Table of Contents

1. Introduction

1.1 Background

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 Royal Thai government to approve measures to prevent and address the on January 21, 2020.

PM 2.5 comes primarily from combustion - Fireplaces, car engines, and coal- or natural gas–fired power plants are all major PM 2.5 sources. PM stands for particulate matter (also called particle pollution): the term for a mixture of solid particles and liquid droplets found in the air. Some particles, such as dust, dirt, soot, or smoke, are large or dark enough to be seen with the naked eye.

Fine particulate matter (PM2.5) is an air pollutant that is a concern for people's health when levels in 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.

To measure this level in 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 the 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.

1.2 Purpose of the system

In this contemporary environment air pollution or air quality have 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. These fine particles can get deep into 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.

Our motive for this project was to analyze the air quality data from the sensor and give a clear vision in a way of dashboard. Therefore, we have developed the web app which will show all the required details related to air quality and accordingly people can take preventive measures.

For this we receive sensor data from nodes or stations with different types of data such as text value and number and process using Big Data Tool.

1.3 System goal

Air Quality is a big question in these times and to reduce the air pollution, we move forward to making an air quality monitoring system. This system will be taking the data from sensors such as PM2.5.

In case of AIT, this application can be used to monitor the air quality on campus. Once the application can give accurate and real time information on air quality from the data imported from sensors then other investors or different organizations can use it.

This web application will be using Hadoop and Apache Kylin as a backend to hold a huge amount of data from sensors, and they will be processed with BI tools and different important insights are visualized using an interactive dashboard. As of now, we have been using with Java Spring Boot framework for developing the system.

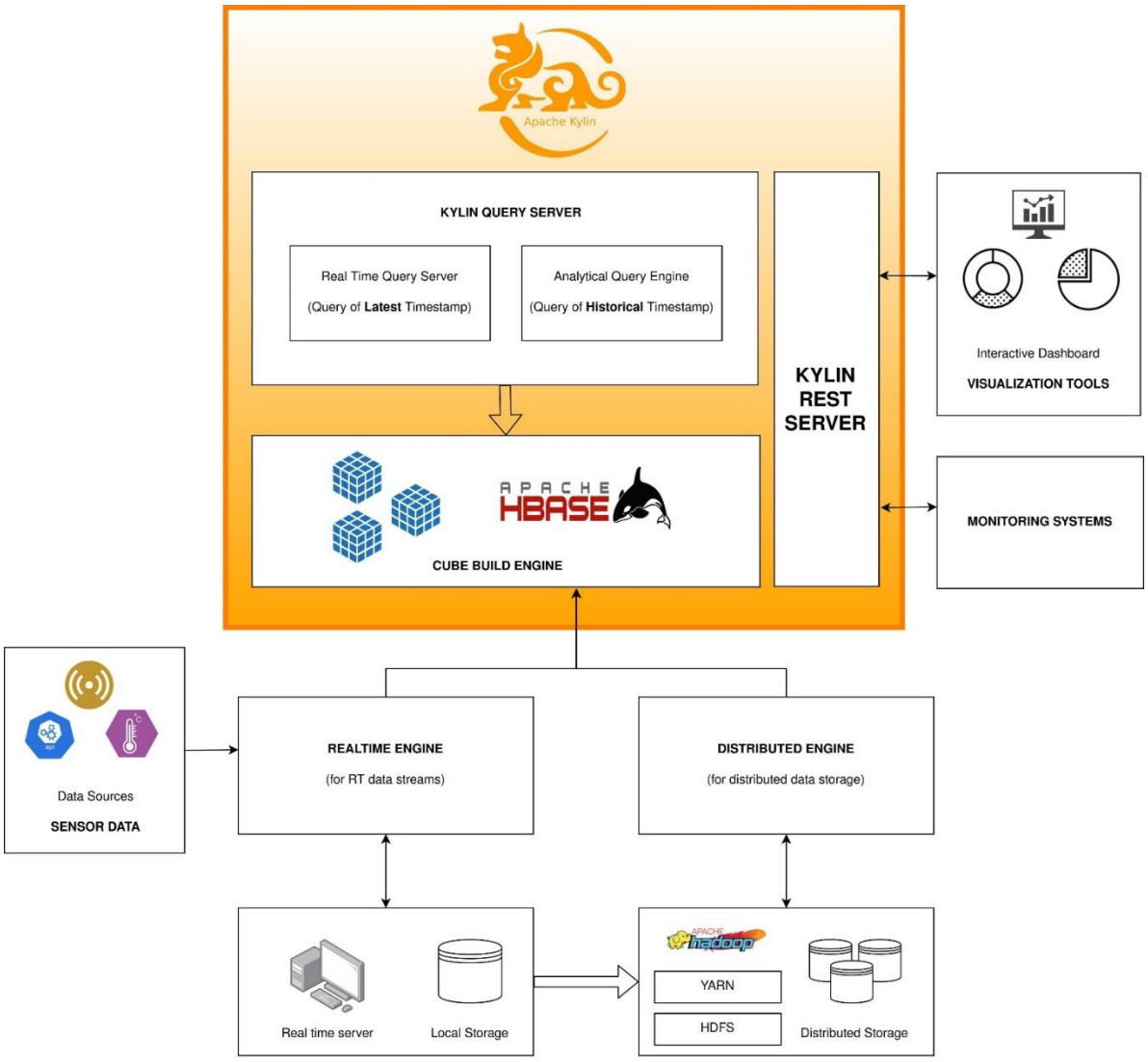


Figure 1. System Architecture

2. Functional Requirement

2.1 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.

2.2 System Admin Module

- The admin should be able to login, logout 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 system.
- The admin should be able to generate reports of specific time periods and export those in varies formats.

2.3 Data Collection Module

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

Note:

- ❖ Air quality and location details provided to the system from the sensor shall be stored in the database (Hadoop).
- ❖ The information shall be accessible via distributed system and JPA (Java Persistence API).
- ❖ The data stored should be able to be manipulated through interface.

3. Non-functional Requirements

3.1 Performance

The system shall be designed with high performance level to handle concurrent or multiple information from sensors which 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, response time and block time.

- ✓ The application should respond to a user within 2 seconds.

- ✓ The application should be able to handle 100 transactions per second in the peak load time.
- ✓ The application will be available with the uptime of 95% between 6:00 am to 1:00 am.

3.2 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. The system will also be thoroughly tested by all team members to ensure reliability.

- ✓ The system should be able to restore backward data of 24 hours (maximum 3 Days) within 2 hours as a recovery function.

3.3 Portability

The system shall be designed in a way that allows it to be 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 must support latest Web browsers for any OS.
- ✓ The web app should be responsive.

3.4 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. No need for a user manual.

3.5 Interoperability

For the application, we just import and integrate various information with different values into the system from the sensors. Therefore, we need to know to segregate the data and proceed forward.

3.6 Scalability

With data, the storage size will increase but can be managed with time. This app can be made horizontally scalable when there are issues of memory storage.

3.7 Reusability

The system should be designed in a way that allows the database to be re-used regularly for the various similar sensors that the organization shall hold.

4. System Design

This section 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.

4.1 Use Case Model

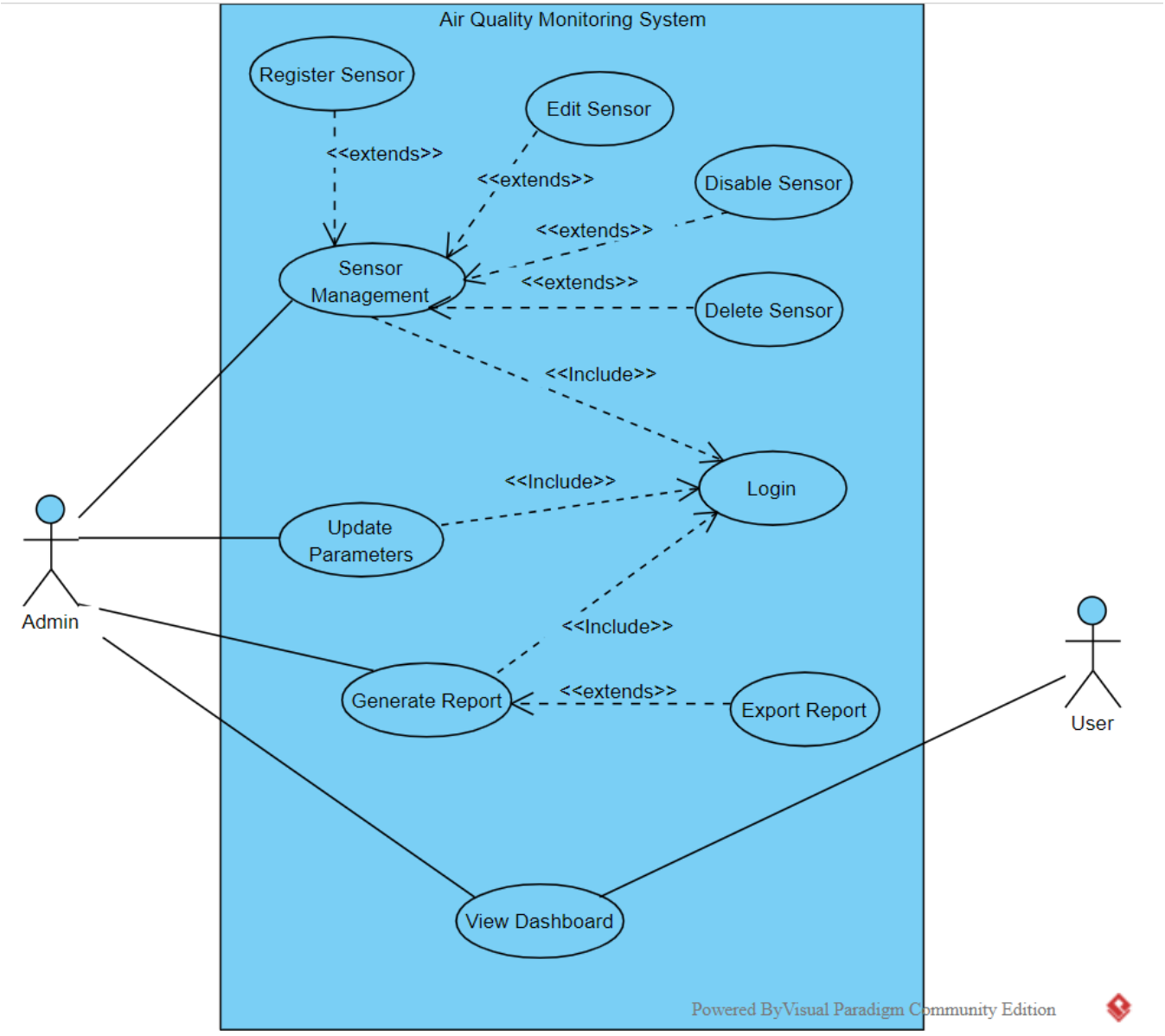


Figure 2. Use case diagram.

4.2 Dynamic models

4.2.1 System Sequence Diagram

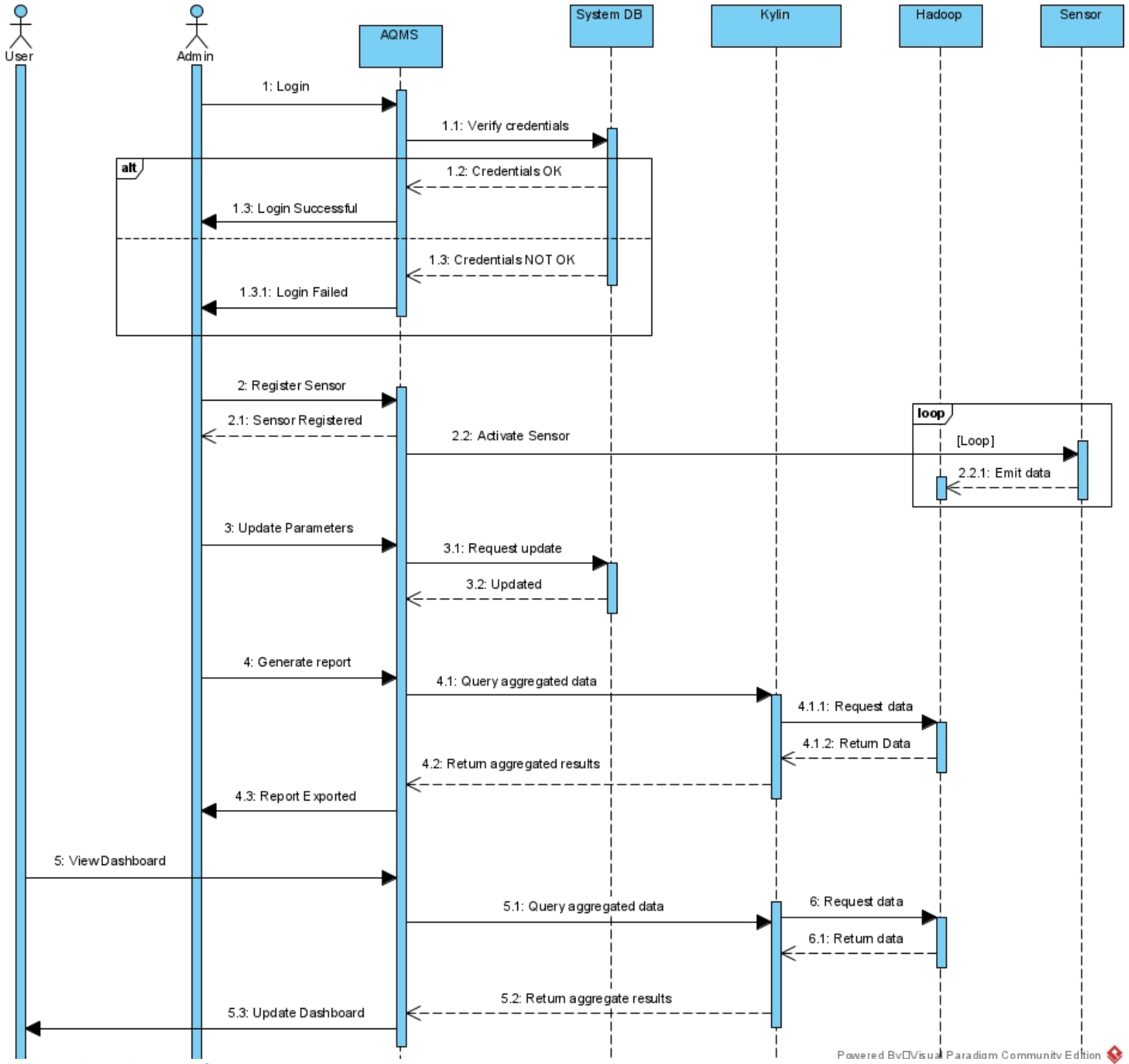


Figure 3. System Sequence diagram

4.2.2 State diagram

a) State diagram for user with internal process.

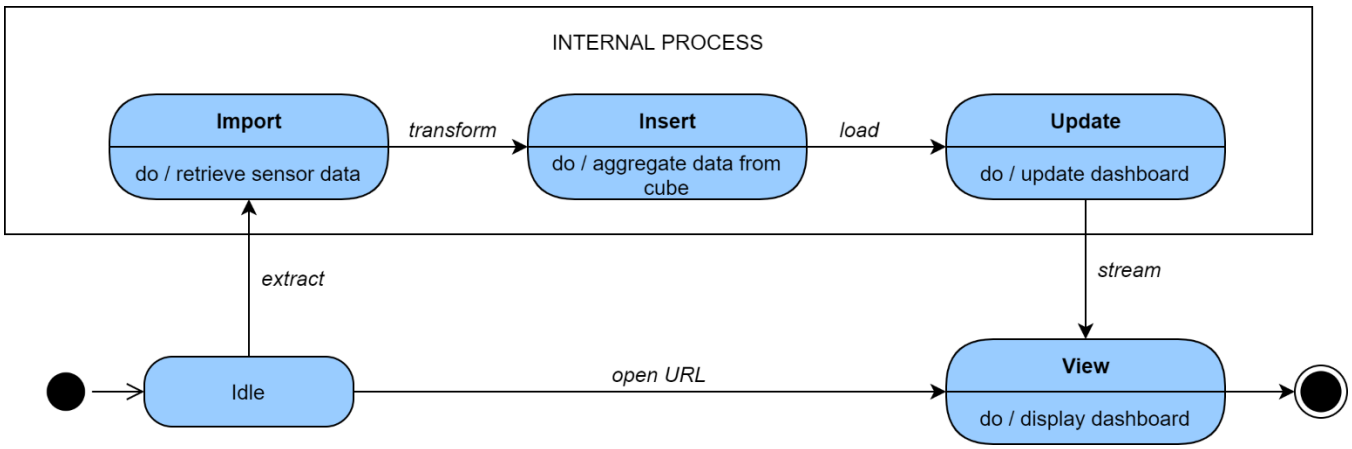


Figure 4. User State diagram

b) State diagram for admin with different state.

