https://doi.org/10.31871/WJRR.13.3.12
World Journal of Research and Review (WJRR)
ISSN: 2455-3956, Volume-13, Issue-3, September 2021 Pages 13-20

IoT-based Platform with Big Data using Apache Kylin: Air Quality Monitoring System (AQMS)

Younten Tshering, Suyogya Ratna Tamrakar, Shubhangini Gontia, Smrity Baral

Abstract*— Air quality has become a big concern as the quality is reducing exponentially. PM2.5 particle size is directly linked to their potential for causing health problems. There are many IoT devices such as the PM2.5 sensor that connects to the internet and make people aware of the problem. From such devices, there are lots of data generated every second and in multiple locations which contribute to Big Data. With Big Data there are lots of challenges from storing to processing it. Hence, Apache Kylin was used to overcome the challenges of big data processing. For the dashboard display of the air quality data from the sensor, we connect Kylin and Tableau with JDBC/ODBC driver. The proposed system mostly deals with Online Analytical Processing (OLAP) on the Hadoop platform using Apache Kylin. Kylin creates OLAP cubes from the data in the hive and stores the cubes in HBase for analysis. As we implemented, we had some problems with Tableau and Kylin getting to connect since JDBC drivers were not compatible and for ODBC we had to contact the vendor. In the end, we have implemented the REST API to connect to Kylin and get the required data. Therefore, we developed a web app with Spring Boot which will show all the required details related to air quality, and accordingly, people can take preventive measures. Solving big data issues was a big challenge, so in our application, we have only used PM2.5 as a factor to indicate the air quality of the given location. This has made the scope of application less significant. Some gaps can be fulfilled by extending this application in the future. But still, we believe that our system will provide some convenient information to the users, making people conscious about the quality of air.

***Index Terms*— Apache Kylin, Big Data, HBase, IoT, OLAP, REST API, Tableau.**

I. INTRODUCTION

The airborne PM2.5 has returned to become a serious issue since the start of 2020. The thickening smog, exceeding the standard safe level, prompted the Thai government to approve measures to prevent and address it on January 21, 2020 [10]. PM2.5 comes primarily from combustion - fireplaces, car engines, and coal or natural gas-fired power plants are all major PM2.5 sources. Fine particulate matter (PM2.5) is an air pollutant that is a concern for people's health when levels in the air are high. PM2.5 are tiny particles in the air that reduce visibility and cause the air to appear hazy when levels are elevated [2]. To measure this level in the air, sensor PM2.5 (PM2.5 refers to particles that are 2.5 microns or smaller in diameter) can be placed at different places where there are chances of such issues. This sensor uses laser scattering to radiate suspending particles in the air, then collects scattering light to obtain the curve of scattering light change with time.

In this contemporary environment, air pollution or air quality has become a big concern as the quality is degrading exponentially. One of which is the PM2.5 crisis where the size of particles is directly linked to their potential for causing health problems. Fine particles (PM2.5) pose the greatest health risk [2]. These fine particles can get deep into the lungs, and some may even get into the bloodstream. Exposure to these particles can affect a person's lungs and heart. On a very clear and non-hazy day, the PM2.5 concentration can be as low as 5 µg/m3 or below. But the PM2.5 is considered unhealthy when it rises above 35.4 µg/m3.

There are many IoT devices such as the PM2.5 sensor that connects to the internet and make people aware of the problem. From such devices, there are lots of data generated every second and in multiple locations which contribute to Big Data. There are three major challenges with Big Data [9] namely storing a huge amount of data, storing a variety of data, and faster processing of those data. Hence, Apache Kylin was used to overcome the challenges of Big Data, and for visualization of the air quality data from the sensor, we use Apache Kylin and Tableau connected with JDBC/ODBC driver. The proposed system mostly deals with Online Analytical Processing (OLAP) on the Hadoop platform using Apache Kylin. Kylin is an OLAP engine that builds OLAP cubes from the data present in the hive and stores the cubes in HBase for further analysis. The cubes stored in HBase are analyzed by firing SQL-like analytics queries. Also, reports and dashboards are further generated by querying the underlying cubes that provide powerful insights into the data [7].

Air Quality is a big question in these times and to reduce air pollution, we are looking forward to making an air quality monitoring system. This system will be taking the data from sensors such as PM2.5. This web application is using Hadoop and Apache Kylin as a backend to hold a huge amount of data generated from sensors, and they will be processed with BI tools and different important insights are visualized using an interactive dashboard. As of now, the Java Spring Boot framework is used to develop the system. Our motive for this

Younten Tshering, Associate Lecturer, Department of Information Technology, Jigme Namgyel Engineering College, Royal University of Bhutan
Suyogya Ratna Tamrakar, Department of Information and Communication Technologies, Master's in Computer Engineering, Asian Institute of Technology
Shubhangini Gontia, Department of Information and Communication Technologies, Master's in Computer Science, Asian Institute of Technology,
Smrity Baral, Department of Information and Communication Technologies, Master's in Computer Science, Asian Institute of Technology,

work is to analyze the air quality data from the sensor and give a clear vision in a way of a dashboard. Therefore, we have developed a web app which will show all the required details related to air quality and accordingly people can take preventive measures. For this, we will receive sensor data from nodes or stations with different types of data such as text value and number and process using Big Data Tool.

