University of Bahrain
College of Information Technology
Department of Computer Science

MAQIAS

Prepared by

Reema Hesham Khairalla	**20189452**
Amna Sameer Ahmed	**20176167**
Zakeya Ahmed Al Saeed	**20185035**

For

ITCE499/ITCS498

Senior Project

Academic Year 2021-2022-Semester 2

Project Supervisor: Dr. Mustafa Hammad

Dr. Ebrahim Janahi [CE]

16 May 2022

Abstract

Air pollution is one of the hardest to maintain and control, with Bahrain's goal of Net-Zero emission by 2060, this project provides a monitoring system that displays the air quality indicator (AQI) emitted by vehicles near traffic lights. The implementation of the monitoring system consists of integrating the hardware sensors and image processing created by using machine learning models to AWS Cloud to display it on a dashboard. The dashboard is equipped with a heat map, and tables and graphs for comparison between AQI in different areas in Bahrain. The dashboard will serve as a base to support on decision making and better planning against the traffic and road planning.

Acknowledgments

We would like to express our deep sense of gratitude to the people whom without their help, this project would not be possible to accomplish. First and foremost, we would like to share our profound feeling of thankfulness to Amazon Web Service (AWS) team and most importantly to Aysha Sayedi, Hamad AlKhal and Abdullatif Rashdan who gave us this golden opportunity to be part of the collaboration between University of Bahrain, Supreme Council of Environment and AWS to do this project. We also want to thank them for their nonstop guidance and assistance during the whole journey. Additionally, we want to acknowledge the sincere efforts and valuable time given by our supervisors Dr. Mustafa Mohammed Hammad and Dr. Ebrahim Janahi who encouraged us to complete our project. We would like also to thanks our senior project coordinator Dr. Mohammed Almeer for being always there to give his advice and response to our inquiries. Lastly, we want to mention our gratitude to our families and friends for continuously supporting us.

Table of Contents

List of Tables

List of Figures

Chapter 1
Introduction

Bahrain announced the intention of reaching the net-zero carbon emissions by 2060 to increase the protection of our earth environment. Prince Salman bin Hamad Al Khalifa stated that the government will start investing in new carbon capture technologies (Bahrain News Agency, 2021). Therefore, this project was developed with the purpose of helping our country achieve goal of reaching a net-zero carbon emissions by 2060. The project is a collaboration between Amazon Web Services (AWS) with University of Bahrain (UoB) and the Supreme Council of Environment (SCE).

Supreme Council of Environment in Bahrain reported that the transportation sector is one of the major factors affecting the air quality in the country. A conducted study in 1997 declared that gas emissions and in particular Carbon Monoxide emission is nearly 39.1% emitted by vehicles as it is accounted approximately 49% of total pollution missioned annually (Bahrain News Agency, 2021).

The project principle concept is to calculate Air Quality Index (AQI) next traffic lights to correlate the relationship between vehicle and AQI. This was done by first, using MQ7 Sensor to read the Carbon Monoxide (CO) readings through Arduino microcontroller using Python programming language in order to calculate the AQI. Then, an integration between a streaming live camera with the small and powerful embedded system Jetson Nano to program it for object detection and image classification to count and identify the vehicle's type along with the vehicles wait time in the traffic light. Finally, for monitoring the dashboard, a site was designed having the monitoring tools to display the data collected from the sensors stationed on the traffic lights.

1.1 Problem Statement

Bahrain does not have access to accurate data when it comes to air pollution near traffic intersections. Many different factors contribute to the level of pollution near traffic lights such as the type of vehicles passing through and traffic congestion. Such factors are well studied and not readily available for view by the SCE which is where the issue lies. The data can help in constructing statistics that can contribute to effective decision making and policy changing to design better city development that assist to enhance the air quality of Bahrain which will then align with Bahrain's 2030 vision and Sustainable Development Goals (SDG) for well controlled air quality.

1.2 Project Objectives

The project aims to provide a monitoring solution with real-time data hosted on Amazon Web Services (AWS) cloud to utilize its computational power and high availability. UOB Cloud Innovation Center collaborated with AWS and Supreme Council of Environment (SCE). The

main objective is to create a system that can help to analyze and correlate air quality indicator (AQI) to traffic pollution next to traffic lights and the factors affecting it. The solution will:

- Help in achieving the goal of bringing carbon emissions to net zero by 2060
- Provide an advanced, user-friendly, and real-time dashboard that uses monitoring tools to display data collected from sensors stationed on traffic lights
- Allow the SCE manager to observe and compare Air Quality Indicator (AQI) at different locations and assorted periods of time
- Implement Machine Learning (ML) to detect and track vehicles next to traffic lights
- Host the designed dashboard which operates using IoT and Machine Learning (ML) technologies on Amazon Web Services (AWS) cloud.
- Provide clearer information to Supreme Council of Environment to have better estimation of the correlation between vehicles emissions and air quality.

1.3 Relevance/Significance of the project

The importance of this project lies in the ability of identifying the air quality in Bahrain's traffic lights which accordingly will align with Bahrain's 2030 vision and the Sustainable Development Goals (SDGs) to provide better planning for Bahrain traffic distribution, and assists real estate investors in selecting the cleaner areas and industrial investors for the industrial areas. Recognizing the correlation between traffic pollution next to traffic lights with the level of pollution being produced in the area will help to better estimate the pollution in accordance with the type of vehicles and their quantity at each area. Additionally, it aids to control the vehicles exhaustion quality in Bahrain's streets. All this will support into net zero carbon emission 2060 Bahrain announcement.

1.4 Report Outline

The following report is divided into 7 chapters. In chapter 2, a literature review demonstrating similar systems and solutions is presented along with their brief summary. In chapter 3, tasks and activities that were performed throughout the lifecycle of the project are presented. This includes an introduction and justification of the Software Development Life Cycle model that was chosen and followed throughout the project's duration, identification of risks that might be encountered during the project's lifecycle, and a breakdown of the project activities and the time that was spent on each activity. In Chapter 4, functional and non-functional requirements of the solution are gathered and analyzed. This includes a discussion of how the requirements were elicited, a list of the requirements that were determined necessary for the project, a section describing the characteristics of the solution and the clients that use it, and finally a section for the system models is presented. In Chapter 5, a deeper dive into the solution's design is presented. This includes further analysis into the solution's various aspects such as database design, user interface design, and diagrams concerning the solution's architecture such as deployment and sequence diagrams. In Chapter 6, implementation decisions and tools used are discussed along with testing phases that have taken place.Strengths and weaknesses of the solution are also highlighted here. Finally in Chapter 7, a conclusion regarding the project is given along with limitations and potential for future work.

Chapter 2
Literature Review

The process of examining and assessing literature related to this project is defined as the literature review. The literature review consists of analyzing the literature, summarizing the information found, evaluating the gathered information by focusing on the features, what the system aims in the current state of these systems, theories, and opinions, and finally presenting the results (McCombes, 2019). As a result, this chapter will demonstrate two systems that are comparable to the system we are implementing. The chapter contains a description of the system's characteristics, purposes, and services.

2.1 Related Systems

2.1.1 IoT Solution for Smart Cities' Pollution Monitoring and the Security Challenges

The Internet of Things (IoT) solution intends to monitor pollution levels, particularly those found in cities, by utilizing a variety of IoT devices and associated sensors. The concept is to have several stations located across cities. These stations upload and send data to the IoT cloud on a regular basis. The authors employed development boards such as Nitrogen iMX 6 or Raspberry Pi for proof of concept and prototyping, but for the final IoT solution, fully calibrated high-quality sensors and industrial IoT gateway devices such as HMS Netbiter or analogous equipment are intended. The built security for the IoT gateway and connections to the IoT Clouds is the primary distinction between this solution and others. For improved prediction and data analysis, artificial intelligence (AI), ontologies, and other technologies may be applied.

Main contributions:
1- Design of a big scalable architecture and infrastructure for collecting pollution metrics as part of an IoT smart city solution.
2- An examination of the technologies and platforms that may be employed in a pollution monitoring system.
3- Using best practices in software development technologies, design and execute a proof of concept (PoC) of an IoT architecture to gather pollution measurements within a smart city.

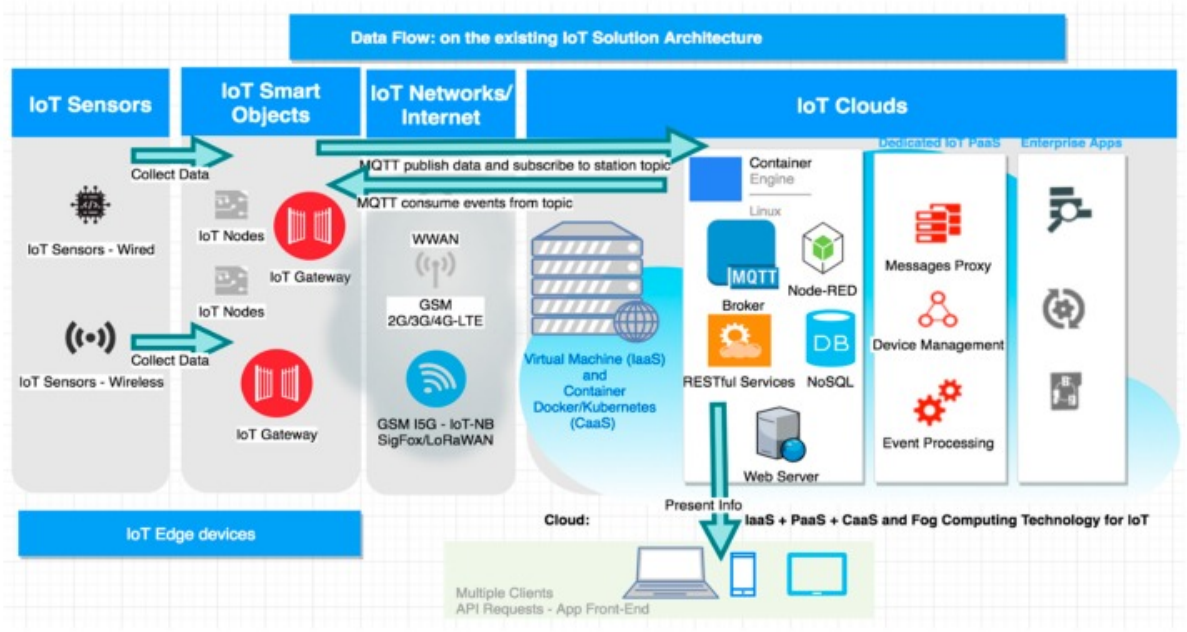

Figure 1: Architecture of the system

Architecture of the system contains the following components:
IoT Edge Devices
IoT Smart Objects:
IoT nodes
IoT gateways
IoT networks/internet
Solution clouds:
IaaS/CaaS (Amazon EC2 with containers as Docker and Kubernetes)
IoT PaaS
Enterprise Systems: business intelligence (BI), artificial intelligence (AI) and data analytics clouds or enterprise systems may be used to provide a strong interpretation of the process data.
Front-end devices for management and reporting (e.g., smart city pollution map creation).

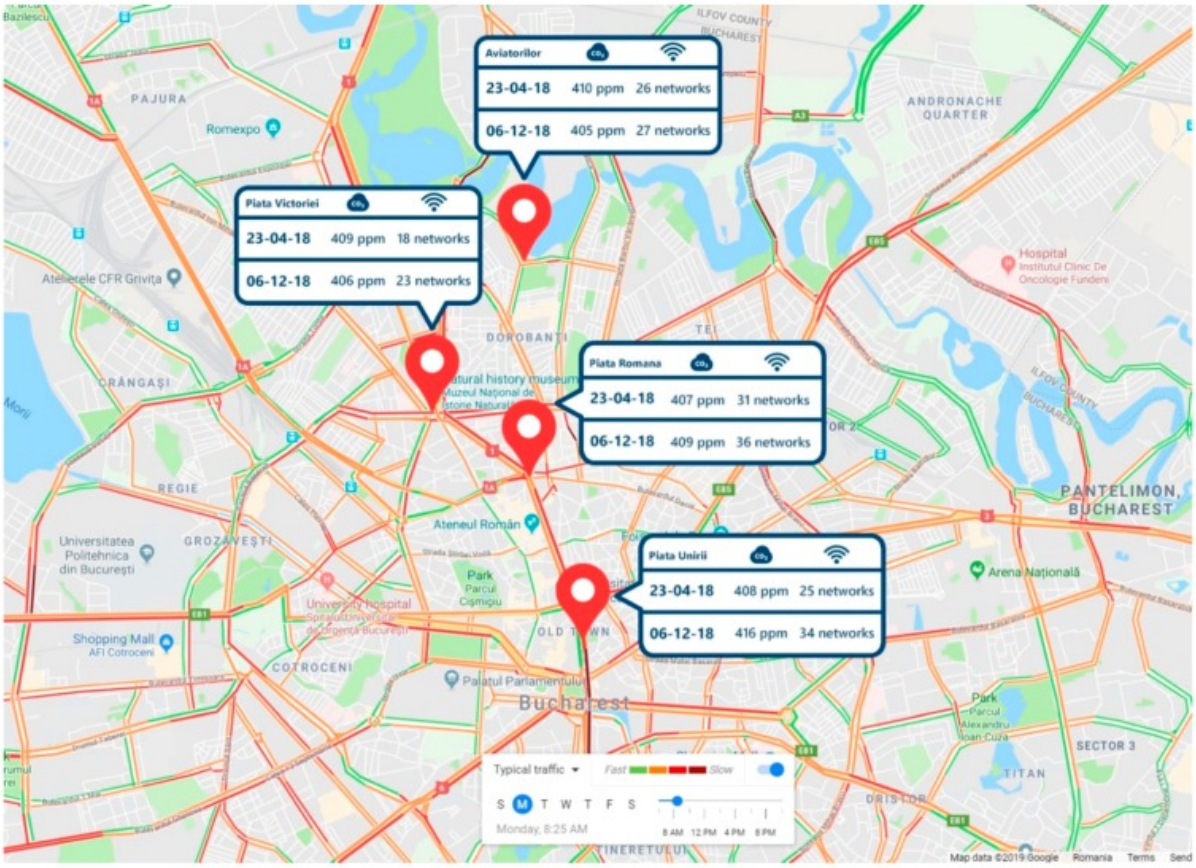

Figure 2: Analytical comparison

For example, in an analytical comparison between 23 April 2018 and 6 December 2018, several regions in Bucharest with pollution indicators (including decibels caused by noise and Wi-Fi networks)

PM2.5 Legend | WHO target | Good | Moderate | Unhealthy for sensitive groups | Unhealthy | Very unhealthy | Hazardous | Unit: µg/m³

Rank	City	2017 AVG	2018 AVG	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
1	Iasi, Romania	-	27	35.1	34.9	34.4	27.9	22.3	20	24.5	20.5	17	29	24.8	31.4
2	Cluj-Napoca, Roma...	-	21.4	26.1	19.2	22.6	21.8	17.1	15.9	-	-	-	26.7	21.7	-
3	Brasov, Romania	-	21.3	48	26.5	24.1	17.8	14.7	13.4	12.2	16.3	16.4	19.8	17.4	26.3
4	Bucharest, Romania	-	20.3	26.1	22.9	26	19.8	17.7	14.8	12.8	17.4	15	24.4	18.9	27.7
5	Ploiesti, Romania	-	19.4	28.5	22.7	29.5	17.9	15.2	15.3	15.5	15	14.7	19	-	-
6	Magurele, Romania	-	18.9	24.5	21.5	20.4	22	16.4	14.1	11.3	16.7	16.4	21.8	17.9	28.8
7	Timisoara, Romania	-	18.6	25.9	21.6	26	17.8	9.9	11.9	12	15.1	16.6	23.4	22.3	22.9
8	Arad, Romania	-	17.9	24.7	24.7	27.9	26.5	12.3	8.9	11.7	14.5	12	21.5	22	-

Figure 3: Top polluted cities in Romania

The figure shows the top polluted cities in Romania in 2018 in terms of PM2.5 according to the AirVisual Project.