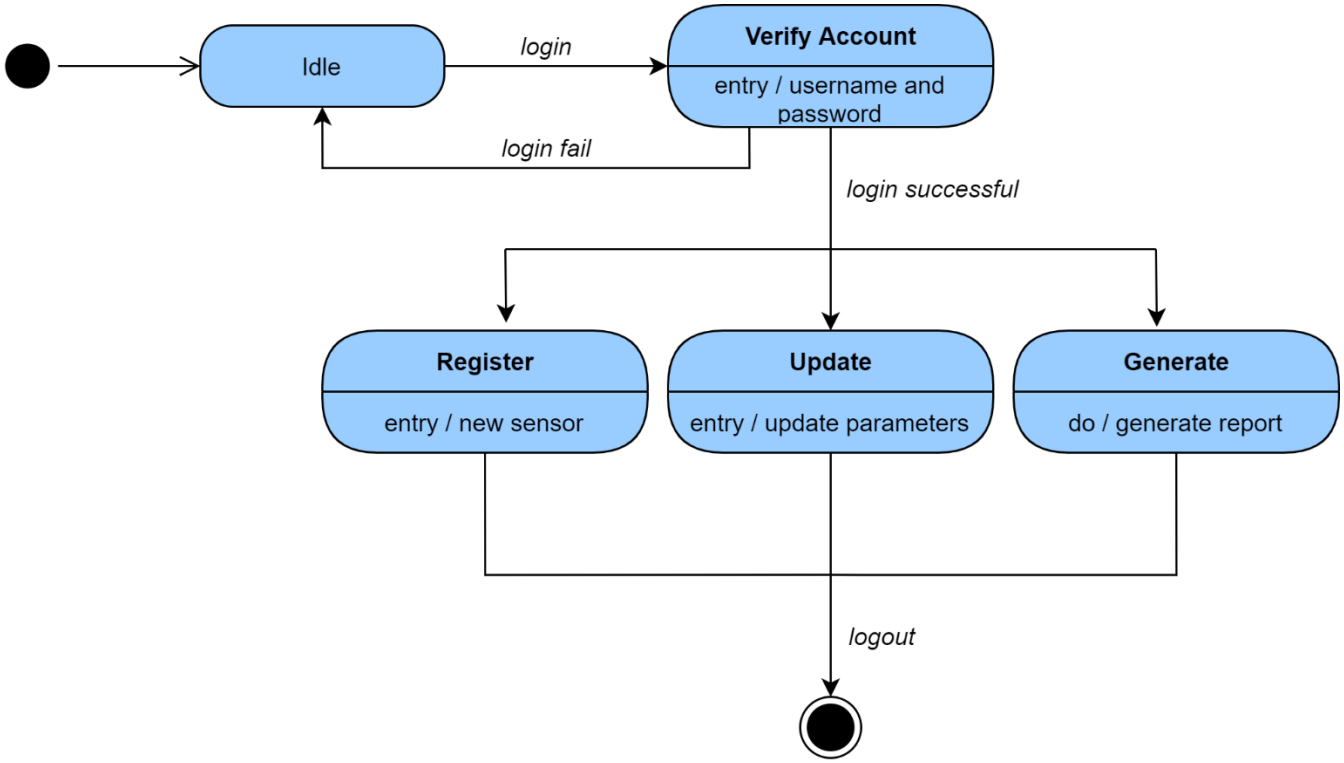


Figure 5. Admin State diagram

4.2.3. Activity diagram

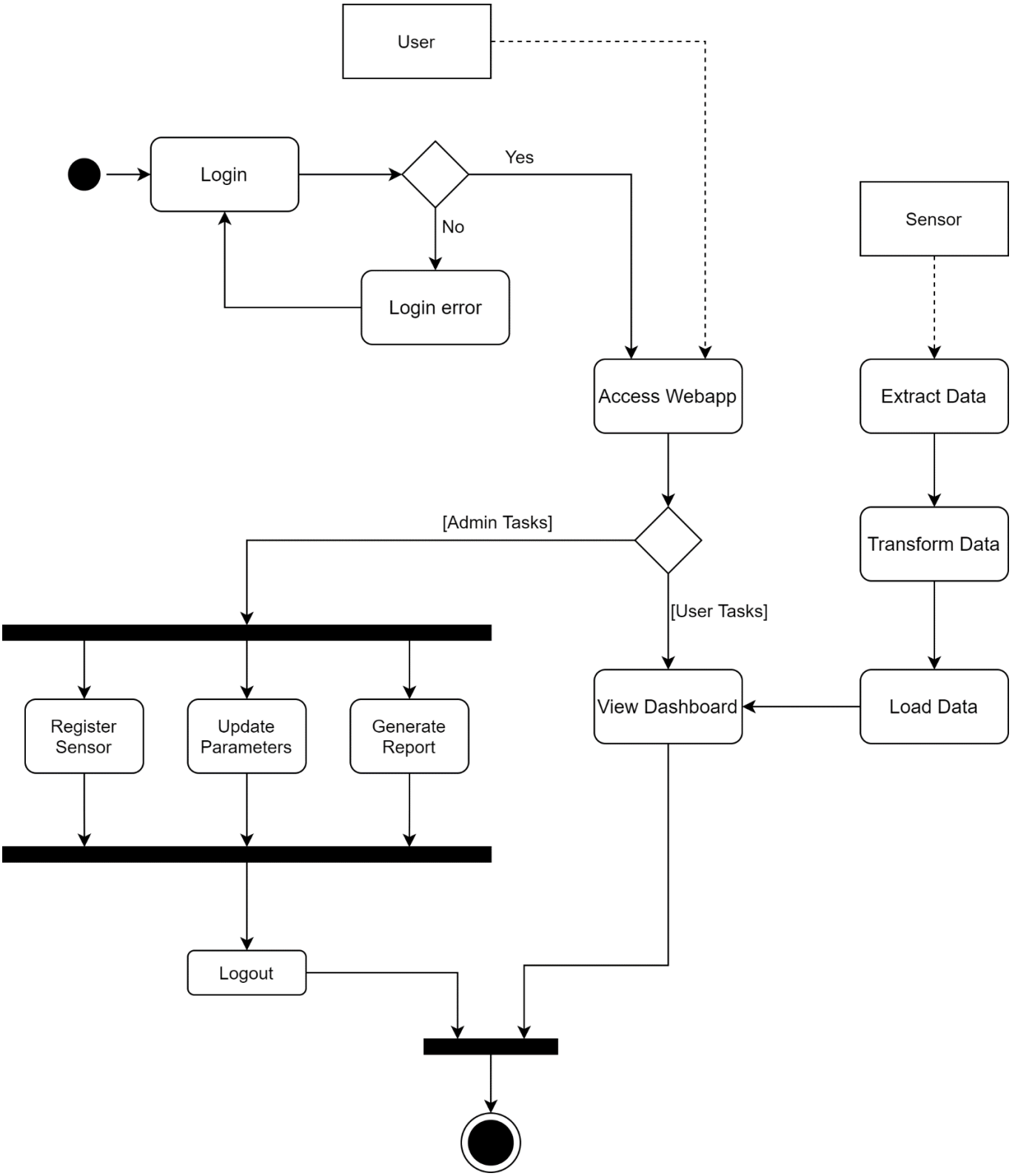


Figure 6. Activity diagram

4.3 Object and class model

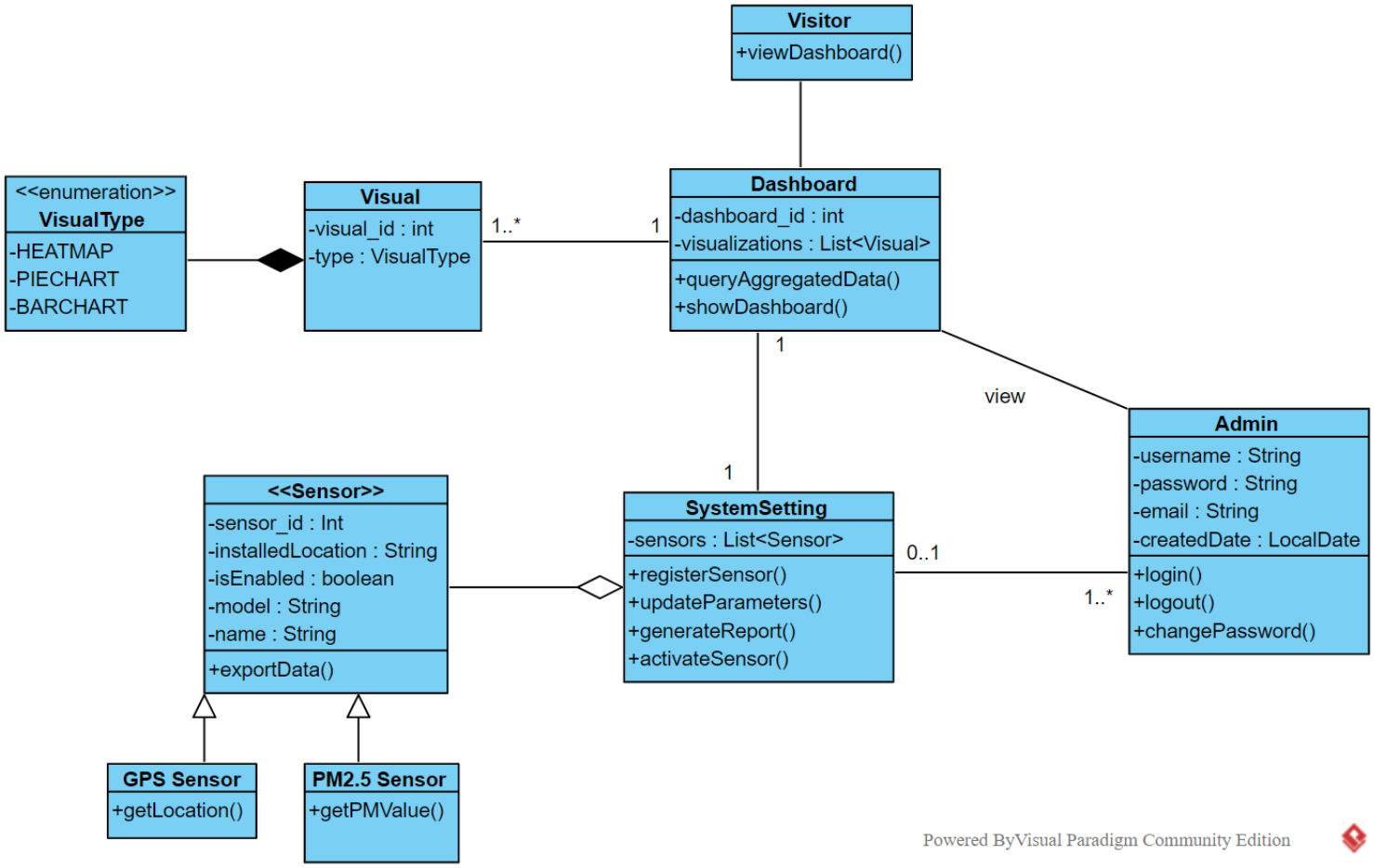


Figure 7. Class diagram

Notes:

Design Documents

Design documentation is a collection of documents and resources that covers all aspects of our product design. It includes all essential implementation details; and design decisions that our team have agreed on.

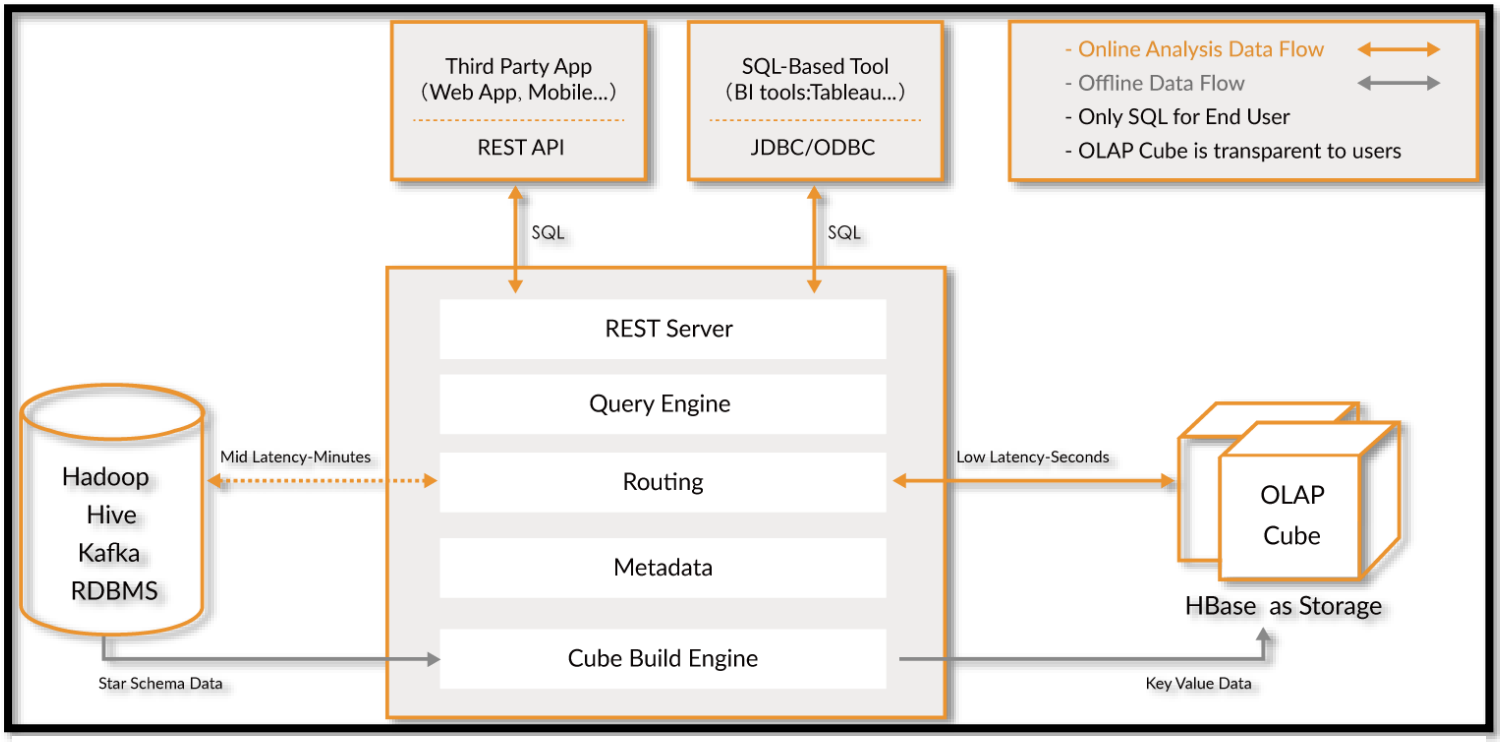


Figure 8.Kylin workflow

For our project, we have implemented the idea as shown in the above architecture or Kylin workflow where we first fetch the data from sensor through our python script (16. Appendix). 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 in localhost which we must sync with model and build cube or refresh if we have previous one which are saved in HBase. Now comes our how our data are visualized, there are two ways as follows:

1. BI tools: Tableau

   For this we have to connect Kylin and tableau with JDBC/ODBC driver.

2. Third Party App

   For this we have to use REST API to connect.

For our project, we have used both ways since we had some problem with Tableau and Kylin getting to connect since JDBC driver were not compatible and for ODBC we had to contact vendor. Now, we have implemented the REST API to connect and get the required data.

5. Design goals

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

5.1 High Performance

The system should handle concurrent or multiple information from sensors which 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.

5.2 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.

5.3 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.

5.4 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. No need for a user manual.

6. Software Architecture

6.1 Subsystem decomposition

After analysis the design model, we reduce system complexity while allowing high coherence dependency among classes where classes in a subsystem perform similar tasks and are related to each other via many associations.

Coupling measures dependency among subsystems, so we made low coupling since a change in one subsystem does not affect any other subsystem. If high coupling is there, then changes to one subsystem will have high impact on other subsystems.

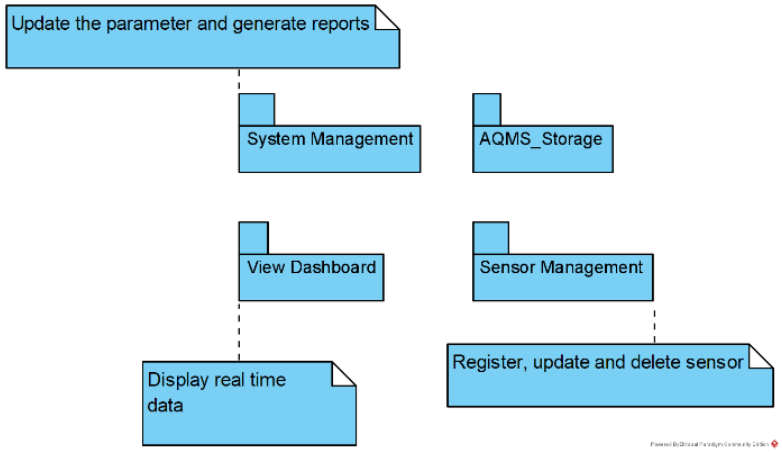


Figure 9. Subsystem decomposition

For our project, we analysis classes, associations, operations, events, and constraints that are closely interrelated with each other to achieve high cohesion and less coupling. In UML, subsystems are modeled as packages or components as shown in above diagram.

6.2 Hardware/Software mapping

For the mapping, in our project Hadoop will store the data in HIVE which is inserted by running python script. The cube will be from Kylin and store in HBase to be used by Tableau using ODBC driver which we could not implement since Drivers were not available. Therefore, we know that we must map our object to Tableau which will be connected to Kylin to fetch data. Network connections can be physical connectivity or logical connectivity. Connecting the software and hardware are difficult when addressing externally imposed hardware and software constraints. For example, Kylin running in docker since its requirement is high and connecting Kylin and Tableau was difficult since they operate in different OS.

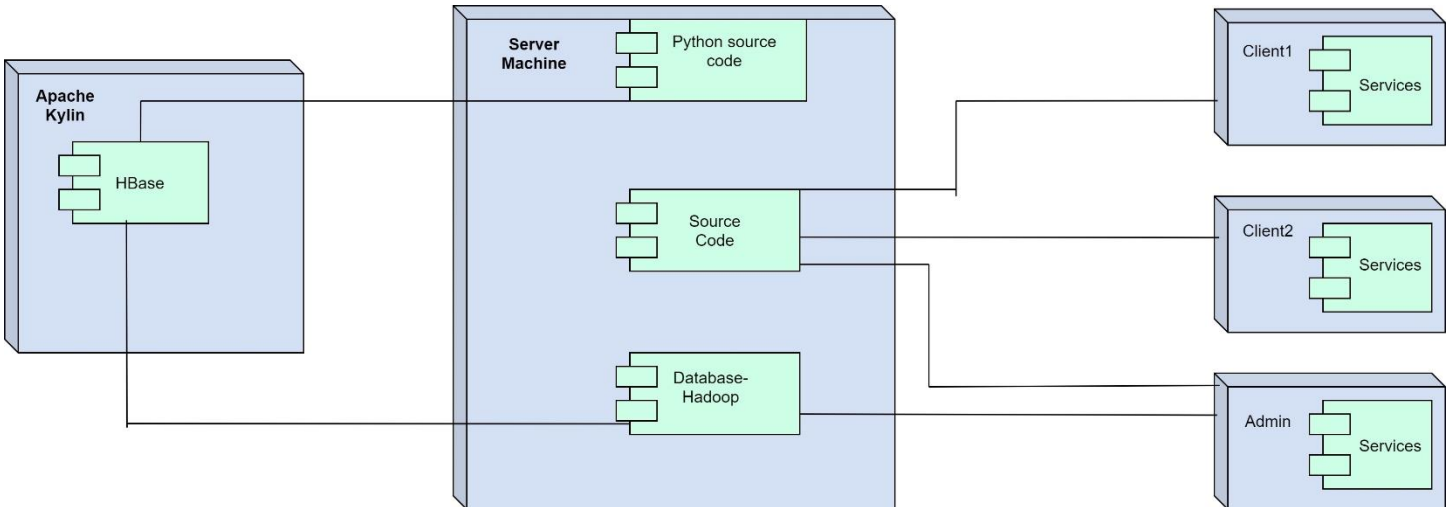


Figure 10. Hardware/Software Mapping

It basically includes how hardware and software are mapped together so that system functions efficiently. The figure 10 shows the hardware and software mapping of our application. Apache Kylin has HBase which is mapped with Server machine via python code and database (Hadoop). Similarly, the clients and admin with various services as per the privileges are mapped to server machine via source code that is Spring boot in java. This way, every hardware and software are mapped to each other in our system.

6.2.1 Deployment Diagram

Deployment diagram comes under structural diagrams that is used in modeling the physical aspects of an object-oriented system. Hardware and software of any system is visualize using deployment diagram. Our system contains Presentation server, Apache Kylin Server and Database server as main servers and Admin and client browsers to send request to the system for the service. When a request is sent by admin or client, Presentation server contains the website as GUI that receives it and then it sends http request to apache Kylin server where query and cube building engine are present. These engines are responsible for querying and cube building. Apache Kylin and Database server that contains PM2.5 data are connected via TCP/IP and this server communicates using this protocol. Thus, every hardware and software are connected to each other so that system runs properly and provides the expected output as intended.

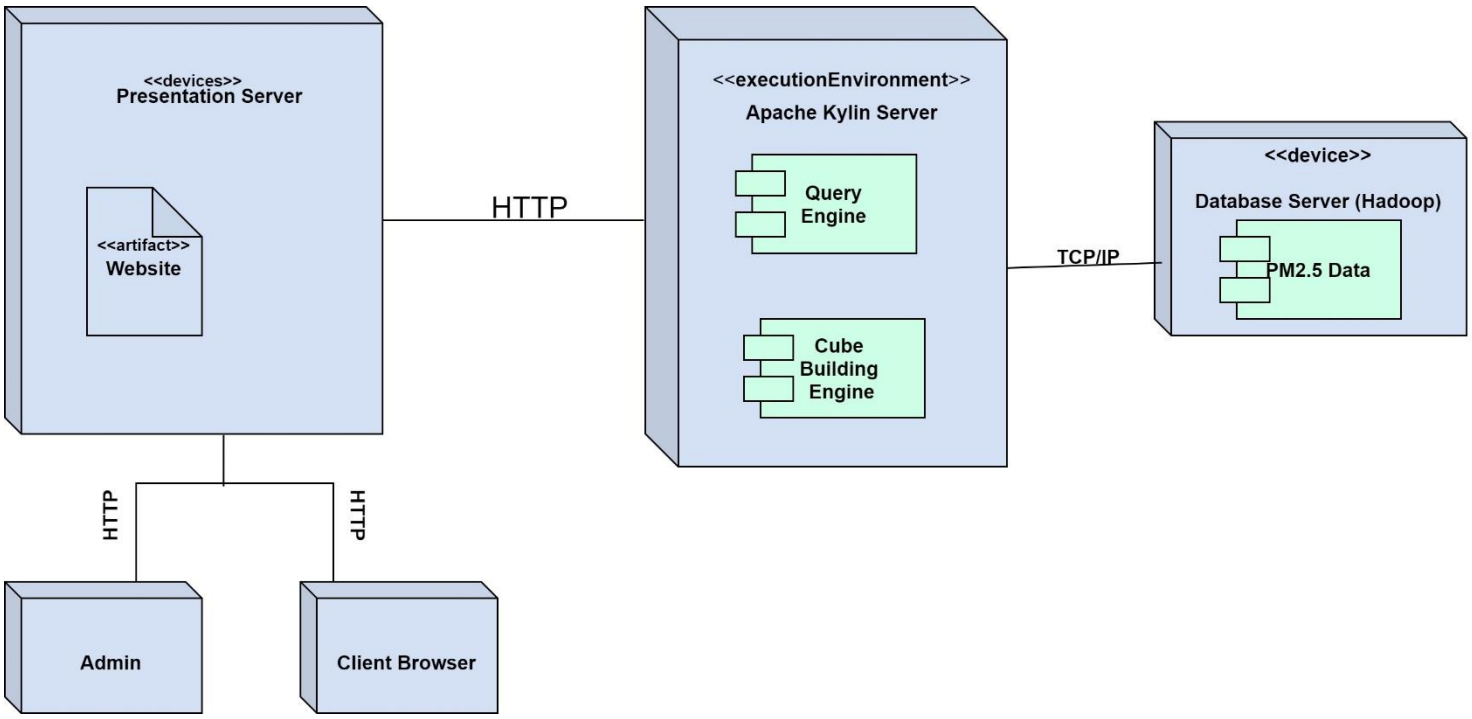


Figure 11.Deployment Diagram

6.2.2 Component Diagram

Component diagrams are used to visualize the organization of system components and the dependency relationships between them. They provide a high-level view of the components within a system. For our system, actor is the user that uses our application – Air quality monitoring system. Data obtained from sensor go to Big data system where process like storage, cube formation and querying of data is done. The application is connected to Big data system that visualizes the output from generated after analyzing the data.

