Web Based Cloud Interaction and Visualization of Air Pollution Data

MELODI NERGIS DEMIRAG
Web Based Cloud Interaction and Visualization of Air Pollution Data

Melodi Nergis Demirag

Master of Science Thesis

Communication Systems
School of Information and Communication Technology
KTH Royal Institute of Technology
Stockholm, Sweden

Examiner: Markus Hidell
Supervisor: Peter Sjödin
Abstract

According to World Health Organization, around 7 million people die every year due to diseases caused by air pollution. With the improvements in Internet of Things in the recent years, environmental sensing systems has started to gain importance. By using technologies like Cloud Computing, RFID, Wireless Sensor Networks, and open Application Programming Interfaces, it has become easier to collect data for visualization on different platforms. However, collected data need to be represented in an efficient way for better understanding and analysis, which requires design of data visualization tools. The GreenIoT initiative aims to provide open data with its infrastructure for sustainable city development in Uppsala. An environmental web application is presented within this thesis project, which visualizes the gathered environmental data to help municipality organizations to implement new policies for sustainable urban planning, and citizens to gain more knowledge to take sustainable decisions in their daily life. The application has been developed making use of the 4Dialog API, which is developed to provide data from a dedicated cloud storage for visualization purposes. According to the evaluation presented in this thesis, further development is needed to improve the performance to provide faster and more reliable service as well as the accessibility to promote openness and social inclusion.

Keywords

Internet of Things, Green IoT, environmental sensing, air quality monitoring, data visualization, cloud computing, wireless sensor network, application programming interface
Sammanfattning


Nyckelord

Internet of Things, Green IoT, environmental sensing, air quality monitoring, data visualization, cloud computing, wireless sensor network, application programming interface
# Table of Contents

1 Introduction .................................................................................................................. 1  
  1.1 Background ............................................................................................................. 1  
  1.2 Problem Statement ............................................................................................... 1  
  1.3 Benefits, Ethics and Sustainability ................................................................. 2  
  1.4 Methodology ......................................................................................................... 4  
  1.5 Thesis Structure .................................................................................................... 5  

2 Background ................................................................................................................. 6  
  2.1 Internet of Things ................................................................................................. 6  
  2.2 Internet of Things Technologies ........................................................................ 6  
     2.2.1 Radio Frequency Identification (RFID) ......................................................... 7  
     2.2.2 Wireless Sensor Networks (WSN) ............................................................... 7  
     2.2.3 Addressing Schemes .................................................................................. 8  
     2.2.4 Data Storage and Analytics ....................................................................... 8  
     2.2.5 Visualization ............................................................................................... 9  
  2.3 Green IoT Uppsala Project ................................................................................... 9  
  2.4 Related Work ....................................................................................................... 11  
     2.4.1 AirNow ......................................................................................................... 11  
     2.4.2 AQICN ....................................................................................................... 12  
     2.4.3 Air Quality Egg .......................................................................................... 12  

3 Development Tools ..................................................................................................... 14  
  3.1 HTML, CSS, JavaScript ...................................................................................... 14  
  3.2 JQuery and Ajax .................................................................................................. 15  
  3.3 Google Maps API ............................................................................................... 16  
  3.4 4Dialog REST API ............................................................................................. 16  

4 Data Visualization Tool ............................................................................................... 20  
  4.1 Software Architecture ......................................................................................... 20  
  4.2 Implementation .................................................................................................... 22  
     4.2.1 Filters ......................................................................................................... 22  
     4.2.2 Point Visualization .................................................................................... 24  
     4.2.3 Heatmap Visualization .............................................................................. 26  
  4.3 Deployment ........................................................................................................... 29  

5 Evaluation ..................................................................................................................... 30  
  5.1 Performance ......................................................................................................... 30  
  5.2 Accessibility ......................................................................................................... 31  
  5.3 Best Practices ....................................................................................................... 31  
  5.4 Progressive Web Application .............................................................................. 32  

6 Conclusions and Future Work ...................................................................................... 33  
  6.1 Conclusions .......................................................................................................... 33  
  6.2 Future Work .......................................................................................................... 33  

References ...................................................................................................................... 35
List of Figures

Figure 1: Internet of Things Components ............................................................. 6
Figure 2: Green IoT Uppsala Project System Architecture .................................. 10
Figure 3: AirNow Visualization Tool .................................................................. 11
Figure 4: AQICN Visualization Tool .................................................................. 12
Figure 5: Air Quality Egg Visualization Tool ...................................................... 13
Figure 6: Green IoT Uppsala Project Overview .................................................... 17
Figure 7: MQTT Broker/Subscriber Model .......................................................... 17
Figure 8: HTTP Client/Server Model ................................................................. 18
Figure 9: Web Application Traditional Flow ....................................................... 20
Figure 10: Single Page Application Flow ............................................................ 21
Figure 11: Software Architecture ...................................................................... 21
Figure 12: Main Page ......................................................................................... 22
Figure 13: Main Page with Point Markers ........................................................... 24
Figure 14: Point Region Request JSON Example ............................................... 24
Figure 15: Point Region Response JSON Example ............................................. 25
Figure 16: Point Visualization on Google Maps ............................................... 26
Figure 17: Main Page with Rectangle Marker ....................................................... 26
Figure 18: Rectangular Region Response JSON Example .................................... 27
Figure 19: Rectangular Region Request JSON Example ...................................... 28
Figure 20: Heatmap Visualization on Google Maps ........................................... 28
List of Tables

Table 1: Use Cases in Environmental Sustainability ........................................ 3
Table 2: Air Quality Measurements .................................................................. 10
### List of Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>ICT</td>
<td>Information and Communication Technologies</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>RFID</td>
<td>Radio Frequency Identification</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Networks</td>
</tr>
<tr>
<td>AQI</td>
<td>Air Quality Index</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets</td>
</tr>
<tr>
<td>JS</td>
<td>JavaScript</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>Ajax</td>
<td>Asynchronous JavaScript and XML</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hypertext Transfer Protocol</td>
</tr>
<tr>
<td>MQTT</td>
<td>Message Queuing Telemetry Transport</td>
</tr>
<tr>
<td>SPA</td>
<td>Single Page Application</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Background

Air pollution is a significant issue globally which leads to several diseases causing death, and many studies have been conducted to understand the depth of the problem. According to the World Health Organization (WHO), 3 million premature deaths globally are linked to ambient air pollution worldwide, which is caused by heart disease, stroke, chronic obstructive pulmonary disease, lung cancer, and acute respiratory infections in children [16]. Particulate matter (PM), ozone (O3), nitrogen dioxide (NO2) and sulphur dioxide (SO2) are the main pollutants known to be causing health issues by evidence.