2.1.2 An IoT-Aware Solution to Support Governments in Air Pollution Monitoring Based on the Combination of Real-Time Data and Citizen Feedback

Proposes a solution to assist governments in monitoring city pollution by combining user feedback/reports with real-time data obtained by specialized mobile IoT sensors dynamically re-located by government personnel to verify the claimed conditions of specific locations. Through machine learning techniques, the mobile devices use dedicated sensors to monitor air quality and collect primary road traffic situations. The system exposes a mobile application and a website to facilitate the collecting of citizen reports and the display of collected data to both institutions and end-users. The suggested method has been prototyped at a medium-sized university campus as a proof-of-concept. Both the performance and functional validation proved the system's viability and efficacy, allowing for the identification of certain lessons learned as well as future development.

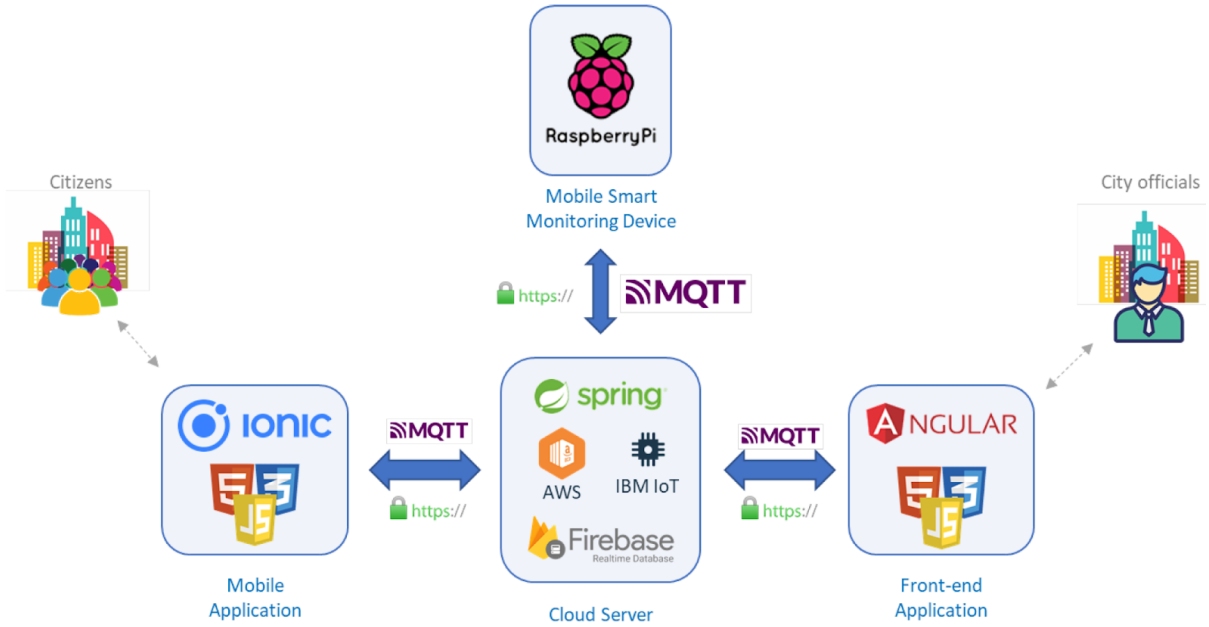

Figure 4: Proposed system architecture

shows the proposed system architecture, designed to satisfy all the reported requirements and serve the two main stakeholders: citizens and city officials.

The shown architecture is composed of four main parts: a Mobile Application (App), a Mobile Smart Monitoring Device, a Cloud Server, and a Front-end Application.

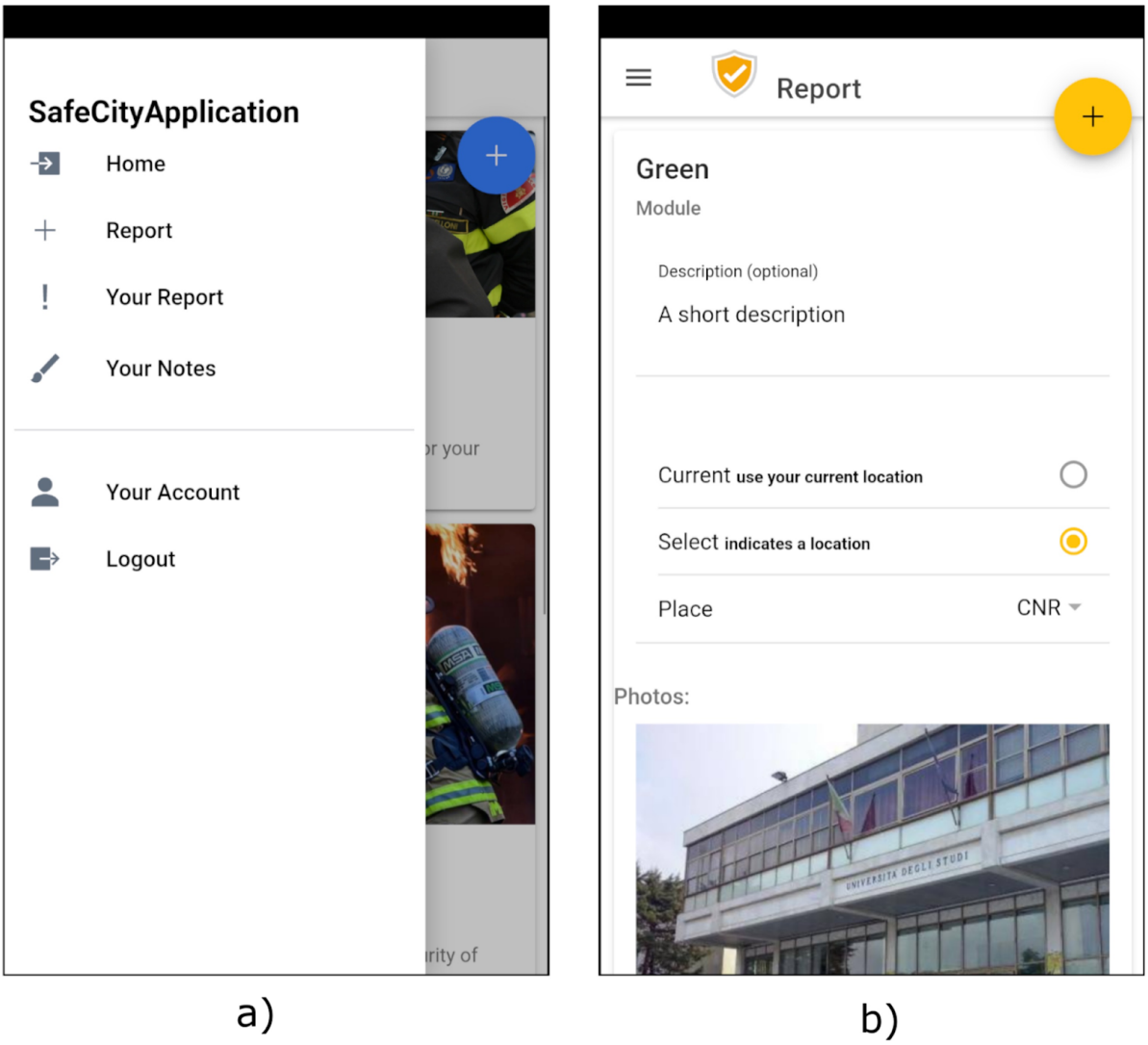

Figure 5: Screenshots of mobile application

The menu of the application (**a**); the form used by the user to report a problem (**b**).
Finally, the Mobile Smart Monitoring Device completes the architecture. The Raspberry Pi 4 Model B board was chosen to implement this component in the first realized prototype, as shown in Figure 4; however, the versatility of the planned architecture allows for the adoption of alternative more suitable hardware solutions (e.g., an ad hoc designed hardware board). It is a tiny computer made up of numerous pins known as GPIOs that can control sensors and actuators. It is primarily responsible for gathering monitoring data. In fact, as illustrated in Figure, this device has multiple sensors attached to measure temperature, gas values (e.g., CO, CO2, AMM), humidity, rain present, and traffic circumstances.

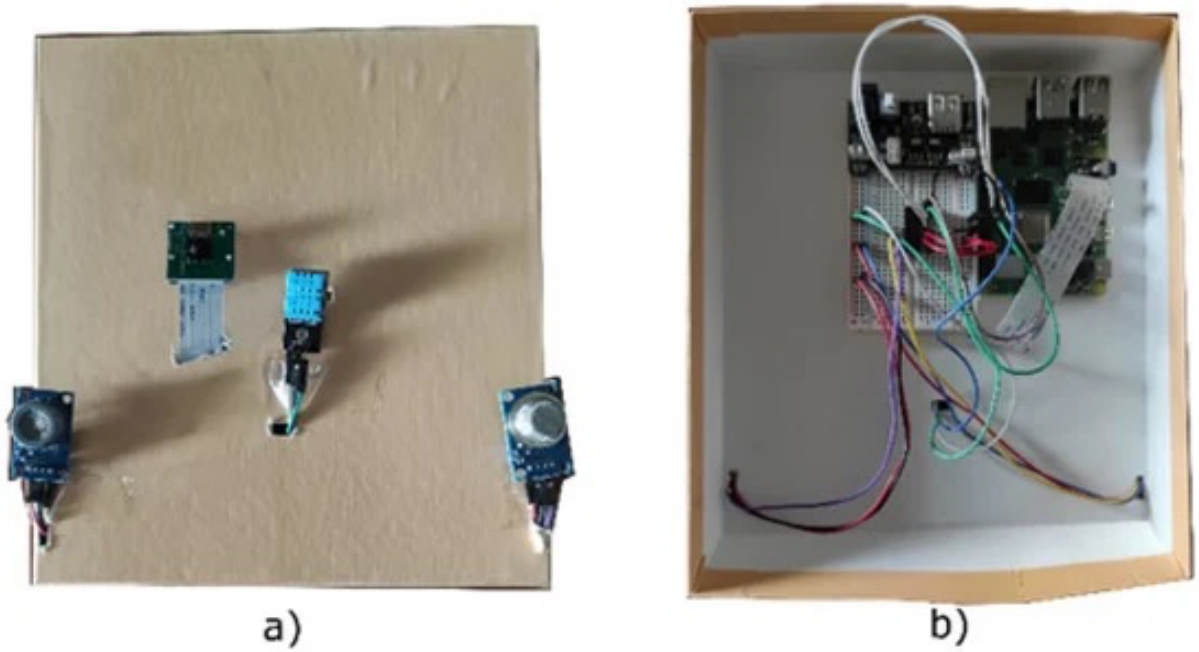

Figure 6: Different sensors used in the solution

The Raspberry was specifically outfitted with the following sensors and interfaces: MQ7 Sensor and MQ135 Sensor to measure air pollution via factors such as carbon monoxide (CO), carbon dioxide, hydrogen, toulene, and ammonia. MQ sensors, in general, detect the presence of gas in the air. The DHT11 Sensor is used to measure temperature and humidity. To detect rain, use the YL-83 rain Sensor. The Raspberry Pi Camera is used to take photographs of city streets and zones and, if necessary, assess traffic conditions by asking the Cloud Server and running the Python Flask machine learning algorithm.

2.2 Summary

Two different solutions that are similar to our project were discussed in this section. Reviewing those solutions contributed to the process of gathering requirements and information to help develop our project. Concepts and techniques used in previous solutions guided us to choose the right tools and approaches to implement our system smoothly. Although the solutions are similar in a way to our project and comparable to it, MAQIAS has its own architecture and functionalities which make it unique.

Chapter 3
Project Management

3.1 Process Model

The project was proposed and given a total of 3 months to be implemented. As such, an Agile Software Development Life Cycle was chosen to be followed throughout the project's development. In an Agile SCRUM approach, work is broken down into small iterative increments called sprints. During each sprint, the team comes together to collaborate, implement a part of the solution, then introduce new functionalities to the system or change them.
Prior to beginning the SDLC, collaborative meetings and a kick off meeting were initiated between the project's team and the SCE Manager to determine the requirements of the project along with the expectations of the final product. After iterating over the initial requirements elicitation a product backlog, sprint backlog and user stories were made. Then, the agile approach was chosen to develop the solution due its nature.
The agile approach is usually chosen due to several reasons. Some of which are how it allows for changes in the solution when the final product is not very clear, it minimizes risks, and it gives the ability to be transparent with the development progress. Additionally, it is usually focused on producing solutions and allowing for changes in a much faster and flexible manner in comparison to other methodologies. Due to this project's flexible nature, short development duration, and having the ability to directly involve the users and focus on their needs this approach was chosen. (Synopsys, 2022)

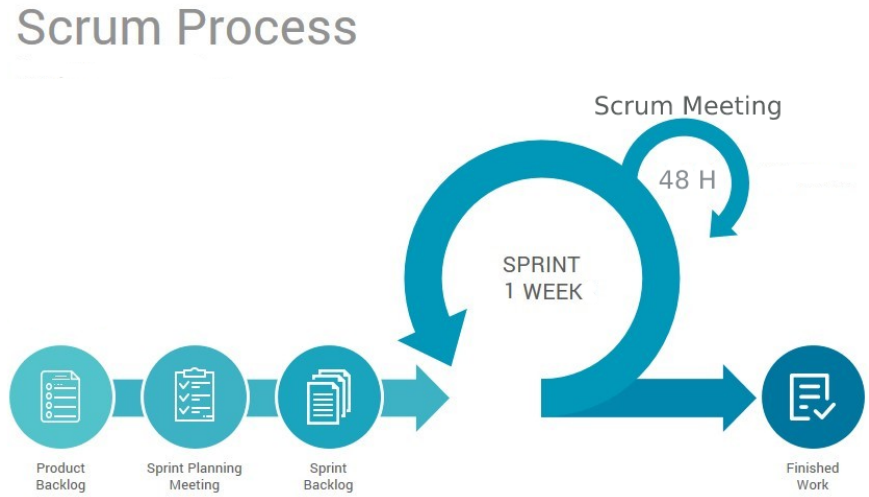

Figure 7: Scrum Process

Despite having the Agile Methodology as the best fit for this project, its ideas and phases were taken and changed to better fit the needs of the project and the nature of the work done. Each work sprint would be one week long instead of two to four weeks long. During this sprint, the project's team would collaborate to develop parts of the solution. The project's team was split into two: a front end/User Interface team and a back end/Hardware team.

These two teams focused on developing their part of the solution and worked synchronously throughout the project.
In each sprint, members would be assigned a task from the backlog first. Then the members would work on completing the task assigned. The task's deliverables would then be either tested or put for review. If changes arise in the next sprint that involves the previous sprint's deliverables, then the task would be iterated over again and modified to better fit the new decisions.
To ensure that work was being tracked and followed, a backlog was made and team meetings were scheduled. The backlog acted as a tool that supports the agile methodology. It involved a flexible breakdown of the work to be done within each sprint. Then the work of each sprint would be discussed in meetings. These meetings were scheduled to be half an hour long every other day between the project development team members then an hour long meeting every wednesday between the development team and the Cloud Innovation Center. During these meetings, members of the development team would update others of their current work status, voice their concerns, or introduce new changes to the solution.

3.2 Risk Management

Despite having reduced risks due to the project's agile approach and flexible nature, some potential problems could still arise. As such, a risk register and a risk management plan were put in place to further reduce the rise of these problems.