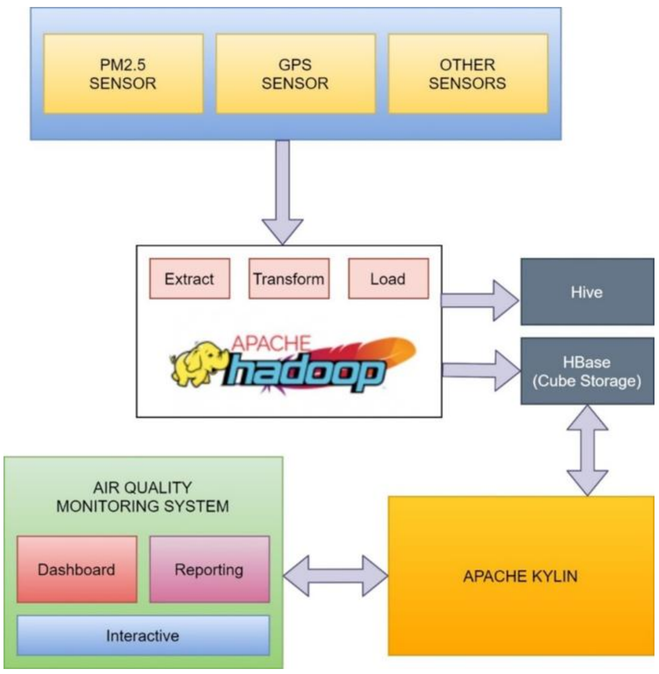


Figure 1 System Architecture Design

II. Related Work

Apache Kylin is an open-source steam engine created to offer a SQL interface and multi-dimensional analysis on Hadoop supporting extremely large datasets. In addition, it integrates with BI tools via ODBC driver, JDBC driver, and REST API. It was established by eBay in 2014, progressed to Top-Level Project of Apache Software Foundation just one year later, in 2015, and won the Best Open-Source Big Data Tool in 2015 as well as in 2016 [1].

Currently, it is being used by thousands of companies worldwide as their critical analytics application for Big Data. While other OLAP engines struggle with the data volume, Kylin enables query responses in milliseconds [1]. It provides sub-second level query latency over datasets scaling to petabytes. It gets its amazing speed by precomputing the various dimensional combinations and the measure aggregates via Hive queries and populating HBase with the results. Apache Kylin lets us query billions of rows at sub-second latency in 3 steps by [3] identifying a Star/Snowflake Schema on Hadoop, building Cube from the identified tables, and query and gets outcomes in sub-second, via ODBC, JDBC, or RESTful API.

Kylin is centered on the bulk synchronization processing (BSP) model to process graph data. Although Kylin is also based on the BSP model, its implementation strategy and optimization techniques distinguish Kylin from other existing systems. [5] highly optimize the BSP engine in Kylin to achieve better performance, in terms of data partitioning, message passing, and memory management.

Now, instead of going on to Apache Kylin, we should understand Big Data, which led to the development of Hadoop and then to Apache Kylin. IoT [4] connects the physical device to the internet and makes it smarter. Now, we have intelligent air conditioners, televisions, etc. Our intelligent air conditioner continuously monitors our room temperature and accordingly determines what should be the temperature of the room. Now guess how much data would be generated by smart air conditioners installed in tens & thousands of houses. By this, we can understand how IoT is contributing a major share to Big Data.

There were three major challenges with Big Data [9] as the first challenge is storing a massive amount of data. Storing massive data in a traditional system is not possible. The reason is obvious, the storage will be limited to one system and the data is increasing at an enormous rate. The second challenge is storing heterogeneous data. The data is not only massive, but it also appears in numerous formats. So, we need to make sure that we have a system to store various types of data that are produced from various sources. Lastly, the third challenge, which is the processing speed. Now the time taken to process this vast amount of data is extremely high as the data to be managed is too large.

To solve the storage problem and processing concern in [8] two core components were created in Hadoop HDFS (Hadoop Distributed File System) and YARN (Yet Another Resource Negotiator). HDFS solves the storage problem as it stores the data in a distributed manner and is effortlessly scalable. And YARN solves the processing concern by lowering the processing time significantly. While setting up Hadoop, we have a choice of selecting a lot of services as part of the Hadoop program, but two services are always compulsory for setting up Hadoop. One is HDFS (for storage) and the other is YARN (for processing).

The steps in [6] and in figure 2 show by what means Kylin gets the data and saves the results. First, sync the input source table. In best cases, it reads data from Hive, next it runs map-reduce / spark jobs to pre-calculate and create each level of cuboids with all possible combinations of dimensions and determine all the metrics at different levels. Ultimately, it stores cube data (aggregated result) in HBase where the dimensions are row keys and measures are column families. After aggregated data is ready, we can integrate with popular BI tools, such as Tableau or Superset, or connect with JDBC from our code.

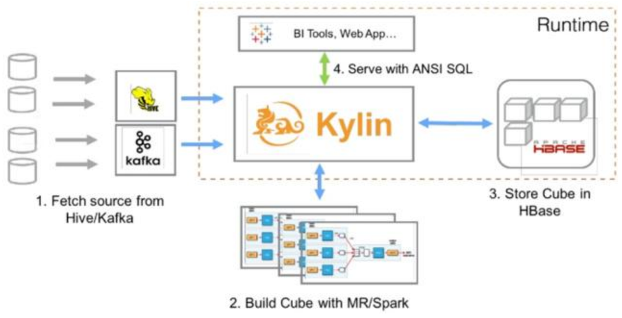


Figure 2 Apache Kylin – Architecture

There has been a lot of study on Hadoop as an option for data warehousing. Only storing the data in the data warehouse is not profitable to the business until the main data is evaluated to gain business visions. Online Analytical Processing (OLAP) is a method to answer multi-dimensional analytical queries quickly and provides support for decision-making and intuitive result views for queries [7]. There is a need to design a system for solving the challenges of computing OLAP data cubes over Big Data. In paper [7] proposed a system that overcomes some critical problems faced by conventional data warehousing and OLAP. The proposed system is beneficial to businesses in many scenarios, to help them make sensible and profitable decisions for their business.

III. Methodology

For developing the system, software lifecycle activities were considered with a set of activities and their relationships to each other to support the development of a software system. With the design, we implemented mapping models to java code and mapping models to a relational schema. As in paper [11], we followed the UML methodology, which has enhanced the process of understanding the system and architecture's design to be robust to control concurrency.

