Date2	End Date	Durée
Project Initialization.	3/1/2024	3/7/2024	7
Project Comprehension.	3/8/2024	3/10/2024	3
Data Cleansing.	3/11/2024	3/14/2024	4
Data Analysis.	3/15/2024	3/28/2024	14
Visualization Strategy.	3/29/2024	4/6/2024	9
Dashbord Development .	4/7/2024	5/4/2024	30
Testing.	5/6/2024	11/05/2024	7

Figure I.3 – Project Tasks

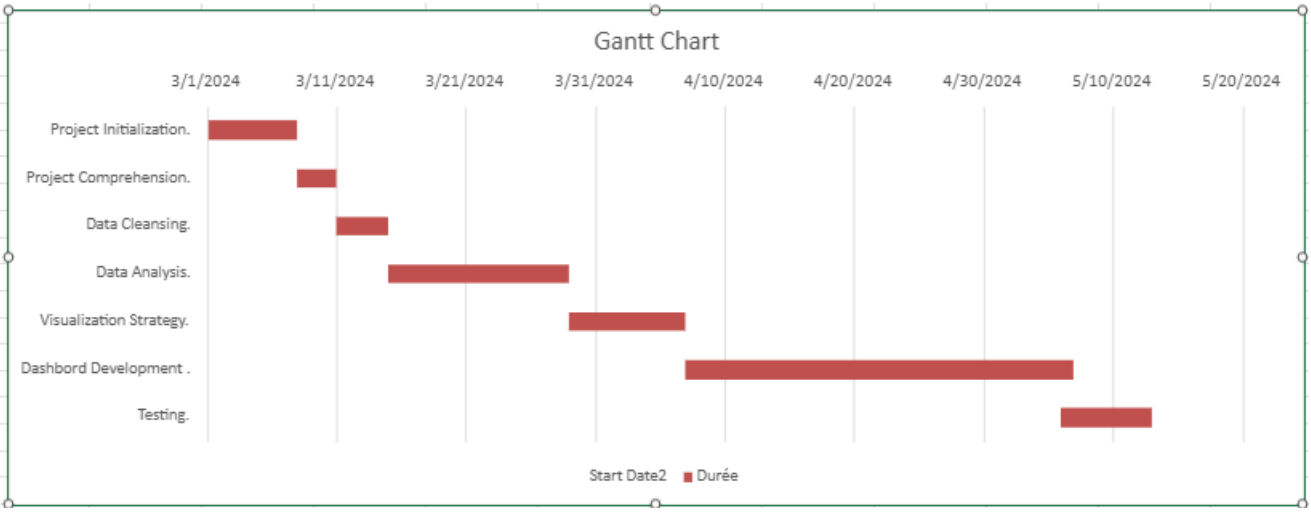

Figure I.4 – Gantt Chart

Conclusion

This chapter outlined the project's context, challenges, and methodology. The need for enhanced air quality data analysis through advanced visualization and optimized infrastructure was established. The primary challenge is managing the volume and complexity of air quality data, requiring sophisticated data processing and visualization.

A structured methodology was adopted, encompassing data scraping, cleaning, analysis, and the development of an interactive dashboard using Angular and AWS. The Agile Scrum methodology was chosen for its adaptability and iterative approach, ensuring continuous improvement and alignment with stakeholder needs. Regular meetings and sprints facilitated effective communication and timely progress.

Part II

Implementation

Chapter II

Extract-Transform-Load (ETL)

Summary

1 Architecture 13

2 Data Extraction 13

2.1 Data Sources 13

2.2 Web scraping 16

3 Data Transformation 17

3.1 Preliminary Data Analysis 17

3.2 Cleaning and Pre-processing 18

3.3 Feature Extraction 19

4 Data Loading 20

4.1 Database Choice 20

4.2 Schema Design 21

5 Full Process 21

Introduction

The Extract, Transform and Load (ETL) process is a methodology utilized in computing to consolidate data from multiple sources, transforming it into a consistent format, and loading it into a target database or data warehouse for analysis, reporting, and decision-making purposes. [3]. This chapter offers an overview of the ETL process for acquiring air quality data, detailing the transformation of raw data into processed data used for analysis.

1 Architecture

A layered architecture is being adopted for its intuitiveness and clear separation of concerns. This architectural approach divides the system into distinct layers as shown in figure II.1, with each layer responsible for specific functionalities:

- **Orchestration Layer** manages the workflow and coordination of tasks within the system, including initiating data retrieval and logging progress to the console.
- **ETL Logic Layer** Responsible for data transformation and loading, ensuring data quality, and avoiding duplication.
- **Data Acquisition Layer** manages the logic for downloading data from various sources using web scraping.
- **Database Layer** handles database interactions and operations, implementing database-specific logic such as querying, indexing, and transaction management.

2 Data Extraction

2.1 Data Sources

The air data is provided by the US Environmental Protection Agency concerning both criteria gases and particulates mainly [4]:

- **Ozone (O_3)** forms when pollutants react with sunlight. Ozone pollution is harmful to humans: it's an irritating gas that can cause headaches, eye irritation, nasal and throat irritation. At high concentrations, it can lead to various respiratory problems.
- **Particle Pollution (PM2.5 & PM10)** are tiny particles suspended in the air. PM2.5 particles have a diameter less than 2.5 microns and PM10 particles have a diameter less than 10 microns. They can penetrate deep into the lungs, causing respiratory and cardiovascular problems.
- **Nitrogen Dioxide (NO_2)** forms in the atmosphere from nitrogen monoxide (NO) emitted mainly during the combustion of fossil fuels. According to the World Health Organization (WHO), NO2 has harmful effects on health: long-term exposure can impair lung function and increase the risk of respiratory disorders.

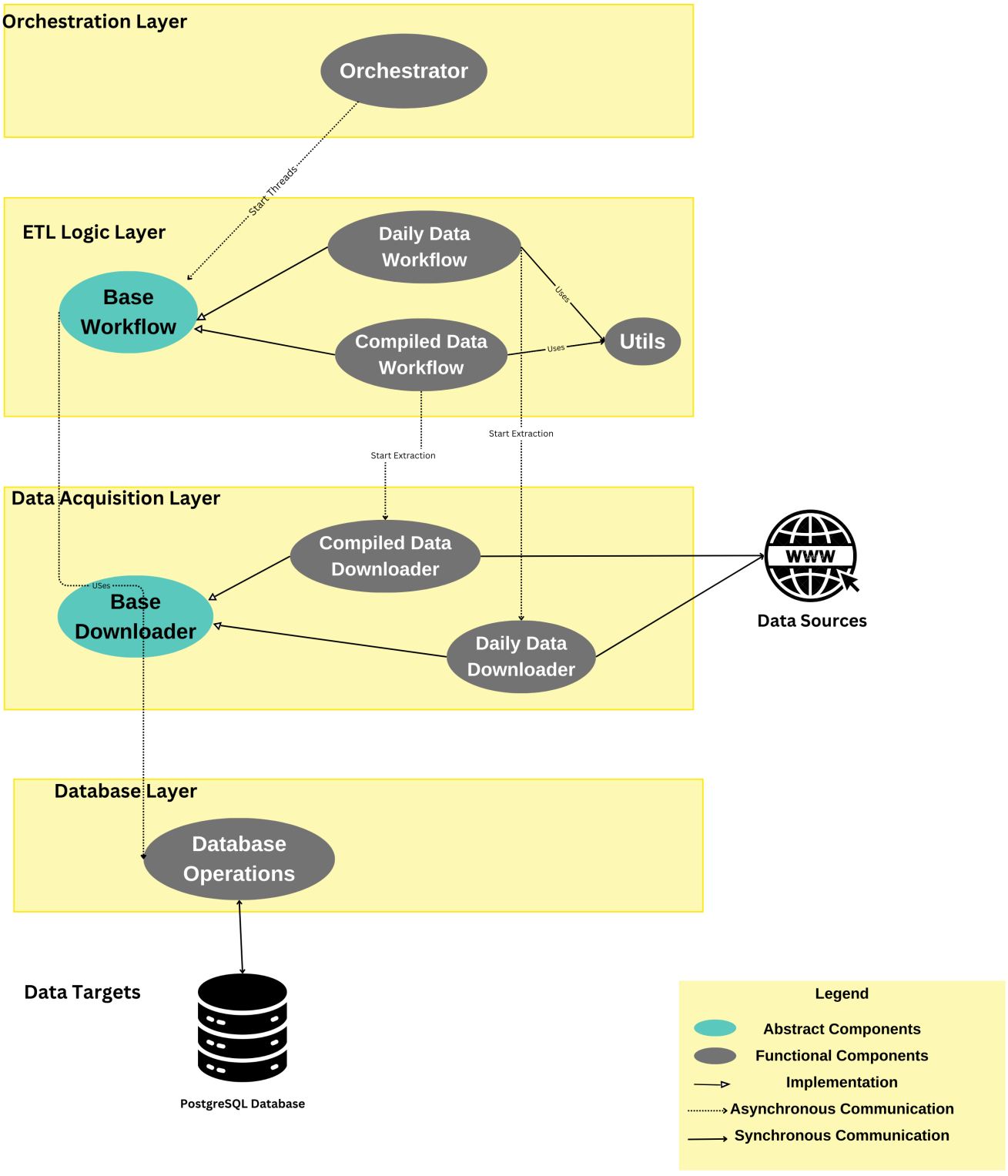

Figure II.1 – ETL process architecture

- **Sulfur Dioxide (SO_2)** is an atmospheric pollutant, considered a key indicator of pollution that is essential to monitor according to WHO recommendations to improve quality of life and public health. SO2 is irritating to the eyes and mucous membranes. At high concentrations, this pollutant can have health effects such as affecting the respiratory system and lung function in adults and causing or exacerbating respiratory diseases in sensitive individuals.
- **Carbon Monoxide (CO)** forms during incomplete combustion of organic matter. Inhalation can lead to headaches, dizziness, and fatigue, and prolonged exposure to this gas can be deadly.

2.1.1 Pre-compiled data

AirData offers access to air quality data collected across various locations in the United States, Puerto Rico, and the U.S. Virgin Islands. The primary source of this data is the Air Quality System (AQS) database, managed by the EPA meaning that this data has already been validated and verified as well as provides extra statistics such as the arithmetic mean of a pollutant's concentration and the hour where the max value was registered. This data source is compiled by year, but is not updated frequently [5].

2.1.2 Daily data

AirNow provides daily air quality summary statistics compiled by year and state. Although these data complement those available on AirData, they are not fully verified and validated through the quality assurance procedures used for the EPA's Air Quality System. However, they provide real-time updates on air quality data [5].

2.1.3 Comparison

In order to compare both sources, some common quality dimensions will be used [6]:

- **Freshness:** Reflects how recent and timely the data is.
- **Completeness:** Ensures all required data elements are present and available.
- **Accuracy:** Measures the correctness and reliability of data.

A comparison of the data quality from both sources is found in the table II.1.

Table II.1 – Comparison of Data Quality

Criteria	AirData	AirNow
Freshness	Non-regular updates (last update on October 2023)	Daily updates
Accuracy	Validated and verified by EPA	Not fully validated
Completeness	No null values	Some Missing values
File Format	CSV	CSV
Query Parameters	Pollutant and year	Pollutant, year and state
Columns	Date Source, Site ID, POC, Daily Mean Pb Concentration, Units Daily AQI, Value, Local Site Name, Daily Obs Count, Percent Complete, AQS Parameter Code, AQS Parameter Description, Method Code, CBSA Code, CBSA Name, State, FIPS Code State, County, FIPS Code County, Site Latitude, Site Longitude	State Code, County Code, Site Num, Parameter Code, POC, Latitude, Longitude, Datum, Parameter Name, Sample Duration, Pollutant Standard, Date Local, Units of Measure, Event Type, Observation Count, Observation Percent, Arithmetic Mean, 1st Max Value, 1st Max Hour, AQI, Method Code, Method Name, Local Site Name, Address State Name, County Name, City Name, CBSA Name, Date of Last Change

2.2 Web scraping

Web scraping is the process of autonomous data mining or gathering information from the Internet and other common databases. Different Web scraping methods have been developed in multiple types of research [7]. Web scraping involves several techniques, each suited to different types of web content and user interactions [8]. For this case, the following methods were employed in table II.2:

Table II.2 – Web Scraping Methodologies

Method	HTML Parsing	DOM Parsing	HTTP Requests
Definition	Technique to extract data from static HTML files by parsing structure.	Programs can obtain dynamic material generated by client-side scripts by placing a developed web browser	Data may be retrieved using socket programming via making HTTP requests to a remote web server
General Usage	Best for extracting data from websites with static content.	Used for websites requiring interaction such as form submissions, navigation menus, or any dynamic content manipulation.	Well-suited for APIs or web services where data is already formatted. Directly fetches data without rendering the page.
Use Case for Project	Retrieve the date where the AirData data was last updated and retrieve state and county information.	Download the AirNow CSV files.	Download the AirData CSV files.
Tool Being Used	BeautifulSoup	Selenium, WebDriver	Python requests library

3 Data Transformation

3.1 Preliminary Data Analysis

Before starting with data cleaning and pre-processing, a preliminary analysis was conducted to understand the data collected from the multiple CSV files. Throughout this analysis, it was noticed that:

- **Redundant Information:** Columns such as parameter name, code, and units of measure were found to contain only a single value throughout the dataset.
- **Strong Correlations:** There was a notable high correlation of 0.95 observed between the first maximum value, the arithmetic mean, and the AQI. This correlation is expected given that these metrics are directly related to the concentration levels of pollutants as shown in figure II.2.
- **Temporal Patterns:** Analysis revealed distinct seasonal and hourly trends in pollution levels, where air quality is worst at both midnight and winter.