With the improvements in Internet of Things (IoT), environmental sensing systems started to gain importance. The industry is foreseen to be worth $14.4 trillion by 2022 [17] and has already launched different frameworks for the applications for sensing technologies in cities to provide environmental data. Hence, representation of data keeps gaining importance although it is currently not complete and open to improvements in terms of accuracy [18]. Current air quality data and applications intend to show data gathered from sensor networks and are open to public in the cities which are affected more from air pollution.

This thesis project is a part of the Green IoT Uppsala initiative, and the wider aim is to provide open data and systems for governmental entities and citizens in order to provide resources for new services and applications to be developed [1]. The initiative offers a combination of smart sensing and cloud computing technologies, which supports a wide range of applications including environmental monitoring. In this thesis, a data visualization tool is implemented with the objective to provide active monitoring to reduce air pollution and make city planning in a better and more reliable way. This web-based application creates inquiries, interacts with a cloud system implemented to execute them, and displays the results as overlays on geographical maps.

1.2 Problem Statement

Air quality is one of the most important measures to be observed real-time since it affects human health. To be able to implement new policies for sustainable urban planning, there is focus on understanding the level and causes of air pollution first. With the improvements in sensor technology, it is possible to collect different pollutant levels real-time. However, good visualization tools are needed to be able to interpret and analyze the data collected. In addition to government agencies, in most cities today, there are also no tools available for the citizens to understand the level of air pollution to avoid or reduce the risks. There is a need of air quality visualization publicly available so that citizens can have the option to make green decisions to lower their exposure to pollutants in their daily lives.
The main goal of this thesis project is to create an environmental web application platform which specifies inquires and visualizes real-time and historic air pollution data in different ways. Open source technologies are used during the implementation to make it available to everyone, which allows people to observe the air pollution level at the location they want to make decisions accordingly. In addition, it can be used for government agencies to make policies regarding sustainable city development.

It is a huge challenge to represent data using visual metaphors, since there are many factors which affect it, like type of the data, relationship among different parts of data, and interpretation of data. Since the level of understanding of data fluctuates among people in the society, the visualization tool should enable users to search, find, analyze and share the sensor data in different ways. This creates two main requirements: providing good interactivity for the user to filter and show desired data in a way that each of them can understand and interpret, and good performance to provide fast and reliable service. This thesis project aims to result in a web-based data visualization tool which has a user-friendly interface and high performance which is analyzed at the end in detail.

**Problem statement:** How to design and implement a web application for visualizing large amounts of environmental sensor data on maps in an understandable way, and how to provide that application to everyone in a fast and reliable way?

### 1.3 Benefits, Ethics and Sustainability

According to World Commission on Environment and Development, the definition of sustainable development is the development made for the needs of today, which does not put the future needs at risk [2]. And another definition according to the International Telecommunications Union (ITU) is improving the quality of human life and supporting the eco-systems being lived in at the same time. In the field, it is defined through three interconnected domains: economic, environmental, and social. This specific project is tightly coupled to issues related to environmental sustainability, as well as social sustainability since the technology used is inclusive.

Environmental sustainability refers to protecting and maintaining the fabric of natural ecosystems for the future. One of the major drivers is reducing the *carbon footprint* to achieve that, which is a term used to describe the total amount of greenhouse gases produced by human activities. There are different topics which affect the carbon footprint, and as a result of different studies on environmental sustainability and smart cities, related natural themes and associated examples have been synthesized [3]. Table 1 is summarized from the studies shown in [3] and shows different factors to be considered while calculating the carbon footprint, like building low-carbon infrastructures, using green energy (using water, wind, or solar energy as opposed to natural gas), consuming organic food, using greener transportation, and so on. One of the key aspects of environmental sustainability shown in Table 1 is related to air
quality, particulates, NOx, and SOx levels, which is the use case for this thesis project.

<table>
<thead>
<tr>
<th>Theme</th>
<th>Examples</th>
</tr>
</thead>
</table>
| City Infrastructure & Governance     | • Policy and Management - Strategy, Administration, Conservation, Environmental Stewardship, Effective Conservation  
                                         • Infrastructure - Urban Planning, Buildings & Physical Structures, Mobility & Transportation, Public Safety |
| Energy & Climate Change              | • CO2 from Energy Production  
                                         • Emissions per Capita  
                                         • Energy Efficiency & Management  
                                         • Energy Conservation & Use of Renewables |
| Pollution & Waste                    | • Waste - Management, Treatment  
                                         • Air Quality, Particulates, Indoor Air Pollution, Ozone Levels, NOx, SOx  
                                         • Water - Drinking, Water Quality, Water Stress, Management  
                                         • Noise Pollution |
| Social, Economics, & Health          | • Social Services  
                                         • Citizen Satisfaction  
                                         • Education  
                                         • Culture & Social Inclusion  
                                         • Demographics (Aging)  
                                         • Economics - GDP, Employment, Financial Resilience  
                                         • Healthcare, Sanitation, Disease Control & Mitigation, Health Infrastructure & Services |

Table 1: Use Cases in Environmental Sustainability

In addition to environmental sustainability, to be able to access, adapt, and create knowledge by using the information and technology available is crucial for social inclusion, which is closely related to social sustainability [4]. A socially sustainable system is described as achieving fair distribution, health and education, gender equity, and political accountability and participation [3]. It is suggested by some people that Information and Communication Technologies (ICT) is a luxury in the developing world, which became less true by the rapid growth of Internet as a medium of both economic and social transaction [4]. This project provides an open platform to everyone and gives them the chance to have active participation by making green choices for themselves and their families.
ICT access is necessary to overcome exclusion in the societies, whether in
developed or developing countries. From this aspect, this thesis project is
implemented by using open source technologies fully, which means that data
provided and Application Programming Interfaces (API) used are available to
everyone in the world. Using open source technologies have several advantages.
As an example, since the code is open, it can be modified by anyone for
improvements. It also frees businesses from vendor lock-in and enables
interoperability with each other. In addition to using open source paradigm for
inclusion, the design of the web application implemented in this project was
done considering accessibility concerns and evaluated at the end. The idea of
this thesis project is to create a platform for all people to visualize air pollution
data, and by that informing people about their environment and support
sustainable city development, as well as providing companies and organizations
APIs to access data to use on their own products and innovations.