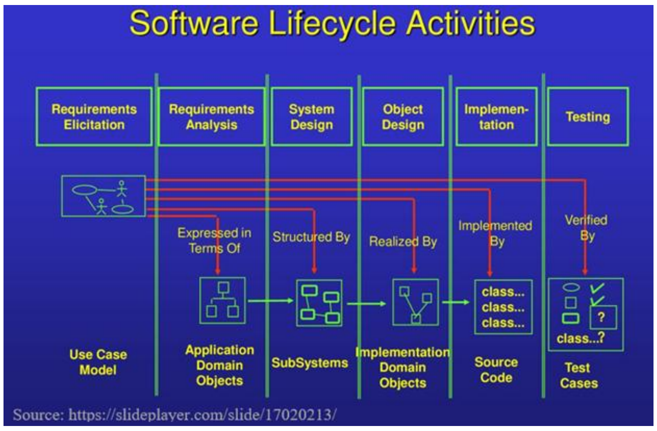


Figure 3 Software Lifecycle Activities

A. *Functional Requirements*

1) Visualization Module

The end-users should be able to see an interactive dashboard of air quality monitoring with different forecasts and insights. The system should be able to stream real-time data from different nodes and stations.

2) System Admin Module

The admin should be able to log in, log out to the system and modify the system parameters, and toggle dashboard controls. The admin should be able to register new sensors and manage the sensors in the system. The admin should be able to generate reports of specific periods and export those in various formats.

3) Data Collection Module

Data will be extracted from sensors and stored in Hadoop which is acting as a data warehouse using Hive. Kylin does aggregation functions on the cube (HBase) and provides the required parameters to the system.

B. *Non-functional Requirements*

We have focused on the application domain and solution domain which we must implement. With that focus we have the following design goal:

1) High Performance

The system should handle concurrent or multiple information from sensors that are placed at different locations in real-time. Large numbers of data collected from the sensor should be displayed on the responsive web application and we need to focus on concurrency and response time.

2) High Reliability

Reliability is one of the key attributes of the system. Back-ups will be made regularly so that restoration with minimal data loss is possible in the event of unforeseen events. For example, the system should be able to restore backward data of 24 hours (maximum 3 Days) within 30 minutes as a recovery function.

3) Portability

The system will run on multiple computers with different browsers. As it is a web application, mobile phone web browsers can also access the application. The web app should be responsive.

4) Usability

Once the system is deployed, the maintenance of the system including tasks such as monitoring the system, repairing problems that arise, updating or upgrading software components, should be easy to be sufficiently performed by any person with a basic understanding of the dashboard system. The web app should be easy to operate by users with a certain navigation menu or option and no need for a user manual.

Based on the functional requirement and non-functional requirements, we have system models and dynamic models where it presents a list of the fundamental sequence diagrams and use-cases that satisfy the system's requirements. The purpose is to provide an alternative, "structural" view of the requirements stated above and how they might be satisfied in the system.

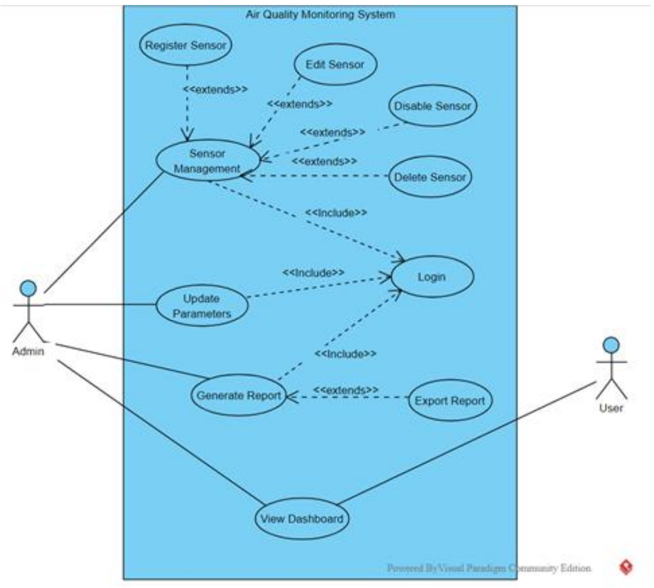


Figure 4 Use case diagram

The Air Quality Monitoring System will take the data from sensors such as PM2.5. This web application will be using

Hadoop and Apache Kylin as a backend to hold a huge amount of data from sensors, and they are processed with BI tools and different important insights are visualized using an interactive dashboard. Therefore, figure 5 will show the system architecture that is used to implement.

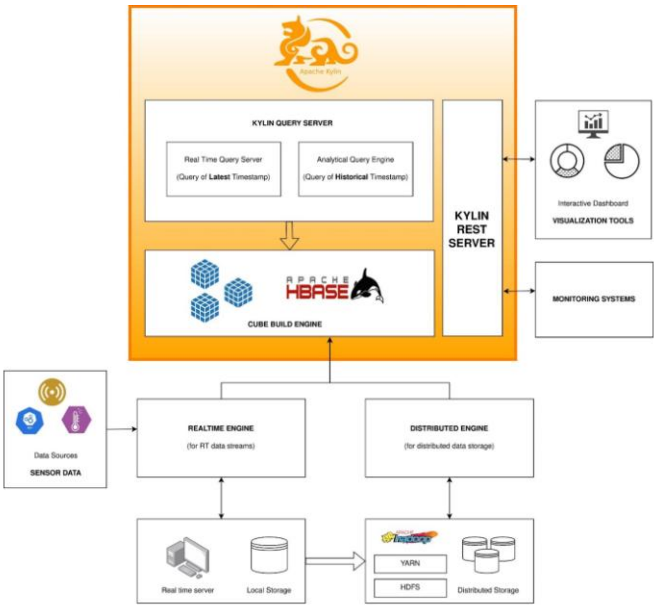


Figure 5 System Architecture

IV. Results

The air quality monitoring system is a web-based system that visualizes details about PM2.5 to show users the condition of air quality at various locations. It is a user-friendly application allowing the user to interact with the application naturally and intuitively. Air Quality Monitoring system can be used by any users who wish to know about the quality of air of the location within the premises of various provinces. There are two users: Admin and Visitors. These users have their accesses based on their privileges (Website privileges based on access control shown in table 1).