1. Risk Management Plan:
   The following risk management plan was created to plan how to identify and overcome potential risks. It involves the following: methodology, roles and responsibilities in risk, budget and schedule, risk categories, risk probability and impact, risk documentation.
   1.1 Methodology:
   The following risk management plan was created as a collaborative effort between the team members. The members identified potential risks, and solutions to overcome them.
   1.2 Roles and activities in risk:
   Individuals involved with the project have responsibilities in mitigating or overcoming risks. These responsibilities are mentioned in table 1 - entities and Responsibilities.

Table 1: Entities and Responsibilities

Entity	Responsibility
Cloud Innovation Center	• Keep track of the project progress to make sure no risks have occurred • Handle responses when risks occur • Be notified of risks • Inform team members of risks
Project Sponsor	• Ensure risks involved with budgeting are minimized
SCE Manager	• Ensure risks involved with satisfying project requirements are kept at minimum

	• Ensure final solution does not contain risks concerned with nature of calculations and design
Team Leader	• Ensure project team is on track with development • Ensure risks are communicated when they arise
Development Team Members	• Communicate when risks arise

1.3 Budget and Schedule:
Each risk-related activity will have a different estimated cost and duration based on its nature

1.4 Risk Categories:
Risks that might occur during this project have been categorized in the Risk Breakdown Structure in the diagram below - Risk Breakdown Structure.

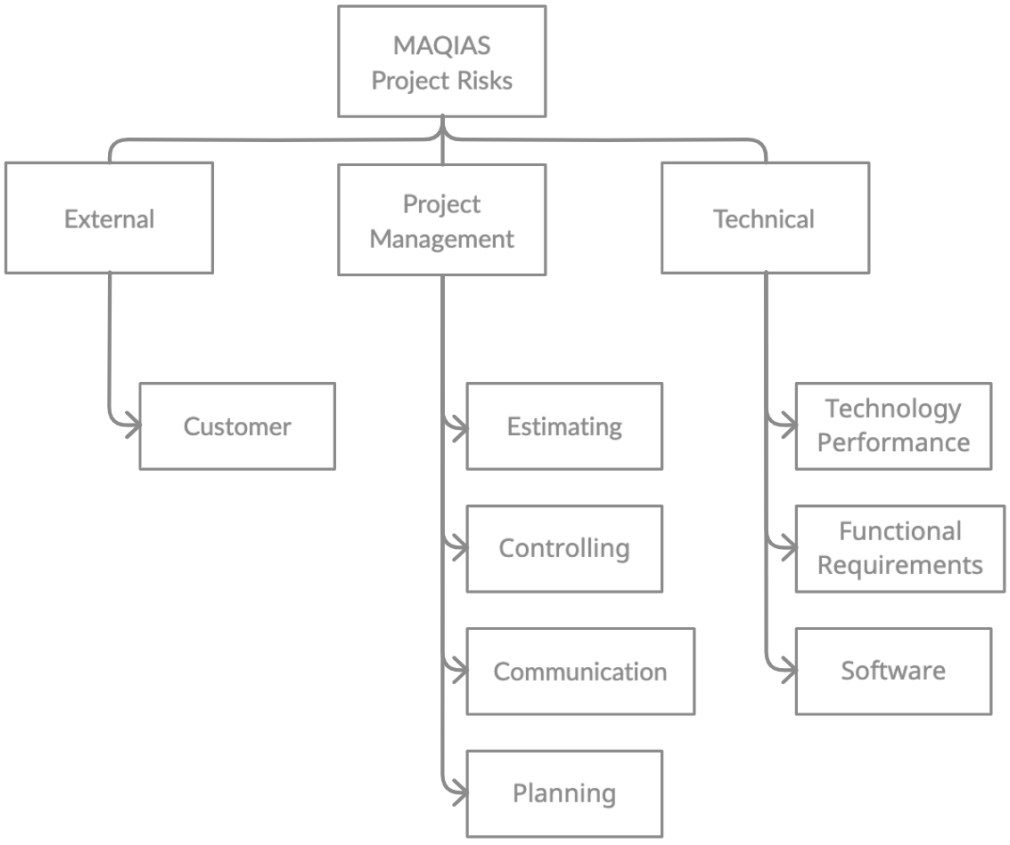

Figure 8: Risk Breakdown Structure

1.5 Risk Probability and Impact:
A probability and impact matrix for risks mentioned in the risk register has been created and presented in diagram 2.0 - probability and impact matrix. The probability varies from high(very likely), medium (somewhat likely) and low(unlikely). The impact varies from high, medium and low.

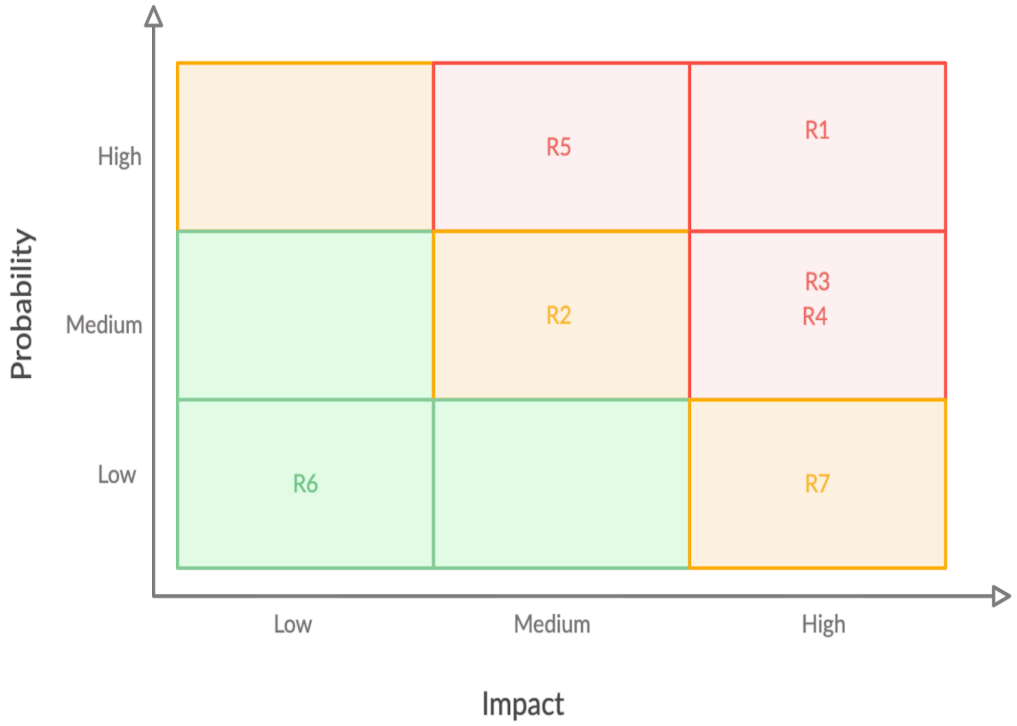

Figure 9: Probability and Impact Matrix

1.6 Risk Documentation:

Risk categories related to the project have been presented in diagram 1.0 - Risk Breakdown Structure. All identified risks have been presented in table 2.0 - Risk Register and mapped based on probability and impact in diagram 2.0 - probability and impact matrix.

2. Risk Register

The table shows Risk Register which presents risks that have potential to occur during the project's development.

Table 2: Risk Register

No.	Rank	Risk	Description	Category	Root Cause	Triggers	Potential Response	Probability	Impact	Status
R1	1	Not on schedule	Estimated Project tasks completion dates are not met	estimating	Inadequate planning and inadequate ability to stay on track	Spending more time on tasks than necessary, not breaking down large tasks	Reduce functionalities to be involved in final solution	High	High	Did not occur

R2	2	Unmet requirements	Requirements have not been properly defined	Functional Requirements / Planning / Customer	Iterating over the same part of the solution without focusing on other parts	Not capturing and keeping track of all requirements	Going over requirements with customer	Medium	Medium	Did not occur
R3	3	Unapproved changes	Changes have been made to solution without gaining approval	Controlling	Lack of management control and communication	Applying the unapproved changes	Go over changes and document them	Medium	High	Did not occur
R4	4	Tools and devices used not working	Tools and devices involved in the project are not working or do not support the solution	Technology Performance/ Software	Insufficient research before selection of tools and devices	Solution cannot use or integrate with tools and devices	Research alternative tools and devices	Medium	High	Occured
R5	5	Undocumented changes	Changes that are made are not documented	Controlling / Communication	Lack of communication	Committing changes without approval	Go over changes and document them	High	Medium	Did not occur
R6	6	Inaccurate calculations within solution	Calculations within source code of the project are inaccurate	Software	Insufficient research prior to implementing calculations	Applying inaccurate calculations	Research and prove calculations	Low	Low	Did not occur

R7	7	Insufficient costs	Estimated costs for the project cannot be covered	Estimating	Inaccurate cost estimation of project	Going over the funded budget.	Request fund increase on account	Low	High	Did not occur

3.3 Project activities Plan

The below Gantt chart represents the workload of the team during the whole process of the project.

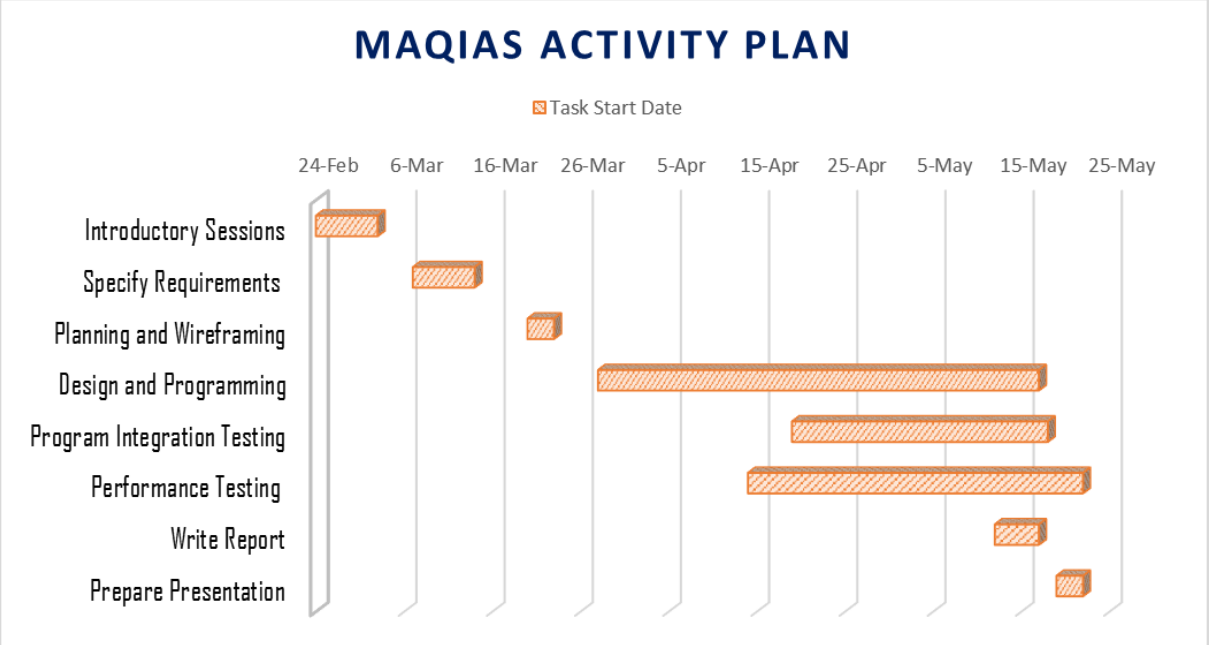

Figure 10: Project Activity Plan

Amazon Web Services (AWS) kick-off session for CIC program took place on the 24th of February, then multiple introductory sessions were conducted as AWS team introduced us to the CIC program and AWS services. After discovering some of AWS services, and the programs needed to ease the procedure of the challenge, 2 solution workshops were scheduled with the sponsor of the challenge/project in order to specify all needed requirements and clarify the project scope. Planning and wireframing sessions were organized with the purpose of determining all services to use and also to have an initial sketch of the dashboard. Next, the group was divided into two subgroups to work on back-end and front-end. Tasks were divided. For more details please see the table below.

Table 3: Project Tasks

Task Name	Duration	Start	Finish
INTRODUCTORY SESSIONS			
Learn by doing session 1	1 day	Sun 2/27/22	Sun 2/27/22
Launch Ready Planning	2 days	Sun 2/27/22	Mon 2/28/22
Resolve Security Token Expiration	1 day	Sun 2/27/22	Sun 2/27/22
Learn by doing session 2	1 day	Tue 3/1/22	Tue 3/1/22
Learn by doing session 3	1 day	Wed 3/2/22	Wed 3/2/22
Specify Requirements			
Solution Workshop 1	1 day	Mon 3/7/22	Mon 3/7/22
User Stories and in/out of scope definition workshop	2 days	Tue 3/8/22	Wed 3/9/22
Solution Workshop 2	1 day	Sun 3/13/22	Sun 3/13/22
Planning and Wireframing			
Review Ready wireframe with Hamad and Rashdan	1 day	Tue 3/22/22	Tue 3/22/22
Discuss Wireframe Changes	1 day	Wed 3/23/22	Wed 3/23/22

Update Changes onto Wireframe	4 days	Wed 3/23/22	Mon 3/28/22
Design, Programming, Testing			
Research how to deploy machine learning model on Jetson Nano	1 day	Thu 3/24/22	Thu 3/24/22
Create New Database and Run Code	1 day	Mon 3/28/22	Mon 3/28/22
Download Arduino program	1 day	Mon 3/28/22	Mon 3/28/22
Pickup sensor and arduino	1 day	Mon 3/28/22	Mon 3/28/22
Conduct a session to review wireframe/best programming language/framework/AQI	1 day	Tue 3/29/22	Tue 3/29/22
Research how to connect arduino with sensor (MQ7)	1 day	Tue 3/29/22	Tue 3/29/22
Deploy Elastic Beanstalk Application	1 day	Wed 3/30/22	Wed 3/30/22
Create and Populate sensorReadings Table	2 days	Wed 3/30/22	Thu 3/31/22
Create and Populate Table for Cities	2 days	Wed 3/30/22	Thu 3/31/22
Create and Populate Table for Intersections	2 days	Wed 3/30/22	Thu 3/31/22
Install TensorFlow on Jetson Nano	12 days	Wed 3/30/22	Thu 4/14/22

Update Dummy Data Generation Code	2 days	Wed 3/30/22	Thu 3/31/22
Write Arduino code to take readings from MQ7	4 days	Wed 3/30/22	Mon 4/4/22
Write Python script that will read from serial boards and display sensor readings	4 days	Wed 3/30/22	Mon 4/4/22
Creating an API to query data from TimeStream	9 days	Sat 4/2/22	Wed 4/13/22
Research + Choose Map API	1 day	Mon 4/4/22	Mon 4/4/22
Implementing Dashboard Layout	6 days	Tue 4/5/22	Tue 4/12/22
Display Map on Dashboard	2 days	Tue 4/5/22	Wed 4/6/22
Add pins on the map	7 days	Thu 4/7/22	Fri 4/15/22
Display Heatmap	7 days	Fri 4/8/22	Sat 4/16/22
Prepare Draft PR/FAQ/Visuals	7 days	Sun 4/10/22	Sat 4/16/22
Implement new Dummy Databases	1 day	Wed 4/13/22	Wed 4/13/22
Implement map intersection pop up	3 days	Thu 4/14/22	Sun 4/17/22
Run the code given in IoT kit and figure out how it works	3 days	Sun 4/17/22	Tue 4/19/22

Deploy object detection machine learning model on Jetson Nano	6 days	Tue 4/19/22	Tue 4/26/22
Integrate sensor code with IoT core	5 days	Tue 4/19/22	Sun 4/24/22
Confirm car type recognition	2 days	Tue 4/26/22	Wed 4/27/22
Deploy and test object tracking machine learning model on Jetson Nano	11 days	Thu 4/28/22	Thu 5/12/22
Decide and implement wait-time calculation	5 days	Sun 5/8/22	Thu 5/12/22
Integrate tracking code with detection code	1 day	Mon 5/9/22	Mon 5/9/22
Write code to calculate AQI average for defined time	2 days	Mon 5/9/22	Tue 5/10/22
Test model using video footage of cars	3 days	Tue 5/10/22	Thu 5/12/22
Test object tracking code	1 day	Tue 5/10/22	Tue 5/10/22
Link CO sensor code + Object tracking + IoT Core connection together	6 days	Tue 5/10/22	Tue 5/17/22
Embed application with Quicksight	6 days	Tue 5/10/22	Tue 5/17/22
Technical Workshop 1	1 day	Fri 5/13/22	Fri 5/13/22

Activate Deeplens and play with it	1 day	Fri 5/13/22	Fri 5/13/22
Integrate back-end and front-end	4 days	Mon 5/16/22	Thu 5/19/22
Integrate Arduino with jetson nano	1 day	Tue 5/17/22	Tue 5/17/22
Connect/Use and actual camera on the Jetson	3 days	Tue 5/17/22	Thu 5/19/22

Chapter 4
Requirement Collection and Analysis

The requirement collection and analysis phase is vital in the software development life cycle. It allows developers to meet the expectations of the customer by setting coherent requirements requested by them. This chapter will discuss the methods used to collect the requirements from the customer. It also discusses the functional and non-functional requirements of the system after their analysis, as well as the data flow and use case diagrams constructed to represent the system.

