



Lakehead
U N I V E R S I T Y

PROJECT REPORT ON

**“AIR QUALITY MONITORING SYSTEM USING
BLOCKCHAIN”**

Submitted in partial fulfilment of the requirements for the award of the degree

MASTER OF SCIENCE

in

COMPUTER SCIENCE

By

Ananya Marigowda

0862889

amarigow@lakeheadu.ca

Supervisor

Dr Ruizhong Wei

Professor & Chair,

Department of Computer Science

ABSTRACT

Every year we are witnessing a lot of data being falsified, entering the biased data instead of the right data. In general, factories try to modify the data to avoid paying huge fines and getting shut down from the government.

And this is not only about the money, but this project might also help to get a solution to people who are suffering from dust and pollution in the air. The World Health Organization (WHO) estimates that 4.6 million people die each year from causes directly attributable to air pollution. For this death, one of the reasons is data modification. If the data was modified, we don't know the actual data, and we trust that those areas are good, and we did not concentrate on those areas to avoid air pollution. On the other side, those areas are filling with more and more dust.

Using Blockchain Technology can help us to overcome these kinds of problems in the air quality monitoring system. With this technology in the air quality monitoring system, nobody can change the actual data of air which is will be stored in the database. So, the blockchain is used as a database here with a web-based application front-end interface. The application shows the hourly air quality data reports and daily data graphical representations. It also allows authorized users to add new air quality data. All users can also view and validate the blockchain. The application is developed using Spring Tool Suite (STS) for user interface and back-end coding and MySQL as the database.

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without mentioning about the people who made it possible because "Success is the abstract of hard work & perseverance, but steadfast of all is encouraging guidance". So, I take this opportunity to acknowledge all those whose continuous guidance and encouragement served as a beacon light & crowned my effort, thus leading to the successful completion of this work.

I take immense pleasure in expressing my deep and sincere gratitude to my supervisor, Dr Ruizhong Wei, Professor & Chair, Department of Computer Science, Lakehead University for his irreproachable guidance, monitoring and constant encouragement throughout the course. His availability and creative suggestions were above and beyond what I expected. His patience, understanding, support and personal guidance have proven invaluable in the preparation of this work. I appreciate his detailed discussions during my time at Lakehead University, and, for deepening my understanding of my chosen field. It has been a pleasure working under him.

I also extend my gratitude to the faculty members of the Department of Computer Science for their instruction during my study. Coming from a different country, it is challenging and can get overwhelming sometimes. I am grateful to Lakehead University for providing me with such an environment to reach my goals and to prosper in the corporate world.

I would also like to thank all my friends, parents who stood behind me like strong pillars of support and never failed to present their immaculate suggestions and thus helped me in giving shape to this project as it is today.

INDEX

1.	Introduction	06
1.1	Topic Chosen for the Project.....	06
1.2	Objectives of the Project.....	06
1.3	Existing System.....	07
1.4	Proposed System.....	09
2.	Literature Survey	10
3.	System Analysis	20
3.1	Software Requirements.....	20
3.2	Hardware Requirements.....	20
4.	System Design	21
4.1	UML Diagrams.....	22
5.	System Implementation	34
5.1	Module Description.....	34
5.2	Code.....	36
6.	Output Screenshots	50
7.	Future Work	56
8.	Conclusion	56
9.	References	57

LIST OF FIGURES

Figure 1	ENERGEO maps.....	7
Figure 2	Aqicn charts.....	8
Figure 3	Why we use blockchain?.....	11
Figure 4	Traditional Database vs Blockchain.....	14
Figure 5	Example of Merkle root.....	15
Figure 6	Blockchain structure.....	16
Figure 7	Block class.....	16
Figure 8	computeHash method.....	17
Figure 9	computeMerkleRoot method.....	17
Figure 10	Blockchain class.....	18
Figure 11	Application process.....	18
Figure 12	Admin use case diagram.....	23
Figure 13	User use case diagram.....	24
Figure 14	Block and Blockchain class diagram.....	29
Figure 15	User Validator class diagram.....	29
Figure 16	User controller class diagram.....	30
Figure 17	Admin sequence diagram.....	31
Figure 18	User sequence diagram.....	32
Figure 19	User Activity diagram.....	33

1. INTRODUCTION

The main concept of this web application is to develop a Blockchain system that virtually stores and manages air quality data securely. Through this application, one can view the air quality data of a particular air sensor station and even can analyze the data through graphical representations date wise. Authorized users can also add new air quality data and add it to the blockchain. This helps in reducing human intervention to store these data and increases data security by storing them in the form of blocks. It is highly secure as its almost impossible to modify or hack the data.

1.1 TOPIC CHOSEN FOR STUDY:

Intending to build an efficient and reliable platform that reduces the amount of human intervention involved in managing and storing the air quality monitoring data, I have chosen to create a Web application with a backend blockchain system. This application acts as a bridge between the event database administrator (Admin) and the users (common people). Admin(s) can add new air quality data to the system and also manage the existing database blockchain. Users can only view the data and the blockchain.

1.2 OBJECTIVES OF THE STUDY:

Objectives shape the study's primary context. Therefore, each study has its own goals that determine the ideal route to reaching the targets. The objectives of the proposed system are as follows:

- To efficiently use blockchain technology as a database
- To ensures a public and permanent, tamper-proof record of all air quality data
- To accomplish a cost-effective blockchain-based solution for a pollution monitoring system

- To provide common people with an easy and understandable perspective of air quality
- To provide a system for logging, immutable storage and decentralized distribution of air quality data
- To assist enhance the air quality index and solve air pollution issues

1.3 EXISTING SYSTEM:

- We discovered a lot of techniques that have already been used, such as in ENERGEO, which resume displaying a map showing the air quality in different areas using colors for each level, but the issue with this technique that it is showing only annual air quality data.

Online source apportionment for decisions on effective measures
Routine Lenschow Approach
Emission inventory
Source selective modelling (ENERGEO, M. Schaap)

Figure ES.2 Relative contribution of international shipping emissions (in %) on annual mean NO_2 and $\text{PM}_{2.5}$ concentrations in the year 2005

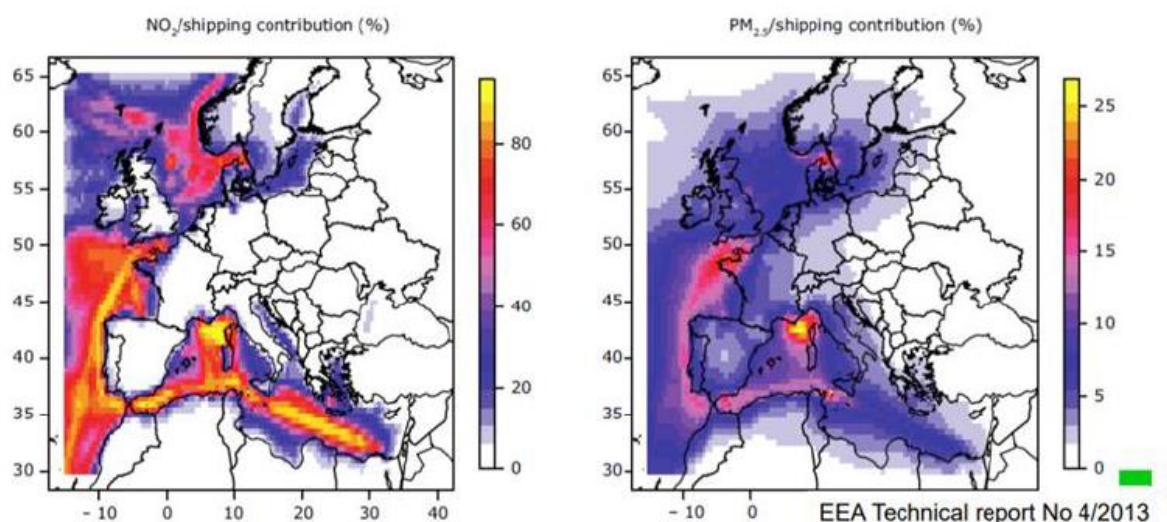


Figure 1: ENERGEO maps

- The aqicn.org website demonstrates the air quality in most of the countries in the world. It shows many characteristics such as humidity, wind, temperature, pressure, air molecules and primary pollutants (PM2.5, O3, NO2, etc.). But just like the previous system, it takes a long time to update and its prone to loss of data.

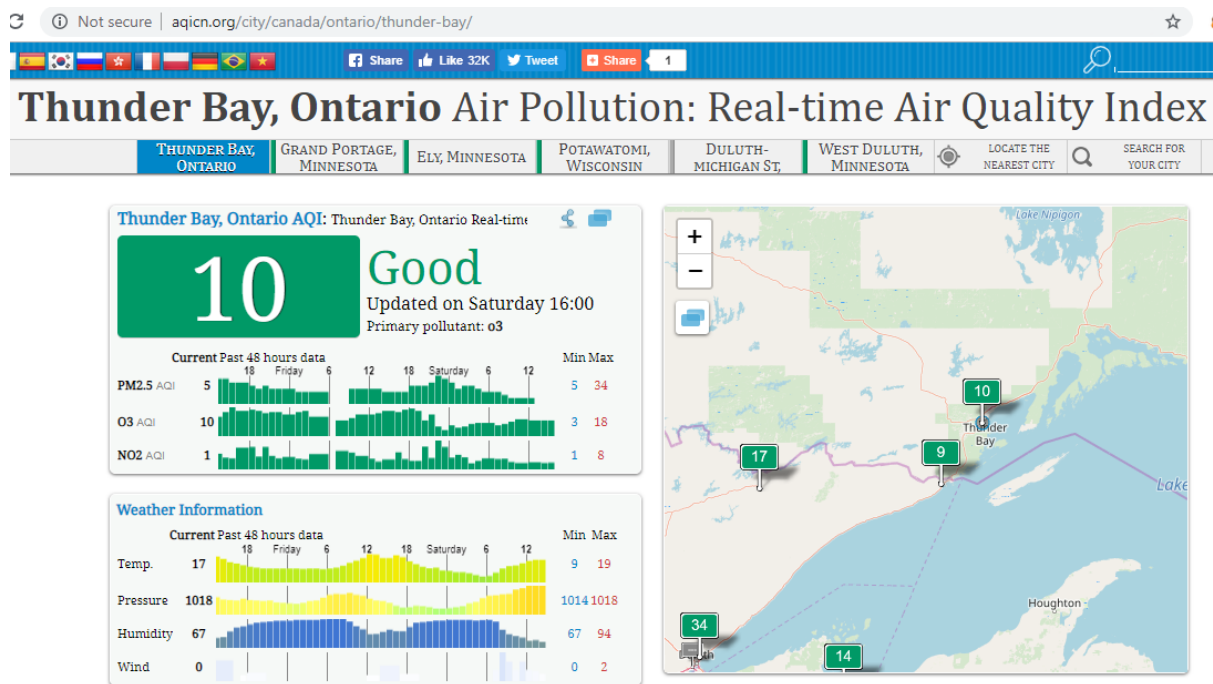


Figure 2: Aqicn charts

To summarize the results of this study, we found out that the main problem of previous methods is the time that engenders data integration problems; therefore, the misinterpretation and loss of data.

There is also another problem with hacking since the non-centralized data is easier to hack, and finally, the most crucial issue is that is if we talk about losing time, we talk about losing money too. Hence, through the proposed system we are trying to find solutions for scalability, cost and trust issues.

1.4 PROPOSED SYSTEM:

The proposed system is quite different from the existing applications as it can be integrated with air sensor devices to collect air quality data in real-time and store it using blockchain technology. A decentralized network for air pollution monitoring is a possible solution for all the problems mentioned above. Decentralized network endpoints can be air sensor devices, sensors owned by individuals like smartphones or wearable devices. With blockchain technology, we can connect all these sensors to create a mesh network.

A blockchain is a decentralized database that makes it possible to create a ledger of information and share it among a distributed network of devices. Every sensor on the network would be able to take action on the ledger without a centralized server. This enables completely distributed decentralized interactions.

For this project, I am taking the year 2017 Air quality data from one air sensor station in Thunder Bay. This dataset is downloaded from the government website: <http://www.airqualityontario.com> . First, I have developed a Java code to create blocks and blockchain from the above data. Then I have created a front-end Spring Boot Web Application to access and view this data. Through the application, authorized users can also add new blocks for new dates other than the dates, i.e. already existing in the blockchain. This helps in avoiding duplication and modification of data. And all users can view the air quality data and the blockchain.

2. LITERATURE SURVEY

BLOCKCHAIN TECHNOLOGY

A **Blockchain** is a database that is shared across a network of computers. Once a record has been added to the chain it is very difficult to change. To ensure all the copies of the database are the same, the network makes constant checks. Blockchains have been used to underpin cyber-currencies like bitcoin, but many other possible uses are emerging. It is a public ledger that can store transaction records or any other data. It is owned by no one, with a copy of it being stored on many personal computers around the world. Anyone can use it and help run the network. This often removes the need for middlemen and allows users to interact in a peer to peer way.

- Keywords:

- ✓ **Participants:** Members of a Business network
- ✓ **Ledger:** The system of record for a business (Business will have multiple ledgers for multiple Business networks in which they participate).
- ✓ **Transaction:** An asset transfers onto or off the ledger.
- ✓ **Contract:** Conditions for a transaction to occur

- The requirement of Blockchain for Business:

- ✓ **Shared ledger:** Records all transactions across the business network, given permission, so participants see only appropriate transactions
- ✓ **Smart contract:** Business rules implied by the contract which is embedded in the blockchain and executed with the transaction
- ✓ **Privacy:** The ledger is shared, but participants require privacy, each transaction needs to be authenticated.
- ✓ **Trust:** The ledger is a trusted source of information, achieved through consensus, provenance, immutability and finality

We use blockchain for these reasons:



Figure 3: Why we use blockchain?

How does blockchain work?

As mentioned earlier, blockchain is a series of blocks joined to one another. Whenever a block stores new data, it is added to the previous existing chain thereby increasing the length of the chain. In order to add the block to the existing blockchain, the transaction should be complete. Four steps are involved in a successful transaction^[1]. They are:

- The transaction must occur
- The occurred transaction must be verified
- The verified transaction must be stored in a block
- A unique hash must be given to the stored block

The hashed block can only be added to the blockchain, and this hash plays a significant role in block identification and helps in rectifying the blockchain if any block present in the chain is manipulated.

Role of blockchain in the banking sector:

Blockchain seems to be ideal in the banking sector. All banks need to communicate with one another when transactions come into place. Using this technology, banks can cooperate under the same blockchain and push their customer's transactions. This improves the transparency, facilitates the transactions auditing,

helps in creation of a decentralized architecture which in turn minimizes the transaction costs [\[2\]](#).

Advantages and disadvantages of blockchain:

Advantages

- Transparent technology with decentralized architecture
- Improved accuracy by verification without human involvement
- Since there is no third-party involvement, the cost is reduced
- The concept of decentralization makes it difficult to modify or tamper the existing blockchain
- Each and every transaction made is secure, private and efficient
- Immutable, i.e. hard to modify the chain
- Provides security by using cryptography

Disadvantages

- Highly complex
- The size of the chain keeps increasing after every transaction performed
- Hard to store with increasing size
- Requires more resources

Use of blockchain as a database:

Blockchain is simply a new type of database. Rather than traditional databases (SQL or NoSQL) that are controlled by single entities, blockchain can be shared by a group of non-trusting parties without requiring a central administrator. This is because if trust and robustness aren't an issue, there's nothing a blockchain can do that a regular database cannot. Remember, blockchains are useful for sharing data with a group of non-trusting parties.

Blockchains offer a way to replace the organizations and their centralized databases with a distributed database that is secured by cryptography and consensus mechanisms. This distributed database eliminates the single point of failure — also called the honey pot problem — characteristic of centralized databases. Traditional

databases are prone to attacks because the attackers only have to target a central server.

Overview of Public vs Private Blockchains

When most people think of blockchain, they think of blockchains like Bitcoin and Ethereum. Anyone can participate in these blockchains, which are not as public blockchains. However, these public blockchains are not the blockchains that most companies will utilize.

Rather, companies, and specifically, groups of non-trust parties, will utilize what are called private blockchains. In private blockchains, there is a control layer built into the protocol, which allows for network participants to have control over who can join the network and participate in the consensus process (hence why it's called private not public). Rather than allowing everyone and anyone to become a node and verify transactions, private blockchains have a select group of companies/organizations that can become nodes. Private blockchains, therefore have a very different level of security than public blockchains like Bitcoin.

For example, think about a consortium of banks that need a shared ledger but that don't trust each other so that one bank can host the entire ledger. Currently, the consortium of banks would find a trusted 3rd party to host the ledger. However, with private blockchains, this can be avoided.