Access control and security of the system

Only admin login is required for the setup of the parameters and another dashboard page is available to everyone who visits the website. The dashboard will be accessed by all the users who visit the website. Although the system settings will only be accessed by the admin. Admin can set the parameters for the dashboard. If the normal user tries to access through login they will not be until they have admin rights (have used the spring security). Admin needs the authentication with username and password to access and control the system.

Table 1 Access Matrix

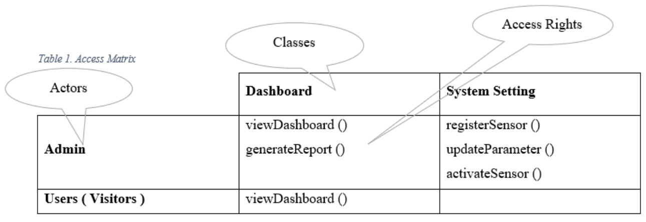


Table 1. Access Matrix
Actors
Classes
Access Rights

	Dashboard	System Setting
Admin	viewDashboard () generateReport ()	registerSensor () updateParameter () activateSensor ()
Users (Visitors)	viewDashboard ()	

For Visitors

Users have the privilege to visit the system without registration /login. Users can simply go to the site URL making a session to view the website and its contents. The webpage included a dashboard that contains a descriptive overview of PM2.5 and its intensity in various locations.
The application contains a dashboard that includes a Map view containing the value of PM2.5 of various locations with various colors used to represent the danger level of PM2.5. Here Green indicates it is a safe zone, yellow means good, orange means moderate and red means it is getting worse. The average value of PM2.5 in various provinces is based on available data and stations. The bar graph shows the most polluted provinces based on the values of PM2.5. Bar graph showing values of least polluted provinces based on the values of PM2.5.

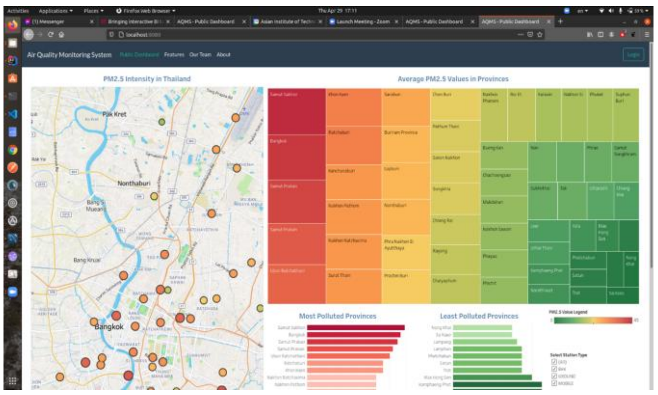
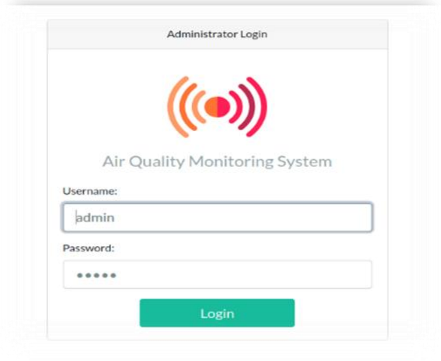


Figure 6 Dashboard view for visitors or Home Page of application

For Admin

This system provides more privileges to Admin making it easy for admin to handle the web application. Admin can simply go on the URL and log in with the credentials. Upon login, the admin will be redirected to the admin Homepage. The homepage of the admin contains a dashboard similar to visitors. In addition to this, there are some menu options inside the navigation bar.


Administrator Login

Air Quality Monitoring System

Username: admin

Password: •••••

Login

Figure 7 Login page for admin

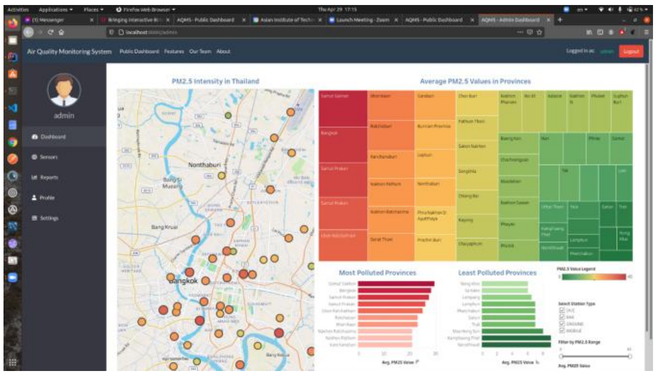


Figure 8 Homepage for Admin

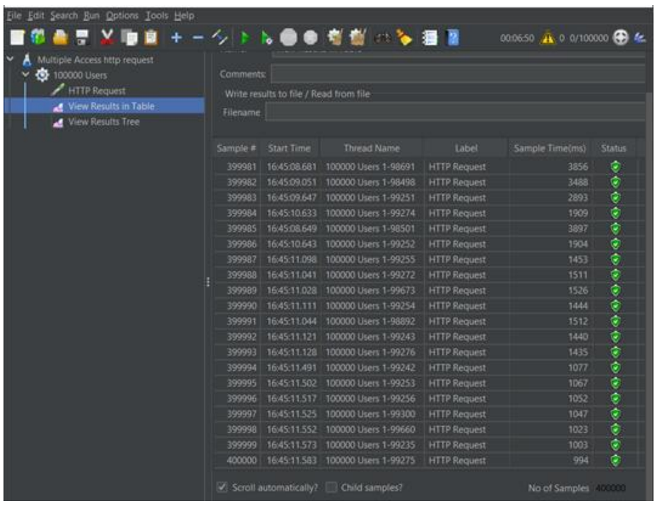
Admin can see various menus on the left navigation bar. The *dashboard* is like the visitor that contains all the features mentioned above. *Sensors* are for sensor management where the admin can edit, disable, or delete sensors as per the need. For now, the admin can start and stop the server as per the need.
Reports are basically about generating necessary reports based on the available information. It may contain forms fields like Title of Report, Descriptive Content, Export File Format, select sensor option, date, and generate submit button. For now, we have the option to export CSV files. *Profile* allows the admin to update the information about the admin like picture, name, email, job description, etc. *Parameter Update* allows admin to modify dashboard tiles, dashboard layout, manage data and data source with an available menu like Enable/Disable Dashboard Tiles, Modify Dashboard Layout, Manage Data Sources, and Data Backup/Restore. *Logout* menu simply makes admin log out from the admin web page and redirect to the visitor page.