1.4 Methodology

The research methodology used is based on design, implementation, and
experimental evaluation. The first step of this approach is a detailed literature
study, followed by design and implementation. At the end, the implementation
is evaluated by using tools chosen to gather information.

To gain a better understanding of the research area and to be able to
obtain a complete overview, a background study has been done. In this
research, the focus was on Internet of Things, health issues related to air
pollution, and widely used air pollution visualization systems implemented so
far. This literature study has been summarized in Section 2 as Background.

After the related literature study, a user interface and software
architecture design to meet the requirements and provide a good user
experience has been done. While working on this part, since the visualization
tool is using third party APIs and libraries, related technologies are investigated
to use in a better way to get a reliable performance. During the implementation,
agile software development approach is followed, which consists of adaptive
planning, evolutionary development, and continuous improvement that leads
the tool to be flexible to change when needed [5].

At last, an evaluation has been done with the help of Google Lighthouse,
which is an open source, automated tool for web developers to evaluate and
improve the quality of web pages implemented [6]. It provides different audits
regarding performance, accessibility, progressiveness and much more in a
detailed report. The results are presented and discussed in Section 5 as
Evaluation.
1.5 Thesis Structure

This thesis consists of 6 different chapters: Introduction, Background, Development Tools, Data Visualization Tool, Evaluation, and Conclusion. Chapter 1 gives a short introduction to the topic, problem statement and goals of this thesis project as well as an analysis from a sustainability point of view, and methodology used. Chapter 2 gives a more detailed background study consisting of Internet of Things technologies, Green IoT Uppsala project structure in detail, and environmental monitoring systems implemented as related work. Chapter 3 focuses on the development tools used while implementing the project for a better understanding, including 4Dialog REST API, which is implemented for this specific project to be used for fetching required data. Chapter 4 goes in the detail of design and implementation of the part of the project (the data visualization tool) the thesis focuses on by explaining the software architecture and how the technologies work together. At the end, in Chapter 5, the evaluation is done by using the Google Lighthouse tool and analyzed from four different points: performance, accessibility, best practices, and progressiveness. Finally, Chapter 6 is the conclusion and possible future work on the specific implementation.
2 Background

2.1 Internet of Things

For many years, computers have been connected to each other via internet and this opened the path to the idea of connecting other objects (things) to provide a global network by making machines and devices interact with each other. This technological paradigm is called Internet of Things (IoT), and the forecast shows that IoT will reach 26 billion units by 2020 and will have the impact on the information available currently [7].

2.2 Internet of Things Technologies

Internet of Things has three main components from a high-level perspective: hardware, middleware, and presentation. Hardware consists of sensors and embedded communication hardware. Middleware is based on computing tools, data storage and analytics. Finally, presentation is the visualization tools supporting different platforms to interpret the data in a better way [8].

In this section, the most commonly used components will be explained. For the hardware layer, Radio Frequency Identification (RFID), Wireless Sensor Networks (WSN), and Addressing Schemes are widely used to collect data. Afterwards, collected data is stored, processed, and analyzed in the middleware layer with the help of different technologies like cloud computing and machine learning. At last, to be able to make analysis more graphical, visualization tools are implemented, which is the presentation layer. An overview of this structure is illustrated in Figure 1.

Figure 1: Internet of Things Components
2.2.1 Radio Frequency Identification (RFID)

RFID technology is an important part of wireless data communication which provides easy design for microchips to be embedded on different devices. They are attached to anything and can provide identification like a barcode on the device. The difference of the tag compared to a barcode is that it stores data in Electronic Product Code (EPC) which is a global identification system [9].

There are three type of RFID tags: Passive tags, active tags, and semi-active tags. Passive tags do not have their own battery power, they use the transfer to use the radio frequency energy. They are used mainly in retail and supply chain management, transportation and access control. Some examples of them currently being used is bank cards and road toll tags [8]. In contrast to passive tags, active tags have their own power supply and can communicate with a reader by using that energy. External sensor can also be embedded to active tags to monitor variables like temperature, pressure, chemicals, hence used in manufacturing and hospital labs [9]. And the last one, semi-active RFID tags, have batteries but that are used mainly to start the communication with the reader.

RFID tags are preferred because they also have physical advantages. They are small microchips attached to an antenna (for both receiving the signal and transmitting) in a small package [10]. The dimensions go as low as 0.4mm x 0.4mm x 0.15mm. So, having a small size and very low cost, as well as lifetime being not limited by battery are the most prominent advantages for RFID tags. In addition, since sensors can be added on top of them, they also support sensing, computing and communication in a passive system [11].

2.2.2 Wireless Sensor Networks (WSN)

Wireless sensor networks (WSN) are used for monitoring environmental conditions and consist of distributed sensors to gather data. They can communicate with RFID devices for better tracking by taking location, temperature, and movements into consideration. Sensor data are shared among sensor nodes and sent to a distributed or centralized system for analytics.

WSN monitoring network includes different components [8]:
- WSN hardware: A node which contains sensor interfaces, processing units, transceiver units, and power supply.
- WSN communication stack: Consists of topology, routing, and MAC layer for nodes to communicate among themselves.
- WSN middleware: Combines cyber infrastructure with a Service Oriented Architecture (SOA) and sensor networks [12].
- Secure data aggregation: Extends the lifetime of the network and ensures reliable data collection from sensors [13].
2.2.3 Addressing Schemes

Since there are many “Things” connected to each other in terms of Internet of Things, it is crucial to be able to uniquely identify each one of them, and control through the Internet. Each thing should be identified by their unique id, location, and functionality [8]. IPv4 supports geographical identification for a group of sensors, but not individually. In case of IPv6, some of the identification problems are solved, but there are different problems introduced due to the heterogeneous nature of wireless nodes, variable data types, concurrent operations and confluence of data from devices [14]. Another aspect of IoT is the persistent network functioning, since traffic should be channeled relentlessly. Even though TCP/IP makes the routing more reliable and efficient, there is still a bottleneck at the interface between the sensor gateway and sensor devices [8]. Furthermore, since sensors can be added on demand, existing network should have a sustainable address.