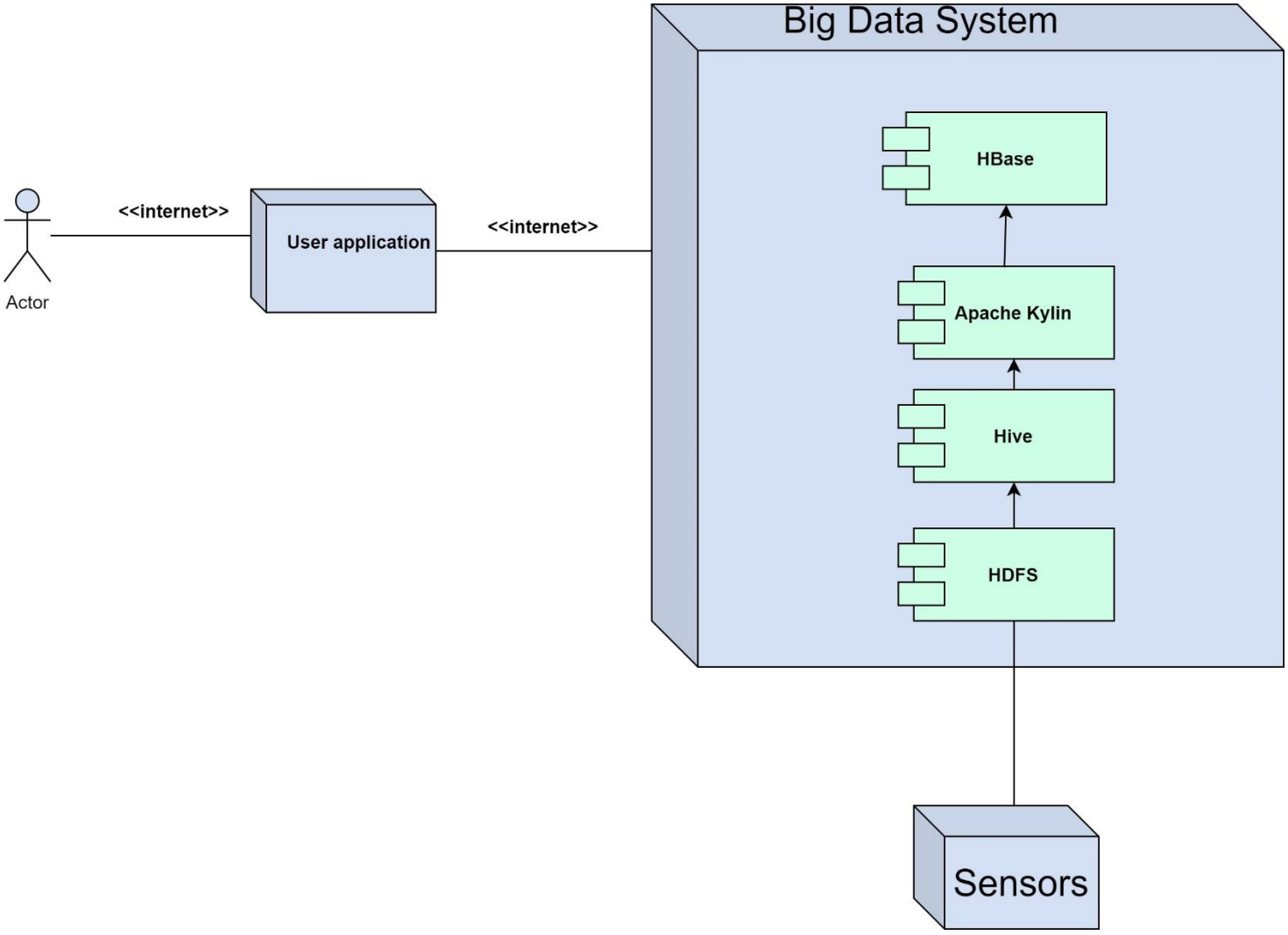


Figure 12. Component Diagram

6.3 Persistent data management

Some objects in the system model need to be persistent and values for their attributes have a lifetime longer than a single execution.

A persistent object can be realized with one of the following mechanisms:

1. Filesystem
2. Database

For our project we will be following Hadoop File system since data are used by multiple readers but a single writer (admin).

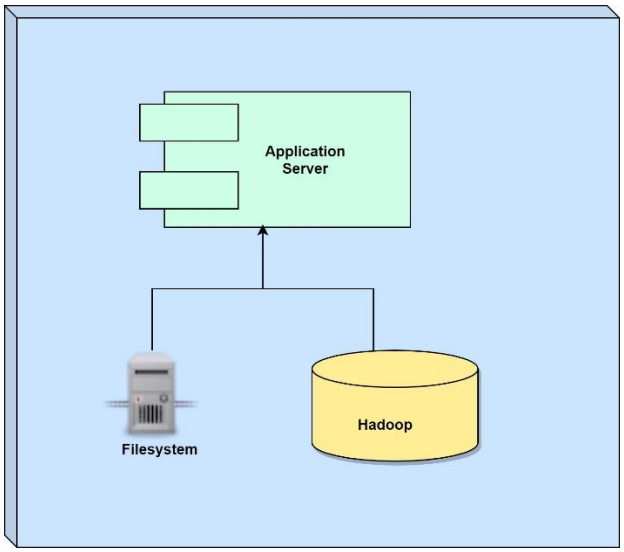


Figure 13. Data management

For Database, we might use it when data are used by concurrent writers and readers. UML object models can be mapped to relational databases by:

- Each class is mapped to its own table
- Each class attribute is mapped to a column in the table
- An instance of a class represents a row in the table
- One-to-many associations are implemented with foreign key
- Many-to-many associations are mapped to their own tables

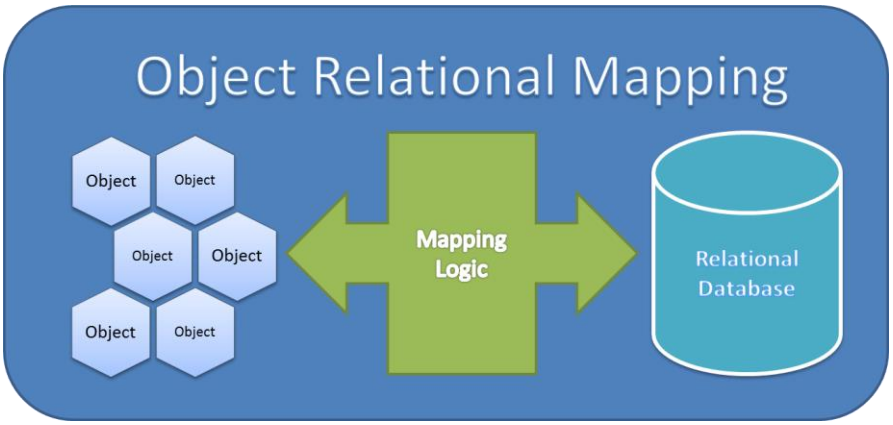


Figure 14. Mapping Object Model

6.4 Access control and security

- ✓ Discusses access control

  The only admin login is required for the setup of the parameters and other dashboard page is available to everyone who visits the website.

- ✓ Describes access rights for different classes of actors

  The dashboard will be accessed by all the users who visits our website. Although the system settings will only be accessed by admin. Admin can set the parameters for dashboard.

- ✓ Describes how object guard against unauthorized access.

  If the normal user tries to access through login, they will not be able to access until they have admin rights (have used the spring security).

- ✓ Does the system need authentication?

  Admin needs the authentication with username and password to access and control the system.

Access Rights

Classes

Table 1. Access Matrix

Actors

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

6.5 Boundary conditions

6.5.1 Initialization

- What data need to be accessed at startup time?

API will be called to fetch the sensor data to display real time at the dashboard. The data could be real time data or the visualization from aggregated queries.

- What services have to registered?

Sensors must be activated, and streaming APIs needs to be initialized to receive real-time updates. For data storage Hadoop server has to run which receives and sends data periodically to Kylin servers for cube building. Apache Kylin REST API will be used to perform aggregated queries as required.

- What does the user interface do at start up time?

It displays the latest updated values of database for PM2.5 sensor in the dashboard. If sensor streaming is active, it shows the real-time stream of the data values otherwise dashboard with aggregated query results.

6.5.2 Termination

- How are updates communicated to the database?

System is going to get the updates at specified time intervals.

6.5.3 Failure

- How does the system behave when a node or communication link fails?

It will retry to connect the call and fetch the data.

- How does the system recover from failure?

We will not have such failure, or the system recover is automatic since we are using Hadoop and it is having fault tolerance feature.

7. Subsystem services

The middleware subsystem services data requests between the user interface and the database subsystem.

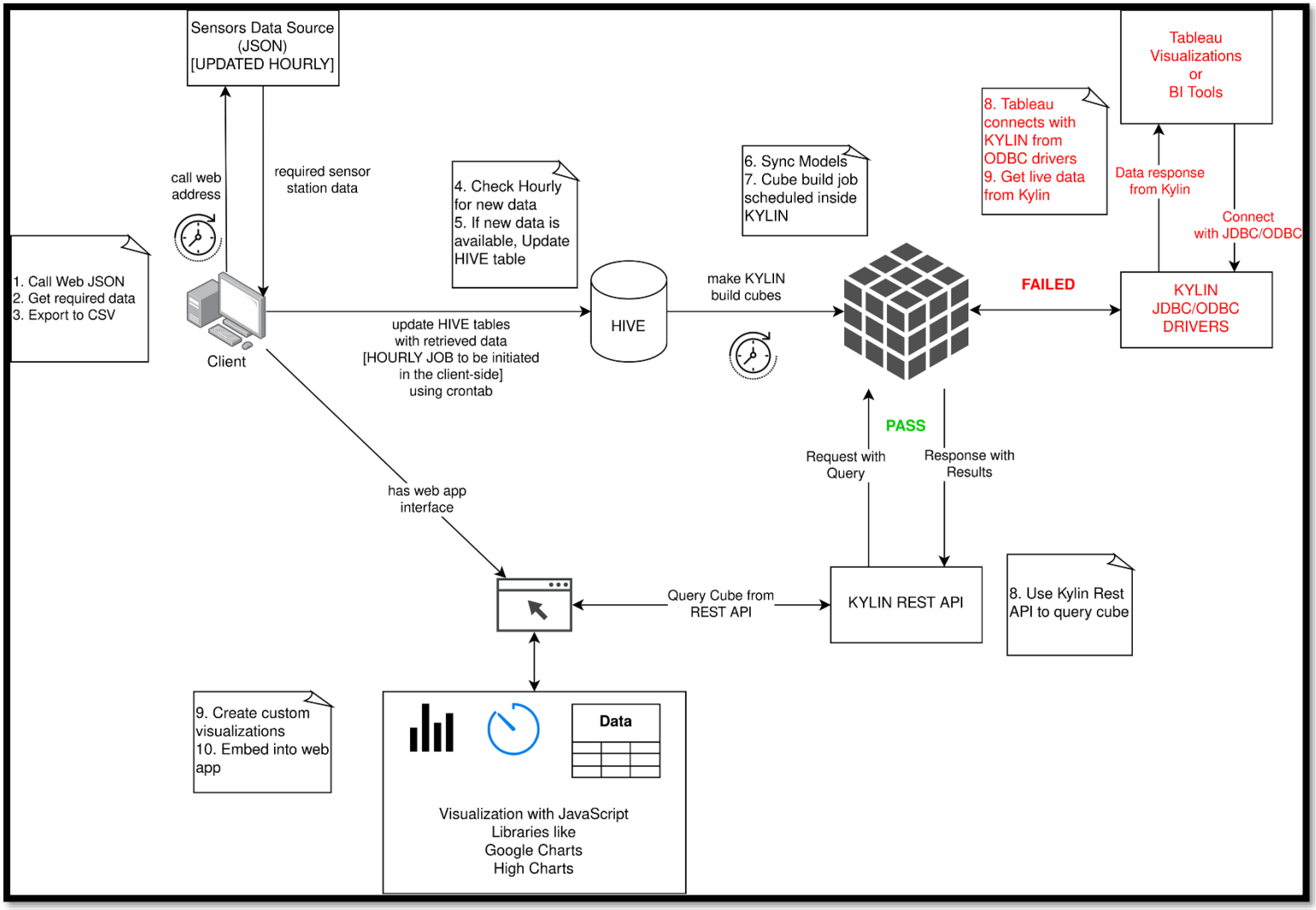


Figure 15. Final System Service details diagram

For our project, we have implemented the idea as shown figure 15 where we first fetch the data from sensor through our python script (16. Appendix). 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 in localhost which we must sync with model and build cube or refresh if we have previous one which are saved in HBase. Now comes out how our data are visualized, there are two ways as follows:

1. BI tools: Tableau

For this we have to connect Kylin and tableau with JDBC/ODBC driver.

2. Third Party App

For this we have to use REST API to connect.

For our project, we have used both ways since we had some problem with Tableau and Kylin getting to connect since JDBC driver were not compatible and for ODBC we had to contact vendor. Now, we have implemented the REST API to connect and get the required data.

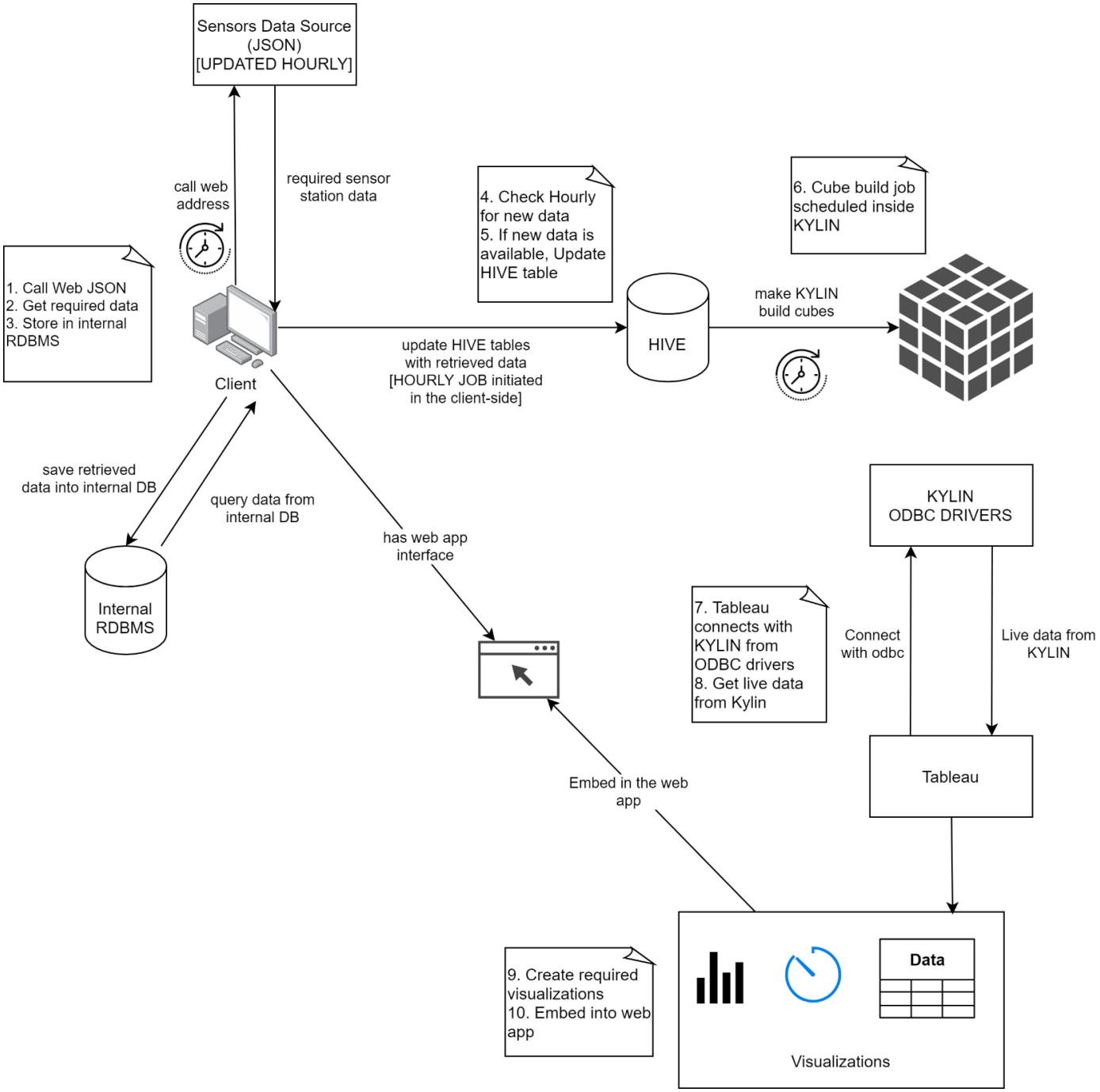


Figure 16. Initial System Service details diagram which did not work because of lack of driver.

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

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

8. Low level design

It refers to component-level design process that describes detailed description of every module.

8.1 Object design trade-offs

Table 2. Quality attribute analysis

Quality attributes	Priority Rating (L, M, H)	Priority Justification
Performance	H	The system shall be designed with high performance level to handle concurrent or multiple information from sensors which 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, response time and block time. ✓ The application should respond to a user within 2 seconds.

		✓ The application should be able to handle 100 transactions per second in the peak load time. ✓ The application will be available with the uptime of 95% between 6:00 am to 1:00 am.
Reliability	H	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. The system will also be thoroughly tested by all team members to ensure reliability. ✓ The system should be able to restore backward data of 24 hours (maximum 3 Days) within 2 hours as a recovery function.
Portability	M	The system shall be designed in a way that allows it to be 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 must support latest Web browsers for any OS. ✓ The web app should be responsive.
Usability	M	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. No need for a user manual.
Scalability	L	With data, the storage size will increase but can be managed with time. This app can be made horizontally scalable when there are issues of memory storage.

8.2 Final object design

After the hierarchy of subsystems has been developed, the objects in the system are identified and their details are designed. The emphasis shifts from application domain concepts toward computer concepts. The objects identified during analysis are drawn out for implementation with an aim to minimize execution time, memory consumption, and overall cost. Therefore, we have UML diagram which was discussed in

design document as a part of system model. The class diagram and sequence diagram have really help us in code base and to understand the operation of the system. Object design helps us in implementation of the system model based on design decision. We reused knowledge from previous experience such as discovering inheritance (specialization and generalization) and reuse functionality already available to build custom object.

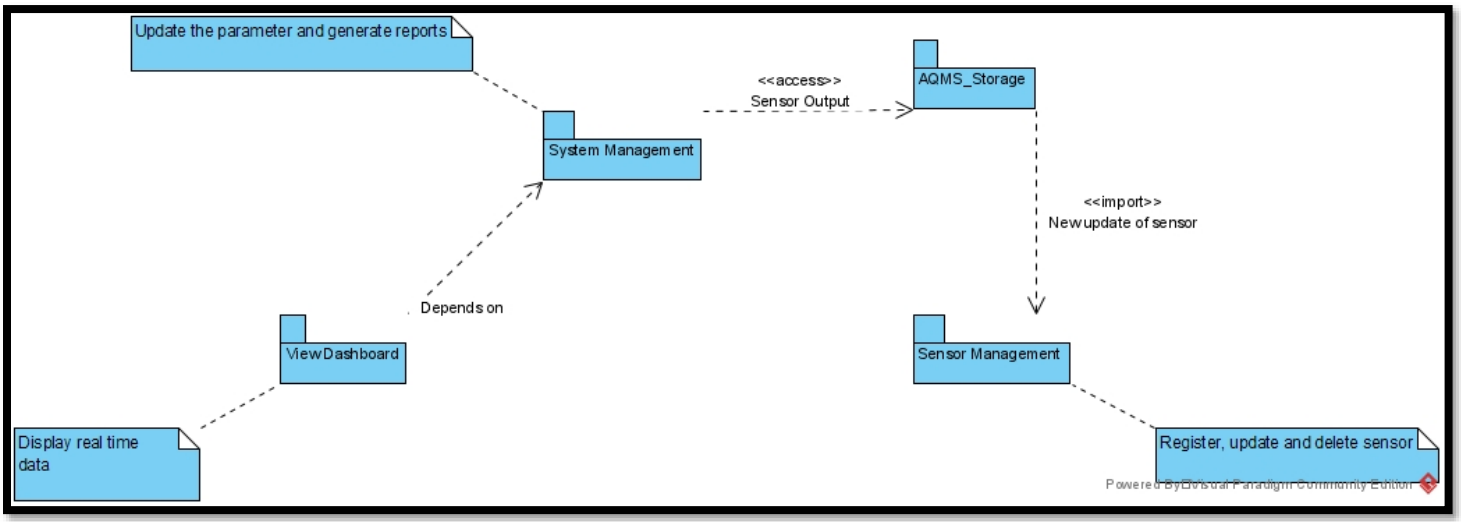


Figure 17. Object design model

8.3 Packages

A package is a logically grouping of related UML elements such as diagrams, documents, classes, or even other packages. During object design, classes and objects are grouped into packages to enable multiple groups to work cooperatively on a project. We can use the UML package mechanism to organize classes into subsystems.

Package diagrams are used, in part, to depict import and access dependencies between packages, classes, components, and other named elements within our system. It helps to simplify the complex class diagrams that can be used to group classes into packages. These groups help define the hierarchy and increase cohesion. In our project we have **model** package which consist of users and main entity, **repo** package which consist of DAO (direct access object), **controller** package which consist the interface controller and **service** package consist of service used to access database and we have **security** package which consist all the security configuration and authentication implements. Our project has followed the software engineering approach to have less coupling and high cohesion through packages.

8.4 Class Interfaces

Interface is a collection of abstract methods that is implemented by class. It may also contain constants, default methods, static methods, and nested types. A class describes the attributes and behaviors of an object. And an interface contains behaviors that a class implements. In Java keyword “implements” is used to represent interface used by class.

In our project, we have many interfaces used/implemented by class to get the service. For more detail, please check our project git:
https://github.com/shubhanginigon/SDQI2021_G1/tree/main/AQMS/src/main/java/com/sdqi2021/AQMS/service

9. Framework/Library

In our project, to simplify the software development process and to integrate existing distributed application we have used the following frameworks and libraries as follows:

1. **Spring Boot Java Framework:**

It is an open-source Java-based Framework that is usually used in creating microservices. This means it supports the deployment of services independently. It is a stand-alone spring application that makes it easy to develop spring applications with increased productivity and reduction in development time. It provides a flexible way to configure Java Beans, XML configurations, and Database Transactions. It includes an embedded servlet container, and everything is auto configured.

2. **Hibernate ORM**

Hibernate in Java Framework simplifies the interaction of java applications with the database. It is an open source, lightweight, ORM (object Relational Mapping) tool. It ORM allows us to develop object-oriented

persistent classes with all known functionality like polymorphism, inheritance, and association. It supports data creation, data manipulation and data access without the use of sql. Using hibernate ORM helps to enhance the performance, supports database independent queries and automatic table creation. In addition to this, it supports scalability, reliability, and extensibility.

3. **JDBC**

It is a Java API that stands for Java Database Connectivity that is used for database independent connectivity between the Java programming language and a wide range of databases. It basically connects to the database, executes queries and update statements to the database and retrieves the result received from the database.

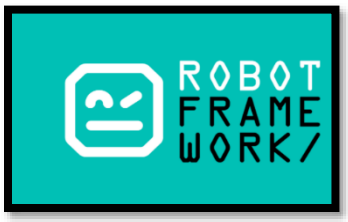
4. **Maven**

It is a powerful project management tool that works based on Project Object Model and is used in project build, dependency, and documentation from a central piece of information. It includes Build plugins, build profiles, POM files, dependencies, and repositories, that helps to easily build a project. Using maven, we can easily integrate our project with the source control system.

