Development of Semantic Middleware for South Africa's Multi-Hazard Early Warning System

by

YOLO MADANI

Submitted in fulfilment of the requirements for the degree

MASTER'S

IN INFORMATION TECHNOLOGY

In the

Faculty of Engineering, Built Environment, and Information Technology:

Department of Information Technology

Central University of Technology, Free State

Promoter: Dr. A Akanbi

Co-Promoter: Ms. M Mbele and Prof. EM Masinde

2024

Copyright Notice

Disclaimer

The introduction of this dissertation contains colour pictures implemented to show the results in a more reasonable shape. Whereas printing or seeing the dissertation in greyscale (dark and white) is conceivable, it is suggested that for clarity, the paper is seen (or printed) in full-colour range.

Dedication

This dissertation is dedicated to the entire Madani family for their unwavering support. I also dedicate this work to my unborn children. It is also dedicated to the loving memory of my late mother, Talenta Olga Madani, and my grandmother, Makhoale Anna Mokalake, whose legacies continue to inspire me.

Declaration

This dissertation is the result of my original research, and has not been submitted for any other university award. All information derived from external sources has been appropriately acknowledged, with clear citation and reference to the relevant literature.

This study was conducted under the primary supervision of Dr. A Akanbi from the Department of Information Technology at the Central University of Technology, Free State, South Africa. The research adhered to rigorous academic standards, and established research methodologies. Additional supervision was provided by Professor Muthoni Masinde, and Ms. Mpho Mbele, both from the Department of Information Technology at the Central University of Technology, Free State.

Yolo Madani

Signature: _____ _____

Date: _____

In our capacity as the supervisors of this dissertation, we certify that the above statements are true to the best of our knowledge.

Dr. A Akanbi

Signature: _____

Date: _____

Professor EM Masinde

Signature: _____

Date: _____

Ms. M Mbele

Signature: _____

Date: _____

Preface and Acknowledgements

I begin this acknowledgement with deep gratitude to the Almighty, the source of all strength and guidance, for believing in me when I sometimes doubted myself. It was through faith that I found the perseverance to embark on this learning journey. Without divine support, I would not have reached this stage in my life. I owe myself a thank you. Through the long nights and difficult times of this master's academic journey.

I express my deepest gratitude to my main supervisor, Dr A. Akanbi, whose wisdom, expertise, and guidance helped me develop this dissertation. Your patience and dedication to developing my research skills have been invaluable. I am truly lucky to have you as a mentor. I accept that the priceless information and experience I picked up from you will always guide and help me in both academic and non-academic contexts . I also thank my co-supervisors, Prof Muthoni Masinde and Ms Mpho Mbele for being a pillar of support with words of encouragement and upliftment.

To my colleagues, friends and family, your support has been immeasurable. The encouraging words, listening ears, and countless meetings during this journey have sustained me. Thank you for your unwavering belief in my abilities. I am grateful to each person who played a part, no matter how small, in helping me achieve this milestone. Your contributions, in the form of advice, encouragement, or simply being there for me have not gone unnoticed. Lastly, my heartfelt thanks go out to everyone who contributed to this dissertation, and to my overall personal and academic growth. This achievement is the result of a collective effort, and I am profoundly thankful for the support and encouragement I received throughout this journey.

Yolo Madani

2024

Bloemfontein, Free State

South Africa

Abstract

The impact of man-made activities and climate change has exacerbated the occurrence of environmental hazards. To improve monitoring systems, integrating Internet of Things (IoT) sensors, legacy systems, and enterprise networks has become crucial for efficient monitoring systems. However, existing heterogenous monitoring systems face issues like data incompatibility, lack of integration, and interoperability, making it challenging for communities to comprehend crucial information. Monitoring systems requirements vary significantly based on the environment, resulting in ad-hoc implementations and integration of different systems and applications. This study addresses these challenges by exploring the use of semantic middleware to harmonize heterogeneous systems, facilitating seamless integration and interoperability within a Multi-Hazard Early System (MHEWS). The study proposes that semantic middleware can overcome the challenges of data representation, integration and system interoperability within a MHEWS.

The study focuses on the environmental challenges in mining operations at Lejweleputswa, Free State Province, South Africa as a case study, and adopts two existing Early Warning Systems (EWS), namely Information Technology and Indigenous Knowledge with Intelligence (ITIKI), and Adaptive Environmental Management System (AEMS) frameworks for a MHEWS. The use of semantic representation mechanisms is crucial for connecting diverse systems and facilitating effective communication in a heterogeneous system. This research study demonstrates seamless integration attainability through application containerization, specifically the use of Docker and Kubernetes, to improve the integration, interoperability, deployment, and portability of diverse systems in the context of semantic representation through a middleware. The application of container orchestration platform - Dockers and Kubernetes further strengthens the MHEWS by enabling autoscaling, load balancing, and fault tolerance to ensure high availability even during infrastructure failures. This study achieved the data integration and systems interoperability through the semantic middleware that improves the overall performance of the Multi-Hazard Early Warning System using the case study of Lejweleputswa district, Free State Province, South Africa.

List of Acronyms

ABM – Agent-Based Middleware

ACR – Azure Container Registry

ADL – Azure Data Lake

AEMS – Adaptive Environmental Management System

AF – Azure Function

AI – Artificial Intelligence

AKS – Azure Kubernetes Service

API – Application Programming Interface

ASF – Azure Service Fabric

AUM – Adaptive Ubiquitous Middleware

AWS – Amazon Web Service

BDRR – Blue Drop Risk Rating

COPD – Chronic Obstructive Pulmonary Disease

COAP – Constrained Application Protocol

CSP – Cloud Service Provider

CLI – Command Line Interface

CUT - Central University of Technology

DDL – Data Definition Language

DEWS – Drought Early Warning System

DDM – Database-Driven Middleware

DSR – Design Science Research

EBM – Event-Based Middleware

ERD – Entity Relationship Diagram

ETL – Extract, Transform, Load

EWS – Early Warning System

EWSs – Early Warning Systems

FG – Functional Group

FR – Functional Requirement

FRIC – Faculty Research and Innovations Committee

FCM – Fuzzy Cognitive Maps

GCP – Google Cloud Platform

HTTP – Hyper Text Transfer Protocol

IaaS – Infrastructure as a Service

ICT – Information and Communication Technologies

IDE – Integrated Development Environment

IDL – Interface Definition Language

IK – Indigenous Knowledge

IKS – Indigenous Knowledge System

IoT – Internet of Things

IS – Information Systems

ITIKI – Information Technology and Indigenous Knowledge with Intelligence

JSON – JavaScript Object Notation

K8'S – Kubernetes Architecture

MHEWS – Multi-Hazard Early Warning System

MA – Mobile Agent

MOM – Message-Oriented Middleware

MAC – Microsoft Azure Cloud

NFR – Non-Functional Requirement

ORB– Object Request Broker

OOCM – Object-Oriented Middleware

OWL – Ontology Web Language

PaaS – Platform as a Service

PEWS – Pollution Early Warning System

RDF – Resource Descriptive Framework

REST – Representational State Transfer

RH – Research Hypothesis

RO – Research Objective

RQ – Research Question

SAAQIS – South Africa Air Quality Information System

SABDS – South Africa Blue Drop System

SaaS – Software as a Service

SBM – Service-Based Middleware

SOA – Service Oriented Architecture

SOAM – Service Oriented Architecture Middleware

SQL – Structured Query Language

SSN – Semantic Sensor Network

SMS – Short Message Service

TOM – Transaction-Oriented Middleware

UML – Universal Modelling Language

VM – Virtual Machine

VMBM – Virtual Machine-Based Middleware

WHO – World Health Organization

WSDL – Web Service Description Language

WSDC – Wireless Sensor Data Collection

WSN – Wireless Sensor Networks

XML – Extensible Markup Language

Glossary

- *Case study*: An in-depth study through analysis of specific instance, event, group or organization aimed at exploring and comprehending complex issues within real life contexts.
- *Data analysis:* The interrogation of acquired data to come up with summaries and trends in the study variable.
- *Event*: An event is an occurrence taking place at a determinable time and place, with or without the participation of human agents.
- *Focus groups:* A selected group of expert/respondents.
- *JSON*: An open-standard file format that uses human-readable text to transmit data objects consisting of attribute–value pairs and array data types.
- *Legacy System*: A computer system, programming or application software that is outdated or that can no longer receive support and maintenance.
- *Open-ended questions*: A set of questions to which respondents are free to give their own responses.
- *Population:* The set of all people in the communities' studies.
- *Qualitative research*: Research focusing on descriptive data and responses.
- *Quantitative research*: Research focusing on several responses.
- *Research design:* A plan for conducting research.
- *Sample:* A subset of a population.
- *Structured interview:* A set of predefined questions to guide researchers and respondents in answering of questions.
- *Validity:* The degree of a result to reflect the meaning of a tested variable.
- *W3C*: The main international standards organisation for the World Wide Web.

Table of Contents

List of Figures

List of Tables

List of Publications

As a result of this research, the author of this dissertation has also authored the following published papers.

i. Madani, Y., Akanbi, A., Mbele, M. and Masinde, M., 2023, August. A Scalable Semantic Framework for an Integrated Multi-Hazard Early Warning System. In *2023 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD)* (pp. 1-6). IEEE.

ii. Maneo Ntseliseng Ramahlosi, Yolo Madani and Adeyinka Akanbi. (2023, October). A Blockchain-based Model for Securing Data Pipeline in a Heterogeneous Information System. Published Online by the SAICSIT 2023 Organising Committee Potchefstroom: South African Institute of Computer Scientists & Information Technologists ISSN:2959-8877 (electronic) (p. 167). ACM.

iii. Madani, Y. and Akanbi, A., 2024, May. A Semantic Middleware using Docker and Kubernetes Orchestration Tools. In *2024 IST-Africa Conference (IST-Africa)* (pp. 01-12). IEEE.

CHAPTER ONE

INTRODUCTION

1.1 Background Information and Motivation

An Early Warning System (EWS) provides advance notice of natural hazards, allowing communities to take proactive measures to reduce vulnerability, and enhance resilience. However, the importance of EWS goes beyond the scope of a single natural hazard. This is where the Multi-Hazard Early Warning System (MHEWS) is essentially needed to consolidate data and information on various hazards such as floods, earthquakes, drought, and pollution into a unified structure. This allows researchers, policy makers, and users to anticipate, understand, and respond to the complexity that arises when multiple threats occur. Though mining significantly boosts the nation's economy, these financial advantages come at a steep price, negatively affecting both the environment and the health of residents as is the case of Lejweleputswa district (Feris & Kotze, 2015). This district for instance is battling with pollution emanating from mining. Pollution represents an unwanted alteration in the physical, chemical, or biological properties of air, soil, or water, which can harm life or present health risks to living organisms (Morakinyo et al., 2017).

South Africa has taken steps to address the negative effects of pollution on public health, the natural environment, and economic stability. To achieve this, the country has introduced various pollution monitoring tools. These tools serve the purpose of safeguarding and evaluating pollution levels by providing information about areas at high risk of pollution, providing real-time data, and monitoring air quality. Notable advances include integration with the South African Air Quality Information System (SAAQIS) (Gwaze & Mashele, 2018), a comprehensive framework for managing and distributing air quality information across the country to improve air quality monitoring and management. By leveraging such tools and systems, South Africa aims to better understand, track, and manage environmental pollution, ultimately reducing its impact on various fronts. Nevertheless, issues with data integration and systems interoperability continue to exist (Gwaze & Mashele, 2018).

In Lejweleputswa, Free State province, South Africa, various forms of pollution, including air pollution from mining activities, water contamination, and soil degradation pose significant environmental and public health risks. While frameworks such as the Adaptive Environmental Management System (AEMS), and the Information Technology with Indigenous Knowledge (ITIKI) drought forecasting tool are available, they operate in silos, lacking integration into a unified environmental management strategy. This disjointed approach diminishes their effectiveness, as it prevents a comprehensive response to the region's multifaceted pollution challenges. Without an integrated EWS addressing all types of pollution, the area remains vulnerable to environmental degradation, with current systems offering limited predictive insights. To address these challenges, it is essential to develop an integrated approach that consolidates these tools into a more effective pollution management framework, related research studies have demonstrated that semantic middleware offers a promising solution to harmonize heterogeneous systems (Akanbi & Masinde, 2020; Aslam & Curry, 2021; Bamhdi, 2021; Zgheib et al., 2019). Semantic technologies such as Ontologies, Resource Description Framework (RDF), and SPARQL protocol provide powerful tools for representing, organizing, and retrieving data in ways that go beyond the limitations of traditional databases (Rodrigues et al., 2019).

In the context of this study, semantic middleware aims to enable seamless data integration, interoperability, and knowledge sharing between the diverse early warning systems involved in the MHEWS employed in this case study for Lejweleputswa, Free State Province, South Africa. This is to increase the effectiveness and accuracy of hazard warnings. This research proposes that utilising semantic representation can deliver precise and valuable forecasting information by combining different data sources within each EWS through a proposed semantic middleware for a MHEWS. Semantic middleware will enable the seamless integration of data collected from multiple sources into various environmental management centers, ensuring effective system interoperability and federation of data to achieve seamless distribution of information, allowing for accurate data translation at every point of entry.

This chapter provides the foundation for the entire study and provides a comprehensive understanding of the context, logic, and structure of the study explains the urgent need to achieve interoperability and data integration in MHEWS for the study area and discusses the limitations of the current systems. Additionally, this chapter provides insight into the rationale on the use of semantic middleware and semantic technology to enable intelligent data integration and interoperability. By Introducing semantic middleware, a paradigm shift is

expected in how hazards are monitored, analysed, and communicated through the integration of multiple EWS making up the MHEWS for faster and more effective disaster response.

1.2 Problem Statement

Pollution as an environmental hazard is a key concern in communities surrounded by mining operations in South Africa, and communities in the Lejweleputswa district in the Free State Province of South Africa are also badly experiencing it. To mitigate the impact of environmental hazards, Multi-Hazard Early Warning Systems (MHEWS) are critical, offering timely and reliable information to local communities. However, the effectiveness of these systems is compromised by persistent challenges related to data representation, integration, and interoperability. In South Africa, the absence of uniform data formats and standards among various Early Warning Systems (EWS) deployed by different agencies and organisations creates substantial barriers to seamless communication and collaboration. These disparities hinder the transfer, transformation, and interpretation of data across systems, leading to inefficiencies and delays in disseminating early warnings. Consequently, MHEWS are unable to operate as cohesive frameworks capable of effectively addressing pollution-related hazards. The research aims to address these challenges by developing a semantic middleware tailored to the specific needs of MHEWS in the Lejweleputswa district, leveraging existing EWS, such as Information Technology with Indigenous Knowledge (ITIKI) and Adaptive Environmental Management System (AEMS), the proposed middleware will provide a standardised and interoperable communication framework addressing the complexity of data representation and ensuring compatibility between heterogeneous EWS and their subsystems, the solution aims to enhance the efficiency of MHEWS in delivering accurate and timely multi-hazard alerts.

1.3 Research Questions and Objectives

To address the research challenges mentioned earlier, the following research questions are considered in the approach to integrating and ensuring interoperability of heterogenous data:

a) **RQ1:** How does the integration of existing monitoring systems enhance the overall effectiveness and efficiency of the MHEWS in comparison to traditional approaches?

b) **RQ2:** How can the integration of the systems through semantic middleware enhance or improve the decision-making process for disaster risk reduction and response by providing actionable insights based on the integrated data?

c) **RQ3:** What are the existing semantic middleware and frameworks that can be adapted for the South Africa Multi-Hazard Early Warning System, and how do they compare in terms of suitability and efficiency?

d) **RQ4:** To what extent will the implementation of semantic middleware improve the efficiency and effectiveness of South Africa's MHEWS in terms of information integration, interoperability, and timely dissemination of warnings across different hazard scenarios?

To address the aforementioned research questions, this study is aimed to develop semantic middleware for data representation, system integration, and interoperability for an existing EWS in a MHEWS. Using ITIKI and AEMS early warning systems as a case study to monitor environmental pollution and weather situations, for a better understanding of how critical interconnected EWS works holistically for monitoring and predictions. This primary aim was delineated by breaking it down into the following sub-objectives:

a) **RO1:** Develop a semantic framework for an MHEWS to improve the automatic configuration of the EWS for data integration and system interoperability.

b) **RO2:** Identifying and evaluating existing semantic middleware and developing a semantic integration framework that has the potential to harmonize the adopted EWS in the context of an MHEWS.

c) **RO3:** Develop and test a prototype middleware layer that semantically represents data from different EWS in the context of pollution monitoring in Lejweleputswa district, Free State.

d) **RO4:** Evaluate to what extent the use of semantic middleware within MHEWS facilitates the system's heterogeneous EWS integration and interoperability.

1.4 The Solution Approach

The methodology addresses the above-mentioned objectives of this study and follows an experimental approach and Design Science Research (DSR) methodology with the development of an MHEWS as a case study. To address the challenges of complex data

representation and translation in MHEWS, the proposed solution involves the development of semantic middleware based on semantic web technologies. The semantic middleware is developed to standardize the communication protocol between EWS and other MHEWS sub-systems. Key components of this solution include building a framework for data transformation and integration and leveraging Docker containerization for increased portability based on semantic representation of the data sources. Additionally, containerization tools – Dockers/Kubernetes serve as an orchestration platform for managing and scaling EWS as a container deployment. The approach implemented improves MHEWS ability to provide accurate multi-hazard warnings and improves data representation, integration, sharing facilitated by semantic representation technologies.

1.4.1. Framework Development

A semantic representation middleware framework is presented in Chapter 3. The framework shows the semantic representation of the heterogenous data and automatic configuration of adopted EWS in the MHEWS for the study area. This framework enables the data representation using Application Programming Interface (APIs) and service layers in the middleware to achieve the data integration and interoperability objectives. The essential objective is to introduce a semantic middleware framework capable of harmonizing heterogenous datasets, i.e., indigenous knowledge, and sensor data (which are structured data, and unstructured data, respectively) to enhance the compatibility among adopted early warning systems. The middleware offers an API that simplifies the abstraction of complex modules, and communication and ensures that data is presented in a machine-readable format thus fostering integration and interoperability.

1.4.2. Middleware Development

Middleware layers serve as hubs for connecting various services and tools through Application Programming Interface (APIs), facilitating communication between different applications. These APIs act as bridges, enabling disparate products and services to interact seamlessly. Alternatively, asynchronous data streaming replicates data in an intermediary storage space, allowing multiple applications to access and share it. The development of middleware layers typically involves four components: an Interface Definition Language (IDL), an interoperability protocol, an Object Request Broker (ORB), and supplementary services. Semantic technology is integrated into the interoperability layer using Ontology Web Language (OWL) and Resource Descriptive Framework (RDF), which are machine-readable languages

based on domain ontology, and particularly tailored for the environmental monitoring sector. This integration establishes a framework for how components exchange information, ensuring data integration and interoperability by defining message types, formats, and content interpretation standards.