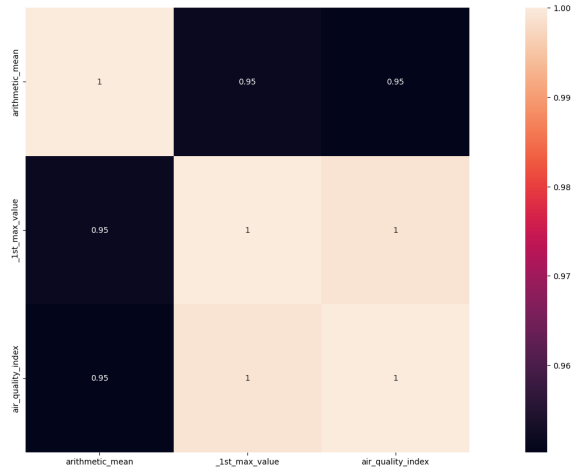

Figure II.2 – Correlation Matrix for first maximum value, arithmetic mean, and AQI

- **Outlier Analysis:** Very few outliers were detected per year, indicating a relatively stable dataset with few anomalies.

3.2 Cleaning and Pre-processing

The initial step in data transformation involved comprehensive data cleaning to address various issues. This process was aimed at enhancing the quality of the data, making it more suitable for analytical models. The following tasks were performed during this phase:

- **Structural Transformation:** Unnecessary columns, including those with a single value or irrelevant to the study's objectives, such as unit parameter name and code, and method name, were removed. The columns were also renamed to ensure clarity and consistency throughout the dataset.
- **Temporal Filtering:** All records preceding the last date saved in the existing tables were removed to avoid data redundancy.
- **Geographical Filtering:** Since this study only concerns the USA, data entries pertaining to locations outside the United States, specifically Canada, Puerto Rico, and Mexico, were identified and excluded from the dataset.
- **Data Integrity Correction:** Negative concentration values, resulting from flawed calibration, were removed.
- **Data Type Standardization:** State and county codes were converted into integers, and dates were transformed into the datetime.

- **Data Separation:** The concentration data was separated from AQI data for easier visualization in the next steps.

3.3 Feature Extraction

Following data cleaning and pre-processing, feature extraction was implemented to derive additional variables from the cleaned dataset. The extracted features are:

- **Time-based Features:** From the datetime information, the day of the week and the season were extracted to study seasonal and daily trends with respect to air pollution.
- **Air Quality Metrics:** Based on the concentration values and the AQI, the air quality category is determined to provide a more intuitive understanding of pollution data. The values are obtained as indicated in the figure II.3. The AQI for each pollutant is calculated using specific formulas that involve identifying the concentration breakpoints that bracket the pollutant concentration. This approach enables the extraction of the air quality category directly from the pollutant concentration [1].

Air Quality Index Levels of Health Concern	Numerical Value	Meaning
Good	0-50	Air quality is considered satisfactory, and air pollution poses little or no risk.
Moderate	51-100	Air quality is acceptable; however, for some pollutants there may be a moderate health concern for a very small number of people who are unusually sensitive to air pollution.
Unhealthy for Sensitive Groups	101-150	Members of sensitive groups may experience health effects. The general public is not likely to be affected.
Unhealthy	151-200	Everyone may begin to experience health effects; members of sensitive groups may experience more serious health effects.
Very Unhealthy	201-300	Health alert: everyone may experience more serious health effects.
Hazardous	> 300	Health warnings of emergency conditions. The entire population is more likely to be affected.

Figure II.3 – AQI Levels [1]

4 Data Loading

4.1 Database Choice

4.1.1 Comparison of Database Systems

Relational Databases (RDBMS) Relational databases, such as PostgreSQL and MySQL, use structured query language (SQL) for defining and manipulating data. They are known for strong consistency, ACID compliance, and robust transactional support.

- **MySQL** is known for its ease of use, flexibility, and efficiency in handling relatively simple query operations. However, it sometimes struggles with complex queries and very large databases or with high concurrency requirements.
- **PostgreSQL** offers advanced features such as complex queries, foreign keys, joins, views, transactional integrity, and reliability. It is highly extensible, supports custom data types, and is more standards-compliant than MySQL.

NoSQL Databases NoSQL databases such as MongoDB provide greater scalability and performance for unstructured data scenarios.They are designed for robust performance across distributed systems due to their ability to scale horizontally, enhancing flexibility and fault tolerance [9].

- **MongoDB** offers a flexible document-oriented structure that is ideal for projects with rapidly evolving schemas. However, it sacrifices some transactional consistency features typical of RDBMS, which can be a drawback for applications requiring strong data integrity.

4.1.2 PostgresSQL

After evaluating various databases, **PostgreSQL** was chosen as the primary database system for this project due to its robustness, advanced feature set, and strong support for data integrity and performance optimizations:

- **Performance:** PostgreSQL's ability to effectively use indexes significantly boosts performance, especially for complex queries involving large datasets.
- **Data Integrity:** PostgreSQL enforces data integrity through the use of constraints such as primary keys, foreign keys, and unique constraints. These ensure that the data adheres to specified rules, thereby reducing errors and enhancing reliability.

Finally, the database will be hosted on AWS RDS, for its scalable infrastructure to ensure high availability and manageability without the need for extensive database administration tasks.

4.2 Schema Design

In the design of the database system, a denormalized schema was adopted. This decision was influenced by the need to optimize read performance, simplify query complexity, and accelerate data retrieval as shown in figure II.4. Moreover, tables containing pollutant data are indexed by month and year, considering that most queries will depend on these two columns. To ensure data integrity, the tables for states and counties were populated separately with information provided by the Federal Information Processing Standards.

5 Full Process

The final ETL process is the same for both data sources and is presented in the figure II.5.

Conclusion

This chapter has provided a detailed look at the development and implementation of an ETL process designed to manage air quality data. The process began with the extraction of raw data from various environmental organizations. A preliminary analysis of the data was conducted to decide on necessary transformations. In the transformation phase, data was cleaned, adapted to the database schema, and new features were extracted. The final stage of the ETL process involved loading the processed data into a relational database hosted on AWS.

Challenges included ensuring complete automation of the process without the need for manual changes before the job is run, by using scraping techniques and caching to determine when the process should start. The end result was a fully automated pipeline that would run monthly to keep the database up to date.