5. **Robot Framework**

It is a generic keyword-driven test automation framework for acceptance level testing and acceptance test-driven development (ATDD) that extends the system level testing capabilities with specifications and test cases associated with the actual app testing.

It is easy to install, supports keyword-driven, behavior-driven, and data-driven style of writing test cases and also supports external libraries like Selenium Library.



6. **Jmeter**

JMeter is an Open-Source testing software. It is a pure Java application for load and performance testing. It supports various categories of tests such as load testing, functional testing, performance testing,

regression testing, etc., and it requires JDK 5 or higher. It has a user-friendly UI and anyone without extensive knowledge of programming can also use it. It eases API testing, database testing and supports testing from browsers or native applications. Also, it provides integration with Jenkins and reporting.

7. **Spring Security**

Spring security when integrated with the spring framework gives a powerful and highly customizable authentication and access control framework that supports authentication and authorization in Java applications. It is easily configurable and extendable to meet the needs of a specific application. It can be used for any type of application from desktop/standalone to web applications and using it we can setup application security in a few hours.

8. **Thymeleaf Template Engine**

Thymeleaf is a modern server-side Java template engine for both web and standalone environments, which can process HTML, XML, text, JavaScript, or CSS files. Thymeleaf allows using templates as prototypes, meaning they can be viewed as static files. It provides an elegant and highly maintainable way of creating web codes while adding powerful features and retaining prototyping abilities.

10. The status of the implementation

As per the objectives of our project, we have completed, and details of work done are in our JIRA.

Link to below Jira software:

https://sqdi2021g1.atlassian.net/jira/software/c/projects/AQMS/boards/4/roadmap

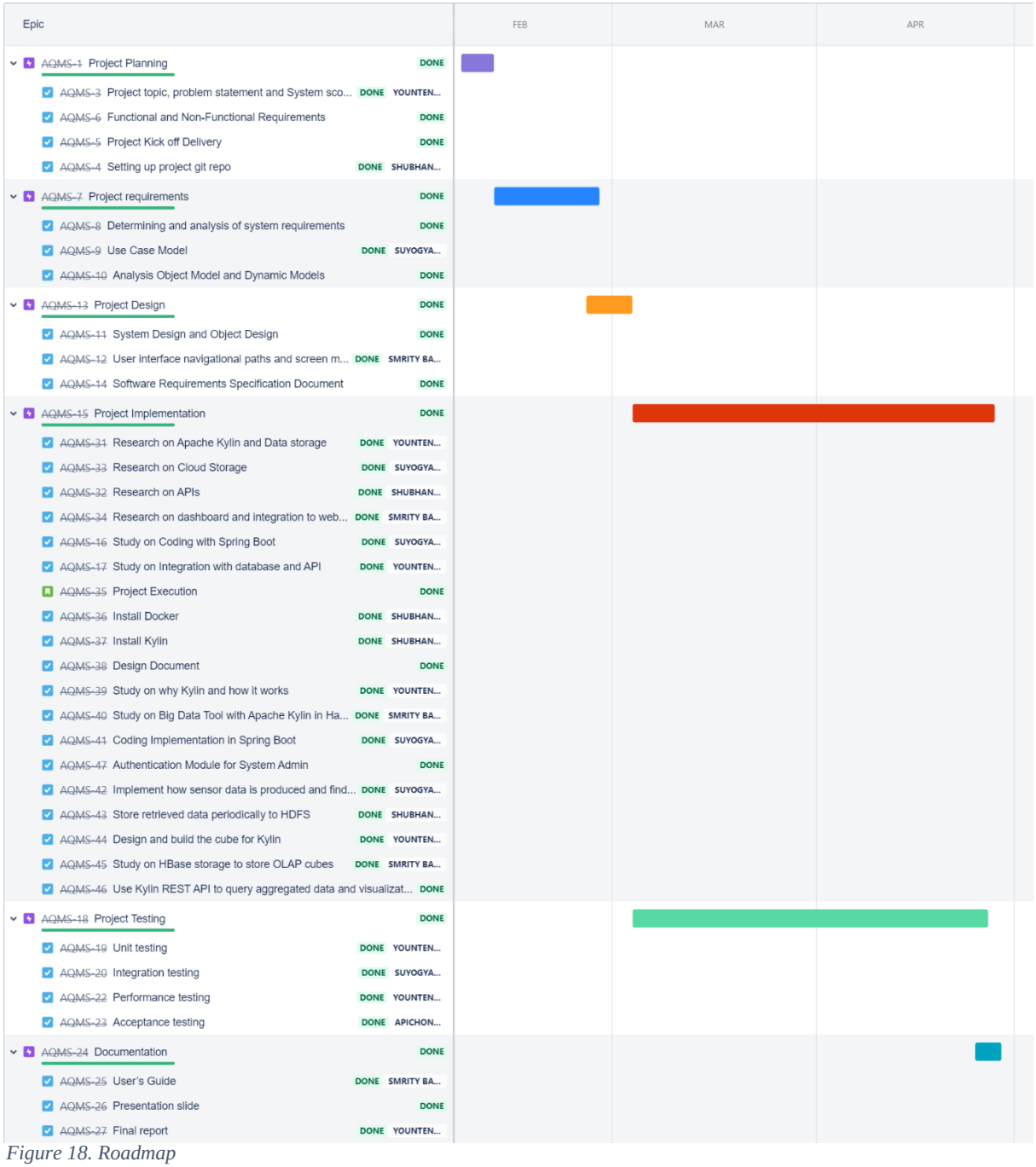


Figure 18. Roadmap

For more details on implementation follow our link:

https://github.com/shubhanginigon/SDQI2021_G1

11. User Acceptance Test Report - Testing

The application testing applied to our system (AQMS) for concurrent programs which is main challenges that will be addressed in handling multiple concurrent transactions and real-time information updates. 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. This part of document will present a systematic view, which is a systemic procedures and experiments in testing that will be applied to our application.

The objective of the 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.

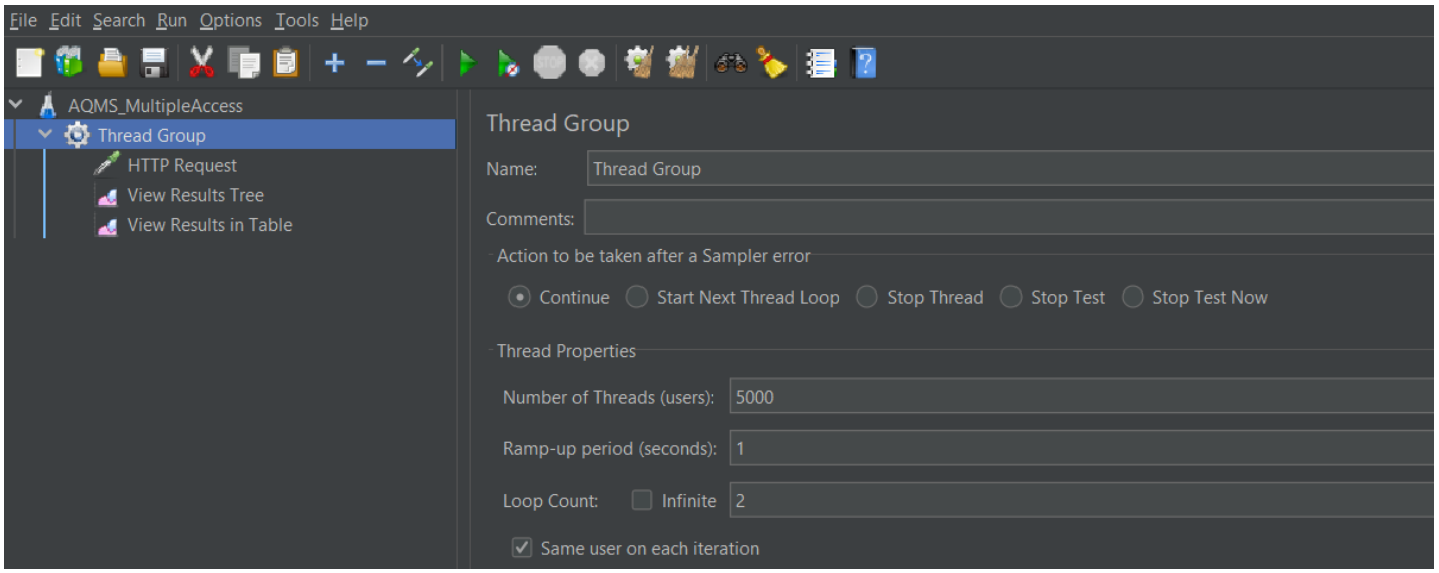
For this testing, we are concern on putting our application into testing, so to prove that the desired qualities are satisfied.

11.1 Test using JMeter

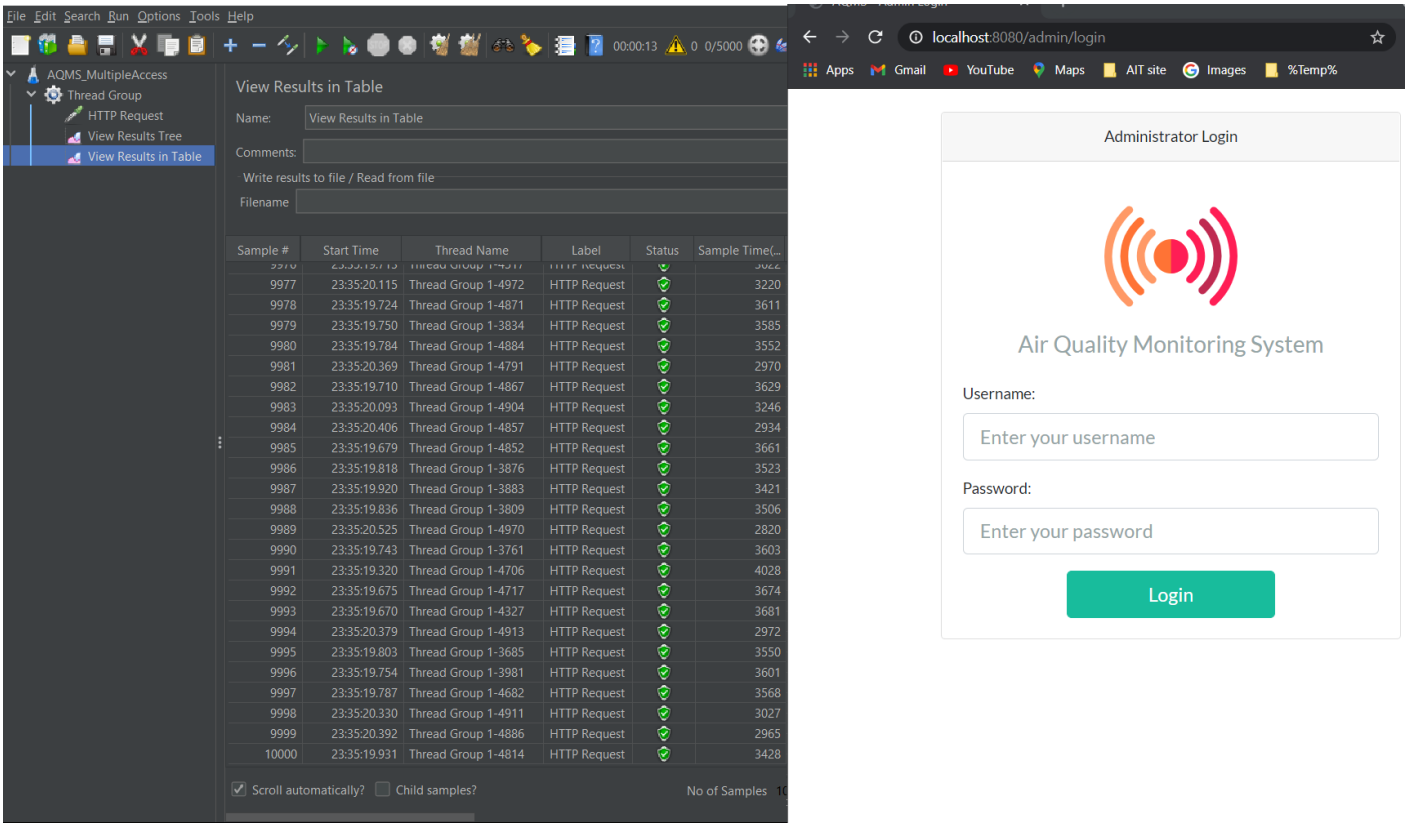
JMeter for the purpose of load testing is considered as the one of the best tools. Apache JMeter is a Java open-source GUI-based software that is used as a performance testing tool for analyzing and measuring the performance of our application.

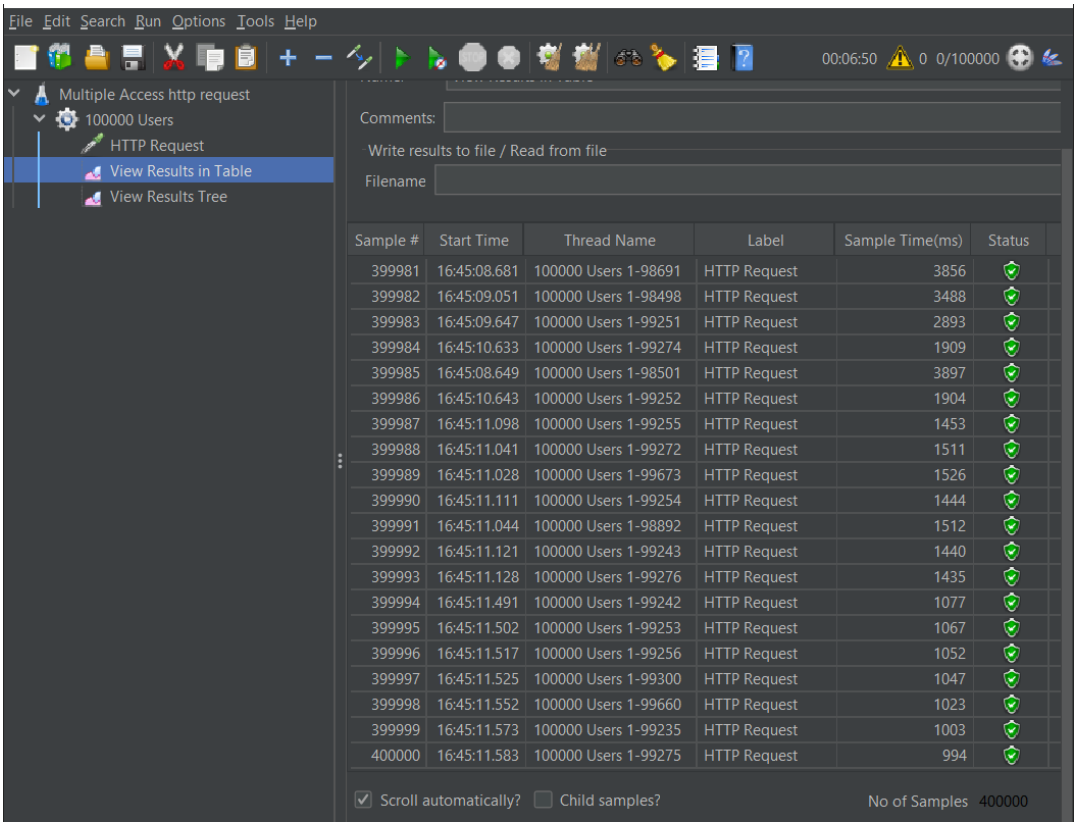
11.1.1 Test multiple access

We configure 5000 threads in 2 loops, making it 10000 threads or users access the localhost.



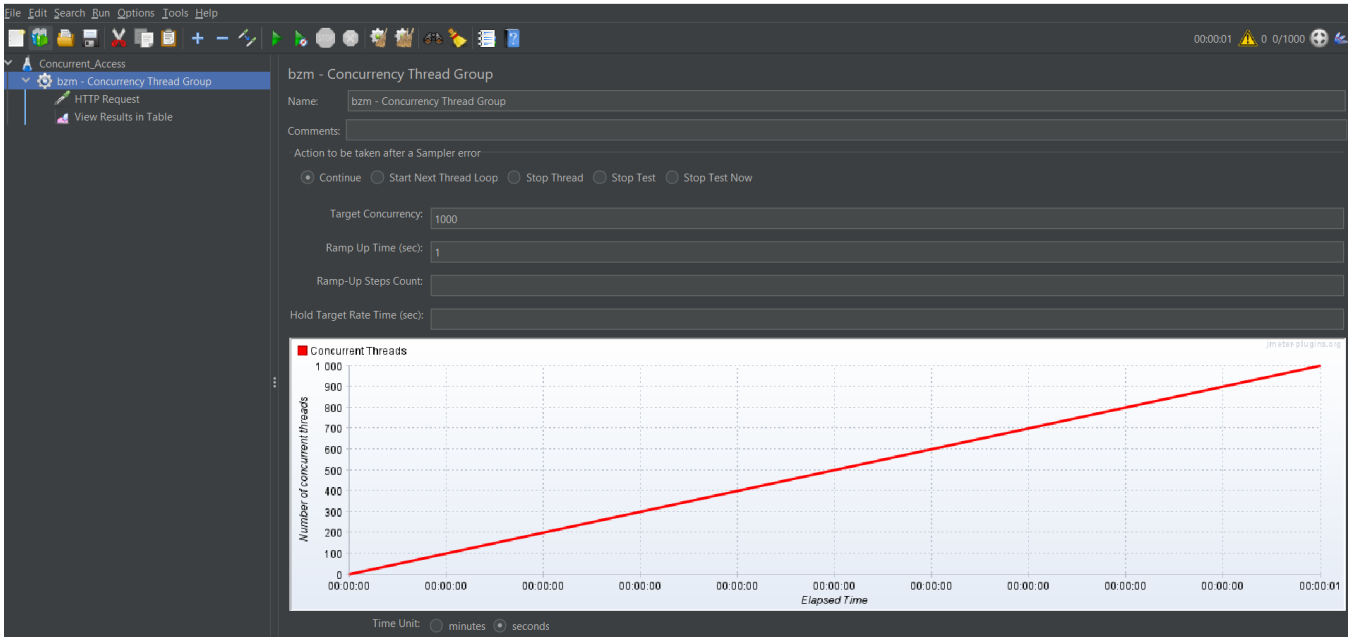
Our application was able to handle the multiple access over 10000 and we tested till 400000.



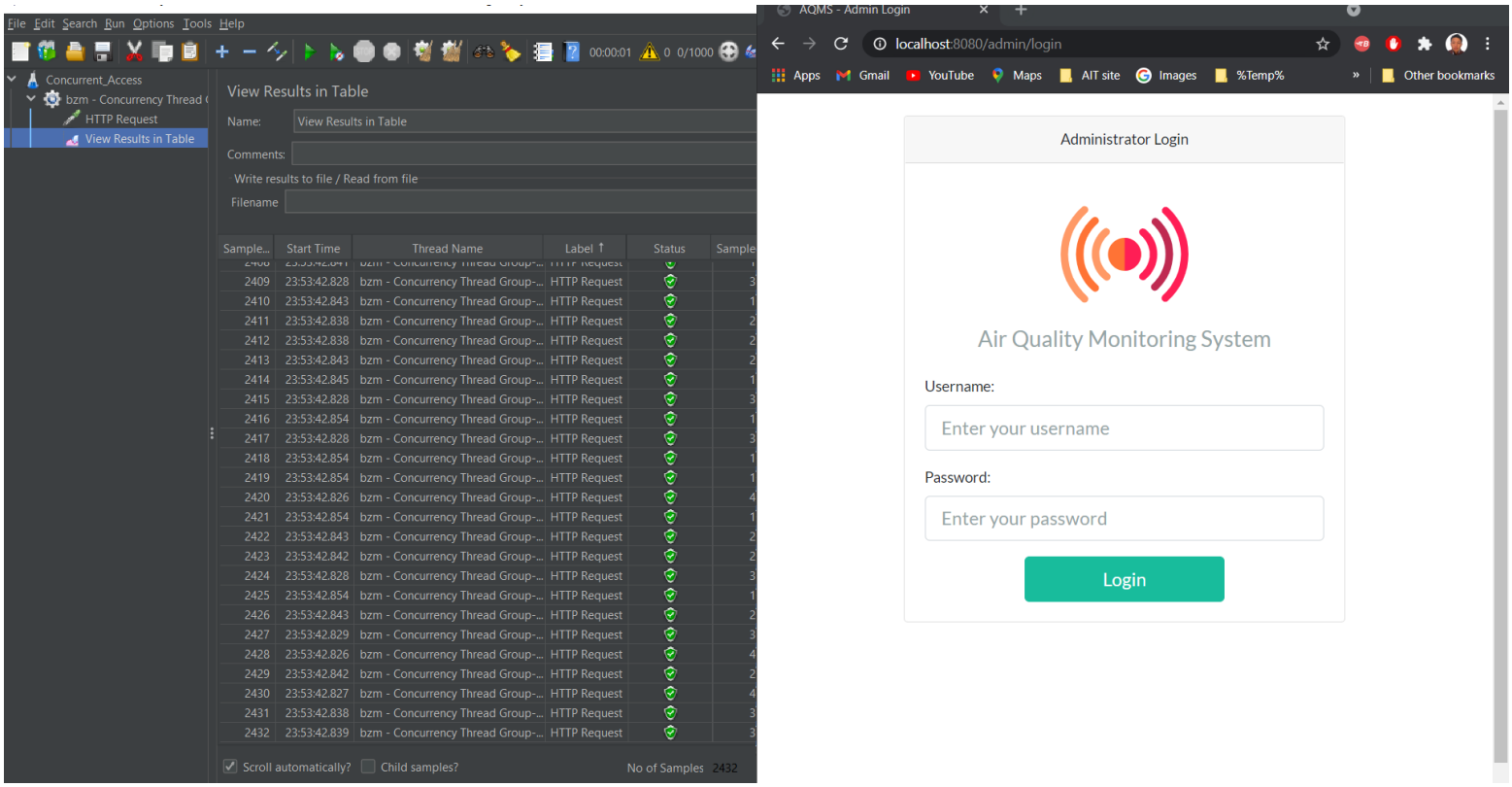


11.1.2 Test Concurrent access

We configure 1000 group threads in 1 second, making it concurrent threads or users access the localhost.