Traditional Databases vs. Blockchains:

Traditional Databases use client-server network architecture where a user can modify data that is stored on a centralized server. Irrespective of their structure (SQL or NoSQL), a single authority controls the database and authenticates a client's credentials if they want to access it. This means that read and write access is only possible via applications that are controlled by the entity. If the security of the single authority is compromised, the data can be altered, deleted, or leaked to the public.

Blockchains consist of dozens, hundreds, and thousands of nodes. At the time of this writing, the Ethereum blockchain has over 15,000 nodes. Each node is essentially another admin; every node verifies new additions to the blockchain and can enter new data into the database. For data to be added to the blockchain, the majority of nodes must reach consensus. This consensus mechanism guarantees the security of the network, but really slows down performance.

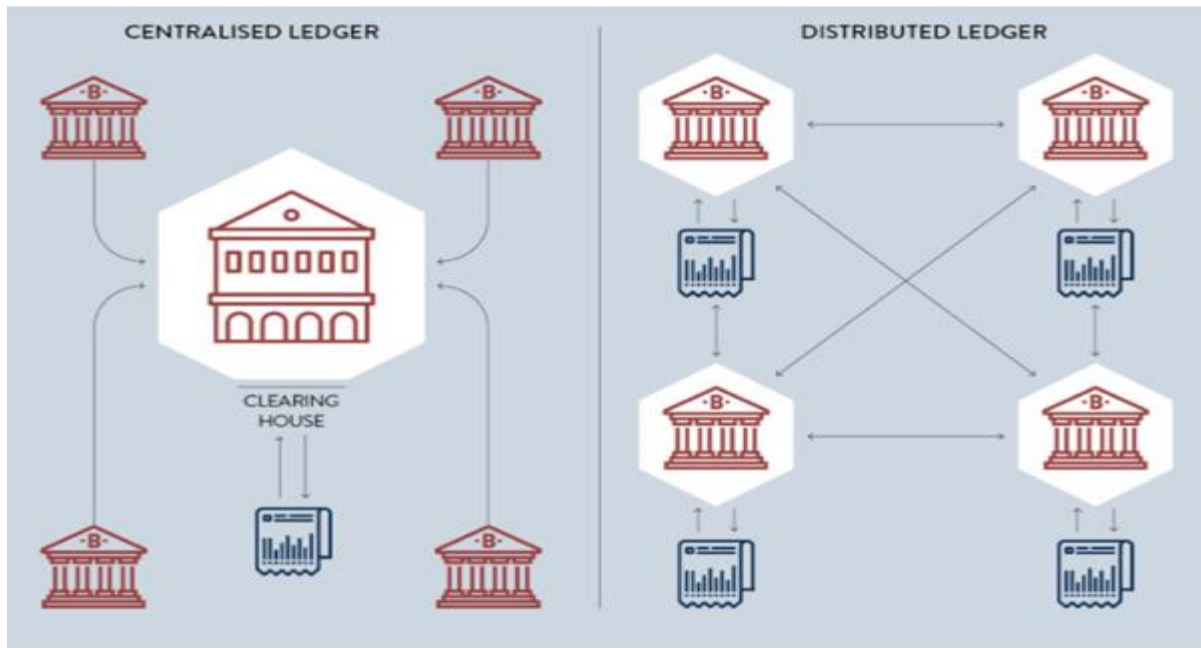


Figure 4: Traditional Database Vs Blockchain

With blockchain read and write access involves multiple parties, rather than just one who gives access via applications. On a public blockchain, all past transactions stay on the blockchain, rather than updating and erasing past entries, as is the case with traditional databases. Therefore, blockchains are referred to as immutable and distributed ledgers.

What is the Merkle root?

Merkle trees are a fundamental part of blockchain technology. A Merkle tree is a structure that allows for efficient and secure verification of content in a large body of data. This structure helps verify the consistency and content of the data.

How do Merkle trees work?

A Merkle tree summarizes all the transactions in a block by producing a digital fingerprint of the entire set of transactions, thereby enabling a user to verify whether or not a transaction is included in a block.

Merkle trees are created by repeatedly hashing pairs of nodes until there is only one hash left (this hash is called the Root Hash, or the Merkle Root). They are constructed from the bottom up, from hashes of individual transactions (known as Transaction IDs).

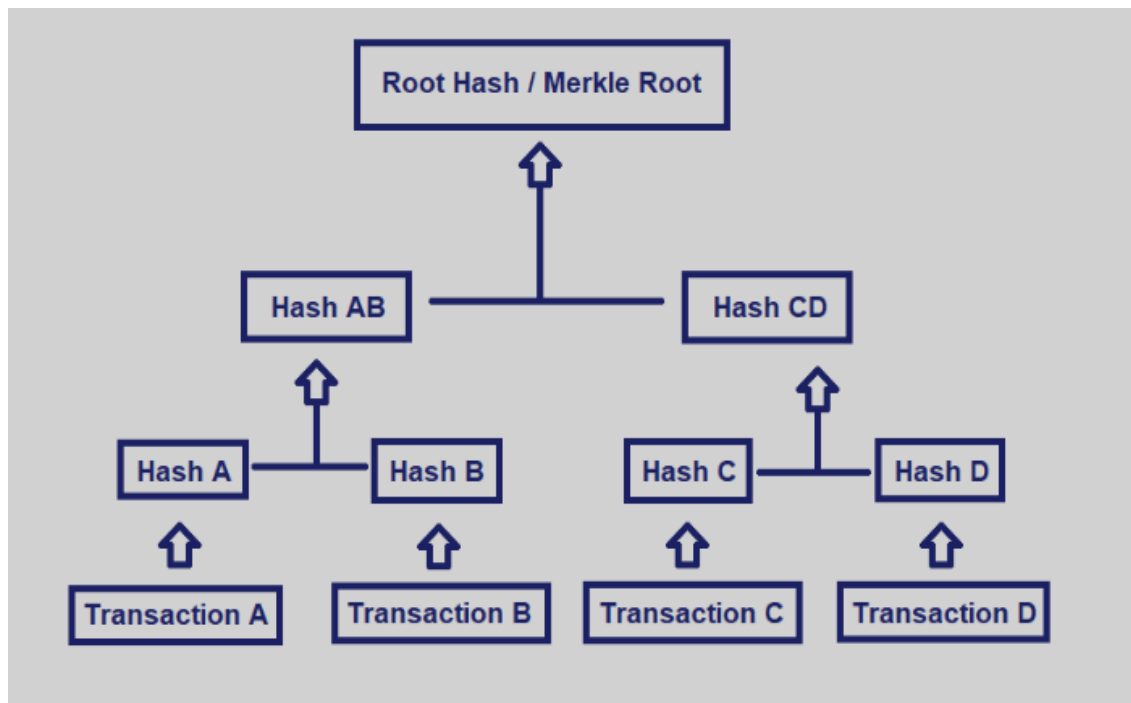


Figure 5: Example of Merkle root

Each leaf node is a hash of transactional data, and each non-leaf node is a hash of its previous hashes. Merkle trees are binary and therefore require an even number of leaf nodes. If the number of transactions is odd, the last hash will be duplicated once to create an even number of leaf nodes.

The Merkle Root summarizes all of the data in the related transactions and is stored in the block header. It maintains the integrity of the data. If a single detail in any of the transactions or the order of the transaction's changes, so does the Merkle

Root. Using a Merkle tree allows for a quick and simple test of whether a specific transaction is included in the set or not.

Proposed private blockchain for project:

A blockchain is just a chain/list of blocks. Each block in the blockchain will have its own digital signature/hash, digital signature of the previous block, Merkle root and have some data/transactions.

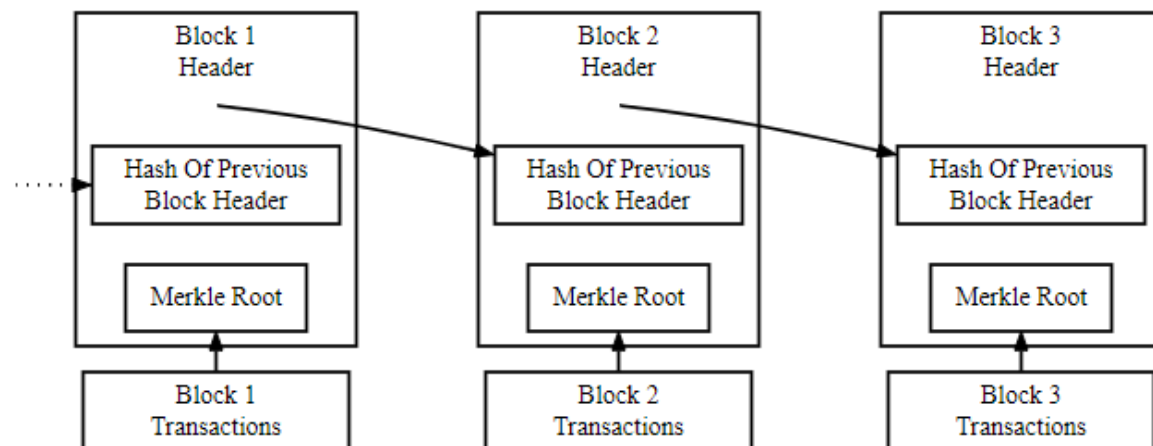


Figure 6: Blockchain structure

I have developed a Blockchain application using Java. Started by creating the classes that will implement the Blockchain approach, as shown below:

```

public class Block {

    public long timeStamp;
    private int index;
    private AirData airData;
    private String hash;
    private String previousHash;
    private static String merkleRoot;
    private String nonce = "0000";

    public Block(int index, AirData airData, String previousHash) {
        this.index = index;
        this.airData = airData;
        this.previousHash = previousHash;
        this.timeStamp = System.currentTimeMillis();
        computeMerkleRoot();
        computeHash();
    }
}

```

Figure 7: Block class

As you can see, our basic Block contains a timeStamp variable that holds the time and date the block was added in 12-byte binary format. The variable index is the block number. The variable String hash will hold our digital signature of the block. The variable previousHash to hold the previous block's hash, String AirData to hold our block data, the variable String merkleRoot holds the Merkle hash root value of the block and the variable nonce in an int called difficulty, this is the number of 0's that must be solved to get the hash value of the block.

```
public void computeHash() {
    setHash(SHA256.generateHash(String.valueOf(timeStamp)
        + String.valueOf(index)
        + merkleRoot + nonce
        + previousHash));
}
```

Figure 8: computeHash method

Next we will need a way to generate a digital signature, there are many cryptographic algorithms you can choose from, however SHA256 fits just fine for this example. We can import java.security.MessageDigest to get access to the SHA256 algorithm.

```
public void computeMerkleRoot() {
    List<String> treeList = merkleTree();
    setMerkleRoot(treeList.get(treeList.size()-1));
}

public List<String> merkleTree() {
    ArrayList<String> tree = new ArrayList<>();
    ObjectMapper oMapper = new ObjectMapper();

    @SuppressWarnings("unchecked")
    Map<String, String> map = (Map<String, String>)oMapper.convertValue(airData, Map.class);
    for (String s : map.keySet()) {
        tree.add(SHA256.generateHash(String.valueOf(map.get(s))));
    }
    int levelOffset = 0;
    for (int levelSize = map.size(); levelSize > 1; levelSize = (levelSize + 1) / 2) {
        for (int left = 0; left < levelSize; left += 2) {
            int right = Math.min(left + 1, levelSize - 1);
            String tleft = tree.get(levelOffset + left);
            String tright = tree.get(levelOffset + right);
            tree.add(SHA256.generateHash(tleft + tright));
        }
        levelOffset += levelSize;
    }
    return tree;
}
```

Figure 9: computeMerkleRoot methods

Later we need to calculate the Merkle root hash of the block. For that I am using the above-shown methods.

Now we need to create a chain from the created blocks for which I have designed another class called Blockchain.

```
public class Blockchain {

    static ArrayList<Block> chain = new ArrayList<Block>();

    public void addBlock (AirData airData){
        String previousHash;

        if (chain.size() ==0){
            previousHash= SHA256.generateHash("genesis");
        }else{
            previousHash = chain.get(chain.size()-1).getHash();
        }
        Block b = new Block(chain.size()+1,airData,previousHash);
        chain.add(b);
    }

    public static boolean verifyBlockchain(){

        for (int i = chain.size() - 1; i > 0; i--) {

            if (chain.get(i-1).getHash().equals(chain.get(i).getPreviousHash())) {
                continue;
            } else {
                return false;
            }
        }
    }
}
```

Figure 10: Blockchain class

This class has two methods. First addBlock() is used to create a blockchain and verifyBlockchain() is used to check the blockchain is valid and intact.

In conclusion, this figure below will explain how it works:

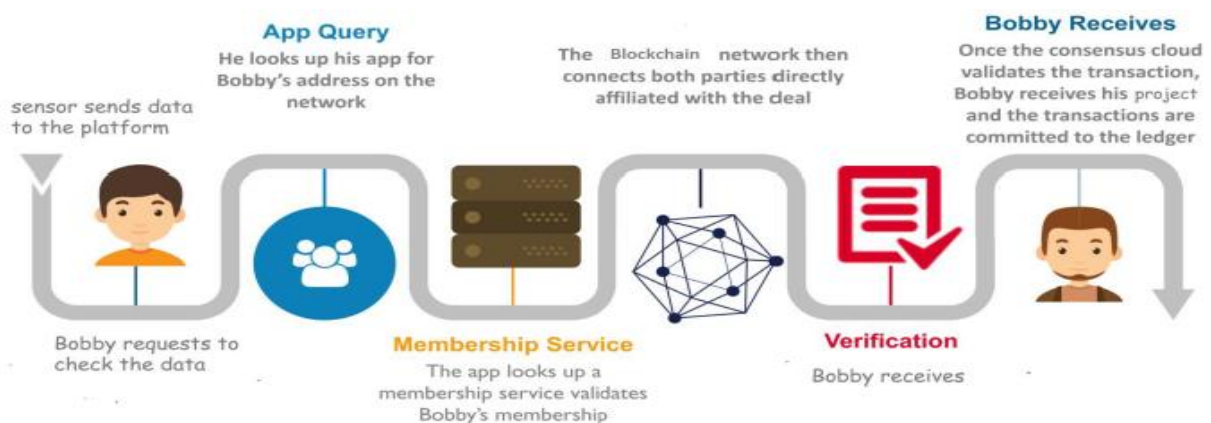
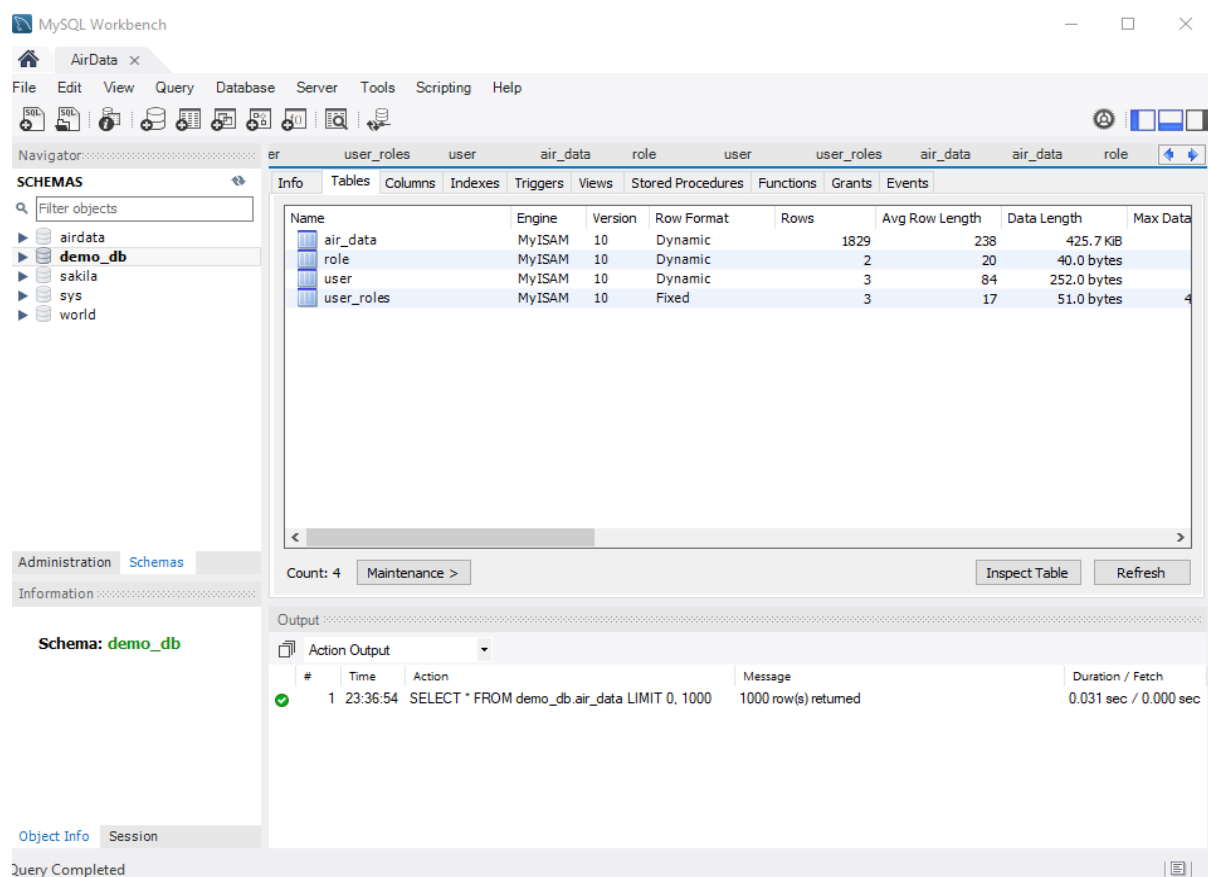


Figure 11: Application process

Database:

Here I have used MySQL database to store all the data related to the project (DB name – demo_db). It has four tables, they are:

1. Air_data – it contains the year 2017 air quality data of Thunder Bay
2. Role – it contains types of roles in the web application, i.e. Admin or user
3. User – it contains a list of user IDs and their passwords(encrypted)
4. User_roles – it contains a list of users and the role assigned to them



Dataset details:

- Source: <http://www.airqualityontario.com>
- Area: Thunder Bay
- Station ID: 63203
- Hourly data of NO, NO₂, NO_x, O₃ and PM_{2.5} Concentrations
- From: January 1 2017 EST To: December 31 2017 EST
- Station Address: 421 James St. S.
- Latitude: 48.379389 Longitude: -89.290167
- Air Intake Height: 15 meters Elevation: 192 meters
- Unit: parts per billion (ppb)
- Remarks: -999 for missing data. 9999 for invalid data.