4.1 Requirement Elicitation

An interview with the customer was conducted on an online platform as the primary requirement collection method. The questions asked helped to better understand the scope of the system.

4.1.1 Interview and analysis

Table 4: Interview outline

Interviewee Dr. Hanan Mubarak Saadalla Albuflasa	**Interviewers** Reema Hehsam Khairalla Yasmeen Omar Albalooshi Zakeya Ahmed Al Saeed Amna Sameer Alawi
Location Microsoft Teams	**Interview Date** 13 March 2022
Questions	**Answers**
1. What type of data would you like us to collect?	Based on what you see fit, but mainly the CO level, and type of cars at the traffic light intersection
2. What information would you like us to include on the dashboard?	A map with pins that displays the AQI for each area and anything else that is applicable
3. Would the data collected be avaiable fot public access?	For now, the solution should only be catered to the SCE manager. However, a puplic version could be implemented in the future
4. Are there currently any stations installed collecting pollution infomation?	Yes, however, they are scattered around Bahrain and not placed near traffic lights which is what this solution is aiming for
5. What type of information is being captured by the stations?	PM 2.5, PM 10, and CO2 emissions

6. Can we use the stations as part of our prototype?	You must get permission from from the Supreme Council for Environment
7. Does the weather have an impact on pollution?	Yes. Weather, humidity and windspeed all have a impact on the level of pollution being generated. Mainly, the higher the temperature, the more pollution detected
8. Does the vehicle type (privat car, bus, truck) contribute to the amount of pollution generated?	Yes. Trucks and busses contribute greatly to the level of pollution
9. Do the make and model of cars contribute to the level of pollution being generated?	Yes. Older cars or sports cars tend to have inefficient fuel consumption which lead to more pollutants being emitted
10. Does the landscape in the area contribute to the level of pollution?	Yes, the greener the area the less pollution there will be
11. Do different areas, industrial or residential, have different pollution levels?	Yes. Industrial areas tend to have more trucks and busses which emit more pollutants compared to normal cars, thus increases the pollution levels

After thoroughly analyzing and studying the customer's interview replies, it has been concluded that the solution will be a dashboard that displays a heat map with pins on every intersection a sensor is deployed. The information being captured and shown will include the AQI calculation after capturing the CO level, the vehicle type (private, bus, truck), the number of cars in the intersection, the wait-time of cars in the intersection, and the type of area (industrial or residential) the sensor is deployed in.

4.2 System Requirements

The interview set clear expectations that were then converted into concrete functional and non-functional requirements.

4.2.1 Functional Requirements

Functional requirements are described as the way the system will behave when certain input criteria are met (ReQtest, 2012).

SCE (Supreme Council for Environment) Manager

1. **See Air Quality Indicator (AQI) based on Carbon Monoxide (CO):** The SCE manager should be able to see AQI differences between each location on a map on a dashboard
2. **See common vehicle types:** The SCE manager should be able to see common vehicle types at each location
3. **See AQI over different times of the day:** The SCE manager should be able to see the AQI over different times of the day
4. **See AQI over different months of the year:** The SCE manager should be able to see AQI over different months of the year

5. **Observe and compare AQI and other parameters between different areas:** The SCE manager should be able to compare and see differences in AQI, common vehicle types, etc. between different locations
6. **Hourly track traffic flow (number of cars) changes in a specific area:** The SCE manager should be able to track traffic flow changes in specific intersections
7. **Sort intersections from highest to lowest pollution:** The SCE manager should be able to sort intersections from highest to lowest AQI
8. **See wait-time in intersections:** The SCE manager should be able to see the wait-time of cars in intersections
9. **See the vehicle types (car, bus, truck) most frequently visiting certain intersections with high pollution:** The SCE manager should be able to see the most common vehicle types passing through high pollution intersections
10. **See the ratio of cars vs. trucks:** The SCE manager should be able to see the number of cars vs. the number of trucks

After an intensive study of the different criteria to include in our solution, it has been decided that certain parameters will be excluded, such as capturing the car's make and model, calculating pollution based on PM 2.5 and PM 10, and viewing weather related data alongside pollution data. The previous parameters were found to be relevant but not the focal point of the current solution.

4.2.2 Non-functional Requirements

Non-functional requirements address all requirements that haven't been included in the functional requirements. They refer to the criteria for judging a system's functionality rather than particular behaviors (ReQtest, 2012).

Table 5: Non-functional requirements

Non-functional requirement	Description
Privacy	The footage of cars collected by the camera will never be shared with anyone, even the SCE manager. Only the captured data will be displayed on the dashboard, which will only be accessible by the manager's log in information
Performance	The system responds quickly and efficiently to user's inputs
Usability	The system is easy to use and navigate for first-time users
Scalability	The system is easy to add more functions to for enhancement
Maintainability	The system will continue functioning normally if certain components are being fixed

4.3 Personas

A persona is a representation of the type of person who would engage with the system. They are fictional characters based on the understanding of real individuals.

Table 6: SCE Manager persona

Persona: SCE Manager	
Photo	
Objectives	• Monitor AQI levels throughout different areas in Bahrain • Make conscious decisions based on the information presented
Frustrations	Using a slow system that hasn't been updated in the hourly supposed time
Scenario	The manager logs in using his user ID and password. He will be presented with a dashboard which includes a heatmap showcasing pins with the AQI's in all intersections with the sensor and camera deployed. Once he clicks on a pin, more information, such as the number of cars and wait-time, will be presented to him. He can also filter the map based on a date range. Based on such information, the manager can make informed decisions to change the structure of roads in Bahrain.

4.4 System Models

4.4.1 Data Flow Diagrams

Data flow diagrams represent how data moves within a system. The following diagrams represent the data flow of the solution at different levels.

- Level 0:

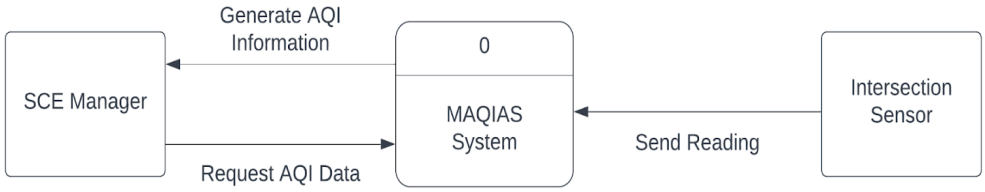

Figure 11: MAQIAS Context Diagram

Figure 11 represents how data flows at level 0.

- Level 1:

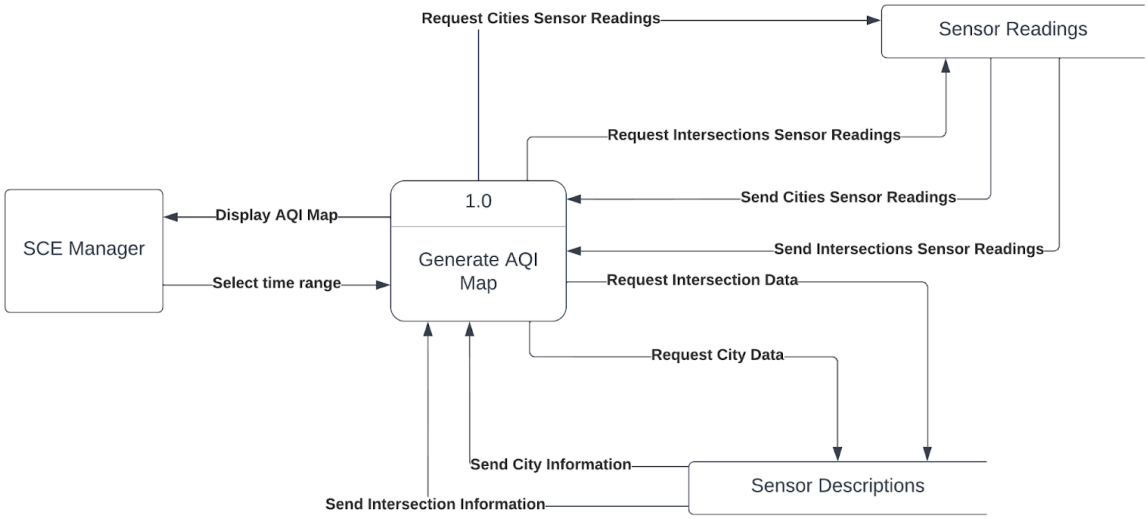

Figure 12: MAQIAS Level 1 Generating AQI map

Figure 12 represents how data flows at level 1 for generating the AQI map.

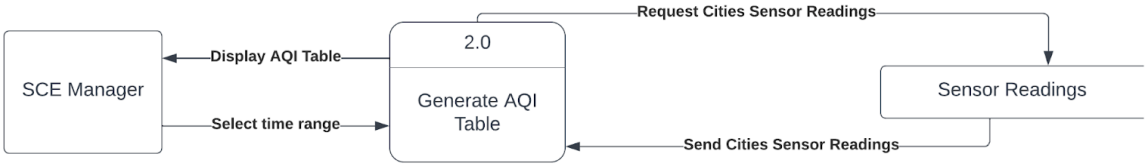

Figure 13: MAQIAS Level 1 Generating AQI Table

Figure 12 represents how data flows at level 1 for generating the AQI Table.

Figure 14: MAQIAS Level 1 Comparing City Information

Figure 14 represents how data flows at level 1 for comparing city information.

- Level 2:

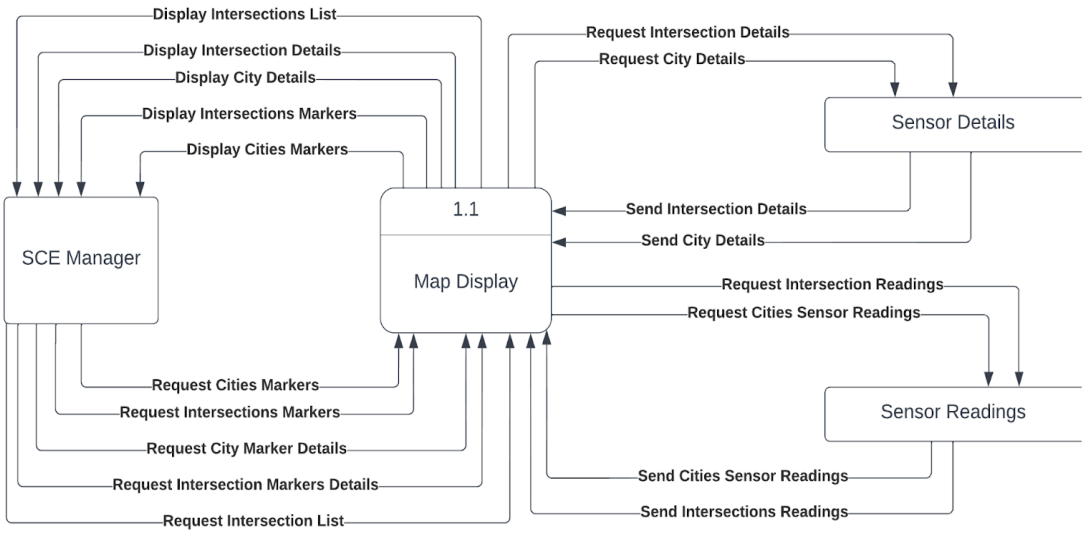

Figure 15: MAQIAS Level 2 Displaying map

Figure 15 represents how data flows at level 2 for displaying the map.

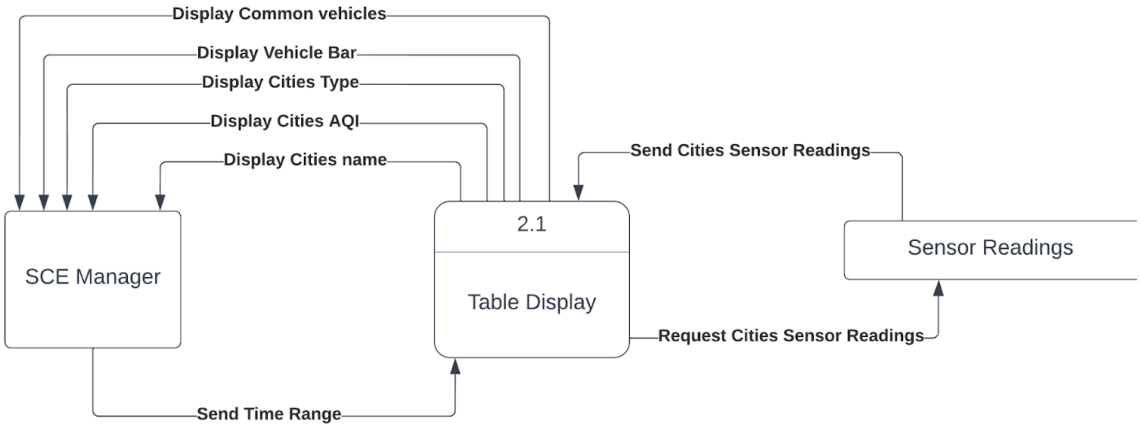

Figure 16: MAQIAS Level 2 Displaying AQI Table

Figure 16 represents how data flows at level 2 for displaying the AQI table.

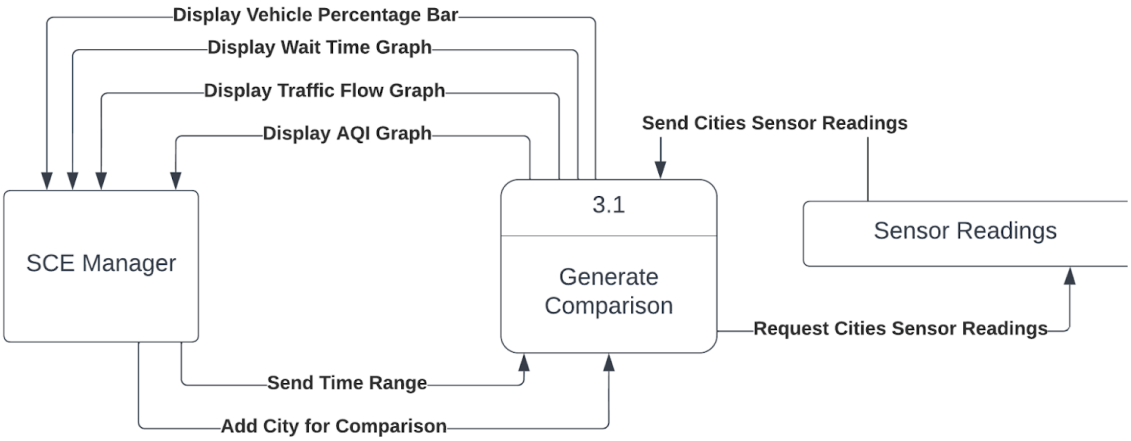

Figure 17: MAQIAS Level 2 Generating City Comparison

Figure 17 represents how data flows at level 2 for generating the city comparison.