Our application was able to handle the more than 1000 concurrent access.



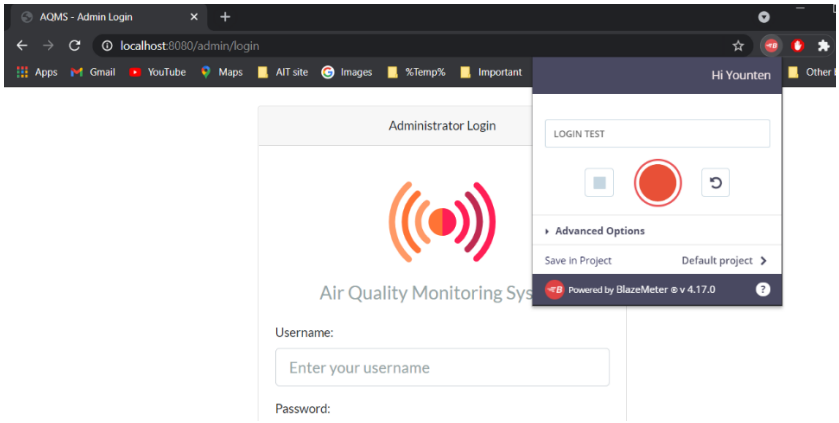
11.1.3 Testing login and other features

Record login test in JMeter by following way:

Step 1: add blaze meter plugin to chrome browser.

Step 2: start blaze meter plugin and login to blazemeter

Step 3: Record our scenario – Stop Recording – Export .jmx

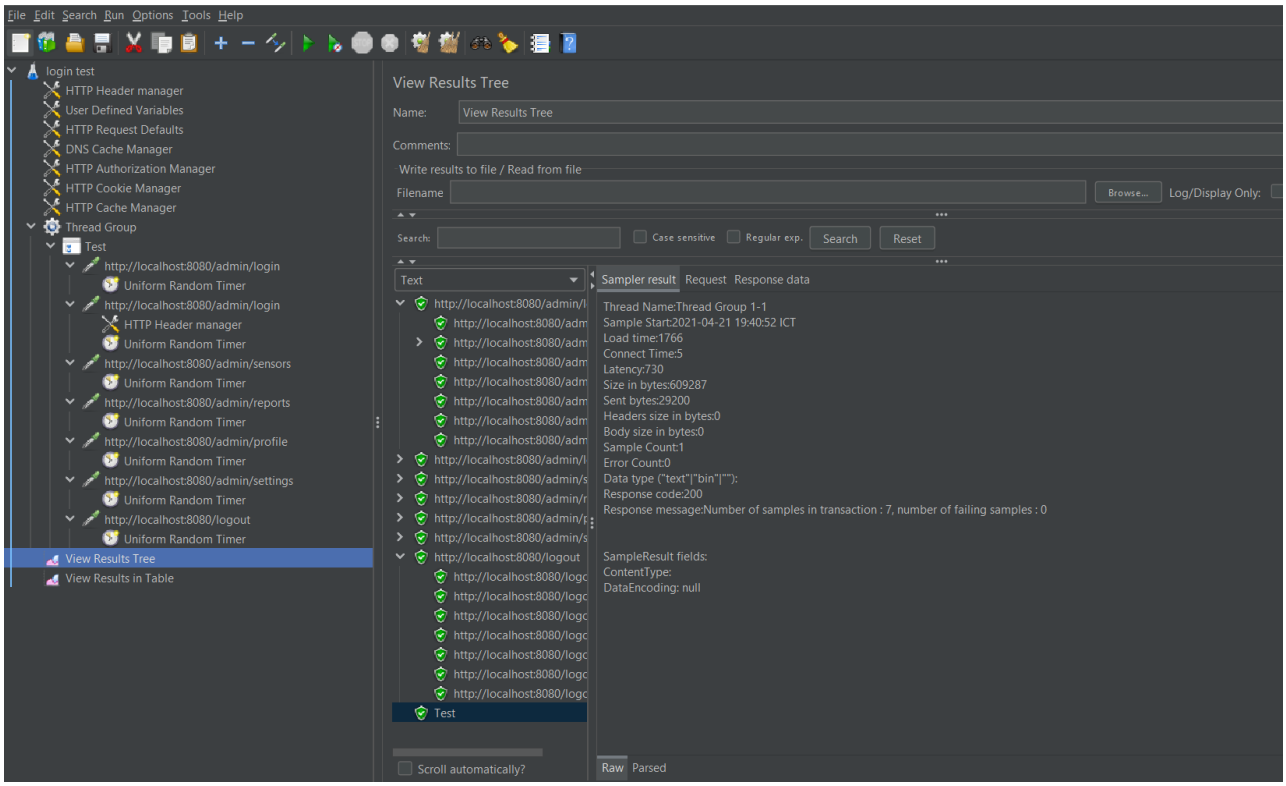


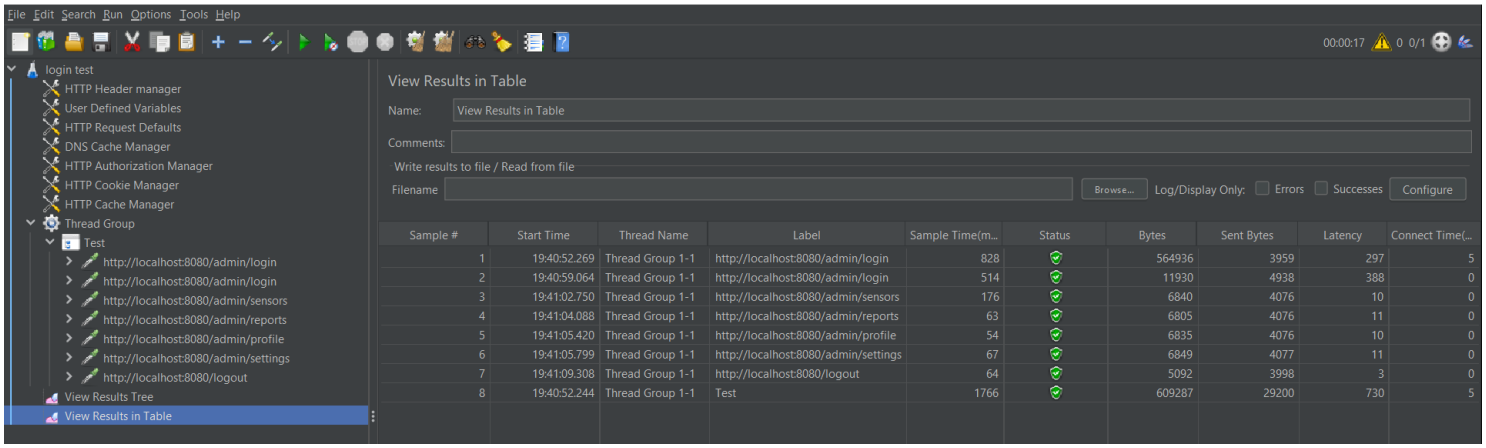
Step 4: Import jmx file in Jmeter

Step 5 : Add Listeners

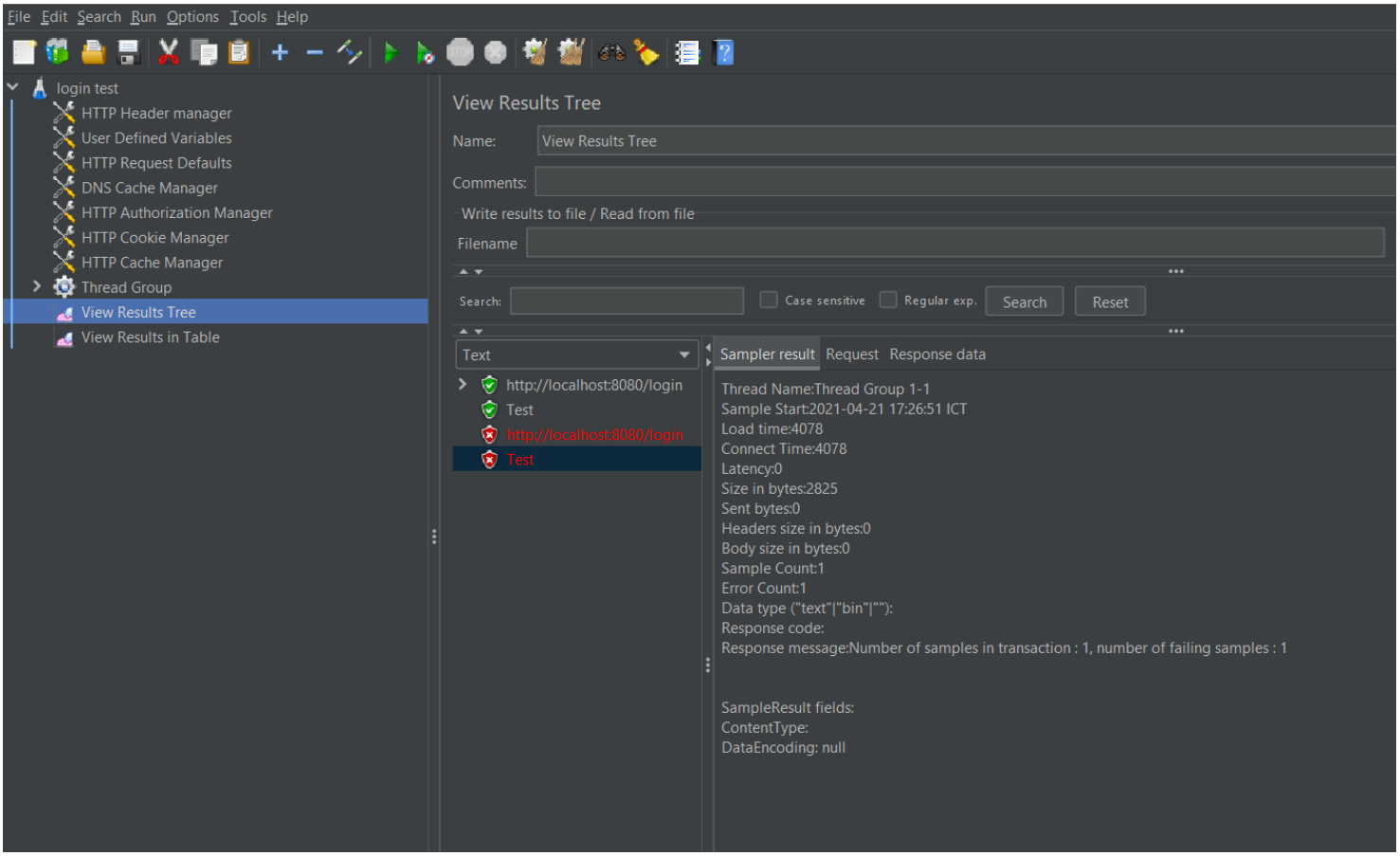
Step 6 : Run and Validate

Result of running the record and displayed in table and tree format.





Checking the failure scenario when the server is down.



11.2 Test using Robot Framework.

For our project, we have used two ways to visualize our data, one way was Tableau and Kylin connecting with JDBC driver which had some compatibility issues at the end and for ODBC we had to contact vendor. Since we knew the cube structure and table format, we made the same format in MySQL database so that

we can test the visualization and show the Tableau connection. The data in database was also fetch using the python script on real time bases (hourly from API). For testing purpose, we have used JMeter and Robot framework to show high performance and usability.

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 own test-cases without programming knowledge. Test Robot with simple webapp access by writing the script in VS Code with robot file extension and running will show our test case pass or fail. In addition to this, it has very descriptive logs including complete debugging capabilities through readable logs.

Following figure shows the test case implemented in robot framework where the test data is in simple and easy-to-edit format. When Robot Framework is started, it processes the test data, executes test cases and generates logs and reports.

11.2.1 Test case for login and start/stop service.

Test case for checking automation is user will open website, then login into system as administrator, then 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.

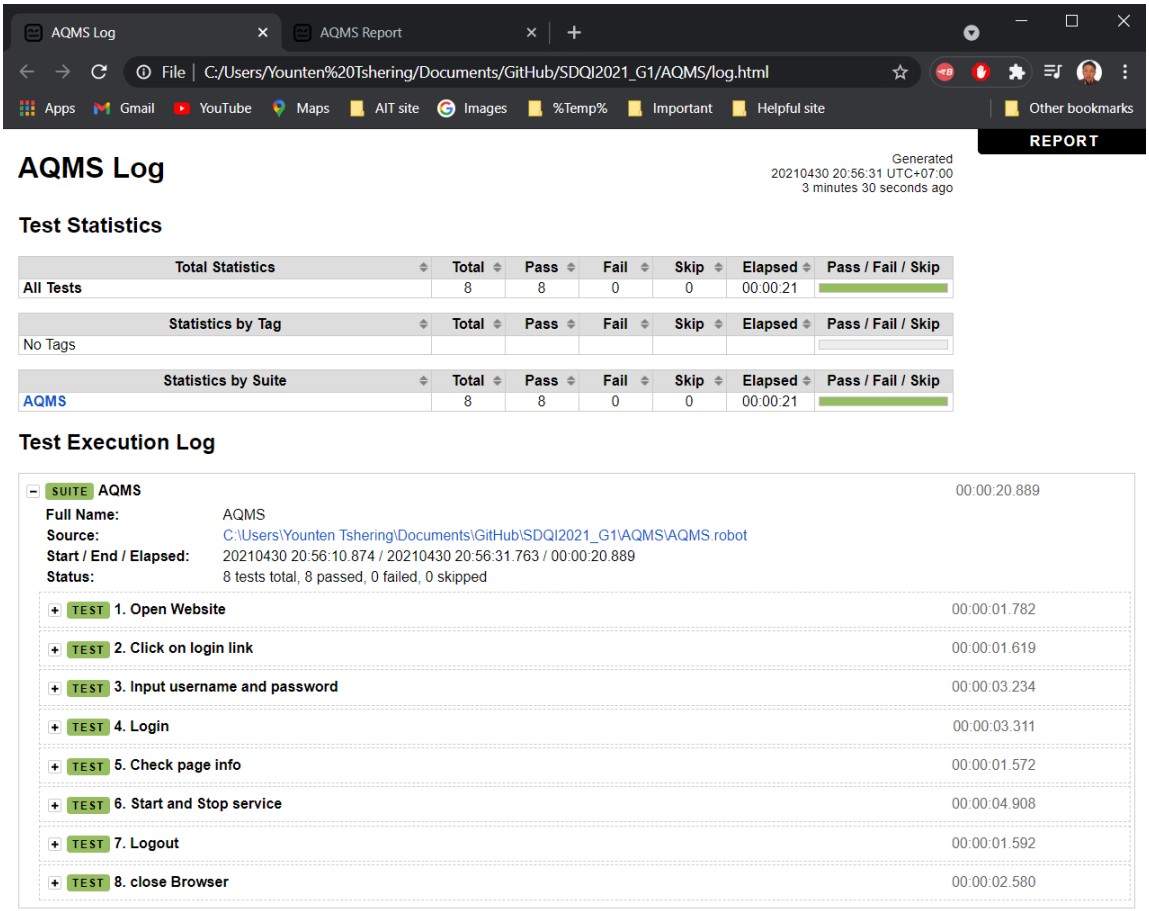
AQMS.robot M

AQMS > AQMS.robot > ...

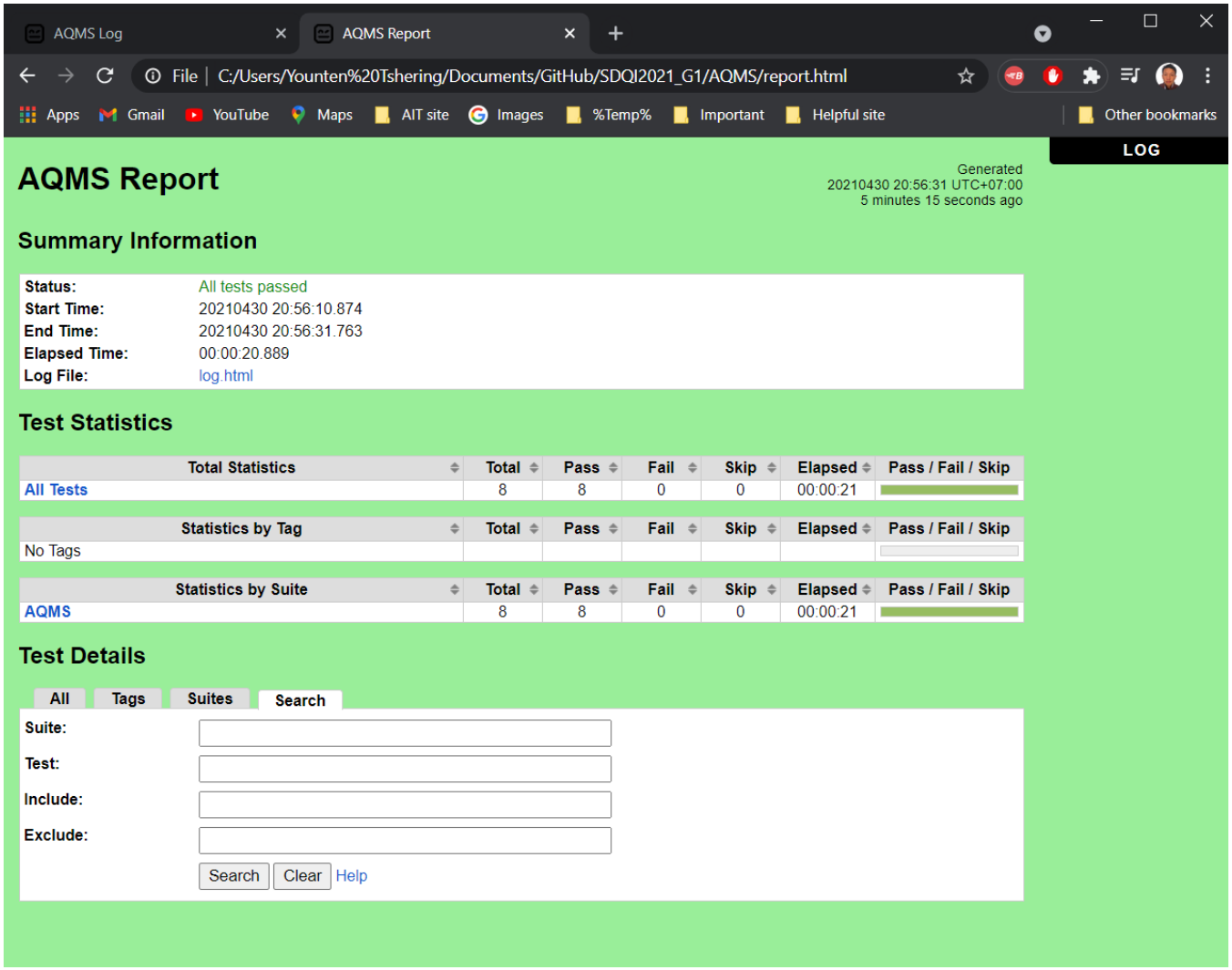
```
*** Settings ***
Library  Selenium2Library

*** 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
```

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.

```
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
```



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



11.2.2 Test case for visualization content for user.

Test case for checking automation is user will open website, then view the content of dashboard with different placeholders and at the end close the browser after checking or monitoring the air quality.

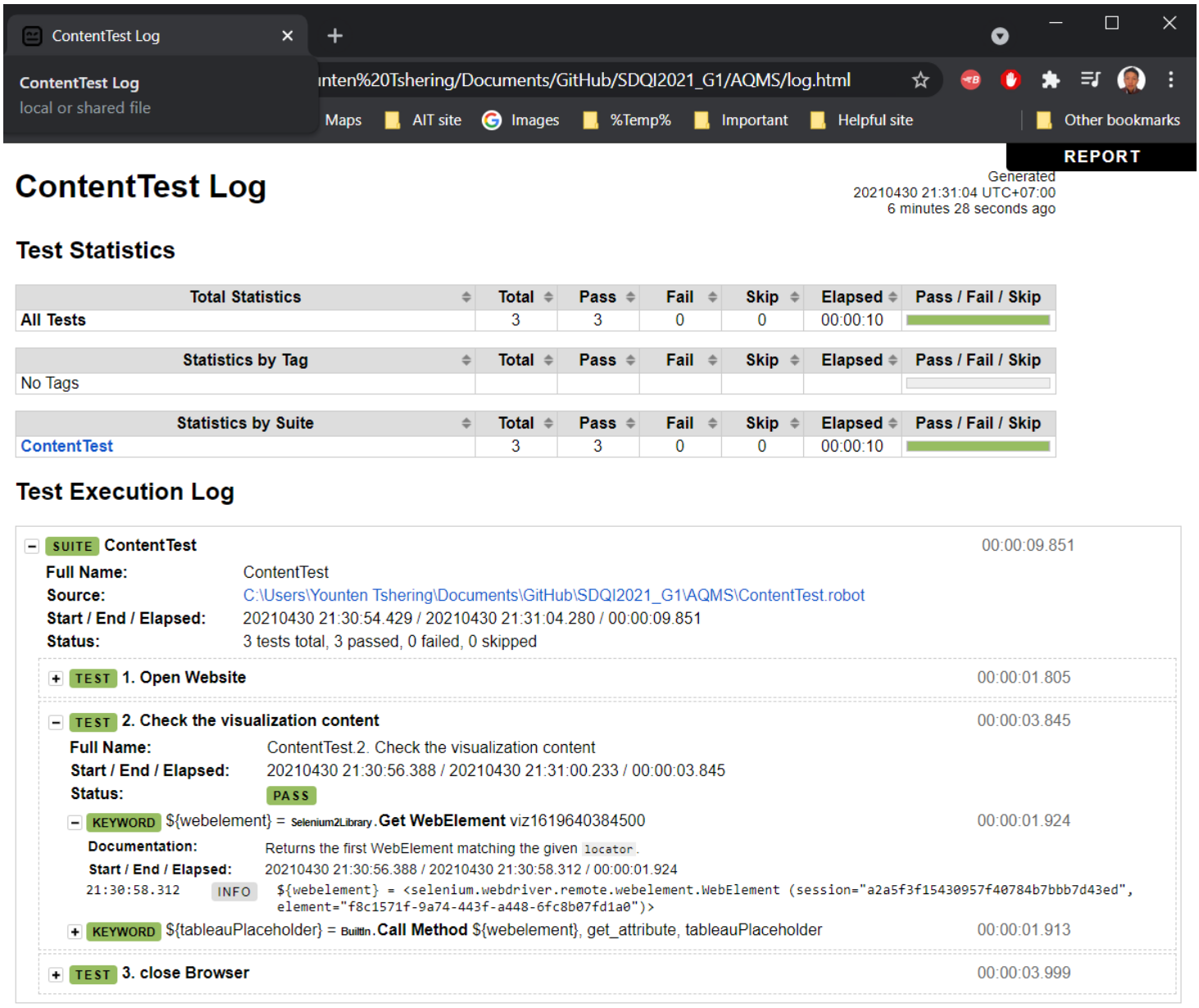
Once we have the test case, then we can run the test and check the test pass or fail. The figure given below shows that all the test cases created were passed and the automation worked perfectly for that scenario.