3. SYSTEM ANALYSIS

3.1 SOFTWARE REQUIREMENTS:

OS	: Windows 10
Programming Language	: Java
Frontend Technologies	: HTML, CSS, JavaScript
Editor/IDE	: Spring Tool Suite 3
Web server	: Embedded Tomcat server
Database	: MySQL

3.2 HARDWARE REQUIREMENTS:

Processor	: Intel i3 (or above)
Hard Disk	: 128 GB (min.)
RAM	: 4 GB

4. SYSTEM DESIGN

Web Application:

An application that runs on a remote server and can be accessed through a web browser over a network (such as the internet) is called a Web Application.

Developing a web application:

Development of a web application mainly involves three parts. They are:

- a. Front-end
- b. Back-end
- c. Joining the front and the back-end

There are various technologies used to develop the applications such as Ajax, CSS, HTML, Java, JavaScript, Node.js, OSGI, Perl, PHP, Python, Ruby etc., and they are classified as front-end and back-end technologies based on the functionalities they provide.

a. Front-end:

The front-end of a website is the part the user interacts with. In terms of Open System Interconnection (OSI) layers, the front-end may be referred to the *Presentation Layer*. It refers to all the code that runs on the client-side. This acts as an interface between the user and the back-end and enables the users to generate requests. If the request made is processed successfully, it also displays the response generated from the back-end.

The front-end is developed using front-end technologies such as HTML, CSS, JavaScript, etc., on an editor such as Atom, Visual Studio Code, Spring Tool Suite etc.

b. Back-end:

The back-end of the website refers to the server-side of an application that enables communication between the database and the browser. In terms of Open System Interconnection (OSI) layers, the back-end may be referred to the *Data Link Layer*, also called *Data Access Layer*. The primary purpose of the back-end is to service requests, generation of appropriate responses to the requests made by the user on the front-end. Most common back-end processors are Web servers and databases.

A Web server is a processor that can respond to an HTTP request and generate a return response to it, whereas a Database is storage medium that stores all the data generated, this data is electronically accessible through a computer system. Most popular back-end technologies are Java, C++, C#, .NET, Node.js etc. and the popular IDE's are Eclipse, NetBeans, Android Studio, Microsoft Visual Studio, etc. Here in my project, for developing the front-end, I have used HTML, CSS as technologies. For back-end, I have used Spring boot framework as technology and Spring Tool Suite 3 as the IDE, MySQL as the database and Embedded Apache Tomcat as the webserver.

4.1 UNIFIED MODELING LANGUAGE DIAGRAMS (UML)

The idiom "*A picture is worth a thousand words*", perfectly describes the concept of UML. UML stands for "UNIFIED MODELING LANGUAGE". UML allows the software engineer to express an analysis model using the modelling notation that is governed by a set of syntactic-semantic and pragmatic rules. UML is broadly classified into two categories among which one represent the structural information (structure diagrams) and the other represents the behavioral information (behavior diagrams).

Use Case Diagrams

Use Case diagram is a behavioral diagram and is used to represent the relationship between the user and different use cases in which the user is involved. Actors and Use Cases are used to describe the system. Actors represent the users involved, and the Use Cases (Ovals) represent the list of steps user performs.

Admin Use Case Diagram

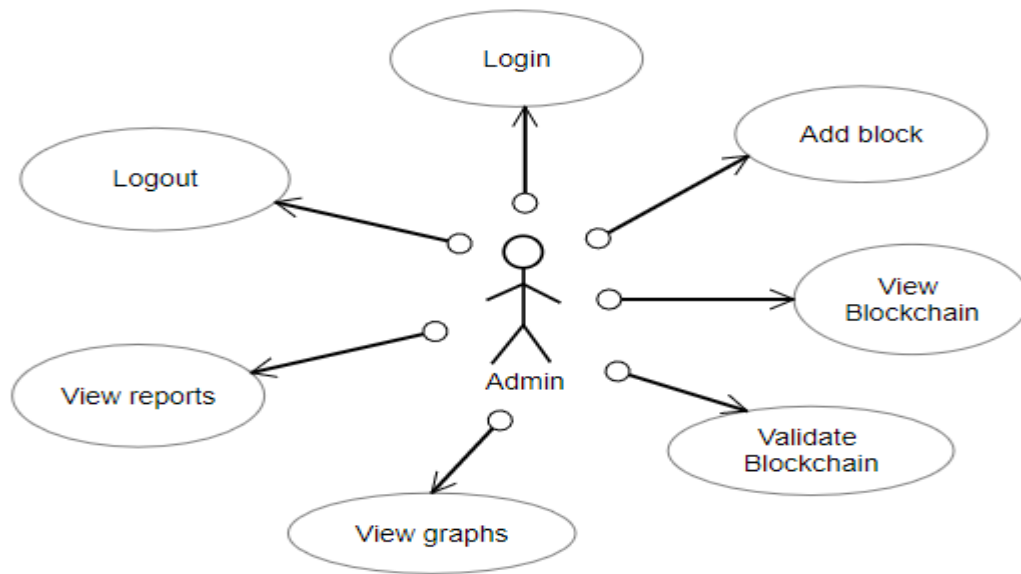


Figure 12: Admin use case diagram

In the above Admin Use Case Diagram,

- Admin is the actor
- The Use Cases are
 - Login
 - Add block
 - Validate blockchain
 - View reports
 - View blockchain
 - View graphs
 - Logout

User Use Case Diagram

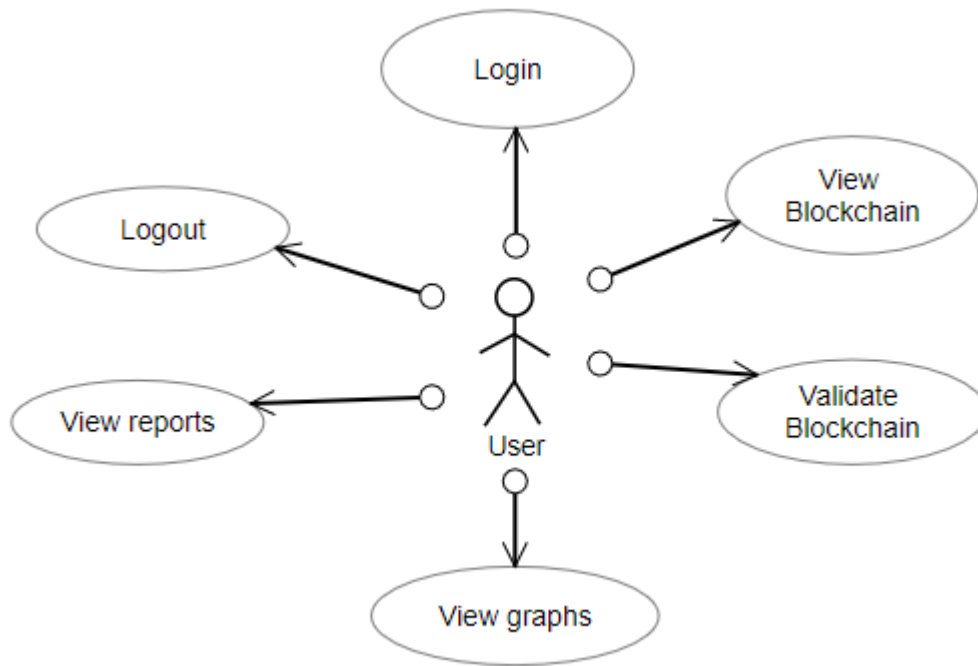


Figure 13: User use case diagram

In the above User Use Case Diagram,

- User is the actor
- The Use Cases are
 - Login
 - Validate blockchain
 - View reports
 - View blockchain
 - View graphs
 - Logout

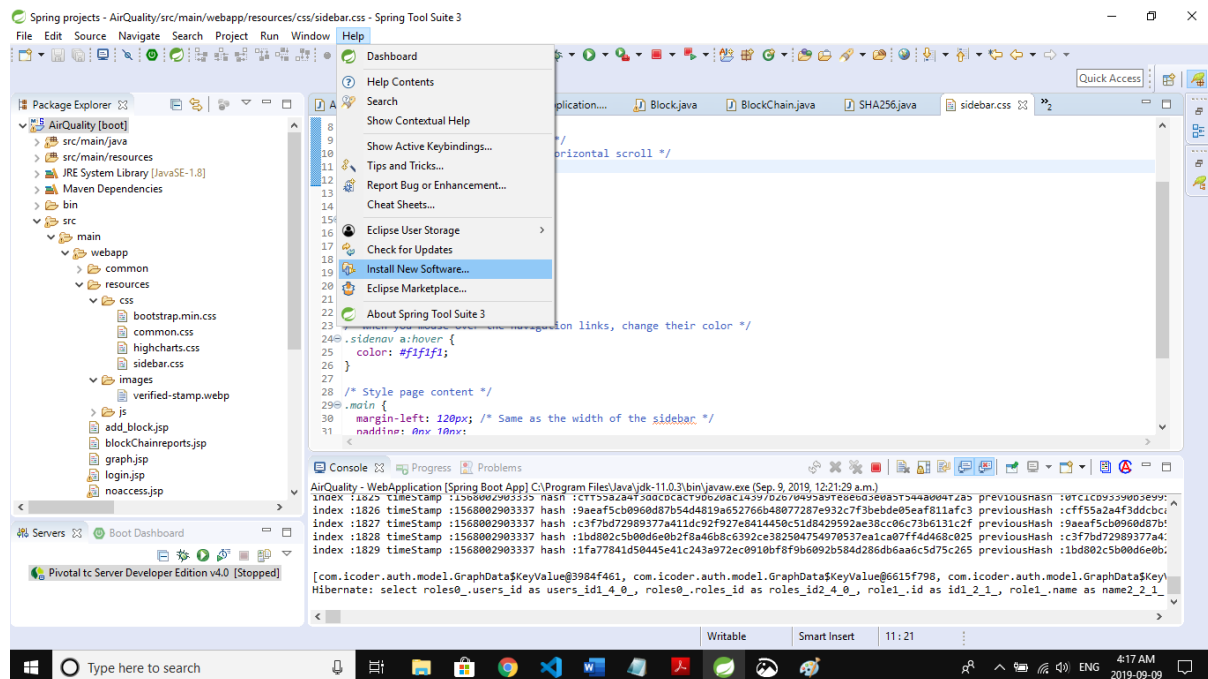
Class Diagram

A class diagram is a structural diagram and is used to describe the structure of a system by representing the system's classes, attributes, operations and the relationship among the objects involved. A class diagram can be generated in different ways using different tools such as StarUML, object aid, visual paradigm, ObjectAid etc. Here I have used the ObjectAid plugin available for eclipse to generate the class diagrams. To do this the ObjectAid plugin should be installed in the Spring Tool Suite 3.

ObjectAid Installation

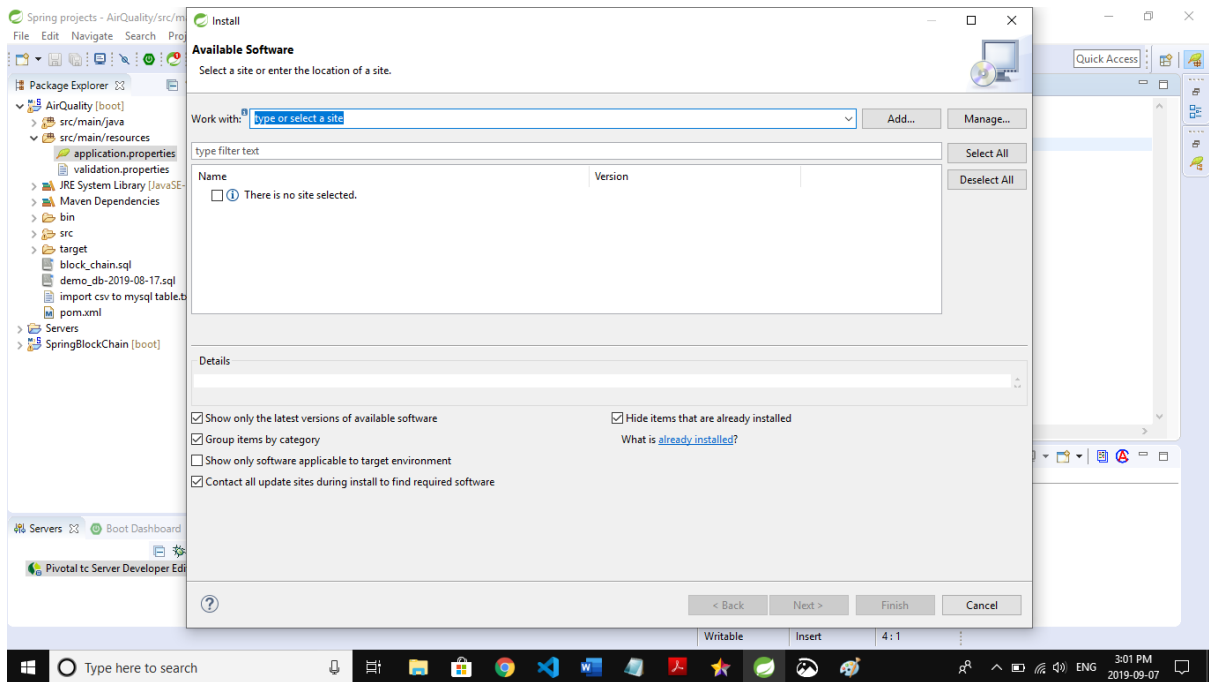
The following are the steps to install ObjectAid within Spring Tool Suite.

Step1: In the Spring Tool Suite main menu, open Help > Install New Software



Air Quality

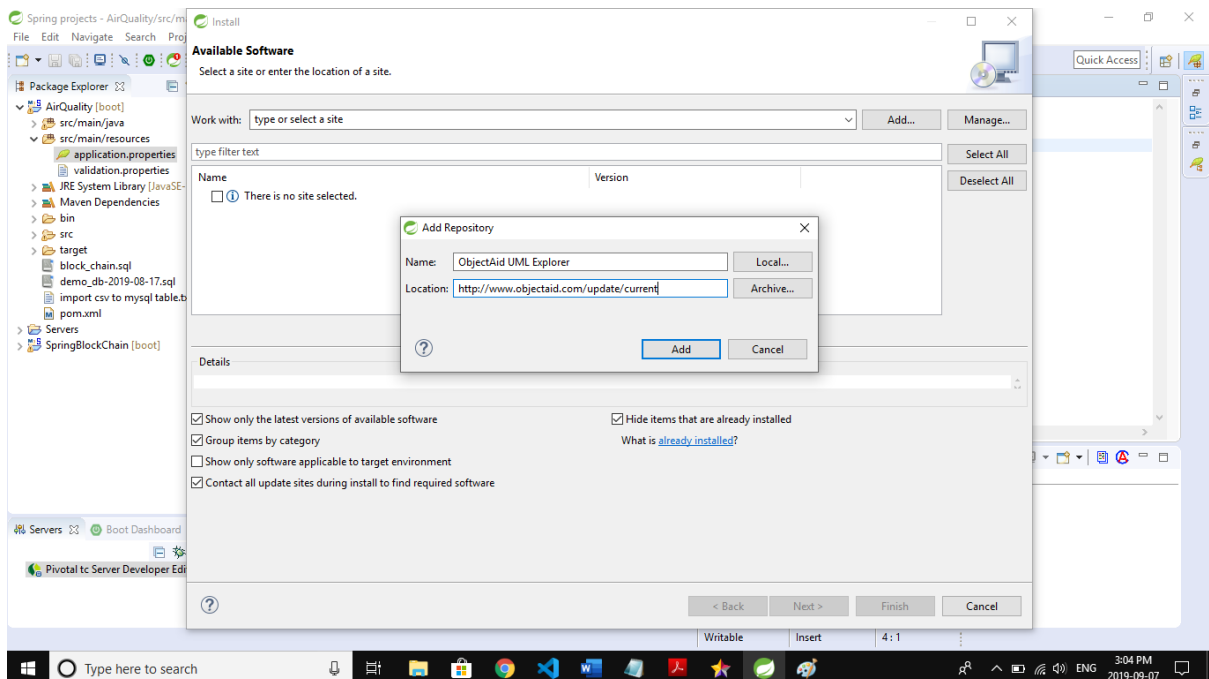
Step 2: The “Install” wizard leads to an “Available Software” page, click on the Add button.