1.5 Limitations of the Research Scope

The scope of the research study mainly focuses on air pollution-related EWS, neglecting other important aspects of MHEWS, such as water-related hazards and other extreme weather conditions. Nonetheless, this study recognizes specific constraints. The semantic middleware relies on the availability and quality of data for semantic integration of early warning systems. This limitation can affect the system's ability to adapt to new hazards and changing environmental conditions and potentially limit the depth of this research study.

1.6 Significance and Contributions of the Study

The scope of this study is on the semantic integration of existing EWS, and two systems, namely ITIKI and AEMS were adopted as the case study of Lejweleputswa district, Free State Province, South Africa. The development of semantic middleware and the containerization of EWS applications in the context of MHEWS using Docker and Kubernetes is a significant contribution in achieving interoperability and integration of ITIKI and AEMS. These contributions have increased not only the objectives of data integration and systems interoperability, but also the efficiency, scalability, and flexibility of these systems in many ways. The introduction of semantic middleware has enabled better data integration between platforms and made understanding the data easier by facilitating the combination of information from different sensors and databases. Relatedly, semantic middleware adds context to data and increases its usability across systems.

Additionally, Docker as a containerization tool has simplified efficient data processing by grouping all necessary dependencies into a single container to ensure consistent behavior across environments, simplifying deployment, and reducing configuration issues. Kubernetes provides automated orchestration and scaling of containers, enabling EWS applications in the MHEWS to seamlessly handle increased workloads during critical events. MHEWS containerization also ensures that ITIKI and AEMS work efficiently and are available anywhere, whether on-premises or in the cloud, with Kubernetes as a tool that supports redundancy and failover mechanisms, ensuring high availability of the systems. In the event of

a hardware or software failure, the system can be restored quickly, and without significant impact on operations.

The semantic middleware layer enables better integration with external systems and data sources, improving the overall performance or interoperability of the system. The adoption of Docker and Kubernetes technologies ensure the long-term sustainability of the adopted EWS, i.e., ITIKI, and AEMS by facilitating continuous and efficient system management, not only by improving system performance, scalability, and reliability, but also by positioning the early warning systems to better archive the purpose, thus these contributions have made the systems more efficient, flexible, and scalable.

1.7 Evaluation Criteria

In assessing this study, each objective was examined in relation to the research findings. The case study was assessed for its alignment with and contribution to achieving the research objectives aimed at addressing the research questions.

1.8 Structure of Dissertation

The rest of the study is structured as follows.

a) **Chapter Two: Literature Review**

   This chapter presents a review of relevant literature on environmental monitoring, MHEWS, semantic technologies, semantic middleware, cloud computing, and related work.

b) **Chapter Three: Research Methodology and System Design**

   This chapter outlines the Methodology employed in designing, developing, and evaluating the semantic middleware framework for the South Africa Multi-Hazard Early Warning System with the presentation of the proposed model.

c) **Chapter Four: System Design, Implementation and Analysis**

   This chapter covers the methodology and approach for the experimental setup for the practical implementation and deployment using the case study Multi-Hazard Early Warning System.

d) **Chapter Final Conclusion**

   This chapter concludes the study by summarizing research contributions and suggesting future directions for further research and development.

1.9 Summary

Chapter One establishes the background and motivation for this research, highlighting the importance of Multi-Hazard Early Warning Systems (MHEWS) in addressing complex environmental challenges, particularly in South Africa's Lejweleputswa district. It discusses the limitations of existing Early Warning Systems (EWS) due to poor data integration and interoperability, emphasizing the need for a unified framework to manage diverse hazards effectively. Introducing semantic middleware as a potential solution, the chapter sets the stage for the upcoming chapters, which delve deeper into the literature, methodology, system design, and implementation of a semantic framework to enhance MHEWS functionality.

CHAPTER TWO

BACKGROUND AND RELATED WORK

2.1 Introduction

The environmental concerns and their impact on society are rapidly escalating and have a major impact on human health or society. The integration of advanced technologies and interdisciplinary knowledge is of more importance to mitigate environmental hazard such as pollution in areas affected by mining pollution such as the Lejweleputswa district. This chapter provides an extensive literature review covering related concepts from EWS, MHEWS, and the integration of local indigenous knowledge with scientific knowledge for pollution control. This Chapter also discusses middleware technologies and their requirements, methods, for semantic application. By analysing related and relevant works, the role of cloud computing and microservices in improving the performance and availability of environmental monitoring systems such as MHEWS are discussed. This literature review provides the starting point for the rest of the chapters with critical insights into the existing knowledge and guides the development of innovative and effective system designs and frameworks for MHEWS.

2.2 Background

2.2.1 Early Warning Systems

EWS has become an essential element, playing a key role in reducing risks, saving lives, and minimizing the impact of disasters such as extreme weather events, earthquakes, tsunamis, droughts, and industrial operations, as emphasized by (Masinde & Thothela, 2019). The specific definition of EWS among various interpretations adopted in this study was agreed on by members of the United Nations in 2017. EWS was defined as:

"*An integrated system of hazard monitoring, forecasting and prediction, disaster risk assessment, communication and preparedness activities, systems and processes that enables individuals, communities, governments, businesses and others to take timely action to reduce disaster risks in advance of hazardous events*" (Luther et al., 2017).

The major international initiatives related to EWS include: (1) the 1994 Yokohama Strategy and Plan of Action for a Safer World; (2) the Hyogo Framework for Action 2005–2015: Building the Resilience of Nations and Communities (HFA); and (3) the Sendai Framework for Disaster Risk Reduction 2015–2030 (Luther et al., 2017). The development and

implementation of these systems have played a key role in saving many lives and minimizing economic and environmental losses associated with disasters.

An effective EWS consists of four key elements that are critical to its functioning to ensure accurate, timely, reliable, and understandable information. These elements are discussed below.

i. Disaster risk knowledge based on the systematic collection of data and disaster risk assessments is the first element. This entails both the knowledge of risk as well as the vulnerability of the people that may be affected by the disasters. The ability to capture the knowledge accurately remains enigmatic (Schneider et al., 2016).
ii. Detection, monitoring, analysis and forecasting of the hazards and possible consequences.
iii. Dissemination and communication, by an official source, of authoritative, timely, accurate and actionable warnings and related information about likelihood and impact.
iv. Ensuring readiness at every entry level to act on the warnings received.

For an EWS system to be termed “multi-hazard’ the above definition provides that such a system addresses several impacts of similar or different types and that these hazardous events occur alone, simultaneously, cascading, or cumulatively over time, but their effects have some potential interrelations. As per the Sendai Framework (Pearson & Pelling, 2015), these hazards are classified under four categories: biological, environmental, geological, hydro-meteorological, and technological processes and phenomena. Further, the Sendai Framework requirement for an end-to-end people-centred multi-hazard early warning system involves empowering the affected individuals and communities to act appropriately and timeously to reduce the negative effects of the hazard.

The integration of indigenous knowledge and scientific knowledge into early warning systems is a significant step forward in addressing environmental problems. In this case study, Adaptive Environmental Management System (AEMS) (Mbele et al., 2016) and Information Technology and Indigenous Knowledge with Intelligence (ITIKI) (Masinde & Bagula, 2011) are two innovative EWS frameworks that use Information and Communication Technologies (ICT) and Artificial Intelligence (AI) to improve weather forecasting and pollution monitoring, respectively. The integration of indigenous knowledge and scientific methods into early warning systems, as demonstrated by AEMS and ITIKI, ensures the importance of recognizing and valuing traditional wisdom alongside modern technology. The EWS represents the

important steps to successfully address environmental challenges, increase community resilience, and support sustainable practices, by combining traditional approaches and scientific approaches for better environmental management and disaster risk reduction.

2.2.2 Multi-Hazard Early Warning System (MHEWS)

Early Warning Systems (EWSs) are integral sub-components of Multi-Hazard Early Warning System (MHEWS). In this case study, a crucial part of the MHEWS is the adoption of existing pollution and other environmental EWS, i.e., ITIKI and AEMS for Lejweleputswa district Free State Province, South Africa. The aforementioned adopted EWS frameworks utilise indicators from wireless sensor networks, weather stations, and local indigenous knowledge and are generally similar to any pollution EWS as conceptualized in Figure 2.1 below.

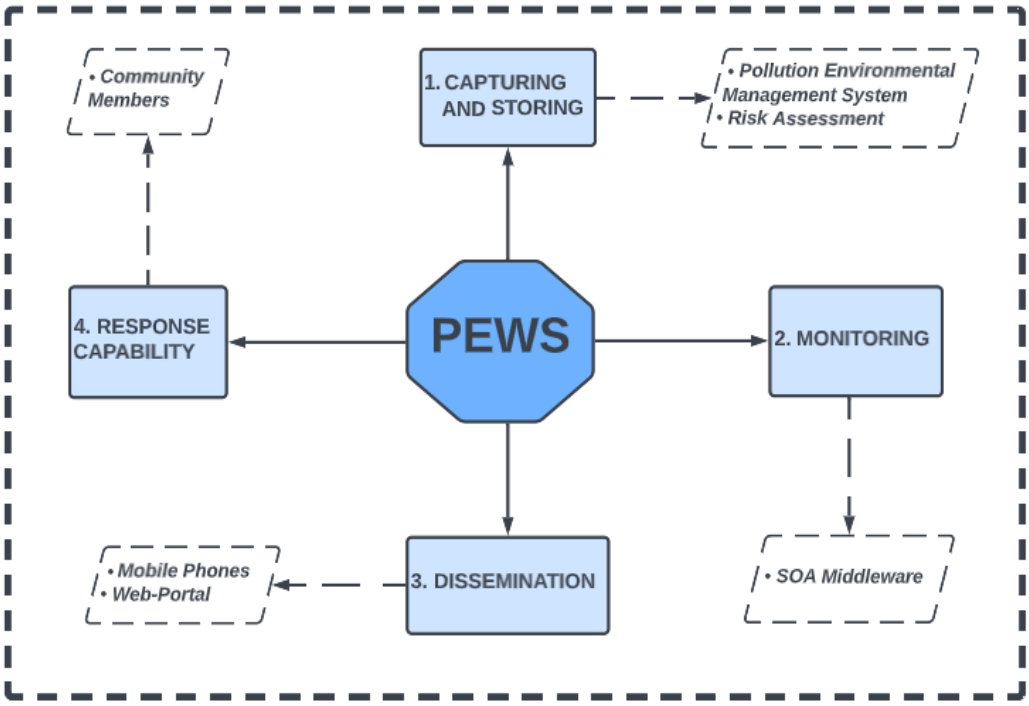


Figure 2.1: Overview of Early Warning System (EWS) Components (Source: Author)

Meanwhile, a working definition of MHEWS is taken as: "*An interrelated set of hazard monitoring and prediction, risk assessment, communication and preparedness sub-systems and processes/activities that enable individuals, communities, governments, businesses and others to take timely action to reduce their risks in advance of hazardous events*" (Luther et al., 2017).

The quest for MHEWS has been pursued for decades, and it was the main agenda at the 2006 International Expert's Symposium on Multi-Hazard Early Warning System, organised by the World Meteorological Organisation (WMO) (World Meteorological Organization (WMO), 2012). In the Sendai Framework for Disaster Risk Reduction (SFDRR) 2012 – 2030, the benefits of a MHEWS are reiterated and enshrined in one of the framework's seven global targets, that is *"Substantially increase the availability of and access to multi-hazard early warning systems (MHEWS) and disaster risk information and assessments to people by 2030"* (Pearson & Pelling, 2015). The framework further stipulates four priorities, which are:

1) Understanding disaster risk reduction,
2) Strengthening disaster risk governance to manage disaster risk,
3) Investing in disaster risk reduction for resilience, and
4) Enhancing disaster preparedness for effective response and to Build Back Better" in recovery, rehabilitation, and reconstruction.

MHEWS allows key stakeholders including governments, businesses, individuals, international organisations and communities to collaboratively act as the first line of defence during a disaster. This is because MHEWS enhances cooperation for data analysis, collection, and operational management, leading to greater effectiveness and efficiency overall. It is in the same light that the use of indigenous warning knowledge has been found to be valuable for enhancing early warning systems and improving community preparedness (Luther et al., 2017). It has also been reported that the sustainability of MHEWS in developing countries has been found to be problematic. Besides, extensive (with low severity and high frequency) disasters such as droughts are on the increase. Here, the major challenge remains what is known as "last mile", where it is evidently more difficult to reach the most remote and vulnerable population. Furthermore, there is a problem with the provision of meaningful and actionable information that integrates gender-sensitive, including perspective.

The key gaps in MHEWS in the developing countries are presented below.

1) Weak coordination and collaboration among the multi-disciplinary actors and agencies involved,
2) Un-interoperable information systems,
3) Lack of Standard Operating Procedures (SOP),
4) Limited public awareness and mostly uncoordinated participation in risk management,
5) Lack political will and commitment and

6) Insufficient financial muscle to implement the systems.

To handle the substantial information needs of the MHEWS, a semantic middleware is proposed in this research study. The developed middleware, integrated into the MHEWS framework, aims to semantically unify various data sources related to pollution and climate events, combining local and scientific knowledge for environmental sustainability. By encoding the gathered information in a machine-readable format, this middleware will enhance system interoperability and support the semantic federation of data.

2.2.2.1 Adaptive Environmental Management System (AEMS)

In this case study, a key component of a comparable project within the study district known as Adaptive Environmental Management System (AEMS) (Mbele et al., 2016) was adopted as an EWS part of MHEWS for monitoring mining-related pollution in the Lejweleputswa district Free State Province, South Africa.

AEMS uses wireless sensor networks, weather stations for data collection and an Artificial Intelligence (AI) tool called Fuzzy Cognitive Maps (FCM) for analysis. FCM are models that capture the relationship between different variables, enabling effective pollution analysis and prediction, see Figure 2.2 below. This approach provides valuable insights into factors influencing pollution, facilitating better environmental management and mitigation decisions. AEMS connects the scientific data collected with the local Indigenous knowledge by involving experts in the field and qualified researchers. The results of the FCM model are interpreted and communicated to the community in an understandable way, enabling informed action and response.

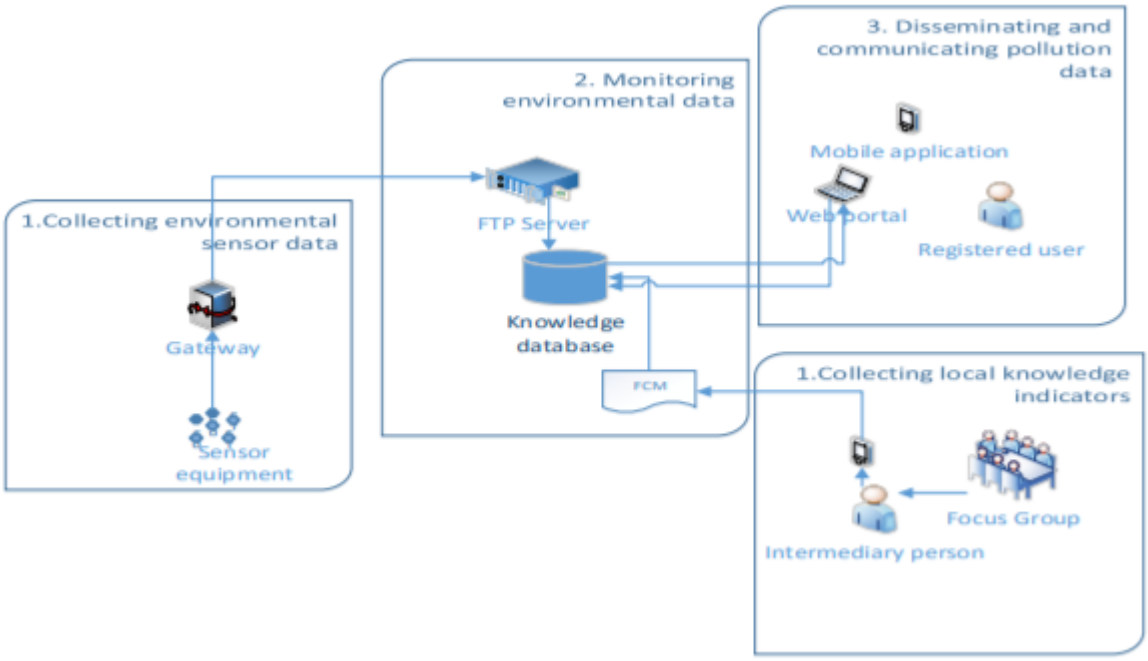


Figure 2.2: Adaptive Environmental Management System (AEMS) Framework (Source: Mbele et al., 2016)

2.2.2.2 Information Technology and Indigenous Knowledge with Intelligence (ITIKI)

ITIKI is a bridge that combines traditional indigenous drought forecasting practices with a scientific approach, by combining mobile phones, wireless sensor networks, and artificial intelligence technologies such as agents, fuzzy logic, and artificial neural networks (Masinde & Bagula, 2011). ITIKI strives to achieve sustainability, accuracy, and acceptability of drought forecasts. The system integrates local knowledge into each of its components, considering the valuable insights and insights of the local communities.

Mobile phones are utilised as input and output devices for the system, working alongside wireless sensor-based weather meters. These tools complement weather stations, enhancing the collection and distribution of weather data. This integration allows for a better understanding of weather patterns and enables early warnings to be communicated effectively to farmers and other stakeholders. The system not only integrates farmers' expertise but also creates a risk management framework by incorporating local knowledge into the process. To further increase accessibility and benefits, the integrated Drought Early Warning System (DEWS) was introduced. A framework was set up that includes natural language processing to translate predictions into local languages. This ensures that information reaches target communities effectively, promoting better understanding and preparation, see Figure 2.3 below.