Uniform Resource Name (URN) system is fundamental for Internet of Things to be able to address the issues mentioned. URN copies the resources and make the replicas accessible through a URL. To transfer the information from a database to the user, using metadata is beneficial, since there are large amounts of spatial data collected [15]. In addition, there is another development which is a lightweight IPv6 that will enable addressing home appliances uniquely [8].

2.2.4 Data Storage and Analytics

The huge amount of data collected by the sensors has created concerns regarding data collection efficiency, data processing, analytics, and security. This makes storage, ownership, and expiry of the data critical [8]. In addition, many applications require massive data storage, high processing speed for real time support, and high-speed networks to stream any kind of data [9]. To be able to benefit from the data collected by the sensors, data should be stored and processed in an efficient way. The storage challenge is tried to be solved by using cloud-based storage recently, which will open the path to cloud based analytics and visualization in the future. In addition, most recent improvements in artificial intelligence and machine learning leads data processing to be easier and more efficient.

Strong data analytics is possible with the use of different data visualization tools, which is the presentation layer. Data visualization transforms the data into insights by filtering and aggregating data points, which helps with more accurate decisions and strategies. It is especially important in Internet of Things, since data start to make sense once it is seen graphically, and correlation between different parameters can be observed easier to focus on areas to improve.
2.2.5 Visualization

Visualization is an important part for IoT since it provides interaction with the environment for the user. With the help of attractive and easy to understand visualizations, users can benefit from IoT and gain more knowledge on different topics IoT deals with. Visualizing data in meaningful ways also enables policy makers to convert data in knowledge for fast decision making [8]. This thesis project presents a work concerning this specific layer, by implementing a new data visualization tool.

2.3 Green IoT Uppsala Project

Green IoT Uppsala Project is a Vinnova sponsored project which consists of a service combining sensing, collecting, storing, processing, inquiring and visualizing air pollution data. The project has several industrial and academic partners including KTH and 4Dialog AB. It is implemented as a testbed in Uppsala, Sweden, since the air pollutant levels exceed the EU standard in Uppsala especially in winter and spring seasons [1].

The main objective of the project is to build an energy-efficient integrated solution in order to help decrease air pollution levels by monitoring it real time. Another objective is using topics like data science and Internet of Things (IoT) to increase the awareness in the society and improve the environment. The key point of the project is that it provides collected open data for organizations and individuals to allow them to use the data for their innovative projects for sustainable urban and transportation planning.

For this thesis project, different air quality measurements are being collected in Uppsala city center via the sensors built and implemented by the partners (KTH and Upwis). The implemented sensors measure different variables like pollutant levels and meteorological data shown on Table 2. Implemented sensor networks are connected to the cloud services through sensor gateways so that collected data can be stored and processed.

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Unit Name</th>
<th>Unit Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature</td>
<td>Celsius</td>
<td>Cel</td>
</tr>
<tr>
<td>Humidity</td>
<td>relative humidity</td>
<td>%RH</td>
</tr>
<tr>
<td>Pressure</td>
<td>hectopascal</td>
<td>hPa</td>
</tr>
<tr>
<td>Wind Speed</td>
<td>meter per second</td>
<td>m/s</td>
</tr>
<tr>
<td>Sound Noise Level</td>
<td>decibel</td>
<td>dB</td>
</tr>
<tr>
<td>Light Strength</td>
<td>lux</td>
<td>Lx</td>
</tr>
<tr>
<td>Ozone</td>
<td>microgram per cubic meter</td>
<td>ug/m3</td>
</tr>
<tr>
<td>NO</td>
<td>microgram per cubic meter</td>
<td>ug/m3</td>
</tr>
<tr>
<td>NO2</td>
<td>microgram per cubic meter</td>
<td>ug/m3</td>
</tr>
<tr>
<td>-------</td>
<td>---------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>CO</td>
<td>microgram per cubic meter</td>
<td>ug/m3</td>
</tr>
<tr>
<td>CO2</td>
<td>microgram per cubic meter</td>
<td>ug/m3</td>
</tr>
<tr>
<td>SO</td>
<td>microgram per cubic meter</td>
<td>ug/m3</td>
</tr>
<tr>
<td>PM (1, 2.5, 10)</td>
<td>microgram per cubic meter</td>
<td>ug/m3</td>
</tr>
<tr>
<td>RSSI</td>
<td>decibel meter</td>
<td>dBm</td>
</tr>
<tr>
<td>LQI</td>
<td>digital</td>
<td>dig</td>
</tr>
</tbody>
</table>

Table 2: Air Quality Measurements

The overall architecture of the system consists of a network of sensors feeding data through one or more gateways into a cloud-based computing platform with backend database storage. An API (4Dialog API) is implemented to inquire postprocessed datasets which are prepared and returned to the client by the cloud-based computing resources. Since all data and APIs are implemented using open technologies, they can be accessed easily to be used to implement different applications.

![Figure 2: Green IoT Uppsala Project System Architecture](image)

As it can be seen in Figure 2, hardware layer part of the system consists of a WSN (Wireless Sensor Network), and a sensor gateway connected to the middleware layer. The middleware of the system includes the cloud storage used, processing done through machine learning technologies, and implemented APIs to make the data accessible. At last, the presentation layer represents the applications implemented by fetching the data from middleware layer through the APIs implemented. This thesis project belongs to the presentation layer, since the implemented solution is a web application fetching data from the cloud system and visualizing for the users in different ways.
2.4 Related Work

2.4.1 AirNow

AirNow is a real-time air quality visualization tool implemented by the US Environmental Protection Agency. It is based on the international standard Air Quality Index (AQI), which shows the air quality data in numbers and colors to make it clearer for the people who will take further action accordingly [19]. Air quality forecasts are built by ozone and particle pollution data collected from over 3000 federal, state, tribal and local air quality agencies.

AirNow can be accessed via several different platforms: website, social media, email alerts, mobile app and mobile widget. It allows users to search the air quality by entering a zip code or state, then the result is visualized with a spectrum of colors which shows whether the air condition is desirable or not. In the spectrum, green represents the desirable conditions and violet is the opposite showing less desirable ones. Using a color scale on the map provides a better user experience by providing easier and better understanding of the current air quality in a quick way. On the other hand, it is not clear from the map which variables are shown and represent the air quality overall. Figure 3 is taken from AirNow public website http://www.airnow.gov.