Step 3: In the “Add Repository” dialog box, enter the following,

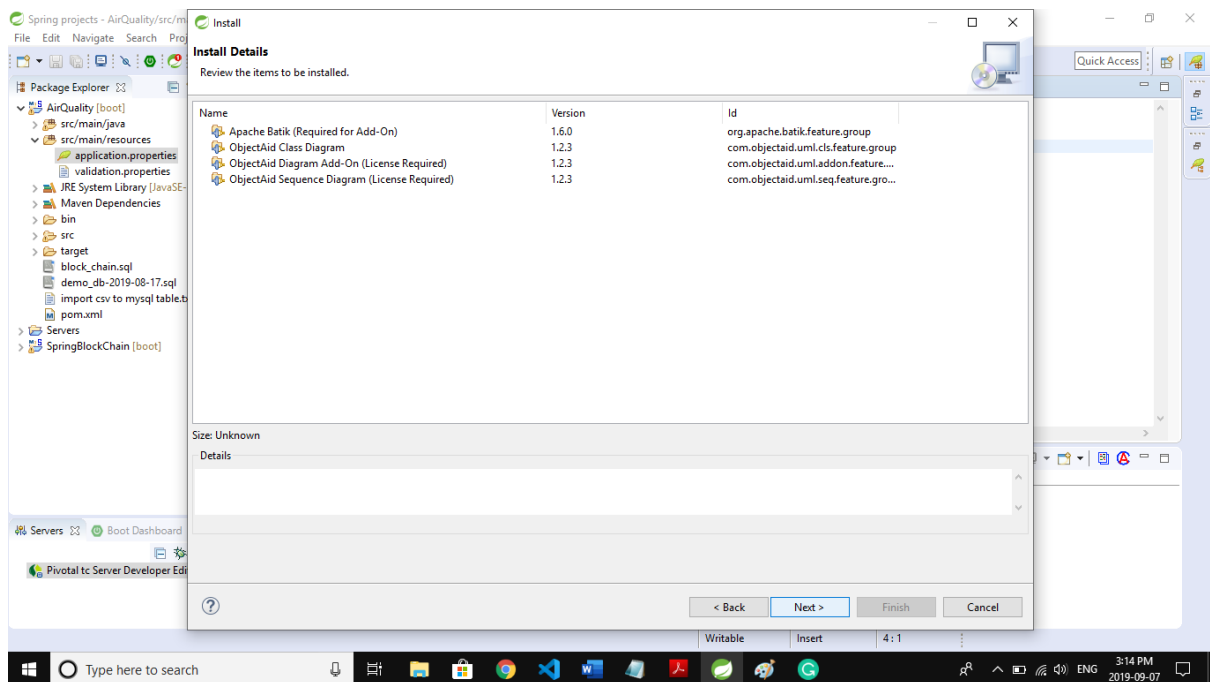
Name: ObjectAid UML Explorer

URL: <http://www.objectaid.com/update/current>

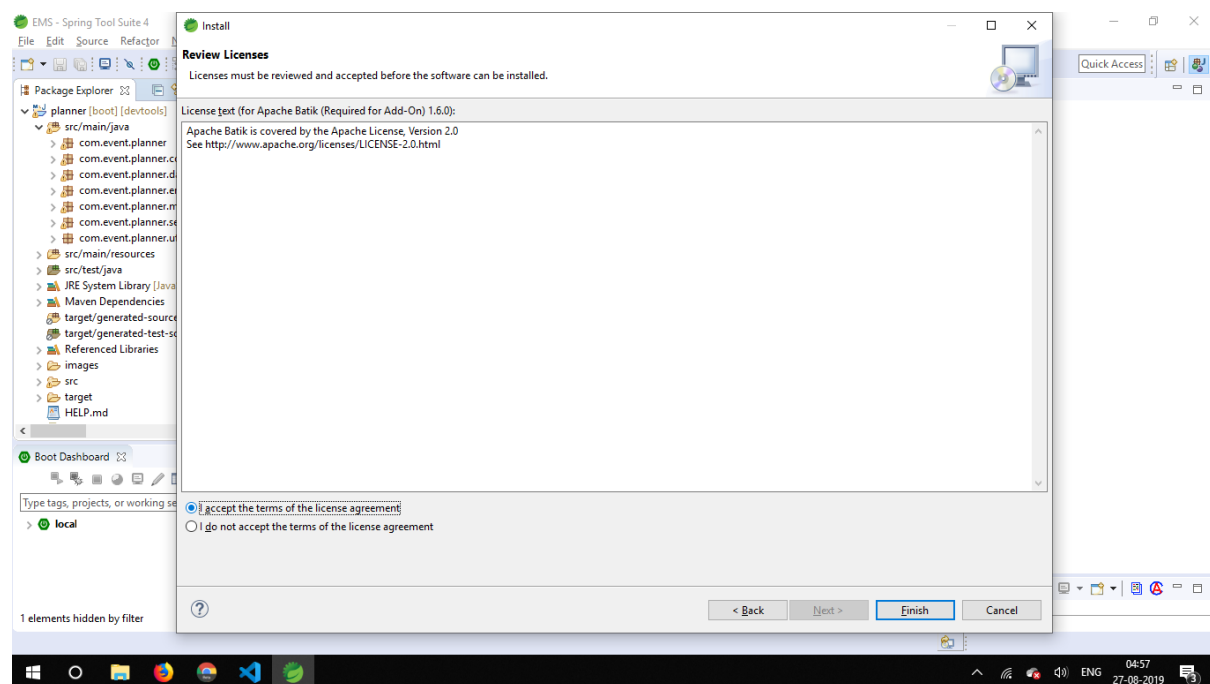


Air Quality

Step 4: The “Install” dialog box will now show the available plugin. Check the “ObjectAid UML Explorer” and hit the next button.



Step 5: On the “Install Details” page hit next, on the “Review licenses” page accept the terms and agreements of the team of ObjectAid and hit Finish to complete the installation.



Generating Class Diagrams

The following steps are involved in creating the class diagrams in Eclipse.

- Create a new class diagram by right clicking on the Package Explorer > New > Other > ObjectAid UML Diagram > ObjectAid Class Diagram
- Drag and drop the .java file from the Package Explorer for which you want to generate the class diagram.
- Right-click on the dragged class > Add > Associated. This adds all the classes associated with that particular class.
- Similarly, we can add all the associated, dependent and generalized classes of the selected class.
- The example is shown below:

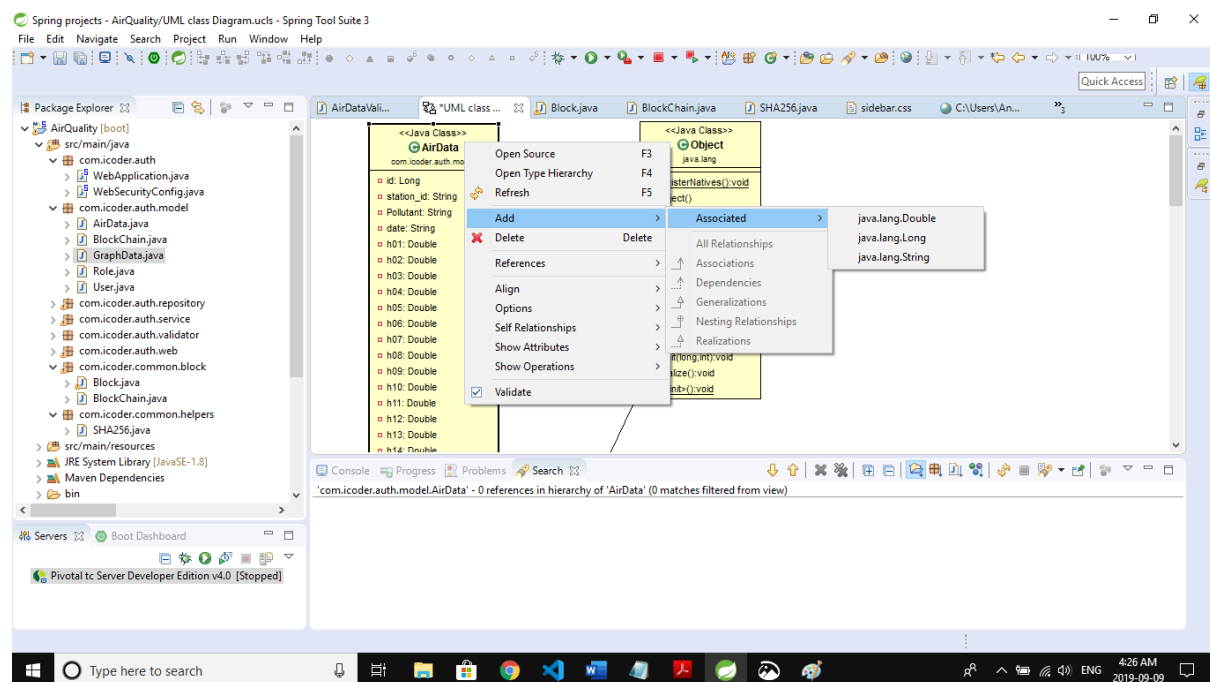


Figure 14: Block and Blockchain Class Diagram

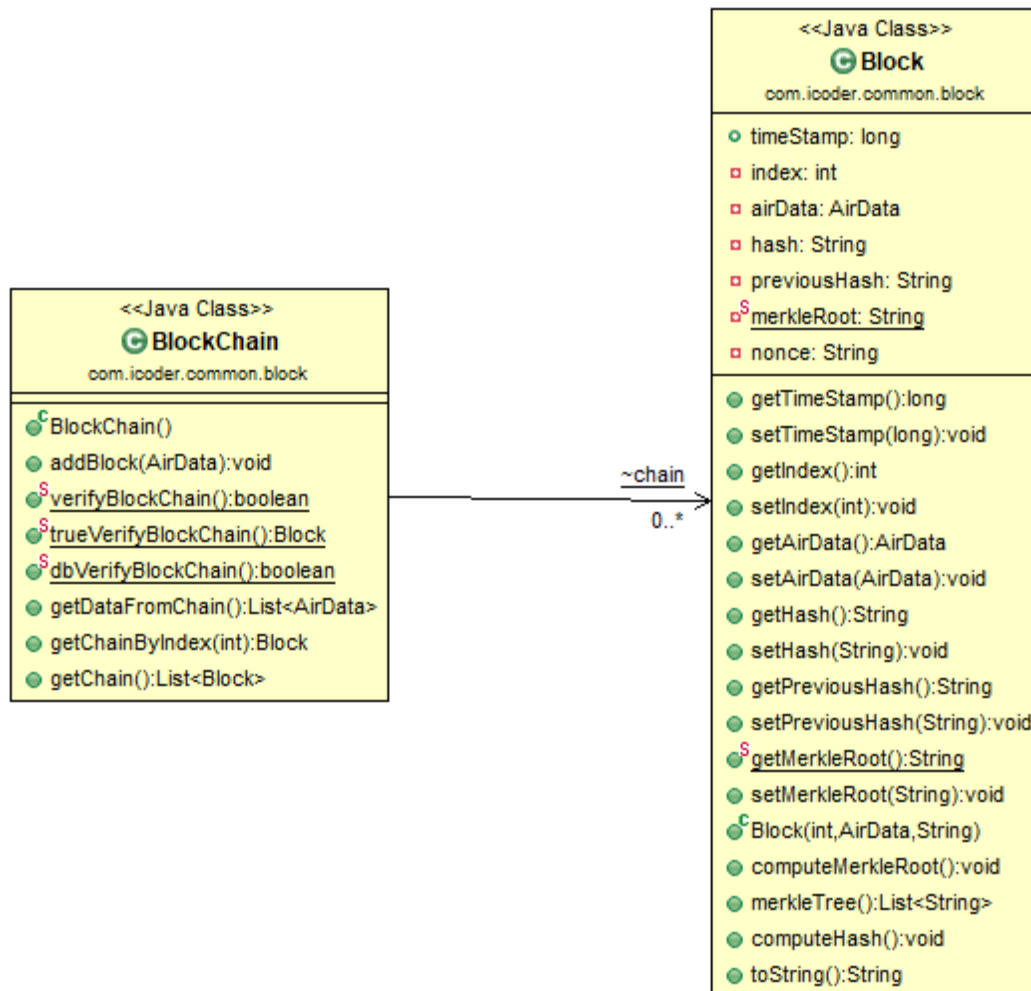


Figure 15: User Validator class diagram

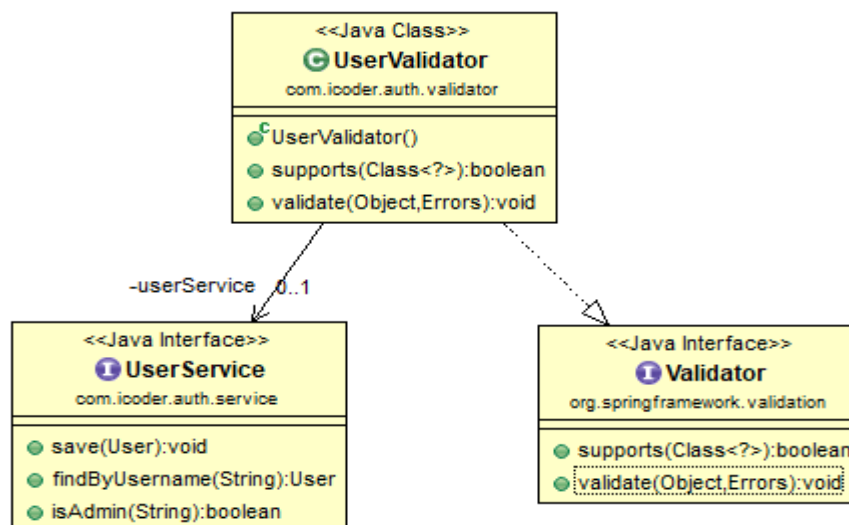
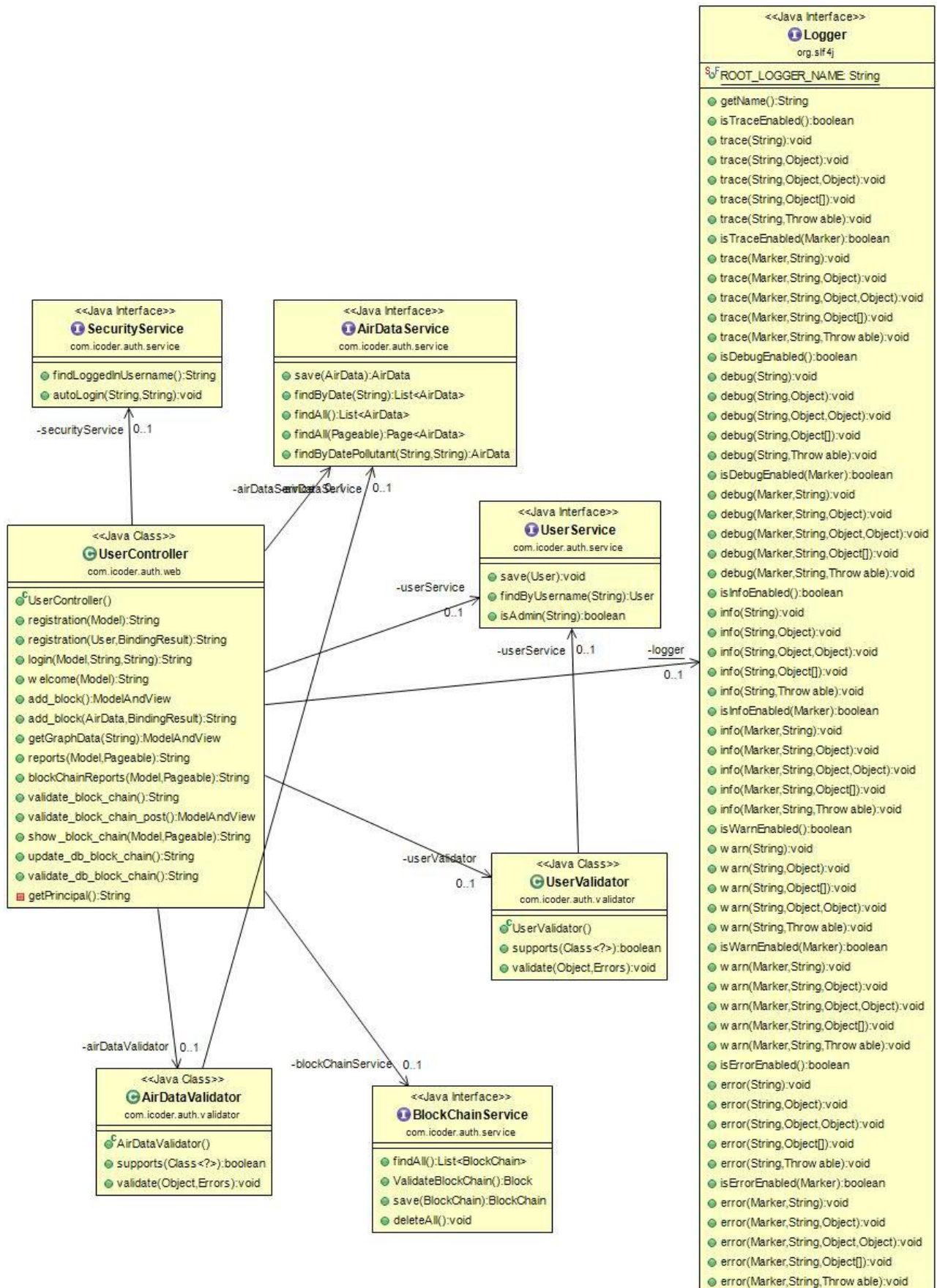


Figure 16: User Controller class diagram:

Sequence Diagram

A Sequence diagram is a behavioral diagram that depicts the interaction between objects in sequential order. In other words, a sequence diagram describes the objects and the classes involved, the sequence of messages exchanged between the objects and the order in which these interactions take place internally.

