IN PARTNERSHIP WITH PLYMOUTH UNIVERSITY

Name: Jayawardena Kavinda	
Student Reference Number: 10952523	

Module Code: PUSL2020	Module Name: System Development Tools and Practices
Coursework Title: Final Test and Report of the Air Quality Monitoring System	
Deadline Date: 29/04/2025	Member of staff responsible for coursework: Dr. Rasika Ranaweera | Ms. Pavithra Subhashini
Programme: BSc. (Hons) in Software Engineering	

Please note that University Academic Regulations are available under Rules and Regulations on the University website www.plymouth.ac.uk/studenthandbook.

Group work: please list all names of all participants formally associated with this work and state whether the work was undertaken alone or as part of a team. Please note you may be required to identify individual responsibility for component parts.

10952523 - Jayawardena Kavinda – Introduction / Run of Testing
10952463 - Dulaj Hewage – Designed Test Cases / Evidences / Finalizing
10952470 - Unagollage Wijesinghe – Functional Test Plan / Conclusion
10952545 - Witharamalage Sirimewan – Testing Strategy
10953075 - Duwage Perera – Role of Mock objects
10952629 - Rathnayaka Rathnayaka – Run of Testing

We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.

Signed on behalf of the group: Dinath

Individual assignment: ***I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.***
Signed :

Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.

I *have used/not used translation software.

If used, please state name of software……

Overall mark _____% **Assessors Initials _____** **Date_____**

*Please delete as appropriateSci/ps/d:/students/cwkfrontcover/2013/14

PUSL2020 – System Development Tools and Practices

Air Quality Monitoring System
Final Test and Report

Group – 34

Plymouth ID	Plymouth Name	NSBM ID	NSBM Name	Degree Program
10952523	Jayawardena Kavinda	30399	JWD Kavinda	SE
10952463	Dulaj Hewage	29981	DD Hewage	SE
10952470	Unagollage Wijesinghe	29676	DAD Wijesinghe	SE
10952545	Witharamalage Sirimewan	29590	WVW Sirimewan	SE
10953075	Duwage Perera	31863	DTT Perera	SE
10952629	Rathnayaka Rathnayaka	30224	MSI Rathnayaka	SE

Table of Contents

Table of Figures

1. Introduction

The Air Quality Monitoring System represents a web-based application which displays actual-time air quality information through its visualization framework. The system permits users to access, monitor and handle simulated Air Quality Index data. This document includes an overview of testing techniques used with the AQMS which acquires and stores current AQI information that it fetches from outside sources. Also information about development evidence alongside test case design, test execution guidance, functional test planning and an assessment of the selected test strategies.

We aimed to verify that the system delivers functional accuracy alongside robust design security features before its deployment phase.

2. Evidence of the Development

Each substantial program component includes screenshots with developer summaries located at the end of each page.

2.1. Cover Page

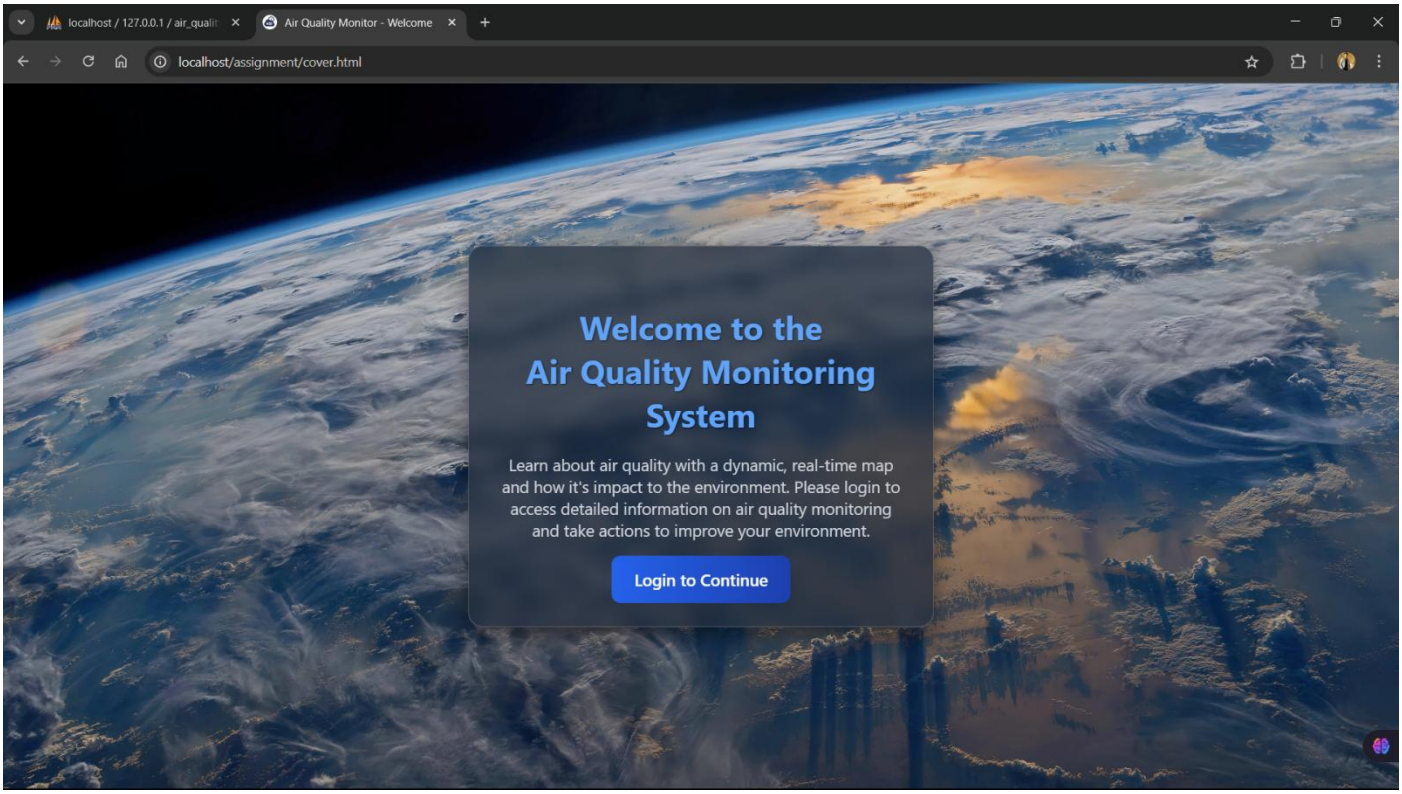


Figure 1 - cover.html

As the first displaying page of the application with a well-maintained structure this will direct users to the Login page.

2.2. Login Page

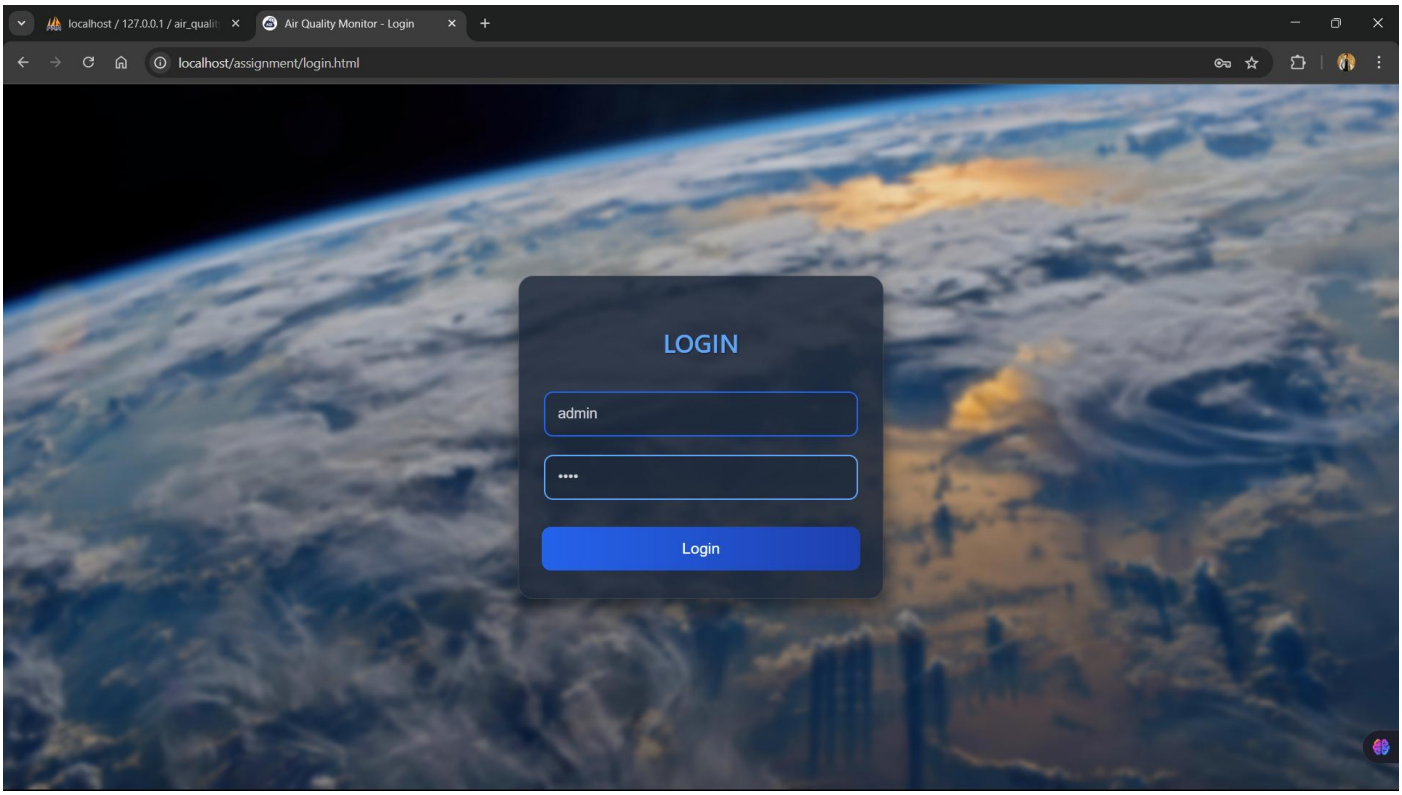


Figure 2 - login.html

Login form page for users. Authentication becomes a required process for Users to access dashboard or admin panel.