V. Testing

The application testing applied to our system (AQMS) for concurrent programs is the main challenge, as handling multiple concurrent transactions and real-time information updates must be addressed properly. Considering the relevance of concurrent programs testing, several research have been conducted in this area, especially involving adaptation of the techniques and criteria applied in sequential programs. The objective of concurrent programming is to increase application performance, allowing better use of available resources. A concurrent application consists of several processes that interact to solve a problem, reducing the computational time in several application domains. For this testing, we are concerned about putting our application into load testing, to prove that the desired qualities are satisfied.

A. Test multiple access using JMeter

We configured 5000 threads in 2 loops, making it 10000 threads or users access the web app. Our application was competent to handle the multiple access over 10000 and we tested till 400000.


File Edit Search Run Options Tools Help
00:06:50 0 0/100000
Multiple Access http request
100000 Users
HTTP Request
View Results in Table
View Results Tree
Comments:
Write results to file / Read from file
Filename

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status
399981	16:45:08.681	100000 Users 1-98691	HTTP Request	3856	✓
399982	16:45:09.051	100000 Users 1-98498	HTTP Request	3488	✓
399983	16:45:09.647	100000 Users 1-99251	HTTP Request	2893	✓
399984	16:45:10.633	100000 Users 1-99274	HTTP Request	1909	✓
399985	16:45:08.649	100000 Users 1-98501	HTTP Request	3897	✓
399986	16:45:10.643	100000 Users 1-99252	HTTP Request	1904	✓
399987	16:45:11.098	100000 Users 1-99255	HTTP Request	1453	✓
399988	16:45:11.041	100000 Users 1-99272	HTTP Request	1511	✓
399989	16:45:11.028	100000 Users 1-99673	HTTP Request	1526	✓
399990	16:45:11.111	100000 Users 1-99254	HTTP Request	1444	✓
399991	16:45:11.044	100000 Users 1-98892	HTTP Request	1512	✓
399992	16:45:11.121	100000 Users 1-99243	HTTP Request	1440	✓
399993	16:45:11.128	100000 Users 1-99276	HTTP Request	1435	✓
399994	16:45:11.491	100000 Users 1-99242	HTTP Request	1077	✓
399995	16:45:11.502	100000 Users 1-99253	HTTP Request	1067	✓
399996	16:45:11.517	100000 Users 1-99256	HTTP Request	1052	✓
399997	16:45:11.525	100000 Users 1-99300	HTTP Request	1047	✓
399998	16:45:11.552	100000 Users 1-99660	HTTP Request	1023	✓
399999	16:45:11.573	100000 Users 1-99235	HTTP Request	1003	✓
400000	16:45:11.583	100000 Users 1-99275	HTTP Request	994	✓

☑ Scroll automatically? ☐ Child samples? No of Samples 400000

Figure 9 Result of multiple user access

B. Test Concurrent access

We configured 1000 group threads in 1 second, creating concurrent threads or users access the web app. Our application was able to manage more than 1000 concurrent access.

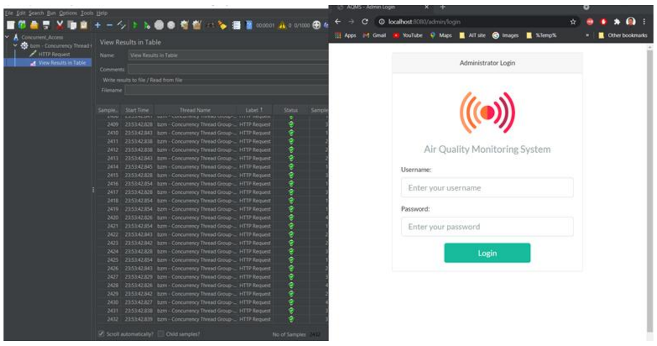
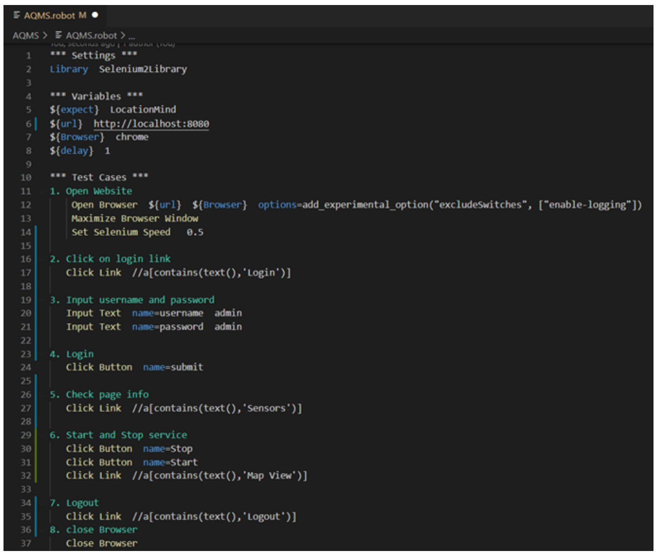


Figure 10 . Result of concurrent user access

C. Automation Test using Robot Framework.

Robot Framework is a test automation framework that is Python-based. This framework supports writing an object-page model in keyword-driven methodology. Robot Framework allows users to write their test cases without programming knowledge. We test Robot with simple web app access by writing the script in VS Code with robot file extension and running it will show our test case pass or fail. In addition to this, it has very descriptive logs including complete debugging capabilities through readable logs. The following figure shows the test case implemented in the robot framework where the test data is in a simple and easy-to-edit format. When Robot Framework is started, it processes the test data, executes test cases, and generates logs and reports.
The test case for checking automation is the user will open website, then login into the system as an administrator, click and open some options and Start/Stop service to pull sensor data, then logout from the system and at the end close the browser after checking or monitoring the air quality.