Chapter 5
System Design

Multi-tier system design is commonly used architecture particularly for applications requiring user interface. In MAQIAS designing we relied on the three-tier architecture to connect units of the system where we found is the most suitable architecture for our project. Three tire architecture basically divides the system into three tiers; Application tier where it can be referred to as logical tier and here where the data is being processed, Data tier or also database tier is basically the tier where all data is being kept and stored, and finally the Presentation tier which represents the user-interface. The below figure shows how three tier architecture was implemented in our project. The middle service (Timestream) represents the data tier, the left part where the hardware rely is the application tier as in that tier all of our data is being processed. The right side of the figure represents the application tier as the dashboard is created.

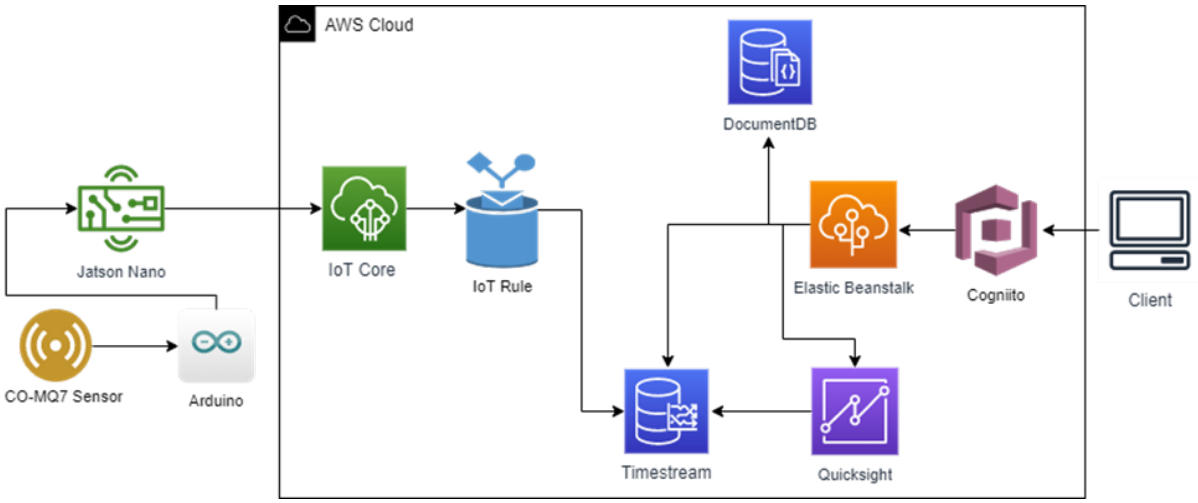

Figure 18: 3 tier architectures

5.1 Database Schema

For the development of this solution, two database services were chosen: AWS Timestream and AWS DocumentDB. Both databases have been chosen to work together during runtime. The Timestream database is a NOSQL database that will contain readings obtained from the sensors. The DocumentDB database is also a NOSQL database. It will contain two collections: a cities collection and an intersections collection. The cities collection will contain information related to each city along with its coordinates. On the other hand, the intersections collection will contain information related to each intersection along with sensors attached to it and its location coordinates. The following diagrams and screenshots have been provided to present the schemas of the databases.

1. DocumentDB
    A. City Collection:

City Collection	
_id	String
cityID	String
location	Object
city	String
cityType	String

Figure 19: City Collection Schema


```
{
        "_id" : ObjectId("625c991ffd0ab54e7b99a712"),
        "cityID" : "0705bc80-e847-4a93-8220-a346719b12b5",
        "city" : "Manama",
        "location" : {
                "coordinates" : [
                        50.5861,
                        26.223
                ],
                "type" : "Point"
        },
        "cityType" : "res"
}
```


Figure 20: City Collection Document Screenshot

B. Intersection Collection:

Intersection Collection	
_id	String
intersectionID	String
location	Object
cityID	String
sensors	Object

Figure 21: Intersection Collection Schema


```
{
        "_id" : ObjectId("625c9850fd0ab54e7b99a70b"),
        "intersectionID" : "2d5299f0-db28-4562-b1b5-011a02843c8b",
        "location" : {
                "coordinates" : [
                        50.58213,
                        26.232337
                ],
                "type" : "Point"
        },
        "cityID" : "0705bc80-e847-4a93-8220-a346719b12b5",
        "sensors" : [
                {
                        "sensorID" : "77675157-46aa-45f7-b12e-62f3bcd1186b",
                        "lastPingTime" : ISODate("2022-04-17T22:44:32.749Z")
                }
        ]
}
```


Figure 22: Intersection Collection Document Screenshot

2. Timestream

Dimensions	TimeStamp	Measure Values
cityType: VARCHAR	DATETIME	busses: INT
city: VARCHAR		cars: INT
intersectionId: VARCHAR		trucks: INT
sensorId: VARCHAR		AQI: INT
		CO: FLOAT
		waittime: INT

Figure 23: Timestream Schema

cityType	city	intersectionId	sensorId	measure_name	time	busses	cars	AQI	trucks	CO	waittime
res	Madinat Hamad	34ef6236-31eb-4bf6-91ac-c8b0adfcf3d4	caa12975-c6b4-478e-bed4-8a9998d7c2d7	dummy_metrics	2021-04-15 10:42:43.610000000	68	85	27	90	2.4	53
res	Madinat Hamad	34ef6236-31eb-4bf6-91ac-c8b0adfcf3d4	caa12975-c6b4-478e-bed4-8a9998d7c2d7	dummy_metrics	2021-04-15 11:42:43.610000000	67	89	45	74	4.0	204
res	Madinat Hamad	34ef6236-31eb-4bf6-91ac-c8b0adfcf3d4	caa12975-c6b4-478e-bed4-8a9998d7c2d7	dummy_metrics	2021-04-15 12:42:43.610000000	78	54	156	72	12.8	115
res	Madinat Hamad	34ef6236-31eb-4bf6-91ac-c8b0adfcf3d4	caa12975-c6b4-478e-bed4-8a9998d7c2d7	dummy_metrics	2021-04-15 13:42:43.610000000	79	59	150	82	12.4	174

Figure 24: Timestream Data Screenshot

5.2 User Interface Design

The wireframes of the solution were designed using Figma. The following figures represent the proposed user interface design.

Figure 25: Map in Initial Dashboard User Interface

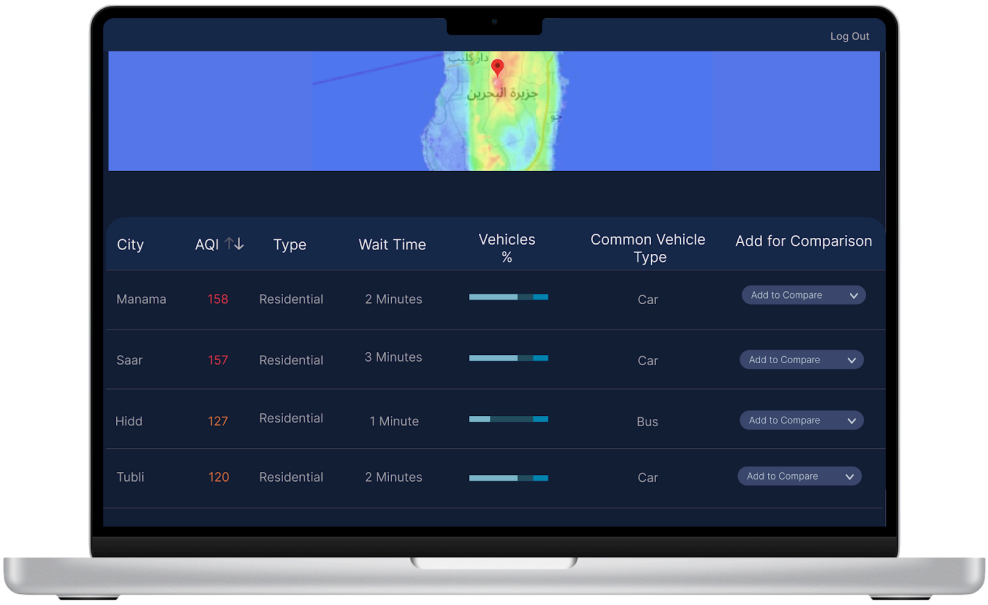

Figure 26: AQI table in Initial Dashboard User Interface

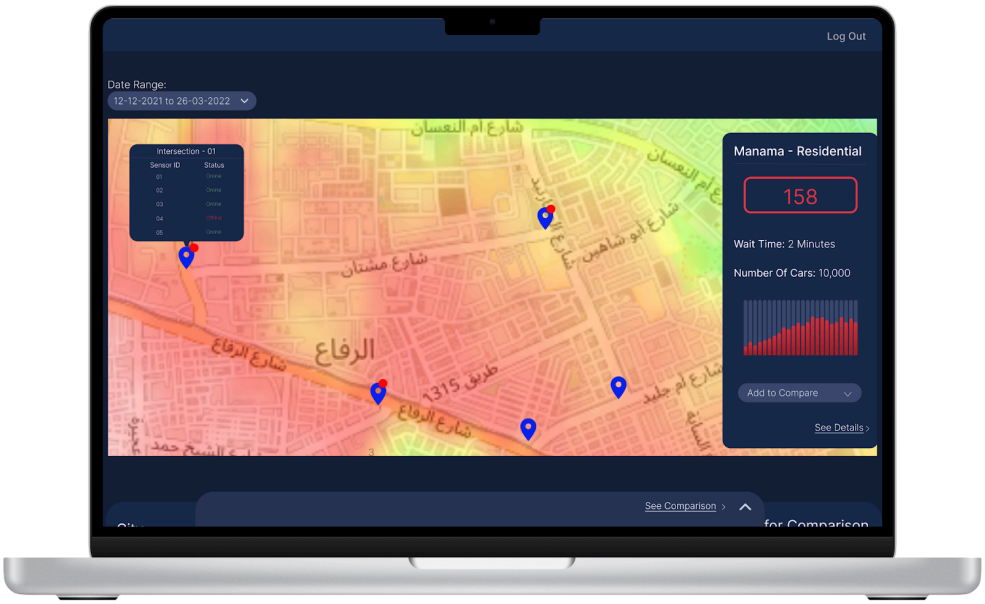

Figure 27: Intersection markers and map side panel

Figure 28: Cities Comparison modal page

Figure 29: city details modal page - AQI and traffic flow graph

Figure 30: city details modal page - Intersections table

Chapter 6
System Implementation and Testing

System implementation is the process of specifying how the information system should be developed, verifying that the information system is operational and being utilized, and ensuring that the information system fulfills quality standards. Testing occurs after the final integrations between the front-end and back-end have been made. It ensures that the system is properly running and displaying the necessary output.

6.1 Software Tools

1. **Visual Studio Code**
   Visual Studio Code, also known as VS Code, is a free and open-source editor. It is compatible on many platforms and operating systems including Linux, macOS, and Windows. It also supports numerous programming languages such as Python, HTML, PHP, GO and many more. VS Code allows you to add extensions such as debuggers which improve user experience (Mustafeez).
2. **Figma**
   Figma is a free, online user interface design application. It helps developers better visualize the system and its flow by creating an interactive design. It runs entirely on a browser which means that projects will be easily accessible anytime, anywhere (themejunkie).
3. **GitHub**
   GitHub is a version control and collaboration tool for programming. It allows people to easily collaborate on projects from any location (GithubDocs).
4. **Linux**
   Linux was used as the primary operating system to run the object detection and tracking algorithms. It is free, open-source, and has many distributions that suit different types of people (Linux). The distribution used for this project is Ubuntu. Linux is also known to have high performance since the file structure in very organized. It allows for quick read-write operations which, as a result, makes the overall performance high and efficient (Soumik, 2021).
5. **Docker Container**
   Machine learning algorithms for detection and tracking tend to heavily consume memory which, as a result, slows down the performance and doesn't run efficiently or quickly on a computer's normal settings. A docker container is used as a solution. It creates an environment where all the code and dependencies are pre-installed which allows for quick and reliable code computation. A Docker container image is a small, independent software package that has everything required to run a program, including code, runtime, system tools, system libraries, and settings (Docker).
6. **PyTorch**
   PyTorch has grown in popularity since its introduction in 2017 by the Facebook AI Research (FAIR) team as a framework for creating efficient Deep Learning (DL) models. This open-source machine learning framework is built on Torch and is intended to give deep neural network and machine learning implementation more flexibility and performance (Simplilearn, 2021).
7. **Tensorflow**

TensorFlow is a machine learning platform that runs from start to finish in an open-source environment. It features a large and flexible array of tools, libraries, and resources that enable academics and developers to push the boundaries of machine learning and to quickly build and deploy machine learning (ML) models. It is simple and easy to implement. It also has the option to build and deploy the ML model in the cloud, in the browser, or on-device (TensorFlow).

8. **AWS IoT Core Service**

   Internet of Things (IoT) Core is a managed cloud service that allow connections of devices securely and easily to interact with the other cloud applications and services or between the other devices. The AWS IoT Core can support trillions of messages along with processing and routing those messages between the AWS endpoints to other billions of devices securely and reliably. The communications with all the devices can be tracked all the time even when not connected. Interconnection between AWS IoT Core and other Amazon's cloud services are easy through IoT Core features from the message broker and rule engine (AWS IoT Core Service, 2021). AWS IoT Core uses X.509 certificate authentication for the client and device connection, which provides several benefits over the other mechanisms for identifications and authentications. The X.509 enables asymmetric keys to be used with the devices. As AWS IoT authentications for the client certificates uses TLS protocol which can be used for encryption of the data, this process is to request the X.509 client certificate and validates the AWS account and the certificate status on the registry of the certificates. Then it will require to proof the ownership of the private key that corresponds to the public key. To facilitate secure connections to servers AWS IoT will require from the client for the send the Server Name Indication (SNI). X.509 certificates verified through a trust certificate authority (CA), which the client can create the certificates with Amazon Root CA. (AWS IoT Core Service, 2021)

   - **AWS IoT Device SDK**

     In order to connect and use AWS IoT Core it will require to use AWS IoT Device SDK to enable easy connect, authenticate, exchange messages by using WebSockets, MQTT, or HTTP protocols. The AWS IoT Device SDK can be programmed with the support of programming languages C, JavaScript, or Python (AWS IoT Core Service, 2021). In MAQIAS, IoT Core service was utilized by connceting the back-end output (average AQI and CO, total number of vehicles, vehicles type and wait time calculation) to AWS by enabling exchanging MQTT messages.

9. **AWS IoT Rule Service**

   IoT rule service is a simple service that belongs to IoT core, its main functionality is to receive whatever data either from devices or from other AWS services, and it resends the data to other AWS services. It specifically receives MQTT messages and its re-format those messages to be in a suitable format specified by the user. The user must define an applicable IoT rule action that suits their system design in order to direct the data to the next wanted AWS service. In MAQIAS, IoT rule was utilized to connect the back-end and the front-end. It starts listening to MQTT messages sent from IoT core, these messages represent the hardware calculated output, and then it utilizes the timestream action which in turn sends the messages to the database created in front-end every hour, so the data in the database is going to updated hourly. Timestream function was chosen because it is the most suitable amazon service to store and analyze sensor reading and data and the mentioned function.