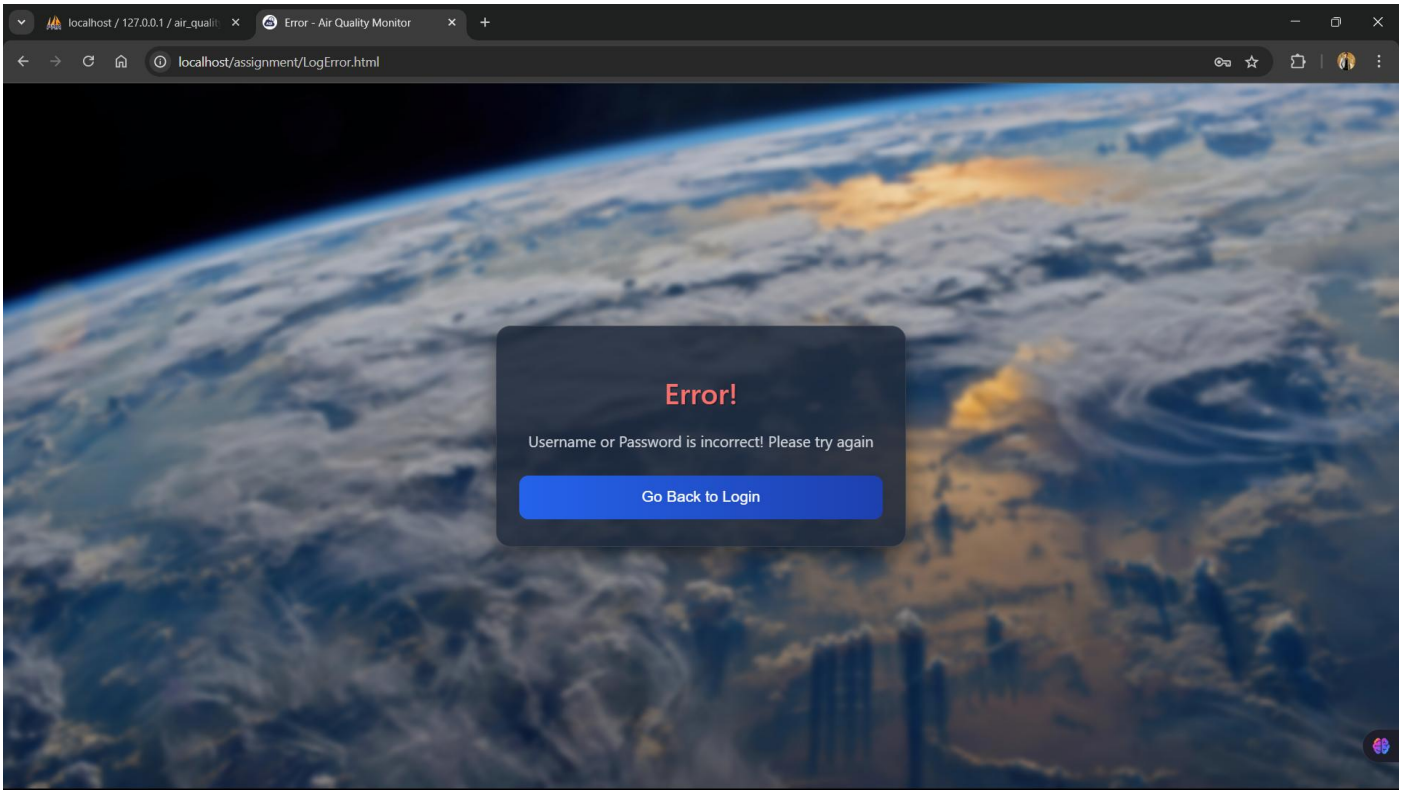


Figure 3 - LogError.html

When a Username or Password entered by the user was incorrect, the user will direct to this page and user can go back to the Login page for a retry.

2.3. Main Dashboard - Home

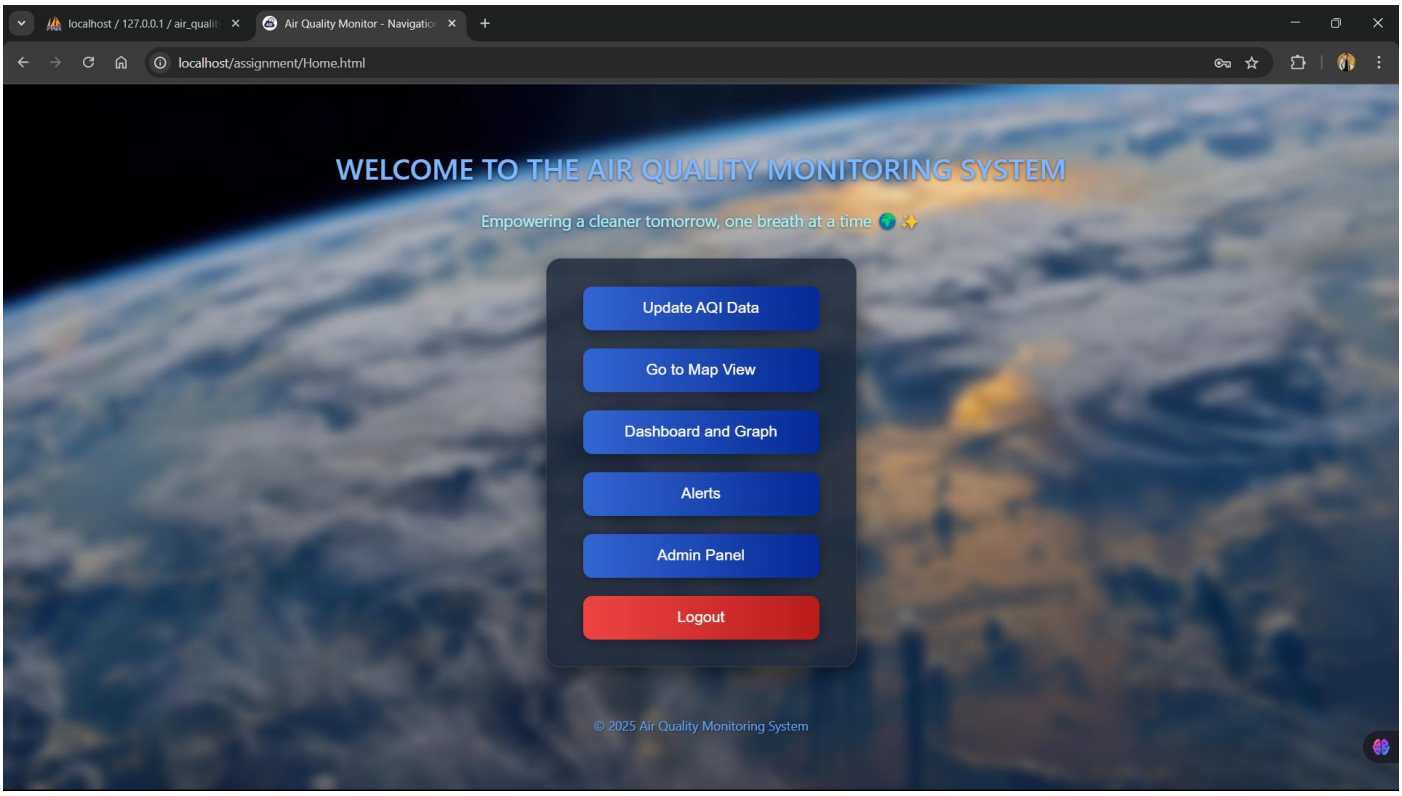
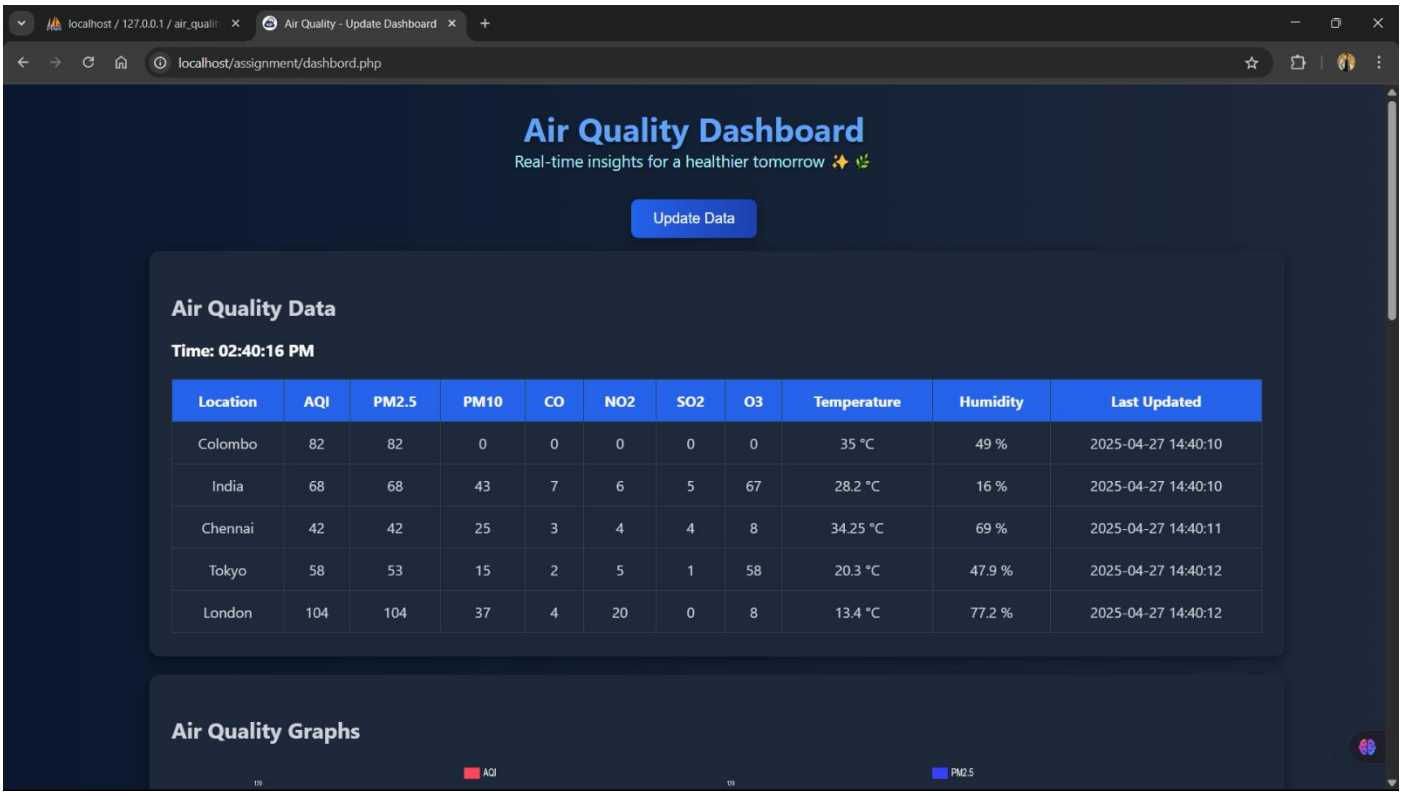


Figure 4 - Home.html

After login, the home page for user navigation.

2.4. Update AQI Data



Location	AQI	PM2.5	PM10	CO	NO2	SO2	O3	Temperature	Humidity	Last Updated
Colombo	82	82	0	0	0	0	0	35 °C	49 %	2025-04-27 14:40:10
India	68	68	43	7	6	5	67	28.2 °C	16 %	2025-04-27 14:40:10
Chennai	42	42	25	3	4	4	8	34.25 °C	69 %	2025-04-27 14:40:11
Tokyo	58	53	15	2	5	1	58	20.3 °C	47.9 %	2025-04-27 14:40:12
London	104	104	37	4	20	0	8	13.4 °C	77.2 %	2025-04-27 14:40:12

Figure 5 - Update AQI

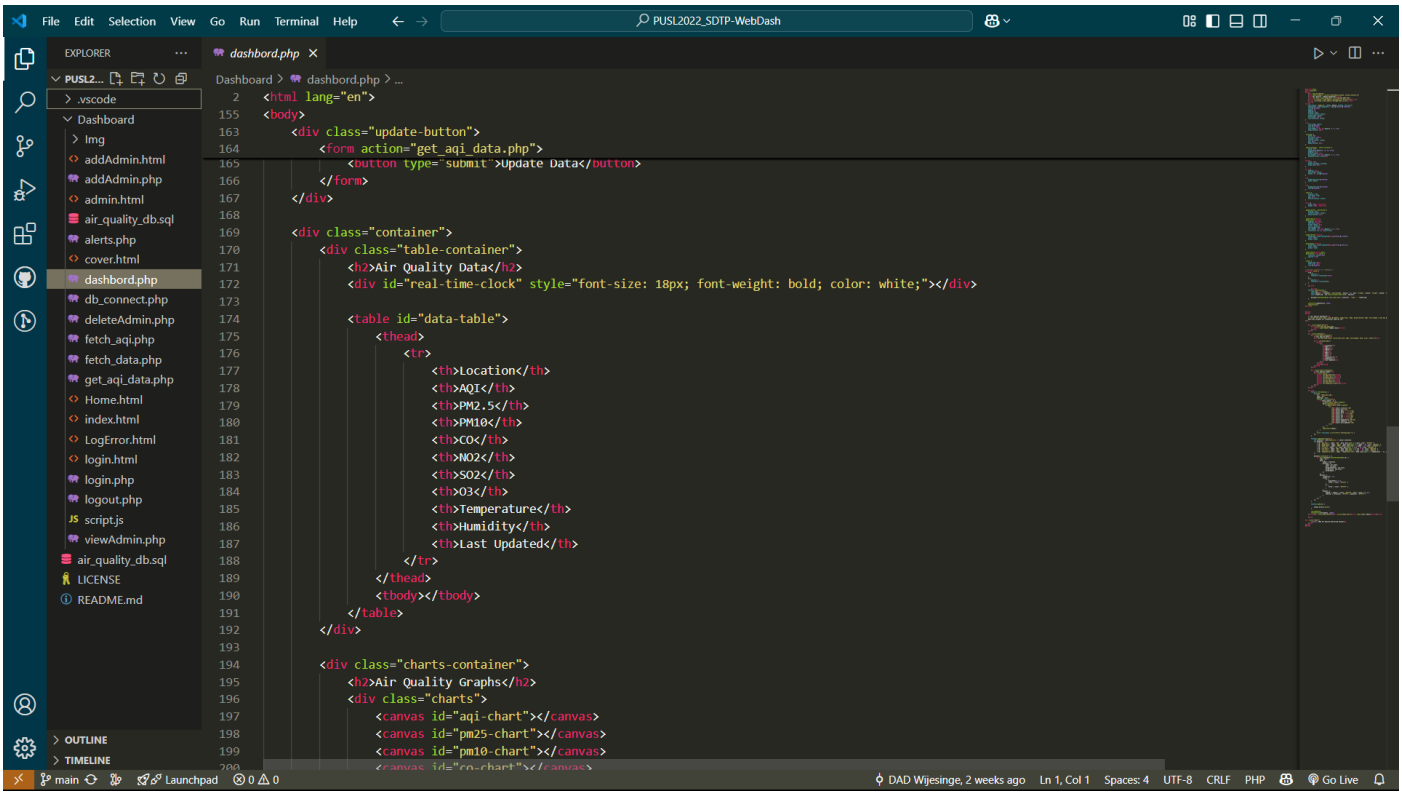


Figure 6 - dashboard.php

This displays the AQI sensor data dynamically fetched from the database by displaying the data in graphical form.