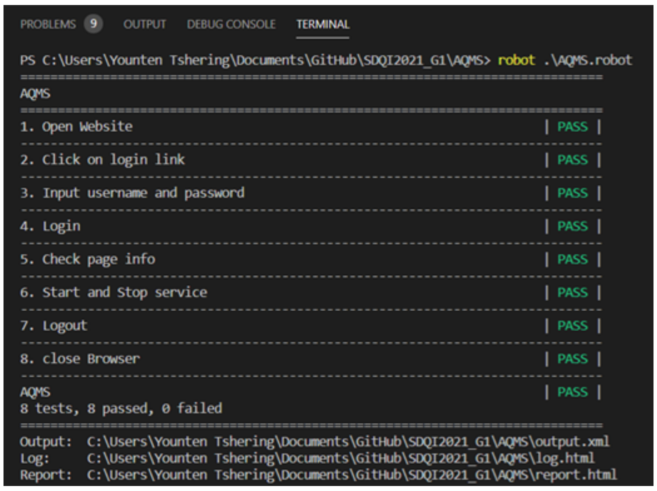


```
AQMS.robot M ●
AQMS > AQMS.robot > ...
*** Settings ***
Library  Selenium2Library

*** Variables ***
${expect}  LocationMind
${url}  http://localhost:8080
${Browser}  chrome
${delay}  1

*** Test Cases ***
1. Open Website
    Open Browser  ${url}  ${Browser}  options=add_experimental_option("excludeSwitches", ["enable-logging"])
    Maximize Browser Window
    Set Selenium Speed   0.5

2. Click on login link
   Click Link  //a[contains(text(),'Login')]

3. Input username and password
   Input Text  name=username  admin
   Input Text  name=password  admin

4. Login
   Click Button  name=submit

5. Check page info
   Click Link  //a[contains(text(),'Sensors')]

6. Start and Stop service
   Click Button  name=Stop
   Click Button  name=Start
   Click Link  //a[contains(text(),'Map View')]

7. Logout
   Click Link  //a[contains(text(),'Logout')]
8. close Browser
   Close Browser
```

Figure 11 Test case for login and start/stop service

Once we have the test case, then we can run the test and check the test pass or fail. The figure given below shows that all the test cases created were passed and the automation worked perfectly for that scenario.

Robot Framework provides good support for external libraries, tools that are open source and can be used for automation. The most popular library used for web development and UI testing is Selenium which works together with Robot Framework. After the test is performed, it generates output, log, and report files to check the output in detail and if there are any fail cases in the test then a screenshot is also generated to know from where the test failed.



```
PROBLEMS 9   OUTPUT   DEBUG CONSOLE   TERMINAL

PS C:\Users\Younten Tshering\Documents\GitHub\SDQI2021_G1\AQMS> robot .\AQMS.robot
==============================================================================
AQMS
==============================================================================
1. Open Website                                                       | PASS |
------------------------------------------------------------------------------
2. Click on login link                                                | PASS |
------------------------------------------------------------------------------
3. Input username and password                                        | PASS |
------------------------------------------------------------------------------
4. Login                                                              | PASS |
------------------------------------------------------------------------------
5. Check page info                                                    | PASS |
------------------------------------------------------------------------------
6. Start and Stop service                                             | PASS |
------------------------------------------------------------------------------
7. Logout                                                             | PASS |
------------------------------------------------------------------------------
8. close Browser                                                      | PASS |
------------------------------------------------------------------------------
AQMS                                                                  | PASS |
8 tests, 8 passed, 0 failed
==============================================================================
Output:  C:\Users\Younten Tshering\Documents\GitHub\SDQI2021_G1\AQMS\output.xml
Log:     C:\Users\Younten Tshering\Documents\GitHub\SDQI2021_G1\AQMS\log.html
Report:  C:\Users\Younten Tshering\Documents\GitHub\SDQI2021_G1\AQMS\report.html
```

Figure 12 Result of test case in VScode

For more details on system testing report, design document, user guide to follow or gain experiences of the system and system documentation for installation and configuration you can follow the link given below:
https://github.com/Younten-Tshering/Projects_and_related_works/tree/main/PM2.5%20Monitoring%20System/Related%20Documents/Final%20Report%20Document

VI. DISCUSSION

For the project, we have used two ways to visualize our data, one way was Tableau and Kylin connecting with the JDBC driver which had some compatibility issues at the end and for ODBC we had to contact the vendor. Since we knew the cube structure and table format, we made the same format in the MySQL database so that we can test the visualization and show the Tableau connection. The data in the database was also fetched using the python script on a real-time basis (hourly from API). For testing purposes, we have used JMeter and Robot framework to show high performance and usability.

Figure 13 represents our system, subsystems, their interconnections, and workflow. Whenever a client calls sensors data sources via API, it receives the sensor's station data as a response to the call. This retrieved data is stored in the internal RDBMS of the client itself. Since the data is usually huge as sensors data are updated on an hourly basis, this data is stored in HIVE. Our system checks for new updated data and updates in the HIVE table are made accordingly. Then comes the Kylin that builds a cube from the data in the hive table. Cubes are formed based on various dimensions present in the table. Working in Kylin makes data processing fast and easy to analyze.

Once the Kylin forms the cubes then these cubes can be further used by the tableau tool which is used for data visualization and data analysis. Kylin ODBC drivers that work as a connector between Kylin and Tableau and via this driver, Tableau gets live data from Kylin. In the tableau, data analysis and data visualization are performed. Our system is for the air quality monitoring system, so it creates a visualization that is based on the PM2.5 value of different locations. The visualization output of the Tableau tool is embedded into the web application that is developed using Spring Boot in Java. Whenever a client uses the web application entering the URL, the mentioned process takes place, and the application displays the dashboard as output in the webpage.