Figure 17: Admin Sequence Diagram

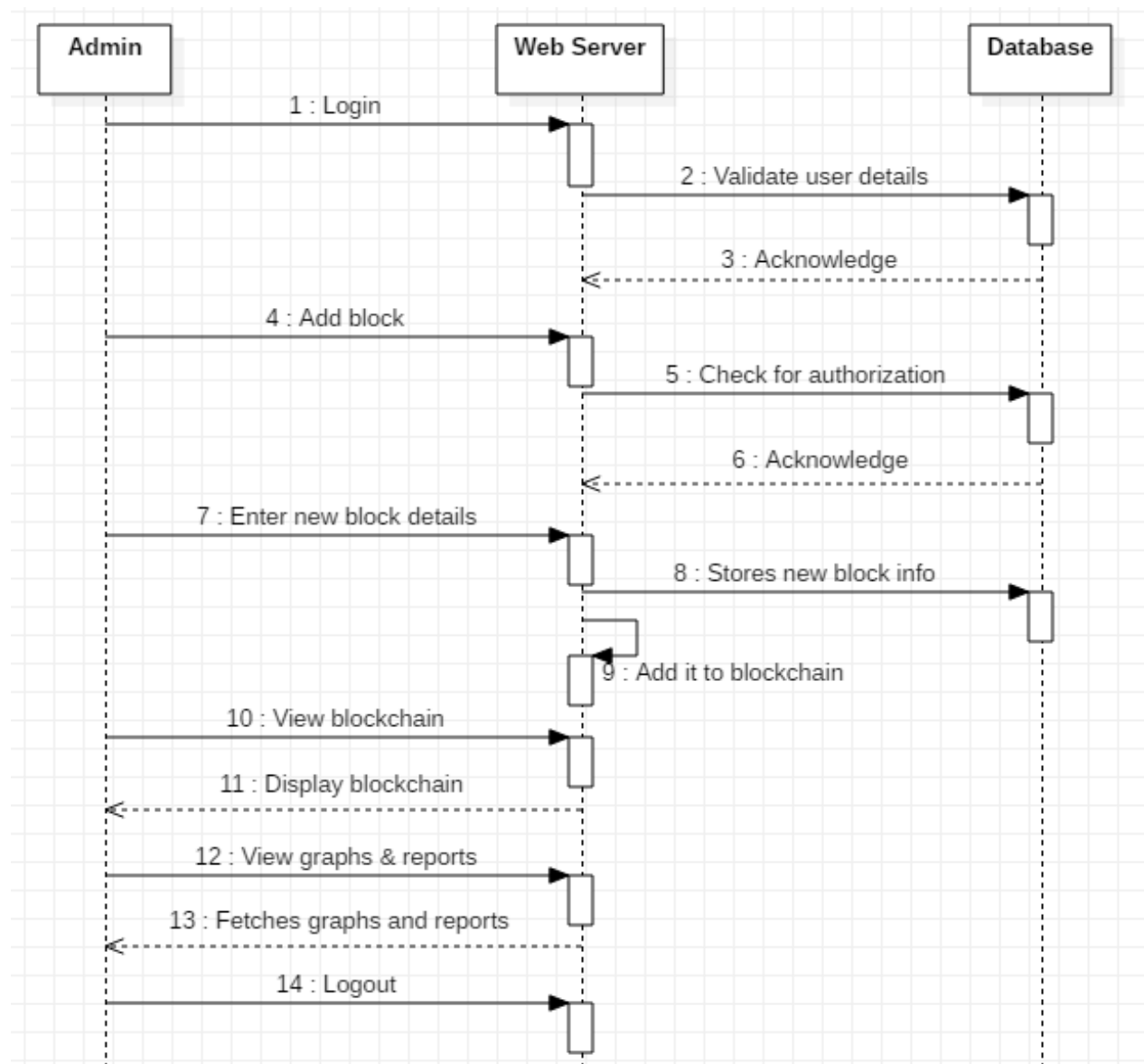
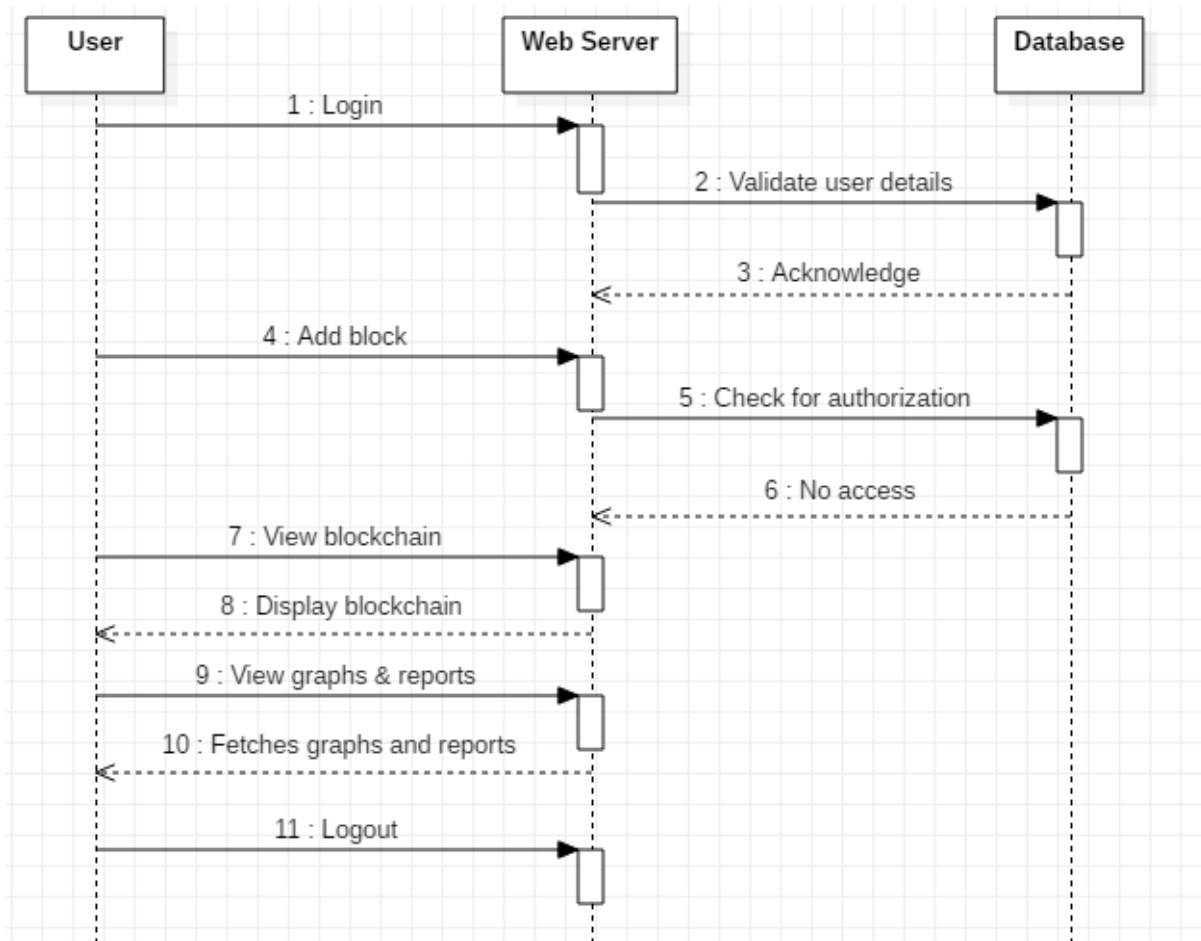


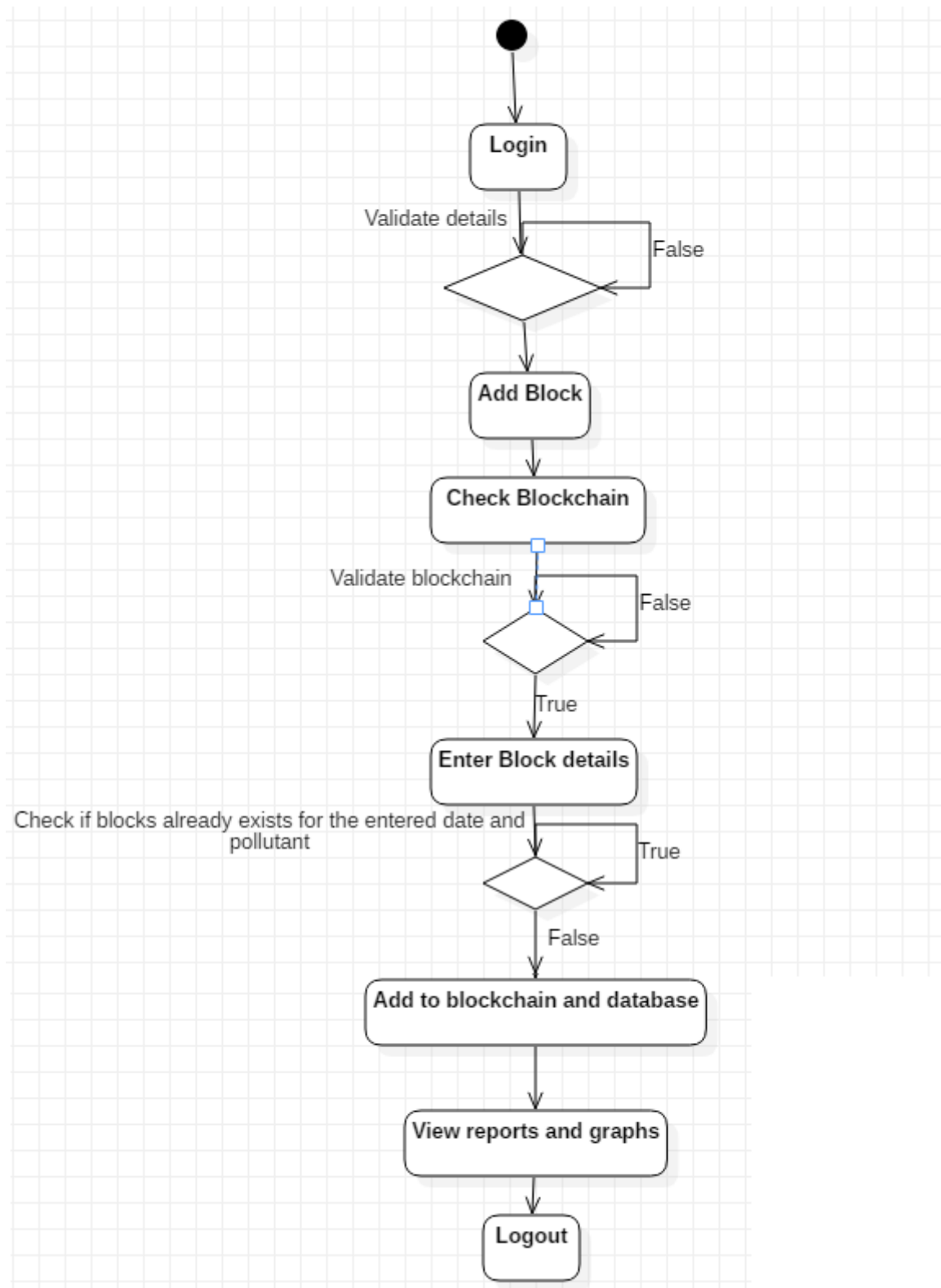
Figure 18: User Sequence Diagram

Activity Diagram

Activity Diagram is a behavioral diagram, and it is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. Different symbols are used to represent the flow from beginning until the end of the activities performed. Each symbol has its own property. Most commonly used symbols are,

- Solid circle > Initial state
- Rounded rectangle > Activity / Action State
- Directed arrow > flow of action
- Rhombus/Diamond > decisions and branching
- Circle over a dot > Final state or the endpoint

Figure 19: User Activity Diagram



5. SYSTEM IMPLEMENTATION

5.1 MODULE DESCRIPTION

The system mainly comprises of two modules.

- a. Admin Module
- b. User Module

Admin(s) are the authorized people who can add blocks to the existing blockchain. They might also have access to database. Every user who signups are considered as a standard public user initially. Later they can be given admin role by modifying in backend database.

The admin or user should signup prior using the application, if not registered, he/she should first signup. The credentials provided during signup are necessary to login. After logging in the user is redirected to the home page which has several tabs. They are:

- Add Block
- Validate Blockchain
- Add Block
- Database report
- Blockchain report
- Show blockchain
- Statistical analysis

Let's discuss their functionalities in brief.

Add block

Add block is page used to add new blocks to the existing blockchain. Only authorized users (Admin) can add blocks. If other users try to access this page, it throws error a message saying, "Sorry!! You are unauthorized to add a block!! Only **Admin** can add blocks!!"

If the user has admin profile role, then he/she can add new blocks. First, they should enter all the block details and click on "Submit". Once submitted,

the system will check if the entered data is appropriate, i.e. it checks if there is an entry in blockchain for same pollutant in the same date. If it is true, it will throw an error saying “This pollutant data already exists in the blockchain for same date!! Sorry you cannot add this block!!”. This check helps in avoiding manipulation or duplication of data.

Once you enter valid values to add a block, in the backend it first validates the blockchain. This is to check if the blockchain is intact and valid, before adding a new block. Then the block is created and added to blockchain.

Validate blockchain

This tab provides a way to validate the blockchain. The backend code compares the hash values and checks the Merkle root hash. If the blockchain is valid, it displays “Verified” and the Merkle root hash value.

Database report

Displays air quality monitoring data directly from the database.

Blockchain report

This tab displays the air quality monitoring data report by retrieving data from the blockchain.

Show blockchain

Shows the actual blockchain values, i.e. Timestamp, hash and previous hash values.

Statistical analysis

This tab shows the date-wise graphical representation of the air quality data. Here I am using pie charts to display the data. This kind of representation helps common people to understand the air quality data easily.

Logout

Used for logging out of the application.