![Figure 3: AirNow Visualization Tool](image-url)
2.4.2 AQICN

AQICN is a non-governmental project for visualizing air quality data based on Air Quality Index (AQI) in various parts of Asia, which is supported by Chinese government, Beijing research community, US embassies in China, and private sponsors [20].

Together with the features of AirNow, AQICN also provides information about what variable is measured by different sensors. In addition to displaying readings and overall air quality of different cities on the map, AQICN also shows distinct graphs for each variable. The variables available include NO2, SO2, O3, CO, PM2.5, PM10, as well as temperature, humidity, pressure, wind speed, and dew point. In this way, users can easily analyze the variables they are concerned with separately on the map and graphs. To visualize all, AQICN uses a color scale like AirNow to show if the quality is desirable or not. The portal also provides articles about air quality, frequently asked questions (FAQ) page, and a detailed chart explaining the color scale. Figure 4 is taken from AQICN public website http://aqicn.org.

![Figure 4: AQICN Visualization Tool](image)

2.4.3 Air Quality Egg

Air Quality Egg (AQE) is a crowdsourced project, which gives people the chance to be more involved in air quality topic by active participation. It consists of a sensing device which measures air quality in the environment and allows users to share the results real-time online. After sensor nodes are purchased fully assembled, users can easily register them on the website to begin streaming the measurements in their environment [21].

Once users register their devices on the website, they are shown on the map with their locations on the landing page. Detailed information of distinct nodes registered are available to other users, and they can easily be accessed by
clicking on the egg icons on the map which represent distinct sensor nodes. The information consists of CO, NO2, temperature and humidity measurements at that specific location. Unlike AirNow and AQICN, Air Quality Egg does not use a standard like AQI nor clear units, and visualization does not use a colour scale. Hence, it is not clear from the visualization whether the air quality is desirable or not at the specific area. In addition, there is no way to compare measurements in time, instead just across sensors nearby at that time.

The main drawback of Air Quality Egg is that there is no data standard like AQI is used, which makes the results harder to understand and compare with other air quality monitoring projects [22]. In addition, since sensors contain data about the customer, including their names and addresses, there are possible security concerns which has not been addressed yet. Figure 5 is taken from AQICN public website [http://airqualityegg.com](http://airqualityegg.com).

![Air Quality Egg Visualization Tool](image)

**Figure 5: Air Quality Egg Visualization Tool**

There are many initiatives in the field of data visualization based on environmental data, and the most known ones are AirNow, AQICN, and Air Quality Egg. These tools do not give the freedom to the user to specify different regions, date ranges, or operations (mean, standard deviation, etc.) to see the specific data they are interested in.

This thesis project presents another application to visualize similar data, in a richer and more user-friendly way. It allows users to visualize the data in different ways, like points or heatmaps, in the regions they specify on the map, and by applying the operation they want to the dataset. With this tool, it is possible to do analysis based on different criteria and make green decisions according to different dates, places, and times.
3 Development Tools

This section describes the tools used in the scope of the thesis project, which is a single page web application without a server side. All of them are used in various parts of the front-end development for high efficiency.

3.1 HTML, CSS, JavaScript

Hypertext Markup Language (HTML) is a markup language designed to create webpages and web applications [23]. HTML files which are stored in a web server or local storage, are rendered into webpages by web browsers, and HTML describes the appearance of a page with its defined elements. It is widely used for web pages and has built-in elements like forms which are used in the project. In addition, it is fast and easy to learn, as well as easy to create with just a simple text editor.

Cascading Style Sheets (CSS) language describes the HTML documents’ style by identifying the elements implemented in it. It simplifies maintenance of the website or web application by separating content and style [24].

JavaScript (JS) is a client-side dynamic scripting language for web which supports object-oriented, imperative, and functional programming styles [25]. It runs the scripts within HTML web pages by the client computer for controlling web browsers, interacting with the user, and altering the content within the web browser window via objects [26].

The reasons of using HTML and CSS separately in the project over other outdated methods are as following [27]:

- **Efficiency**: Every time a user opens a webpage, the related HTML file is downloaded which wastes more bandwidth in case its size is huge. In addition, in case of larger HTML files, it takes much more time to download and cost is more. Hence, the better alternative is using simple HTML files for each page by putting the styling and layout information in one separate CSS file to be downloaded just once.
• **Maintenance:** When the styling and layout information is in one place, it means the developer needs to change just that file instead of changing all pages (HTML files) to change the website’s appearance.

• **Accessibility:** Visually impaired users can use screen reader softwares to access the information on the websites through sound rather than sight by the software reading the page to them out loud. And for users with mobility impairments, there are voice input softwares which allows users to navigate on the webpage by voice commands. Using simple HTML files makes it easier to navigate on the page and get the information faster.

• **Device compatibility:** Since the HTML file is plain markup without styling information, different style sheets can be applied to make it compatible with different devices.

• **Web crawlers/search engines:** For webpages to be found by search engines easily, the search engine crawlers should be able to read the pages easily to find the content. Using simple HTML pages makes it easier for crawlers to interpret headings, which makes the pages’ ranks higher in search results.

• **Good practice:** It is a known standard currently to separate content, style, and behavior to develop a web application.

3.2 JQuery and Ajax

JQuery is a cross-platform JavaScript library which makes it easier to manipulate a document, select elements, handle events, create animations, and develop Ajax (Asynchronous JavaScript and XML) applications [28]. JQuery is lightweight, flexible and easy compared to other JavaScript frameworks, which makes it the best option for developers [29]. In addition, JQuery is more than enough for browser compatibility which is required for any good web application to be reached by users from different channels. In addition, JQuery is used to invoke different web services or APIs and process the response in JSON (JavaScript Object Notation), which reduces the post back and improves the performance [30].