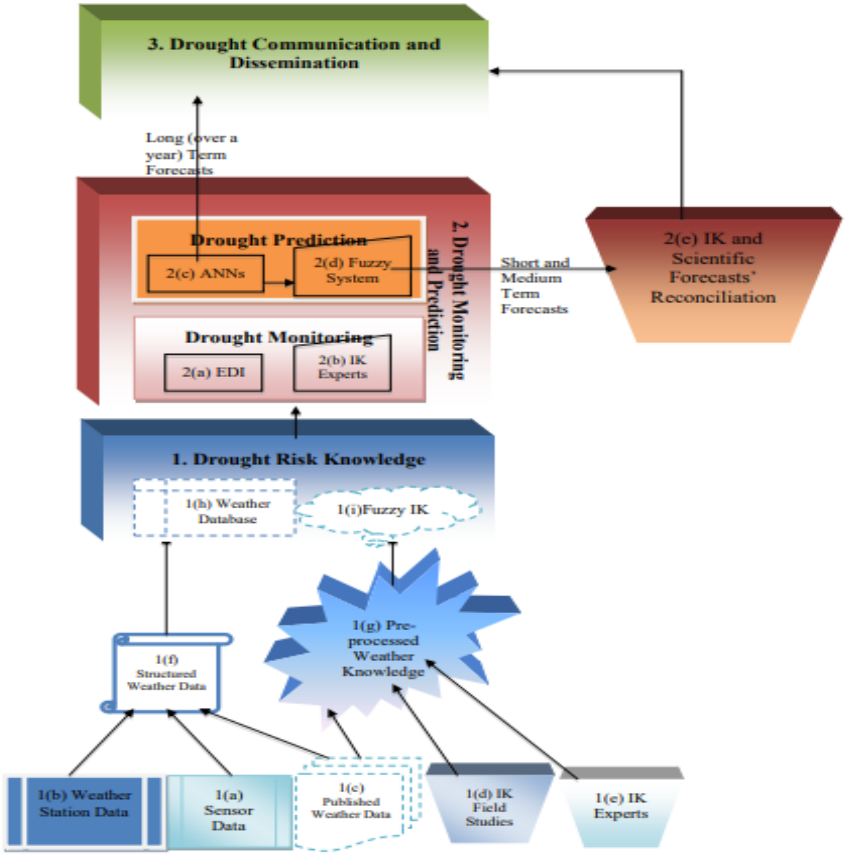


Figure 2.3: Integrated Drought Early Warning System Framework (Source: Masinde & Bagula, 2011)

2.2.3 Environmental Monitoring System

In environmental monitoring systems, see Figure 2.4 below, diverse sensors and legacy systems from various manufacturers measure different environmental properties using distinct data format structures and terminology, resulting in data heterogeneity (Akanbi & Masinde, 2015). The importance of addressing heterogeneity to effectively use integrated heterogeneous systems has been increasingly emphasized in recent research, making it a thoroughly investigated topic in the literature (Akanbi & Masinde, 2020; Malucelli et al., 2006).

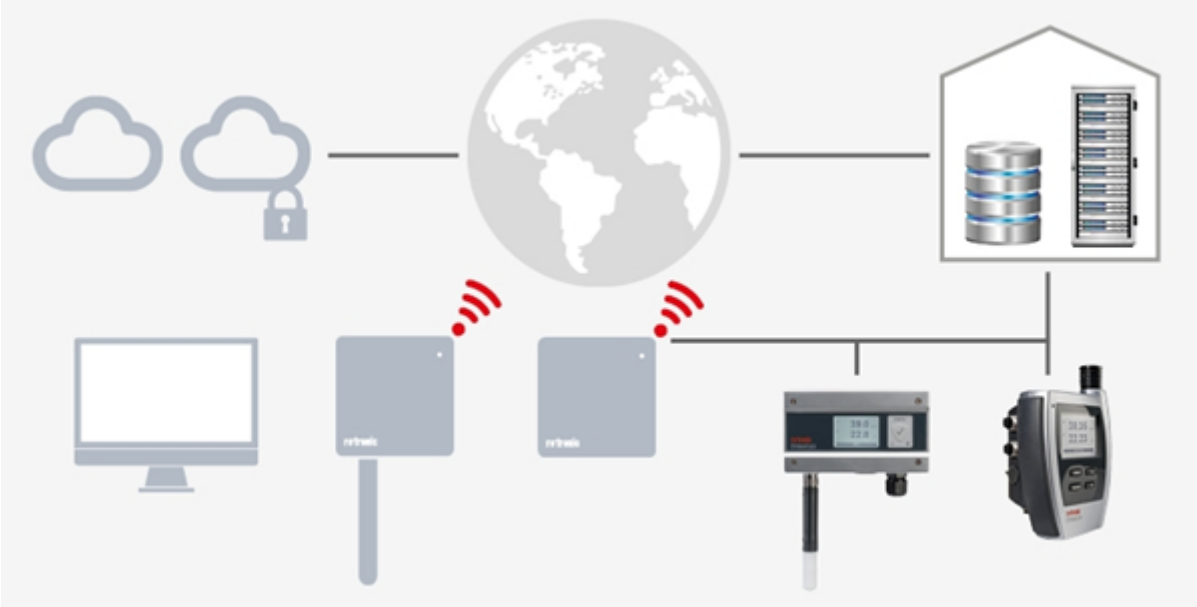


Figure 2.4: A Sample Heterogeneous Environmental Monitoring System (Source: Malucelli et al., 2006)

The use of middleware, positioned between network operating systems and application components, facilitates the seamless integration of subsystems, thereby simplifying the implementation of distributed systems (Zgheib et al., 2019). Middleware addresses heterogeneity, allowing distributed components to integrate and operate together smoothly. Applying semantic technologies within middleware contexts helps manage the complexity of data integration and system interoperability (Akanbi & Masinde, 2020; Malucelli et al., 2006). Semantic technology encompasses a range of methods and tools designed to identify complex relationships among diverse objects and datasets. The importance of integrating semantic technology into middleware is to uncover relationships at the object level, enable machine-readable data representation, and support data integration and interoperability across heterogeneous applications in a multi-system environment (Justin Dhas & Jeyanthi, 2019).

Environmental monitoring encompasses various factors, including weather patterns, flooding, drought, and mining pollution. Acquiring accurate climate data is essential for successfully adapting to and building resilience against climate change. This necessitates the implementation of a comprehensive EWS to detect future climate risks and hazards (Akanbi & Masinde, 2018). The concept of an integrated MHEWS is therefore proposed. This system features a loosely coupled distributed architecture that can integrate diverse data sources (both structured and unstructured), process and analyse data, and disseminate early warning

information at different levels. It aims to enhance risk knowledge, awareness, and early warning response capabilities in the Lejweleputswa district municipalities.

2.2.3.1 Air Pollution

Air pollution is a mixture of airborne particles and gaseous pollutants, it remains a major environmental problem worldwide and is recognized as a major public health risk. Air pollution is also known as a trigger of Chronic Obstructive Pulmonary Disease (COPD) and it has influenced the setting of air quality standards in many countries (Morakinyo et al., 2017). Specifically, air pollution from mining activities is an environmental problem for people living around mining communities. Although mining plays an important role in the country's economy, it often releases harmful pollutants such as dust particles, heavy metals, and greenhouse gases into the atmosphere, and these are capable of degrading the air quality, and pose a threat to public health and damage the ecosystem (Morakinyo et al., 2017). The South African government and mining companies are working towards introducing stricter environmental regulations and sustainable practices to mitigate these impacts. In South Africa, air pollution is currently monitored by South Africa's Air Quality Information System (SAAQIS) to reduce air pollution in South Africa (Gwaze & Mashele, 2018). The SAAQIS was recently upgraded to the second-generation system that was unveiled at the annual Lekgotla Air Quality Management in October 2017 (Gwaze & Mashele, 2018). SAAQIS offers the public real-time updates on air quality, accessible through their website and mobile app, available on both Android and iOS devices.

2.2.3.2 Environmental Mining Pollution Monitoring System

Mining server as a key economic driver for South Africa's economy. The economic advantages are accompanied by significant drawbacks, including detrimental effects on the local environment, and the health of nearby residents due to mining activities (Feris & Kotze, 2015). Pollution from mining involves the introduction of harmful substances or energy into water, soil, or air, potentially causing immediate or long-term harm to the planet's ecological balance or diminishing the quality of life (Coker et al., 2009). The World Health Organization (WHO) (Rimayi et al., 2018) reported that pollution is responsible for about a quarter of diseases worldwide. Pollution that comes from gold mining is mainly related to the release of harmful elements from tailings piles and other mining tailings (Fashola et al., 2016). It is estimated that gold mining waste accounts for 221 million tons or 47% of all mineral waste generated in South Africa, making it the country's largest source of waste and pollution (DWAF, 2001). Pollution

from mining activities around mining communities has negative environmental impacts such as acidic mine drainage and toxic waste (Feris & Kotze, 2015), such is the case in a mining community in the Lejweleputswa district of the Free State, South Africa.

Several mining pollution monitoring tools are used to monitor and alert on excessive pollution levels. A typical example in the context of water quality management is the South African Blue Drop System (SABDS) (Rivett et al., 2013). The Blue Drop system is now used in South Africa and around the world to ensure water quality management and sustainably improve wastewater management. The Blue Drop system is designed to provide effective, efficient water and sanitation services to communities (Rivett et al., 2013). The disadvantage of this system is that it requires scientific interpretation of the results using a set of criteria called the Blue Drop Risk Rating (BDRR) tool, which identifies, quantifies, and manages risks associated with the provision of drinking water services in nine provinces of South Africa (Rivett et al., 2013) and incompatibility with other early warning systems within the domain of environmental monitoring.

2.2.3.3 Local Indigenous Knowledge and Scientific Knowledge on Environmental Pollution

Over the years, people have developed knowledge from their understanding of their environment, and this is known as environmental knowledge. They have relied more on observing and monitoring environmental changes as they have observed and experienced them over the years (Mbele et al., 2017). Local knowledge sometimes called indigenous knowledge or traditional ecological knowledge is the body of knowledge, practices, and beliefs related to the biophysical environment and human involvement in it. It is acquired through observation and personal experience, but also passed on through social networks (Hopping et al., 2016), including learning from elders, participation in community management institutions, natural resources, and discussion with peers (Hopping et al., 2016), This knowledge is passed on from generation to generation (Akanbi & Masinde, 2020). Meanwhile, various kinds of pollutants had necessitated having a tailored treatment at their source and having some specific disposal techniques to mitigate their adverse effects on the environment. The problem with local knowledge is that it does not provide accurate situational awareness and cannot predict future environmental changes. The use of indigenous knowledge in environmental monitoring involves the use of local knowledge on climate, local weather, and observations.

Oppositely, scientific knowledge is advanced understanding based on strict scientific methods, including large-scale models and satellite data observation (Gbangou et al., 2021). This type of knowledge is often seen as the most reliable, precise, and useful for making informed decisions. However, in many parts of the world, monitoring systems are mainly run by indigenous people and local communities. These groups have unique knowledge systems and practices specific to their regions and cultures. Increasingly, research recognises indigenous and local knowledge as valuable for studies, policymaking, and ecosystem management (Gbangou et al., 2021; Tengö et al., 2021).

2.3 Middleware

Middleware is a type of software that serves as a bridge, facilitating communication and data handling between various applications or between users and applications. It sits between the operating system, and the application running on it, and essentially acting as a hidden translation layer (Akanbi & Masinde, 2018; Farsi et al., 2019), see Figure 2.5 below. Middleware enables communication and data management in distributed and heterogeneous applications. The middleware layer provides a set of programming abstractions to facilitate the integration and communication of heterogeneous components (Akanbi & Masinde, 2018). Additionally, middleware is an important component of architectural software that supports distributed systems and allows application engineers and developers to focus on application requirements and delivery. It offers developers a solution to integrate a range of systems and applications through a common service interface (Emmerich, 1997).

Noticeably, middleware can exist between all software layers and provides many services to achieve interoperability and real-time information flow. The use of semantic technologies in the context of middleware applications helps to exploit the complexity of heterogeneous applications and achieve data integration and system interoperability (Zgheib et al., 2019). Semantic technology uses data and knowledge representation through a range of methods and tools to discover deep relationships within different objects and categorized datasets (Rumiński & Walczak, 2020). The aim of incorporating semantic technology into middleware is to uncover the fundamental relationship between objects, allow data to be presented in a way that machines can read and support the integration of data with interoperability across heterogeneous applications in a multi system (Akanbi & Masinde, 2018).

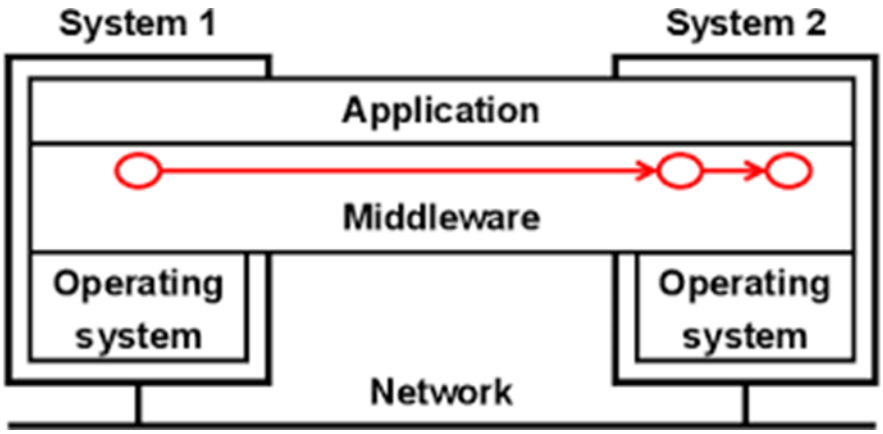


Figure 2.5: Middleware Overview (Source: Emmerich, 1997)

2.3.1 Middleware Requirements

The middleware should build modularity, high reliability, high scalability, maintainability, and user-friendliness and support rapid development. Based on the attributes mentioned, this section lists the requirements for the middleware and its application as follows.

- Interoperability: Research indicates that the diversity within the Internet of Things (IoT) necessitates addressing interoperability issues in IoT middleware. Interoperability can be viewed through both syntactic and semantic lenses. Syntactic interoperability refers to the use of different data access formats, protocols, and service description languages to facilitate information exchange and connect services. On the other hand, semantic interoperability involves employing standardized semantic technologies like XML and Ontology to accurately represent data, the relationships between components, and their interactions (Zhang et al., 2021).
- Reliability: Middleware must provide users and developers with a stable and reliable experience while the system is running, even if the system encounters issues. This reliability should be evident in areas such as communication, data, calculations, and more (Zhang et al., 2021).
- Real-Time: The dependability of a system hinges not only on its logical accuracy but also on its operational speed. For applications that are sensitive to time, it is essential to guarantee real-time data delivery and prompt decision-making. Moreover, users must have the capability to set deadlines for packet delivery.

- Scalability: Adjusting the number of nodes or services and enhancing functionality by adding components should not affect the middleware. The middleware should maintain stability in EWS applications despite these changes (Alonso et al., 2018). The scalability of the system can be achieved through middleware.
- Security: EWS data is considered sensitive data and must be protected. Data is often dirty in too many forms and formats such as images or files, may be incomplete or missing entirely, containing too many errors, files are not properly tracked, or are unstructured. Therefore, middleware must be able to protect the information sent over the network (Alonso et al., 2018).

Interoperability and security requirements are mandatory, but the purpose of the middleware is to enhance automatic system configuration for data integration and system interoperability. Middleware can adapt to heterogeneous environments (Bamhdi, 2021). With the rise in multimedia big data, energy-efficient processing has become a critical challenge. The growing number of multimedia applications in IoT has therefore led to the use of heterogeneity, which is seen as a key issue for the future of IoT (Aslam & Curry, 2021).

2.3.2 Middleware Approaches

This section categorizes IoT middleware approaches into five types: database-based, agent-based, virtual machine-based, event-based, and service-based (Zhang et al., 2021). The next subsections briefly discuss each approach.

2.3.2.1 Database-Based Approach

Database Driven Middleware (DDM) conceptualizes network nodes as distributed data objects and views the network itself as a virtual distributed relational database (Pradeep et al., 2021) . Users request data with SQL-like queries, which the system parses to create a network query template. This template is then divided into subqueries sent to the relevant nodes in the network. Once these requests are accepted, the sensor nodes process the network data and combine the information before sending the results back to the users (Pradeep et al., 2021). DDM supports the detection of temporal relationships between data and events. However, a global network structure must be maintained, thus limits scalability. In addition, DDM often lacks data aggregation and knowledge discovery capabilities.

2.3.2.2 Agent Approach

A Mobile Agent (MA) is a self-sufficient software program that operates independently without requiring user input (Zhang et al., 2021). The agents need to run in a specific virtual machine

environment, so Agent-Based Middleware (ABM) can be considered a special case of virtual machine-based middleware agent. Agent applications are split into separate programs that can be easily inserted into a network for smoother operation and deployment using mobile agents and ensure efficient agent operations (Badica et al., 2019).

2.3.2.3 Virtual Machine-Based Approach

Virtual Machine-Based Middleware comprises Virtual Machines (VMs) and interpreters that create an operational environment for applications (Pradeep et al., 2021). These applications are usually divided into modules, which are then distributed across the network. Each network node has a virtual machine that interprets these modules (Pradeep et al., 2021). The virtual machine sets a series of instructions based on the operating system's interfaces. However, using virtual machines is not a cost-efficient way to handle the diverse nature of IoT infrastructure, as prioritizing portability over performance diminishes flexibility.

2.3.2.4 Event-Based Approach

In Event-Based Middleware, all elements including components, applications, and participants interact through events. Message-Oriented Middleware (MOM) is a type of middleware that relies on messages for communication and adheres to the publish/subscribe model (Aslam & Curry, 2021) Event processing systems handle user subscriptions using standard languages to respond to events and facilitate many-to-many communication (Aslam & Curry, 2021). Nevertheless, system interoperability, adaptability, execution speed, and context sensitivity are not sufficiently addressed, and the programming paradigm is not flexible enough, making the developer's task difficult.

2.3.2.5 Service-Based Approach

The Service-Based Middleware (SBM) architecture draws its inspiration from the conventional Service-Oriented Architecture (SOA). It involves dividing resources into services and merging them with pre-existing internet services to furnish users with unified computing capabilities (Zhang et al., 2021). Generally deployed in cloud-enabled settings, SBM proves beneficial in addressing issues of heterogeneity and interoperability by facilitating adaptable service compositions in cases where certain services are absent. SBM features a service layer that relies heavily on middleware technologies to support service providers and users, which helps abstract system complexity. Finally, according to Aslam and Curry (2021), and Mesmoudi et al. (2020), in terms of interoperability and high scalability, SBM outperforms other solutions. Therefore, SOA is considered a good solution for IoT middleware to address heterogeneity and

interoperability issues (see Table 2-1 below). However, SBM offers only limited security through authentication, therefore service-based solutions must be self-adaptive (Zhang et al., 2021).

Comparison of existing IoT middleware.

