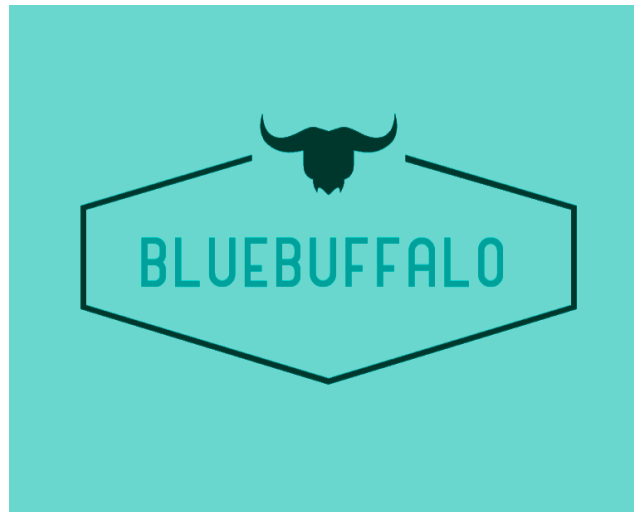# COSC 310: Software Engineering

MILESTONE 5: Final Report & Delivery

Team BLUE BUFFALO

Taii Hirano
Leo Kaiya
Putri Leksono
Karen Masuda
Joy Umejiego

Contents

Video Walkthrough

Link To A Video

YouTube Link

https://youtu.be/cAChrtcvIeQ

==

Update the Requirement Document

Part I Description of the Software

Brief Description of the Software

Our software aims to create an educational dashboard offering comprehensive insights into the air quality of U.S. cities, meticulously organized by date. Through visually engaging maps, charts, and plots, students and educators alike can explore and analyze real-time and historical air quality data. This platform serves as an invaluable educational resource, enabling learners to deepen their understanding of environmental science, pollution, and its impacts on public health. With accessible and interactive features, our software empowers students to engage actively in learning about environmental issues and inspires them to become advocates for positive change in their communities.

===

Part II User Groups and Example Scenarios

Visitors/Users

Scenario: An individual concerned about air quality visits the dashboard to explore current pollution levels across different cities. They navigate through various pages to understand trends and comparisons between locations. Finding the information useful, they decide to create an account to bookmark favourite visualizations and receive updates on air quality.

Administrators/Admins

Scenario: The administrator of the dashboard logs in to manage the platform's content and user interactions. They review user comments, ensuring discussions remain respectful and relevant. The admin also monitors data sources to maintain accuracy and reliability, making necessary adjustments as needed to uphold the platform's quality standards.

===

Part III Final List of Requirements

Authentication and User Management

- Authenticate users securely during login processes using their credentials.
- Provide functionality for users to register by submitting necessary information.
- Allow users to modify their account information and request password resets securely.
- Authorize access to features based on user roles (e.g., regular user, admin).
- Send email notifications to users upon successful account creation.

Data Visualization and Interaction

- Retrieve relevant data for visualizations, such as city rankings, pollution maps, and graph visualizations of pollution.
- Generate visualizations, including bar charts, scatterplots, pie charts, and line graphs, to represent air quality data effectively.
- Allow users to interact with visualizations, adjust dates using a calendar, and view historical data on the dashboard.
- Generate a US map with pinpoints indicating locations with pollution, retrieve location information and air quality index (AQI) data from the database to populate the map.
- Implement colour-coding to represent pollution levels, and enable users to interact with map pins for detailed information.
- Predict and display the future AQI based on the existing data.

System Performance, Security, and Usability

- Ensure the dashboard loads quickly and responds efficiently to user interactions, and optimize data retrieval, and processing to minimize latency.
- Implement robust security measures to protect sensitive user data, including encryption techniques for data transmission and storage, secure handling of user authentication and session management.
- Design an intuitive and user-friendly interface with clear navigation, visually appealing visuals, and features such as tooltips and explanatory text to assist users in understanding displayed information and performing tasks effectively.
- Implement error handling mechanisms to gracefully handle user input errors and system failures, log system activities, and errors for troubleshooting and auditing purposes.