2.5. Go to Map View

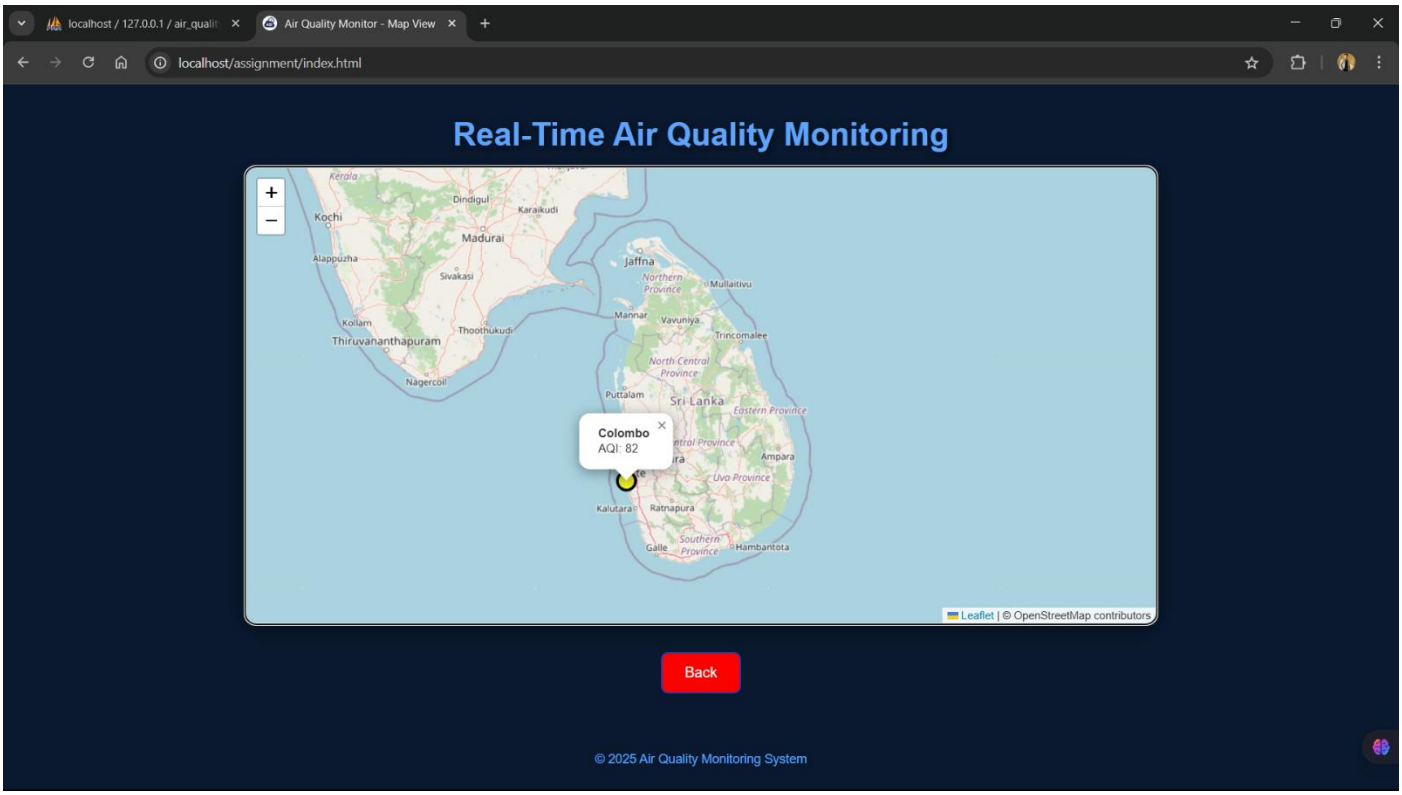


Figure 7 - Map View

The Map View page serves as a real time AQ monitoring interface. The implementation includes the integration of the Leaflet.js open-source library to render a live AQI map.