Type	Middleware	Platform	Interoperability[a]	Real-timeness	Scalability	Security	Light-weightness	Context awareness	Knowledge discovery	self-adaptability
Service based	Hydra	Sensor node	SE	√	√	√	√	√	×	×
	CoCaMAAL	Sensor node, cloud	SY	√	√	×	×	√	×	×
	BDCaM	Sensor node, cloud	SE	√	√	×	×	√	√	×
	Atlas	Sensor node	SE	√	√	√	×	√	×	√
Event based	IoT-MP	Sensor node, cloud	SY	√	×	√	×	×	×	×
	PRISMA	Sensor node	SY	√	×	×	×	√	×	√
	SeCoMan	Sensor node, cloud	SE	√	×	√	×	√	√	×
	SenSocial	Mobile phone	SY	√	×	√	√	×	×	×
VM based	Mate	TinyOS	SY	√	√	×	√	√	×	√
	SensorWare	Linux, ARM platform	SY	√	×	×	√	√	×	×
	Actinium	JVM	SY	×	√	√	×	×	×	√
	MODE	JVM	SY	√	√	×	×	√	×	√
Database based	TinyDB	TinyOS	×	√	×	×	√	×	×	×
	SINA	Sensor node	×	√	×	×	√	×	×	×
	Cougar	Berkeley MICA, Unix	×	√	×	√	√	×	×	×
	DSWare	JVM	SY	√	×	×	√	×	×	×
Agent based	Agilla	MICA2 Mote, TinyOS	SE	√	×	×	√	√	×	√
	Eagilla	Sensor node	SE	√	√	√	√	√	×	√
	ACOSO	Smart object	SY	√	√	√	√	√	×	√
	Sensomax	JVM, ARM platform	SY	√	√	√	√	×	×	√

Table 2-.1: Comparison of existing IoT Middleware (Zhang et al., 2021)

2.3.3 Middleware Technologies

Different types of middleware technologies can be classified for distributed systems and heterogeneous components such as Object-Oriented Component Middleware (OOCM), Message Oriented Middleware (MOM), and Transaction-Oriented Middleware (TOM), just to mention a few (Emmerich, 1997). The Service-Oriented Architecture middleware (SOAM) is recommended for this study (see Figure 2.6 below).

2.3.3.1 Service-Oriented Architecture Middleware (SOAM)

SOAM refers to a software design approach facilitating communication between hardware and software applications. It achieves this through a secure middleware product and technology layer, independent of specific protocols (Reis & Gonçalves, 2018). The middleware serves as an interface or a link that connects heterogeneous domains of applications over heterogeneous interfaces and client applications (Akanbi & Masinde, 2020). In distributed environmental

monitoring systems, many applications need real-time monitoring that is highly accurate, performs well, and is reliable. There is a need to develop a middleware with a design based on SOA with semantic technology that tackles the difficulties associated with their integration, representation, and reliability. Both structured data and unstructured data can be integrated by the middleware through efficient semantic representation of data in machine-readable languages. The middleware furnishes an API for communication at the physical layer, simplifies intricate network communication, and formats the data in a machine-readable way for convenient utilization and interoperability (Justin Dhas & Jeyanthi, 2019).

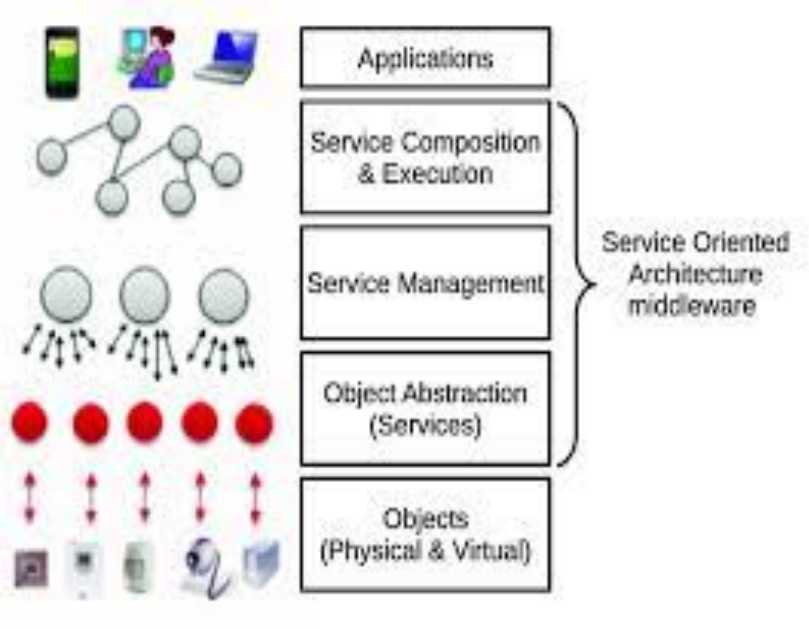


Figure 2.6: Service-oriented Architecture-based middleware (Source: Wortmann & Flüchter, 2015).

2.3.3.2 Semantic Middleware

Using semantic technologies in middleware applications simplifies the complexity of heterogeneous systems by enabling data integration and ensuring systems can work together effectively (Zgheib et al., 2019). Semantic technology employs data and knowledge representation using a collection of techniques and tools for uncovering detailed connections among diverse groups of objects and data sets (Rumiński & Walczak, 2020). By embedding semantic technology within middleware, the aim is to delve into underlying object-level relationships, ensure machine-readable data representation, and promote the integration of data

with interoperability across heterogeneous applications in multi-system environments (Akanbi & Masinde, 2018).

Querying data and facilitating information exchange among the various subsystems of a complex hierarchical system poses a difficulty because of the extensive array of diverse components each with its unique traits. To tackle this challenge, efforts in research and standardization have primarily concentrated on developing models and representing data through a fundamental ontology. Given the challenge of interoperability, since different devices will need to communicate and share information with each other, the application of semantic techniques such are knowledge representation through an ontology using Ontology Web Language (OWL), Resource Description Language (RDF), and Web Services Description Language (WSDL) through a Service-Oriented Middleware (SOM) provides the necessary adaptation layer in a plug-n-play mode (Rodrigues et al., 2019) or standardized languages like XML, Web Service Definition Language (WSDL) and JavaScript Object Notation (JSON).

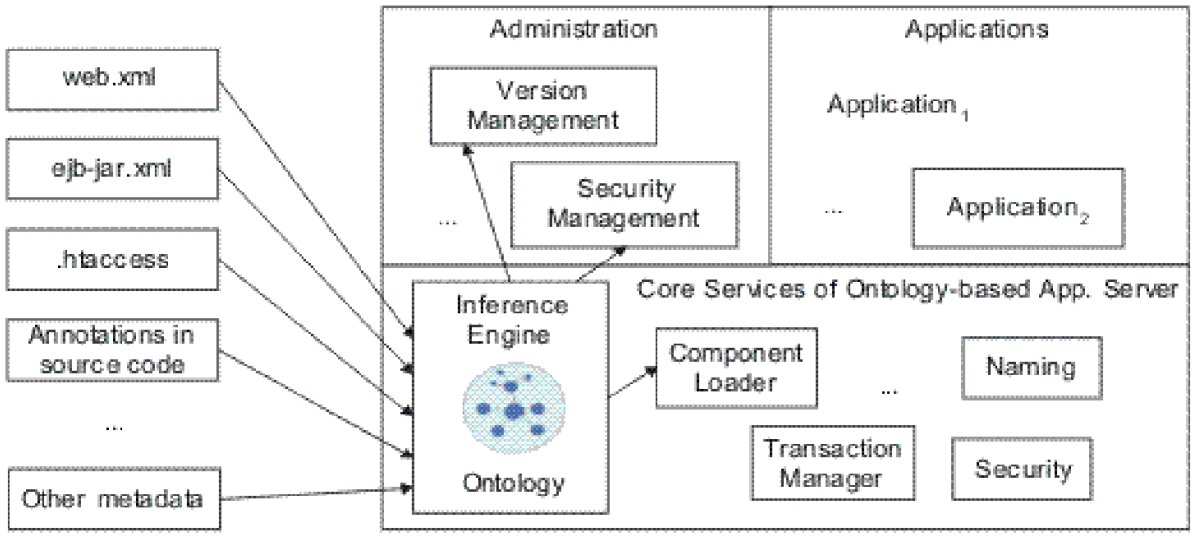


Figure 2.7: Sample semantic middleware architecture using ontology repository for data representation (Zgheib et al., 2019)

2.4 Cloud Computing

Cloud computing is presented as a next-generation technology. It is an internet technology that provides users with high-quality services, including data and software on remote servers (Srivastava & Khan, 2018). It offers customers numerous options such as accessing, purchasing, installing, or downloading any of a variety of unlicensed applications. Cloud

computing users have the ability to retrieve files and utilise applications from any device with internet access through a standard web browser. The customers only pay for the services they use because they do not have to buy any infrastructure. Furthermore, cloud computing provides enhanced security through data encryption, robust access controls, key management, and security analytics (Srivastava & Khan, 2018). One major advantage of cloud computing is the capability to share resources like hardware and software over a network with more storage space than traditional storage systems.

Cloud computing has gained significant traction in both industry and organizations owing to its practicality and straightforward cloud service-oriented frameworks. Distributed computing encompasses two primary models: Service Models, and Deployment Models. The three service models include Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure-as-a-Service (IaaS). In addition, there are four distribution models, namely public, private, and hybrid clouds.

2.4.1 Service Models

Cloud computing service models are divided into three basic models as mentioned previously. Understanding these models and the differences between them could enable the two adopted systems to achieve their goals and the research aim to solve heterogeneity problems.

2.4.1.1 Software as a Service (SaaS)

Software as a Service (SaaS) is an application service hosted on a centrally managed cloud infrastructure and charged on a subscription basis (Yoo & Kim, 2018). SaaS enables users to access and utilize applications operating on cloud infrastructure, with the service provider handling the management of all underlying infrastructure, platforms, application software, and data. This includes taking care of the hardware and software as well as ensuring the application's availability and security (Pham, 2020). Applications like Gmail and Google Docs, hosted on the cloud can be accessed from various devices such as smartphones and laptops. From the customer's perspective, the SaaS model appears as a web-based application that delivers services through an internet browser.

2.4.1.2 Platform as a Service (PaaS)

Platform as a Service (PaaS) provides the user with a complete cloud development and deployment environment to deploy anything from simple cloud-based applications to cloud-enabled enterprise applications using software programming languages, libraries, frameworks, and tools supported by the provider (Pham, 2020; Srivastava & Khan, 2018). It can run

software solutions on the cloud computing platform without the support of a developer to provide hardware and software for the cloud computing system. PaaS is widely used by organisations in specific scenarios such as development, analytics, and business intelligence frameworks.

2.4.1.3 Infrastructure as a Service (IaaS)

In Infrastructure as a Service (IaaS), a cloud provider supplies virtualized computing resources, including servers, operating systems, memory, storage, and application software, through the cloud. By leveraging virtualization technology, IaaS transforms physical resources into logical ones that users can access dynamically as needed (Rashid & Chaturvedi, 2019). This model allows for rapid scaling and a pay-as-you-go payment structure. By using cloud infrastructure, you can bypass the expenses and complexities associated with buying and managing your own physical servers and data center hardware (Pham, 2020).

2.4.2 Deployment Models

There are four types of cloud deployment models: public, private, community, and hybrid. Each model is defined based on the location of the infrastructure supporting the environment.

2.4.2.1 Public Cloud

This cloud is available on the internet to all customers who can subscribe to the cloud and use cloud resources on a consumption basis. This cloud is not as secure as a private cloud (Rashid & Chaturvedi, 2019), but due to its openness, it is accessible to all internet users. It is relatively less configurable than a private cloud. The cloud infrastructure is owned and operated by a major Cloud Service Provider (CSP) (Srivastava & Khan, 2018). It offers infrastructure and services to individuals and organizations, with resources being shared by many users. Public cloud services are currently available from providers like Google Cloud Platform (GCP), Amazon Web Services (AWS), and Microsoft Azure (MA) to mention a few.

2.4.2.2 Private Cloud

This type of cloud is designed for individual organizations or businesses. It utilizes internal computing resources exclusively for one organization, keeping them inaccessible to the public. These resources can be located in a physical data center at the organisation's site or hosted by a third-party provider (Rashid & Chaturvedi, 2019). Private clouds are commonly employed by financial institutions, government bodies, and medium to large companies to ensure a more secure and controlled environment.

2.4.2.3 Hybrid Cloud

This type of cloud combines both private and public clouds. Despite their integration, each maintains its own identity, allowing for various deployments (Rashid & Chaturvedi, 2019). Customers can opt for a public cloud for tasks needing extensive storage and lower security, while a private cloud is suitable for sensitive data or confidential internal operations (Pham, 2020).

2.4.3 Cloud Computing Characteristics

There are several essential characteristics of cloud computing, among which are on-demand self-service, broad network access, measured service, maintenance, and resource pooling. These characteristics are highlighted below.

- On-Demand Self-Service: Cloud services such as server time, storage, web access, CUP time, and web applications can be automatically allocated based on customer demand without requiring human interaction (Rashid & Chaturvedi, 2019).
- Broad Network Access: Cloud computing resources are accessible over the network and can be utilised via different client platforms, including mobile phones and laptops, as long as there is a stable internet connection. This enhances the user experience when using cloud services online (Pham, 2020).
- Measured Services: Cloud resources and services are managed, regulated, and optimized by the cloud service providers (CSPs) under a pay-per-use business model. Users utilise these services similarly to how they consume utilities like electricity, water, and gas (Rashid & Chaturvedi, 2019).
- Maintenance: Cloud computing applications are simpler to maintain since they don't require installation on each individual computer and can be accessed from various locations, leading to cost savings. (Srivastava & Khan, 2018).
- Resource pooling: Cloud computing resources are shared among multiple users through a multi-tenant model. This approach enables the cloud service provider to dynamically distribute both physical and virtual resources according to user demand, ensuring the privacy and security of each client's data. Cloud service providers typically have extensive and adaptable resource pools to accommodate customer needs and benefit from economies of scale (Pham, 2020).

2.4.4 Microservices

The conventional monolithic application architecture is being replaced by the more adaptable and modular microservices architecture. Microservice applications are typically developed on Microsoft Azure's public cloud infrastructure, utilising services like Azure Service Fabric (ASF), Azure Kubernetes Service (AKS), and Azure Functions (Microsoft, 2017). Microservices are created to deliver specific functions and can be developed and deployed independently with ease, see Figure 2.9 below.

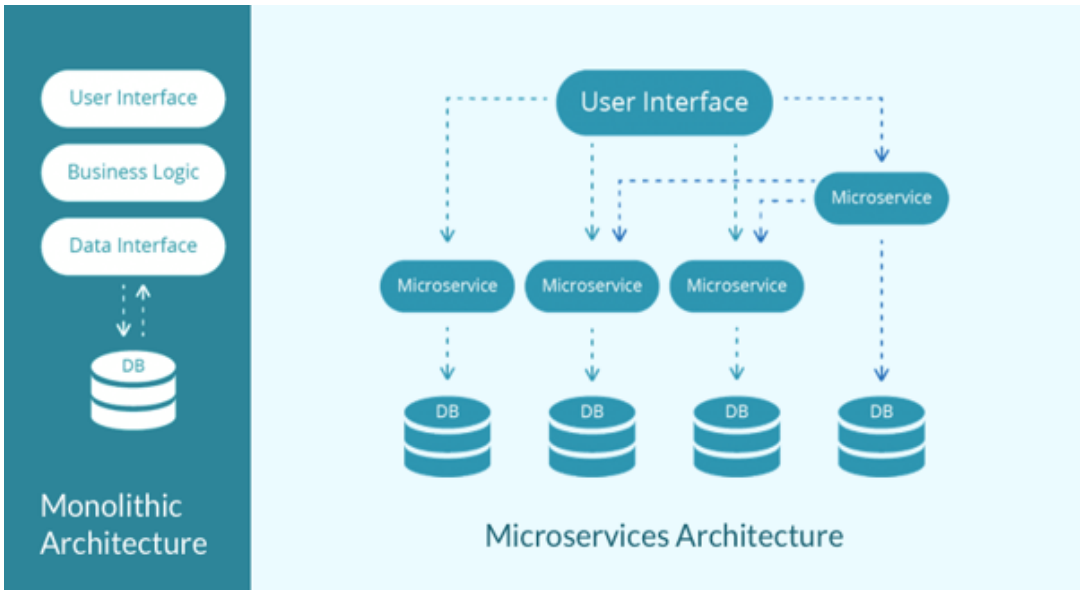


Figure 2.8: Monolithic approach vs Microservices approach (Source Author)

The two systems: AEMS, and ITIKI operate on a local host, which can make accessing them challenging at times. Incorporating cloud computing offers a reliable and scalable way to deliver information and applications via the Internet (Younan et al., 2020). Recently, numerous industries and organizations have embraced cloud computing due to its accessible and convenient service-oriented models. Distributed computing comes in two forms, namely Service Models, and Deployment Models. Cloud computing enables users to access files and applications from any device using a standard internet browser.

In our proposal, the middleware layer functions as a microservice that processes incoming data from inputs and directs the resulting data to the next destination through the outputs. The concept of a microservice has various definitions and purposes (Namiot & Sneps-Sneppe, 2014; Sun et al., 2017), describing it as a new approach to software development designed for

distributed applications like web and cloud-based systems. Simply put, microservices are small, independent services that collaborate.

The microservice architecture offers benefits like improved resilience, technological flexibility, enhanced scalability, and speed of application operations. However, it also brings challenges such as increased development, implementation complexity, the need for service-to-service communication mechanisms, and difficulties in performing end-to-end testing (Petcu, 2021). To minimize deployment time, microservices containers can be used. Containers streamline the execution, relocation, and redeployment of microservices in different environments, making them a viable alternative to traditional virtualization.

2.4.4.1 Microservice Deployment

Deploying a monolithic architecture is relatively straightforward because developers deploy the entire application on a single physical or virtual server (Petcu, 2021). When an application needs to run on multiple servers, a common practice is to install the application on each server and use a load balancer to distribute the workload across the different copies (Petcu, 2021). Microservices architecture, on the other hand, relies on loose coupling, which requires running the application across multiple service instances (Akanbi & Masinde, 2020) This can be accomplished by deploying each service on a separate virtual machine, utilizing a container for each service, or deploying multiple services within a single virtual machine or container. Containerization provides advantages like quicker deployment, enhanced flexibility, and lower resource consumption. Moreover, it enables each microservice instance to operate independently on a host. In this study, both the Kubernetes container orchestration platform and Docker-based containerization tools are employed.

Importantly, the Docker tool simplifies the process of creating and launching containers. As modern applications use more containers and distributed servers, container orchestration technologies have become increasingly crucial for managing and deploying these complex systems. Container orchestration allows for the selection, deployment, monitoring, and dynamic management of multi-container applications (Al Jawarneh et al., 2019). Kubernetes, an open-source solution, was developed to automatically deploy and manage clusters of Docker containers. This means that service developers can create Docker images containing the desired components, which Kubernetes then deploys, manages, and monitors, including the relationships between these components (Kubernetes, 2019).