==

Status of the Software Implementation

Part I Initial Requirements

Implemented Initial Requirements

We found that our initial requirements sufficiently capture the details needed for the project. We only found ourselves adding one new feature beyond the initial requirements

Requirement Type	Requirement	Delivered and tested (Yes/No)	Notes
User Requirement	Only registered users can access the dashboard	Yes	
User Requirement	Users can register by using a username and password	Yes	
User Requirement	Users can modify their account information	Yes	
User Requirement	Users can request to reset their login information	Yes	
User Requirement	Users can access the charts of city rankings	Yes	
User Requirement	Users can access pollution visualization maps	Yes	
User Requirement	Users can access graph visualization of pollution	Yes	
User Requirement	Users can adjust the date by calendar and see the historical dashboard	Yes	
User Requirement	Users can comment on the graphs	Yes	
Functional Requirement	Main Visuals → Chart.js or Plotly	Yes	
Functional Requirement	Rank top 10 cleanest to least clean cities	Yes	
Functional Requirement	Rank top 10 least clean cities	Yes	
Functional	Show correlation with population and	Yes	

Requirement	other external data		
Functional Requirement	Chart of each pollutant (CO, NO2, O3 and SO2)	Yes	
Functional Requirement	Calendar for users to switch between dates and see dashboards from different dates	Yes	
Functional Requirement	US map with pinpoint of the place with pollution (map)	Yes	
Functional Requirement	Analysis page with graphs and a comment section (user post)	Yes	
Functional Requirement	Analysis page graphs have a future prediction	Yes	
Functional Requirement	Login with username and password	Yes	
Functional Requirement	Forgot password/username	Yes	
Functional Requirement	Password/username is reset and an email is sent with new login info	Yes	
Functional Requirement	Change username, password, and user information	Yes	
Functional Requirement	Create a new account (username, password, user information)	Yes	
Functional Requirement	Users will get a notification about email verification	Yes	
Non-Functional Requirement	Performance: The dashboard should load quickly and respond to user interactions	Yes	
Non-Functional Requirement	Security: The dashboard should implement appropriate security measures	Yes	
Non-Functional Requirement	Usability: The dashboard should have an intuitive and user-friendly interface	Yes	

Unimplemented Initial Requirements

Requirement Type	Requirement	Delivered (Yes/No)	If No, why not?
User Requirement	Users can get notifications of the air pollution in their location	No	Time constraints, the focus was on building core features
Functional Requirement	Users will get a notification once every week for a small trivia regarding the pollution data	No	Time constraints, the focus was on building core features
Functional Requirement	Users can modify the email address associated with their account	No	Users can only edit their names and passwords, not their email
Functional Requirement	Feedback (contact us)	No	Time constraints, the focus was on building core features
Non-Functional Requirement	Accessibility: The dashboard should be accessible to users with disabilities	No	Time constraints, the focus was on building core features

----

Tasks in the Backlog

of tasks in the backlog: 1

| Task	Notes
Make dashboard Scatter Plot interactive (when pressed, jump to the analysis page of the city)	This is the only new feature we have added since the initial requirements but did not get to it due to time constraints since we wanted to focus more on our key features.

===

Part II System Architecture

Outlined below is our planned system architecture.