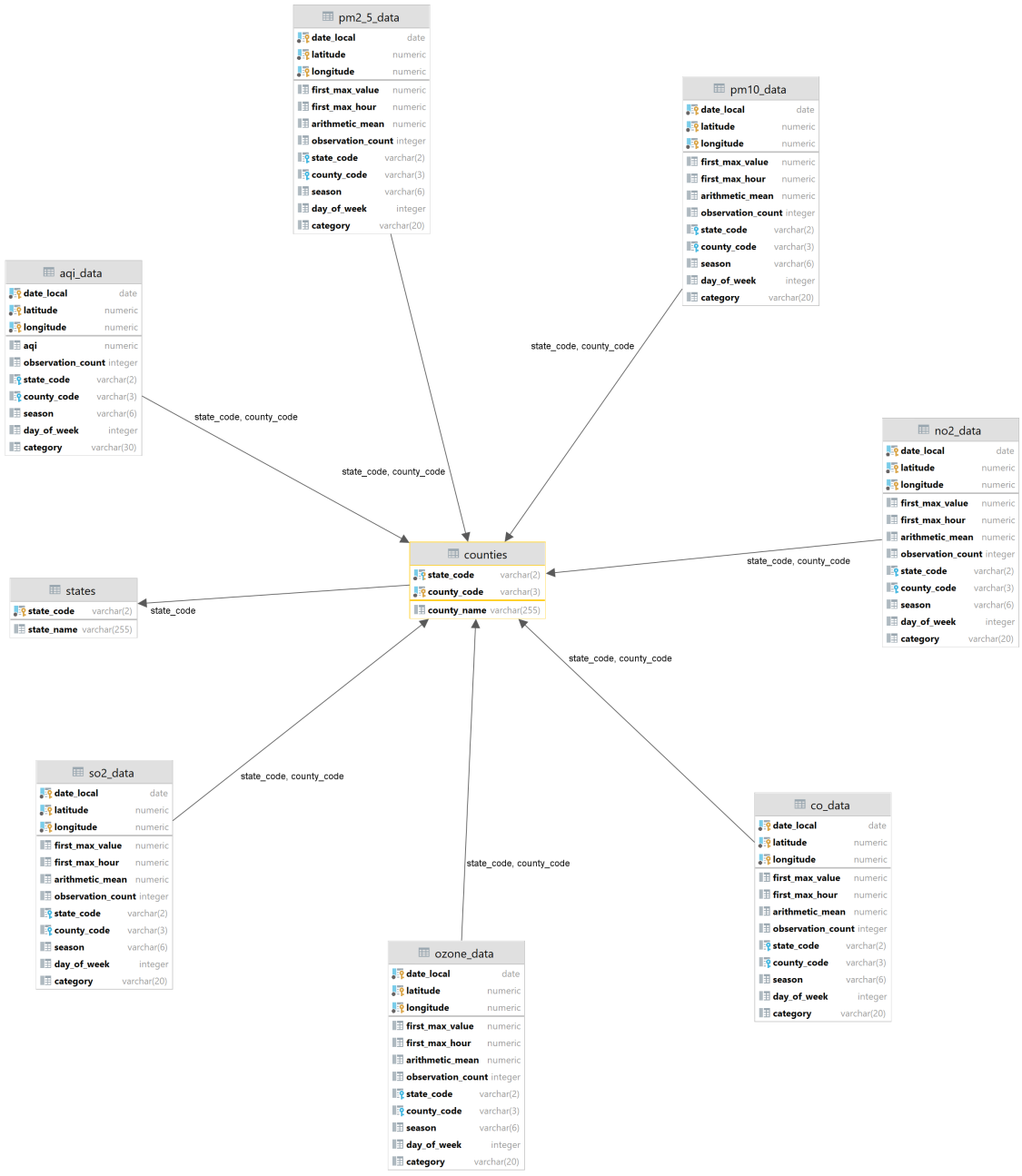

Figure II.4 – Database Schema

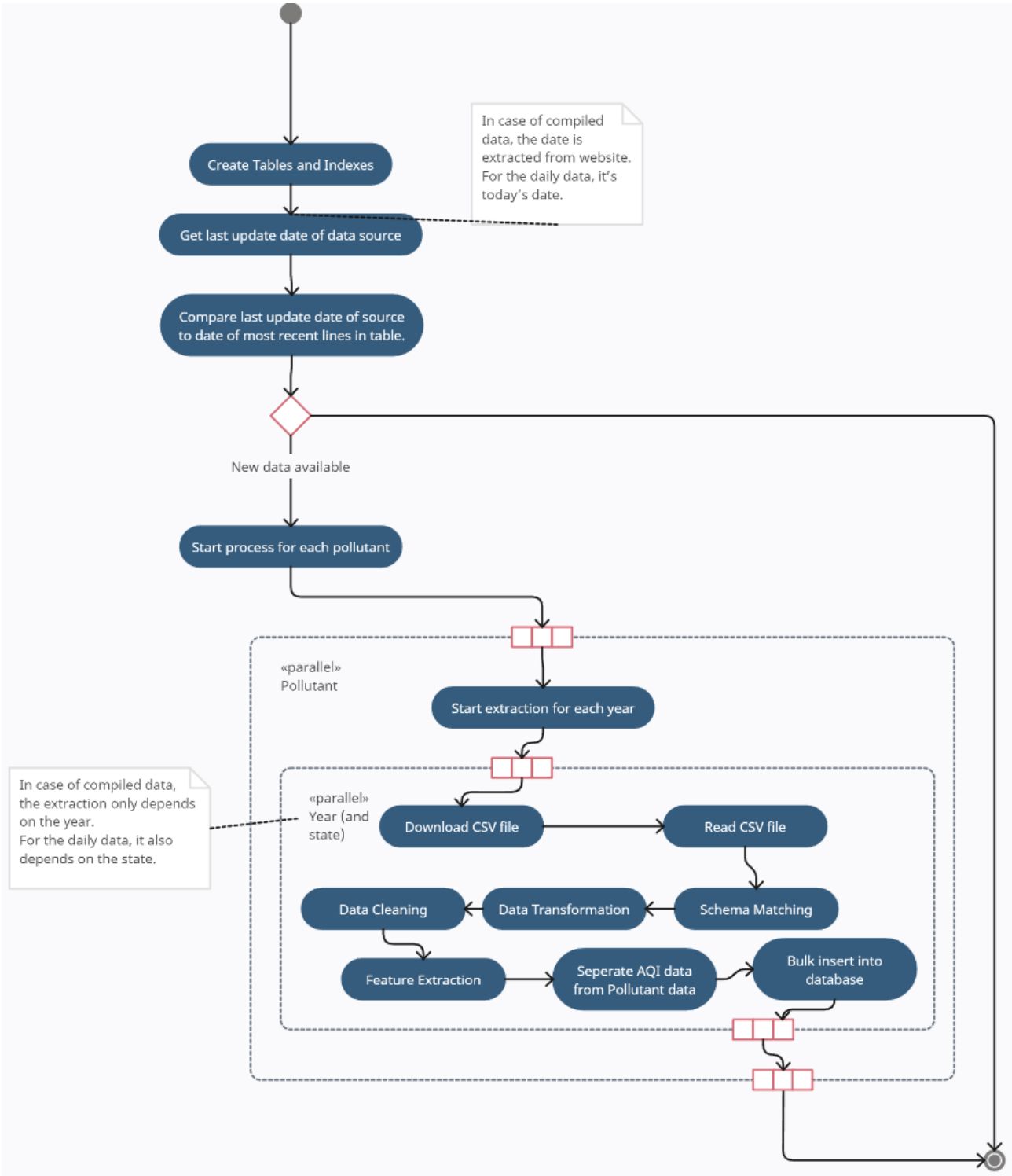

Figure II.5 – Activity Diagram for the ETL Process

Part III

Dashboard Design and Development

Chapter III

Dashbboard Design and Development

Summary

1 Dashboard Design 26

1.1 System Requirements 26

1.2 User Interface Design 28

2 Backend Development 32

2.1 Architecture 32

2.2 API Development 33

3 Frontend Development 34

3.1 Architecture 34

3.2 Feature Implementation 35

3.3 API Integration 39

3.4 Client-side Caching 39

4 Testing and Quality Assurance 39

4.1 API Unit Testing 39

4.2 Dashboard Unit Testing 40

4.3 Performance Testing 40

Introduction

The Dashboard Design and Development process encompasses a series of steps aimed at creating an efficient and user-friendly platform for visualizing and analyzing air quality data. This chapter provides an overview of the methodologies and practices employed in the design and

development of the air quality dashboard, highlighting the critical stages from conceptualization to testing. It covers initial design, backend infrastructure, UI development, and testing procedures to ensure high-quality, user-satisfactory results.

1 Dashboard Design

1.1 System Requirements

1.1.1 Functional Requirements

The functional requirements outlined in the figure III.1 include:

1. **View Levels of Concern Across the USA** The user can access a chart that illustrates the distribution of air quality levels across different regions of the United States. This chart provides insights into areas with varying pollution severity, helping users understand the overall air quality landscape.

2. **Interact with Map** The user can interact with an interactive map developed using the Syncfusion library. This map allows for various interactions to explore detailed air quality data.

    - **Zoom Map [«extends»]:** The user can zoom in and out of the map to focus on specific areas, enhancing the granularity of the data being viewed.
    - **Focus on State [«extends»]:** The user can select a state to focus on, which adjusts the map view to center on the chosen state and display its detailed air quality data.
    - **View State Details [«extends»]:** Upon focusing on a state, the user can view detailed information about the air quality distribution within that state, often broken down by county.
    - **View County Details [«extends»]:** The user can further drill down to view specific air quality details for individual counties within the selected state.

3. **View AQI Evolution Through Time** To achieve this use case, we implemented several functionalities:

    - **Date Picker with Play/Pause Button**: Users can visualize specific data by selecting a month and year, or they can click on the play button to see the AQI evolution over time. This dynamic feature allows for an interactive exploration of air quality trends.

- **Charts of AQI by Season, Year, Days, and Hours**:
  - **By Season**: These charts enable users to analyze seasonal variations in AQI, helping to identify patterns and understand how air quality fluctuates across different seasons.
  - **By Year**: Yearly charts provide a long-term perspective on air quality trends, allowing users to observe changes and trends over multiple years.
  - **By Days**: Daily charts help users find patterns and define specific days where AQI levels drop or rise, offering insights into daily air quality fluctuations.
  - **By Hours**: Hourly charts allow users to pinpoint specific hours where AQI levels change, providing a detailed view of how air quality varies throughout the day.

4. **Choose Pollutant Elements & View its Charts** The user can select specific pollutant elements such as CO, NO2, SO2, Ozone, PM10, and PM2.5. Based on the selection, the user can view charts that illustrate the distribution and trends of the chosen pollutants over time, offering detailed insights into each pollutant's impact on air quality.

5. **View General Information** The user can access general information about air quality data, including the average value of AQI, the number of records, and the number of observations. This provides a summary view of the overall air quality data, offering key metrics at a glance.

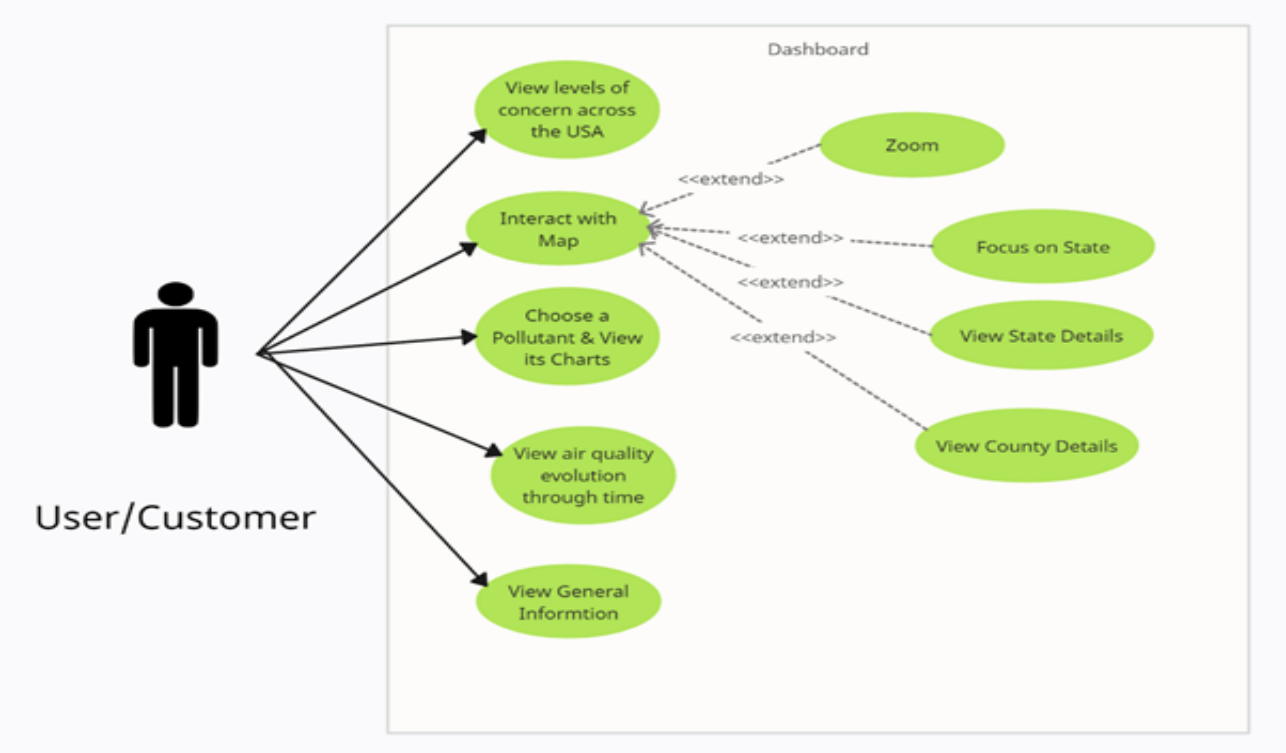

Figure III.1 – Use Case Diagram

1.1.2 Non-functional Requirements

In addition to the functional requirements, the dashboard design and development process also prioritized addressing key non-functional requirements, particularly modifiability and performance. Given the directive to utilize JavaScript, the adoption of Angular (specifically TypeScript) emerged as a natural choice that greatly facilitated the fulfillment of these requirements. Angular's modular and component-based architecture inherently fosters modifiability, enabling easier updates, enhancements, and maintenance of the application over time. Additionally, Angular's native optimization features, like its change detection strategies, combined with the option to implement client-side caching, enhance overall performance.

1.2 User Interface Design

1.2.1 Design Principles

In designing the user interface, several principles of simplicity were adhered to to ensure a user-friendly and intuitive experience by following the "Less is More" philosophy advocated by Dieter Rams, we emphasized clarity and minimalism to avoid overwhelming users with unnecessary

complexity [10]. Additionally, the principle of "Hick's Law" guided our approach, emphasizing the importance of reducing the number of choices presented to users to enhance decision-making and task completion [11]. By incorporating these principles, we aimed to create an interface that is both easy to navigate and efficient to use, ultimately enhancing user satisfaction and engagement.

1.2.2 User Flow

User flows are visual representations of how users navigate through various screens or steps to accomplish tasks, helping designers and developers understand user behavior and optimize the user experience. The user flows for this dashboard are shown in the figures III.2, III.4 and III.3.