ContentTest.robot U

AQMS > ContentTest.robot > ${delay}

```
*** 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   1.9

2. Check the visualization content
   ${webelement}=  Get WebElement  viz1619640384500
   ${tableauPlaceholder}=  Call Method  ${webelement}  get_attribute  tableauPlaceholder

3. close Browser
   Close Browser
```

```
PS C:\Users\Younten Tshering\Documents\GitHub\SDQI2021_G1\AQMS> robot .\ContentTest.robot
==============================================================================
ContentTest
==============================================================================
1. Open Website                                                       | PASS |
------------------------------------------------------------------------------
2. Check the visualization content                                    | PASS |
------------------------------------------------------------------------------
3. close Browser                                                      | PASS |
------------------------------------------------------------------------------
ContentTest                                                           | PASS |
3 tests, 3 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
```



11.2.3 Test case for visualization content for user.

Test case for checking automation is user will open website, then login into system as administrator, then view the content of dashboard with different placeholders and at the end close the browser after checking or monitoring the air quality.

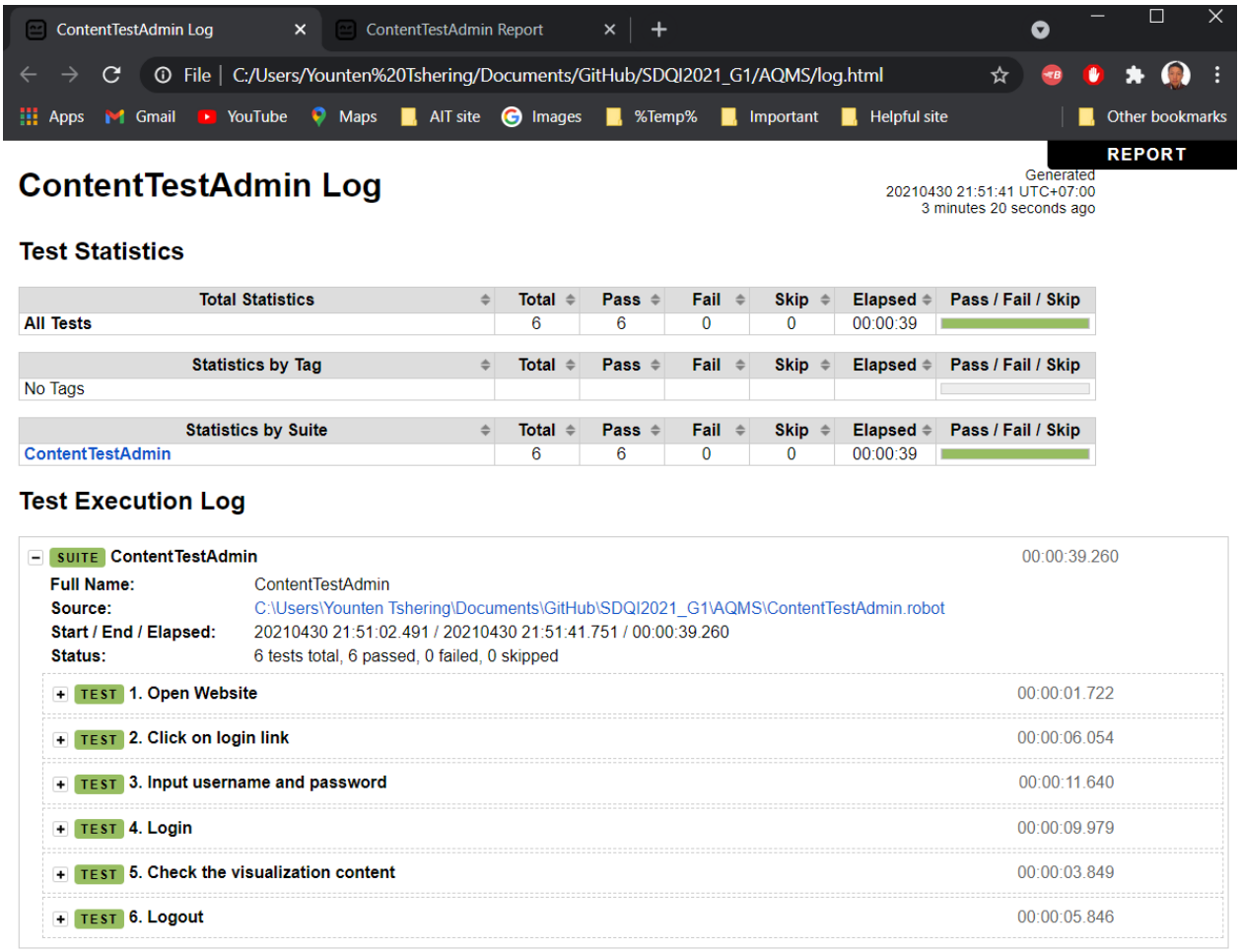
Once we have the test case, then we can run the test and check the test pass or fail. The figure given below shows that all the test cases created were passed and the automation worked perfectly for that scenario.

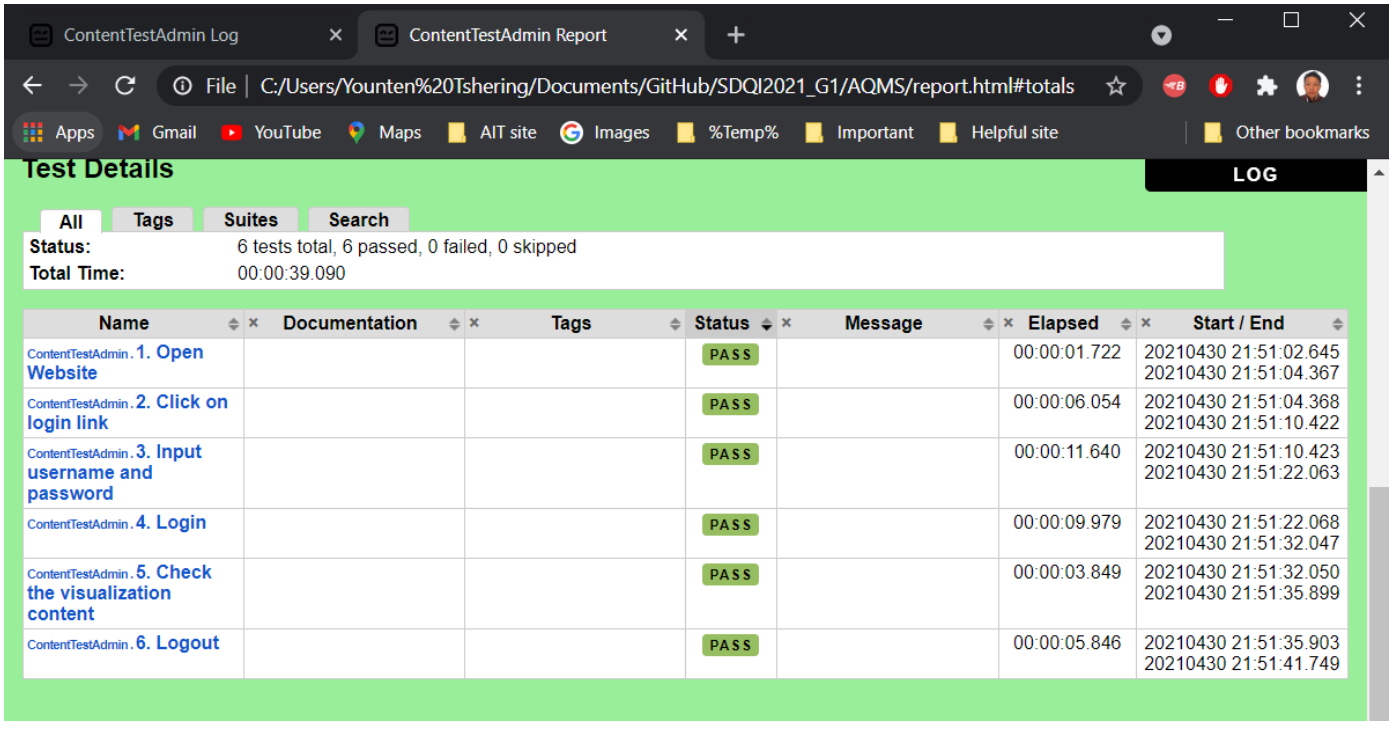
ContentTestAdmin.robot U

AQMS > ContentTestAdmin.robot > 6. Logout

```
*** 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   1.9

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 the visualization content
   ${webelement}=  Get WebElement  viz1619640384500
   ${tableauPlaceholder}=  Call Method  ${webelement}  get_attribute  tableauPlaceholder

6. Logout
   Click Link  //a[contains(text(),'Logout')]

```

```
PS C:\Users\Younten Tshering\Documents\GitHub\SDQI2021_G1\AQMS> robot .\ContentTestAdmin.robot
==============================================================================
ContentTestAdmin
==============================================================================
1. Open Website                                                       | PASS |
------------------------------------------------------------------------------
2. Click on login link                                                | PASS |
------------------------------------------------------------------------------
3. Input username and password                                        | PASS |
------------------------------------------------------------------------------
4. Login                                                              | PASS |
------------------------------------------------------------------------------
5. Check the visualization content                                    | PASS |
------------------------------------------------------------------------------
6. Logout                                                             | PASS |
------------------------------------------------------------------------------
ContentTestAdmin                                                      | PASS |
6 tests, 6 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
PS C:\Users\Younten Tshering\Documents\GitHub\SDQI2021_G1\AQMS>
```


ContentTestAdmin Log
ContentTestAdmin Report
File | C:/Users/Younten%20Tshering/Documents/GitHub/SDQI2021_G1/AQMS/log.html
Apps
Gmail
YouTube
Maps
AIT site
Images
%Temp%
Important
Helpful site
Other bookmarks
REPORT
ContentTestAdmin Log
Generated
20210430 21:51:41 UTC+07:00
3 minutes 20 seconds ago
Test Statistics
Total Statistics | Total | Pass | Fail | Skip | Elapsed | Pass / Fail / Skip
All Tests | 6 | 6 | 0 | 0 | 00:00:39
Statistics by Tag | Total | Pass | Fail | Skip | Elapsed | Pass / Fail / Skip
No Tags
Statistics by Suite | Total | Pass | Fail | Skip | Elapsed | Pass / Fail / Skip
ContentTestAdmin | 6 | 6 | 0 | 0 | 00:00:39
Test Execution Log
SUITE ContentTestAdmin 00:00:39.260
Full Name: ContentTestAdmin
Source: C:\Users\Younten Tshering\Documents\GitHub\SDQI2021_G1\AQMS\ContentTestAdmin.robot
Start / End / Elapsed: 20210430 21:51:02.491 / 20210430 21:51:41.751 / 00:00:39.260
Status: 6 tests total, 6 passed, 0 failed, 0 skipped
TEST 1. Open Website 00:00:01.722
TEST 2. Click on login link 00:00:06.054
TEST 3. Input username and password 00:00:11.640
TEST 4. Login 00:00:09.979
TEST 5. Check the visualization content 00:00:03.849
TEST 6. Logout 00:00:05.846


ContentTestAdmin Log
ContentTestAdmin Report
File | C:/Users/Younten%20Tshering/Documents/GitHub/SDQI2021_G1/AQMS/report.html#totals
Apps
Gmail
YouTube
Maps
AIT site
Images
%Temp%
Important
Helpful site
Other bookmarks
Test Details
LOG
All
Tags
Suites
Search
Status: 6 tests total, 6 passed, 0 failed, 0 skipped
Total Time: 00:00:39.090
Name | Documentation | Tags | Status | Message | Elapsed | Start / End
ContentTestAdmin.1. Open Website | PASS | 00:00:01.722 | 20210430 21:51:02.645 20210430 21:51:04.367
ContentTestAdmin.2. Click on login link | PASS | 00:00:06.054 | 20210430 21:51:04.368 20210430 21:51:10.422
ContentTestAdmin.3. Input username and password | PASS | 00:00:11.640 | 20210430 21:51:10.423 20210430 21:51:22.063
ContentTestAdmin.4. Login | PASS | 00:00:09.979 | 20210430 21:51:22.068 20210430 21:51:32.047
ContentTestAdmin.5. Check the visualization content | PASS | 00:00:03.849 | 20210430 21:51:32.050 20210430 21:51:35.899
ContentTestAdmin.6. Logout | PASS | 00:00:05.846 | 20210430 21:51:35.903 20210430 21:51:41.749

12. Conclusion

Poor Air quality has become a serious hazard to human health these days. People must be made aware about this condition via any channel. So, we have come up with the web application for addressing the air quality. The motive of this application is to present the recent scenario of the air quality based on PM2.5 of various locations of Thailand. This system uses the actual data from various sensors located at different areas and based on the data obtained, it visualizes the condition of air quality level as 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 savvy people, without the need of any registration allowing all concerned people to take precaution for their health. We expect this system to be user friendly and useful to all the concerned users.

The Air Quality Problem is diverse. There are many factors which 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 given location. This has made the scope of application smaller. There are some gaps that can be fulfilled by extending this application in 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.

13. Glossary & references

13.1 Glossary

Here are some domain terms we are using in document:

- **Framework** serves as a foundation for programming that acts as a platform for developing softwares.
- **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 is 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**: OLAP is an acronym for online analytical processing
- 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.

13.2 References

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

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

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

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

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

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

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

14. Appendix – Hadoop and Apache Kylin

Apache Kylin is an open-source distributed analytics engine designed to provide a SQL interface and multi-dimensional analysis on Hadoop supporting extremely large datasets. It was originally developed by eBay and is now a project of the Apache.

In addition, it easily integrates with BI tools via ODBC driver, JDBC driver, and REST API. It was created by eBay in 2014, graduated 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.

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 the milliseconds. 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 you query billions of rows at sub-second latency in 3 steps.

1. Identify a Star/Snowflake Schema on Hadoop.
2. Build Cube from the identified tables.
3. Query using ANSI-SQL and get results in sub-second, via ODBC, JDBC or RESTful API.

14.1 Big Data Tool with Apache Kylin

Now, before moving on to Apache Kylin, let us start the discussion with Big Data, that led to the development of Hadoop and then to Apache Kylin.

IoT connects our physical device to the internet and makes it smarter. Nowadays, we have smart air conditioners, televisions etc. Our smart air conditioner constantly monitors our room temperature along with the outside temperature and accordingly decides what should be the temperature of the room. Now imagine how much data would be generated in a year by smart air conditioner installed in tens & thousands of houses. By this we can understand how IoT is contributing a major share to Big Data.

- **Why Big Data is a problem statement and how Hadoop solves it.**

There were three major challenges with Big Data:

1. **The first problem is storing the huge amount of data**. Storing huge 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 a tremendous rate.

2. **The second problem is storing heterogeneous data**. The data is not only huge, but it is also present in various formats i.e. unstructured, semi-structured and structured. So, we need to make sure that we have a system to store different types of data that is generated from various sources.
3. **Finally, the third problem, which is the processing speed**. Now the time taken to process this huge amount of data is quite high as the data to be processed is too large.

To solve the storage issue and processing issue:

- Two core components were created in Hadoop — **HDFS (**Hadoop Distributed File System**) and YARN(**Yet Another Resource Negotiator**)**. HDFS solves the storage issue as it stores the data in a distributed fashion and is easily scalable. And YARN solves the processing issue by reducing the processing time drastically.

While setting up a Hadoop cluster, we have an option of choosing a lot of services as part of your Hadoop platform, but there are two services which are always mandatory for setting up Hadoop. One is HDFS (storage) and the other is YARN (processing). HDFS stands for Hadoop Distributed File System, which is a scalable storage unit of Hadoop whereas YARN is used to process the data i.e. stored in the HDFS in **a distributed and parallel fashion.**

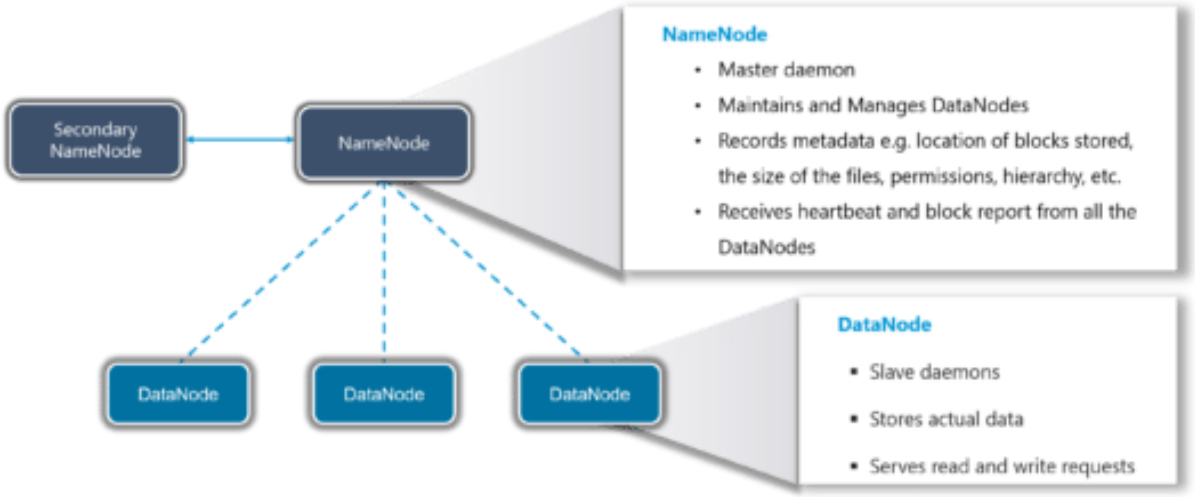


Figure 19. HDFS

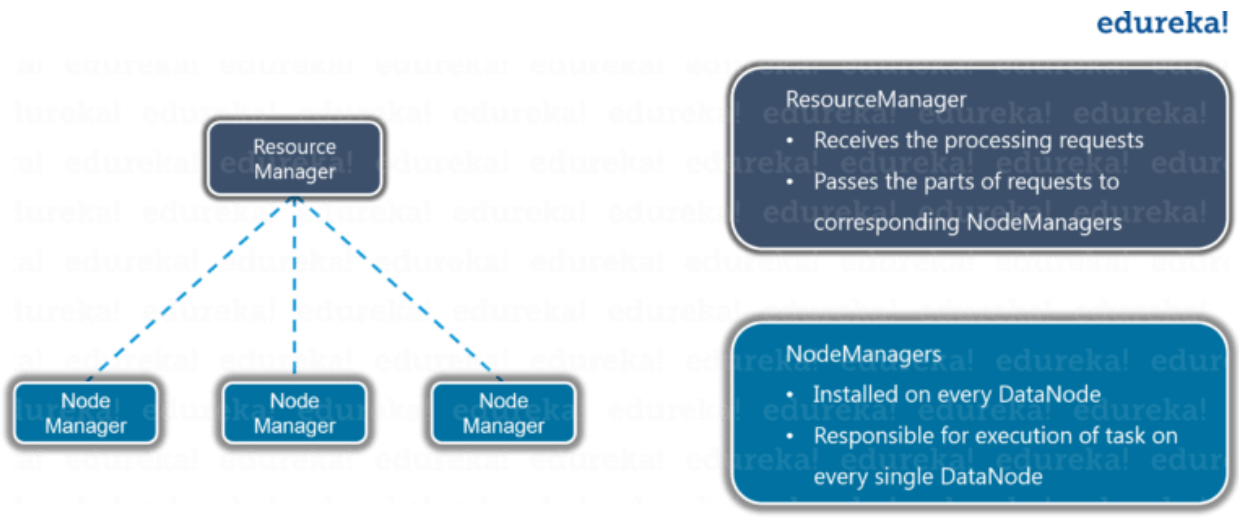


Figure 20. YARN

Data Warehouses have served very good purposes of Data Storage and Data Analytics. But with the exponential growth of data and with the potent intelligence lying in them, enterprises are pushing toward adopting Big Data Technologies.

While Hadoop provides a reliable, scalable, and distributed computing power, its initial design was towards enabling batch processing of large data. But if you need to move our analysts to use Hadoop, to unleash the power of insights into large data, we need to enable:

1. Interactive querying capability
2. Reporting and dashboard development through BI Tools

14.2 Hadoop Eco-system:

The ecosystem around Hadoop is rapidly developing and adding on tools for various data needs. No single tool can provide both and hence we would have to go with a combination of tools.