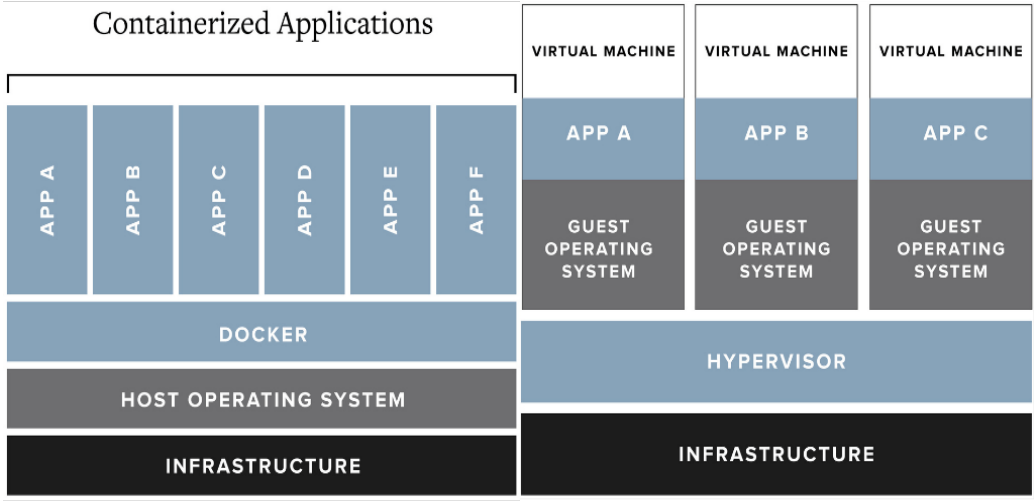


Figure 2.9: Docker Container Model and Virtual Machines (Source: Author)

2.5 Related Works

So far, many excellent studies have been conducted, and some approaches have been made in the field of IoT Middleware to address critical challenges such as high communication cost, very heterogeneous data and complexity of data representation for translation and usability.

The IOT faces challenges related to heterogeneity, and researchers have proposed solutions to address these issues. A middleware based on a Service-Oriented Architecture for heterogeneity problems on the Internet of Things (MSOAH-IoT), is proposed to solve heterogeneity problems (Mesmoudi et al., 2020). The methodology is implemented in three phases: beginning with data collection from heterogenous hardware sensors using the REST APIs, secondly introducing heterogeneous network interfaces, and finally testing the middleware on gateways running on different operating systems. The middleware searches for connected devices through various network interfaces and provides its services through a website using REST APIs. Notwithstanding, the middleware only works on Windows and Linux platforms. Therefore, it does not offer full interoperability with other operating systems such as Android and IOS platforms.

Additionally, a Lightweight Middleware platform designed for distributed computing in wireless sensor networks aims to support the development of distributed applications in these environments (Gaglio et al., 2014). This platform distinguishes itself from other middleware solutions through its simplicity. The research highlights that FORTH is a software development

methodology and a language that merges the benefits of various languages with pure assembly language. FORTH utilizes a layered design approach that seamlessly integrates high and low-level programming without a distinct separation, avoiding a minimal software layer between the hardware and application layers (Gaglio et al., 2014). Its scalability is notable, as new words can be easily added to the language, and the middleware platform can be readily extended. FORTH maintains a high level of expressiveness since computations are described simply as sequences of word names, making it readable like natural language, and suitable as a metalanguage for developing high-level languages. This middleware enables interactive and step-by-step application development, even on distributed nodes accessible only via wireless connections, eliminating the need for cross-compilation and significantly reducing development time.

A new lightweight SOA integration model based on a set of communication techniques that improves efficiency at all communication levels in the production plant was introduced by (Espí-Beltrán et al., 2017). They assessed the feasibility of an SOA approach by bypassing the WS-* stack in favour of other protocols based on the REST design style. To validate their proposal, a prototype of a Lightweight SOA RESTful platform was deployed in a production environment. The experiments used the Hypertext Transfer Protocol (HTTP) and the Constrained Application Protocol (CoAP), each with fundamental characteristics. CoAP proved to be the fastest for fine-grained communication components in near-field and control networks. However, HTTP, being the most commonly used protocol in RESTful web services, is advantageous for high-level communication near ICT systems due to its widespread adoption. Although HTTP is slower than CoAP, it still performs adequately to transfer data within the required time frame for most industrial installations.

According to Pradeep et al. (2021), the Adaptive Ubiquitous Middleware-driven context-aware IoT ecosystem (AUM-IoT) is a sophisticated and comprehensive system where users, data sources, intelligent objects, and services interact seamlessly. This ecosystem ensures secure device communication, efficient message flow management, data handling, and delivery of context-aware services tailored to users' needs. The AUM-IoT architecture consists of perception, middleware, fog cloud, and application layers. AUM supports the real-time monitoring of diverse elements, intelligent, and automated context-aware management, optimal provision of contextual, security, and privacy services, and timely information access. It employs a multi-agent, multi-protocol middleware, utilizing agents for effective

communication management and protocol handling to offer superior contextual services to users.

2.6 Summary

Chapter Two provided an extensive literature review covering various aspects of early warning systems and middleware technologies for effective environmental monitoring. The study of EWS and MHEWS highlighted their essential role in risk reduction and disaster management. The integration of local indigenous knowledge with scientific knowledge has demonstrated a holistic approach to pollution assessment. A review of middleware technologies, including requirements, approaches, and semantic middleware, highlighted their importance in facilitating seamless communication and data processing. In addition, related work shows the evolving landscape of cloud computing and microservices to increase the scalability and availability of environmental monitoring systems. This chapter served as a foundation for the upcoming chapters by providing an overview of the existing body of knowledge. The next chapter examines the development of an innovative and effective environmental monitoring framework.

CHAPTER THREE

RESEARCH METHODOLOGY AND DESIGN

3.1. Introduction

This chapter provides a detailed explanation of the research methodology and design employed in this study. It outlines the steps taken to address the research design, as well as describe the methods used for data collection, the types of data gathered, and the analysis techniques applied.

The term Research Methodology" can be broken down into two components: "research", and "methodology." Research involves gathering, analysing, and interpreting data to address inquiries, employing scientific approaches to tackle issues, and generating new universally relevant knowledge. Scientific methodologies encompass structured observation, categorization, and data interpretation (Goundar, 2012). Methodology, in contrast, refers to the structured and theoretical examination of the techniques used within a particular area of study. It involves analysing the underlying methods and principles within a field of knowledge, encompassing elements like framework development, theoretical models, stages, and both quantitative and qualitative approaches (Igwenagu, 2016).

According to Sarantakos (1998), research methodology is the theory of methods, it is the way in which one makes sense of the object of inquiry. A problem identified in Chapter One, aims to simplify data representation for easier translation and integration of the Pollution Early Warning System (PEWS) with other EWS systems within the context of the MHEWS. To address the issue, a theory is put forward suggesting that by employing semantic middleware, the difficulties associated with data representation, integration, and system interoperability in an MHEWS can be tackled. This section outlines our approach to testing this theory.

3.2 Research Design

Research design is the core planning of the research study it gives direction to the study, and it is used as a guideline in collecting and analysing the data. The study follows an experimental approach with the development of a semantic middleware for a MHEWS using Lejweleputswa, Free State Province as a case study. To achieve the objectives mentioned in Chapter One, the study employed the Design Science Research (DSR) methodology that creates and evaluates innovative frameworks and prototypes involving data collection, data representation tools,

framework development, middleware layer development, and prototyping of a fully integrated and interoperable system. Design Science Research (DSR) is a systematic methodology used in information systems and related fields to create and evaluate artifacts, such as models, frameworks, or systems, aimed at solving identified organisational or technical problems. In this research, DSR provides the foundational methodology for the development and validation of semantic middleware within the Multi-Hazard Early Warning System (MHEWS). A semantic middleware was developed and implemented for optimising the automatic configuration of an EWS and achieving the federation of heterogeneous data.

3.2.1 Mixed Methods

This research utilised a mixed methods design, incorporating both qualitative and quantitative approaches to achieve its objectives. The qualitative approach involves gathering, analysing, and interpreting data based on observations of people's statements and behaviours (Goundar, 2012). In this study, it was employed to gain an in-depth understanding of the application of local Indigenous Knowledge on pollution. This was achieved through unstructured and structured data collection methods, including in-depth interviews and focus groups. A quantitative methodology stands as a frequently employed strategy in research design, rooted in positivism. Essentially, it involves a number-centric investigative discipline that statically gauges attitudes, behaviours, and performance, yielding results in percentages for straightforward interpretation. (Goundar, 2012). It can translate data into graphs and charts that are statically valid. Thus, this approach is used to test the hypothesis mentioned in Chapter One to archive findings with an acceptable degree of accuracy. As a result, quantitative research can be used in the collection and analysis of data.

The mixed methods approach is an ideal methodological design for this study, given its ability to capture the multifaceted nature of indigenous knowledge. This approach allows for a comprehensive understanding by combining the strengths of both qualitative and quantitative research methodologies. The qualitative component is particularly essential for exploring the depth and context of indigenous knowledge systems, which are often embedded in cultural practices, oral traditions, and community-specific experiences. Methods such as in-depth interviews, participant observations, and storytelling techniques provide an opportunity to capture the richness and complexity of indigenous knowledge. On the other hand, the quantitative component enhances the study by enabling the identification of broader patterns and trends, such as the prevalence and impact of indigenous practices across different communities or regions. The integration of both qualitative and quantitative data also ensures

triangulation, which strengthens the validity and reliability of the findings. Given the unique context of indigenous knowledge, which is often localised and context-specific, the mixed methods approach ensures that both individual and collective experiences are adequately represented, while also providing generalisable insights. This methodological approach aligns closely with the study's objectives of documenting, preserving, and assessing the value of indigenous knowledge, and it allows for a more holistic understanding of its significance in addressing contemporary global challenges.

3.2.2 Alignment of Methodology with Research Questions

The research problem identified in Chapter one focuses on the challenges of data representation and integration within MHEWS. These challenges hinder the system's ability to deliver timely and accurate multi-hazard alerts. The research questions aim to address this problem by guiding the development and evaluation of a semantic middleware solution. To ensure that the mixed methods approach effectively addresses the research objectives, Table 3.1 outlines how each research question is aligned with the methodologies used, demonstrating the rationale behind the integration of qualitative and quantitative methods.

Methodology Aspect	**Research Question Addressed**	**Explanation**
Literature Review	**RQ3:** What are the existing semantic middleware and frameworks that can be adapted for South Africa's MHEWS?	Identifies current semantic middleware and frameworks, evaluates their suitability, and highlights gaps in existing solutions.
System Integration Analysis	**RQ1:** How does the integration of existing monitoring systems enhance the effectiveness and efficiency of the MHEWS?	Examines how integrating existing systems impacts MHEWS performance compared to traditional approaches.
Middleware Design and Development	**RQ2:** How can semantic middleware enhance the decision-making process for disaster risk reduction and response?	Develops middleware to improve data integration and provides actionable insights to inform disaster response decisions.

Comparative Framework Evaluation	**RQ3:** What are the existing semantic middleware and frameworks, and how do they compare?	Analyses and compares frameworks to determine the most suitable for South Africa's MHEWS needs.
Validation through Case Study	**RQ4:** To what extent will semantic middleware improve the efficiency of MHEWS in information integration and alerts?	Tests the middleware's real-world application to measure improvements in interoperability, timeliness, and warning deliver

Table 3-1: Methodology Alignment with Research Questions (Source: Author).

This alignment highlights the rationale for adopting a mixed methods approach and demonstrates how the integration of qualitative and quantitative data ensures a thorough investigation of the research problem.

3.3 Data Collection and Analysis Methods

Data may be gathered using both primary and secondary data collection techniques. Primary data refers to information collected explicitly for the ongoing research endeavour, acquired through two research approaches, surveys and case studies (Akanbi & Masinde, 2020). Secondary data refers to information that has been previously gathered in a different context from the current study. It serves to furnish essential background details and aids in elucidating the problem during the initial phases of exploratory research.(Akanbi & Masinde, 2020). The qualitative data delved into exploring community insights on pollution and data collected from wireless sensor networks. Conversely, the quantitative data focuses on representing knowledge, processing data, and implementing semantic middleware on a cloud platform.

The qualitative data collected through interviews and focus groups, thematic analysis will be used as the primary analytical technique. This approach involves identifying, analysing, and reporting patterns or themes within the data. The process begins with familiarisation, where transcripts are reviewed in detail to gain a deep understanding of the data. Subsequently, initial codes will be generated to represent key features of the data, which will then be organised into broader themes that align with the research objectives. The coding process will follow an inductive approach, allowing themes to emerge organically from the data without preconceived categories, ensuring that the findings remain grounded in participants' experiences. The final

themes will be interpreted within the context of the study, linking them to existing literature and theoretical frameworks.

Quantitative data will be analysed using statistical methods to address the research questions and test the study's hypotheses. Descriptive statistics, such as frequencies, percentages, and measures of central tendency, will be used to summarize the data and provide an overview of key variables. Inferential statistics, including regression analysis and correlation, will be employed to examine relationships between variables and identify significant trends or patterns. The statistical software package SPSS (Statistical Package for the Social Sciences) will be used for data analysis. The software's capabilities will allow for accurate calculations, efficient data management, and graphical representation of results. The analysis will also involve data cleaning and validation to ensure accuracy, with missing or inconsistent data handled appropriately to minimize biases.

3.3.1. Data Types

The system receives data in many forms structured (wireless sensor data) and unstructured (Indigenous knowledge) formats. This research study absorbed heterogeneous data from different sources for pollution monitoring system, pulled them together, and loaded into a database or data warehouse. The wireless sensor data collected was structured, and represented in data representations formats (JSON and XML). Indigenous knowledge is mostly unstructured and available in oral format and documentation. This data is captured and represented in a knowledge representation model.

3.3.2. Data Sources

3.3.2.1. Indigenous Knowledge Data Collection

Figure 3.1 below illustrates how local indigenous knowledge was acquired from community members through various methods, including focus groups, file uploads, and mobile surveys. This data was processed through a human interaction stage before being temporarily stored in IK General Storage. Using Extraction Tools, the data was refined and then transferred to Azure Data Lake (ADL) for scalable, long-term storage, and further analysis. This architecture supports the secure and efficient handling of both structured and unstructured data, ensuring it is prepared for analysis, decision-making, and reporting. The major components and their roles are discussed below.

- Focus Group: Represents qualitative data collected from group discussions or interviews to provide insights into human experiences or behaviours.
- File Upload: Allows structured or unstructured data, such as documents or reports, to be uploaded into the system for analysis.
- Mobile Survey: Collects real-time data from respondents via mobile devices, typically through survey applications or web forms.
- Human Processing: A stage where manual verification, validation, or cleaning of data takes place to ensure quality and integrity before storage.
- IK General Storage: Serves as an intermediate storage system for organizing, securing, and temporarily holding data before it is processed further.
- Extraction Tools: Software tools that process and transform the stored data, filtering and preparing it for final storage and analysis.
- Azure Data Lake (ADL): A cloud-based storage solution that offers scalable, secure, and cost-efficient storage for large volumes of data, making it accessible for advanced analytics and decision-making.

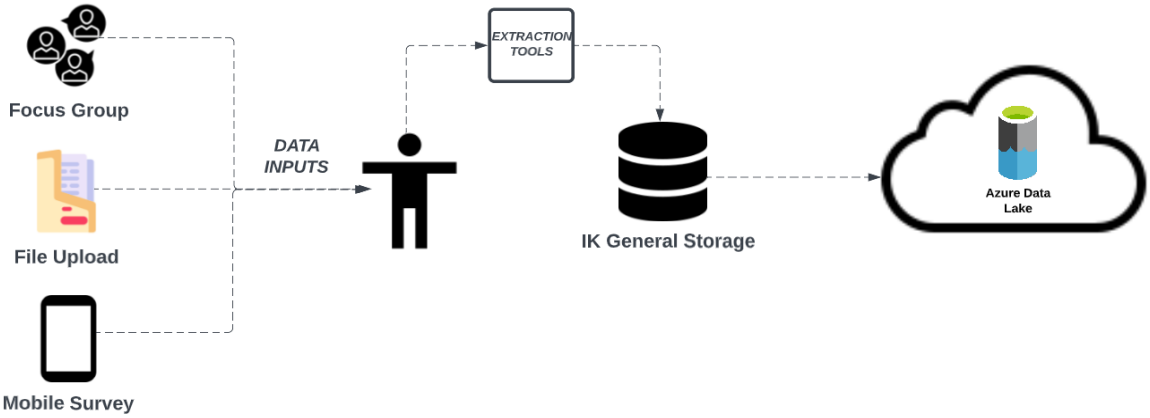


Figure 3.10: Indigenous Knowledge Data Collection (Source: Author)

3.3.2.2. Wireless Sensor Data Collection

Figure 3.2 below illustrates the architecture of a Wireless Sensor Network (WSN) designed to collect environmental or physical data from distributed IoT (Internet of Things) sensor devices,

process it locally, and transfer it to the Azure Data Lake (ADL) for long-term storage and analysis. This architecture is integral to the efficient functioning of multi-hazard early warning systems (MHEWS) and can be applied to critical monitoring tasks such as air quality measurement and other real-time environmental monitoring systems. Each component of this architecture plays a vital role in ensuring the seamless flow of data from collection to analysis. The major components and their roles are discussed below.

- IoT Sensor Devices: These sensors collect real-time environmental data and transmit it wirelessly.
- IoT Gateway: Acting as a bridge, the gateway aggregates sensor data, performs preliminary processing, and ensures secure communication between the sensors and the cloud.
- Database Storage: A temporary storage layer for buffering and pre-processing data before it is sent to the cloud.
- IoT Cloud (Azure Data Lake): A scalable, secure cloud storage solution where the data is stored long-term, enabling advanced analysis. The wireless sensor data is organized and presented in JSON and XML formats for easy interpretation.

This architecture ensures efficient, reliable data flow from collection to storage, and analysis, making it ideal for real-time monitoring in MHEWS.