2.6. Dashboard and Graphs

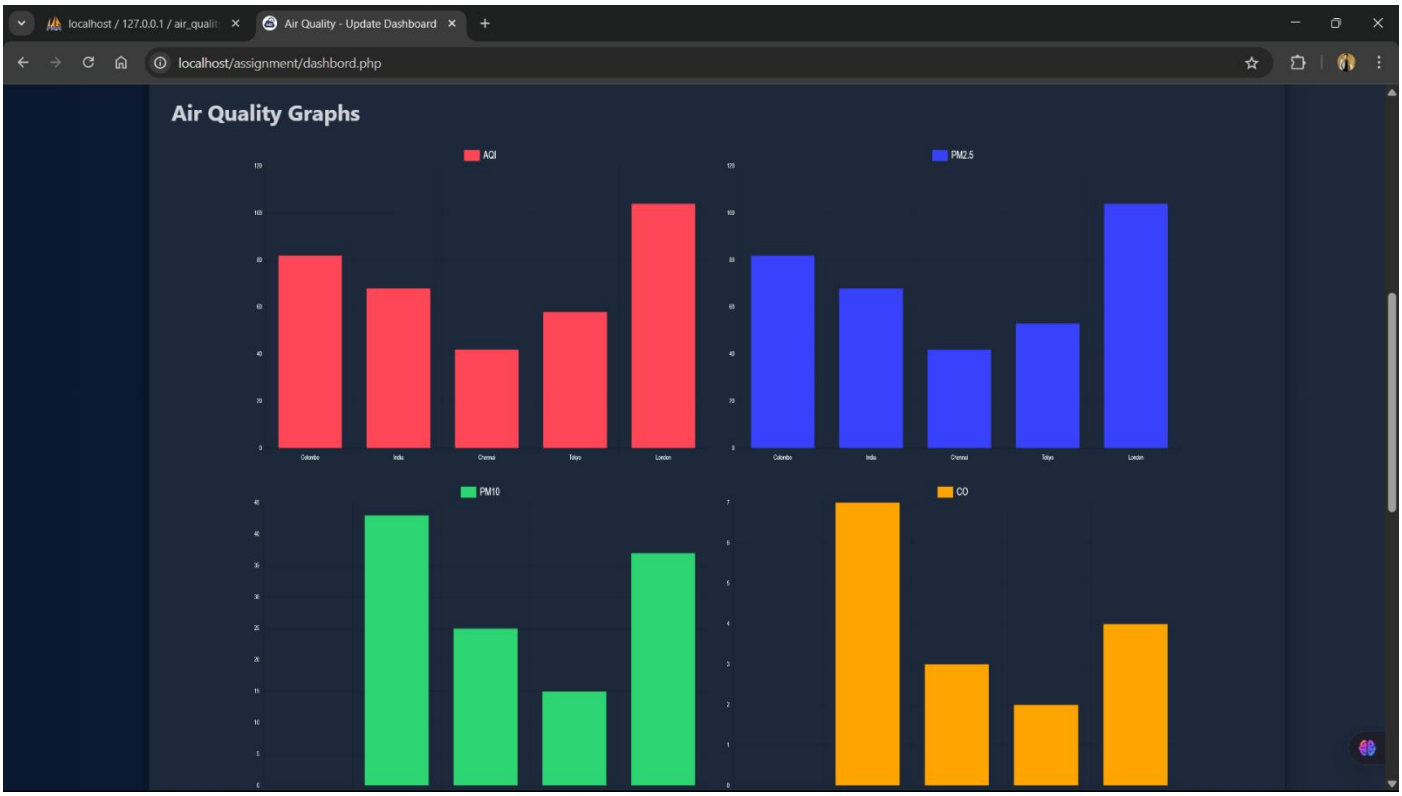


Figure 8 - Dashboard and Graphs

2.7. Alerts

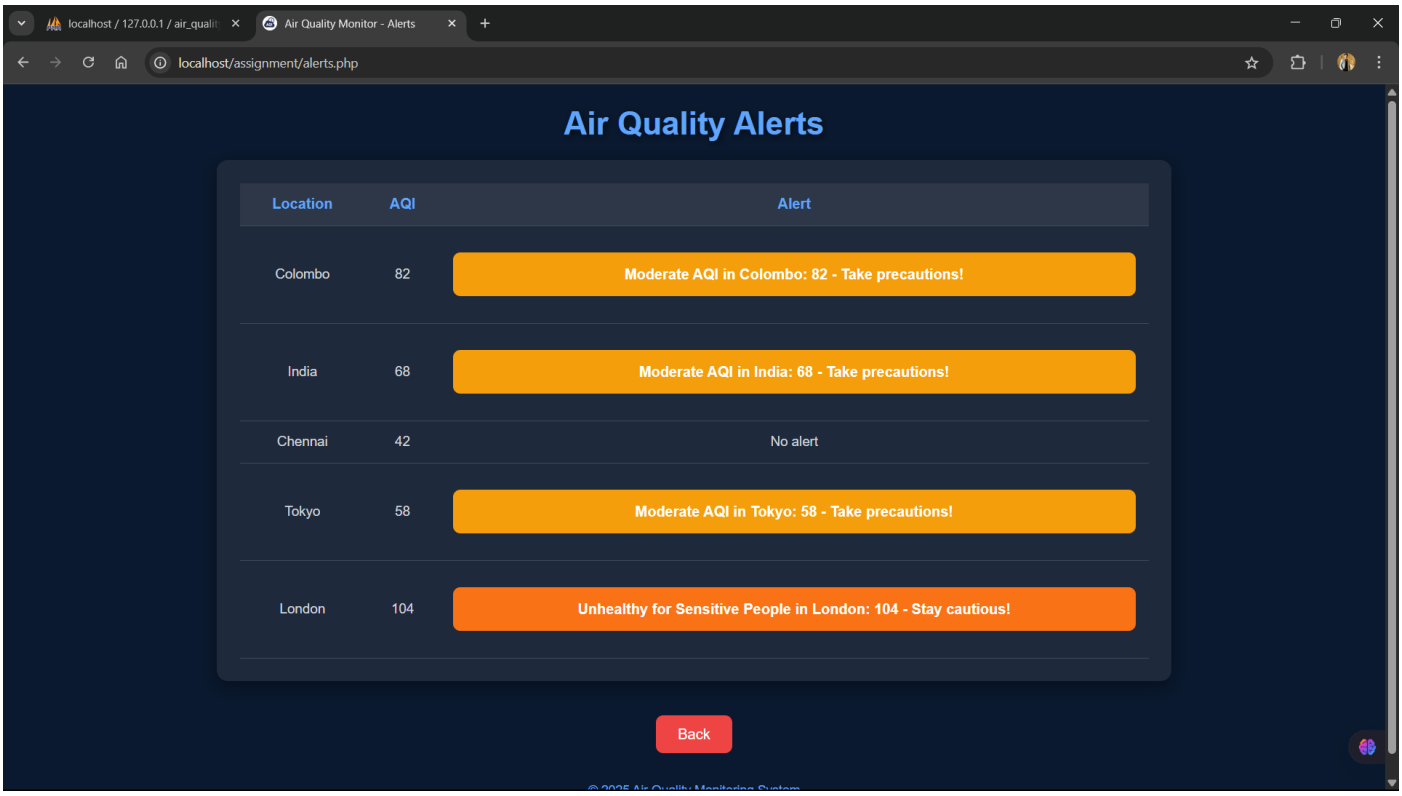


Figure 9 – Alerts

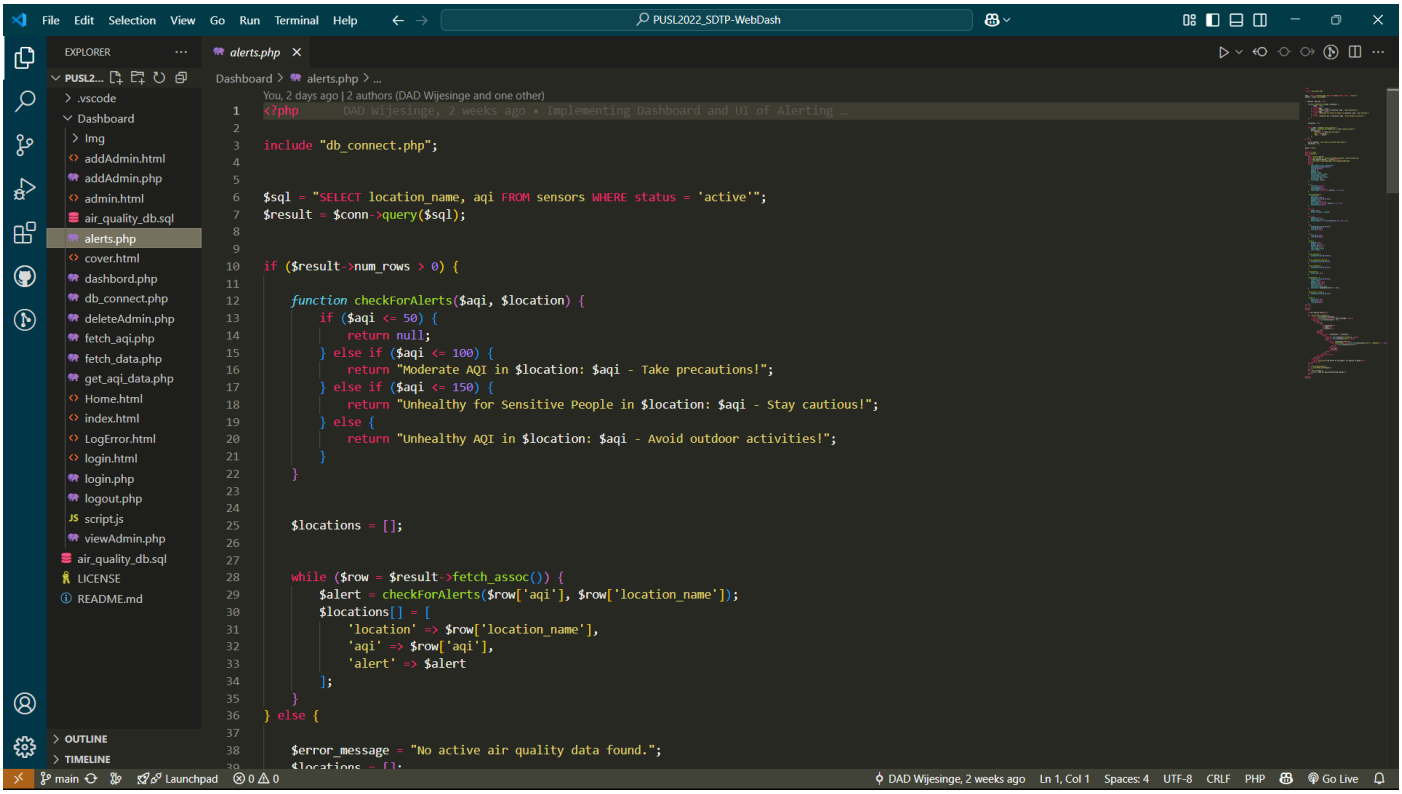


Figure 10 - alerts.php

A PHP script which can show alerts, or any other warnings related to air quality thresholds.

2.8. Admin Panel

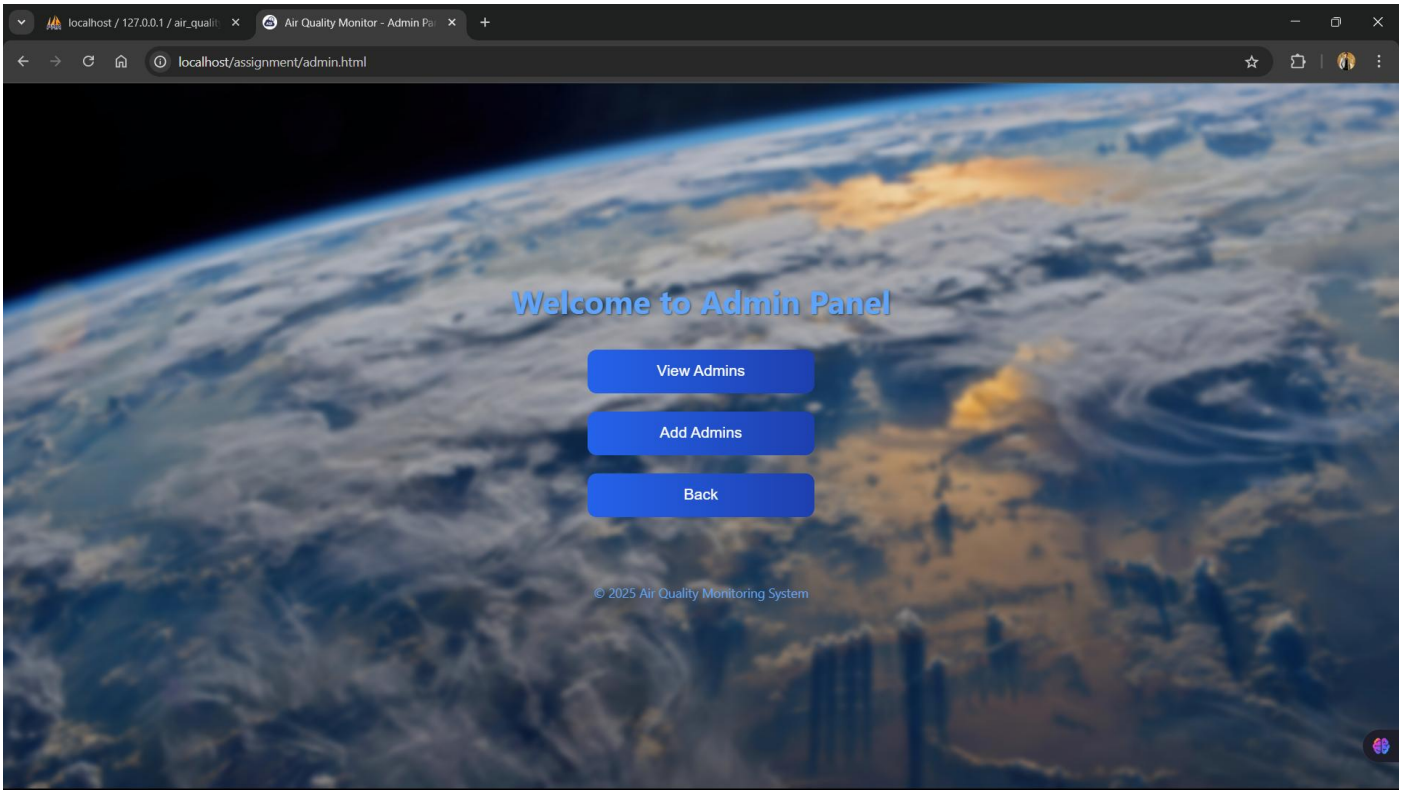


Figure 11 - Admin Panel

Main Admin Dashboard interface. It provides the admin functionalities for admin users.