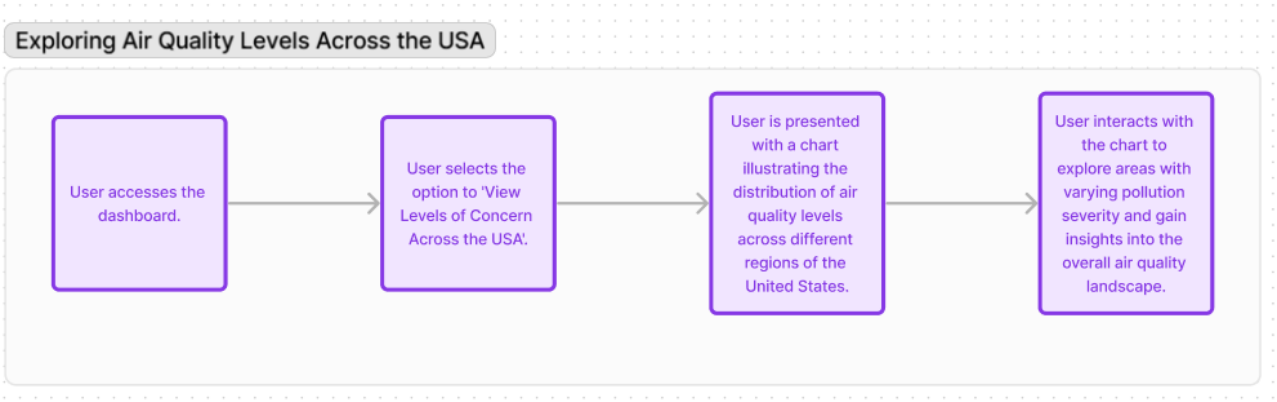

Figure III.2 – User Flow for exploring Air Quality Levels Across the USA

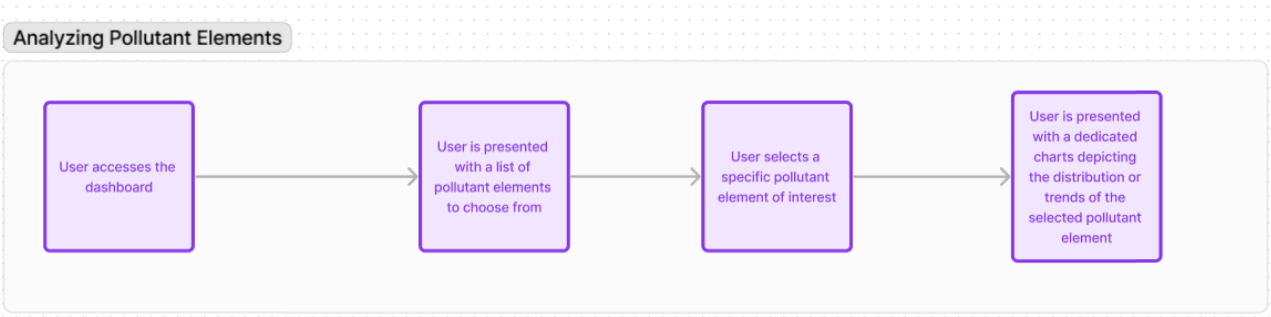

Figure III.3 – User flow for exploring Air Quality Trends Over Time

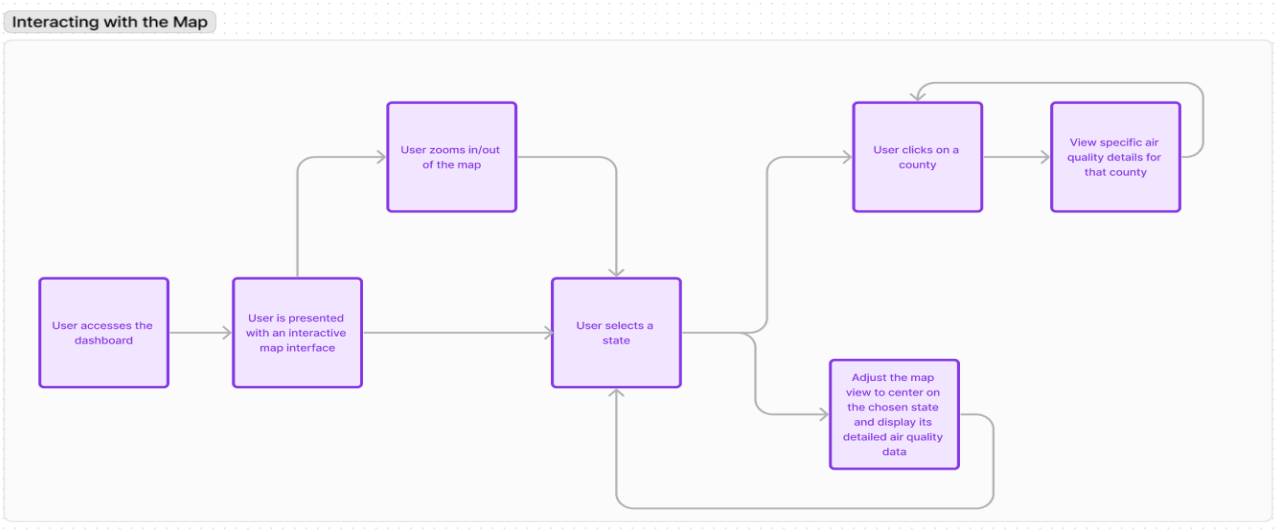

Figure III.4 – User flow for interacting with the Map

1.2.3 Website Prototype

The website prototype provides a hands-on preview of the final product, allowing stakeholders to give feedback before development begins. The prototype is shown in the figures III.5, III.6, III.7 and III.8.

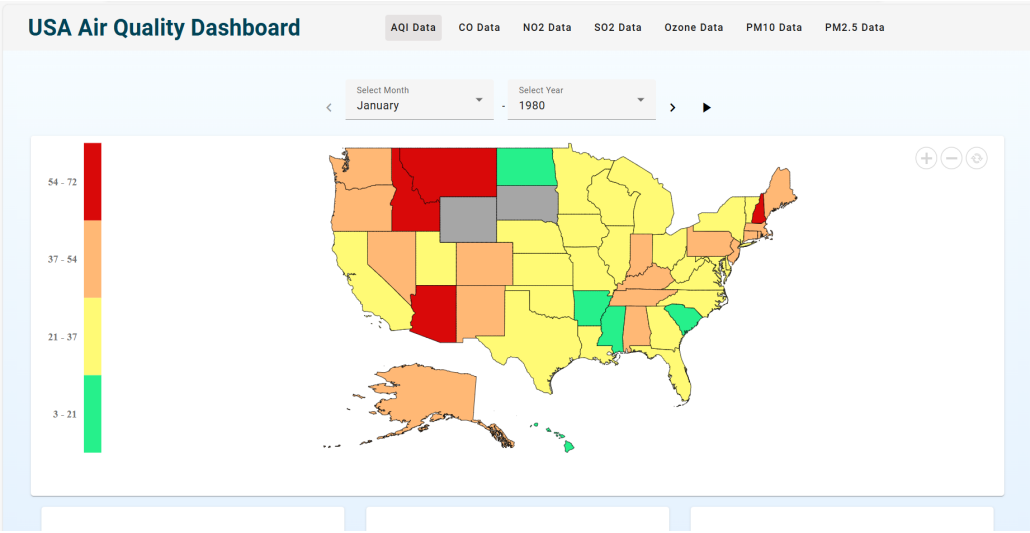

Figure III.5 – Description of website prototype 1

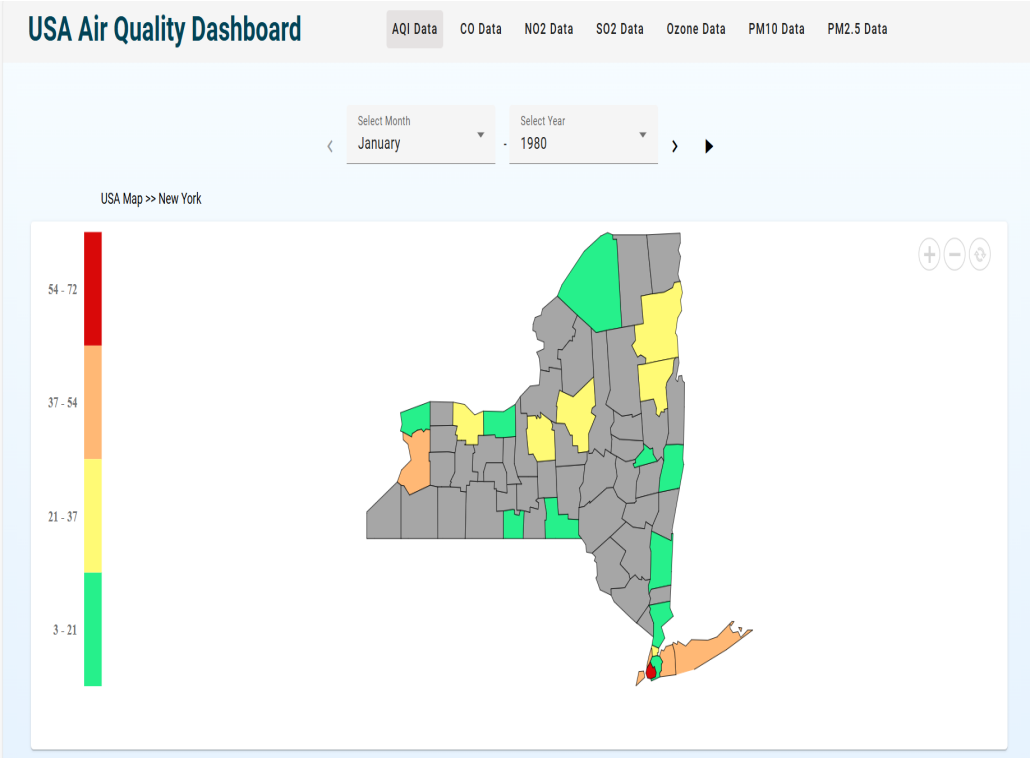

Figure III.6 – Description of website prototype 2

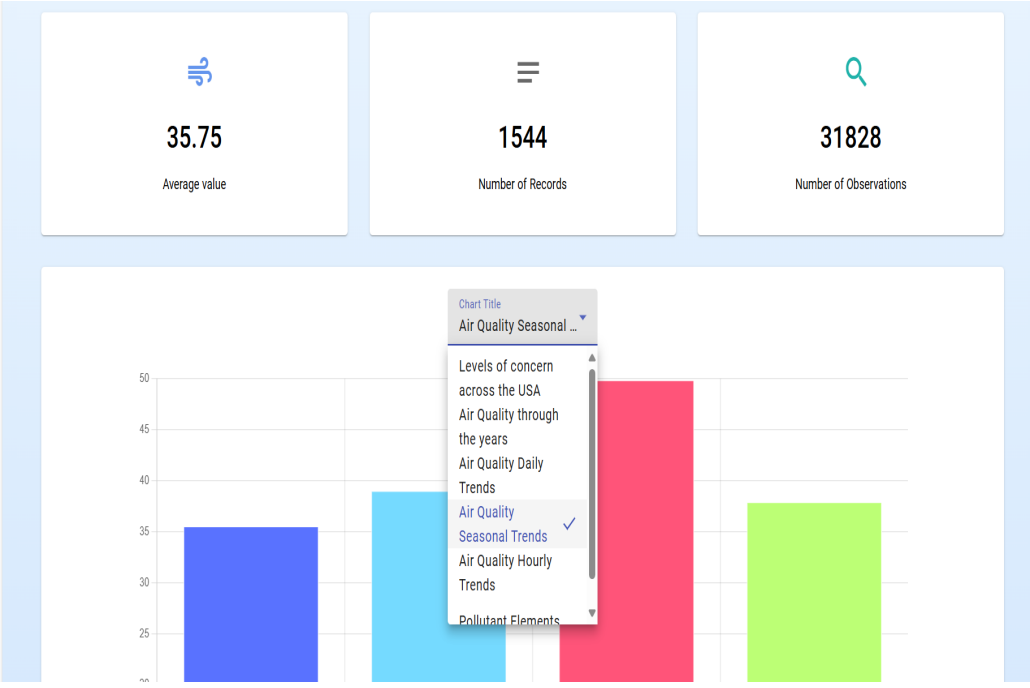

Figure III.7 – Description of website prototype 3

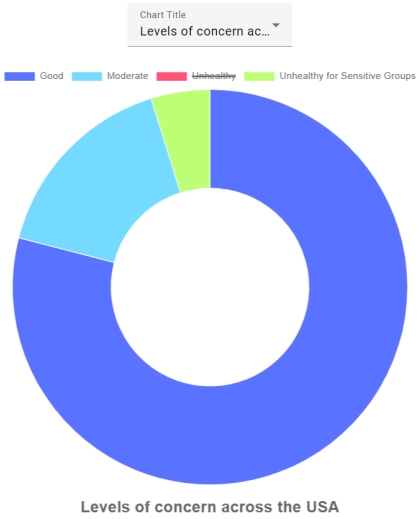

Figure III.8 – Description of website prototype 4

2 Backend Development

2.1 Architecture

The backend architecture was selected prioritizing simplicity and a clear seperation of concerns to streamline development and ensure smooth data retrieval and endpoint exposure, as shown in Figure III.9. Given the straightforward nature of the backend operations, which primarily involve fetching data from the database and exposing endpoints to serve aggregated data and statistics, Flask was chosen for the development of the endpoints.

Flask is known for its simplicity and flexibility, making it an ideal choice for projects with relatively uncomplicated backend requirements. With Flask, it was possible to quickly set up the backend infrastructure and define endpoints without unnecessary configurations.

Additionally, the decision to use Flask was influenced by the uniformity of our data models. Most the models, including AQI and the different pollutants, share similar attributes and structures, which made it straightforward to define them within Flask's framework. These models can inherit from an AbstractModel to ensure consistency and ease of maintenance across our data schema.

Furthermore, Flask's lightweight nature and minimalistic approach allowed us to keep our backend codebase concise and focused. This simplicity not only enhances development speed but also facilitates easier debugging and maintenance in the long run.