Ajax gives the freedom to communicate with the server asynchronously which is different than the traditional synchronous website-server interaction [31]. In the traditional approach, when user submits a request, the server sends back a new webpage as the respond, which causes the whole page to refresh even though there is a small change made. This leads to bad user experience and waste of bandwidth. Instead, Ajax uses XMLHttpRequest to communicate with the server asynchronously, meaning it will send the request, update and inquiry the database at the same time [29]. Afterwards, JavaScript and CSS are used to update the partial page rather than refreshing the whole page [32]. So, it allows the developer to work on other parts of the webpage since the page is partially refreshed thanks to asynchronous communication.
3.3 Google Maps API

Application programming interfaces (APIs) are used for communication between different software components through a set of defined methods. Google Maps API allows developers to fetch data from diverse sources, as well as displaying that data on top of maps to analyze easily. It is the most common API used for mapping, which has built-in functions to add data and control how to display the content in a range of styles [33].

The API consists of a set of JavaScript classes which let developers to customize and embed the map in their webpages. In addition, it allows users to put markers to show the point of interest, draw a shape or create a heatmap layer on the Google map embed.

3.4 4Dialog REST API

4Dialog API has been developed specifically for GreenIoT Uppsala project, by using REST (Representational State Transfer) principles. REST is a term referring to a software architecture style, which is invented by Roy Fielding in their dissertation [41]. It consists of 6 different rules to comply:

- The client and server should have specific roles to be separated, which leads to portability of the user interface to be used in different platforms and improve scalability by making servers simpler.

- Communication should be stateless, and the server should not save the client state. This leads each request to have the information needed for the server to understand the request, and serve more clients, which is crucial for scalability.

- Response data should be labeled as cacheable or non-cacheable, in case user wants to reuse the response data later. This increases network efficiency and reduces the average latency.

- There should be a uniform interface between components, which leads to a simplified architecture and better visibility of interactions.

- The architecture should consist of different hierarchical layers, which improves security by preventing one layer to access the immediate next one. It also improves scalability by allowing to add new layers.

- Clients should be able to download and/or execute applets or scripts, which makes it easier to expand the system without needing features to be pre-implemented.

The API is used to fetch data from the implemented cloud system, which stores air pollutant concentrations’ interpolated values to be used in applications. Sensor network can send data to the cloud system via both MQTT (Message Queuing Telemetry Transport) and HTTP interfaces. Applications
can subscribe to MQTT broker for real time raw sensor data or request processed data through the HTTP interface, and how HTTP interface works from a high level architecture can be seen in Figure 6.

Figure 6: Green IoT Uppsala Project Overview

MQTT is a lightweight messaging protocol, which is based on publish/subscribe model (shown in Figure 7). It is widely used in IoT applications since it is designed to overcome the challenge of connecting different physical devices, like sensors. Since MQTT transfers data as a byte array, it is lightweight, which leads to lower power consumption, and smaller packets to be transmitted. All of these make it a good option for metered or limited connections and sending data efficiently to different recipients via its broker/subscriber model [42].

Figure 7: MQTT Broker/Subscriber Model
HTTP, on the other hand, is based on a client-server model (shown in Figure 8), which requires the client to send a request to the server and the server to send back a response. It is a widely used protocol for the Internet and can also be used for IoT devices to publish data. It is flexible, easy to implement, and has a huge developer base. It also helps simplify the architecture by not requiring a client library to be able to fetch data.

![HTTP Client/Server Model](image)

**Figure 8: HTTP Client/Server Model**

In this project, HTTP interface is used to get the specific data according to the filters user selects. Based on specifications from 4Dialog, the cloud provides an API for applications to fetch data by using HTTP POST messages. In the payload of the HTTP POST, applications need to specify types, geographical coordinates, and timestamps of target data. Applications that require historical or processed data (e.g. interpolated values), can request data from the cloud with 4Dialog API, which can be accessed via HTTP with the domain name “data.greeniot.it.uu.se” and port 80.

The request JSON has the following structure:

- **Time Interval**: The application needs to specify the time interval of the requested data using “From” and “To” fields in the ISO time format e.g. 2017-02-23 17:02:03.

- **Region**: The shape of the requested data needs to be specified with some geographical information.
- **Sensor Region**: Returns the readings for a grid of points available in the rectangle region between 2 points (North East & South West points). The server is assumed to convert the longlat values to meters in an internal cartesian coordinate system and then subsample this grid. If the server uses a precalculated grid of points, it can fulfil this request by selecting a rectangular subset of its internal grid and then, if required subsample this to reach the approximate resolution requested. The client must be prepared to handle replies with grid resolution from 1/2 to 2 times the requested resolution. Note that the reply for each grid point contains the exact LongLat position for the point.

- **Point Region**: In this region type, the application can put one or more geographical locations to fetch its data under the section “Points”. The geographical locations are specified using the “Longitude” and “Latitude” fields.

- **Statistics**: Allows the client to specify some processing to be done before the data is returned. The processing is done individually for each requested position / sensor.
4 Data Visualization Tool

One important part of the project is to implement functionality to specify inquiries and visualize real time and historic air pollution data in different ways on different platforms. The web browser platform is one of the most important platforms for private users. This master thesis work is intended to result in a web-based application to create inquiries, interact with the cloud to get them executed and display the results as overlays on geographical maps.

To build such application, the tools explained in Chapter 3 are used. To visualize the data, a graphical user interface is designed and implemented which communicates with the cloud service via the 4Dialog API. The map is built using Google Maps service and data is visualized on top of it as another layer in two different ways according to the preference of the user: markers or heatmap.

4.1 Software Architecture

The visualization tool is implemented as a Single Page Application (SPA) to prevent from inferior interactivity and responsiveness caused by native web application nature. SPA is a type of web application which consists of a single page that does not reload regularly, thus it provides a better user experience due to better performance. Traditional web applications have performance issues because of the logic in the server side, which can be seen in Figure 9. SPA addresses this issue by performing the user interface logic in a web browser by using HTML templates, and the difference can be observed in Figure 10.

![Figure 9: Web Application Traditional Flow](image-url)
Figure 10: Single Page Application Flow

Traditional native applications load HTML, CSS, and JS every time a page loads again. SPAs reduce this server load by separating the server and front end, and provides a better user experience this way.

Figure 11: Software Architecture
Figure 11 shows the software architecture of the visualization tool implemented in this project with the help of SPA concept. Web browser consists of the web page and JavaScript, and the backend is separate with the HTTP server, public 4Dialog API used in the application, and the cloud storage connected where data is fetched from. It is important to state that this project used the already built cloud storage and 4Dialog API, those parts are not a part of the implementation but rather tools used for fetching data to visualize.