2.9. View existing Admins

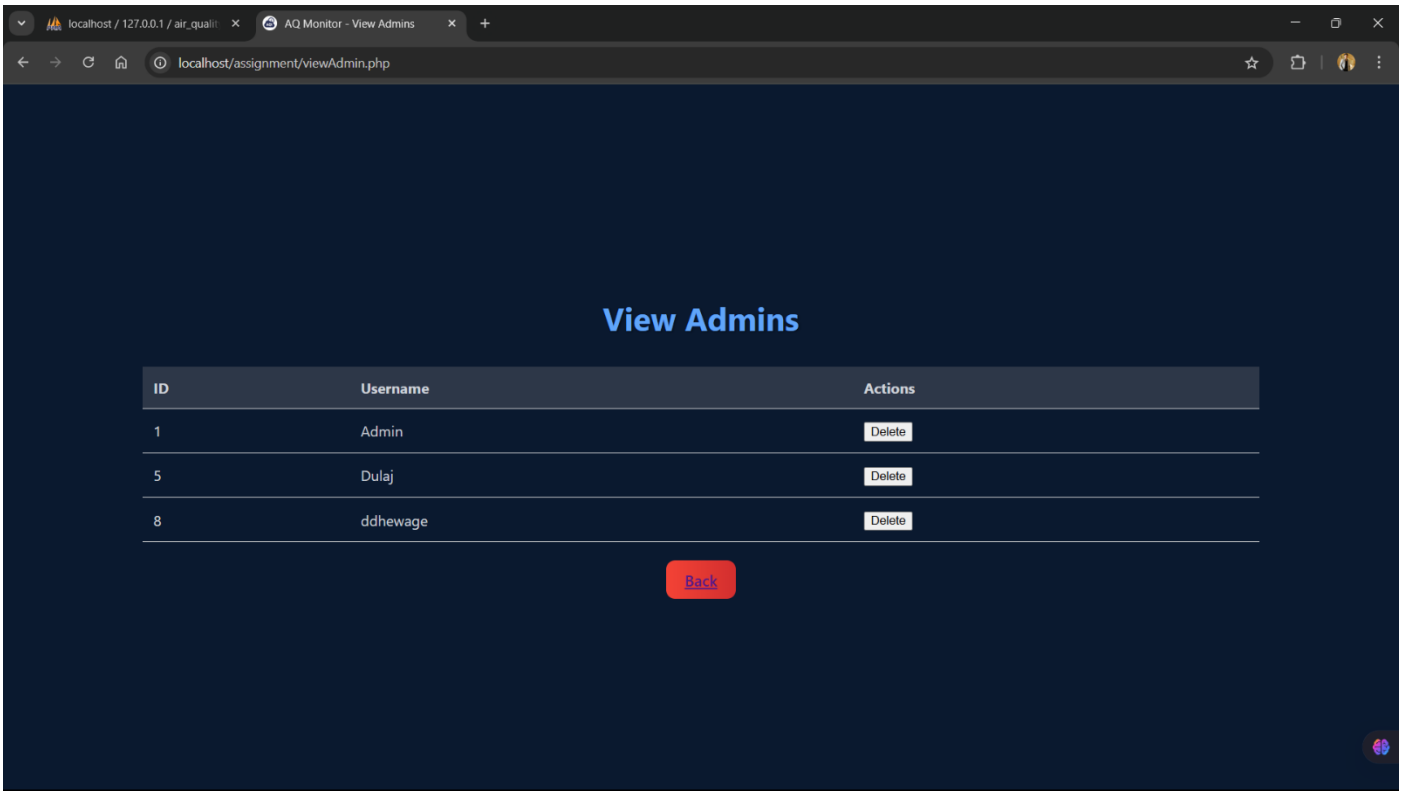


Figure 12 - View Admins

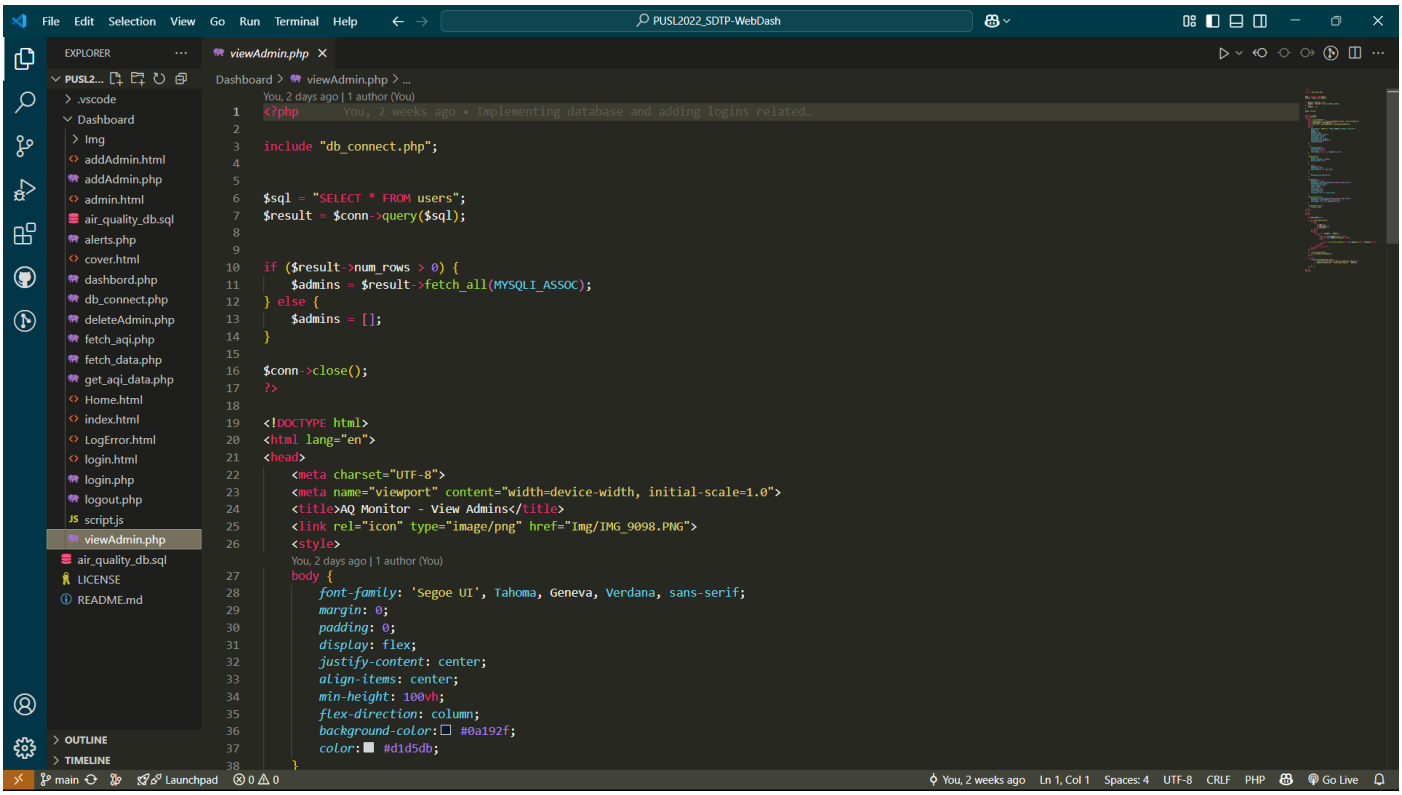


Figure 13 - viewAdmin.php

Display a list of all admin users, who have registered by displaying data from the database in tabular format using PHP page.

2.10. Add a New Admin

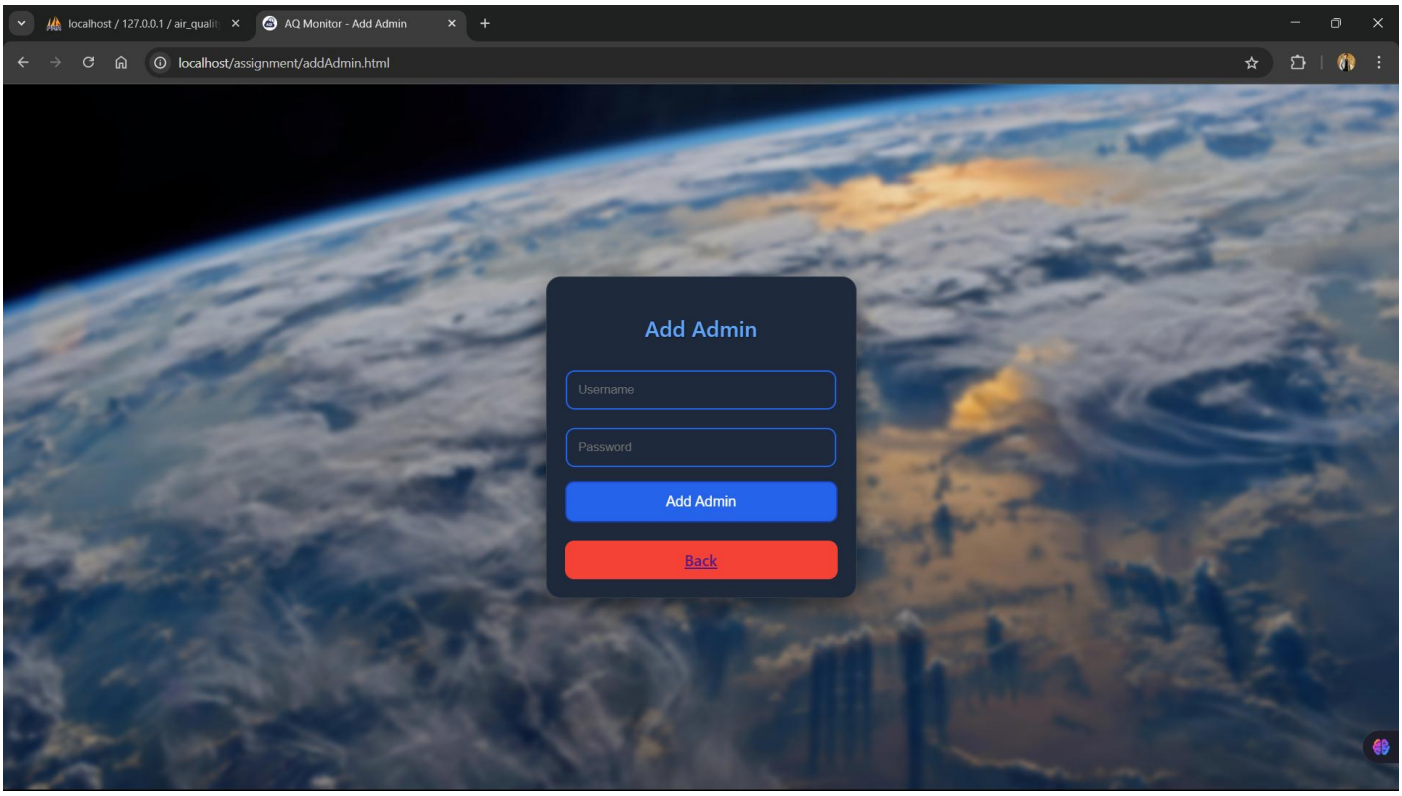


Figure 14 - Add a new Admin

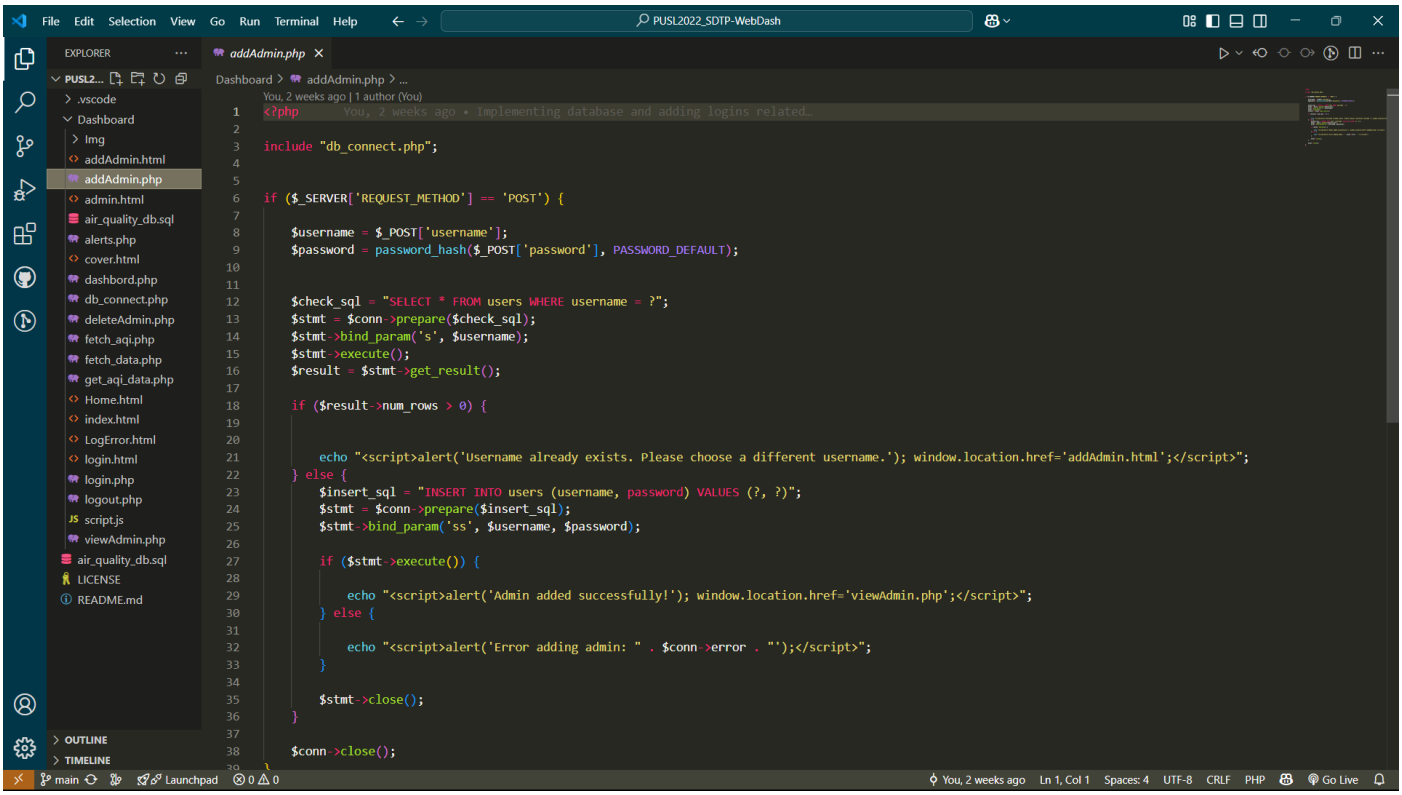
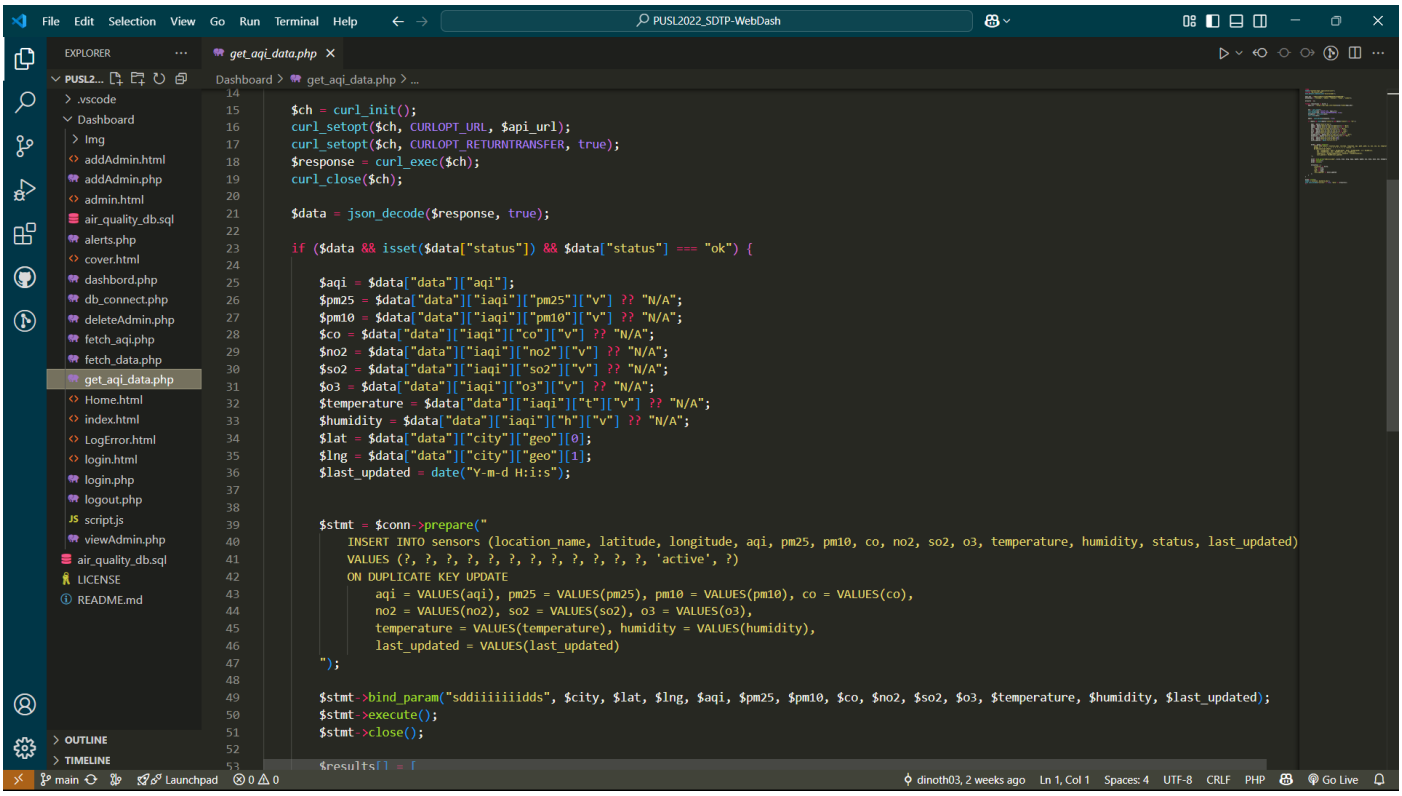