Key Components

Sequence Diagram of Registration:
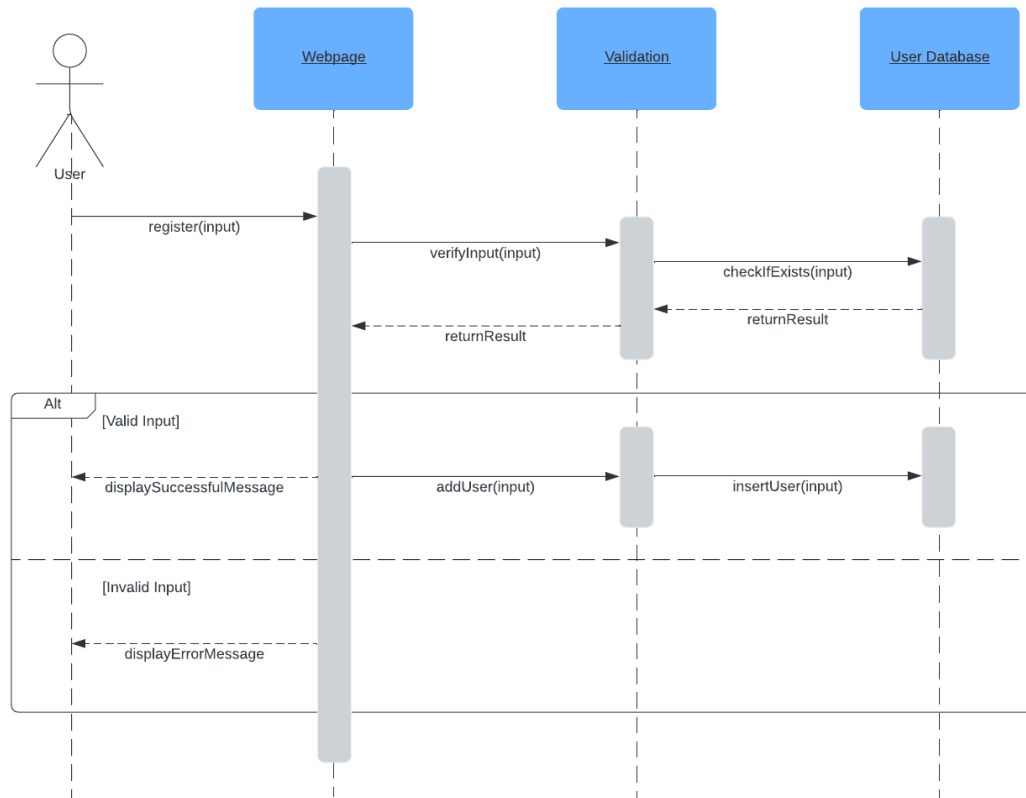
1. The user initiates the registration process by interacting with the webpage (registration page).
2. Upon receiving the registration request, the webpage triggers the "register" operation, which involves input validation.
3. The validation process verifies the input provided by the user. In the case of valid input, the validation process returns a "Valid Input" result.
4. The sequence proceeds to the database interaction, which includes further processing such as verifying input and checking if the user already exists in the database.
5. If the input is invalid, The webpage communicates with the user database to perform operations such as adding the user and displaying a successful message upon successful registration.
6. If the input is invalid, the sequence follows a different path. The validation process returns an "Invalid Input" result.
7. Upon receiving the "Invalid Input" result, the webpage displays an error message to the user, indicating that the provided input is not valid.

Sequence Diagram of Deleting Comment
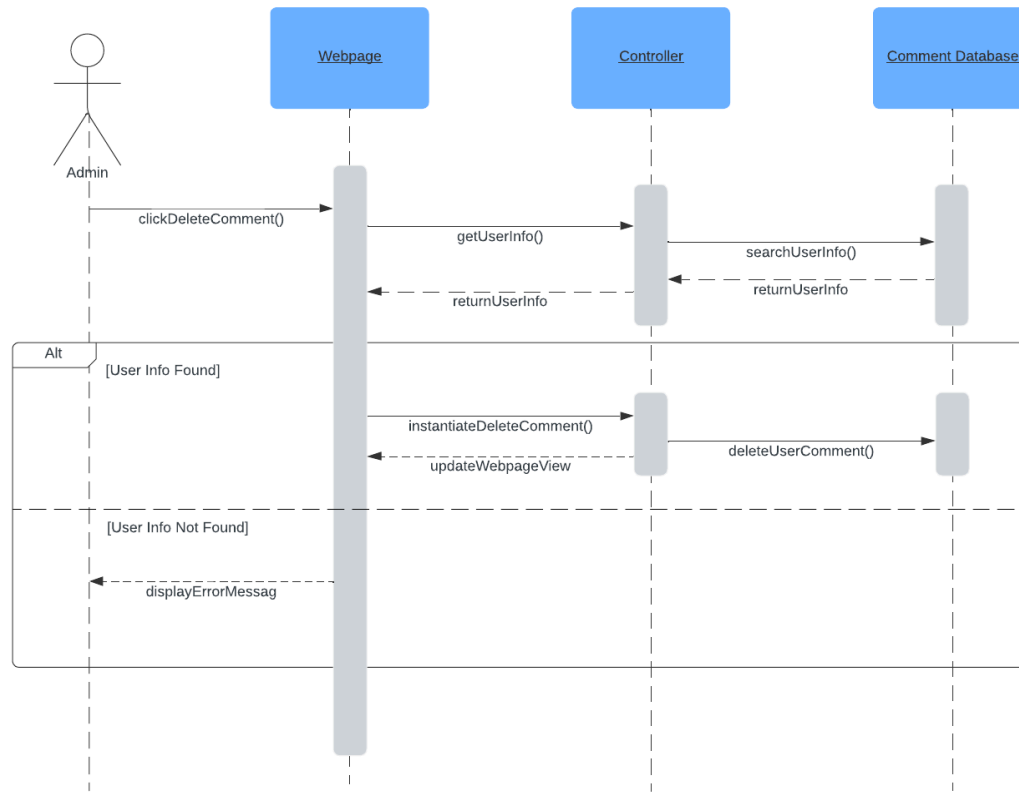
1. The admin user interacts with the webpage to perform actions related to deleting user comments.
2. Upon receiving the admin's request, the controller triggers the "getUserInfo" operation to retrieve information about the user from the comment database.
3. Comment Database: The webpage controller communicates with the comment database to search for user information.
4. If user information is found in the database, the sequence proceeds to the "deleteUserComment" operation to delete the user's comment. This includes removing the user's comment from the comment database.
5. If user information is found and the comment is successfully deleted, the webpage view is updated to reflect the changes.
6. If user information is not found in the database, the sequence proceeds to display an error message on the webpage.