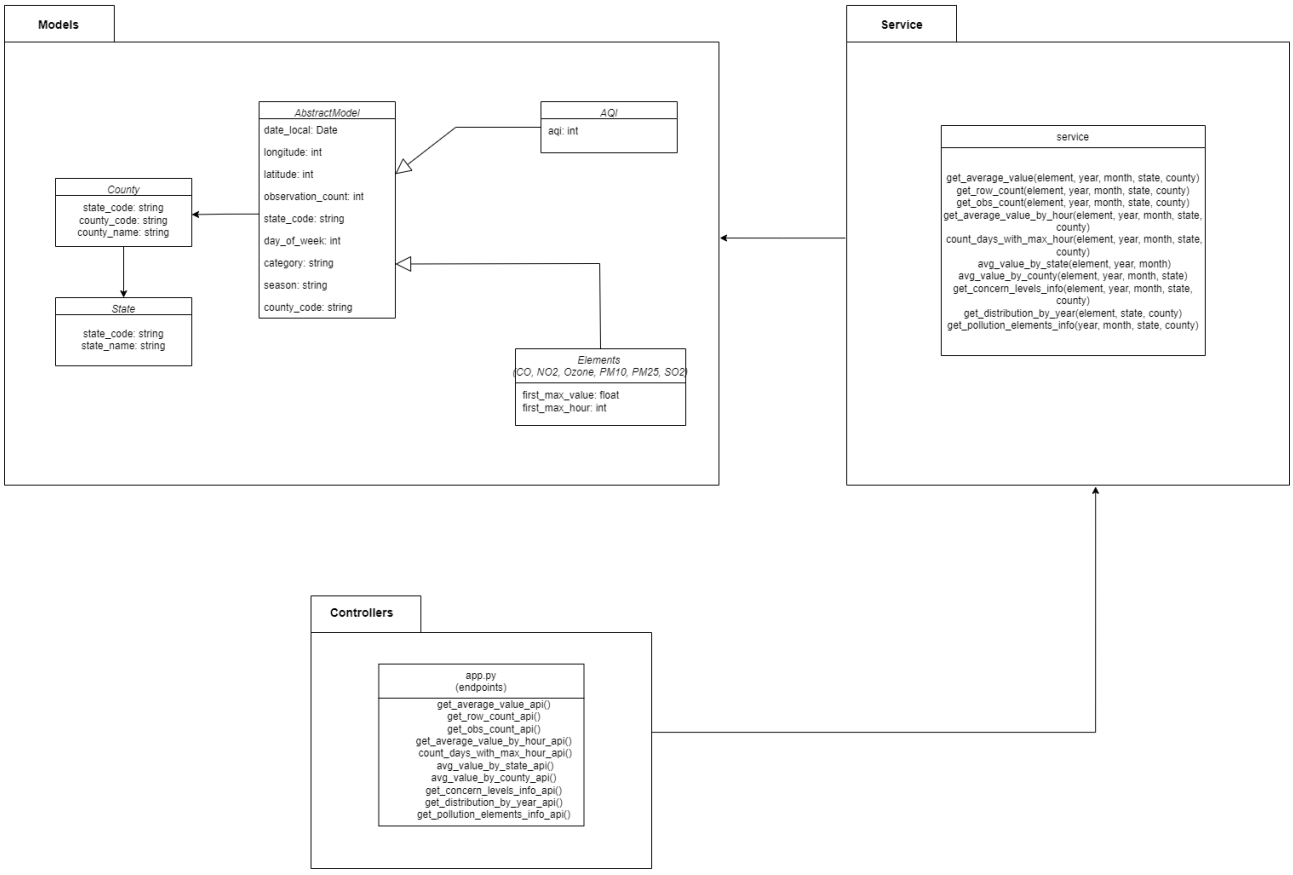

Figure III.9 – Backend's Architecture

2.2 API Development

API development for frontend-backend communication and data retrieval begins with the identification of endpoints required to serve the frontend application's needs. In the provided code, this involves defining routes in `app.py` that correspond to specific data queries or operations, such as retrieving average pollution values (`/average_value`), fetching row counts (`/count`), or obtaining air quality category information (`/air_quality_category`). These routes are designed to accept parameters from frontend requests, such as the pollutant element, year, month, state, and county, using the `request` object in Flask.

Once the routes are defined, the next step is to implement the corresponding controller logic, where service functions from `service.py` are invoked to interact with the database models and perform data retrieval or processing tasks. The service functions encapsulate the business logic and database operations, abstracting away the complexities of querying the database and performing calculations.

Error handling is implemented to handle cases where invalid data or no data is found for a given query, ensuring robustness and reliability in the API responses. Finally, the API responses are formatted as JSON objects and returned to the frontend application, enabling seamless communication between the frontend and backend components of the application.

3 Frontend Development

3.1 Architecture

The architecture of the frontend is designed with Angular components and services to ensure modularity, maintainability, and efficient communication between different parts of the application as shown in the figure III.10.

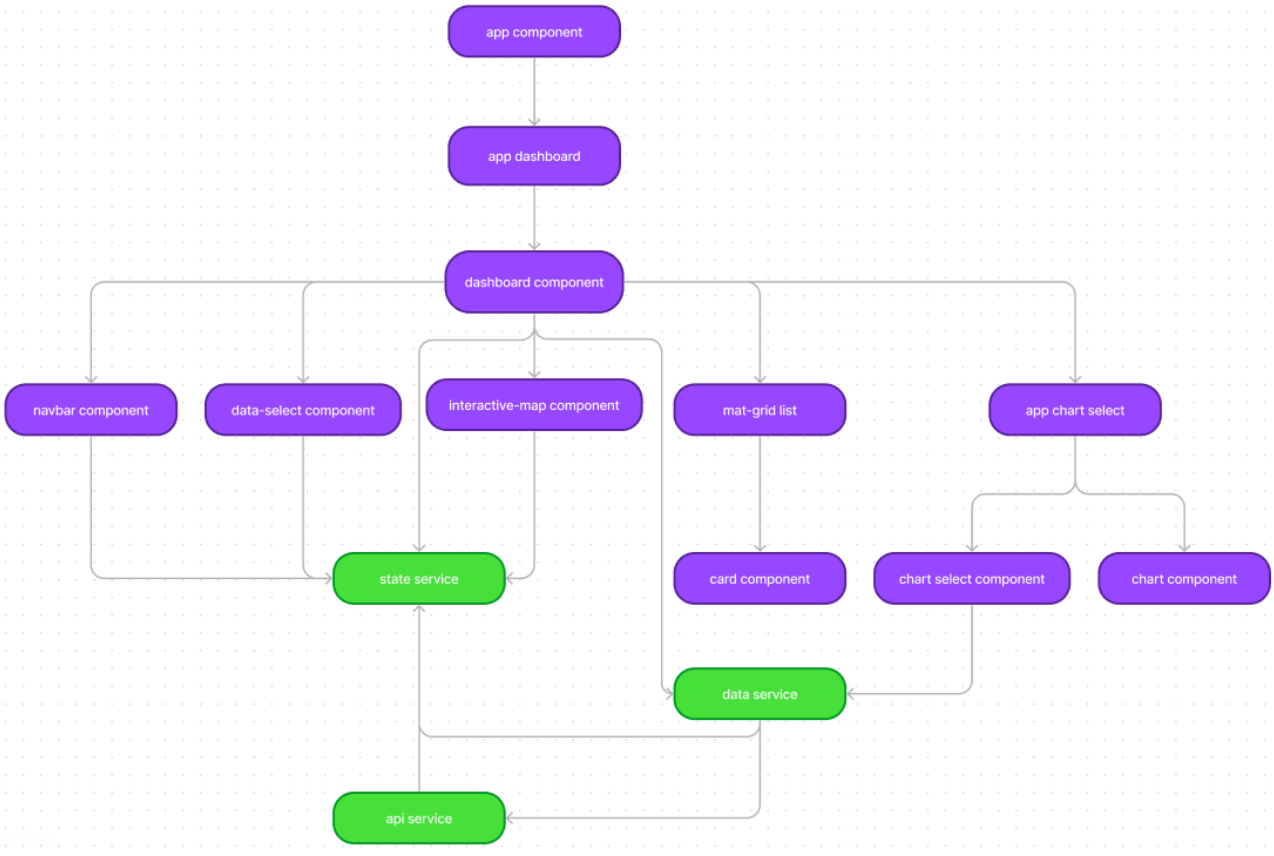

Figure III.10 – Frontend's Architecture

Angular was chosen for its use of components that encapsulate different sections of the user interface, such as the dashboard, navbar, date selector, interactive map, and charts. This

component-based approach allows for better organization of code and easier management of UI elements.

Services play a crucial role in handling data flow and business logic within the application. There are three main services as shown in the table III.1.

Table III.1 – Service Descriptions

Service	Description
ApiService	This service handles API requests, constructs URLs for API endpoints, and includes methods to make HTTP requests. It abstracts away the details of API communication, providing a clean interface for fetching data from the server.
DataService	Responsible for managing the data flow in the application, DataService interacts with the ApiService to fetch data based on selected elements. It processes and stores data fetched from the API as observables, making it available for consumption by other parts of the application.
StateService	Manages the state of selected elements in the application, such as chosen state, county, year, month, and pollution element. StateService provides methods to update selected elements and exposes observables to track changes, enabling components and services to react to these changes and trigger data fetching accordingly.

Additionally, an interceptor **CacheInterceptor** was used to implement client-side caching. This interceptor intercepts HTTP requests and responses to cache responses based on the request URL. By caching responses, we aim to improve performance and reduce redundant requests to the server, enhancing the overall user experience.

The architecture follows an event-driven approach, where changes and updates rely on a BehaviorSubject saved in the StateService. This ensures that components and services react to changes in selected elements and trigger appropriate actions, such as data fetching or UI updates.

3.2 Feature Implementation

3.2.1 Charts

1. **Doughnut Chart for Levels of Concern Across the USA**
   The doughnut chart provides a visual representation of the distribution of air quality

levels across different regions of the United States. This chart helps users understand the overall landscape of air quality concerns. Users can interact with the chart by eliminating any elements to compare the remaining ones more effectively, as shown in the figure **??**.

2. **Bar Chart for Air Quality Through the Years**
   The bar chart visualizes air quality data over multiple years, enabling users to track long-term trends and changes in pollution levels. This chart provides insights into how air quality has evolved, as shown in the figure III.11.

3. **Bar Chart for Air Quality Daily Trends**
   This bar chart displays the daily fluctuations in air quality, helping users identify patterns and trends on a day-to-day basis. It provides a detailed view of how air quality varies daily, as shown in the figure III.12.

4. **Bar Chart for Air Quality Seasonal Trends**
   The bar chart for seasonal trends allows users to analyze air quality variations across different seasons. This chart offers insights into how pollution levels fluctuate throughout the year, as shown in the figure III.13.

5. **Bar Chart for Air Quality Hourly Trends**
   This bar chart provides a detailed view of hourly air quality trends, enabling users to pinpoint specific hours when pollution levels change. It helps in understanding the variations in air quality throughout the day, as shown in the figure III.14.

6. **Pie Chart for Pollutant Elements**
   The pie chart illustrates the distribution of various pollutant elements, allowing users to compare their contributions to overall air quality. Users can interact with the chart by eliminating any elements to visualize a comparison between the remaining pollutants, as shown in the figure III.15.