Figure 15 - addAdmin.php

Adding a new Admin details from here, it will store in the database with encrypted details that only each data will know the each user.

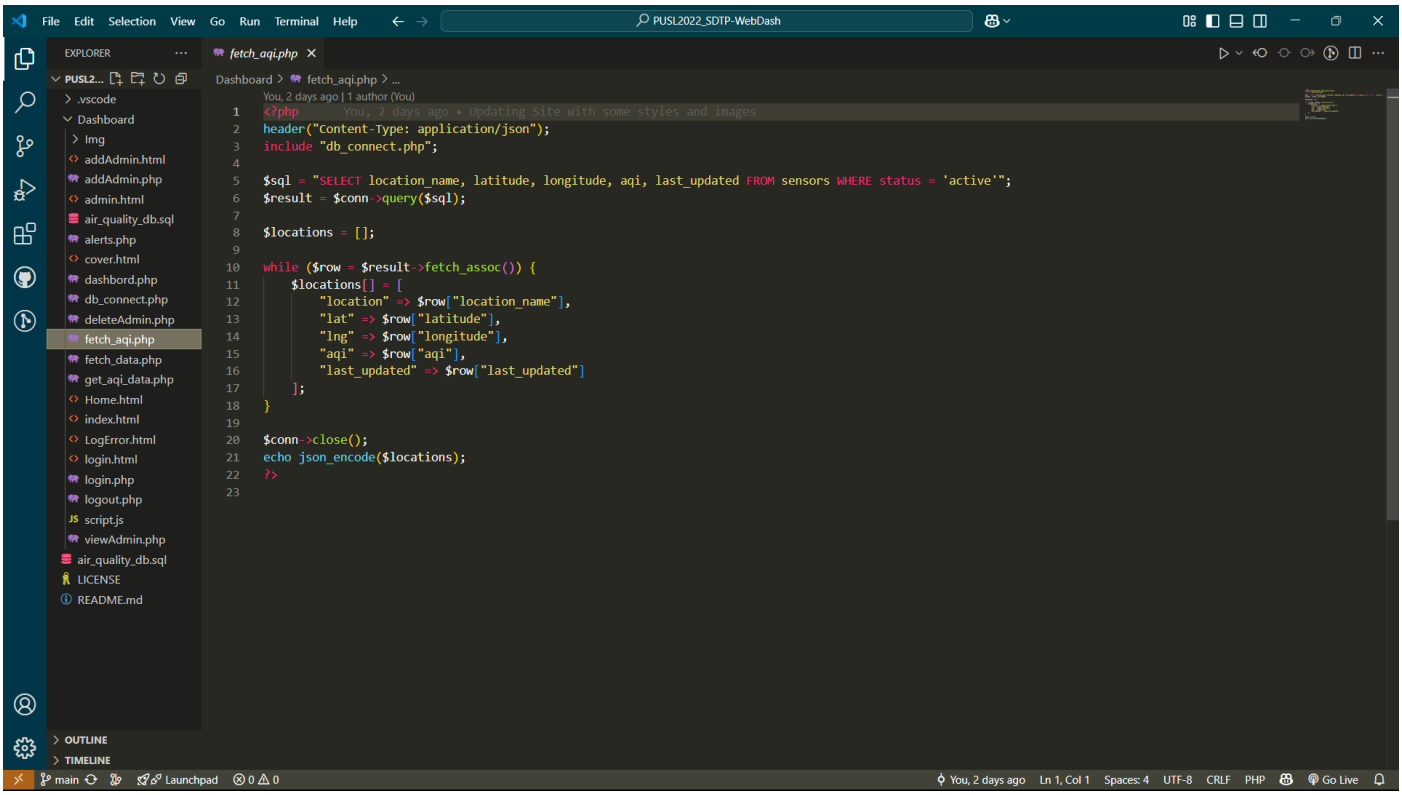
2.11. AQI Implementation



```php
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $api_url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
$response = curl_exec($ch);
curl_close($ch);

$data = json_decode($response, true);

if ($data && isset($data["status"]) && $data["status"] === "ok") {

    $aqi = $data["data"]["aqi"];
    $pm25 = $data["data"]["iaqi"]["pm25"]["v"] ?? "N/A";
    $pm10 = $data["data"]["iaqi"]["pm10"]["v"] ?? "N/A";
    $co = $data["data"]["iaqi"]["co"]["v"] ?? "N/A";
    $no2 = $data["data"]["iaqi"]["no2"]["v"] ?? "N/A";
    $so2 = $data["data"]["iaqi"]["so2"]["v"] ?? "N/A";
    $o3 = $data["data"]["iaqi"]["o3"]["v"] ?? "N/A";
    $temperature = $data["data"]["iaqi"]["t"]["v"] ?? "N/A";
    $humidity = $data["data"]["iaqi"]["h"]["v"] ?? "N/A";
    $lat = $data["data"]["city"]["geo"][0];
    $lng = $data["data"]["city"]["geo"][1];
    $last_updated = date("Y-m-d H:i:s");


    $stmt = $conn->prepare("
        INSERT INTO sensors (location_name, latitude, longitude, aqi, pm25, pm10, co, no2, so2, o3, temperature, humidity, status, last_updated)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, 'active', ?)
        ON DUPLICATE KEY UPDATE
            aqi = VALUES(aqi), pm25 = VALUES(pm25), pm10 = VALUES(pm10), co = VALUES(co),
            no2 = VALUES(no2), so2 = VALUES(so2), o3 = VALUES(o3),
            temperature = VALUES(temperature), humidity = VALUES(humidity),
            last_updated = VALUES(last_updated)
    ");

    $stmt->bind_param("sddiiiiiiidds", $city, $lat, $lng, $aqi, $pm25, $pm10, $co, $no2, $so2, $o3, $temperature, $humidity, $last_updated);
    $stmt->execute();
    $stmt->close();

    $results[] = [
```

Figure 16 - get_aqi_data.php

A PHP script that imports AQI data from API and updates the sensors table in database.



```php
<?php
header("Content-Type: application/json");
include "db_connect.php";

$sql = "SELECT location_name, latitude, longitude, aqi, last_updated FROM sensors WHERE status = 'active'";
$result = $conn->query($sql);

$locations = [];

while ($row = $result->fetch_assoc()) {
    $locations[] = [
        "location" => $row["location_name"],
        "lat" => $row["latitude"],
        "lng" => $row["longitude"],
        "aqi" => $row["aqi"],
        "last_updated" => $row["last_updated"]
    ];
}

$conn->close();
echo json_encode($locations);
?>
```

Figure 17 - fetch_aqi.php

In PHP, same script fetches and display AQI data dynamically from the database. Creates API and dashboard updates.

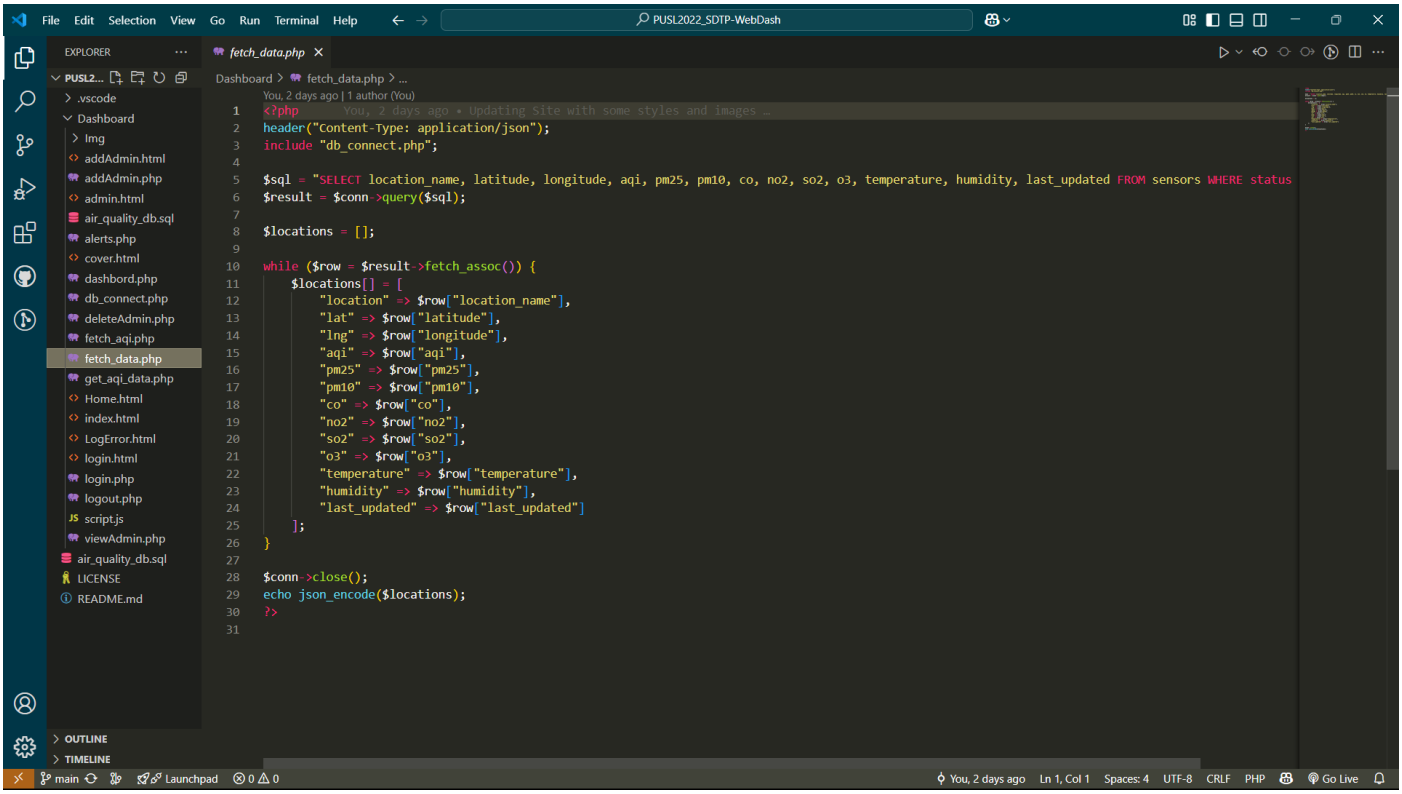


Figure 18 - fetch_data.php

Outputs active current sensor data to a JSON response (for tracking AQI and incorporating on the map).