10. **AWS Timestream**

    Timestream is a NoSQL database service provided by AWS focused on managing time series data. It is serverless and as such can scale up or down without having to manage underlying infrastructure. The advantages of using Timestream instead of a more traditional database is that Timestream are that it provides high performance, allows for simple data access, does not need servers to manage or capacities to provision, is encrypted, and most importantly is built for time series data. Use cases of Timestream include IoT applications such as temperature collection sensors applications, DevOps applications such as applications that measure CPU/memory utilization over time, and analytics applications such as web traffic analysis applications (AWS, 2022).

11. **AWS DocumentDB**

    DocumentDB is a database service provided by AWS for operating MongoDB workloads. It is a type of NoSQL databases and as such allows for the insertion, querying, and indexing over JSON formatted data(Amazon DocumentDB, 2022). The database service runs in an Amazon Virtual Private Cloud and is able to be isolated in the virtual network. Its' firewall settings can also be configured to control network access to the database. It works by providing clusters that can hold upto 16 instances that all support reads. The data of the cluster is stored within the cluster volume which has several copies in different availability zones. . DocumentDB allows for the querying of Geospatial data. It uses the 'Point' type used for GeoJSON to store the geospatial data (AWS, 2022).

12. **AWS Cognito**

    Cognito is provided by AWS which allows for user sign up, sign in and access to web and mobile applications. It provides a scalable identity store, supports identity and access management standards such as OpenID Connect, allows multi-factor authentication, encryption of data at rest and in transit, abd provides access control to AWS resources (AWS, 2022).

13. **AWS Quicksite**

    Quicksight is a serverless business intelligence service provided by AWS. It allows for the creation of interactive dashboards and automatically analysis patterns with machine learning. Analytics performed by Quicksight can be embedded into applications. It connects to data within AWS and creates complex models from them. It has built in end to end data encryption (AWS, 2022).

14. **AWS Elastic Beanstalk**

    Elastic Beanstalk is a service provided by AWS for deploying web applications developed using Java, .NET, PHP, Node.Js, Python, Ruby, Go, and Docker. It allows for upload application code to AWS and it will automatically handle the deployment of the application while still allowing access to the underlying AWS resources at any time. It handles provisioning of capacities, load balancing, and autoscaling the application (AWS, 2022).

15. **OpenLayers**

    OpenLayers is an opensource JavaScript library that focuses on displaying map data in applications. It allows for displaying map tiles, vector data, and markers onto the map. OpenLayers provides an API for access from applications. It supports adding map data in many formats including GeoJSON, Keyhole Markup Language, and GeoRSS (OpenLayers - Welcome, 2022).

6.2 Hardware Tools

1. **Jetson Nano**
   Jetson Nano is a Nvidia-owned compact, powerful computer that allows you to run many neural networks in parallel for image classification and object detection amongst other uses. It is all contained within a user-friendly platform that consumes only 5 watts of power (Nvidia).
2. **Arduino**
   Arduino is a hardware board with an open-source that reads inputs from sensors, button inputs to be processed to an output as motor activation and controlling, online publishing. Arduino is a microcontroller platform available for physical computing which can utilize this functionality with set of tools and libraries through programming wrapped in easy-to-use packages. Arduino selected for it simple and accessible user experience and flexible for advance usage. This microcontroller is a cross-platform supported with IDE that can be run on all operation systems Windows, Macintosh OSX, and Linux unlink most microcontroller systems are strained and limited to Windows. Due it is relativity as inexpensive microcontroller compared to the other microcontroller platforms and extensibility on the software and hardware levels makes it the best option for programming environment to use it with air sensor conveniently for this project. The source tools can be expanded through C++ libraries and leap from the sensors to the Arduino via AVR C which have a role of a compiler, creates a binary source from a coded with high level C language code which can be uploaded into Arduino board microcontroller (Arduino, 2018).
3. **MQ7 Sensor**
   A sensor is a piece of hardware that detects specific physical type of event or input which additionally generates an analog signal output of what the measures that sensed. The physical input could be light, temperature, frequency or even gas atoms. MQ7 sensor which is known also as Carbon Monoxide (CO) Sensor is a sensing device that is configured to operate a certain functionality; measuring and sensing CO concentration in the air. It has the ability of sensing CO-gas concentration in parts per million (ppm) which represents the mass of a synthetic or pollution for every unit volume of water. MQ7 considered as a simple and easy-to-use device as it requires an uncomplicated electro circuit connection. Besides, it measures CO anywhere from 10 to 500ppm at temperature from -10 to 50°C. This sensor consumes less than 150 mA at 5 V which weighed as very low voltage consumption. One advantage of MQ7 sensor that it has long life at low cost and it is applicable for various applications (Circuits DIY, 2021).

6.3 Programing Languages

1. **Python**
   Python programming language was released in 1991. It is amongst the most common languages in programming field. Python is considered as a powerful high-level language. It is used to develop many applications and specially software and web development. Python utilizes many operating systems such as UNIX, MAC OS, windows and others. Python is an easy-to-use language. Moreover, it is a beginner-friendly language. It contains many features such as it is an object-oriented, open-source, and fully supported programming language. Currently, Python is one of the mostly used programming language in the world as it has numerous numbers of libraries and module. Python is being utilized in so many technologies starting from web development and all the way to machine learning (TechTarget, 2021). In our project, we used python as the main programming language in all of our codes both in

back-end for tracking, detection, sensor reading and in the front-end in developing the dashboard.

2. **C**
   C programming language was firstly created in 1972 by Dennis Ritchie. It was developed from other several programming languages such as ALGOL and BCPL. It is considered as a robust programming language that utilize UNIX operating system. 'C' is considered as a base for various different languages that are as of now being used, for example, C++ and Java are created from 'C'. These languages are broadly utilized in different advancements. C programming language is simple, easy-to-use, and pliable language. It is also a well-known language in the programming field. It is fundamental to be familiar with PC memory systems and mechanisms since it is a significant perspective while managing the C programming language. C is used in many technology applications, for example it is broadly used in IoT applications and compilers productions (Guru99, 2022). In this project, C programming language was utilized to connect the Arduino microcontroller with the MQ7 sensor through Arduino program, which results in displaying the CO sensor reading in ppm.

6.4 Back-end Components and Integration

6.4.1 Object Detection and Tracking

Object detection is the process of identifying and classifying a specific object in an image or a single video frame. It is mainly achieved by using deep learning methods. The two most common are creating and training a custom object detection model or using a pretrained object detection model. In this solution, the latter has been selected and implemented because the model has previously been trained on numerous photos and videos, thus providing faster results compared to training a model from start to end (MathWorks).
Object tracking is the process of locating objects in a video frame and creating a unique identifier for each of them, then continue to track the located objects as they move frame by frame (Meel). The steps included in this process are drawing bounding boxes to outline each object, appearance modeling that allows the tracker to continue tracking the object even after certain environmental changes or distortions occur such as change in lighting or angles, motion estimation which refers to the model's ability to approximately anticipate the object's next position, and target positioning which finds the exact location of the object after it has been approximated by motion estimation (Barla, 2022).
The first approach taken in the object detection and tracking solution was to use Amazon's Deeplens. Deeplens is the world's first camera with deep learning capabilities. It provides a programmable camera and pre-trained deep learning models that are ready to use (AWS). However, it was quickly excluded as an option since many errors occurred during the setting up and registration process which made it difficult to proceed. It also had many compatibility issues. The next approach was to use the Jetson Nano alongside a pre-trained object detection and tracking algorithm. The Jetson Nano is an AI optimized board computer. It was chosen after finding a source that lists the Jetson Pretrained Model Benchmarks for object detection algorithms and compatible frameworks (https://developer.nvidia.com/embedded/jetson-benchmarks) made by Nvidia. It is important to note that since finding this benchmark, Nvidia has changed its documentation numerous times by removing and adding new benchmarks which affected the flow of the progress, especially when initially deciding to use TensorFlow as our main framework. Nvidia has also discontinued the production of Jetson Nano. This means that our solution was entirely created and run on a device that is discontinued and discredited.

Before Nvidia removed TensorFlow as an acceptable framework to be used on the Jetson Nano, we decided that TensorFlow is the next best approach in terms of framework as it features a large and flexible array of tools, libraries, and resources that quickly build and deploy machine learning models. However, many issues were faced when trying to implement TensorFlow in our solution, thus it was decided that it will not be used as the primary framework. Next, we decided to utilize TensorFlow Hub which showcased many ready-made object detection models. However, when running the model, it appeared to be very slow as it is not optimized to run on edge devices (Jetson Nano). All previous approaches have either failed due to incompatibility issues or were very slow due to the model running on the host device's CPU instead of GPU.
The final and chosen approach was as follows: A PyTorch docker container that also has TensorFlow installed in it was created. Tensorflow was necessary to find the features of every object during the object tracking process. Docker containers are based by Nvidia, which means that they are designed in a way that accesses the GPU of the host device which greatly accelerated the object detection and tracking process.

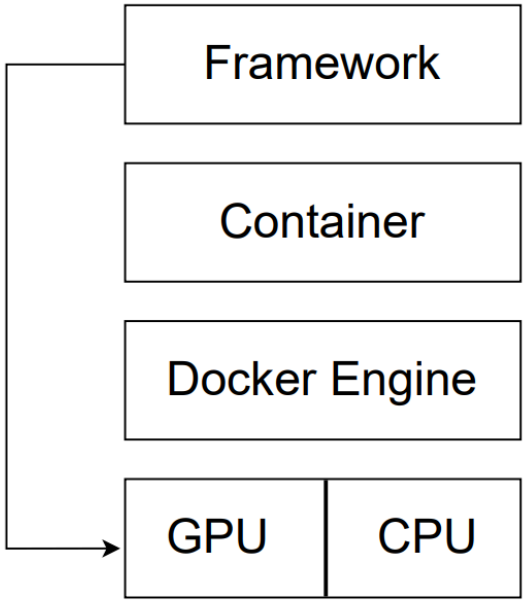

Figure 31: Docker Container

The container included all libraries and frameworks (PyTorch and TensorFlow) to access the GPU for optimized detection and tracking.
Object detection was done using YOLOv5 which is an object detection model based on the PyTorch framework and trained on the COCO dataset, which is a huge collection of labeled data to feed and train an object detection model on. YOLO works by dividing photos or video frames into a grid, each grid cell identifying a single item. To determine the accuracy of each prediction, the grid cells anticipate all the bounding boxes and assign a confidence score to each one. It is fast, efficient, and significantly outperforms other object detectors. It can process pictures at roughly 155 frames per second (FPS), reaching double the FPS of other object detectors (Fredrick, 2021).
Object tracking was done using the DeepSORT algorithm. DeepSORT is a very fast and intuitive algorithm that works by using the detections captured by the object detection model (YOLOv5) for every frame and tries to match it with the detections captured in the previous frame (Cohen, 2019).

6.4.1.1 Object Detection and Tracking Codes and Algorithms

- Object detection and tracking code:

```
import time
import cv2
import numpy as np
import serial
from elements.yolo import OBJ_DETECTION
# deep sort imports
from deep_sort import nn_matching
from deep_sort.detection import Detection
from deep_sort.tracker import Tracker
from application_util import preprocessing
from application_util import visualization
from tools import generate_detections as gdet
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
from scipy.optimize import linear_sum_assignment as linear_assignment

#OBJECT TRACKING AND DETECTION

Object_classes = ['person', 'bicycle', 'car', 'motorcycle', 'airplane',
'bus', 'train', 'truck', 'boat', 'traffic light',
                'fire hydrant', 'stop sign', 'parking meter', 'bench',
'bird', 'cat', 'dog', 'horse', 'sheep', 'cow',
                'elephant', 'bear', 'zebra', 'giraffe', 'backpack',
'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee',
                'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat',
'baseball glove', 'skateboard', 'surfboard',
                'tennis racket', 'bottle', 'wine glass', 'cup', 'fork',
'knife', 'spoon', 'bowl', 'banana', 'apple',
                'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog',
'pizza', 'donut', 'cake', 'chair', 'couch',
                'potted plant', 'bed', 'dining table', 'toilet', 'tv',
'laptop', 'mouse', 'remote', 'keyboard', 'cell phone',
                'microwave', 'oven', 'toaster', 'sink', 'refrigerator',
'book', 'clock', 'vase', 'scissors', 'teddy bear',
                'hair drier', 'toothbrush']

Object_colors = list(np.random.rand(80,3)*255)
Object_detector = OBJ_DETECTION('/home/JetsonYolo/weights/yolov5s.pt',
Object_classes)

# Definition of the parameters
max_cosine_distance = 0.4
nn_budget = None
nms_max_overlap = 1.0
# initialize deep sort
# calculate cosine distance metric
model_filename = '/home/JetsonYolo/model_data/mars-small128.pb'
encoder = gdet.create_box_encoder(model_filename, batch_size=1)
metric = nn_matching.NearestNeighborDistanceMetric("cosine",
max_cosine_distance, nn_budget)
# initialize tracker
```

```
tracker = Tracker(metric)

# begin video capture
video_path="/home/JetsonYolo/cars.mp4"
try:
    vid = cv2.VideoCapture(int(video_path))
except:
    vid = cv2.VideoCapture(video_path)
out = None

print('start object detection')
allowed_classes = ['car', 'motorbike', 'bus', 'truck']

fpsCounter = 0
fpsSum = 0
total_start_time = 0
start = True
wait_frame_count = {}
WarmUpCount = 0
fpscv2 = vid.get(cv2.CAP_PROP_FPS)

# To flip the image, modify the flip_method parameter (0 and 2 are the most
common)
while vid.isOpened():
    return_value, frame = vid.read()
    if return_value:
        start_time = time.time()
        detections = Object_detector.detect(frame)
        if WarmUpCount < 6:
            WarmUpCount += 1
            continue
        if start:
            total_start_time = time.time()
            start = False
        boxes = []
        labels = []
        scores = []

        for obj in detections:
            if obj['label'] in allowed_classes and obj['score']>0.5:
                [(xmin,ymin),(xmax,ymax)] = obj['bbox']
                h = ymax - ymin
                w = xmax - xmin
                boxes.append((xmin,ymin, w, h))
                labels.append(obj['label'])
                scores.append(obj['score'])

        boxes = np.array(boxes)
        labels = np.array(labels)
        scores = np.array(scores)

        # Start non max supression
        features = encoder(frame, boxes)
        detections = [Detection(box, score, label, feature) for box, score,
label, feature in zip(boxes, scores, labels, features)]
```

```
        boxs = np.array([d.tlwh for d in detections])
        scores = np.array([d.confidence for d in detections])
       # run non-maxima supression
        indices = preprocessing.non_max_suppression(boxes, nms_max_overlap,
scores)
        detections = np.array([detections[i] for i in indices])
        # Call the tracker
        tracker.predict()
        tracker.update(detections)

        # update tracks
        cars = 0
        trucks = 0
        busses = 0
        print('Objects being tracked:
{}'.format(str(len(tracker.tracks))))
        for track in tracker.tracks:
            if not track.is_confirmed() or track.time_since_update > 1:
                continue
            bbox = track.to_tlbr()
            class_name = track.get_class()
            if class_name == 'car':
                cars += 1
            elif class_name == 'truck':
                trucks += 1
            elif class_name == 'bus':
                busses += 1

            #counting frames for each track
            trackID=str(track.track_id)
            if trackID in wait_frame_count.keys():
                wait_frame_count[trackID] += 1
            else:
                wait_frame_count[trackID] = 1


            # output tracker information
            print("Tracker ID: {}, Class: {},  BBox Coords (xmin, ymin,
xmax, ymax): {}".format(str(track.track_id), class_name, (int(bbox[0]),
int(bbox[1]), int(bbox[2]), int(bbox[3]))))
            # visualize
            color = Object_colors[int(track.track_id) % len(Object_colors)]
            cv2.rectangle(frame, (int(bbox[0]), int(bbox[1])),
(int(bbox[2]), int(bbox[3])), color, 2)
            cv2.rectangle(frame, (int(bbox[0]), int(bbox[1]-30)),
(int(bbox[0])+(len(class_name)+len(str(track.track_id)))*17, int(bbox[1])),
color, -1)
            cv2.putText(frame, class_name + "-" +
str(track.track_id),(int(bbox[0]), int(bbox[1]-10)),0, 0.75,
(255,255,255),2)

        print('At the current frame: {} cars, {} busses, {}
trucks'.format(cars, busses, trucks))
        cv2.imshow("output", frame)
```

```
            # frame = cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), color,
2)
            # frame = cv2.putText(frame, f'{label} ({str(score)})',
(xmin,ymin), cv2.FONT_HERSHEY_SIMPLEX , 0.75, color, 1, cv2.LINE_AA)
        fps = 1.0 / (time.time() - start_time)
        fpsSum += fps
        fpsCounter += 1
        print("FPS: %.2f" % fps)
        if cv2.waitKey(1) & 0xFF == ord('q'): break
    # timing in seconds
    elif time.time() - total_start_time  > 60:
        current = time.time()
        # Duration in minutes
        diff = (current - total_start_time) / 60
        fps_average = fpsSum / fpsCounter
        print('FPS average for {} min = {} fps'.format(str(diff),
str(fps_average)))

        print(wait_frame_count)
        wait_time_count = {}
        for k in wait_frame_count:
            waittime = wait_frame_count[k] / 29.73
            if waittime > 0:
                wait_time_count[k]  = waittime
        print(wait_time_count)

        # average wait time
        average_wait_time = sum(wait_time_count.values()) /
len(wait_time_count)
        print('The average wait time is:
{}'.format(str(average_wait_time)))
        print(f"fps: {fpscv2}")
        break
    else:
        print('Restarting the video')
        # break
        vid = cv2.VideoCapture(video_path)

vid.release()
cv2.destroyAllWindows()
```