Class Diagram
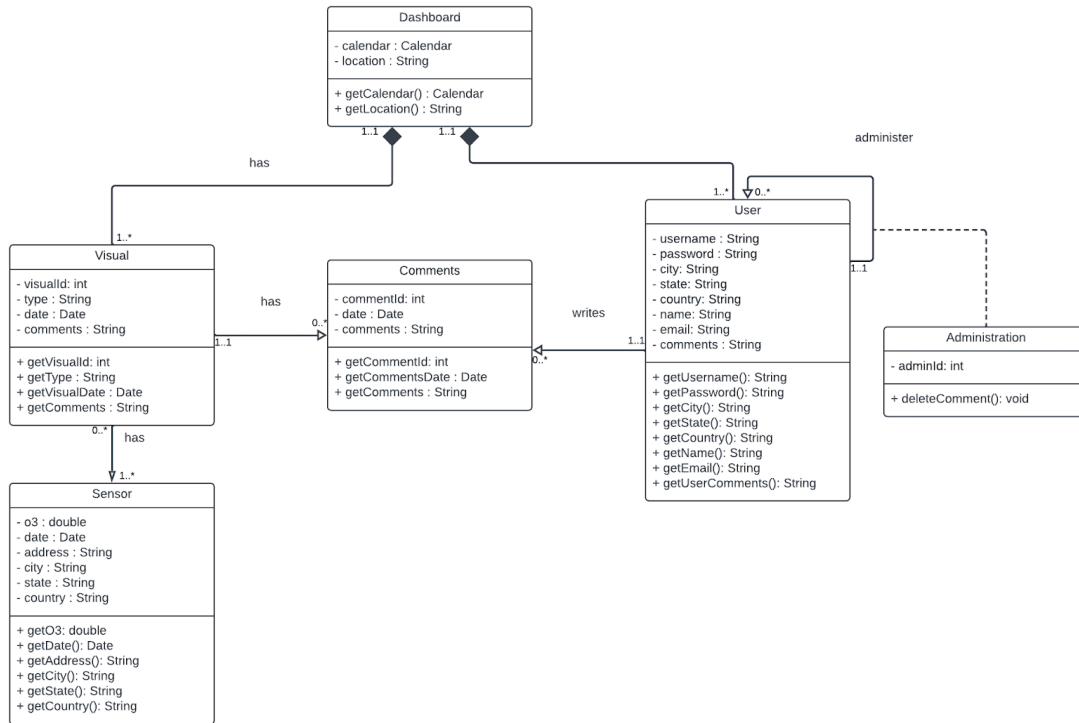
The class diagram depicts the architecture of the dashboard system. Users, including administrators, have access to various functions within the dashboard. Administrators possess all the functionalities available to regular users and additionally have the capability to delete comments. The dashboard presents visuals generated from sensor data (air pollution dataset). Users, upon registering their accounts, can view and write comments on these visuals.