The solution could be a combination of one Interactive Querying tool and one OLAP (online Analytical Processing) tool on Hadoop. **OLAP on Hadoop – Apache Kylin**

- **What is Kylin?**

Kylin is an open source Distributed Analytical Engine that provides SQL interface and multidimensional analysis (OLAP) on Hadoop supporting extremely large datasets. Apache kylin pre-calculates OLAP cubes and store the cubes into a reliable and scalable datastore (HBase).

❖ **Why Kylin?**

In most of the use cases in bigdata, we see the challenge is to get the result of the query within a sec. It takes lot of time to scan the database and to return the results. This is where the concept of OLAP in Hadoop emerged to combine the strength of OLAP and Hadoop and hence gives a significant improvement in query latency.

❖ **How it works?**

Below are the steps on how Kylin fetches the data and saves the results.

- First, sync the input source table. In most of the cases, it reads data from Hive
- Next it runs map reduce / spark jobs (based on the engine you select) to pre-calculate and generate each level of cuboids with all possible combinations of dimensions and calculate all the metrics at different levels.
- Finally, it stores cube data(aggregated result) in HBase where the dimensions are rowkeys and measures are column family.

After aggregated data is ready, we can integrate with popular BI tools, such Tableau or Superset or connect with JDBC from our code. One more option is Kylin also provides the simple UI for quickly visit data.

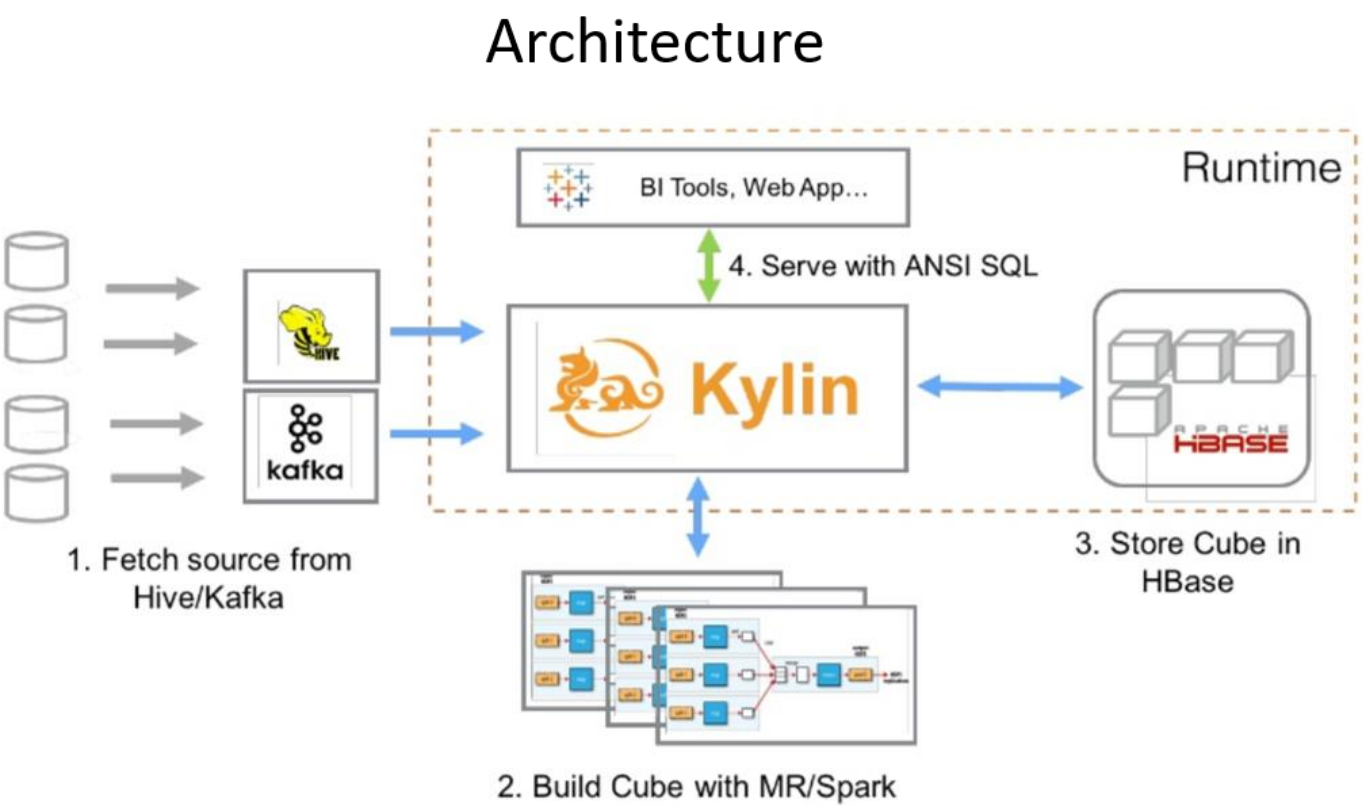


Figure 21. Apache Kylin – Architecture

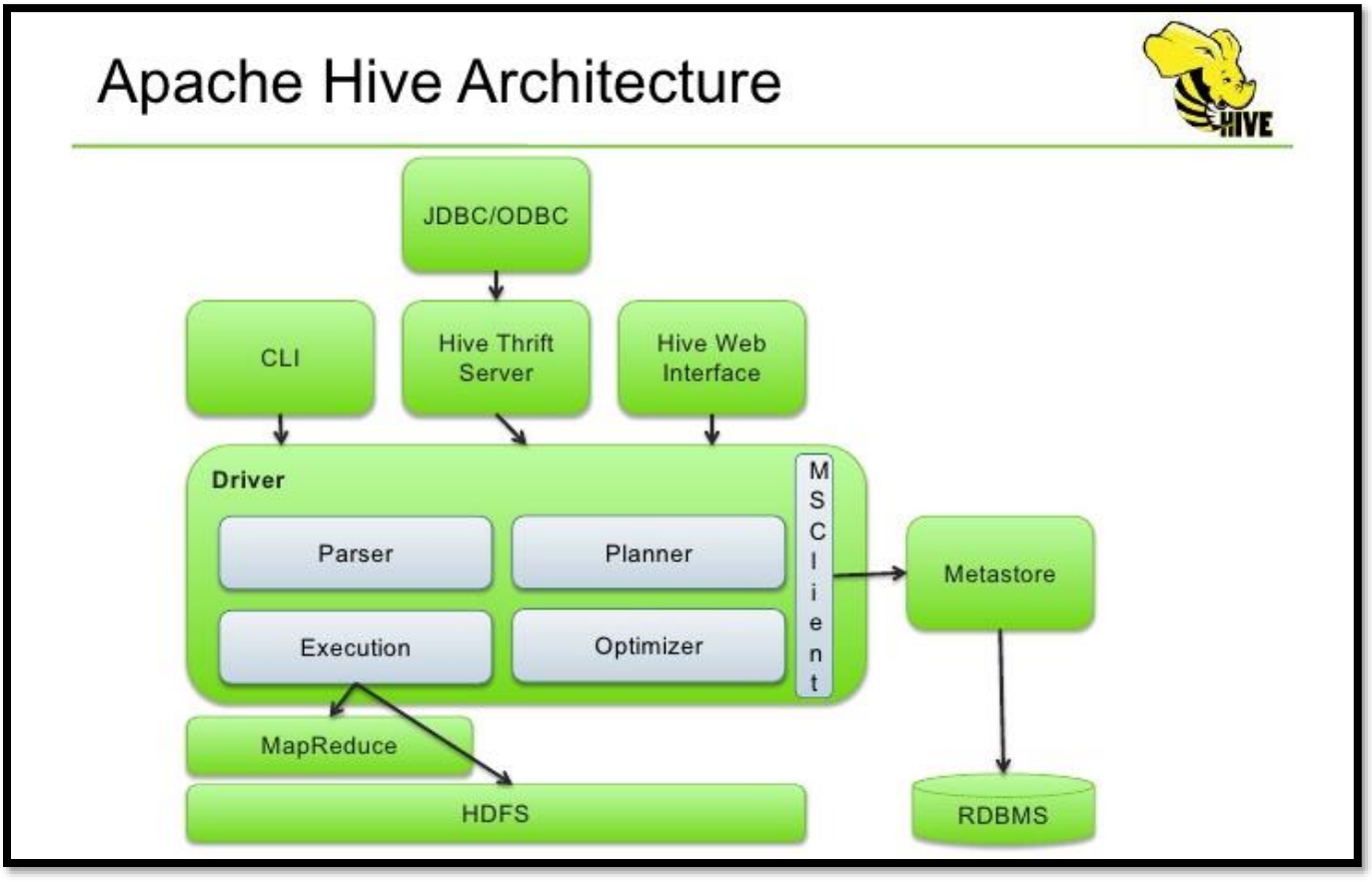
14.2.1 Hive

Apache Hive is a data warehouse system built on top of Hadoop or in Hadoop Ecosystem and is used for analyzing structured and semi-structured data. Basically, it provides a mechanism to project structure onto the data and perform queries written in HQL (Hive Query Language) that are similar to SQL statements. Internally, these queries or HQL gets converted to map reduce jobs by the Hive compiler.

Hive is not only a savior for people from a non-programming background, but it also reduces the work of programmers who spend long hours writing MapReduce programs.

Apache Hive supports Data Definition Language (DDL) and Data Manipulation Language (DML)

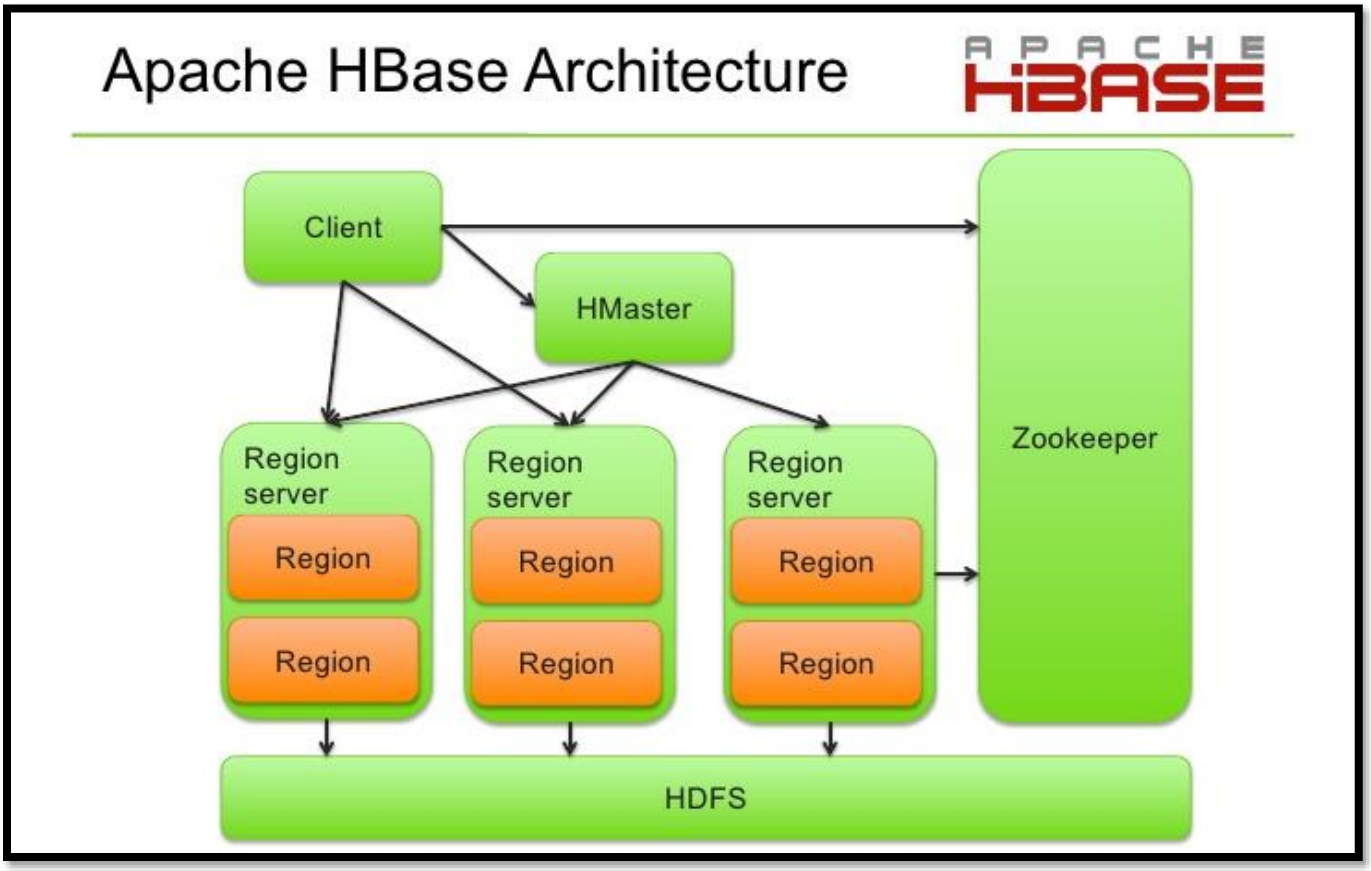
SQL + Hadoop MapReduce = HiveQL



14.2.2 HBase

HBase is an open-source, multidimensional, distributed, scalable, and a NoSQL database written in Java. HBase runs on top of HDFS (Hadoop Distributed File System) and provides Bigtable like capabilities to Hadoop.

HBase supports random read and writes while HDFS supports WORM (Write once Read Many or Multiple times).



Where we can use HBase?

We should use HBase where we have large data sets (millions or billions of rows and columns) and we require fast, random, and real-time, read, and write access over the data.

OLAP cube definition

First the user must identify the tables and the relationship between them over the data model stored in the source system chosen. Secondly, to reduce the effects of the dimensionality curse, it is very important to determine the optimal type for each dimension column. Finally, for those dimensions defined as “Normal”, we must create one or more aggregation groups and apply to them the possible optimizations allowed.

15. Appendix - User interface (navigational paths and screen mock-ups)

Our system is a web-based application system that visualizes the important details about the PM 2.5 to give clear vision on air quality of the selected location.

Air quality monitoring web-app is expected to be user-friendly with easy access of the features, as we will be using a navigation containing each component allowing the user to interact with the application in a natural and intuitive way.

For our system there are two users: Admin who will have privilege to manage the website, control sensors, make changes to the website as well as generate reports and next is Visitor who will have access to dashboard only.

15.1 Interface for Visitors

For guest users, once the URL is entered, they will be able to see the air quality monitoring system webpage. This webpage will include a dashboard that contains descriptive analysis of the air quality of the place based on PM2.5. This will give a quick view of the detail of the webpage. The option to choose location will be provided. Visualization can be about the current situation of air quality or for the given time, within the selected location such as pollution level, level of PM2.5, level of quality of air, etc. The mockup screen for visitor is given below:

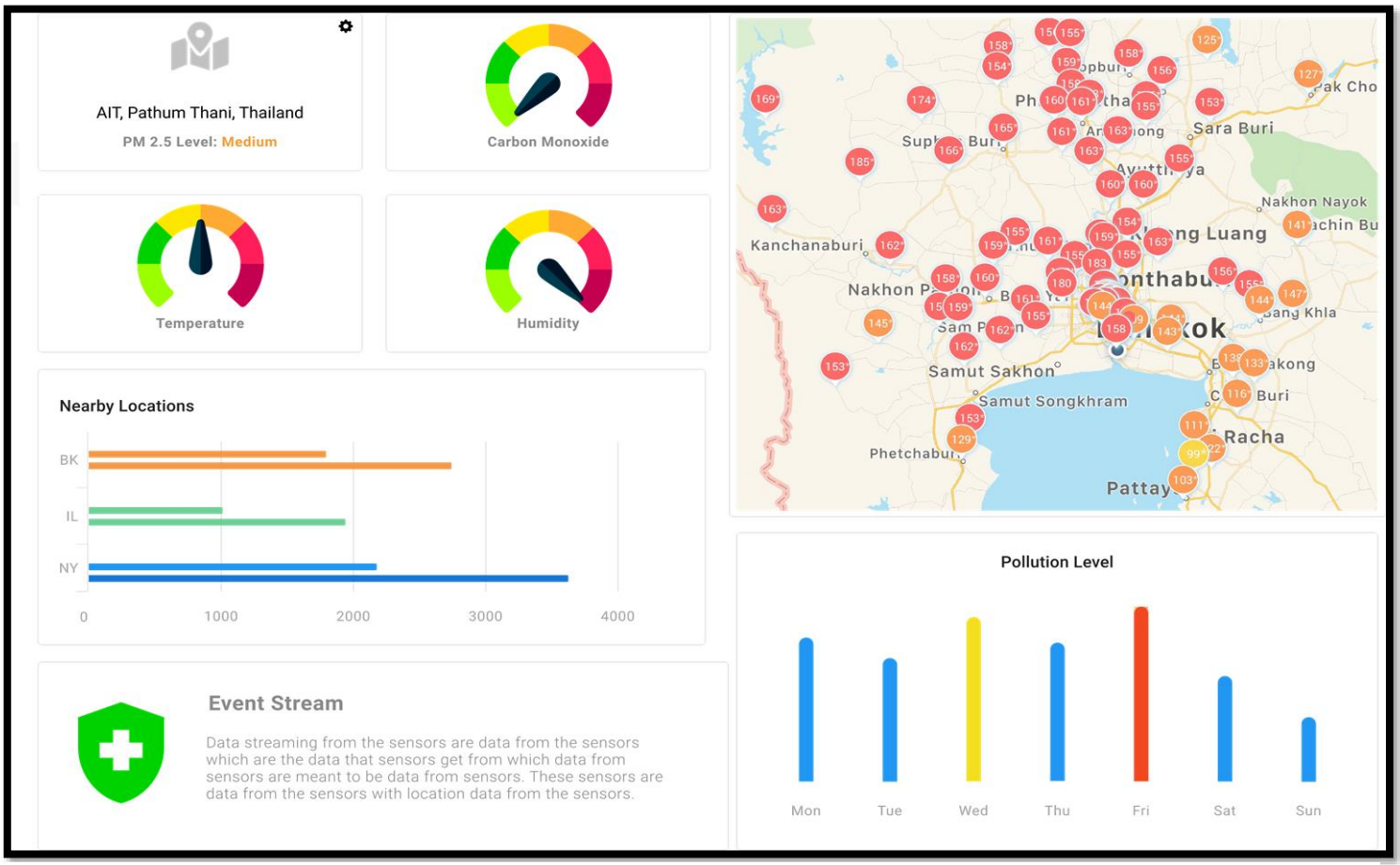


Figure 12. Dashboard for visitor

15.2 Interface for admin

Admin Login page: Once the admin enters the URL of the admin webpage, login-form will pop-up. Once the correct credentials (username and password) are given, admin will be directed to the admin page. This page contains various menu options in the sidebar.

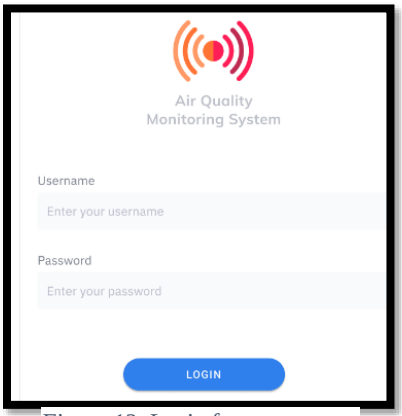


Figure 13. Login form

15.3 Admin Dashboard

Once logged in, admin will be directed directly to the dashboard. Admin will be able to access the dashboard that contains the visualization same as visitors as explained above.

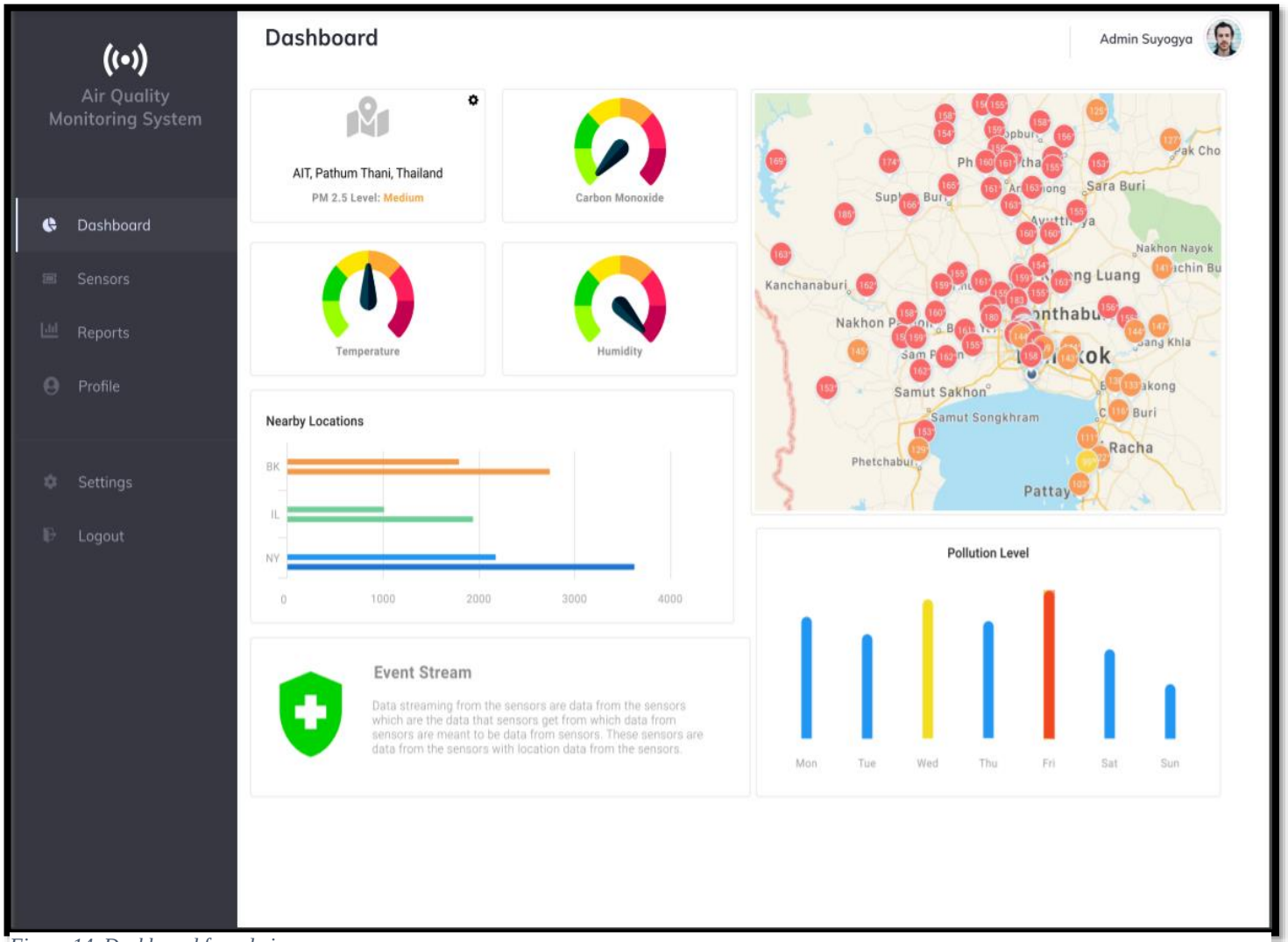


Figure 14. Dashboard for admin

15.4 Sensors Management

This menu contains the registration of new sensors as well as edit, disable and delete options for the registered sensors. It also displays the details about the registered sensors such as: Name, Model, API, type, and Region.