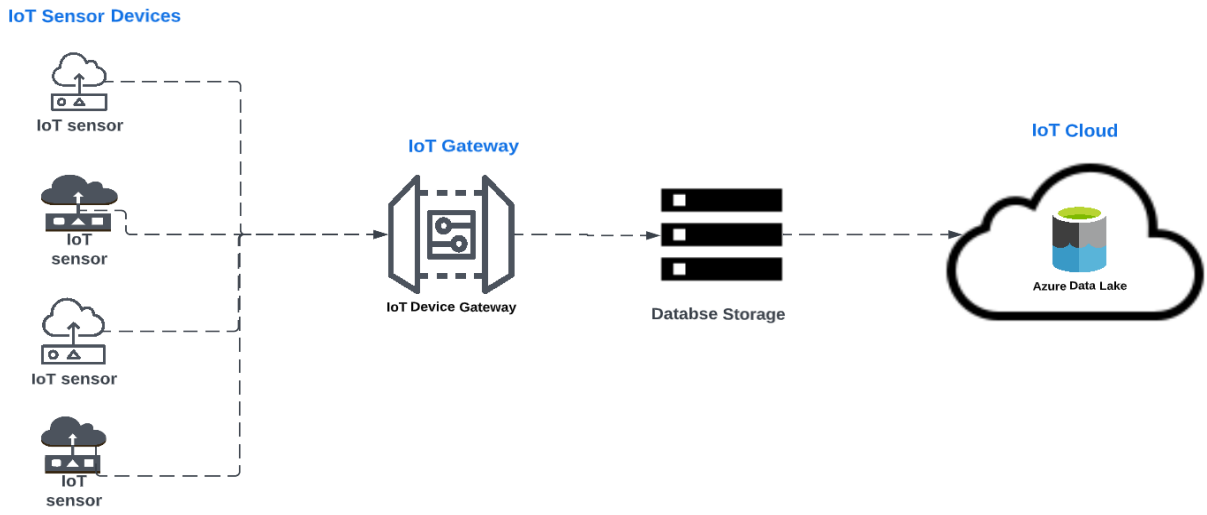


Figure 3.11: Wireless Sensor Data Collection (Source: Author)

3.3.3. Sampling Techniques

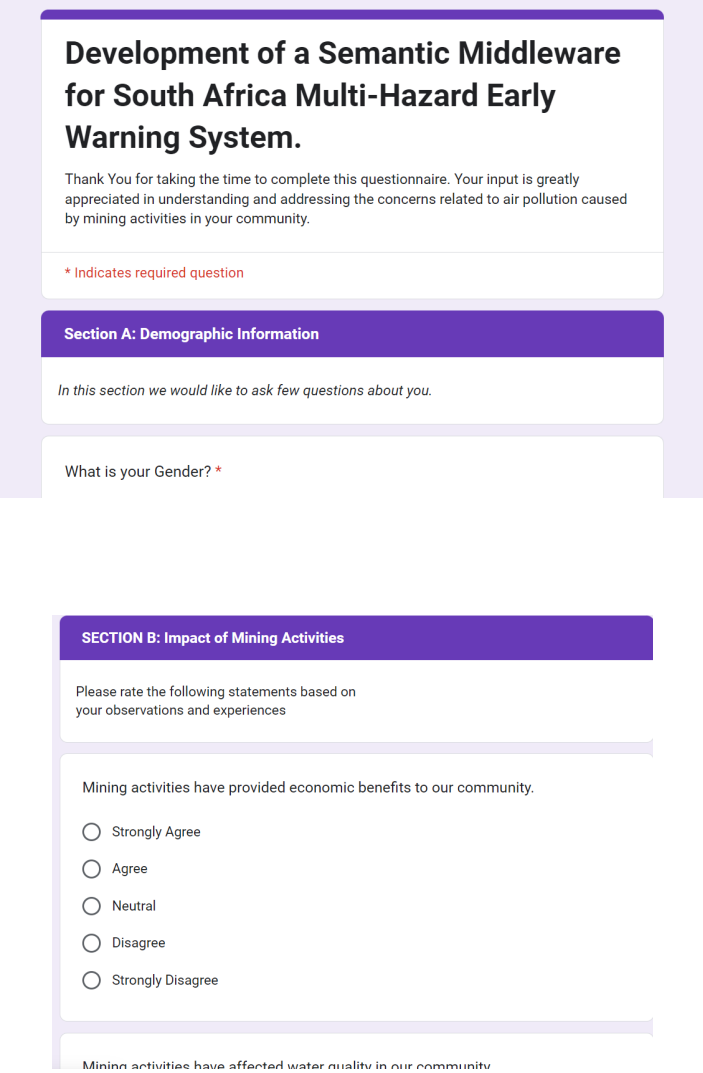
3.3.3.1. Questionnaire

To effectively collect qualitative data from key stakeholders, including students, local community members, and mine workers, regarding mining activities that contribute to air pollution, this study employed an online survey hosted on Google Forms. The use of Google Forms offers several advantages, including structured data collection and streamlined, automated analysis. This method allows participants to access and complete the questionnaire conveniently from any location, ensuring broad participation and flexibility. The participants recruited for this study were directed to the survey via a unique hyperlink provided in the research materials (see Appendix D for the full questionnaire).

Google Forms Link:
https://docs.google.com/forms/d/1cxZ2m9_ZTQPje9vDTzR9brUPhkAqICqCNhCsZxHIMCU/edit

The selection of Google Forms as a data collection tool aligns with the research emphasis on accessibility, ease of use, and data reliability. By leveraging the platforms capabilities, responses can be automatically aggregated and analysed, reducing potential biases and human error that can arise from manual data entry and processing. Furthermore, Google Forms ensures that data privacy is maintained while also enabling researchers to collect and organize a wide range of responses efficiently. This technological approach also allows for real-time tracking of responses, providing timely insights that can be used in preliminary analyses if needed. The questionnaire is divided into four segments, and each was designed to capture distinct yet interrelated aspects of the study's focus. The segments are:

- Segment A - Demographic Information
- Segment B – Impact of Mining Activities
- Segment C – Air Pollution
- Segment D – Pollution Early Warning System
- Segment E – Request for additional information



Development of a Semantic Middleware for South Africa Multi-Hazard Early Warning System.

Thank You for taking the time to complete this questionnaire. Your input is greatly appreciated in understanding and addressing the concerns related to air pollution caused by mining activities in your community.

* Indicates required question

Section A: Demographic Information

In this section we would like to ask few questions about you.

What is your Gender? *

SECTION B: Impact of Mining Activities

Please rate the following statements based on your observations and experiences

Mining activities have provided economic benefits to our community.

- Strongly Agree
- Agree
- Neutral
- Disagree
- Strongly Disagree

Mining activities have affected water quality in our community

Figure 3.12: Research Questionnaire (Source: Author)

3.3.3.2. Interviews and Focus Groups

Depending on participant selection and availability, video calls, voice calls, or personal interviews were conducted. Video calling solutions such as ZOOM, Microsoft Teams, and WhatsApp video calls were used for conducting the interviews. This session was audio-recorded to accurately obtain participants' comments, and to achieve detailed analysis.

3.4. Study Area

Lejweleputswa is one of the five districts in South Africa's Free State Province, widely recognized for its gold and diamond mining activities. The district comprises key towns such as Welkom, Virginia, Odendaalsrus, Allanridge, and Theunissen, forming the Free State Goldfields, where the majority of the province's gold and diamonds are mined. Agriculture and mining are the district's primary industries, contributing 28.6% to its economy (DDM, 2020).

As of 2019, Lejweleputswa District Municipality accounted for 22% of the Free State's total population, with 634,462 inhabitants. The district's population growth rate was 1.5% annually, with a projected average growth rate of 0.3% between 2019 and 2024, bringing the population to an estimated 644,000 by 2024 (DDM, 2021).

The demographic profile reveals a slightly higher proportion of females (50.31%) compared to males (49.69%), with a median age of 27 years, slightly above the provincial (26 years) and national (25 years) averages. The young working-age group (25-44 years) constitutes the largest share of the population at 28.07%, the older working-age group (45-64 years) makes up 21.20%, while those aged 65 and older comprise only 7.65% (DDM, 2020). In terms of racial composition, the population in 2019 was predominantly African (89%), with smaller proportions of White (8%) and Coloured (2%) individuals, reflecting the district's evolving socio-economic landscape. According to the spatial planning for Lejweleputswa, the district has 3 190 855 hectares of land area, constituting approximately 26.4% of the total provincial land area of about 12 969 028 hectares (DDM, 2021).

Figure 3.13: Lejweleputswa District Municipality-Map (Source: DDM, 2020).

3.4.1. Sample Methods

To ensure a representative sample of participants for both the interviews and surveys, a combination of purposive and random sampling methods was employed. Purposive sampling was used initially to select key participants with specific knowledge and experience related to the study's focus on indigenous knowledge and its applications in the district's primary industries, particularly mining and agriculture. These participants were chosen based on their expertise, involvement in local community practices, and their role in either the agricultural or mining sectors. For the survey, stratified random sampling was used to ensure that the sample accurately reflects the district's demographic composition, including gender, age, and racial diversity. Participants were selected from different towns within the district to capture a broad range of perspectives. The strata were divided based on the main demographic categories: gender, age group, and race (African, White, and Coloured).

The sample size was determined by considering the population size of Lejweleputswa and the need for statistical reliability. A sufficient number of respondents were included in the survey to ensure robust data representation and minimize potential errors. This approach aligns with the study's objectives and the demographic diversity of the district. For interviews, a manageable number of participants were selected to gather in-depth qualitative insights. This sample size was carefully chosen to balance the richness of detailed perspectives with the practical scope of the study.

3.4.2. Ethical Considerations

Participants were recruited through a combination of community outreach and local networks, including collaboration with community leaders, industry representatives, and local organizations. Recruitment was done through informed consent, where participants were fully briefed on the study's purpose, methodology, and their rights as participants, including their right to confidentiality and the voluntary nature of participation. Ethical considerations included ensuring confidentiality of personal information and maintaining anonymity in the reporting of survey results and interview findings. All participants were informed about the study's ethical guidelines, including their right to withdraw at any stage without consequence. Ethical approval for the study was obtained from Central University of Technology, Free State, from the ethical research committee and all data collection procedures adhered to established research ethics protocols, including compliance with data protection laws and respect for indigenous knowledge systems.

3.5. Semantic MHEWS Framework

The Semantic MHEWS framework is introduced as a semantic framework for representing data, demonstrating how a MHEWS can effectively monitor environmental pollution by showcasing its ability to integrate various data sources, predict pollution events, and provide timely alerts for proactive measures. This framework serves to illustrate the interconnected nature of EWS, enhancing comprehension of their holistic functioning in monitoring and predicting events. The primary aim of introducing the semantic middleware framework is to facilitate the integration of sensor data and indigenous knowledge, both structured and unstructured, thus enhancing semantic interoperability among EWS and their components. The middleware layer offers APIs for communication and abstraction of intricate modules, presenting data in a machine-readable format to foster integration and interoperability. As illustrated in Figures 3.5, the framework comprises four layers, spanning from data collection to the publication of annotated data via software programmes, and detailed in subsequent sub-sections.

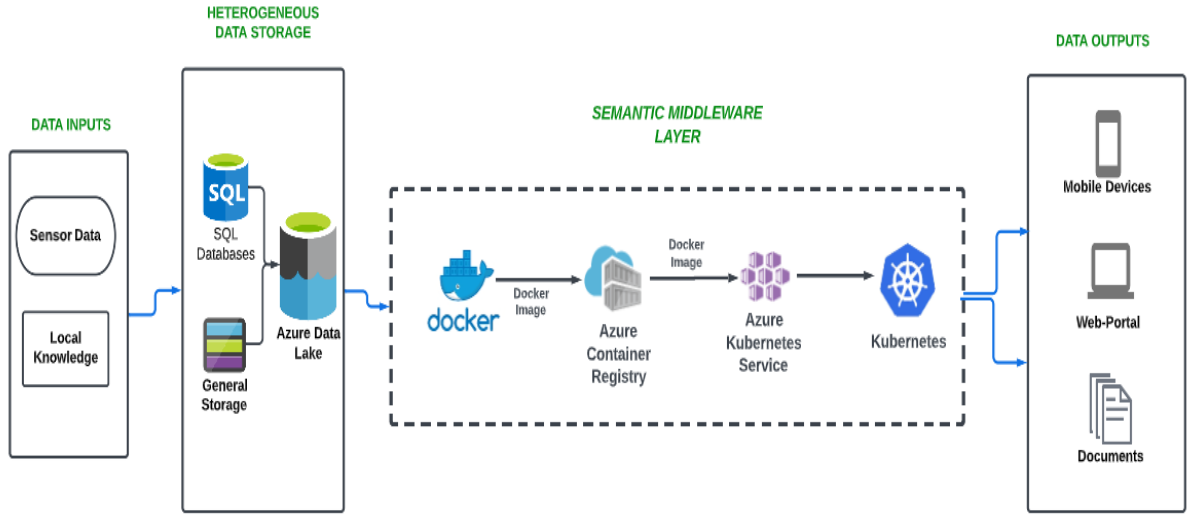


Figure 3.14: Semantic MHEWS Framework (Source: Author)

3.5.1. Heterogenous Data Storage

3.5.1.1 General Storage

General storage refers to unstructured or semi-structured data repositories typically used to store files, images, logs, documents, or raw data in formats such as CSV, JSON, XML, or flat files. This type of storage is not optimized for relational querying but is crucial for storing raw

or pre-processed data that is later transformed into usable insights. Unstructured and semi-structured data is ingested into Azure Data Lake (ADL), often without the need for extensive transformations.

3.5.1.2 SQL Database

SQL (Structured Query Language) databases are relational data storage systems designed for structured, transactional data. They enable efficient querying, indexing, and manipulation of data through well-defined schemas. SQL databases are particularly useful in maintaining normalized data structures, managing relationships between data entities, and executing complex queries. Examples include Microsoft SQL Server, MySQL, and PostgreSQL. Data from SQL databases undergoes an ETL (Extract, Transform, Load) process before being moved to Azure Data Lake (ADL). During extraction, structured data is retrieved, transformed into a suitable format and loaded into ADL for further analytics and storage.

3.5.1.3 Azure Data Lake

Azure Data Lake (ADL) is a cloud-based platform within Microsoft Azure, designed specifically to facilitate the analysis of large-scale data. It seamlessly collects data from diverse sources, accommodating various data types and formats. Offering boundless storage capacity, it caters to both sensor data and local knowledge. Through a gateway, structured and unstructured data are transmitted to ADL on the Microsoft Azure Cloud (MAC) for mining and presentation purposes. ADL is engineered to store a multitude of data formats on the Microsoft Azure Cloud platform, as depicted in the figures below. To streamline management and enhance accessibility across different operating systems like Windows, macOS, and Linux, the Microsoft Azure Storage Explorer application has been introduced for efficient interaction with ADL storage, aiding in the consolidation of data.

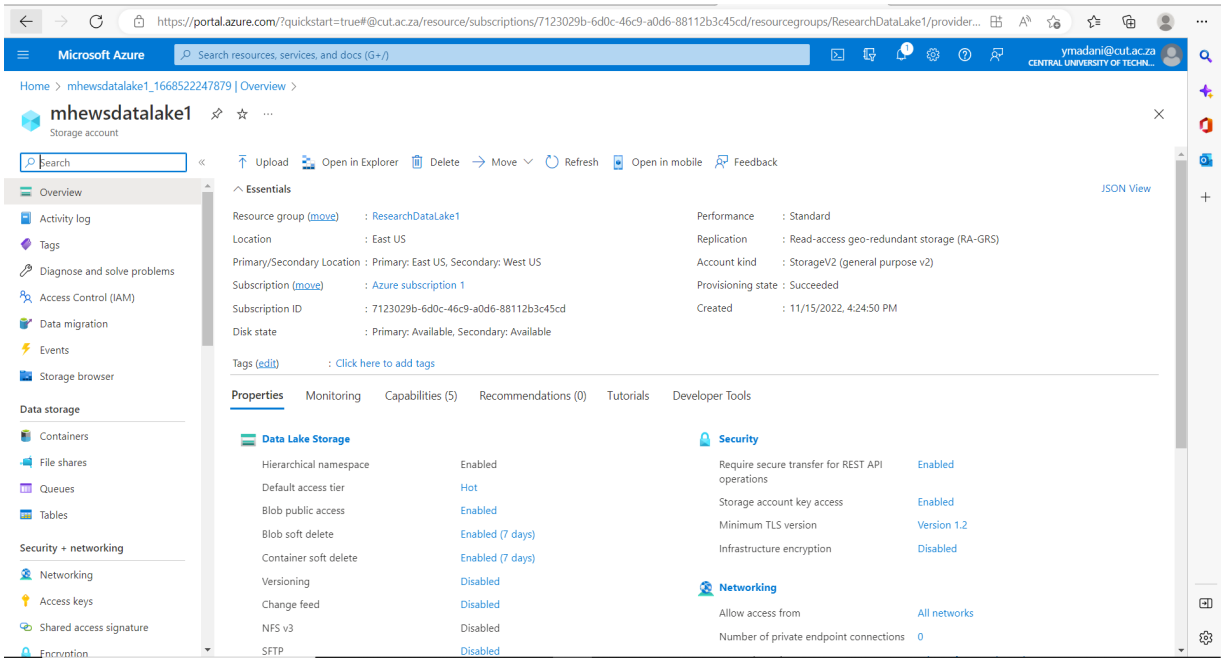


Figure 3.15: Overview of MHEWS Microsoft Azure Data Lake (Source: Author)

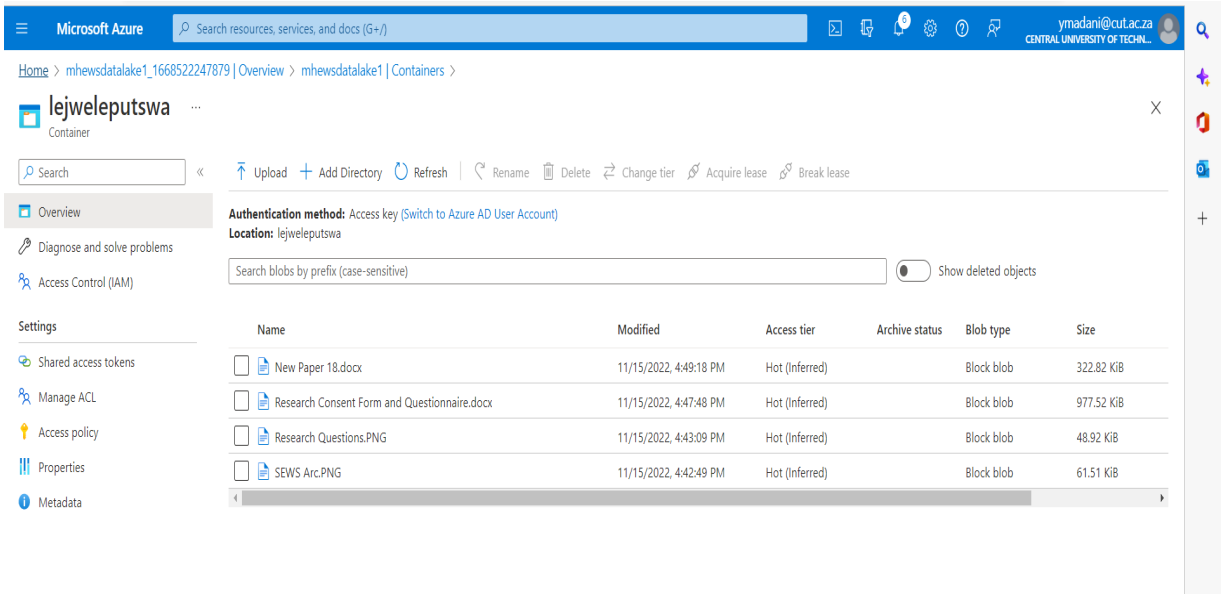


Figure 3.16: Overview of Lejweleputswa Container on the Azure Data Lake (Source: Author)

3.5.2. Semantic MHEWS Middleware Layer

Semantic middleware is a software layer that facilitates the communication and integration between different applications by integrating semantic technologies. The focus is on understanding the meaning and semantics of data. This middleware leverages technologies such as OWL and RDF to represent and encode the meaning of data in a machine-readable

format, using domain ontologies to provide a common understanding of concepts and relationships with a given domain such as environmental domain. The main function of semantic middleware is to enhance data interoperability. This is achieved by defining standardized formats, message types, and interpretation guidelines based on semantic understanding encoded in ontologies. This approach allows applications to communicate the meaning of the data and share understanding.