Design Patterns

Adapter Pattern

<Adapter>
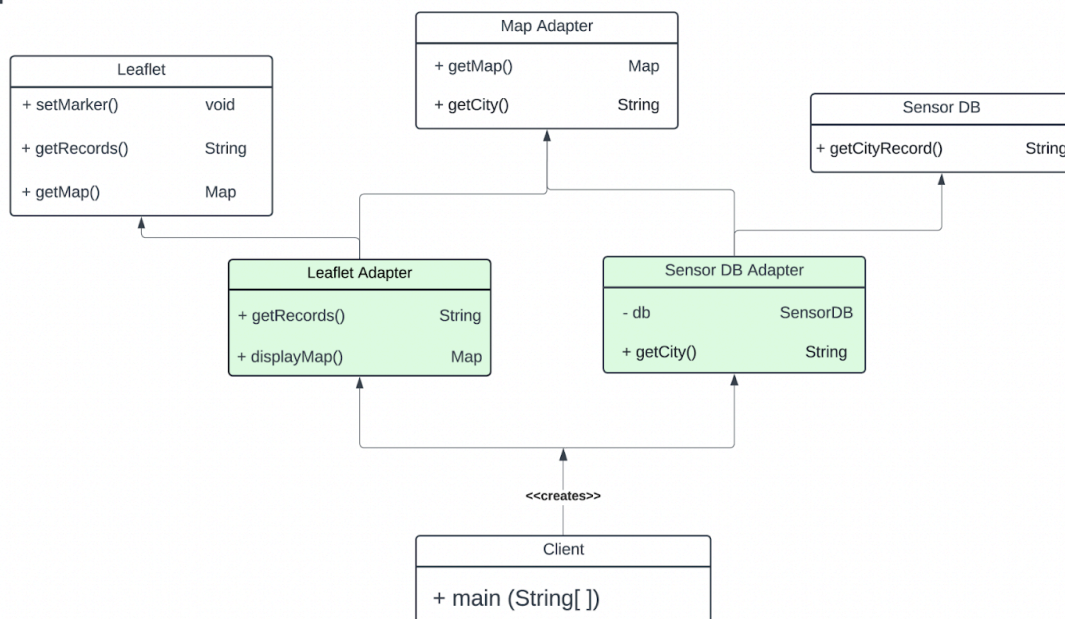
We chose an Adapter Design Pattern to allow incompatible interfaces to work together. In this design, a Client class wants to use a Map interface. There are two potential providers of map data: Leaflet and SensorDB. However, the interfaces they provide are not compatible with the Map interface that the client expects.

Here, the Leaflet Adapter and SensorDB Adapter translate the interfaces of Leaflet and SensorDB into the Map interface that the client expects. This allows the client to use either Leaflet or SensorDB without having to know or care about the specifics of their interfaces.

Factory Pattern

<Factory>
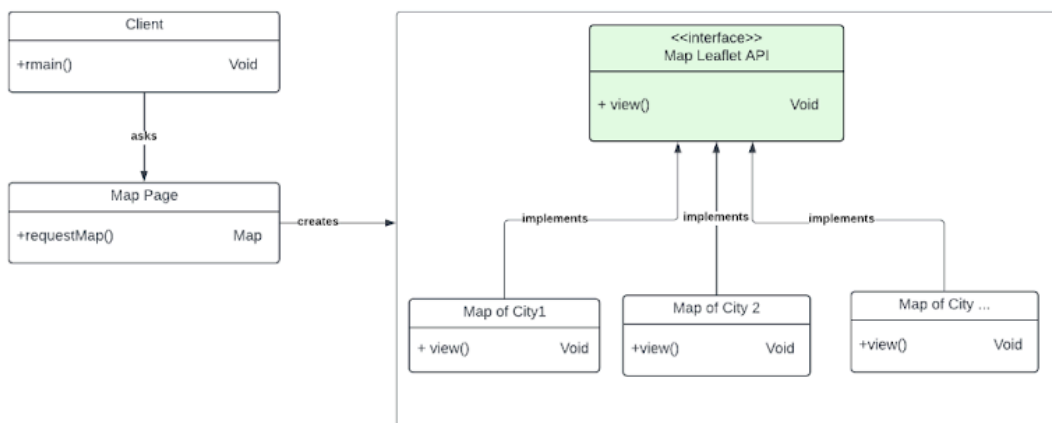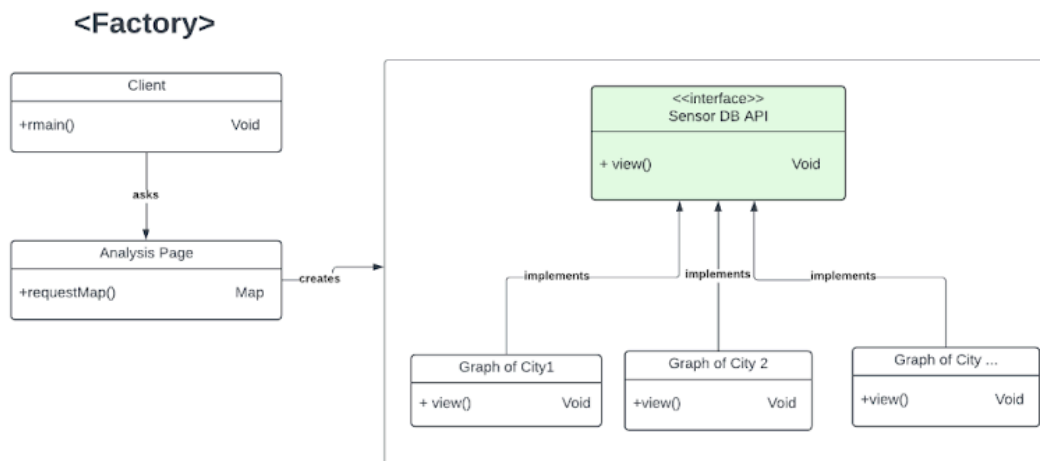
We chose the factory design pattern because it is the most compatible among all the design patterns that we have been introduced to.