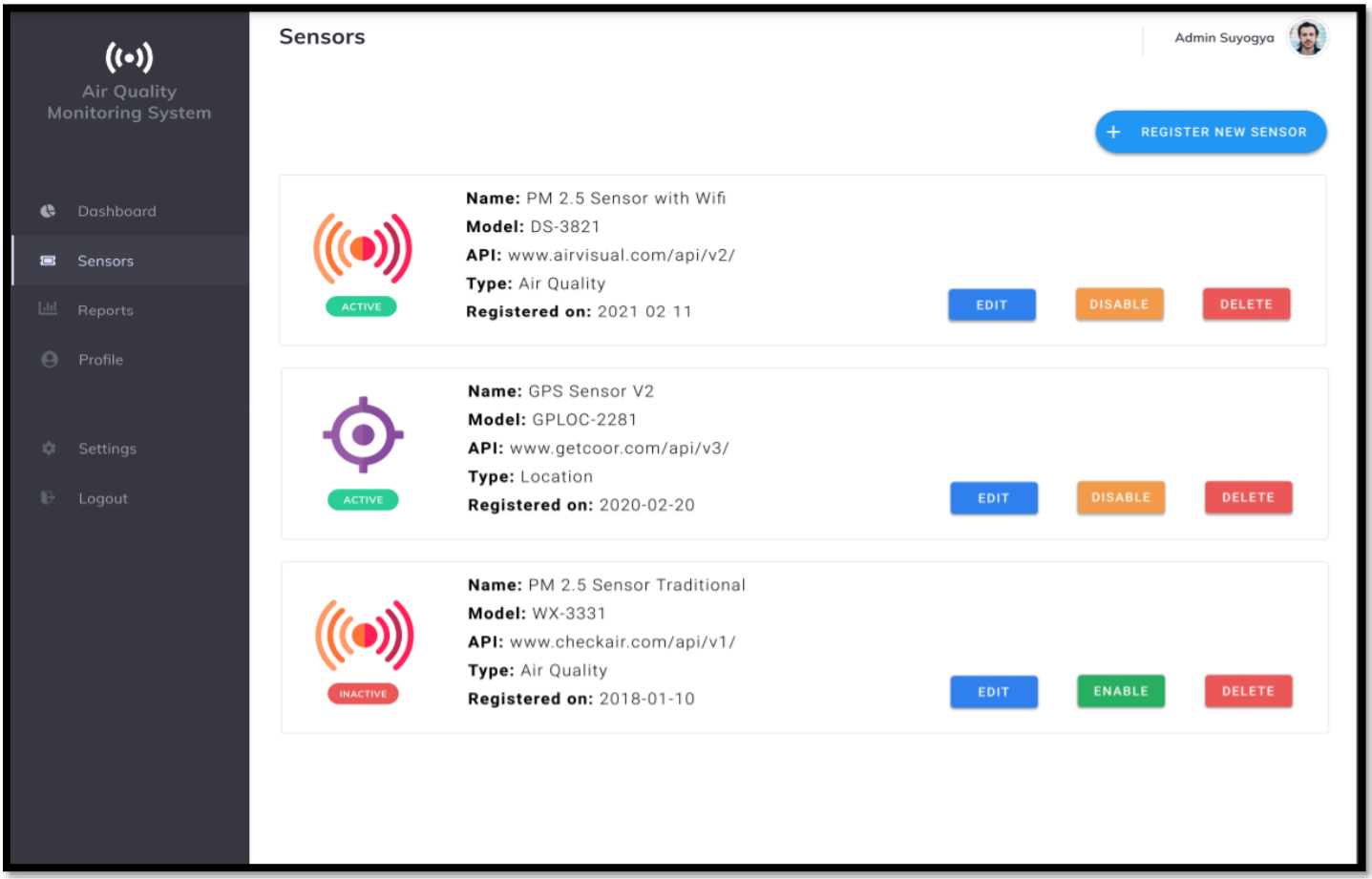


Figure 14. Sensor Management

15.5 Reports

Admin can generate reports in charts, tables, and other visualization forms, as per the requirement, using this Reports menu. It contains forms fields like Title of Report, Descriptive Content, Export File Format, select sensors option, Date, and generate submit button.

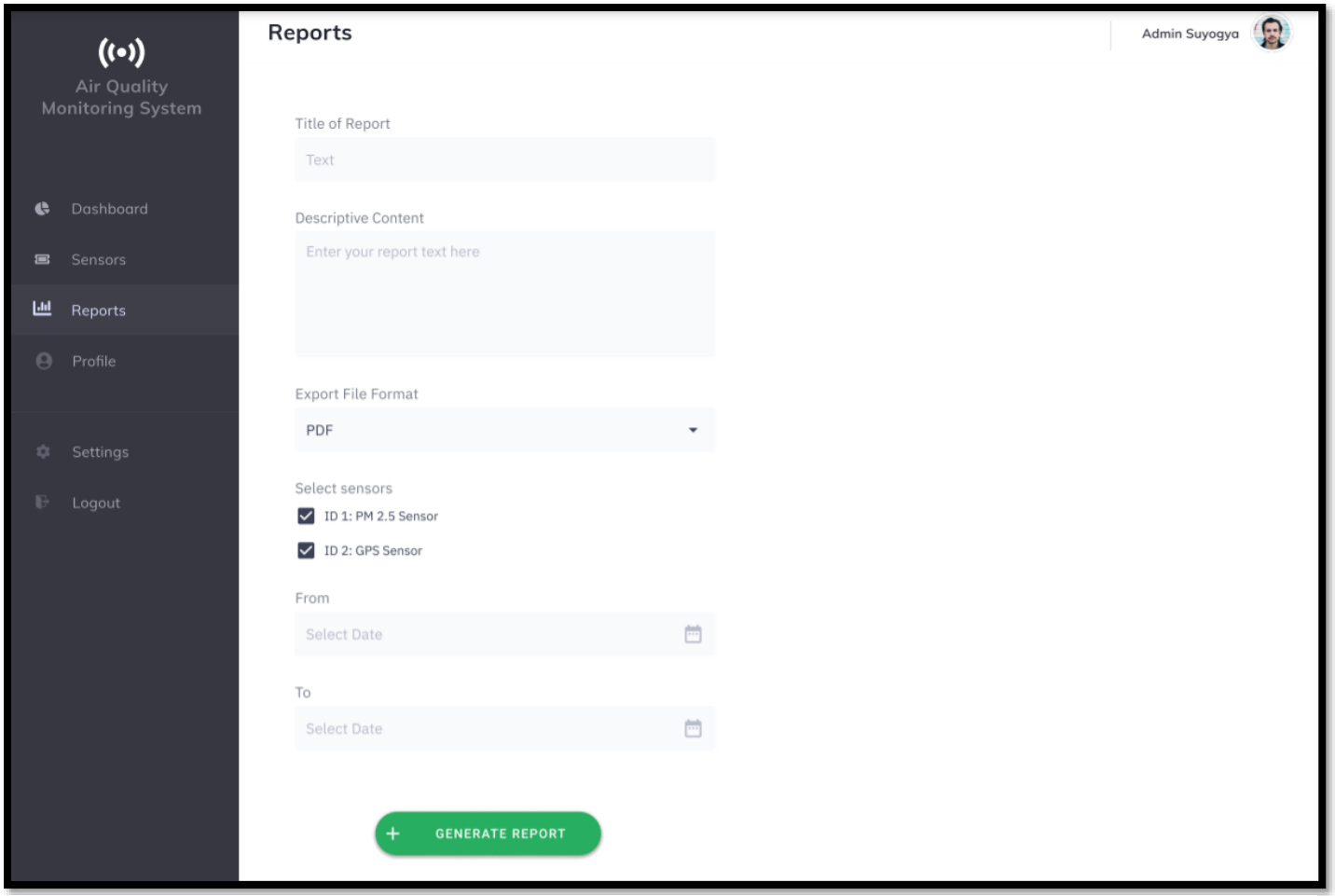


Figure 15. Generate report.

15.6 Logout and Update Profile

Admin can logout from the admin page with this menu. Once logged out, admin will be redirected to the login page.

This menu contains the information of the admin. Fields like Name, Email and Change Password options are available. Admin can update these fields, if needed.

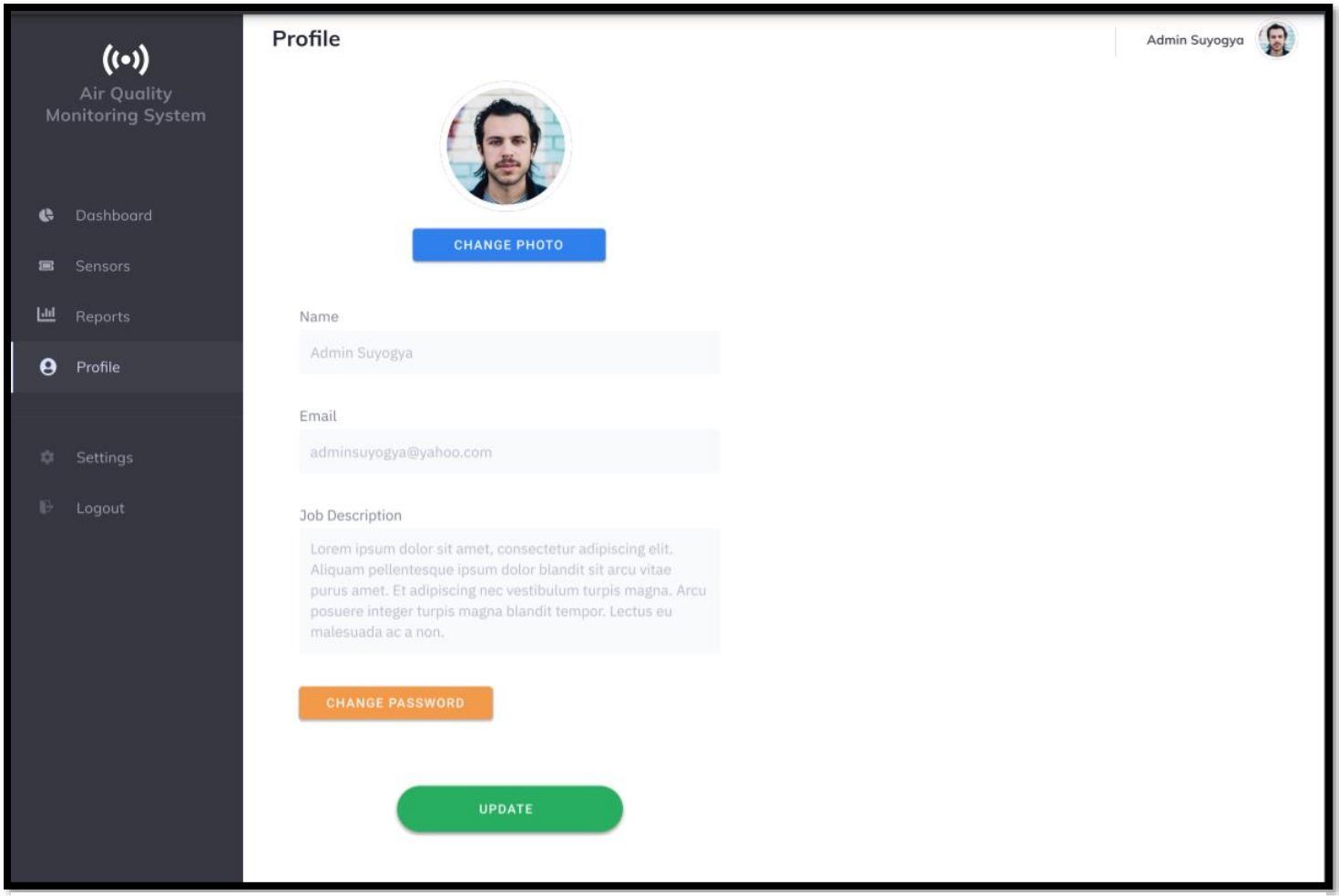


Figure 16. Update admin profile

15.7 Parameter Settings

Admin is privileged to update parameters of the system. As per the requirement admin can modify dashboard tiles, dashboard layout, manage data and data source with available menu like Enable/Disable Dashboard Tiles, Modify Dashboard Layout, Manage Data Sources and Data Backup/Restore.

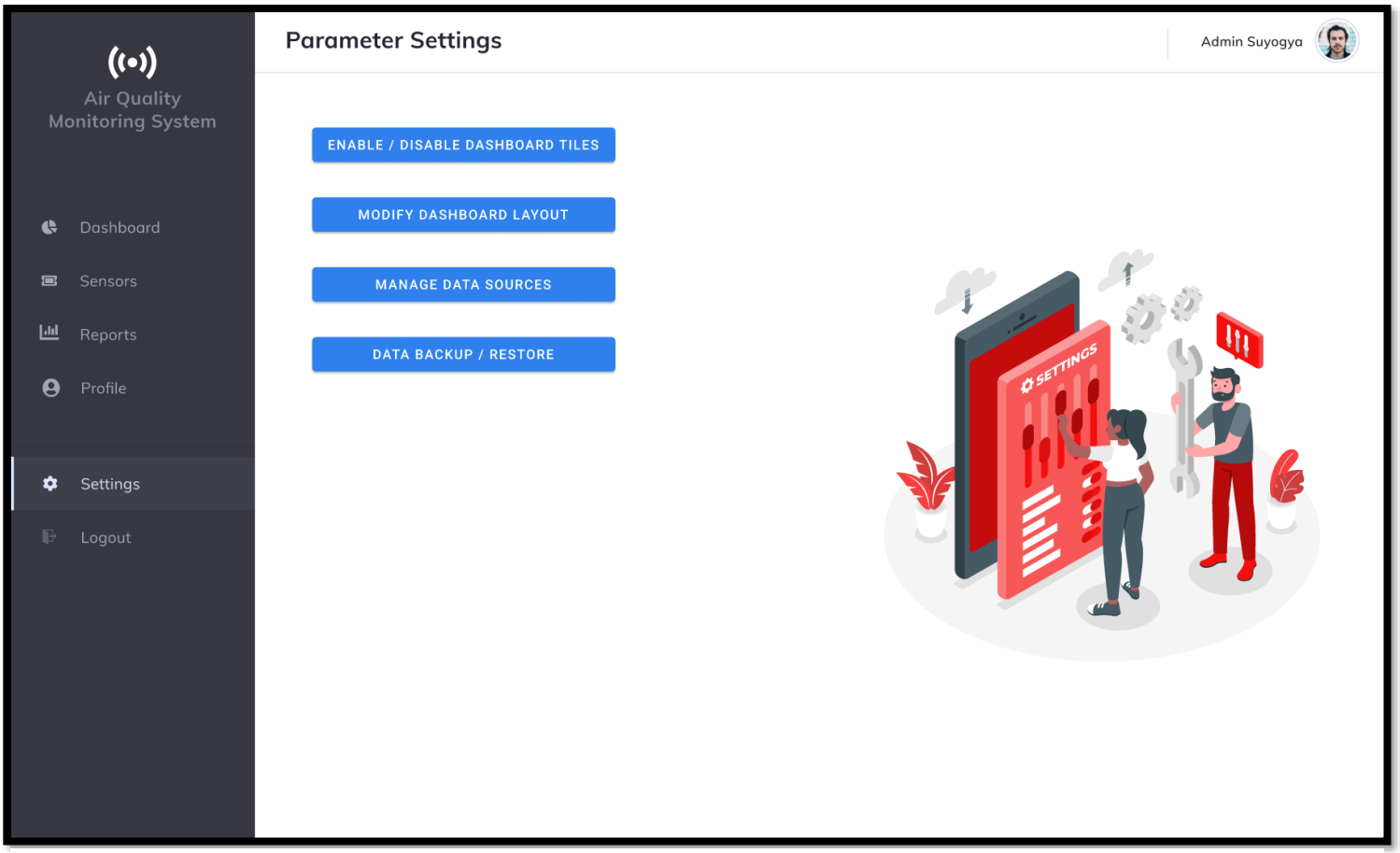


Figure 17. Update parameters

16. Appendix - Python Script to pull data

```
import pandas as pd
import csv
import os

########################################
##  DOWNLOAD DATA AND EXPORT TO CSV  ##
########################################

# Static Configurations
CSV_OUTPUT_FILE = 'aq_data.csv'

# Fetch JSON from URL and convert to DataFrame
url = "http://www.air4thai.com/services/getNewAQI_JSON.php"
data = pd.read_json(url).stations
os.system("echo Data downloaded from API.")

# Flatten the nested dictionaries
data = pd.json_normalize(data)
os.system("echo Preprocessing started...")

# Drop unnecessary columns
data.drop(
    columns=['nameTH', 'areaTH', 'forecast', 'LastUpdate.PM25.unit',
       'LastUpdate.PM10.unit', 'LastUpdate.O3.unit', 'LastUpdate.CO.unit',
       'LastUpdate.NO2.unit', 'LastUpdate.SO2.unit', 'LastUpdate.AQI.Level', 'LastUpdat
e.AQI.aqi',
       'LastUpdate.PM10.value', 'LastUpdate.O3.value', 'LastUpdate.CO.value', 'LastUpda
te.NO2.value', 'LastUpdate.SO2.value'],
    inplace=True)

# Rename necessary columns
renamed_columns = {
    'stationID': 'station_id',
    'nameEN': 'station_name',
    'areaEN': 'area',
    'stationType': 'station_type',
    'lat': 'latitude',
    'long': 'longitude',
    'LastUpdate.date': 'date',
    'LastUpdate.time': 'time',
    'LastUpdate.PM25.value': 'pm_value'
    }
data.rename(columns=renamed_columns, inplace=True)
```

```
# To handle COMMAS, replace commas with dot
data.replace('\,','.',regex=True, inplace=True)

# Add 'province' column
# It is extracted as the last value of 'area' column
data["province"] = data.area.values[0].split('.')[-1].strip()
os.system("echo New column 'province' added.")

# Add 'date_time' column
# It is formed by combining existing date and time strings
data["datetime"] = data.date + " " + data.time

# Drop area column, date column, time column, name column
data.drop(columns=['date', 'time'], inplace=True)

# Export to CSV file
## Index Removed
## Header Removed
## Quotes added (to handle strings with comma)
data.to_csv(
    CSV_OUTPUT_FILE,
    index=False,
    quoting=csv.QUOTE_NONE,
    header=None)

# Print out the summary
os.system("echo File with name {} exported successfully.".format(CSV_OUTPUT_FILE))
os.system("echo # of rows: {}".format(data.shape[0]))
os.system("echo # of columns: {}".format(data.shape[1]))


#####################################################
##  COPY CSV FILE FROM HOST TO DOCKER CONTAINER  ##
#####################################################

### MAKE SURE YOUR DOCKER CONTAINER IS STARTED AND RUNNING

# Static Configurations
CONTAINER_ID = '0631d0791fa3'
HOST_CSV_FILE = '/home/srt/Documents/PythonTests/{}'.format(CSV_OUTPUT_FILE)
CONTAINER_FILE_PATH = '{}:/home/admin/data/'.format(CONTAINER_ID)

# Copy CSV file from host to container
os.system("docker cp {} {}".format(HOST_CSV_FILE, CONTAINER_FILE_PATH))
```

```
os.system("echo CSV file copied to docker container successfully.")


#############################################################
##  COPY HIVE SQL FILE FROM HOST TO DOCKER CONTAINER  ##
#############################################################

# Static Configurations
SQL_FILE = 'load_aq_data.sql'
HOST_SQL_FILE = '/home/srt/Documents/PythonTests/{}'.format(SQL_FILE)

# Copy SQL file from host to container
os.system("docker cp {} {}".format(HOST_SQL_FILE, CONTAINER_FILE_PATH))
os.system("echo SQL file copied to docker container successfully.")


#####################################################################
##  EXECUTE COMMAND TO LOAD SQL FILE IN DOCKER TO HIVE TABLE  ##
#####################################################################

# Static Configurations
HIVE_LOAD_COMMAND = 'hive -f /home/admin/data/{}'.format(SQL_FILE)

os.system("echo Executing HIVE SQL command...")
os.system("docker exec -it {} {}".format(CONTAINER_ID, HIVE_LOAD_COMMAND))
```