4.2 Implementation

Since the visualization tool is implemented as a Single Page Application, it consists of one main page which has filters to be selected on the left side and a map where a point or region to be picked and visualized on in the right side, which is shown in Figure 12.

![Figure 12: Main Page](image)

4.2.1 Filters

User must select one option for each filter, and the filters are implemented as following according to the 4Dialog API implementation:

- Region Type: Point Region or Rectangular Region
- Date: From date and To date to specify the date range
- Data Type: NO or NO2
- Operation: Mean, standard deviation, maximum, or minimum
- Interval: Ten minutes, hour, day of the week, week, month, or year
Various kinds of visualizations are made according to the “region type” choice. If Point Region is selected, user can draw circles on the map which are the specific points user is interested in visualizing different gas emission ratios. If Rectangular Region is chosen, user can draw a rectangle of any size on the map to get a heatmap visualization of the selected region.

Date input boxes are used to specify a time interval to retrieve the data for. “From” date represents the start date/time, and “To” date represents the end date/time. Data between this interval is fetched combined with the other filter choices. Data type currently is implemented for NO and NO2 values in 4Dialog API and fetch these kinds of emissions to be visualized.

Statistics section of filters consists of Operation and Interval. These statistics allow users to specify the type of processing to be done either before or after data is fetched and this processing is done for each point individually. For Operation filter, processing is done in the client side of the application rather than in backend by the API since it is not implemented in the API. However, Interval filter is calculated in the backend and fetched using the API without including the client side.

The Interval filter consists of different options which changes the value fetched from the API:

- Ten Minutes: One value per 10 minutes in selected date range
- Hour: One value per hour in selected date range
- Day of the Week: One value per day of the week in selected date range
- Week: One value per week in selected date range
- Month: One value per month in selected date range
- Year: One value per year in selected date range

After the resulting value is fetched from the API for each point, the selected Operation filter is applied to calculate a value to visualize in client side of the application. The calculations are done for mean, standard deviation, maximum, and minimum using an external JavaScript library called Math.js.
4.2.2 Point Visualization

As seen in Figure 13, if the user selects Point Region as the region type on the page, the map allows him/her to click and put green circle markers on the map.

![Image of a map with point markers](image)

**Figure 13: Main Page with Point Markers**

Latitude and longitude values of each point are sent to the API combined with other filter choices to fetch one resulting value for each point to visualize. Example request and responses for the point region is given in Figure 14 and 15 relatively.

```json
{
    "TimeInterval": {
        "From": "2017-02-23 20:00:00",
        "To": "2017-02-23 22:03:04"
    },
    "Region": {
        "Type": "PointRegion",
        "Points": [
            {
                "Longitude": 17.6473,
                "Latitude": 59.85670000000007
            },
            {
                "Longitude": 17.6418,
                "Latitude": 59.857
            }
        ]
    },
    "Dataset": "NO",
    "Statistics": {
        "Operation": "cNone",
        "Interval": "cHour",
        "GetAccuracies": false
    }
}
```

**Figure 14: Point Region Request JSON Example**
Figure 15: Point Region Response JSON Example
After fetching data for each point from the API, they are put as the radius of red circles and visualized after clicking on Submit button at the end of the form, which is shown in Figure 16.

Figure 16: Point Visualization on Google Maps

The differences of the resulting values can be observed on the map easily by checking the circle sizes. This simplified approach provides a better user experience by simplifying the information visualized on the map.

4.2.3 Heatmap Visualization

In the case of user selecting Rectangular Region as region type, the map is updated for the user to be able to draw a rectangle on the map of intended size, and it can be seen in Figure 17.

Figure 17: Main Page with Rectangle Marker
Figure 18: Rectangular Region Response JSON Example
Figure 19: Rectangular Region Request JSON Example

```
{
    "TimeInterval": {
        "From": "2017-02-23 21:00:00",
        "To": "2017-02-23 22:03:04"
    },
    "Region": {
        "Type": "RectRegion",
        "NorthEast": {
            "Longitude": 17.647607,
            "Latitude": 59.860500
        },
        "SouthWest": {
            "Longitude": 17.645766,
            "Latitude": 59.858612
        },
        "Resolution": 100
    },
    "Dataset": "NO",
    "Statistics": {
        "Operation": "cNone",
        "Interval": "cHour",
        "GetAccuracies": false
    }
}
```

The selected region’s south west and north east coordinates are sent to the API combined with other filters (example request in Figure 18), and multiple values for each point inside the region are fetched (example response in Figure 19). In order to simplify the heatmap, mean of the fetched values for each point are set as the weight, then heatmap layer is visualized on the map accordingly. The visualization can be analysed in Figure 20.

Figure 20: Heatmap Visualization on Google Maps
The variations of different points in the rectangular region can be observed based on the red circular marks combined. When the value is higher, the red part at that point are larger compared to others. This way, the user can easily compare different points in the selected region to analyze.

4.3 Deployment

In order to be able to check the performance of the web application, it is deployed to a selected host name by using a virtual machine in cloud platform. This approach is followed since cloud applications are easier to build, deploy and scale fast and efficiently. One drawback is that the application performance depends on the specifications of the virtual machine in some extent, and this will be discussed in next chapter as Evaluation.

Static domain hosting is done via FreeDNS. It provides selected domain name to be hosted on user’s own connection to be accessed from a computer with a name instead of an IP address and run own HTTP server using it [34]. For this project, “www.greeniot.tru.io” is picked to run the web application.

In addition, a virtual machine is purchased via DigitalOcean to be used as the server to run the application on [35]. Virtual machine specifications are as following:
- Location: Europe (France)
- 1 GB memory
- 25 GB disk storage
- Operating system: Ubuntu 16.04.3 x64
5 Evaluation

In order to evaluate the performance of the web application, Google Lighthouse tool, which runs several audits on the selected webpage and gives a report to analyze and improve on if needed, is used. It analyzes web applications and web pages, collects distinct performance metrics and insights on developer best practices [36].

The analysis is done based on 4 categories: Performance, Accessibility, Best Practices, and Progressive Web Apps. A series of tests are run on the page to evaluate these categories, like running the application on different device sizes and network speeds.