- Docker file code:

```
FROM nvcr.io/nvidia/l4t-pytorch:r32.6.1-pth1.9-py3
RUN apt-get update -y
RUN apt-get install libhdf5-serial-dev hdf5-tools libhdf5-dev zlib1g-dev
zip libjpeg8-dev liblapack-dev libblas-dev gfortran -y
RUN pip3 install -U  pip testresources setuptools==49.6.0
RUN pip3 install requests
RUN pip3 install tqdm
RUN pip3 install PyYaml
RUN pip3 install matplotlib
RUN pip3 install opencv-python
RUN pip3 install scipy
```

```
RUN pip3 install pandas
RUN pip3 install -U --no-deps future==0.18.2 mock==3.0.5
keras_preprocessing==1.1.2 keras_applications==1.0.8 gast==0.4.0 protobuf
pybind11 cython pkgconfig
RUN apt-get install pkg-config -y
RUN env H5PY_SETUP_REQUIRES=0 pip3 install -U h5py==3.1.0
RUN pip3 install --pre --extra-index-url
https://developer.download.nvidia.com/compute/redist/jp/v46 tensorflow
RUN apt-get install python3-pyqt5 -y
RUN pip3 install pyserial
CMD ["python3", "/home/JetsonYolo/JetsonYolo.py"]
```

All the dependencies required by TensorFlow, PyTorch, YOLOv5, and DeepSORT have been included in the docker file and will run automatically whenever the docker run script is initiated. This will save time and ensure all the dependencies needed by the algorithms and frameworks are installed.

- Docker run script:

```
#!/bin/bash
sudo docker build -t detector .
sudo xhost +
sudo docker run -e DISPLAY=$DISPLAY -v /tmp/.X11-unix:/tmp/.X11-unix -it --
rm --runtime nvidia --network host -v
/home/trafficpollution/repo/SCEPollution:/home detector
```

Dockers do not have graphical user interface capabilities, which became an issue since the visualization of the object detection and tracking was necessary. Xhost was used to solve this issue. It moves images and videos from the container to the host machine to be able to see it.

6.4.2 Sensor and Arduino

6.4.2.1 AQI Calculation

Air Quality Index (AQI) is created by United States Environmental Protection Agency (U.S EPA) and the purpose of it is to state and report how pure, clean and polluted the air in a specific area is. The AQI is determined for five significant air pollutant managed by the Clean Air Act which are Carbon Monoxide (CO), Sulfur Dioxide (SO2), Ground-level Ozone, Particulate matter (PM), and Nitrogen Dioxide (NO2). EPA has laid out public air quality principles for each of the pollutants mentioned earlier to save general human wellbeing. Ground-level ozone and airborne particles are the top two contaminations that represent the highest danger to human wellbeing. The table in **the figure below** presents values of AQI for each pollutant and health breakdowns. According to EPA, the increase in AQI value means the higher the pollution is which results in greater health concerns. EPA divided AQI values into 6 categories of colors and each color represent a certain level of health concern. Good category that appears in the green color is when there Is no health risks and the value of AQI is between 0 and 50, Moderate in yellow represents AQI between 51 to 100 and it indicates medium health concerns for some individuals, Unhealthy for Sensitive Groups in the color orange means that some individuals of sensitive group may be affected with risks of health concerns with AQI values from 101 to 150, The color red specify Unhealthy category with

AQI from 151 to 200 and that category means that all individuals might be pollution health effected, Very Unhealthy is represented by the purple color from 201 t0 300 AQI values and this category triggers an alert because everyone might face serious health problems, and finally the Hazardous category in the maroon color and AQI values greater than 300 and it triggers wellbeing alerts of crisis conditions (Jagran Josh, 2019).

O_3 (ppb)	O_3 (ppb)	$PM_{2.5}$ (µg/m^3)	PM_{10} (µg/m^3)	CO (ppm)	SO_2 (ppb)	NO_2 (ppb)	AQI	AQI
$C_{low} - C_{high}$ (avg)	$C_{low} - C_{high}$ (avg)	$C_{low} - C_{high}$ (avg)	$C_{low} - C_{high}$ (avg)	$C_{low} - C_{high}$ (avg)	$C_{low} - C_{high}$ (avg)	$C_{low} - C_{high}$ (avg)	$I_{low} - I_{high}$	Category
0–54 (8-hr)	—	0.0–12.0 (24-hr)	0–54 (24-hr)	0.0–4.4 (8-hr)	0–35 (1-hr)	0–53 (1-hr)	0–50	Good
55–70 (8-hr)	—	12.1–35.4 (24-hr)	55–154 (24-hr)	4.5–9.4 (8-hr)	36–75 (1-hr)	54–100 (1-hr)	51–100	Moderate
71–85 (8-hr)	125–164 (1-hr)	35.5–55.4 (24-hr)	155–254 (24-hr)	9.5–12.4 (8-hr)	76–185 (1-hr)	101–360 (1-hr)	101–150	Unhealthy for Sensitive Groups
86–105 (8-hr)	165–204 (1-hr)	55.5–150.4 (24-hr)	255–354 (24-hr)	12.5–15.4 (8-hr)	186–304 (1-hr)	361–649 (1-hr)	151–200	Unhealthy
106–200 (8-hr)	205–404 (1-hr)	150.5–250.4 (24-hr)	355–424 (24-hr)	15.5–30.4 (8-hr)	305–604 (24-hr)	650–1249 (1-hr)	201–300	Very Unhealthy
—	405–504 (1-hr)	250.5–350.4 (24-hr)	425–504 (24-hr)	30.5–40.4 (8-hr)	605–804 (24-hr)	1250–1649 (1-hr)	301–400	Hazardous
—	505–604 (1-hr)	350.5–500.4 (24-hr)	505–604 (24-hr)	40.5–50.4 (8-hr)	805–1004 (24-hr)	1650–2049 (1-hr)	401–500	

Figure 32: Table on air quality principles

In this project, AQI calculation was depended on CO value only as MQ7-CO sensor was used. The following formula was utilized to basically convert from concentration and compute the value of AQI:

$$I = \frac{I_{high} - I_{low}}{C_{high} - C_{low}}(C - C_{low}) + I_{low}$$

Figure 33: AQI Calculation

where I indicates the air quality index, pollutant (CO) is indicated by C, I_{LOW} represents index breakpoint corresponding to C_{LOW}, also, I_{HIGH} represents index breakpoint corresponding to C_{HIGH} , C_{LOW} is the concentration breakpoint that is less than or equal C and C_{HIGH} is the concentration breakpoint that is greater than or equal C (Prana Air, 2021).

6.4.2.2 Arduino and Sensor Connection Code

For the connection between the Arduino and MQ7-CO sensor the circuit in the figure below was applied. (Kookye, 2018)

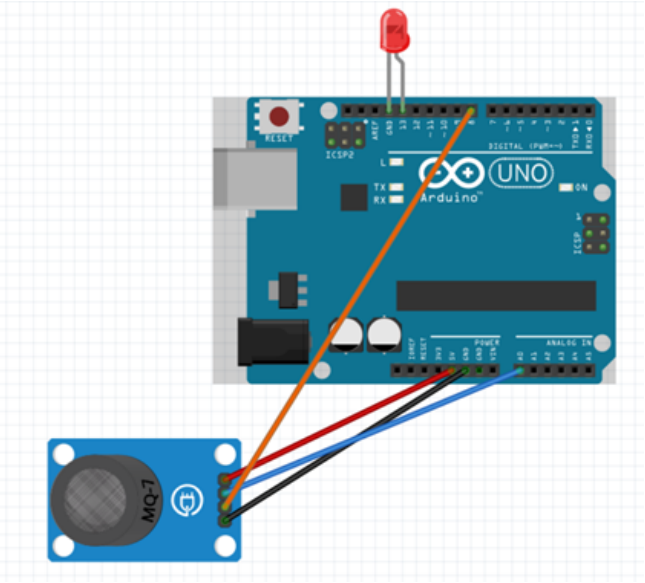

Figure 34: Arduino and MQ7 connection

- Displaying MQ7 Sensor Reading:

```
float sensor_volt;
float RS_gas;
float R0;
int R2 = 2000;
void setup() {
 Serial.begin(115200);
}
void loop() {
  int sensorValue = analogRead(A0);
  sensor_volt=(float)sensorValue/1024*5.0;
  RS_gas = ((5.0 * R2)/sensor_volt) - R2;
  R0 = RS_gas / 1;
  Serial.print("R0: ");
  Serial.println(R0);
}
o   Converting sensor reading to ppm code:
float RS_gas = 0;
float ratio = 0;
float sensorValue = 0;
float sensor_volt = 0;
float R0 = 32711.0;
void setup() {
 Serial.begin(115200);
}
void loop() {
   sensorValue = analogRead(A0);
   sensor_volt = sensorValue/1024*5.0;
   RS_gas = (5.0-sensor_volt)/sensor_volt;
   ratio = RS_gas/R0; //Replace R0 with the value found using the sketch
above
   float x = 1538.46 * ratio;
   float ppm = pow(x,-1.709);
   Serial.print("PPM: ");
   Serial.println(ppm);
```

```
   delay(1000);
        }
```

The Arduino code was implemented to test the reading and the formula for CO measurement. The reading was displayed through the serial monitor tool in the Arduino IDE. The testing was intended to be before implementing the python script to to read directly from the Arduino's serial port and to be compared with the python readings output.

- Python script was written which reads from serial port in arduino. It displays average CO readings and average AQI calculation every hour:

```
# Importing Libraries
import serial
import time

# Set arduino serial COM3
arduino = serial.Serial(port='COM3', baudrate=115200, timeout=.1)

# Function to read from the arduino
def write_read():
    data = arduino.readline()
    return data

# AQI Function to calculate the Air Quality
def gen_aqi(co):
    if co<=4.4:
        Ih=50
        Il=0
        BPh=4.4
        BPl=0
    elif co<=9.4:
        Ih=100
        Il=51
        BPh=9.4
        BPl=4.5
    elif co<=12.4:
        Ih=150
        Il=101
        BPh=12.4
        BPl=9.5
    elif co<=15.4:
        Ih=200
        Il=151
        BPh=15.4
        BPl=12.5
    elif co<=30.4:
        Ih=300
        Il=201
        BPh=30.4
        BPl=15.5
    elif co<=40.4:
        Ih=400
        Il=301
```

```
        BPh=40.4
        BPl=30.5
    else:
        Ih=500
        Il=401
        BPh=50.4
        BPl=40.5
    aqi=int(((Ih-Il)/(BPh-BPl))*(co-BPl)+Il)
    return aqi

# Declaring variables for the calculation
count = 0
start = time.time()
stop = time.time()
sum = 0.0
avg = 0.0
Zeros = False
AQI = ''

# Infinite loop for the continuous reading and calculation
while True:
    # Save the arduino read value and convert from Byte to String
    value = write_read().decode('UTF-8')
    # if the value null consider it as Zero
    if value == '':
        value = "0"
        Zeros = True
    else:
        Zeros = False
    # Extract the digit values into one String
    emp_str = ""
    for m in value:
        if m.isdigit():
            emp_str = emp_str + m

    # Clear the string value from newline character
    valueclear = emp_str.strip('\n')
    valueclear = valueclear.strip('\r')
    valueclear = valueclear.strip('')
    valueclear = valueclear.rstrip('\r')
    valueclear = valueclear.rstrip('\n')
    valueclear = valueclear.rstrip('')

    # Convert string to float and divide it by 100
    valuenum = int(float(valueclear.rstrip()))/100

    # Check the value if it is Zero or non-Zero
    if Zeros == False:

        # Checking if the time didn't reached the defined period
        if (stop - start) < 3600:

            # Get the sum, update the end-time, increament the count
            sum = sum + valuenum
            stop = time.time()
```

```
            count = count + 1
        else:

            # if the time reached the period, get the avarage and update
start-time and end-time
            avg = sum / count
            start = time.time()
            stop = start
        print("Time: ", stop - start)
        if avg == 0.0:
            continue
        else:
            print("Avg CO: ", avg) # printing the value
            print("AQI Avg: ", gen_aqi(avg))
            avg = 0.0
            sum = 0.0
            count = 0
    else:

        # Update end-time to skip the Zero values
        stop = time.time()
```

6.5 Front-end Components and Integration

6.5.1 Codes and Algorithms

- AWS Timestream database connection code:
  The following Python code was written to connect programmatically through python to the AWS Timestream database. The function takes a query as a string that will be sent to AWS Timestream. The response returned by the call is then changed into a format that is compatible with GeoJSON. Lastly, the function returns the variable compatible with GeoJSON.

```
import boto3
client = boto3.client('timestream-query', region_name='us-east-1' )
def mainTimestreamQueryCall(query): #Used to receive timestream Query
results
   response = client.query(QueryString = query)
   rqst=GeoJSONDataCreation(response)
   ts_json= json.dumps(rqst)
   return ts_json
cityQuery='Select city,  ROUND(avg(AQI),0) as averageAQI,cityType,
ROUND(avg(waittime),0) as averageWaittime, sum(cars+busses+trucks) as
sumOfVehicles, sum(busses) as sumOfBusses,sum(trucks) as
sumOfTrucks,sum(cars) as sumOfCars from
dummySensorDB."sensorReadings" Group by city,cityType ORDER BY
averageAQI DESC'
      ts_query = mainTimestreamQueryCall(cityQuery)
```