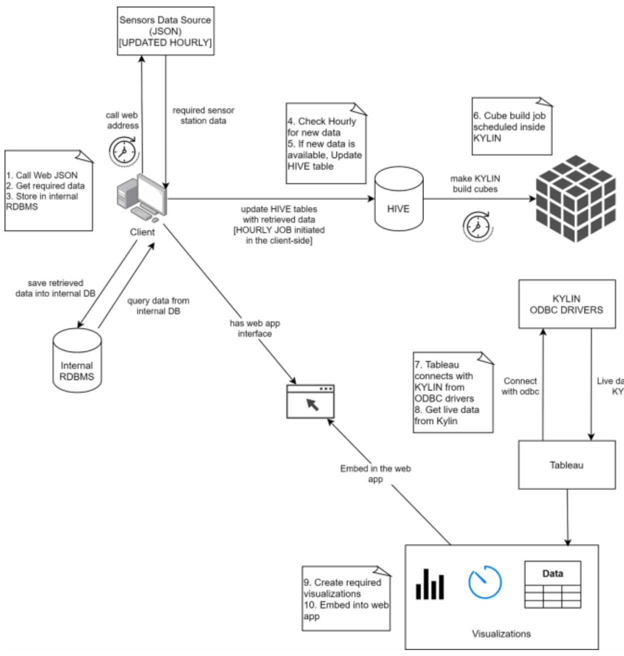

Sensors Data Source (JSON) [UPDATED HOURLY]
call web address
required sensor station data
1. Call Web JSON
2. Get required data
3. Store in internal RDBMS
Client
4. Check Hourly for new data
5. If new data is available, Update HIVE table
update HIVE tables with retrieved data [HOURLY JOB initiated in the client-side]
HIVE
6. Cube build job scheduled inside KYLIN
make KYLIN build cubes
save retrieved data into internal DB
query data from internal DB
Internal RDBMS
has web app interface
KYLIN ODBC DRIVERS
7. Tableau connects with KYLIN from ODBC drivers
8. Get live data from Kylin
Connect with odbc
Live data from KYLIN
Tableau
Embed in the web app
9. Create required visualizations
10. Embed into web app
Data
Visualizations

Figure 13 . Initial System Service details diagram which did not work because of lack of driver

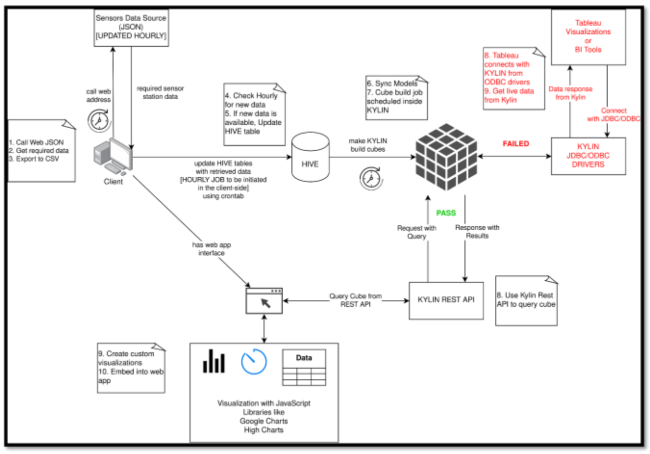

Sensors Data Source (JSON) [UPDATED HOURLY]
call web address
required sensor station data
1. Call Web JSON
2. Get required data
3. Export to CSV
Client
4. Check Hourly for new data
5. If new data is available, Update HIVE table
update HIVE tables with retrieved data [HOURLY JOB to be initiated in the client-side] using crontab
HIVE
make KYLIN build cubes
6. Sync Models
7. Cube build job scheduled inside KYLIN
FAILED
PASS
8. Tableau connects with KYLIN from ODBC drivers
9. Get live data from Kylin
Tableau Visualizations or BI Tools
Data response from Kylin
Connect with JDBC/ODBC
KYLIN JDBC/ODBC DRIVERS
Request with Query
Response with Results
has web app interface
Query Cube from REST API
KYLIN REST API
8. Use Kylin Rest API to query cube
9. Create custom visualizations
10. Embed into web app
Data
Visualization with JavaScript Libraries like Google Charts High Charts

Figure 14 Final System Service details diagram

For the project, we have implemented the idea as shown in the architecture or Kylin workflow in [1] where we first fetch the data from the sensor through our python script. We then take the CSV file or SQL file into Hive (Hadoop) and we can see how many rows are there. After this, we have our Kylin server which we must sync with the model and create a cube or refresh if we have the earlier one saved in HBase. To visualized data, there are two ways as follows:

1) BI tools: Tableau

For this, we must connect Kylin and Tableau with JDBC/ODBC driver.

2) Third-Party App

For this, we must use the REST API to connect.

We have used both ways since we had some problems with Tableau and Kylin getting to connect since JDBC drivers were not compatible and for ODBC we had to contact the vendor. In the end, we have implemented the REST API to connect with our app and get the required data.

VII. Conclusion

Poor air quality has become a serious hazard to human health these days. People must be made aware of this condition via any channel. So, we have come up with a web application for addressing air quality issues. The motive of this application is to present the recent scenario of the air quality based on PM2.5 of various locations. This system uses the actual data from various sensors located in different areas and based on the data obtained, it visualizes the condition of air quality level as a dashboard in the web application. This application is supposed to address the unhealthy situation of air. Such information can be used by anyone, even non-tech knowledge people, without the need for any registration allowing all concerned people to take precautions for their health. We expect this system to be user-friendly and useful to all the concerned users.

The Air Quality Problem is diverse. Many factors are recognized or yet to be recognized. In our application, we have only used PM2.5 as a factor to indicate the quality of the air in a given location. This has made the scope of application smaller. Some gaps can be fulfilled by extending this application in the future. We hope to consider other factors like other air pollutant gases SO2, CO, CO2, wind speed, and direction as well as trying to cover more geographical areas with more sensors and data. But still, we believe that our system will provide some convenient information to the users, making people conscious about the quality of air.