2.12. Database and Connection

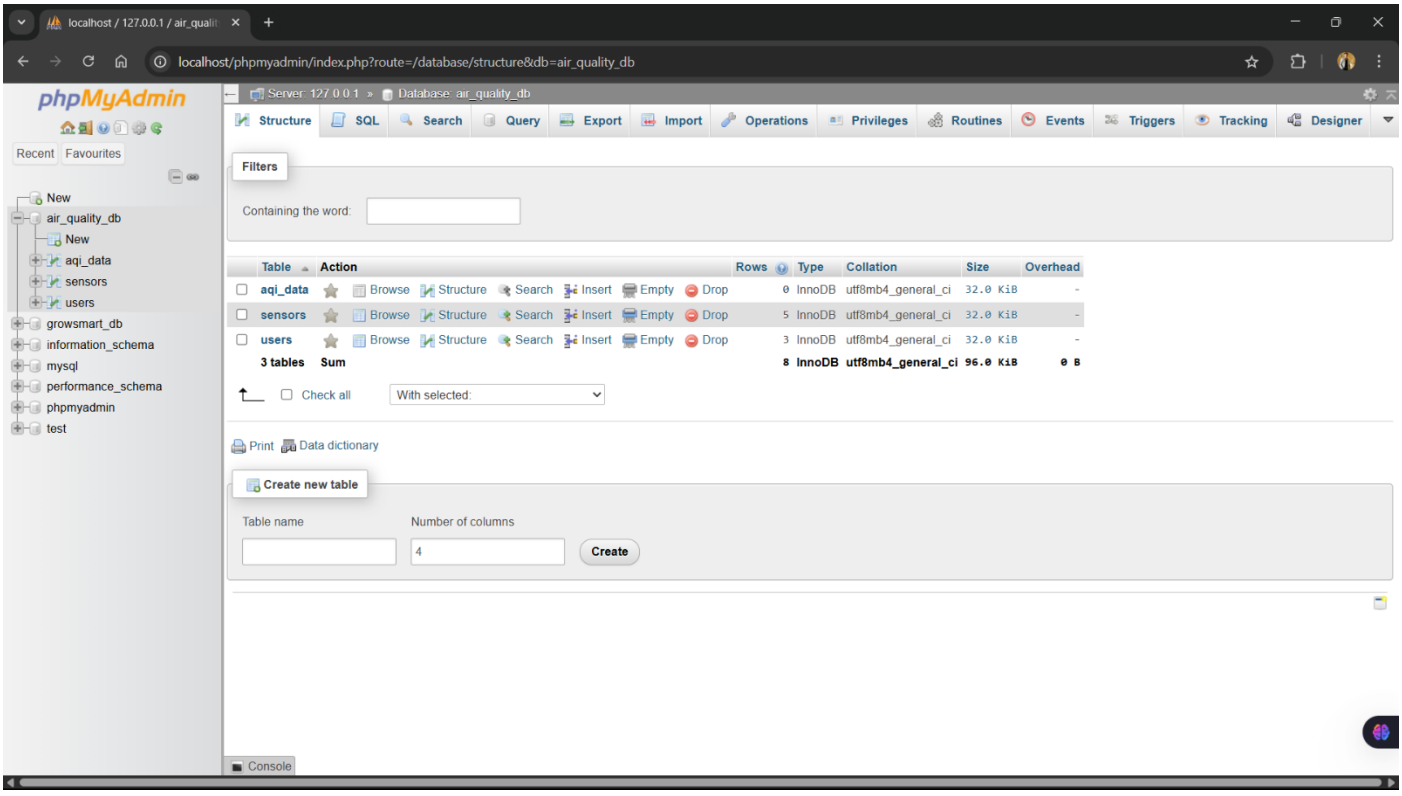


Figure 19 - air_quality_db

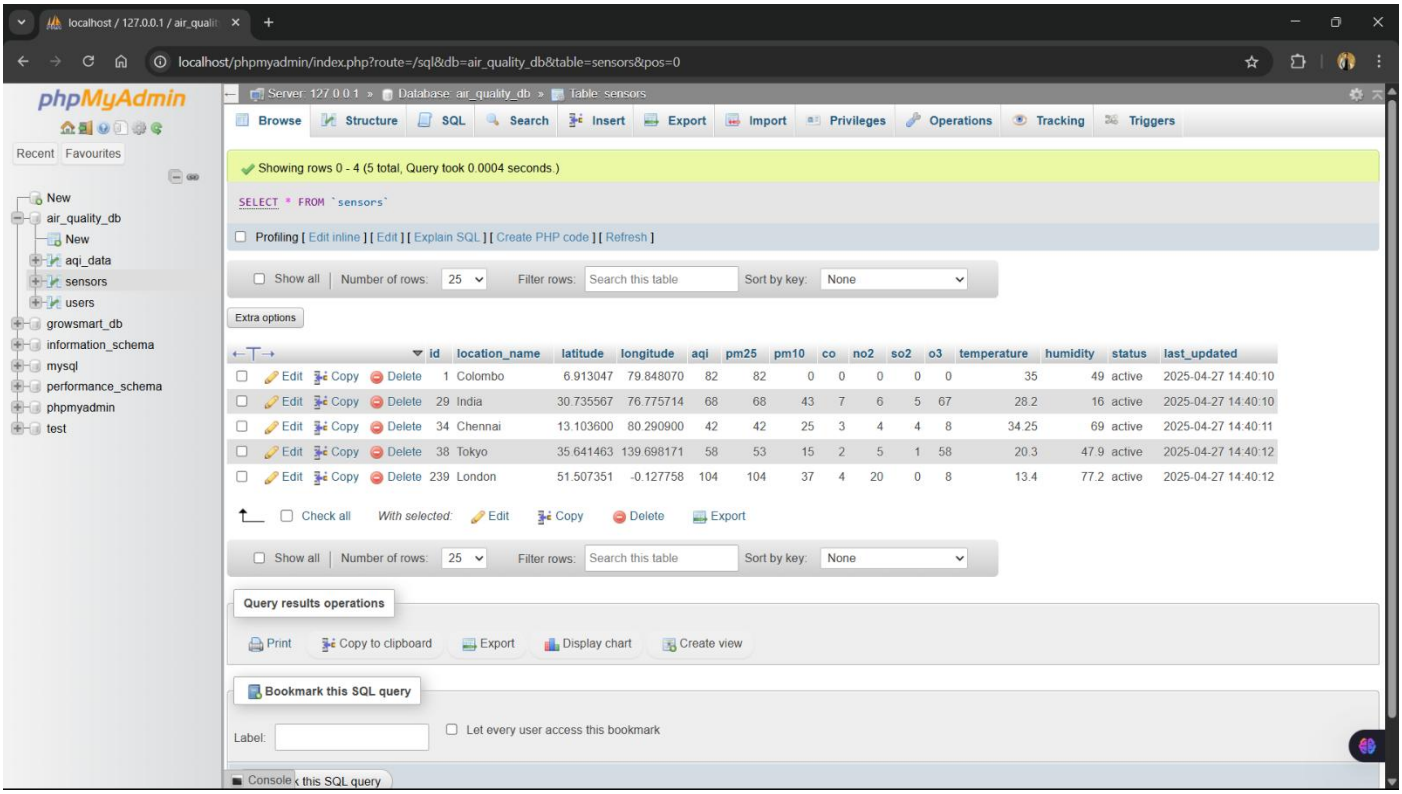


Figure 20 - Sensors table

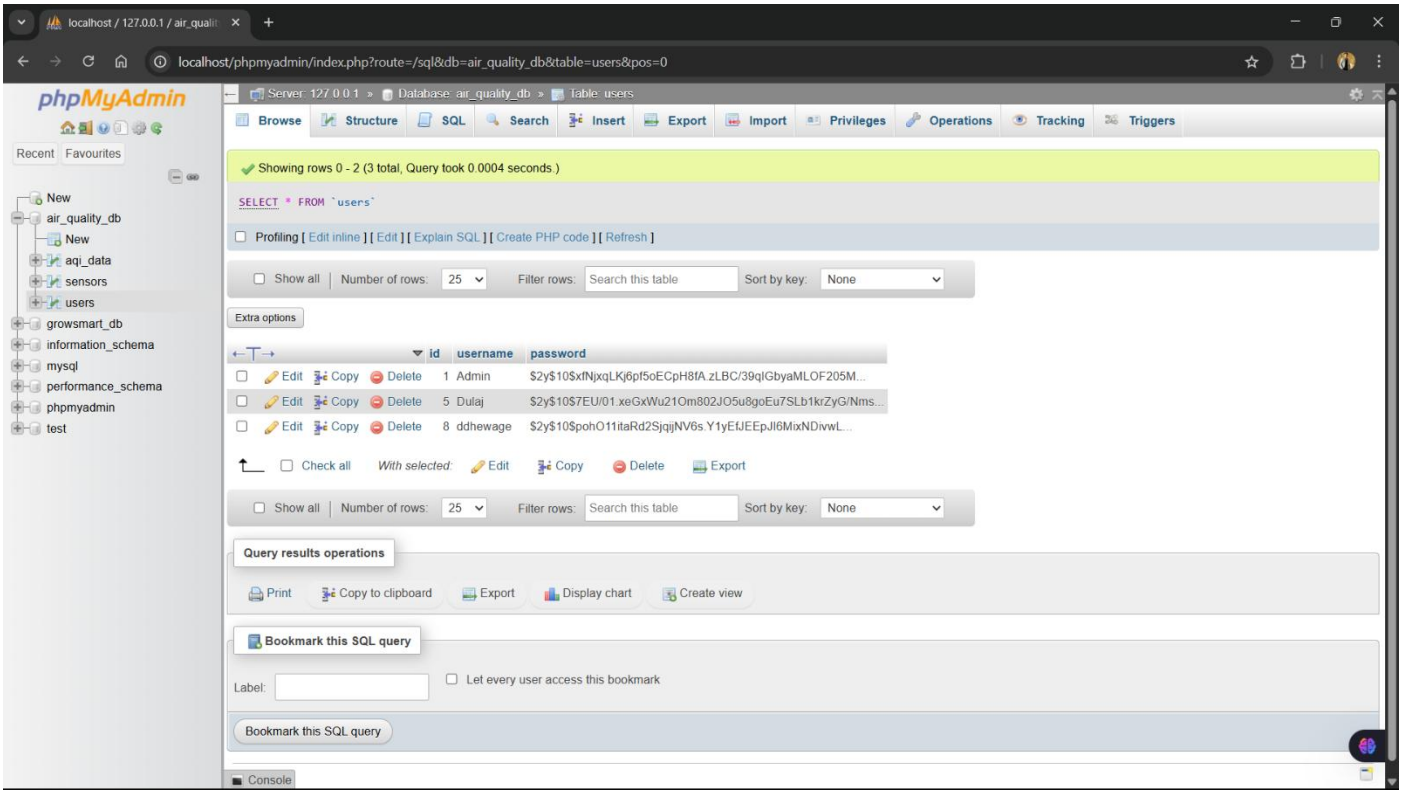


Figure 21 - Users Table

The “air_quality_db” represents the system database. The system contains three essential tables. :-

- aqi_data :- Stores historical AQI readings.
- sensors :- Sensors table contains AQI readings together with detailed pollutant information about PM2.5 along with PM10, CO, NO2, SO2, O3, Temperature and Humidity.
- users :- Every secure part of the system requires admin login credentials which are handled by this table.