5.2 CODE

Login Page code:

```
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>

<c:set var="contextPath" value="${pageContext.request.contextPath}"/>

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Log in with your account</title>

<link href="${contextPath}/resources/css/bootstrap.min.css"
rel="stylesheet">
<link href="${contextPath}/resources/css/common.css" rel="stylesheet">
</head>

<body>

<div class="container">
<form method="POST" action="${contextPath}/login" class="form-signin">
<h2 class="form-heading">Log in</h2>

<div class="form-group ${error != null ? 'has-error' : ''}">
<span>${message}</span>
<input name="username" type="text" class="form-control"
placeholder="Username"
autofocus="true"/>
<input name="password" type="password" class="form-control"
placeholder="Password"/>
<span>${error}</span>
<input type="hidden" name="${_csrf.parameterName}"
value="${_csrf.token}"/>

<button class="btn btn-lg btn-primary btn-block" type="submit">Log
In</button>
<h4 class="text-center"><a href="${contextPath}/registration">Create an
account</a></h4>
</div>
</form>
</div>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></
script>
<script src="${contextPath}/resources/js/bootstrap.min.js"></script>
</body>
</html>
```

Registration page code:

```

<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>

<c:set var="contextPath" value="${pageContext.request.contextPath}"/>

<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Log in with your account</title>

<link href="${contextPath}/resources/css/bootstrap.min.css"
rel="stylesheet">
<link href="${contextPath}/resources/css/common.css" rel="stylesheet">
</head>

<body>

<div class="container">
<form method="POST" action="${contextPath}/login" class="form-signin">
<h2 class="form-heading">Log in</h2>

<div class="form-group ${error != null ? 'has-error' : ''}">
<span>${message}</span>
<input name="username" type="text" class="form-control"
placeholder="Username"
autofocus="true"/>
<input name="password" type="password" class="form-control"
placeholder="Password"/>
<span>${error}</span>
<input type="hidden" name="${_csrf.parameterName}"
value="${_csrf.token}"/>

<button class="btn btn-lg btn-primary btn-block" type="submit">Log
In</button>
<h4 class="text-center"><a href="${contextPath}/registration">Create an
account</a></h4>
</div>
</form>
</div>

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></
script>
<script src="${contextPath}/resources/js/bootstrap.min.js"></script>
</body>
</html>

```

Usercontroller.java:

```
package com.icoder.auth.web;

import java.util.ArrayList;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.support.PagedListHolder;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.web.PageableDefault;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;

import com.icoder.auth.model.AirData;
import com.icoder.auth.model.GraphData;
import com.icoder.auth.model.GraphData.KeyValue;
import com.icoder.auth.model.User;
import com.icoder.auth.service.AirDataService;
import com.icoder.auth.service.BlockChainService;
import com.icoder.auth.service.SecurityService;
import com.icoder.auth.service.UserService;
import com.icoder.auth.validator.AirDataValidator;
import com.icoder.auth.validator.UserValidator;
import com.icoder.common.block.Block;
import com.icoder.common.block.BlockChain;

@Controller
public class UserController {

    private static final Logger logger =
        LoggerFactory.getLogger(UserController.class);

    @Autowired
    private UserService userService;

    @Autowired
    private SecurityService securityService;

    @Autowired
    private UserValidator userValidator;

    @Autowired
```

```
private AirDataService airDataService;

@Autowired
private AirDataValidator airDataValidator;

@Autowired
private BlockchainService blockchainService;

//    @Autowired
//    private GraphData graphData;

@GetMapping("/registration")
public String registration(Model model) {
    model.addAttribute("userForm", new User());

    return "registration";
}

@PostMapping("/registration")
public String registration(@ModelAttribute("userForm") User userForm,
    BindingResult bindingResult) {
    userValidator.validate(userForm, bindingResult);

    if (bindingResult.hasErrors()) {
        return "registration";
    }

    userService.save(userForm);

    securityService.autoLogin(userForm.getUsername(),
        userForm.getPasswordConfirm());

    return "redirect:/welcome";
}

@GetMapping("/login")
public String login(Model model, String error, String logout) {
    if (error != null)
        model.addAttribute("error", "Your username and password is invalid.");

    if (logout != null)
        model.addAttribute("message", "You have been logged out successfully.");

    return "login";
}

@GetMapping({"", "/welcome"})
public String welcome(Model model) {
    return "welcome";
}

@GetMapping("/add_block")
public ModelAndView add_block() {
    ModelAndView model=new ModelAndView();
    if(userService.isAdmin(getPrincipal())){
```

```

List<String> PollutantList=new ArrayList<>();
PollutantList.add("Ozone");
PollutantList.add("Nitrogen Oxides");
PollutantList.add("Nitrogen Oxide");
PollutantList.add("Nitrogen Dioxide");
PollutantList.add("Fine Particulate Matter");

model.setViewName("add_block");
model.addObject("airDataForm", new AirData());
model.addObject("PollutantList",PollutantList);
}else{
model.setViewName("Unauthorized");
}
return model;
}
@PostMapping("/add_block")
public String add_block(@ModelAttribute("airDataForm") AirData
airDataForm, BindingResult bindingResult) {
if(userService.isAdmin(getPrincipal())){
airDataValidator.validate(airDataForm, bindingResult);

if (bindingResult.hasErrors()) {
return "add_block";
}

AirData airData=airDataService.save(airDataForm);
BlockChain blockChain=new BlockChain();
blockChain.addBlock(airData);

return "redirect:/reports";
}else{
return "Unauthorized";
}
}

@GetMapping(value = "/graph")
public ModelAndView getGraphData(@RequestParam(value = "date", required =
false, defaultValue = "2017-01-01") String date) {
ModelAndView model=new ModelAndView();
AirData airData=new AirData();
airData.setDate(date);
model.setViewName("graph");
model.addObject("airDataForm",airData);
System.out.print(date);
GraphData.PieData(date);
List<KeyValue> pieDataList = GraphData.getPiechartData();
System.out.print(pieDataList);
model.addObject("pieDataList", pieDataList);
return model;
}
@GetMapping("/reports")
public String reports(Model model, @PageableDefault(value=50, page=0)
Pageable pageable) {

```



```

Page<AirData> pages = airDataService.findAll(pageable);
model.addAttribute("number", pages.getNumber());
model.addAttribute("totalPages", pages.getTotalPages());
model.addAttribute("totalElements", pages.getTotalElements());
model.addAttribute("size", pages.getSize());
model.addAttribute("airDataList", pages.getContent());
return "reports";
}

@GetMapping("/blockChainreports")
public String blockChainReports(Model model, @PageableDefault(value=50,
page=0) Pageable pageable) {

    BlockChain blockChain=new BlockChain();
    List<AirData> airDataList=blockChain.getDataFromChain();

    PagedListHolder<AirData> page = new PagedListHolder<>(airDataList);
    page.setPageSize(pageable.getPageSize()); // number of items per page
    page.setPage(pageable.getPageNumber());    // set to first page

    model.addAttribute("totalPages",page.getPageCount());
    model.addAttribute("size", page.getPageSize());
    model.addAttribute("airDataList",page.getPageList());

    return "blockChainreports";
}

@GetMapping("/validate_block_chain")
public String validate_block_chain() {
    return "validate_block_chain";
}

@PostMapping("/validate_block_chain")
public ModelAndView validate_block_chain_post() {
    BlockChain blockChain=new BlockChain();
    boolean verify=BlockChain.verifyBlockChain();
    System.out.println("Verified results:- ");
    System.out.println(verify);

    ModelAndView model=new ModelAndView();
    model.setViewName("validate_block_chain");
    model.addObject("verify",verify);
    model.addObject("MerkleRoot",Block.getMerkleRoot());
    return model;
}

@GetMapping("/show_block_chain")
public String show_block_chain(Model model, @PageableDefault(value=50,
page=0) Pageable pageable) {
    BlockChain blockChain=new BlockChain();
    List<Block> blockChainList=blockChain.getChain();
    blockChainList.forEach(val->{
        System.out.println(val.toString());
    });
}

```

```

PagedListHolder<Block> page = new PagedListHolder<>(blockChainList);
page.setPageSize(pageable.getPageSize()); // number of items per page
page.setPage(pageable.getPageNumber()); // set to first page

model.addAttribute("totalPages",page.getPageCount());
model.addAttribute("size", page.getPageSize());
model.addAttribute("blockChainList",blockChainList);
return "show_block_chain";
}

@GetMapping("/update_db_block_chain")
public String update_db_block_chain() {
    BlockChain blockChain=new BlockChain();
    List<Block> blockChainList=blockChain.getChain();
    blockChainService.deleteAll();
    blockChainList.forEach(val->{
        System.out.println(val.toString());
        com.icoder.auth.model.BlockChain blockChainModel=new
        com.icoder.auth.model.BlockChain();
        blockChainModel.setIndex(Long.valueOf(val.getIndex()));
        blockChainModel.setTime_stamp((Long.valueOf(val.getTimeStamp())));
        blockChainModel.setHash(val.getHash());
        blockChainModel.setPrevious_hash(val.getPreviousHash());
        blockChainService.save(blockChainModel);
    });
    return "show_block_chain";
}

@GetMapping("/validate_db_block_chain")
public String validate_db_block_chain() {
    Block block=blockChainService.ValidateBlockChain();

    logger.info(String.valueOf(block.getIndex()));

    return "show_block_chain";
}

private String getPrincipal(){
    String userName = null;
    Object principal =
    SecurityContextHolder.getContext().getAuthentication().getPrincipal();

    if (principal instanceof UserDetails) {
        userName = ((UserDetails)principal).getUsername();
    } else {
        userName = principal.toString();
    }
    return userName;
}
}

```

Add block page code:

```

<div class="container">

<jsp:include page="${contextPath}/common/sider.jsp"/>

<div class="main">
<input type="hidden" name="${_csrf.parameterName}"
value="${_csrf.token}"/>
<form:form method="POST" modelAttribute="airDataForm" class="form-signin">
<h2 class="form-signin-heading">Add Block</h2>

<div class="row">
<div class="col-sm-4">
<spring:bind path="date">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="date">Date</label>
<form:input type="date" path="date" class="form-control"
placeholder="Date"
autofocus="true"></form:input>
<form:errors path="date"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-4">
<spring:bind path="station_id">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="station_id">Station Id</label>
<form:input type="text" path="station_id" class="form-control"
placeholder="Station Id"
autofocus="true"></form:input>
<form:errors path="station_id"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-4">
<spring:bind path="pollutant">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="pollutant">pollutant</label>
<form:select path="pollutant" items="${PollutantList}" class="form-
control"/>
<form:errors path="pollutant"></form:errors>
</div>
</spring:bind>
</div>
</div>
<div class="row">
<div class="col-sm-1">
<spring:bind path="h01">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h01">H01</label>
<form:input type="text" path="h01" class="form-control" placeholder="H01"
autofocus="true"></form:input>
<form:errors path="h01"></form:errors>

```

```

</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h02">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h02">H02</label>
<form:input type="text" path="h02" class="form-control" placeholder="H02"
autofocus="true"></form:input>
<form:errors path="h02"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h03">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h03">H03</label>
<form:input type="text" path="h03" class="form-control" placeholder="H03"
autofocus="true"></form:input>
<form:errors path="h03"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h04">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h04">H04</label>
<form:input type="text" path="h04" class="form-control" placeholder="H04"
autofocus="true"></form:input>
<form:errors path="h04"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h05">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h03">H05</label>
<form:input type="text" path="h05" class="form-control" placeholder="H05"
autofocus="true"></form:input>
<form:errors path="h05"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h06">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h06">H06</label>
<form:input type="text" path="h06" class="form-control" placeholder="H06"
autofocus="true"></form:input>
<form:errors path="h06"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">

```

```

<spring:bind path="h07">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h07">H07</label>
<form:input type="text" path="h07" class="form-control" placeholder="H07"
autofocus="true"></form:input>
<form:errors path="h07"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h08">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h08">H08</label>
<form:input type="text" path="h08" class="form-control" placeholder="H08"
autofocus="true"></form:input>
<form:errors path="h08"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h09">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h09">H09</label>
<form:input type="text" path="h09" class="form-control" placeholder="H09"
autofocus="true"></form:input>
<form:errors path="h09"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h10">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h10">H10</label>
<form:input type="text" path="h10" class="form-control" placeholder="H10"
autofocus="true"></form:input>
<form:errors path="h10"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h11">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h11">H11</label>
<form:input type="text" path="h11" class="form-control" placeholder="H11"
autofocus="true"></form:input>
<form:errors path="h11"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h12">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h12">H12</label>
<form:input type="text" path="h12" class="form-control" placeholder="H12"

```

```

autofocus="true"></form:input>
<form:errors path="h12"></form:errors>
</div>
</spring:bind>
</div>
</div>
<div class="row">
<div class="col-sm-1">
<spring:bind path="h13">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h13">H13</label>
<form:input type="text" path="h13" class="form-control" placeholder="H13"
autofocus="true"></form:input>
<form:errors path="h13"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h14">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h14">H14</label>
<form:input type="text" path="h14" class="form-control" placeholder="H14"
autofocus="true"></form:input>
<form:errors path="h14"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h15">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h15">H15</label>
<form:input type="text" path="h15" class="form-control" placeholder="H15"
autofocus="true"></form:input>
<form:errors path="h15"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h16">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h16">H16</label>
<form:input type="text" path="h16" class="form-control" placeholder="H16"
autofocus="true"></form:input>
<form:errors path="h16"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h17">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h17">H17</label>
<form:input type="text" path="h17" class="form-control" placeholder="H17"
autofocus="true"></form:input>
<form:errors path="h17"></form:errors>

```

```

</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h18">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h18">H18</label>
<form:input type="text" path="h18" class="form-control" placeholder="H18"
autofocus="true"></form:input>
<form:errors path="h18"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h19">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h19">H19</label>
<form:input type="text" path="h19" class="form-control" placeholder="H19"
autofocus="true"></form:input>
<form:errors path="h19"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h20">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h20">H20</label>
<form:input type="text" path="h20" class="form-control" placeholder="H20"
autofocus="true"></form:input>
<form:errors path="h20"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h21">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h21">H21</label>
<form:input type="text" path="h21" class="form-control" placeholder="H21"
autofocus="true"></form:input>
<form:errors path="h21"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h22">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h22">H22</label>
<form:input type="text" path="h22" class="form-control" placeholder="H22"
autofocus="true"></form:input>
<form:errors path="h22"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">

```

```

<spring:bind path="h23">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h23">H23</label>
<form:input type="text" path="h23" class="form-control" placeholder="H23"
autofocus="true"></form:input>
<form:errors path="h23"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1">
<spring:bind path="h24">
<div class="form-group ${status.error ? 'has-error' : ''}">
<label for="h24">H24</label>
<form:input type="text" path="h24" class="form-control" placeholder="H24"
autofocus="true"></form:input>
<form:errors path="h24"></form:errors>
</div>
</spring:bind>
</div>
</div>

<div class="row">
<div class="col-sm-12"><button class="btn btn-primary center-block"
type="submit">Submit</button></div>
</div>
</form:form>
</div>

</div>
<jsp:include page="${contextPath}/common/footer.jsp"/>

```

Statistical analysis page code:

```

<script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
<jsp:include page="${contextPath}/common/header.jsp"/>
<c:set var="contextPath" value="${pageContext.request.contextPath}"/>
<div class="container">

<jsp:include page="${contextPath}/common/sider.jsp"/>

<div class="main">
<input type="hidden" name="${_csrf.parameterName}"
value="${_csrf.token}"/>

<div class="container">
<input type="hidden" name="${_csrf.parameterName}"
value="${_csrf.token}"/>
<form:form method="GET" modelAttribute="airDataForm" class="form-signin">
<br/><br/>
<div class="row">
<div class="col-sm-2">
<label for="date">Date</label>