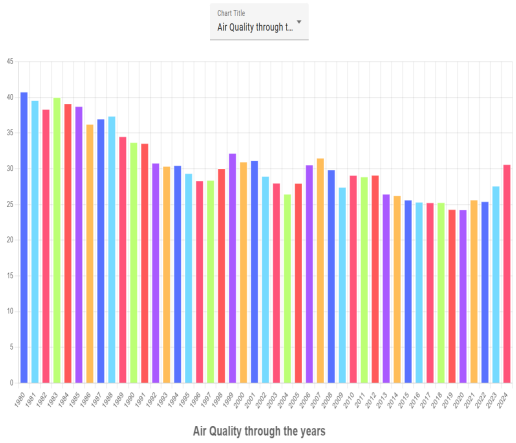

Figure III.11 – Doughnut Chart for Levels of Concern Across the USA

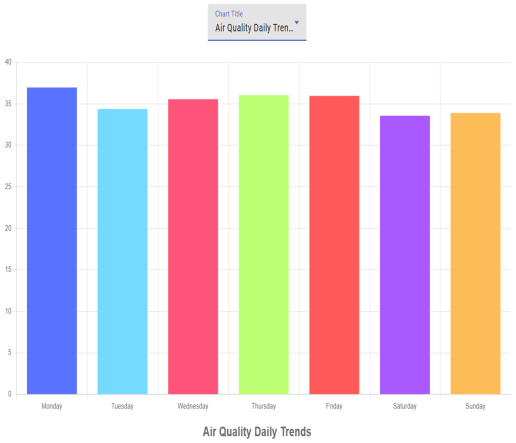

Figure III.12 – Bar Chart for Air Quality Through the Years

Figure III.13 – Bar Chart for Air Quality Daily Trends

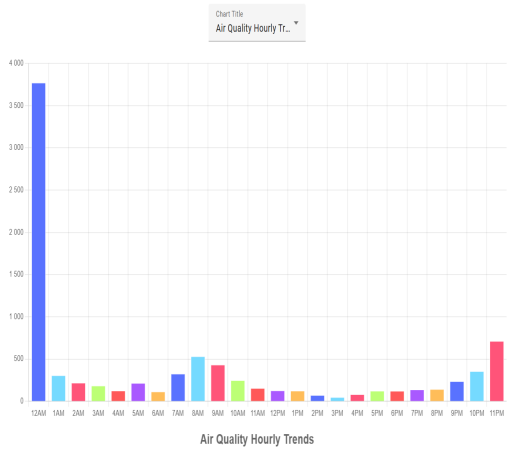

Figure III.14 – Bar Chart for Air Quality Seasonal Trends

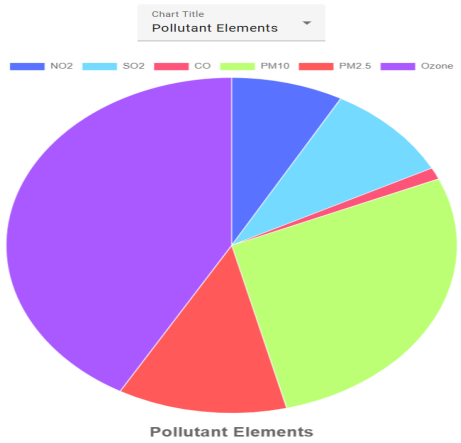

Figure III.15 – Bar Chart for Air Quality Hourly Trends

3.2.2 Map Integration

Libraries

For integrating maps into our dashboard, Syncfusion library was used. Syncfusion provides robust and customizable components and services for Angular, including tooltips, legends and selection highlights. This choice was driven by Syncfusion's ease of integration and comprehensive documentation, which facilitated the development process.

Interactions

The map integration offers several interaction capabilities to enhance user experience. Users can zoom in and out of the map to focus on specific areas, improving data granularity. The map allows users to select and focus on individual states, centering the map view and displaying

detailed air quality data for the chosen state. Further, users can drill down to view air quality details for specific counties within a state, as shown in the figures III.5 and III.6. The counties' JSON data was manually incorporated into an existing USA map JSON file due to the absence of a comprehensive pre-built JSON file.

3.2.3 Timeline Component

The timeline component includes a date picker with a play/pause button, allowing users to visualize the evolution of the AQI over time. By selecting specific dates or ranges, users can view air quality data for those periods. The play/pause functionality enables an animated progression through the data, providing a dynamic and interactive way to observe changes in air quality. This feature is particularly useful for identifying trends and patterns over various time frames as shown in the figure III.5.

3.3 API Integration

Integrating the frontend with backend services and APIs was achieved using the `ApiService`, as discussed in 2.2. The `ApiService` provides a clean interface for data retrieval, ensuring efficient communication between the frontend and backend components.

3.4 Client-side Caching

To improve performance and reduce the load on the backend, we implemented caching mechanisms on the client side. This is crucial for achieving better performance, which is a key non-functional requirement as discussed in 1.1.2.

We utilized a cache interceptor to handle caching of HTTP responses. This interceptor stores responses in a cache based on the request URL, allowing subsequent requests to retrieve data directly from the cache if available. This reduces redundant requests to the server and enhances the overall user experience by providing faster response times. The implementation of this caching mechanism is outlined in 3.1.

4 Testing and Quality Assurance

4.1 API Unit Testing

The unit tests in `tests.py` aim to ensure the correctness and robustness of the API endpoints in the Flask application. Each test function is dedicated to a specific endpoint and follows a

similar structure. The `test_endpoint` function serves as a generic test function that takes a client, an endpoint, and a custom assertion function as parameters. It iterates over a predefined set of parameters and sends requests to the endpoint with these parameters, asserting that the response status code is 200 and that the response data meets the criteria specified by the custom assertion function. Additionally, it tests the behavior of the endpoint when provided with incorrect parameters, asserting that it returns a 400 status code and an error message indicating invalid input.

Individual test functions are then defined for each endpoint, such as `test_avg_value`, `test_row_count`, `test_avg_by_day`, etc. Each of these functions calls the `test_endpoint` function with the respective endpoint and a custom assertion specific to the data returned by that endpoint. For example, `test_avg_by_day` asserts that the response data contains average pollution values for all seven days of the week.

The `test_heartbeat` function ensures that the heartbeat endpoint ("/") returns the expected string indicating that the server is running.

The `conftest.py` file contains fixture functions that provide a Flask test client and CLI runner for the tests, configured for testing by setting the `TESTING` flag to `True`. These fixtures are utilized by the test functions to interact with the Flask application and execute the API requests.

4.2 Dashboard Unit Testing

In unit testing the dashboard components, we aim to ensure the functionality and reliability of each individual component within the dashboard interface. For this purpose, unit tests are created for every component, including the card, chart, chart-select, dashboard, date-select, interactive-map, and navbar components. Each unit test focuses on specific aspects of the component's behavior and functionality. For example, tests for the card component may verify that the card renders correctly with the expected content and styling. Similarly, tests for the chart component may check that the chart renders data accurately and responds appropriately to user interactions. The unit tests are designed to cover various scenarios and edge cases to validate the behavior of the components under different conditions. Mock data and stubs may be used to simulate different states and inputs for testing purposes.

4.3 Performance Testing

Performance testing is vital to ensure that the dashboard functions correctly and meets performance standards. To evaluate its performance, we utilized various testing methodologies,

including website performance monitoring tools like Pingdom [12], as shown in Figure III.16. The results from these tests were generally favorable, indicating that the dashboard performs well under normal conditions. However, a notable observation was the slightly longer initial load time, attributed to the necessity of accessing the backend and subsequently the database for the first render. Nevertheless, this initial delay is mitigated by client-side caching, ensuring that subsequent requests for the same data do not require repeated backend and database accesses, thus improving load times for subsequent renders.

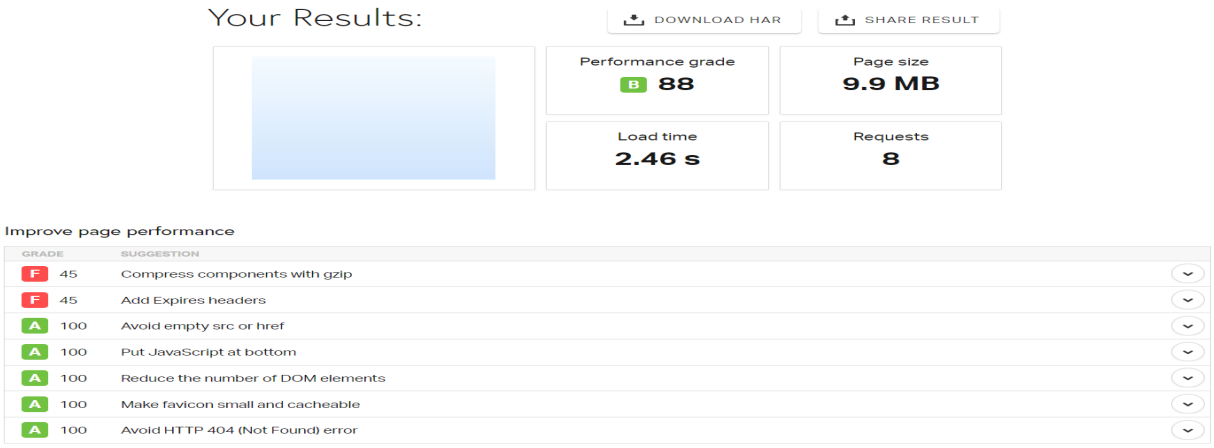

Figure III.16 – Performance Results

Conclusion

The development of the dashboard involves careful design considerations, robust backend architecture, efficient frontend implementation, and thorough testing procedures. The dashboard's design prioritized user experience and accessibility, ensuring intuitive navigation and meaningful data visualization. Angular components and services facilitated modularity and maintainability in the frontend architecture, while Flask provided a straightforward yet powerful backend framework for API development. Unit and performance testing were conducted to validate the functionality, reliability, and performance of both the frontend interface and backend services. Through this comprehensive approach, the dashboard has evolved into a tool for monitoring and analyzing air quality data, positioned to deliver valuable insights while maintaining optimal performance and usability.

Part IV

Deployment

Chapter IV

Deployment

Summary

1 Continuous Integration/Continuous Deployment (CI/CD) 43

2 Deployment Pipeline Setup 45

2.1 Version Control Systems Used 45

2.2 Automated Deployment Tools 45

3 Monitoring and Maintenance 49

3.1 Performance Monitoring and Alerting 49

3.2 Regular Maintenance Tasks 50

4 Scalability and Future Enhancements 51

Introduction

DevOps is a software development approach that combines development (Dev) and operations (Ops) teams, emphasizing collaboration and automation throughout the software development lifecycle. It aims to improve deployment frequency, lead time for changes, and overall reliability of software releases. This chapter provides an overview of the deployment process for the air quality monitoring and visualization system, emphasizing DevOps practices and the CI/CD pipeline. It covers pipeline setup, monitoring, maintenance, scalability strategies, and future enhancements.

1 Continuous Integration/Continuous Deployment (CI/CD)

The Continuous Integration and Continuous Deployment (CI/CD) process is an integral part of the deployment strategy for the air quality monitoring and visualization system. Leveraging

AWS as the cloud platform of choice, the CI/CD approach ensures that code changes are automatically tested and deployed, enabling rapid iteration and maintaining high software quality. The CI/CD process starts with code commits to the GitHub repositories **aqi-quality-apis** for the backend and **aqi-dashboard** for the frontend. Each commit triggers AWS CodePipeline, which orchestrates the entire CI/CD workflow, providing a streamlined architecture.

For the backend, AWS CodePipeline uses AWS CodeBuild to compile the Flask application, run tests, and create deployment artifacts. Successful builds are deployed to AWS Elastic Beanstalk, which simplifies the management of the application environment by handling provisioning, scaling, and monitoring. AWS Elastic Beanstalk integrates seamlessly with AWS RDS, ensuring efficient database management.

For the frontend, AWS CodePipeline also uses AWS CodeBuild to compile the Angular application, run tests, and produce build artifacts. These artifacts are then deployed to Amazon S3, where they are hosted as a static website. Amazon S3 offers a reliable and scalable solution for static web hosting, ensuring that the frontend is quickly and efficiently delivered to users.

This automated CI/CD pipeline ensures consistent integration and deployment of both backend and frontend components, minimizing manual intervention and reducing the risk of errors. By utilizing AWS services, the process achieves high reliability, scalability, and maintainability. The CI/CD workflow supports continuous delivery of new features and updates, ensuring the system remains up-to-date and responsive to user needs. The CI/CD workflow is illustrated in the figure IV.1, providing a visual overview of the deployment pipeline and demonstrating the efficiency of the AWS-based architecture.