Runtime environment used has the following specifications:
- User agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
  AppleWebKit/537.36 (KHTML, like Gecko) Chrome/63.0.3239.132
  Safari/537.36
- Device Emulation Nexus 5X: Enabled
- Network Throttling 562.5ms RTT, 1.4Mbps down, 0.7Mbps up: Enabled
- CPU Throttling 4x slowdown: Enabled

5.1 Performance

*Performance* means how fast the web page is downloaded and displayed on the web browser. Higher speed leads to more user satisfaction and better user experience, since it means that the web page is working better with even slow connection and different size of screens.

- First Meaningful Paint shows the time when the primary content of the page is available: 4660ms
- First Interactive marks the time at which page is minimally interactive, meaning necessary scripts are loaded: 13760ms
- Consistently Interactive marks the time at which page is fully interactive, meaning all network resources are finished loading: 17330ms

Depending on those three audits, two performance metrics are calculated:

- Perceptual Speed Index value shows the speed of contents of the page visibly populated [37]: 1946
- Estimated Input Latency shows how long it takes the application to respond to user input [38]: 54

For good performance, Perceptual Speed Index should be smaller than 1250, however the visualization tool exceeded it by getting 1946. This means that the application should be optimized better to load faster. On the other hand, Estimated Input Latency target should be smaller than 100 for good performance, and the application fits the criteria by getting 54.
5.2 Accessibility

Accessible means that the website content and functionality is available to literally anyone, including people with any kind of impairment or disability [39]. Hence, accessibility analysis is based on checking assistive technologies and screen readers working correctly on the page. Unlike the other audits, each accessibility audit is pass or fail. This means a page doesn't get points for partially passing an accessibility audit.

The visualization tool got 96 out of 100 in the accessibility scale, which is a huge success considering the high result. The scale consists of different audits with different weights, which was decided by Web Content Accessibility Guidelines (WCAG) [40] these audits are based on.

The audits passed are as following:

- Elements use attributes correctly
- ARIA (Accessible Rich Internet Applications) attributes follow best practices
- Elements have discernable names
- Elements describe contents well
- Elements are well structured
- Page specifies valid language
- Meta tags used properly

According to the audits, the reason the score is lower than 100 for this webpage is as the following:

- Form element should have associated labels
- Background foreground contrast ratio should be more sufficient
- Different languages should be supported

5.3 Best Practices

Best practices refer to the recommendations for increasing the performance given by the tool. The visualization tool got 65 out of 100 as the score, and the problems to be improved are as following:

- Does not use HTTPS
- Does not use HTTP/2
- Includes front-end JavaScript libraries with known security vulnerabilities (jQuery)

The reasoning behind not using HTTPS and HTTP/2 is that the 4Dialog API used in the application to fetch data does not support them. All GET and POST requests had to send to an API URL with HTTP, hence this is something to be improved from the API side. By this fix and checking security with external libraries used, the score can be increased dramatically in the future.
The reason of audits showing the library used (jQuery) having security vulnerabilities is because the version of jQuery used has multiple vulnerabilities, which are all fixed in the later versions. This can be easily improved by upgrading the library to the latest version.

5.4 Progressive Web Application

For an application to be progressive, it should be fast enough for any internet connection, reliable, and engaging to the users. Google Lighthouse tool checks the baseline and gives the basics to cover as a list. The visualization tool should be improved because it does not cover whole baseline due to following reasons:

- Does not respond with 200 when offline
- Does not use HTTPS
- Does not redirect HTTP traffic to HTTPS
- Page load is not fast enough on 3G

The reason behind not using HTTPS is (like explained Section in 5.3) the API not using it. Since the API URL uses HTTP, to be able to connect and fetch data HTTP had to be used. However, the traffic is not redirected to HTTPS from HTTP to handle this problem, since the application was not in production and used by real users. It is a solution to be implemented to ensure security if the application is public to users.
6 Conclusions and Future Work

6.1 Conclusions

The concept “Smart City” used Internet of Things services to provide everyone (people, companies, organizations) better understanding of the challenges every society faces, like air quality, transportation, and energy efficiency. The first step for better solutions is providing open data and standards so that everyone has access to data to analyze and base their work on. The second step is providing protocols and interfaces for third-party to make it easier to access the data, which is provided by 4Dialog API in this specific project. Finally, the third step is building applications and experiments with the data gathered so that every single person can understand and improve their knowledge, not only organizations or developers who can use the open interfaces.

This thesis concerns all the steps, but implements the third one, which is a web application visualizing data gathered for everyone to understand and easily use. It presents the whole development process as well as detailed background information to help users filter and visualize data in specific regions, between specific dates, and in different shapes. The design of the user interface is done by brainstorming for the best user experience and making data visible in an easily understandable way so that users from every background (technical or non-technical) could use the same tool and understand the results. By providing such platform, the aim of improving the knowledge of everyone especially about environmental issues can be achieved easily, and governments can act on it this way, since knowledge is power.

6.2 Future Work

The performance of the web application is evaluated according to four criteria: Performance, Accessibility, Best Practices, and Progressive Web Apps, which are explained in detail in Chapter 5. When the results are analyzed, the performance seems to be good, but there is space for improvement for better performance. Especially since inclusive practices are important in technology, audits on Accessibility should be analyzed in detailed and improved. This would make the application available for more people, which is the main objective of the overall project.

For the improvements with the user interface of the web application, an adequate number of people should be interviewed. In this way different kind of users can test the user interface and give feedback about the parts they think should be changed. Afterwards, the results can be gathered, and necessary improvements should be made accordingly for a better user experience. Especially the data visualization on maps part is important in this case, since the aim is to make people understand the data in a nice way.

Since GreenIoT Uppsala project is a large project with different parts, the web application depends on the performance of other parts like the 4Dialog API and the cloud system. Any improvement made on those parts would affect
the performance of the application in a better or worse way. That is the reason there should be end-to-end testing by including all parts and decide on what and how to improve each part by testing the flow from one end to the other.

Finally, related metrics should be decided on and monitored in order to evaluate the overall performance of the project. The monitoring part can be done easily with third-party tools, like Google Analytics, which provides different graphs per metric and gives detailed reports. The metrics should include how many users are actively using the application, how long they stay on the webpage, what kind of data they are interested in, from what regions the user traffic is more, etc. After monitoring, the defined metrics for some time, necessary technical changes can be made for a better performance.
References


[40] “Web Content Accessibility Guidelines (WCAG) 2.0.” Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C, www.w3.org/TR/WCAG20/.