3.5.2.1. Docker

Docker's core revolves fundamental images, serving as the building blocks for containers. These images can be tailored with applications, essentially forming a blueprint for container construction. Using Docker files containing a set of instructions, commands in Docker containers can be carried out manually or automatically. These containers can be initiated, stopped, and terminated, and interconnected to construct multi-tiered applications. Interaction between Docker daemon and containers occurs via the Command Line Interface (CLI) or REST APIs. The steps below outline the process of application containerization with Docker.

1. Develop a container image using a Docker File.
2. Embed the application within the Docker Image.
3. Upload the Docker image to Docker hub.
4. Instantiate a Docker container based on the image.
5. Publish the container to Azure Container Registry using the Docker push command.

3.5.2.2. Azure Container Registry

Azure Container Registry (ACR) manages Docker registry service based on the open-source Docker Registry 2.0, and it enabled us to create and maintain azure container registries to store and manage Docker container images. Placing container registries close to your application servers speeds up the deployment time and reduces downtime associated with application releases. Docker images were pushed to ACR using Docker CLI and Docker push command.

Code snippet 1:

docker push [OPTIONS] NAME[:TAG]

3.5.2.3. Kubernetes (K’8s) Architecture

A Kubernetes cluster to function requires at least 2 vCPUs with 2GB memory (Dhas & Jeyanthi, 2019). A cluster of real or virtual machines, along with other infrastructure resources, is known as a Kubernetes cluster and is used to run containerized applications. It provides a

scalable and resilient environment for deploying, managing, and scaling containerized workloads efficiently. It works with a range of container tools including Docker, and has a large rapidly growing ecosystem.

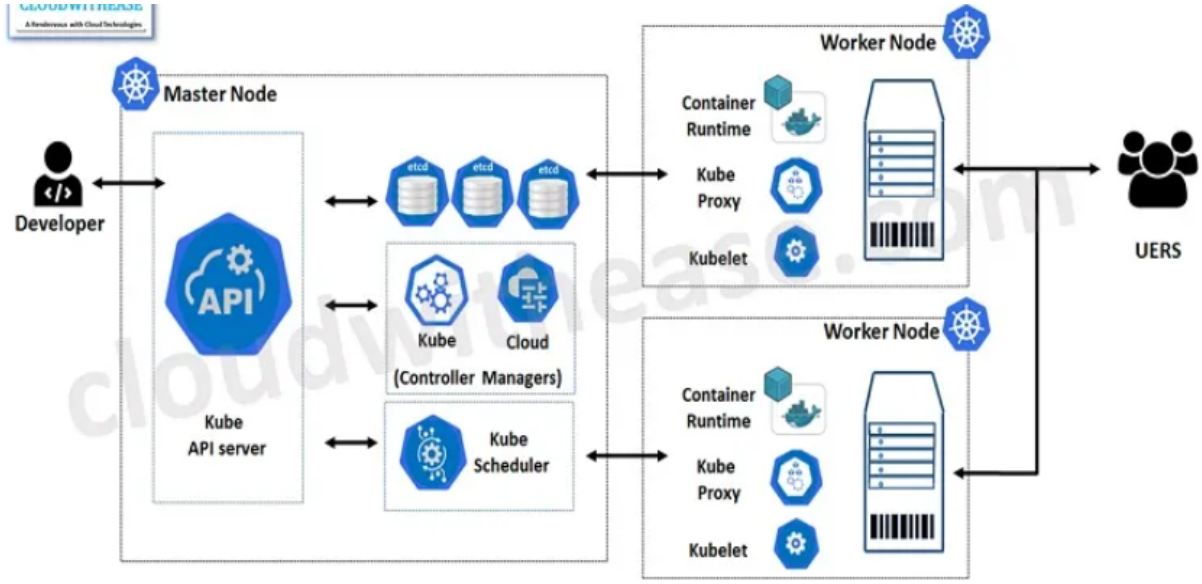


Figure 3.17: Kubernetes Cluster (Source: Author)

Key Components of a Kubernetes Cluster are shown below.

Master Node

- The master node is the control plane of the Kubernetes cluster.
- Components such as *etcd*, API server, schedular and controller manager run on the master node.

API Server

- The API server functions as the interface to the Kubernetes control panel, providing access to the core Kubernetes APIs for management tools like kubectl or the Kubernetes dashboard.

Worker Nodes

- The operative units, or minions, situated within the cluster are tasked with executing application workloads..
- Worker nodes run container runtimes like Docker to execute containers.

Scheduler

- The scheduler identifies which worker nodes are capable of handling the workload and executes it on those nodes.
- Schedular ensure that containers run on nodes that meet the application requirements.

Etcd

- Etcd serves as a distributed repository for storing the configuration data of the cluster. It functions as a reliable key-value store system integrated into Kubernetes, ensuring high availability.

Controller Manager

- The controller manager facilitates the integration of Kubernetes with various cloud providers. Kubernetes offers plugins tailored for cloud providers, allowing for seamless integration and platform customization.
- In the controller manager various controllers such as the Replication Controller and the Statefulset Controller, manage different types of workloads.

Kubelet

- The Kubelet is an agent that operates on each node. It retrieves the pod configuration from the API server and ensures the specified containers are running and operational.

Container Runtime

- The Container Runtime is a component of the Kubernetes system that handles the task of executing and managing containers within a Kubernetes cluster.
- A container runtime enables applications that are packaged into containers to operate smoothly. It also ensures that these applications can communicate and interact effectively with necessary resources, including virtual networks and storage systems. In essence, the container runtime acts as a mediator, allowing the containerized applications to access and use the various resources they need to function properly.

Overall, Kubernetes is a key solution in the evolving landscape of containerization and cloud computing. Its architecture addresses the challenges of deploying, managing, and scaling container workloads in a scalable and resilient manner. Its importance is justified by its ability

to increase application reliability, deployment processes and to ensure portability in cloud environments. In this study, Kubernetes was used as a platform to monitor docker containers.

3.6. Prototype

The primary goal of this prototype is to develop a MHEWS that integrates air pollution and weather data, to provide accurate real-time alerts to users and community members. The reason for creating this prototype is to reduce the effects of air pollution and bad weather. The system collects real-time air pollution data from monitoring stations and combines this data to analyse the relationship between air pollution and weather conditions. Data analysis algorithms are used to identify patterns that lead to air pollution and weather conditions. When dangerous conditions are detected, the system generates real-time alerts and alerts the community members. The system has a user interface that allows users to log in and receive notifications through communication methods such as Short Message Service (SMS), web portal, and mobile app push notifications. In addition, the system also provides a risk assessment report that assesses potential health environmental impacts from air pollution. This information helps users to make some informed decisions, and take appropriate actions. The system is developed using python Integrated Development Environment (IDE) to create an application that can be accessible by multiple platforms.

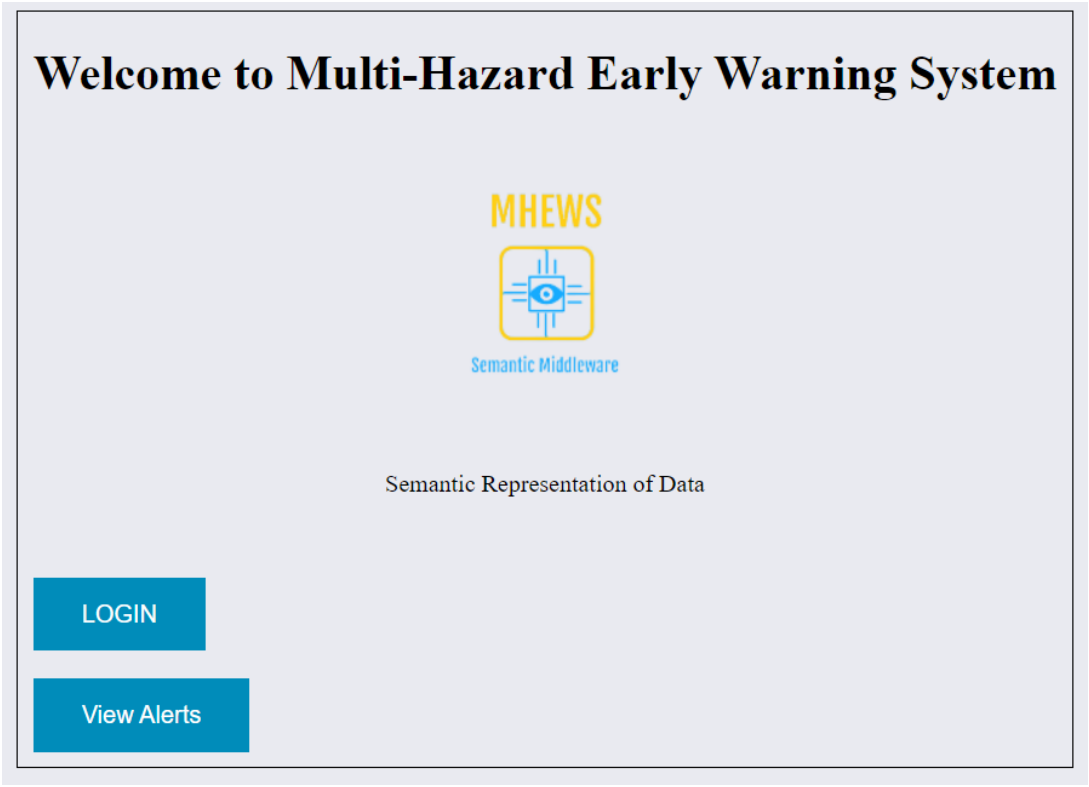


Figure 3.18: MHEWS Application (Home-Screen) (Source: Author)

Multi-Hazard Early Warning System

Weather Data

Date	Temperature	Humidity	Wind Speed
2023-06-22	25	70	10
2023-06-23	26	68	12
2023-06-24	27	72	15

Air Polution Data

Date	Gas Level	Location
2023-06-22	61.3	Carbon Monoxide
2023-06-23	77.9	Carbon Dioxide
2023-06-24	20.8	Hydrogen Sulfide
2023-06-24	39.5	Methane Gas

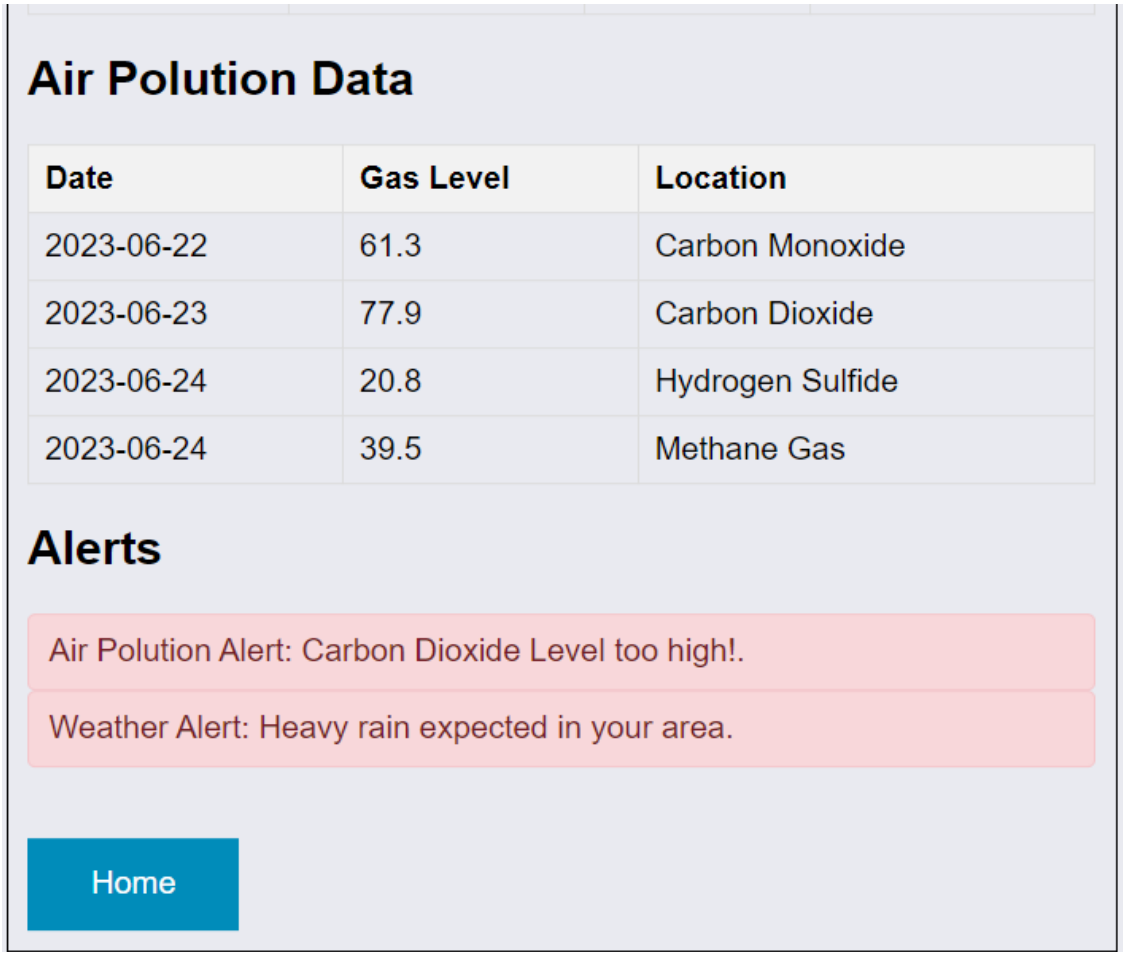


Figure 3.19: MHEWS Application Overview (Source: Author)

The system maintains a historical database of air pollution and weather data, allowing users to analyse trends, patterns, and relationships over time. This capability facilitates research and enables the identification of long-term impacts and mitigation strategies.

3.7. Summary

This chapter outlined the research methodology adopted for developing the Semantic MHEWS in the Lejweleputswa district, combining both quantitative and qualitative approaches. A mixed methods design was employed, integrating indigenous knowledge gathered through interviews and questionnaires with quantitative data collected via a wireless sensor network. The diverse data types were efficiently stored and managed in Azure Data Lake (ADL), which supports seamless data integration. To enhance data processing and application deployment, semantic middleware was utilized, with Docker and Kubernetes providing robust containerisation and orchestration capabilities. The methodological framework not only facilitated the collection

and management of complex datasets but also laid the foundation for the development of a pollution warning system tailored to the unique needs of the region. Integrating advanced technologies with local knowledge, this chapter demonstrates how the research design directly addresses the problem statement and contributes to the overall goal of creating an effective and sustainable MHEWS prototype. This methodological groundwork is pivotal for interpreting the findings and drawing conclusions in subsequent chapter.

CHAPTER FOUR

SYSTEM DESIGN, IMPLEMENTATION AND ANALYSIS

4.1 Introduction

This chapter introduces the design and implementation of semantic middleware using a modern approach, application containerization. The use of application containerization in the form of technologies such as Docker and Kubernetes brings new levels of efficiency, scalability and portability to the design and implementation of semantic middleware. Containerisation technologies such as Docker and Kubernetes offer standardised and efficient approaches for packaging, deploying, and managing applications in distributed environments. The need to bridge the gap between different heterogenous systems, and to ensure meaningful communication between the systems is where the semantic representation mechanism plays an important role. This mechanism is essential to a shared language and meaning, enabling effective communication and collaboration. Using Docker for containerisation, we can encapsulate these semantically enriched heterogeneous systems in isolated environments, ensuring their seamless deployment and management. This ensures that sensor reading and user notifications are effectively exchanged between different components in the system.

Containerising software applications using Docker and Kubernetes offers advantages for MHEWS. As a container orchestration platform, Kubernetes enables autoscaling, load balancing, and fault tolerance, and ensures high availability and reliability of MHEWS even if the infrastructure fails. The combination of semantic middleware and containerization technology gives MHEWS great benefits. This enables easy integration and interoperability of various software components and increases system scalability and reliability. In the following sub-sections, we detail architectural design and implementation considerations for MHEWS, with a focus on using semantic middleware and containerisation technologies, we will explore how the technologies can improve data interoperability and improve system performance.

4.2 System Analysis Design

The analysis of the system design begins with identifying the functional and non-functional requirements of MHEWS. Following a critical evaluation of the previously adopted EWS frameworks in the Lejweleputswa district, this section addresses their limitations as discussed in Chapter One. Subsequently, a new design is proposed, focusing on resolving these issues

while aligning with the functional and non-functional requirements identified. This section is categorized into two broad areas: Functional Requirements (FR), and Non-Functional Requirements (NFR).

4.2.1. Functional Requirements (FR)

FR1: Detection and Monitoring

The system should be able to detect and monitor air pollution and weather data. It should utilise sensors, data collection mechanisms, and algorithms to identify potential threats accurately.

FR2: Early Warnings

The system is responsible for generating and disseminating accurate early warnings. These alerts should provide detailed information on air pollution levels, and recommended actions, enabling communities to respond appropriately to potential threats.

FR3: Data Acquisition and Processing

The system must collect and process both structured and unstructured data from the specified study area in real-time. It should process and analyse data in real-time to send out reliable information.

FR4: Authentication

The system must implement a secure authentication mechanism to allow authorized users access to critical data, including real-time pollution levels and affected areas within the defined study area.

FR5: Alert Dissemination

The system must include a publishing system for making pollution warnings known to people through different platforms such as mobile applications and web portal. This means that such system should send out alerts whenever the level of pollution is too high for communities.

FR6: Documentation and Support

Adequate documentation and support resources should be provided to system administrators and end-users to ensure smooth operation and troubleshooting of the system.

4.2.2. Non-Functional Requirements (NFR)

NFR1: Reliability and Availability

The system must guarantee high reliability and availability. Given its critical role in early warning, it must be accessible online at all times, with minimal downtime.

NFR2: Usability and User Experience

The system should prioritise user-friendliness, ensuring that users can easily navigate and operate the platform. Information should be clearly presented to minimize confusion and facilitate quick decision-making.