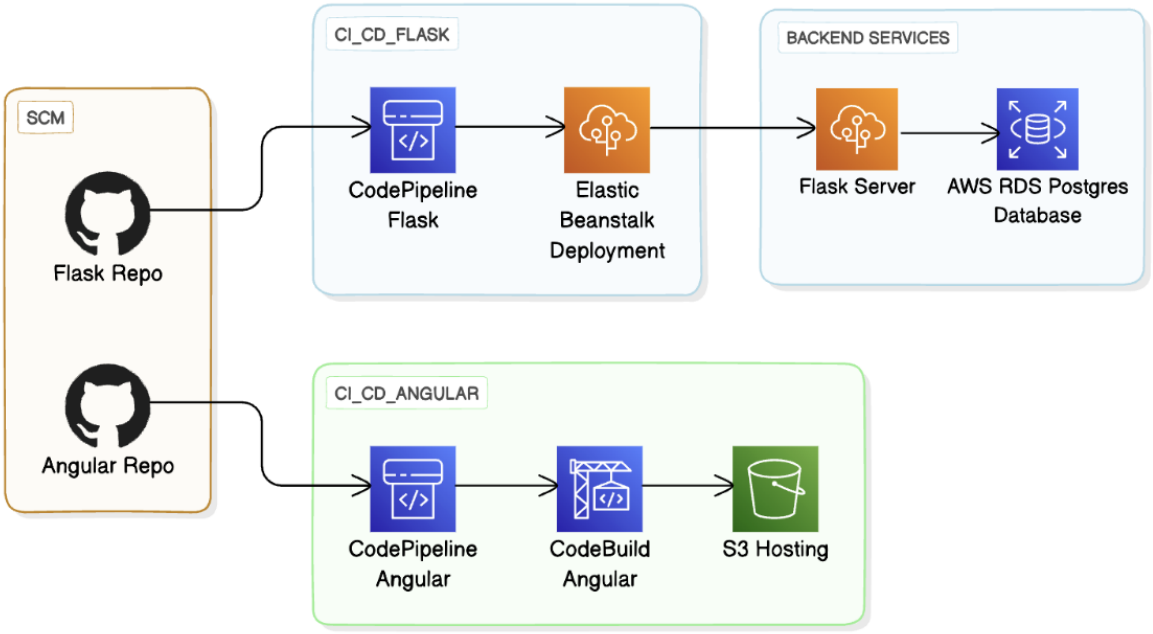

Figure IV.1 – CI/CD Workflow for the Air Quality Monitoring and Visualization System

2 Deployment Pipeline Setup

2.1 Version Control Systems Used

Version control systems play a pivotal role in the deployment pipeline setup for the air quality monitoring and visualization system. GitHub serves as the primary version control platform, facilitating collaborative development and ensuring version history integrity. The repositories **aqi-quality-apis** for backend APIs and **aqi-dashboard** for the frontend dashboard are meticulously maintained on GitHub. This enables developers to track changes, manage branches, and coordinate code merges efficiently. GitHub's robust features, including pull requests, issue tracking, and code reviews, streamline the development process and promote code quality. By utilizing GitHub as the version control system of choice, the deployment pipeline setup ensures transparency, traceability, and collaboration across all stages of development and deployment.

2.2 Automated Deployment Tools

Automated deployment tools are essential components of the deployment pipeline setup for the air quality monitoring and visualization system. Leveraging AWS services, specifically AWS CodePipeline and AWS CodeBuild, ensures a streamlined and efficient deployment process.

AWS CodePipeline serves as the backbone of the deployment workflow, orchestrating the entire process from source code changes to deployment. It provides a visual representation of the workflow and allows for the automation of build, test, and deployment stages.

For the backend APIs, AWS CodeBuild compiles, tests, and packages the Flask application code into deployable artifacts. These artifacts are seamlessly deployed to AWS Elastic Beanstalk, a service that simplifies the provisioning, scaling, and management of the server environment. AWS Elastic Beanstalk integrates seamlessly with AWS RDS for efficient database management, ensuring a robust backend infrastructure.

On the other hand, for the frontend dashboard, AWS CodeBuild similarly compiles, tests, and packages the Angular application code into deployable artifacts. These artifacts are then deployed to Amazon S3, which offers reliable and scalable static web hosting.

To illustrate, the figure IV.2 depicts the CI/CD pipeline for the backend APIs, showcasing the stages from code commits to deployment to AWS Elastic Beanstalk.

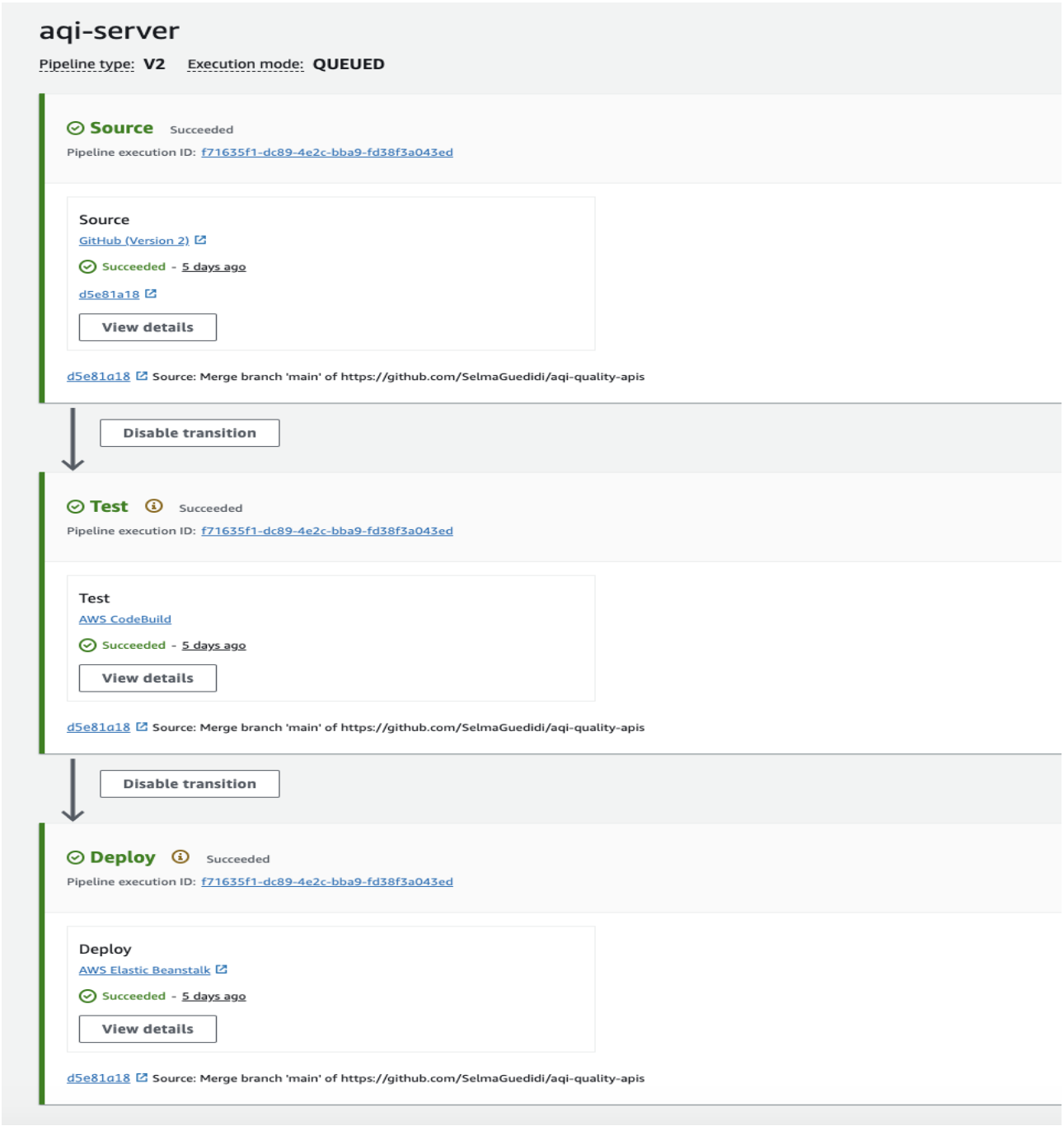

Figure IV.2 – CI/CD Workflow for the Flask server

Similarly, the figure IV.3 represents the CI/CD pipeline for the frontend dashboard, highlighting the workflow from code commits to deployment to Amazon S3.

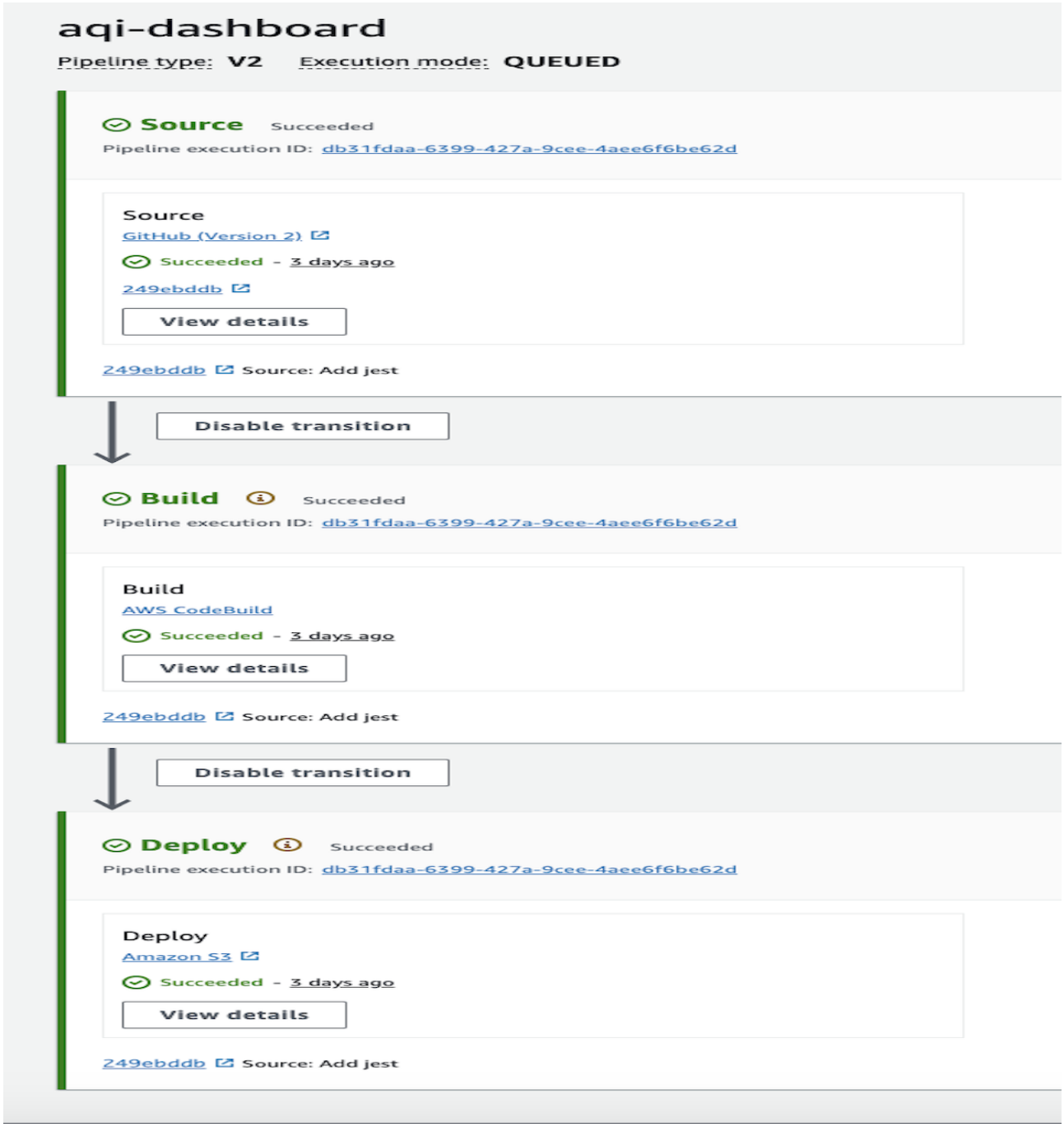

Figure IV.3 – CI/CD Workflow for the Angular Application

These automated deployment tools and AWS services ensures that the deployment pipeline setup achieves efficiency, repeatability, and scalability, enabling rapid iteration and deployment of new features and updates while maintaining high software quality and reliability.

3 Monitoring and Maintenance

3.1 Performance Monitoring and Alerting

Efficient performance monitoring and alerting mechanisms are integral to the upkeep of the air quality monitoring and visualization system. Leveraging AWS services such as Amazon CloudWatch, Amazon CloudTrail, and native logging features, the system ensures comprehensive performance monitoring and proactive alerting.

Amazon CloudWatch serves as a central hub for monitoring various components of the system in real-time, including server instances, databases, and application logs. Through CloudWatch, administrators can track critical performance metrics such as CPU utilization, memory usage, and network traffic. CloudWatch Alarms can be configured to send notifications when these metrics exceed predefined thresholds, enabling immediate response to performance anomalies. Additionally, dashboards are set up in CloudWatch to visualize the Flask server environment health, providing administrators with insightful data for monitoring and troubleshooting as shown in the figure IV.4.