```



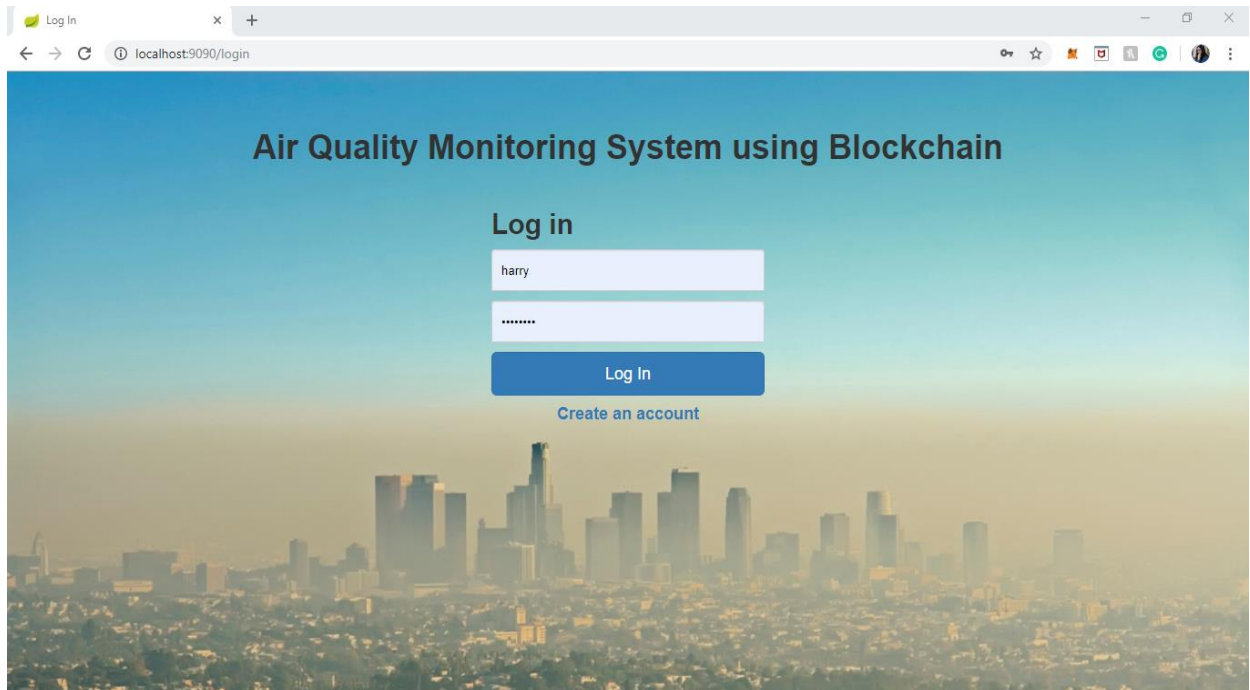
```

</div>
<div class="col-sm-4">
<spring:bind path="date">
<div class="form-group ${status.error ? 'has-error' : ''}">
<form:input type="date" path="date" class="form-control"
placeholder="Date"
autofocus="true"></form:input>
<form:errors path="date"></form:errors>
</div>
</spring:bind>
</div>
<div class="col-sm-1"><button class="btn btn-primary center-block"
type="submit">Submit</button></div>
</div>
</form:form>
<div id="chart_wrap">
<div id="piechart"></div>
</div>
</div>
</div>
</div>
<jsp:include page="${contextPath}/common/footer.jsp"/>
<script type="text/javascript"
src="https://www.google.com/jsapi"></script>
<script type="text/javascript">
// Load the Visualization API and the piechart package.
google.load('visualization', '1.0', {
'packages' : [ 'corechart' ]
});
// Set a callback to run when the Google Visualization API is loaded.
google.setOnLoadCallback(drawChart);
// Callback that creates and populates a data table,
// instantiates the pie chart, passes in the data and
// draws it.
function drawChart() {
// Create the data table.
var data = google.visualization.arrayToDataTable([
['Pollutant', 'Hour'],
<c:forEach items="${pieDataList}" var="entry">
[ '${entry.key}', ${entry.value} ],
</c:forEach>
]);
var options = {
'title' : 'Pollutant Data',
is3D : true,
pieSliceText: 'value-and-percentage',
tooltip : {showColorCode: true},
'width' : 900,
'height' : 500};
// Instantiate and draw our chart, passing in some options.
var chart = new
google.visualization.PieChart(document.getElementById('piechart'));
chart.draw(data, options);
</script>

```

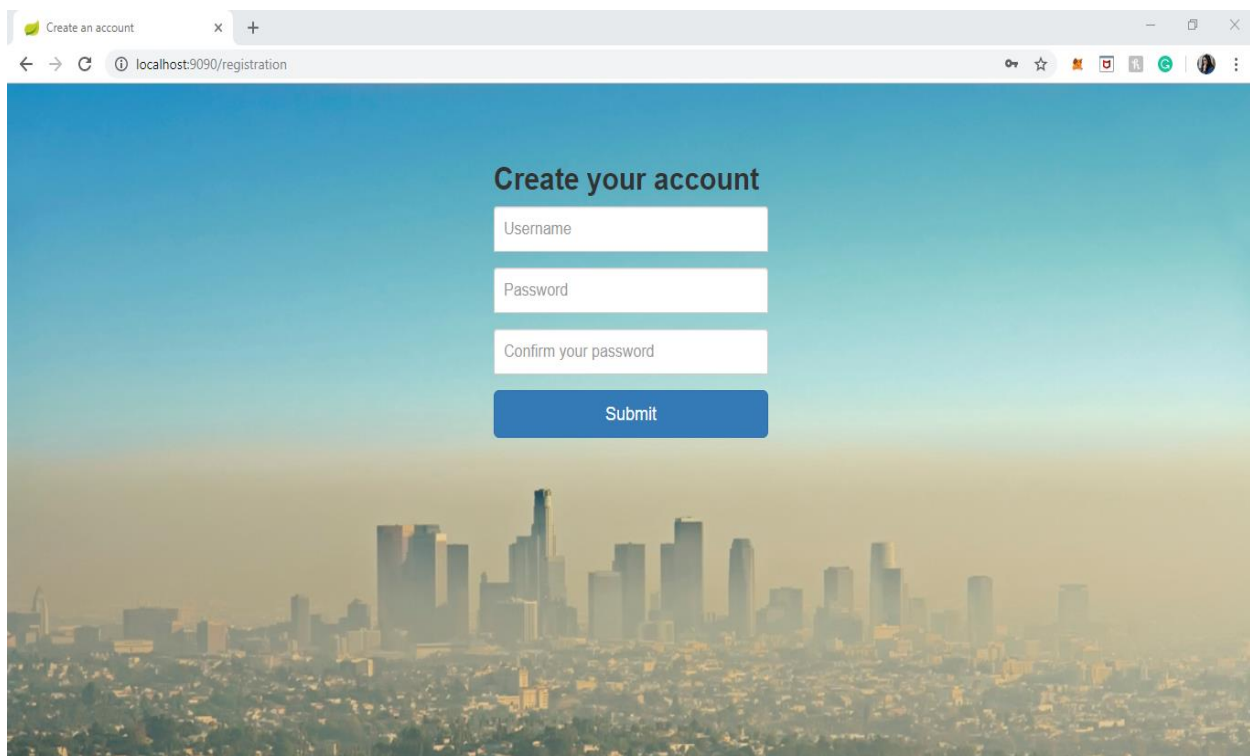
6. OUTPUT SCREENS

Home page/Login page:



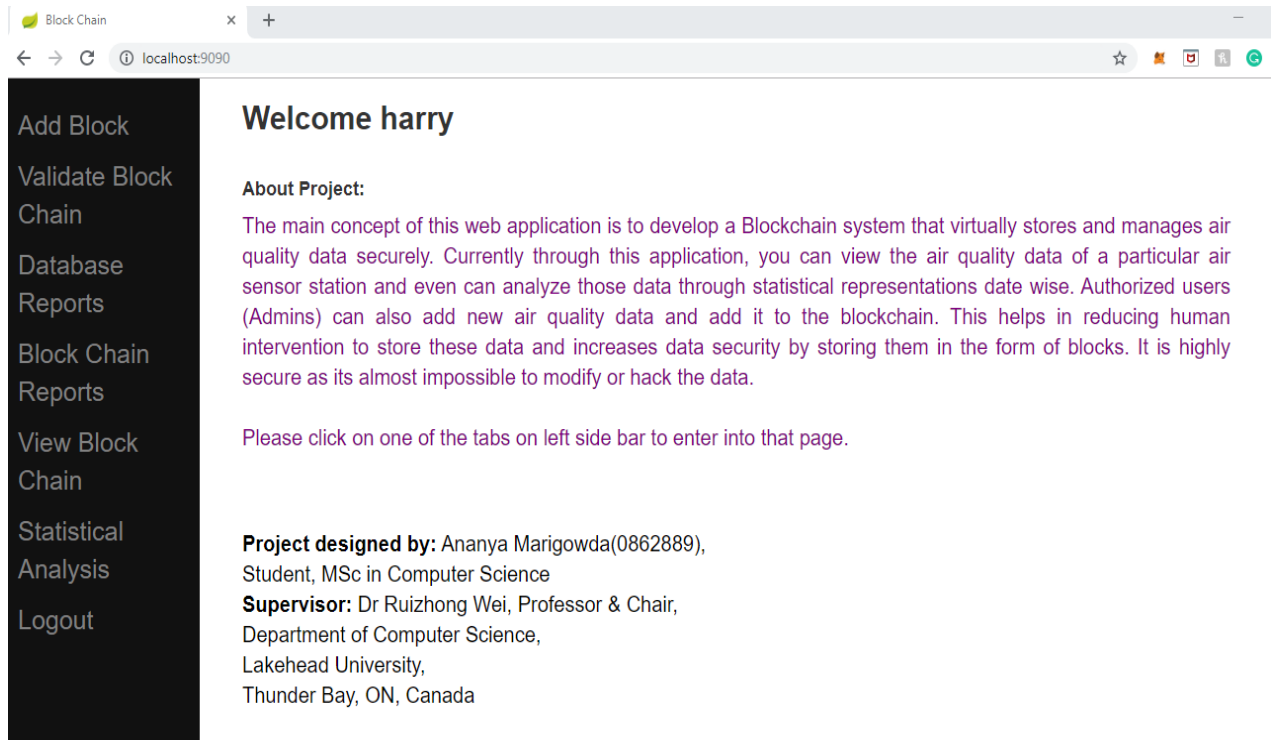
A screenshot of a web browser displaying the login page of an "Air Quality Monitoring System using Blockchain". The browser's address bar shows "localhost:9090/login". The page features a background image of a city skyline under a hazy sky. The main heading is "Air Quality Monitoring System using Blockchain". Below it, the "Log in" section includes a text input field containing "harry", a password input field with masked characters, a blue "Log In" button, and a link that says "Create an account".

Registration Page:

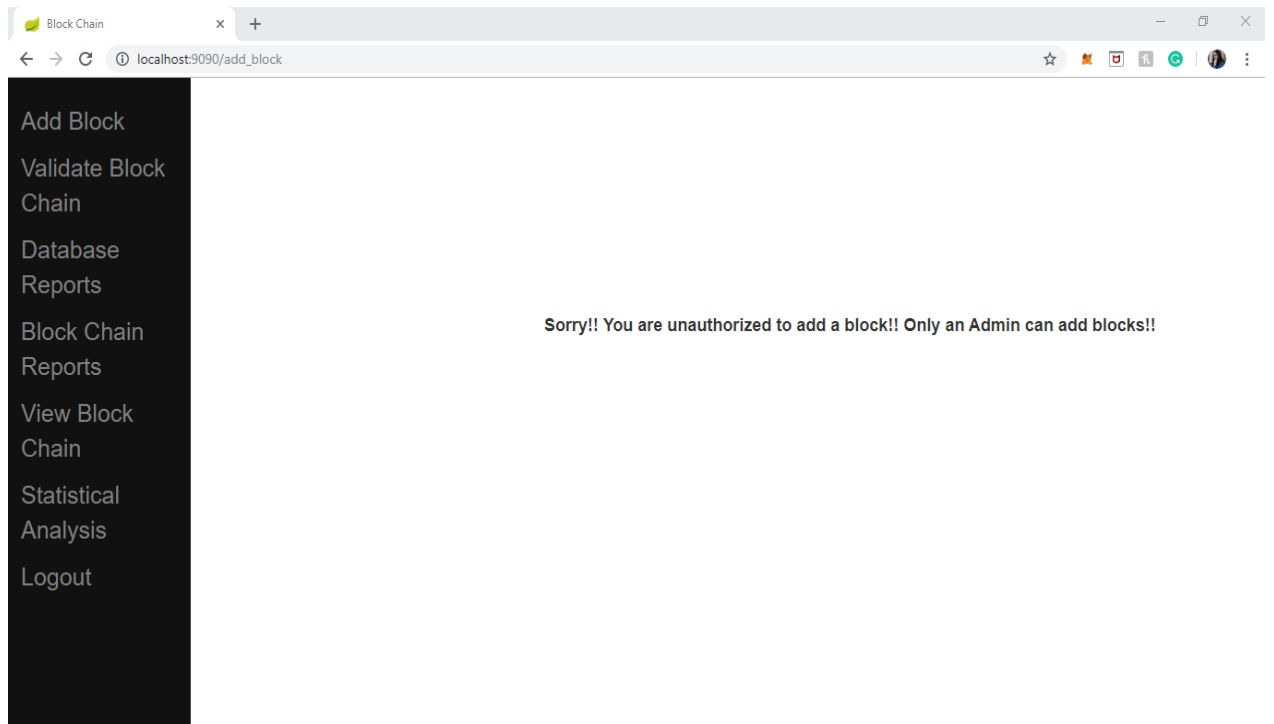


A screenshot of a web browser displaying the registration page of the "Air Quality Monitoring System using Blockchain". The browser's address bar shows "localhost:9090/registration". The page features the same background image of a city skyline. The main heading is "Create your account". Below it, the registration form includes three text input fields labeled "Username", "Password", and "Confirm your password", followed by a blue "Submit" button.

Welcome page:



Add block page for normal user:



Add block page for an Admin:

Add Block

Date:

Station Id:

Pollutant:

H01	H02	H03	H04	H05	H06	H07	H08	H09	H10	H11	H12
<input type="text" value="H01"/>	<input type="text" value="H02"/>	<input type="text" value="H03"/>	<input type="text" value="H04"/>	<input type="text" value="H05"/>	<input type="text" value="H06"/>	<input type="text" value="H07"/>	<input type="text" value="H08"/>	<input type="text" value="H09"/>	<input type="text" value="H10"/>	<input type="text" value="H11"/>	<input type="text" value="H12"/>
H13	H14	H15	H16	H17	H18	H19	H20	H21	H22	H23	H24
<input type="text" value="H13"/>	<input type="text" value="H14"/>	<input type="text" value="H15"/>	<input type="text" value="H16"/>	<input type="text" value="H17"/>	<input type="text" value="H18"/>	<input type="text" value="H19"/>	<input type="text" value="H20"/>	<input type="text" value="H21"/>	<input type="text" value="H22"/>	<input type="text" value="H23"/>	<input type="text" value="H24"/>

When an Admin tries to add block for already existing data in same date:

Add Block

Date:

Station Id:

Pollutant:

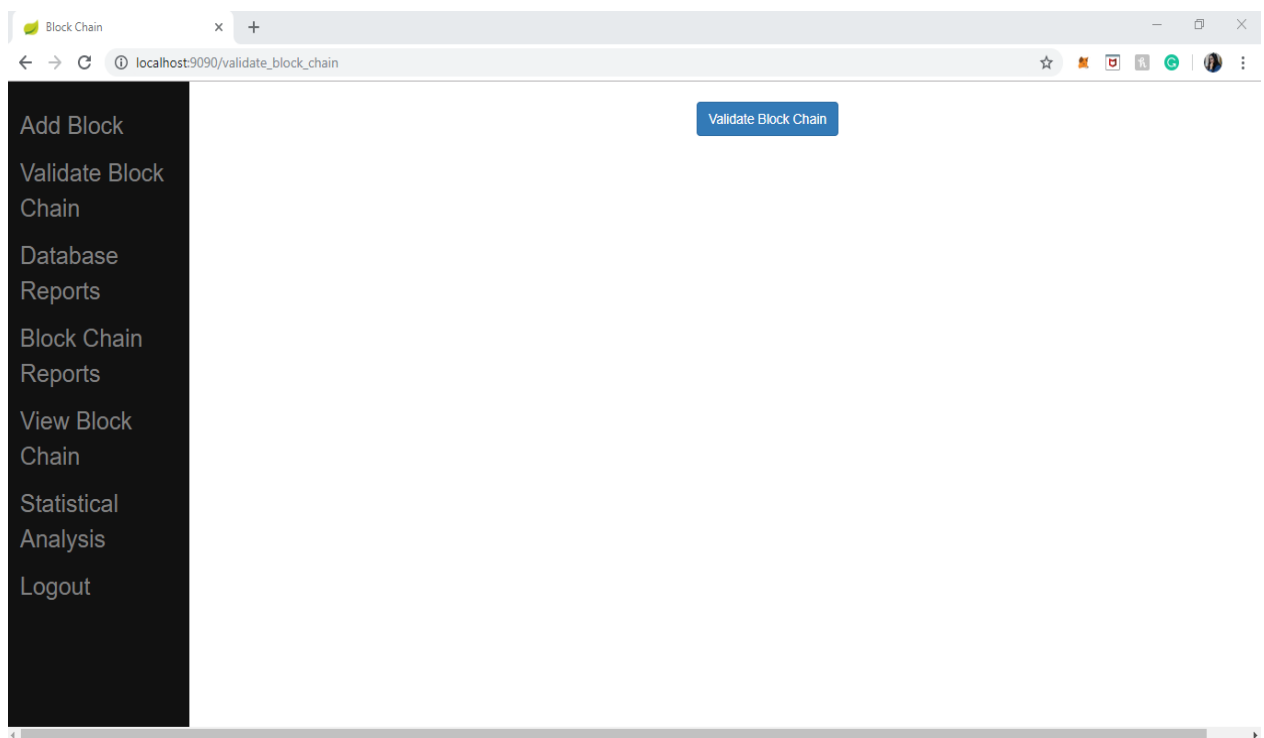
"This pollutant data already exists in the database for the same date. You can add the data only once."