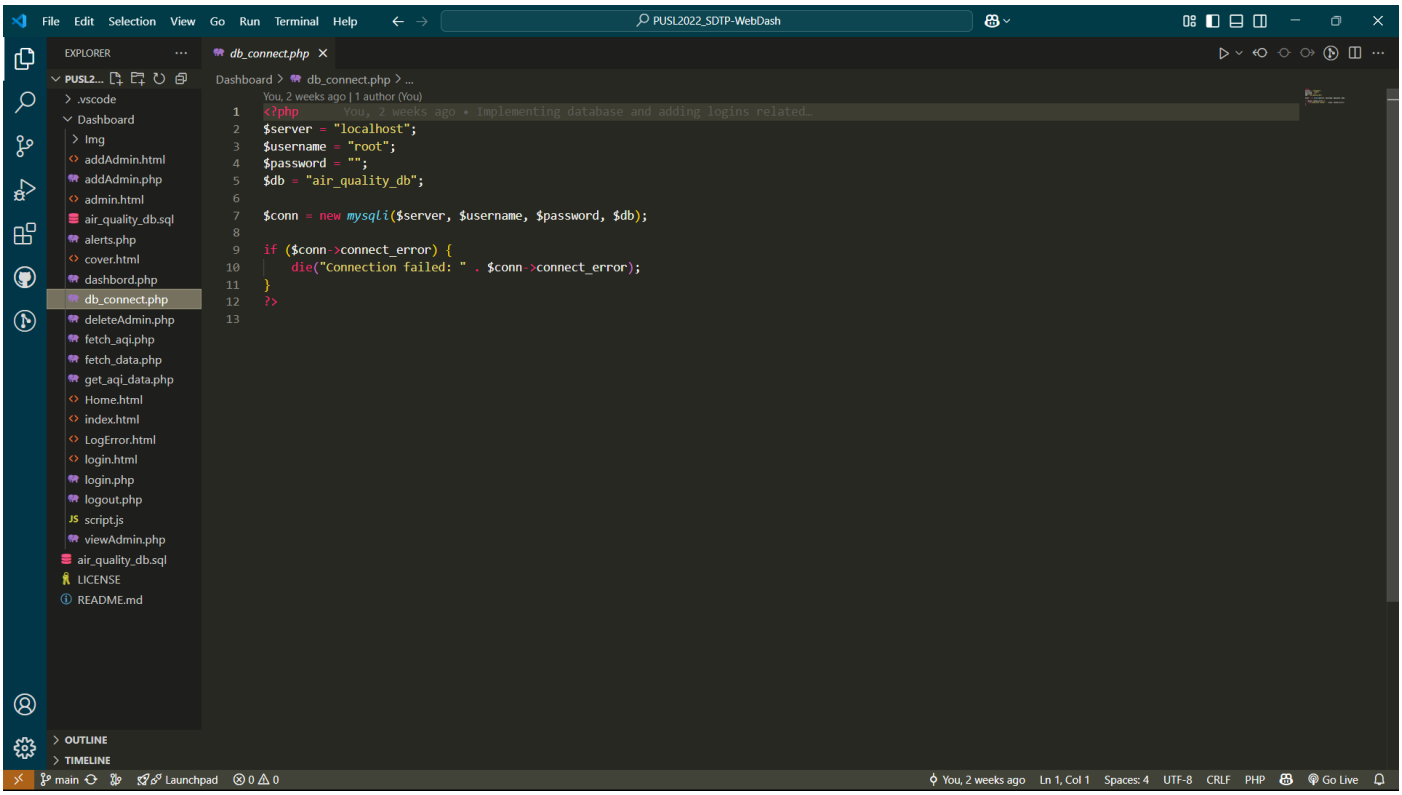


Figure 22 - db_connect.php

The PHP script connects to “air_quality_db” through “MySQLi” with default account credentials. The script contains error handling that ends the script execution in a safe manner when connection problems arise.

3. Designed Test Cases

Project Name	AQ Monitoring System
Module Name	AQMS Test Cases
Created By	Dulaj Hewage
Created Date	27th April 2025
Reviewed By	Group - 34

Test Case ID	Test Title	Test Summary / Objective / Description	Test Case Steps	Test Data	Expected Result	Post Condition	Actual Result	Status (Pass/Fail)	Notes / Comments
TC-001	Test Successful DB Connection	Verify connection to the database with correct credentials	1. Open db_connect.php 2. Run script	Valid DB credentials	Connection successful	Database connection established	Connection successful	Pass	-
TC-002	Test Failed DB Connection	Verify handling of wrong credentials	1. Open db_connect.php 2. Enter wrong password 3. Run script	Invalid DB credentials	Connection fails with error	No database connection	Connection failed with error message	Pass	-
TC-003	Fetch AQI Data (Valid data)	Test fetching AQI data for a valid data	1. Trigger get_aqi_data.php 2. Monitor response	Valid API key, valid	AQI data fetched correctly	Data available in database	Data retrieved and stored	Pass	Real API tested
TC-004	Fetch AQI Data (Invalid data)	Test system handling of invalid data	1. Modify API call with invalid 2. Run script	Valid API key, invalid	Proper error handling, no crash	System remains stable	Error handled properly, no crash	Pass	Handled through try-catch
TC-005	Insert New Sensor Data	Insert new sensor data into database	1. Fetch data for a new 2. Insert into DB	New sensor values	Data inserted into table	New row created	Data inserted correctly	Pass	Verified in phpMyAdmin
TC-006	Update Existing Sensor Data	Update existing sensor AQI values	1. Fetch data for existing 2. Trigger update	Updated AQI data	Data updated in table	Row updated, not duplicated	Data updated successfully	Pass	Updated timestamp and values
TC-007	Fetch Sensor Locations	Retrieve basic sensor location data	1. Call fetch_aqi.php		JSON returned with location data	JSON structure valid	JSON response received correctly	Pass	Structure matches API contract
TC-008	Fetch Full Environmental Data	Retrieve all environmental metrics for sensors	1. Call fetch_data.php		Full sensor data in JSON	JSON structure valid	Full detailed JSON received	Pass	Matches expected fields
TC-009	API Failure Handling	Verify system behavior during API failure	1. Disconnect internet 2. Trigger get_aqi_data.php	No internet	Graceful error handling	System does not crash	Error caught, logged	Pass	Simulated network failure
TC-010	SQL Injection Prevention	Test defense against SQL Injection	1. Submit malicious input 2. Observe behavior	SQL injection attempt strings	No injection, data remains safe	Database protected	No security breach	Pass	Prepared statements used
TC-011	Secure Credential Handling	Verify password protection in scripts	1. Inspect PHP files 2. Check for exposed credentials	Codebase	No exposed credentials	Database login safe	No credentials exposed	Pass	db_connect.php secured

Figure 23 - Test Cases

4. Run of Unit Tests and Integration Tests

4.1. Unit Tests

The absence of built-in PHP unit test execution features requires separate manual unit tests through PHPUnit libraries though our team applied this approach on every PHP script.

- Open Postman
- Send GET request to :-
  - localhost/fetch_aqi.php
  - localhost/fetch_data.php
- Verify that the JSON responses follow the correct format.
- A temporary renaming of database or table names simulates database connection loss conditions.

A stepwise process was used to verify the independent functionality of database connection and API retrieval and data processing operations.

4.2. Integration Tests

Testing of integrated system functions occurred through Integration Testing to validate the complete workflow.

Steps :-

- The visit of “get_aqi_data.php” through a browser triggers its execution.
- The database contains correctly updated and new records in the sensors table.
- The application executes queries with “fetch_aqi.php” and “fetch_data.php”.
- Check that the retrieved data corresponds to the expected data within the API data minimal API.

Successful integration tests ensured:

- The database updated dynamically.
- The query responses displayed recent information.

5. Functional Test Plan

Version	Change Date	By	Description
001	27.04.2025	Unagollage Wijesinghe	Functional Test Plan

5.1. Introduction

The AQMS project requires a functional testing strategy that defines its scope, establishes objectives and schedules through identified resources.
The validation process aims to assure that the AQMS fulfills its functional needs for all the functionalities.

5.2. Scope

- User Authentications
- The system handles the process of fetching, storing, updating AQI data
- Display of Dashboard
- Real-Time Map view of AQI
- Alerts based of AQI thresholds
- Admin Functionalities like View Admins, Add Admins
- Database Connection and Operations

5.3. Quality Objective

- The web application of AQMS must execute the established functional requirements.
- The system should verify correct data during active AQI updates and when alert conditions occur.
- The administrator features should be functional and protected for access.
- Detect operational problems during the essential functionality stage for a final launch.

5.4. Test Methodology

Agile testing methodology will guide our approach to obtain iterative improvement through functional testing round feedback.

5.4.1. Test Levels

- Unit Testing: Scripts such as “fetch_aqi.php”, “db_connect.php”, etc.
- The system integration testing stage checks the complete data flow that starts from the AQI database fetching operations through to dashboard display.
- The system testing encompasses full user scenario execution which includes login procedures and admin functions and notification mechanisms.
- Acceptance Testing: Final validation against the defined project requirements.

5.4.2. Bug Triage

- Bugs will receive four ratings which include Critical, High, Medium and Low priority.
- Projects require all Critical and High bug fixes to become operational prior to release.

5.4.3. Test Completeness

- 100% functional requirements covered.
- The number of closed critical and high priority defects was all.
- Successful demonstration of all primary use cases.

5.5. Testing Tools and Environments

- Browser-based testing (Chrome, Brave, Safari)
- API Validation via Postman
- phpMyAdmin for Database
- XAMPP for localhost
- OS :- Windows 10 or above
- Minimum 4GB, dual-core CPU

5.6. Terms

Terms	Definition
API	Application Programming Interface
AQI	Air Quality Index
AQMS	Air Quality Monitoring System
AUT	Application Under Test
DB	Database

6. Critical Analysis of Testing Strategy

Due to the dynamic nature of live AQI data, our testing strategy for the most part was manual functional testing along with mock simulation.

- The application demanded real-time handling of AQI data while we needed both dynamic live tests together with static mock tests.
- Testing with mock objects prevented our application from being affected by third party API system disruptions.
- The main application foundation relied on proper database management therefore we dedicated significant testing efforts toward database integration procedures.
- Manual testing provided flexible simulation capabilities because it required minimal complicated setup.
- The testing paid particular attention to SQL injection and connection security because public APIs participate in the system.

7. Structure and Role of Mock Objects

The system works directly with actual real-world systems which include an external API and live database features. Its primary function consists of managing real-time data instead of generating simulated data. Therefore mock objects are **not necessary and used** because of the system handles :-

- Real-time API interactions
- Live Database Operations
- Direct Data handling

The direct interaction with live data makes mock objects redundant for the current system functionality if testing environments require component isolation.

8. Conclusion

A complete testing phase for the Air Quality Monitoring System has included manual functional testing alongside mock object testing and database testing alongside API integration testing.
The system has achieved verification standards for accuracy as well as performance and error tolerance alongside security measures.

Future Enhancements :-

- Automated testing scripts (using PHPUnit).
- Adding continuous integration (CI) pipelines
- Structural mock servers act as data storage for scalable mock response management.

9. Breakdown of the Individual Contribution

Plymouth ID	Plymouth Name	Contribution
10952523	Jayawardena Kavinda	• get_aqi_data.php / login.php / logout.php • Resource for finding AQ data • Report contribution
10952463	Dulaj Hewage	• Implementation of Database / db_connect.php • Login.html / LogError.html / fetch_data.php / fetch_aqi.php • Handling Test Cases and Report contribution
10952470	Unagollage Wijesinghe	• Home.html / Dashboard.php / alerts.php • Implementation of home page and AQ Alerts • Test Planning and Report contribution
10952545	Witharamalage Sirimewan	• Admin.html / addAdmin.html / addAdmin.php • Interaction with Admin panel • Report contribution
10953075	Duwage Perera	• deleteAdmin.php / viewAdmin.php • Admin Panel handling • Report contribution
10952629	Rathnayaka Rathnayaka	• cover.html / index.html / script.js / air_quality_db • making cover / map view pages and DB handling • Report contribution