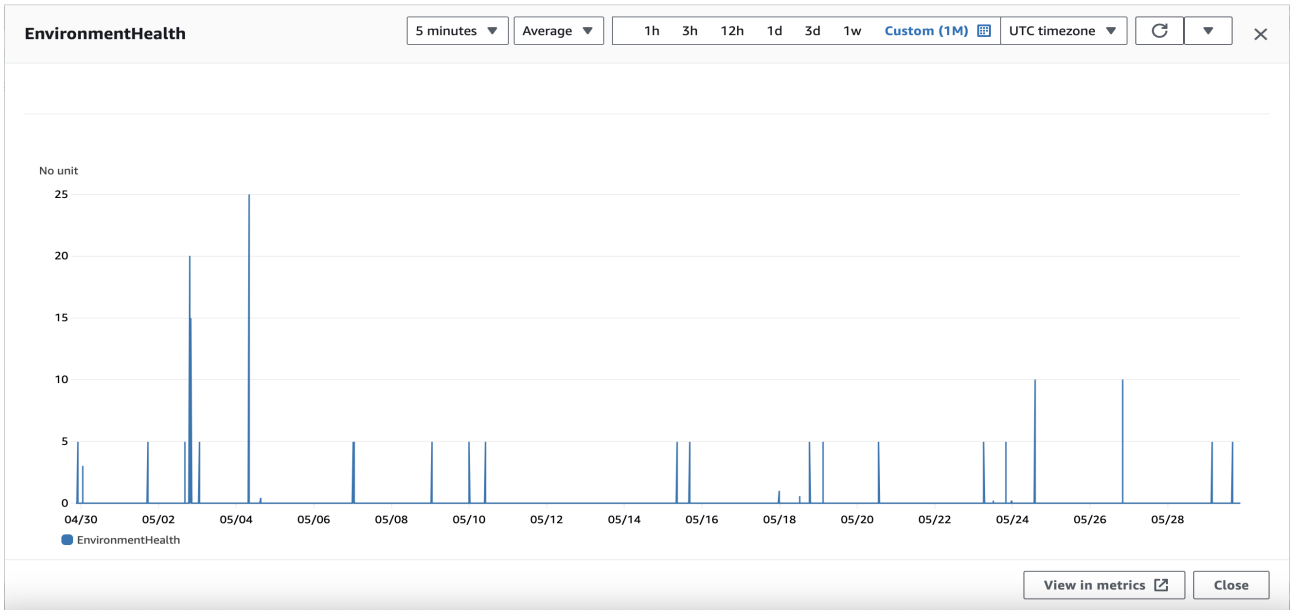

Figure IV.4 – CloudWatch dashboard for flask server environment health

Furthermore, the system utilizes Amazon S3 to host the frontend dashboard artifacts, with logging enabled for the S3 bucket. These logs, including server access logging, provide valuable insights into website traffic, access patterns, and response times, facilitating performance opti-

mization and user experience enhancement. Administrators can also utilize S3 storage metrics available in CloudWatch to monitor the usage and performance of the aqi-dashboard bucket, ensuring optimal storage management and availability as shown in the figure IV.5.

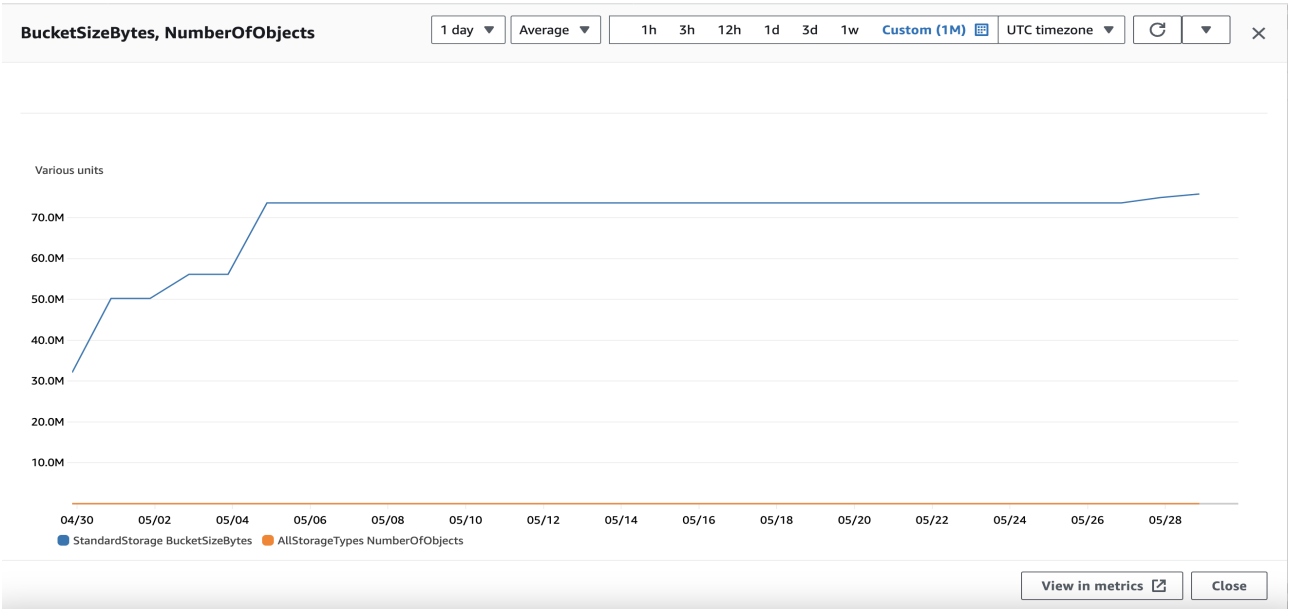

Figure IV.5 – CloudWatch dashboard for aqi-dashboard bucket metrics

Furthermore, Amazon CloudTrail logs API calls and actions within the AWS environment, providing a comprehensive audit trail for system activities. Monitoring CloudTrail logs enables administrators to track changes, detect unauthorized access attempts, and ensure compliance with security policies.

Using CloudWatch, CloudTrail, and custom logging features, the air quality monitoring and visualization system ensures proactive performance monitoring and alerting. This proactive approach enables administrators to identify and address performance issues promptly, ensuring optimal system performance and user satisfaction.

3.2 Regular Maintenance Tasks

Regular Maintenance tasks encompass various aspects of system management and optimization to mitigate risks, address issues, and enhance user experience.

1. **Database Maintenance**: Regular database maintenance is essential to ensure data integrity and efficient query execution. This includes tasks such as database backups,

performance optimization, and index maintenance to keep the database system running smoothly.

2. **Server Updates and Patching**: Keeping server instances up-to-date with the latest security patches and software updates is crucial for mitigating security vulnerabilities and ensuring system stability. Regular updates help maintain system integrity and protect against emerging threats.

3. **Log Rotation and Management**: Effective log rotation and management practices are necessary to manage storage space and maintain log data for compliance and troubleshooting purposes. Regular rotation and archival of logs help ensure that log data remains accessible and manageable.

4. **Security Audits and Compliance Checks**: Conducting regular security audits and compliance checks helps identify and address potential security vulnerabilities and ensures adherence to regulatory requirements. By proactively addressing security concerns, administrators can mitigate risks and protect sensitive data.

4 Scalability and Future Enhancements

Adopting scalable infrastructure strategies ensures that the system can handle increased data volume, user traffic, and computational requirements without compromising performance or reliability. Some key strategies for scaling the infrastructure include:

1. **Horizontal Scaling**: Implementing horizontal scaling by adding more server instances or utilizing auto-scaling groups allows the system to dynamically adjust resources based on workload demands. This ensures that the system can handle fluctuations in traffic and workload efficiently.

2. **Database Sharding**: Sharding the database enables distributing data across multiple servers, allowing for increased data throughput and reduced query response times. By partitioning data horizontally, database sharding enhances scalability and improves overall database performance.

3. **Caching Mechanisms**: Utilizing caching mechanisms such as Amazon ElastiCache can significantly improve system performance by reducing latency and offloading database load. Caching frequently accessed data or query results helps alleviate the burden on the database and improves overall system responsiveness.

Conclusion

The deployment section of the air quality monitoring and visualization system highlights the critical processes and strategies employed to ensure the seamless integration and continuous delivery of both backend APIs and frontend components. Leveraging AWS services such as CodePipeline, CodeBuild, Elastic Beanstalk, and S3, the deployment pipeline setup achieves efficiency, repeatability, and scalability, enabling rapid iteration and deployment of new features and updates. The automated CI/CD workflow, coupled with robust monitoring and maintenance practices, ensures the system's reliability, performance, and security. Looking ahead, scalability strategies and potential future enhancements are poised to further enhance the system's capabilities, empowering users with actionable insights and contributing to improved environmental awareness and public health outcomes. By prioritizing scalability, reliability, and innovation, the deployment section sets a solid foundation for the continued evolution and success of the air quality monitoring and visualization system.

Conclusion and Perspectives

The goal of this project was to visualize air quality trends in the United States over the years to support analysis and decision-making. The process included several key phases: data processing, dashboard development and automated deployment.

First of all, data extraction, transformation, and loading (ETL) of air quality data was done. Tools such as Selenium, BeautifulSoup, and HTTP requests were employed to collect data from various sources. This data was subsequently cleaned and organized, ensuring its suitability for analysis. It was stored in a PostgreSQL database on AWS RDS.

Furthermore, an interactive dashboard was developed using Angular and Flask, enabling dynamic visualization of the data. This platform allows users to view air quality for different pollutants like CO, SO2, NO2, Ozone, PM2.5, and PM10, as well as the Air Quality Index (AQI). Features include timeline visualizations, an interactive map for selecting states and counties, and charts that display air quality trends and distributions over time and by hour.

Finally, the deployment of the dashboard was automated with a CI/CD pipeline using AWS CodePipeline and AWS CodeBuild. This setup ensured that any code changes were automatically tested, built and deployed, keeping the application updated. The backend, developed in Flask, was hosted on AWS Elastic Beanstalk, while the frontend, created with Angular, was deployed to an Amazon S3 bucket.

Perspectives

In future work, it may be useful to consider these paths:

- **Short-Term:** More data sources and visualizations could be added to expand the functionality of the dashboard. Additionally, new visualization tools such as heatmaps and trend analysis charts could be developed.

- **Medium-Term:** Data could be organized into views to simplify access and improve performance. This includes creating aggregated views that compile data across different time periods and geographic areas, enabling faster queries and more efficient data management. These views support more complex analysis and enable faster loading times for interactive elements on the dashboard.

- **Long-Term:** Machine learning techniques could be explored to study the effects of pollution on health, with the aim of linking air quality conditions to health outcomes. Pre-

dictive models could be developed to predict pollution levels and potential health risks, providing early warning to both policymakers and the public. Additionally, data analytics could be used to identify long-term trends and associations between air quality and population health, which could contribute to more targeted and effective public health interventions.

References

[1] US EPA, OAR. Aqi breakpoints. https://aqs.epa.gov/aqsweb/documents/codetables/aqi_breakpoints.html. Accessed on 2024-05-25 21:50:47. v, 19

[2] Sean Peek. Scrum methodology. https://www.businessnewsdaily.com/4987-what-is-agile-scrum-methodology.html. Accessed on 2024-05-28 12:39:48. 7

[3] Qin Hanlin, Jin Xianzhen, and Zhang Xianrong. *Research on extract, transform, and load (etl) in land and resources star schema data warehouse*. 2012 Fifth International Symposium on Computational Intelligence and Design (2012). 12

[4] US EPA, OAR. Criteria air pollutants. https://www.epa.gov/criteria-air-pollutants. Accessed on 2024-05-24 16:28:20. 13

[5] US EPA, OAR. Air data basic information. https://www.epa.gov/outdoor-air-quality-data/air-data-basic-information. Accessed on 2024-05-24 16:56:42. 15

[6] Manel Souibgui, Faten Atigui, Saloua Zammali, Samira Si-Said Cherfi, and Sadok Ben Yahia. Data quality assessment in etl processes: Initial investigations. , **159**, pages 676–687. Elsevier (2019). 15

[7] Chaimaa Lotfi, Swetha Srinivasan, Myriam Ertz, and Imen Latrous. Web scraping techniques and applications: A literature review. In *SCRS Conference Proceedings on Intelligent Systems*, pages 381–394 (2021). 16

[8] Moaiad Khder. *Web scraping or web crawling: State of art, techniques, approaches and application*. International Journal of Advances in Soft Computing and its Applications **13**, 145–168 (2021). 16

[9] Gianina Mihai (Rizescu). *Comparison between relational and nosql databases*. Annals of Dunarea de Jos University of Galati Fascicle I Economics and Applied Informatics **XXVI**, 38–42 (2020). 20

[10] D. Rams. *Less but better: Dieter rams's 10 principles of good design*. (2009). 29

[11] W. E. Hick. *On the rate of gain of information*. Quarterly Journal of Experimental Psychology **4**(1), 11–26 (1952). 29

[12] Pingdom AB. Pingdom website monitoring software. 41