In the factory pattern, we use two parts of the use case, one for when the user is accessing the map of the air pollution distribution in each city, and the other one is for the graph of each city.

When the client is accessing our product, they will go to either the map page or the analysis page. After they ask for the map page, the map page will try to create the maps by using the leaflet API, and they will produce the maps that are requested by the users. When they ask for the analysis on the analysis page, the analysis page will create a graph using the API that will process the data from the sensor, and produce the graph for each city's air pollution.

Flask for our Web Framework with SQLite

As we were building a light website, our team decided Flask was best suited for the job. Our team is all experienced in Python, which only made Flask a good choice. Our air pollution data is static and only covers data from January 1, 2000, to January 1, 2020, so we used SQLite for our database. Additionally, there are many resources outlining the use of Flask and SQLite, which were helpful.

Degree and Level of Reuse

As a team, we competed in a hackathon and built a website using Flask, with a similar goal. From this project we were able to reuse a lot of the front-end HTML and CSS templates and styles, however, the backend was all new code.

===

Part III Reflection and Bugs

Reflection on Plan vs. Result

We followed our system architecture plan detailed above with no changes. We implemented the sequence diagrams as outlined above. We did have to update our class diagram to account for data we realized we needed or did not need to include. For example, we needed to hold data for five different pollutants, not just O3. We implemented the design pattern Adapter and Factory as outlined above. We found our planning to be of great use and we followed our plans closely.

Know Bugs and Potential Fixes

There are no potential bugs and fixes in the system.

==

Step-By-Step Guide

Part I Installation Details

Installation Process

1. Clone the project repository from GitHub or download the project files.
   Git-Hub repository: https://github.com/leolelion/bluebuffalo
2. Navigate to the project directory "bluebuffalo" in your terminal/command prompt.
3. Run 'pip install -r requirements.txt' to install all dependencies.
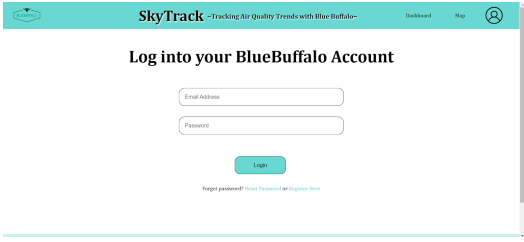4. Run 'python main.py'

===

Part II Website Tour

Credentials
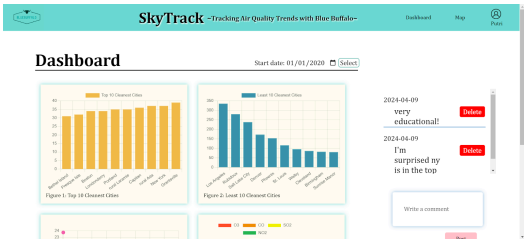
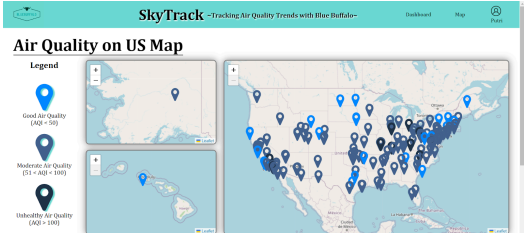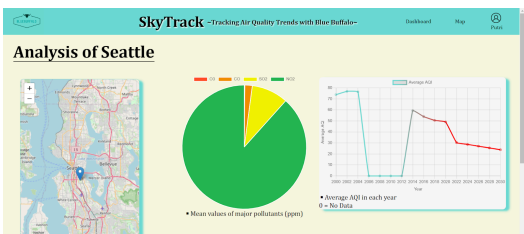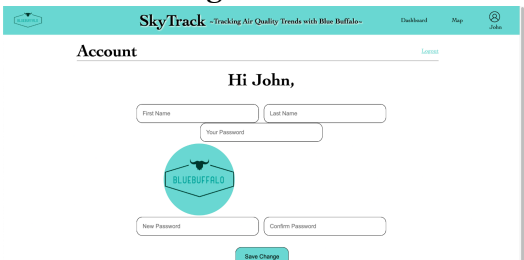| Admin User	Non-Admin User
Email address: pleksono@student.ubc.ca Password: putri310	Email address: john@example.com Password: john