Glossary

Here are some domain terms we are using:

- *Framework* serves as a foundation for programming that acts as a platform for developing software.
- Concurrency is the tendency for things to happen at the same time in a system. Concurrency is when multiple sequences of operations are run in overlapping periods of time.
- *Coherence* defines the degree to which the elements of a module belong together. Thus, cohesion measures the strength of relationships between pieces of functionality within a given module
- *APIs* stands for Application Programming Interface and is defined as a set of programming code that enables data transmission between one software product and another. An API is a way to programmatically interact with a separate software component or resource
- *Dashboard* is an information management tool that visually tracks, analyzes and displays key performance indicators (KPI), metrics and key data points to monitor the health of a business, department or specific process.
- *Data Warehouse*: a data warehouse (DW or DWH), also known as an enterprise data warehouse (EDW), is a system used for reporting and data analysis.
- *Business Intelligence*: Business intelligence (BI) is the set of techniques and tools for the transformation of raw data into meaningful and useful information for business analysis purposes.
- *OLAP Cube*: an OLAP cube is an array of data understood in terms of its 0 or more dimensions.
- *Dimension*: A dimension is a structure that categorizes facts and measures in order to enable users to answer business questions. Commonly used dimensions are people, products, place and time.

Acknowledgment

The success and outcome of this work required immense guidance and assistance from many people, and it is always a pleasure to remind the fine people of the sincere guidance we received to uphold our work. With much respect and reverence, we would like to pay our heartfelt gratitude to professor Apichon Witayangkurn for his constant encouragement, support, and guidance who helped us in completing our work. Lastly, we would also like to acknowledge ourselves for demonstrating exceptional team spirit and teamwork.

References

[1] Apache Kylin. (2015). Bring OLAP back to big data! Retrieved from Apache Kylin | Analytical Data Warehouse for Big Data

[2] Fann,N.,& Risley,D. (2011,January 5). The public health context for PM2.5 and ozone air quality trends. Air Qual Atmos Health 6, 1–11 (2013). https://doi.org/10.1007/s11869-010-0125-0

[3] Geetha,S.M.N. (2021, March 19). Hadoop for Analyst-Apache Druid, Apache Kylin and Interactive query tools. Retrieved from https://www.saigeetha.in/post/hadoop-for-analysts-apache-druid-apache-kylin-and-interactive-query-tools?fbclid=IwAR0RRXXxKmv8onswnS-g5mV5Hh_L5R9zOSWly6YO8d4kb6oYYW4rrjF5wlo

[4] Gupta,A.k., & Johari,R. (2019). IOT based electrical device surveillance and control system. International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), https://doi.org/10.1109/IoT-SIU.2019.8777342

[5] L. Ho, T. Li, J. Wu and P. Liu, "Kylin: An efficient and scalable graph data processing system," 2013 IEEE International Conference on Big Data, 2013, pp. 193-198, doi: 10.1109/BigData.2013.6691574.

[6] Neethu, M.V. (2018, September 25). OLAP on Hadoop-Apache Kylin. Retrieved from https://medium.com/@mvneethu90/olap-in-hadoop-apache-kylin-bf0377d8b44f

[7] Ranawade, Supriya & Navale, Shivani & Dhamal, Akshay & Deshpande, Kuldeep & Ghuge, Chandrashekhar. (2017). Online Analytical Processing on Hadoop using Apache Kylin. International Journal of Applied Information Systems. 12. 1-5. 10.5120/ijais2017451682.

[8] Sinha, S. (2016, October 28). Hadoop ecosystem- Get to know the Hadoop tools for crunching Big Data. edureka. Retrieved from https://medium.com/edureka/hadoop-ecosystem-2a5fb6740177

[9] Sinha,S. (2014, October 9). Hadoop tutorial- A comprehensive guide to Hadoop. edureka. Retrieved from https://medium.com/edureka/hadoop-tutorial-24c48fbf62f6

[10] Supasri, Titaporn & Sampattagul, Sate & Macatangay, Ronald. (2020). Particulate matter (PM2.5, PM10) monitoring low cost sensor data: a case study in Southern Thailand.

[11] Tshering, Younten & Tamrakar, Suyogya & Gontia, Shubhangini. (2021). Bid Buy-Sell system using client-server architecture, solution of concurrent users for the web application through optimistic locking and multithreading. Volume 9,. 2977-2986. Retrieved from http://www.ijaresm.com/bid-buy-sell-system-using-client-server-architecture-solution-of-concurrent-users-for-the-web-application-through-optimistic-locking-and-multithreading

Younten Tshering is an Associate Lecturer in the Department of Information Technology, Jigme Namgyel Engineering College, Royal University of Bhutan. He has completed B.Sc. in Computer Science with honors and received an Academic Excellence certificate. He is currently pursuing his Master of Computer Science with a Specialization in Software Engineering at the Asian Institute of Technology. Research interest is in software development & management and designing ontology for open data.

Suyogya Ratna Tamrakar is a software engineer with real world experience in developing mobile applications. He completed his B.E. in Computer Science from Kathmandu University, Nepal, and currently pursuing master's degree in Computer Engineering at Asian Institute of Technology, Thailand. His research is based on Natural Language Processing for Nepali. He is also an active member of Callijatra, a youth-led group in Nepal, who are working for preservation and revival of beautiful ancient scripts.

Shubhangini Gontia is pursuing her Masters of Computer Science with a Specialization in Software Engineering at the Asian Institute of Technology. She completed her B.E. in Computer Engineering from A.D. Patel Institute of Technology, Gujarat, India, and worked as a Quality Assurance Engineer for automation testing in Cybage Softwares for almost 3 years.

Smrity Baral worked as Assistant Lecturer in the Department of IT and Computer Engineering, Cosmos College of Management and Technology, Pokhara University, Nepal. She completed her bachelor's in computer engineering and is currently studying for a Master of Computer Science at Asian Institute of Technology in Thailand. Her research interests are in Data Management and Data Analysis.