H01	H02	H03	H04	H05	H06	H07	H08	H09	H10	H11	H12
<input type="text" value="8.0"/>	<input type="text" value="5.0"/>	<input type="text" value="4.0"/>	<input type="text" value="5.0"/>	<input type="text" value="6.0"/>	<input type="text" value="4.0"/>	<input type="text" value="6.0"/>	<input type="text" value="4.0"/>	<input type="text" value="6.0"/>	<input type="text" value="7.0"/>	<input type="text" value="4.0"/>	<input type="text" value="6.0"/>
H13	H14	H15	H16	H17	H18	H19	H20	H21	H22	H23	H24
<input type="text" value="4.0"/>	<input type="text" value="5.0"/>	<input type="text" value="6.0"/>	<input type="text" value="H16"/>	<input type="text" value="5.0"/>	<input type="text" value="5.0"/>	<input type="text" value="5.0"/>	<input type="text" value="5.0"/>	<input type="text" value="54.0"/>	<input type="text" value="H22"/>	<input type="text" value="4.0"/>	<input type="text" value="4.0"/>

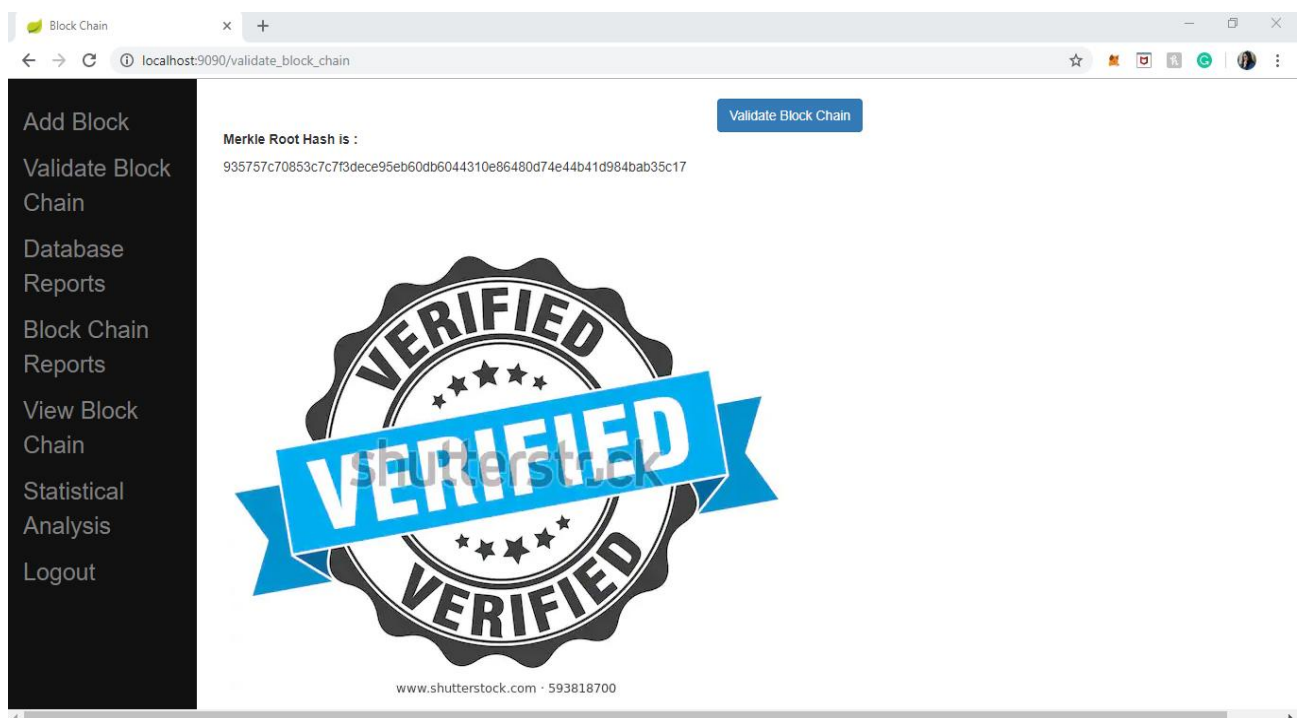
This field is required.

This field is required.

Validate page:



Validation of blockchain after clicking on “Validate Block Chain”



Database Reports page:

	Id	Station Id	Pollutant	Date	H01	H02	H03	H04	H05	H06	H07	H08	H09	H10	H11	H12	H13	H14	H15	H16	H17	H18	H19
1	63203	Nitrogen Oxide	2017-01-01	1.0	1.0	3.0	9999.0	1.0	1.0	1.0	1.0	0.0	1.0	2.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2	63203	Nitrogen Dioxide	2017-01-01	12.0	12.0	14.0	9999.0	10.0	14.0	13.0	14.0	9.0	9.0	7.0	3.0	2.0	2.0	3.0	3.0	3.0	3.0	3.0	6.0
3	63203	Nitrogen Oxides	2017-01-01	13.0	13.0	17.0	9999.0	11.0	15.0	14.0	15.0	10.0	10.0	9.0	4.0	3.0	3.0	3.0	3.0	4.0	4.0	4.0	7.0
4	63203	Ozone	2017-01-01	16.0	16.0	15.0	9999.0	21.0	17.0	16.0	15.0	21.0	23.0	24.0	30.0	32.0	32.0	32.0	32.0	31.0	30.0	26.0	
5	63203	Fine Particulate Matter	2017-01-01	8.0	13.0	12.0	8.0	8.0	8.0	7.0	8.0	7.0	6.0	7.0	5.0	5.0	4.0	4.0	4.0	4.0	4.0	5.0	
6	63203	Nitrogen Oxide	2017-01-02	1.0	1.0	1.0	2.0	1.0	4.0	1.0	1.0	1.0	3.0	7.0	11.0	16.0	16.0	14.0	23.0	35.0	41.0	30.0	
7	63203	Nitrogen Dioxide	2017-01-02	12.0	15.0	17.0	16.0	13.0	13.0	12.0	7.0	10.0	13.0	15.0	16.0	8.0	9.0	10.0	13.0	16.0	27.0	16.0	
8	63203	Nitrogen Oxides	2017-01-02	14.0	16.0	18.0	18.0	15.0	17.0	13.0	8.0	11.0	16.0	22.0	27.0	24.0	25.0	24.0	36.0	52.0	68.0	46.0	
9	63203	Ozone	2017-01-02	14.0	11.0	10.0	9.0	13.0	12.0	14.0	17.0	16.0	12.0	13.0	17.0	29.0	29.0	29.0	28.0	26.0	20.0	22.0	
10	63203	Fine Particulate	2017-01-02	8.0	8.0	8.0	10.0	8.0	10.0	9.0	7.0	7.0	7.0	7.0	7.0	3.0	3.0	3.0	3.0	3.0	6.0	4.0	

Blockchain Reports page:

	Id	Station Id	Pollutant	Date	H01	H02	H03	H04	H05	H06	H07	H08	H09	H10	H11	H12	H13	H14	H15	H16	H17	H18	H19
1	63203	Nitrogen Oxide	2017-01-01	1.0	1.0	3.0	9999.0	1.0	1.0	1.0	1.0	0.0	1.0	2.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2	63203	Nitrogen Dioxide	2017-01-01	12.0	12.0	14.0	9999.0	10.0	14.0	13.0	14.0	9.0	9.0	7.0	3.0	2.0	2.0	3.0	3.0	3.0	3.0	3.0	6.0
3	63203	Nitrogen Oxides	2017-01-01	13.0	13.0	17.0	9999.0	11.0	15.0	14.0	15.0	10.0	10.0	9.0	4.0	3.0	3.0	3.0	3.0	4.0	4.0	4.0	7.0
4	63203	Ozone	2017-01-01	16.0	16.0	15.0	9999.0	21.0	17.0	16.0	15.0	21.0	23.0	24.0	30.0	32.0	32.0	32.0	32.0	31.0	30.0	26.0	
5	63203	Fine Particulate Matter	2017-01-01	8.0	13.0	12.0	8.0	8.0	8.0	7.0	8.0	7.0	6.0	7.0	5.0	5.0	4.0	4.0	4.0	4.0	4.0	5.0	
6	63203	Nitrogen Oxide	2017-01-02	1.0	1.0	1.0	2.0	1.0	4.0	1.0	1.0	1.0	3.0	7.0	11.0	16.0	16.0	14.0	23.0	35.0	41.0	30.0	
7	63203	Nitrogen Dioxide	2017-01-02	12.0	15.0	17.0	16.0	13.0	13.0	12.0	7.0	10.0	13.0	15.0	16.0	8.0	9.0	10.0	13.0	16.0	27.0	16.0	
8	63203	Nitrogen Oxides	2017-01-02	14.0	16.0	18.0	18.0	15.0	17.0	13.0	8.0	11.0	16.0	22.0	27.0	24.0	25.0	24.0	36.0	52.0	68.0	46.0	
9	63203	Ozone	2017-01-02	14.0	11.0	10.0	9.0	13.0	12.0	14.0	17.0	16.0	12.0	13.0	17.0	29.0	29.0	29.0	28.0	26.0	20.0	22.0	
10	63203	Fine Particulate	2017-01-02	8.0	8.0	8.0	10.0	8.0	10.0	9.0	7.0	7.0	7.0	7.0	7.0	3.0	3.0	3.0	3.0	3.0	6.0	4.0	

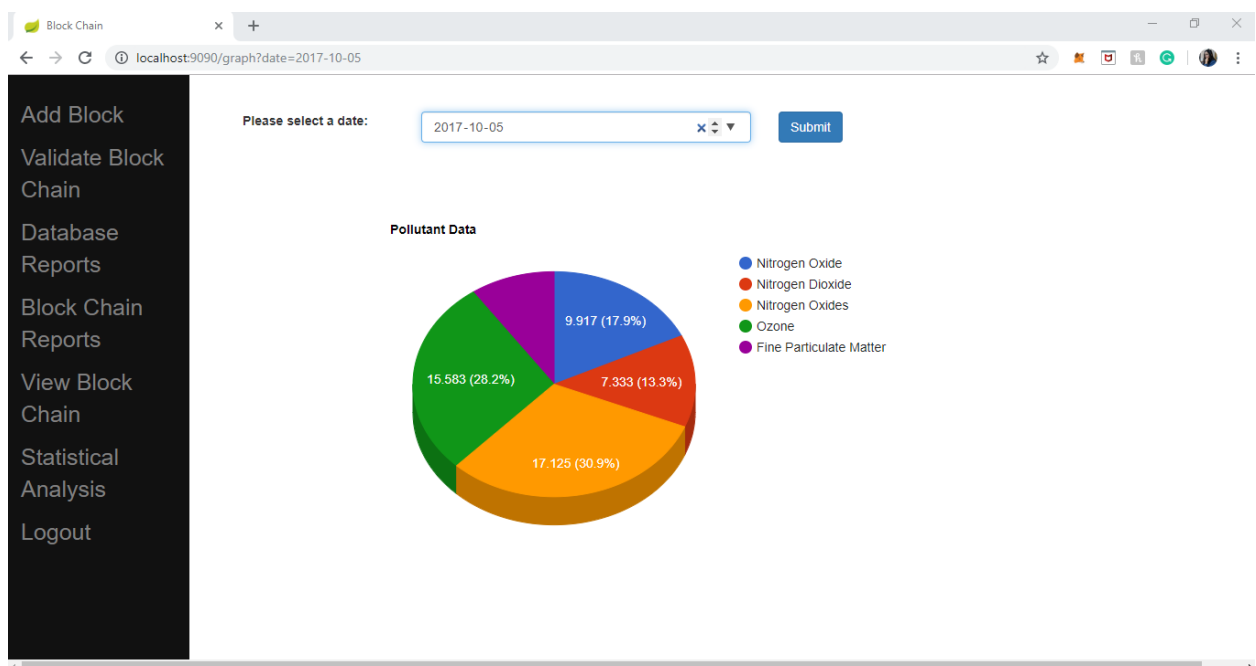
View Blockchain page:

Block Chain

localhost:9090/show_block_chain

Index	Time Stamp	Hash	Previous Hash
1	1568163433094	f4684f34f39487c1c52a33483a51953b512f7b2a571e80b39ea58d879ced42aa	aeebad4a796fcc2e15dc4c6061b45ed9b373f26adfc798ca7d2d8cc5818:
2	1568163433147	a1a4ae52d5f494c88c77bfebefd150d38e0819377cea95bb777aad341cf49baa	f4684f34f39487c1c52a33483a51953b512f7b2a571e80b39ea58d879ce:
3	1568163433147	b2bb5a0efc6e19f68c6e9a9b75e00d75b130c09eb1a188346110757f3b4666fb	a1a4ae52d5f494c88c77bfebefd150d38e0819377cea95bb777aad341cf:
4	1568163433163	32a4e93d51c68088ee2e2d119f1557514dfc9a3a693884ad015f11257de75216	b2bb5a0efc6e19f68c6e9a9b75e00d75b130c09eb1a188346110757f3b4
5	1568163433163	42639dee08a0c9297d94e29f1537cbcb3526878d808108a99ccaf30795f50796	32a4e93d51c68088ee2e2d119f1557514dfc9a3a693884ad015f11257de
6	1568163433163	ca902f7532f1cd1aa9989379b555f4f8286c411fc65278ad81a1406debd5c318	42639dee08a0c9297d94e29f1537cbcb3526878d808108a99ccaf30795f
7	1568163433179	0b5c6bdddeab9d1fe29f3545155f2f153c7a33554be78329cf85493371cd78dd	ca902f7532f1cd1aa9989379b555f4f8286c411fc65278ad81a1406debd5
8	1568163433179	5263a40805885c3bd8c0025ab2aa0f17ba00544673195349102efdf4163b0ef9	0b5c6bdddeab9d1fe29f3545155f2f153c7a33554be78329cf85493371cd
9	1568163433179	efc36ecd902bc5406abf2c65b1d54864a9a21a28607921574a9d3b61bac23c3	5263a40805885c3bd8c0025ab2aa0f17ba00544673195349102efdf416c
10	1568163433179	35d75715f81b52f97c04423e23d6024539046fe893ce5a3026e39c3077e7b871	efc36ecd902bc5406abf2c65b1d54864a9a21a28607921574a9d3b61ba
11	1568163433194	26fd3a88ec53030f561cd587d6c56ec46fe8200d6c89b290ab68be70d3a3c623	35d75715f81b52f97c04423e23d6024539046fe893ce5a3026e39c3077e
12	1568163433194	0222c3c6927695901d2c90b0874e3824f773811243dff9309144af333882313e	26fd3a88ec53030f561cd587d6c56ec46fe8200d6c89b290ab68be70d3a
13	1568163433194	944e236ce93b0b12fac0fc40768638eae17785e4f56b029da9c5f3c981277e36	0222c3c6927695901d2c90b0874e3824f773811243dff9309144af33388;
14	1568163433194	16adb783ba0557bb4f3cd1f3eedd354827c6220878b2946bc33151a008bb2775	944e236ce93b0b12fac0fc40768638eae17785e4f56b029da9c5f3c9812;
15	1568163433194	d1682628d3cbeefacb590a35dcea35a1b17e2c60f64595bcbeee41449149fc8a	16adb783ba0557bb4f3cd1f3eedd354827c6220878b2946bc33151a008b
16	1568163433210	c453f15868a7137aacc24b9d67cf87fd5efca315cfc7f77dc557adf08014d45c	d1682628d3cbeefacb590a35dcea35a1b17e2c60f64595bcbeee414491;

Statistical Analysis page:



7. FUTURE WORK

This project has a lot of scope of improvement in future. It can be designed-

- To directly collect data from air sensors and store them in the form of blockchains
- To send alerts, when the air pollutants value reaches a high critical point
- To access multiple area's air quality data from a single web application, i.e. various blockchain accessible from one point

8. CONCLUSION

This project presents a blockchain-based solution which is representing an air monitoring system for restricting the data manipulation generated by the air-devices. This report presents a technical architecture for a blockchain-based system with the goal of becoming an architecture for environmental development of the air monitoring devices. It mainly focuses on the part of the data storing system to avoid the modification of data which is produced by the air-devices. This prototype can be used for the easy maintenance of data with fewer risks and avoiding the risk of data modification. Right now, this blockchain is developed to prevent the modification of data; in the future, we can develop a blockchain to restrict the data flow in the storage system.

To conclude if we can protect the actual data of air pollution, we can see a lot of changes in the air and the universe, which will help us in creating a healthy environment for the future generations.

9. REFERENCES

- [1] “Blockchain explained”, reviewed by Luke Fortney, Investopedia,
<https://www.investopedia.com/terms/b/blockchain.asp#what-is-blockchain>
- [2] A systematic literature review of blockchain-based applications: Current status, classification and open issues, Authors: Fran Casino, Thomas K. Dasaklis, Constantinos Patsakisa

Websites:

- [3] <https://medium.com/blockworks-group/is-blockchain-better-than-a-database-d518743bdafa>
- [4] <http://graphics.reuters.com/TECHNOLOGY-BLOCKCHAIN/010070P11GN/index.html>
- [5] <https://hackernoon.com/merkle-trees-181cb4bc30b4>
- [6] https://en.wikipedia.org/wiki/Main_Page
- [7] <https://www.objectaid.com/home>