===

Walkthrough

Registration 	- Go to the top right and select "Create Account" from the dropdown menu. (Put your email to receive a confirmation email!) - After submission, you will be asked to log in with your information.

Dashboard	- After successful login, you will be navigated to the dashboard page which is our home page. - The admin has the additional capability to delete comments, whereas regular users do not possess this function.
Map	- You can visit the map page by pressing the map button on the top right and clicking a marker of the city in the US to see the detailed pollution data.
Analysis	- Once you press the "Go to Analysis" button, you will be able to view the charts in the specific cities.
Account Setting	- A logged-in user will have the ability to modify their account settings, including their first name, last name, and password.

===

Part IV Dependencies

List of Dependencies
- Chart.js for visualizing pollution data in charts and graphs.
- Leaflet.js for displaying maps and markers.

Required Modules
- pytest
- flask
- flask-Login
- flask-SQLAlchemy
- pandas
- SQLAlchemy
- werkzeug
- jinja2
- scikit-learn

==

Reflections

Part I Project Management

Challenges and Lessons Learned

The project management works well for the team. The most challenging aspect is that the schedules of some members may not align with the deadlines, consequently resulting in delayed commits to GitHub. This can lead to conflicts that need to be resolved when the commits are significantly behind. What we would do differently is to keep track of who is working on each page, enabling effective communication among team members regarding priority tasks. Thus, when one member completes their work first, others can update on their progress, facilitating smoother collaboration and reducing delays.

==

Part II Assessing Initial Requirements and Planning

Evaluating Initial Requirements

The project's requirements were sufficiently detailed, providing a comprehensive roadmap for implementation. As a result, we successfully incorporated all specified requirements into the project's development. This clarity in requirements facilitated a smooth workflow, enabling the team to address each aspect with precision and accuracy. Ultimately, the thoroughness of the requirements ensured that the project met its objectives effectively and efficiently.

Overlooking Key Aspects

One aspect overlooked in our initial project planning was the need for effective communication among team members when one member encounters difficulties or is unable to complete their assigned tasks on time. This lack of communication mechanisms resulted in challenges when coordinating efforts and addressing issues promptly. Moving forward, we recognize the importance of establishing clear channels of communication to keep all team members informed about progress and any obstacles encountered. By fostering open communication, we can enhance collaboration, troubleshoot issues efficiently, and ensure smoother project execution.

==

Part III Reimagining Testing Practices

Team Approach

We believe we would maintain our current approach. The testing procedures we implemented proved to be adequate and effectively met the project's requirements. Therefore, there seems to be no immediate need for significant alterations or additional testing measures.

===

Part IV Revisiting Effort Estimation

Reflecting on Project Realities

Despite initial estimations revealing a gap, our team successfully bridged it and accomplished the project within the designated time frame. Certain team members contributed additional effort, yet this was effectively managed, ensuring timely project completion. This experience highlighted the importance of adaptability and collaboration in overcoming challenges and meeting deadlines in Agile projects. Ultimately, our ability to address discrepancies and leverage collective effort underscored the efficacy of our Agile approach.

===

Part V Pioneering Innovation

Learning Moments

Through our first foray into Agile methodology using GitHub and a Kanban board, We've gained valuable insights into efficient project management. Weekly meetings provided a structured platform for collaboration and progress tracking, enhancing team cohesion. Embracing Agile principles encouraged adaptability and responsiveness to evolving project requirements, fostering a dynamic workflow. Overall, this experience underscored the significance of iterative development and transparent communication in achieving project success.

===