- AWS DocumentDB connection code:
  The following Python code was written to connect programmatically to AWS DocumentDB and retrieve a document that contains the dictionary passed in the function's parameters.

```
import pymongo
def docDBQuery(queryDictionary,collectionname): #queryCall for
documentDB, takes dictionary of {"key":"value"} as input along with
collection name
cl=pymongo.MongoClient('mongodb://docDBUser:12345678@trafficpollution
dbreg.cluster-cciqodtzbuum.us-east-
1.docdb.amazonaws.com:27017/?replicaSet=rs0&readPreference=secondaryP
referred&retryWrites=false')
   db = cl.test
   col = db[collectionname]
   x = col.find_one(queryDictionary)
         return x
```

- Displaying map code:
  The following JavaScript code snippet renders a base map onto the dashboard. The map is displayed using the OpenLayers API. Firstly, a map view is created which contains coordinates of Bahrain such that when the map initially renders it will be centered at Bahrain. In addition to that, a zoom limit of 11 has been placed to avoid having the user zoom out beyond the scope. Then the OpenLayers API is used to display the map with the selected view.

```
function initmap(){
 //render base map
 view =new ol.View({
 center: ol.proj.fromLonLat([50.538699,26.180643]),
 zoom: 11,
 minZoom:11,
});
 var map = new ol.Map({
   view: view,
   layers:[
     new ol.layer.Tile({
       source: new ol.source.OSM()
     })
   ],
   target: 'map'
 })
}
```

- Rendering heatmap code:

  The following JavaScript code snippet renders a heatmap layer over the base map. Firstly, a gradient is made that signifies the AQI colors. Then, the OpenLayers API is used to render the heatmap. As a source for the heatmap data, a GeoJSON file including intersection readings data is used. The GeoJSON file contains multiple features each with their own set of information. The code will get the ‘magnitude’ of each feature and change the color of the heatmap based on its value.

```
//customized gradient for heatmap
const rainbow = [
 'rgba(102, 245, 66, 1.0)',
 'rgba(255, 247, 28, 1.0)',
 'rgba(255, 142, 28, 1.0)',
 'rgba(245, 25, 10, 1.0)',
 'rgba(143, 10, 245, 1.0)',
 'rgba(105, 9, 9, 1.0)',
]
const htlayer = new ol.layer.Heatmap({
```

```
 source: intersectionsource,
 blur: 35,
 gradient: rainbow,
 radius: 20,
 opacity:2,
 weight: function (feature) {
   return feature.get("magnitude");
 },
});
map.addLayer(htlayer);
```

- Rendering intersection pins code:

  The following JavaScript snippet displays the intersection pins onto the map. The OpenLayers API was used to create an intersection pins layer. The source of this layer is a GeoJSON file that contains intersection data. As a source of the image of each intersection pin an intersectionpinfunction is called. It checks whether the intersection's sensor's status is online or offline and changes the image of the intersection based on that.

```
var intersectionspinfunction =function(feature){
 var status=feature.get('OverallsensorStatus');
 if(status=='offline'){
   picstring=intersectionspinoffline;
 }
 else if(status=='online'){
   picstring=intersectionspinonline;
 }

 var retstyle=new ol.style.Style({
   image: new ol.style.Icon(({
     scale: [0.17, 0.17],
     src: picstring
   }))
 })
 return retstyle;
}
var intersectionsMarkerLayer = new ol.layer.Vector({
  minZoom:12,
  title:'intersectionsMarkerLayer',
 source: intersectionsource,
 style:intersectionspinfunction
});
      map.addLayer(intersectionsMarkerLayer);
```

- City GeoJSON file creation code:

  The following Python code appends to a GeoJSON file with data used to display the map features. It takes in a JSON compatible TimeStream response. Then it accesses GeoJSON file stored in the Django media file. Whilst the file is open, it will create a string that contains the city information. Then it will append the string to the GeoJSON file.

```
def cityGeoJSONAppend(tscityquery):
   filename = os.path.join(settings.MEDIA_ROOT,
"GeoJSON/cities.GeoJSON")
   f = default_storage.open(os.path.join(settings.MEDIA_ROOT,
"GeoJSON/cities.GeoJSON"), 'w')
   filestring=''
```

```
   filestring+=str('{"type":"FeatureCollection", \n "features":\n [')
#fixed GeoJSON start line
   cqueryinfo=json.loads(tscityquery)
   for i in cqueryinfo:
       filestring+=str('\n{"type": "Feature",
\n"properties":{"coordinateType":"city",')#Fixed for each feature
       docdbquerydict={}
       for k, v in i.items():
           filestring+=str('\n"'+k+'":'+'"'+v+'",') #prints each
key+value in time stream query
           if k=='city' or k=='cityType':
               docdbquerydict.update({k:v})
       filestring+=str(docDBCityPrint(docdbquerydict))
       filestring+=str('},')#closes block of info for a feature
   filestring=filestring.rstrip(',')
   filestring+=str(']}') #closes entire feature list
   f.write(filestring)
   f.close()
```

6.6 System Testing

6.6.1 Unit Testing

- **MQ7 Sensor Reading Testing**
  After connecting the CO sensor with the Arduino, an experiment was done to test the correctness of the taken readings. The CO concentration in the air was increased by covering a candle with jar so the oxygen under the jar be used up, and by that, the flame will turn off leading to CO emissions. Then the sensor placed inside the jar to sense CO. The testing failed couple of times as the readings of CO average and AQI average were too high due to ppm conversion error. The error was solved and the sensor was tested many times successfully. This testing was initially done with the Arduino Code and reattempted with the python script and the reading results were matching for the variation of values with the smoke levels.

- **Jetson Nano object detection and tracking testing**
  After the integration of the detection and tracking algorithm, the new code was tested using video footage of moving vehicle of various types and sizes. The initial results were promising as it has successfully detected vehicles and their types, however it would sometimes misclassify vehicles which lead to an overall inaccurate result. As a solution, we have set a threshold value that ensures only cars over a certain confidence score will be counted in the detection process and passed as a value to he tracker. This guarantees that the detector and tracker will start giving accurate results by excluding objects with low confidence score.

6.6.2 Functional Testing

The testing was done with duration of 30 minutes that collected the data for every 5 minutes and send it to the AWS Cloud services. Initially the Jetson Nano will use the pre-record video to detect the vehicles objects along with the wait time and meanwhile the CO-MQ7 Sensor will take the readings of carbon from the jar testing expermint simulating a similar CO of the actual pollutions emitted from the vehicles at the traffic lights, as stage one of the functional testing. The collected data for 5 minutes will be sent as MQTT to IoT Core to be processed through IoT Rule to be sent to TimeStream as second stage of the backend level. on the other

side, the frontend will be utilizing the Elastic Beanstalk for hosting the application for MAQIAS monitoring dashboard and AWS congnito for the website hosting to be accessed by the client. Utilizing the site will require to pull the data stored which been done through DocumentDB database having all the information gathered by the backend. For better understanding and analysis a visualization of the data from tables and graphs were created with the use of Amazon's Business Intellegence (BI) called Quicksight. All the events were processed and analyzed with the help of AWS TimeStream by keeping the recent data in the memory based on the defined policies.

Chapter 7
Conclusion and Future Work

To conclude, this project delivers a monitoring dashboard that is considered to be one of the milestones for Bahrain's emission for net zero carbon 2060, by integrating the vehicles detection using Jetson Nano and Carbon Monoxide measurement through CO-MQ7 Sensor with AWS IoT Core which processed and visualized for analysis and display that will help for better planning that targets minimizing the concentration of CO in the intersection within the traffic lights. During this project an integration between software and hardware been done with the help of AWS services as the integration medium. This experience was a great interest for us to complete as it allowed us to contribute in bigger project that will help the society of Bahrain and vision of cleaner environment.

7.1 Limitation and Future Work

Many challenges we overcome and learned from, along with it our enthusiasm grown to develop and enhance this project. Unfortunately time and knowledge are limited with our level to have a bigger steps to deliver a better version, however there are ideas we would love to add for the current version.

- **Different type of Gases**

  Air Pollution is not limited to Carbon Monoxide, and according to Public Commission for the Protection of Marine Resources, Environment and Wildlife there are more 4 air pollutant reasons emitted from transportation we would like to include in collecting data as part of the air pollution; Sulfur Oxide (SO), Nitrogen Oxide (NO), Particulate Matter (PM), Hydrocarbons (HCs).

- **Optimization**

  On of the general aims for the project is to have smoother and better performance along with adding more nodes with enhance accuracy, bigger coverage, more data collected, increased analysis and visualization of the data.

- **Readings with different approaches**

  Making this tool to have reading on a moving vehicles on highways, intersection of roundabouts, which will give a variety of the collected data, that will provide an advance comparison between the road types relatively to quantity of vehicles, which will help on narrowing the best options for decision making policy issuers in regards for traffic regulations.

- **Vehicle Types and Models**

  Distingushing between a car or truck is essential for analysis, adding vehicle's brands, types, or released models will greatly help on understanding the behaviour of air pollution with these information and how to formulate a policy against the most emitted pollution.

- **Target pollution**

Trying for know the pollution emitted from a targeted vehicles will help to know the cars required fixing, or retiring for a cleaner environment.

- **Map play features**

  Adding a feature to the dashboard's map that allows users to select a time range then generate a play feature. This play feature shows the map's data for the different days within the selected time range as multiple frames. It allows users to view the AQI changes that occur over time in a more visual way.

References

ReQtest (2012) *Why is the difference between functional and Non-functional requirements important?* Available at: https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/ (Accessed: 14 May 2022).

Mustafeez, A.Z. *What is Visual Studio Code?* Available at: https://www.educative.io/edpresso/what-is-visual-studio-code (Accessed: 14 May 2022).

themejunkie. *What is Figma? (And How to Use Figma for Beginners).* Available at: https://www.theme-junkie.com/what-is-figma/ (Accessed: 14 May 2022).

GithubDocs. *Hello World.* Available at: https://docs.github.com/en/get-started/quickstart/hello-world (Accessed: 14 May 2022).

Linux. *What Is Linux?* Available at: https://www.linux.com/what-is-linux/ (Accessed: 14 May 2022).

Soumik, S.K. (2021) *13 Reasons Why Linux Is Better Than Window.* Available at: https://medium.com/swlh/13-reasons-why-linux-is-better-than-windows-6fa304454ae#:~:text=Linux%20tends%20to%20be%20a,access%20to%20its%20source%20code (Accessed: 14 May 2022).

Docker. *Use containers to Build, Share and Run your applications.* Available at: https://www.docker.com/resources/what-container/ (Accessed: 14 May 2022).

Simplilearn (2021) *What is PyTorch, and How Does It Work: All You Need to Know.* Available at: https://www.simplilearn.com/what-is-pytorch-article (Accessed: 14 May 2022).

TensorFlow. *An end-to-end open source machine learning platform.* Available at: https://www.tensorflow.org/ (Accessed: 14 May 2022).

Nvidia. *Jetson Nano Developer Kit.* Available at: https://developer.nvidia.com/embedded/jetson-nano-developer-kit (Accessed: 14 May 2022).

MathWorks. *What Is Object Detection?* Available at: https://www.mathworks.com/discovery/object-detection.html (Accessed: 14 May 2022).

Meel, V. *Object Tracking in Computer Vision (Complete Guide).* Available at: https://viso.ai/deep-learning/object-tracking/ (Accessed: 14 May 2022).

Barla, N. (2022) *The Complete Guide to Object Tracking [+V7 Tutorial].* Available at: https://www.v7labs.com/blog/object-tracking-guide#:~:text=Object%20tracking%20refers%20to%20the,single%20frame%20of%20the%20video (Accessed: 14 May 2022).

AWS. *The world's first deep learning enabled video camera for developers.* Available at: https://aws.amazon.com/deeplens/#:~:text=The%20world's%20first%20deep%20learning,to%20expand%20deep%20learning%20skills (Accessed: 14 May 2022).

Fredrick, A. (2021) *Object Detection with YOLOv5 and PyTorch.* Available at: https://www.section.io/engineering-education/object-detection-with-yolov5-and-pytorch/ (Accessed: 14 May 2022).
Cohen, J. (2019) *Computer Vision for tracking.* Available at: https://thinkautonomous.medium.com/computer-vision-for-tracking-8220759eee85 (Accessed: 14 May 2022).
The National (2021) *Bahrain plans for net-zero carbon emissions by 2060.* Available at: https://www.thenationalnews.com/gulf-news/bahrain/2021/10/24/bahrain-plans-for-net-zero-carbon-emmission-by-2060/ (Accessed: 14 May 2022).

McCombes, S. (2019). *How to Write a Literature Review | Guide, Video, & Template*. Available at: https://www.scribbr.com/dissertation/literature-review/ (Accessed: 14 May 2022).

Toma, C., Alexandru, A., Popa, M. and Zamfiroiu, A., 2019. IoT solution for smart cities' pollution monitoring and the security challenges. *Sensors*, *19*(15), p.3401.
Montanaro, T., Sergi, I., Basile, M., Mainetti, L. and Patrono, L., 2022. An IoT-Aware Solution to Support Governments in Air Pollution Monitoring Based on the Combination of Real-Time Data and Citizen Feedback. *Sensors*, *22*(3), p.1000.
AWS (2021) *An Introduction to AWS IoT*. Available at: https://aws.amazon.com/blogs/startups/an-introduction-to-aws-iot-core/ (Accessed: 14 May 2022).
Arduino (2018) *What is Arduino?* Available at: https://www.arduino.cc/en/Guide/Introduction (Accessed: 14 May 2022).
Nawazi, F. (2021) *MQ7 Carbon Monoxide (CO) Gas Sensor Module*. Available at: https://circuits-diy.com/mq7-carbon-monoxide-co-gas-sensor-module/ (Accessed: 14 May 2022).
Thompson, B. (2022) *What is C Programming Language? Basics, Introduction, History* Available at: https://www.guru99.com/c-programming-language.html (Accessed: 14 May 2022).
Zola, A. (2021) *Python Definition.* Available at: https://www.techtarget.com/whatis/definition/Python (Accessed: 15 May 2022).
Singh, H. (2021) *What is Air Quality Index and how is it calculated?* Available at: https://www.jagranjosh.com/general-knowledge/what-is-air-quality-index-1573026691-1 (Accessed: 14 May 2022).
Sharma, S. (2019) *What is Air Quality Index and how is it calculated?* Available at: https://www.pranaair.com/blog/what-is-air-quality-index-aqi-and-its-calculation/ (Accessed: 14 May 2022).
Synopsys (2022) *What Is the Agile SDLC and How Does It Work? | Synopsys.* Available at: https://www.synopsys.com/glossary/what-is-agile-sdlc.html (Accessed 15 May 2022).

Josh, A. (2016) *Arduino lesson – MQ-7 Gas Sensor* Available at: https://kookye.com/2018/11/16/arduino-lesson-mq-7-gas-sensor/ (Accessed: 29 March 2022).

AWS (2022*) Amazon Timestream – Time Series Database – Amazon Web Services*. Available at: https://aws.amazon.com/timestream/?whats-new-cards.sort-

by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc&amazon-timestream-blogs.sort-by=item.additionalFields.createdDate&amazon-timestream-blogs.sort-order=desc (Accessed 16 May 2022).

AWS (2022) *Amazon DocumentDB*. Available at: https://aws.amazon.com/documentdb/ (Accessed 16 May 2022).

AWS (2022) *Amazon Cognito - Simple and Secure User Sign Up & Sign In*.Available at: https://aws.amazon.com/cognito/ (Accessed 16 May 2022).

AWS (2022) *Amazon QuickSight - Business Intelligence Service.* Available at: https://aws.amazon.com/quicksight/ (Accessed 16 May 2022).

AWS (2022) *AWS Elastic Beanstalk – Deploy Web Applications*. Available at: https://aws.amazon.com/elasticbeanstalk/ (Accessed 16 May 2022).

AWS (2022) *Amazon Cognito - Simple and Secure User Sign Up & Sign.* Available at: https://aws.amazon.com/cognito/ (Accessed 16 May 2022).