NFR3: Interoperability

The system should be able to integrate with existing early warning systems, and receive data to enable smooth data exchange.

4.2.3. Hardware and Software Components

The hardware and software components required for the development and deployment of MHEWS are outlined in Table 4.1. These include various software tools and hardware infrastructure needed for system functionality and integration.

Software Requirements	Hardware Requirements
MHEWS Application Code	**Computer**
Docker Desktop v4.20.0	**Network (WIFI)**
Azure Kubernetes Services	**Wireless Sensor Network**
Operating System (Windows10-11)	
Python anywhere	
Visual Studio Code 2017	
Azure Data Lake server	
Azure Container Registry	
Apache Jena	

Table 4.1: Hardware and Software Requirements (Source: Author)

4.2.4. System Use Case Modelling

Use case modelling is used to describe the interactions between different actors, and the system being developed. It helps to identify the functionalities and behaviour of the system. The following subsections elaborate on the key use cases that define the MHEWS architecture.

4.2.4.1. MHEWS Use Case

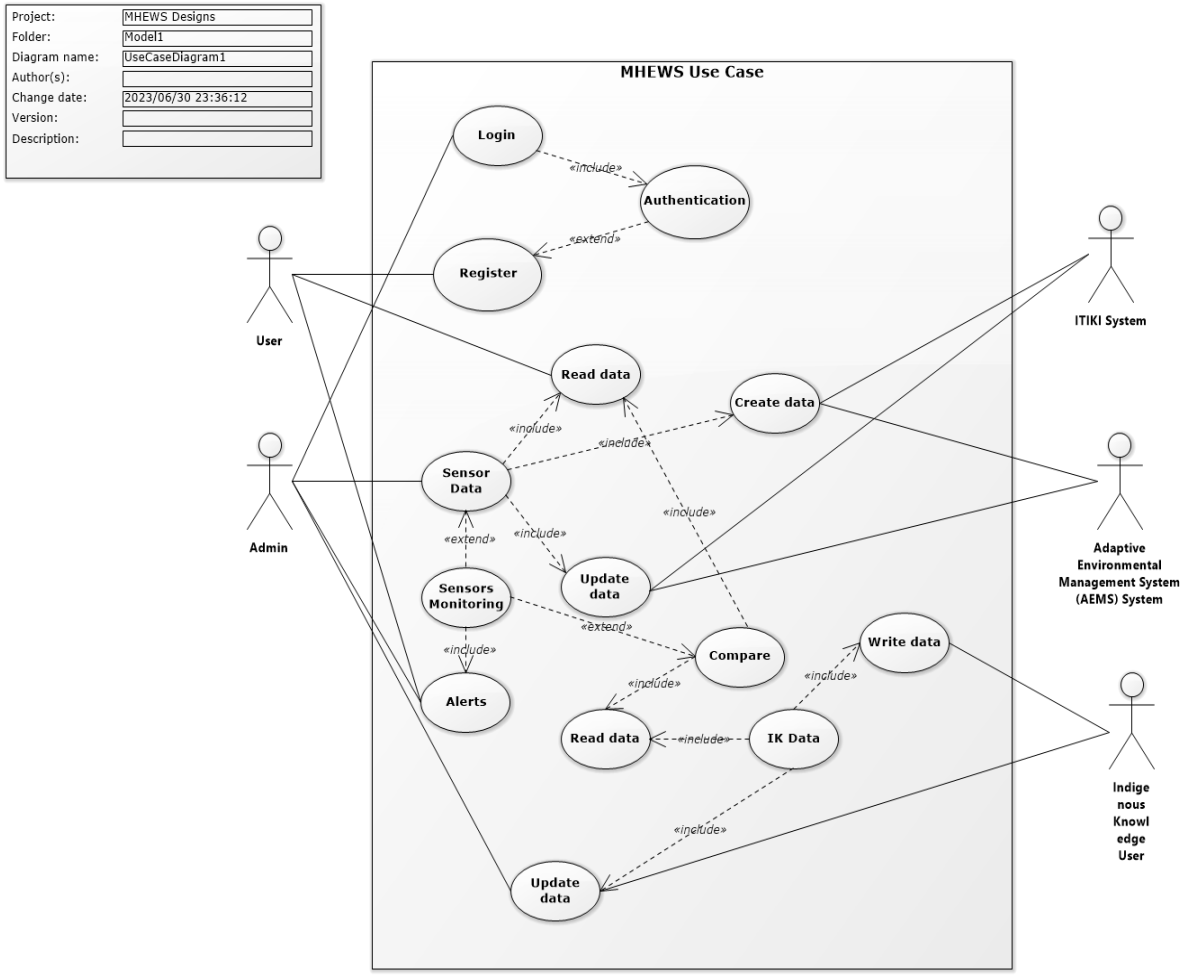


Figure 4.20: MHEWS Use Case (Source: Author)

- **Authentication**

  Register: Users provide credentials to authenticate and access the MHEWS system.

  Logout: The user logs out of the system, and ends the session.

- **Sensor Data:**

  ITIKI & AEMS: Sends critical data to the MHEWS system.

: Wireless sensors send data to the MHEWS system, including information such as location of the polluted area and measurements.

Receive Sensor Data: The system receives and processes sensor data for analysis, and measurements.

- **Indigenous Knowledge**

  Indigenous Knowledge helps to obtain how local people observe, monitor, and prevent environmental pollution around them using techniques such as cultural beliefs and traditional warning signs.

- **Sensor Monitoring**

  Sensor Status: Administrator monitors sensor status to ensure that sensors are functioning properly.
  Sensor Alerts: The monitoring system alerts the administrator if a sensor malfunction or no longer provides data.

- **Alerts**

  Send Notifications: The MHEWS system generates and sends alerts to users based on the readings of air pollution, weather data and sensor data.
  Receive Notifications: Users receive notifications through various communication channels such as web portals, SMS, mobile applications and stay informed about potential threats.

4.2.5 Data Modelling

In designing the data architecture for MHEWS, it is essential to model the data in a way that reflects real-world relationships and interactions between different components. The following sections detail two primary data modelling approaches used. These approaches are: Entity Relationship Diagram (ERD), and Class Diagrams (UML). These diagrams visualise the relationships between entities in the system, and provide a framework for the system's data structure.

4.2.5.1. MHEWS Entity Relationship Diagram

The ERD offers a conceptual blueprint of the database, and the relationships between different actors such as sensors, administrators, and users within the system. It demonstrates how data

flows between entities, and serves as a foundation for understanding the system's database structure.

Actors and Relationships:

- Sensor Data: Sensors collect environmental data, such as air quality measurements, and transmit this data to the MHEWS system for analysis.
- Users: Users receive notifications and access system data through authenticated interfaces.
- Indigenous Knowledge: Information gathered from local communities about traditional environmental monitoring practices is also stored in the system for correlation with sensor data.
- Alerts: The system sends notifications to users based on sensor data and indigenous knowledge analysis, maintaining a continuous feedback loop between the data and the system's users.

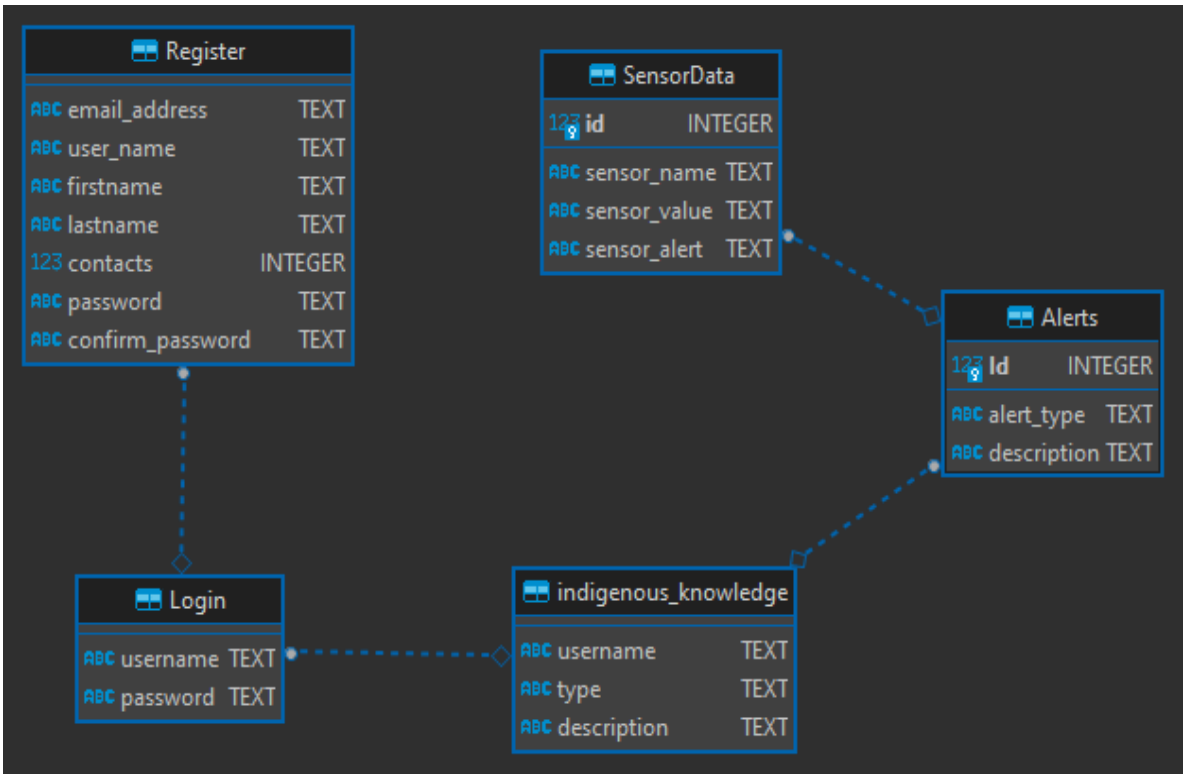


Figure 4.21: MHEWS Entity Relationship Diagram (Source: Author)

4.2.5.2. Class Diagram

The class diagram outlines the structure of MHEWS by modelling the system's classes, attributes, and methods. It offers a static view of the system, focusing on its object-oriented structure. The diagram is essential for defining the system's internal architecture and understanding how different components interact at the code level.

Core Classes:

- Sensor: Represents physical devices that capture environmental data.
- User: Manages user profiles, including authentication, notifications, and data access.
- Alert: Manages the dissemination of warnings and alerts based on real-time data.
- Indigenous Knowledge: Stores and processes traditional knowledge for integration with modern data sources.

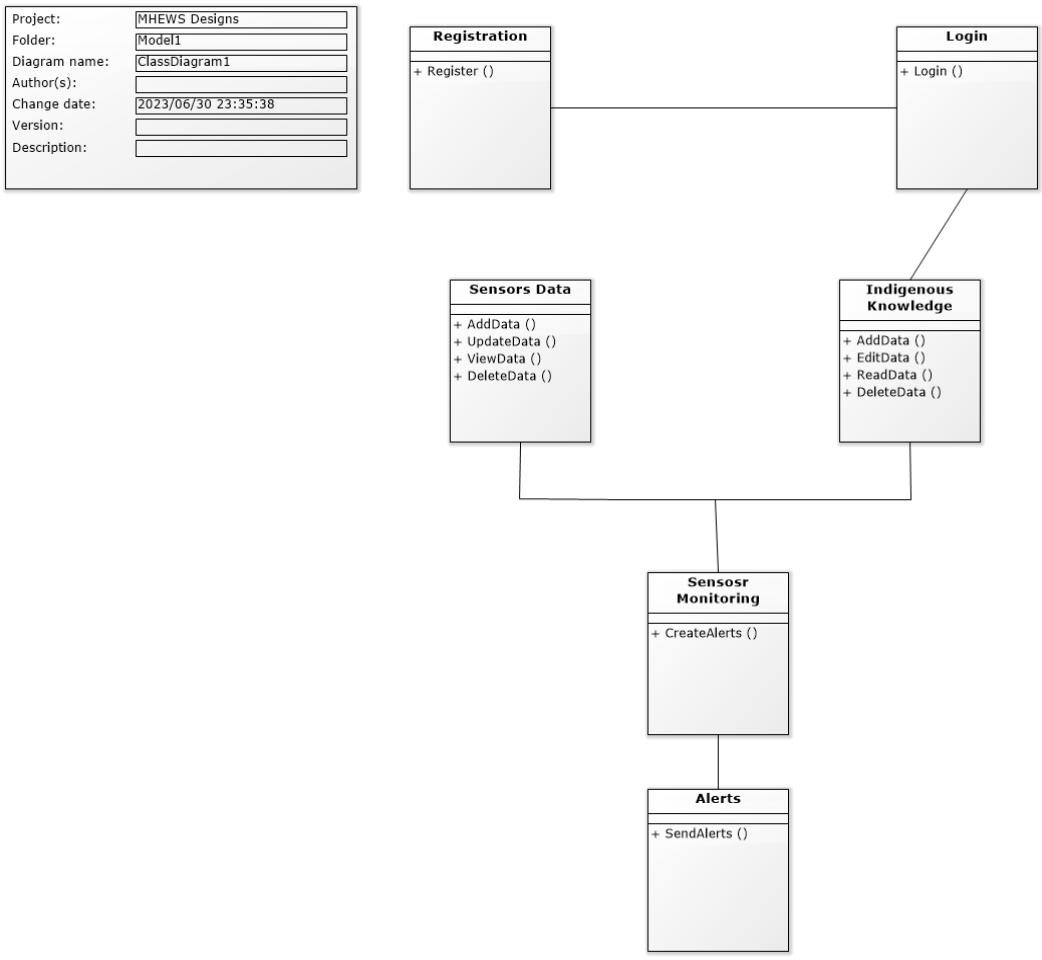


Figure 4.22: MHEWS Class Diagram (Source: Author)

4.2.6. System Integration/ Unified Modelling Language (UML)

The aim is to identify various software components and system that needs to be integrated into MHEWS. This includes definitions of protocol and data formats required for seamless integration. Integration testing and validation should be performed to ensure the smooth operation of the integrated system.

4.2.6.1. Sequence Diagram

The sequence diagram models the flow of data and operations within MHEWS. It tracks the interactions between system actors (e.g., users, and administrators); and components (e.g., sensors, and databases) in real-time, ensuring that all processes are synchronised and properly executed. Figures 4-4 and 4-5 offer detailed visual representations of the sequence diagram.

1. Login

1.1) Authentication

1.1.1. Request alerts

1.1.2. Send/ Receive alerts

1.2) User enters login details.

The system responds with an authentication message.

1.3) If the user does enter invalid login details, an error message is displayed check username and password.

2. Registration

2.1) If the user does not exist, redirect to a registration page, and enter registration details.

2.1.1) Check if the user is a normal user or an indigenous knowledge user.

2.2) Create an account, check if valid details are entered and respond with a success message

3. Sensor data

3.1) Sensors send data to the database.

3.2) Sensors update data.

3.3) Sensor monitors reads data from sensors.

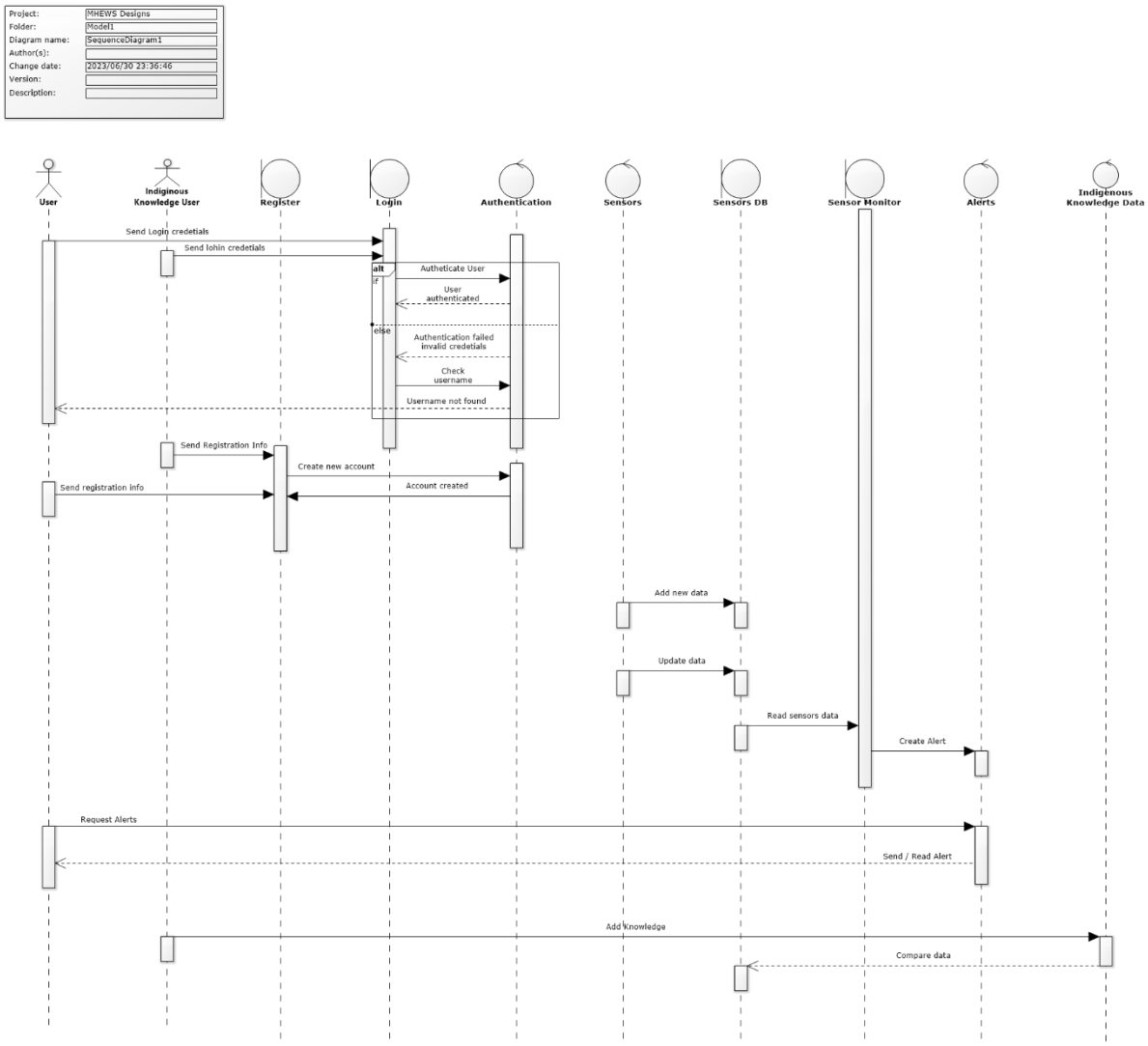


Figure 4.23: MHEWS Sequence Diagram (Source: Author)

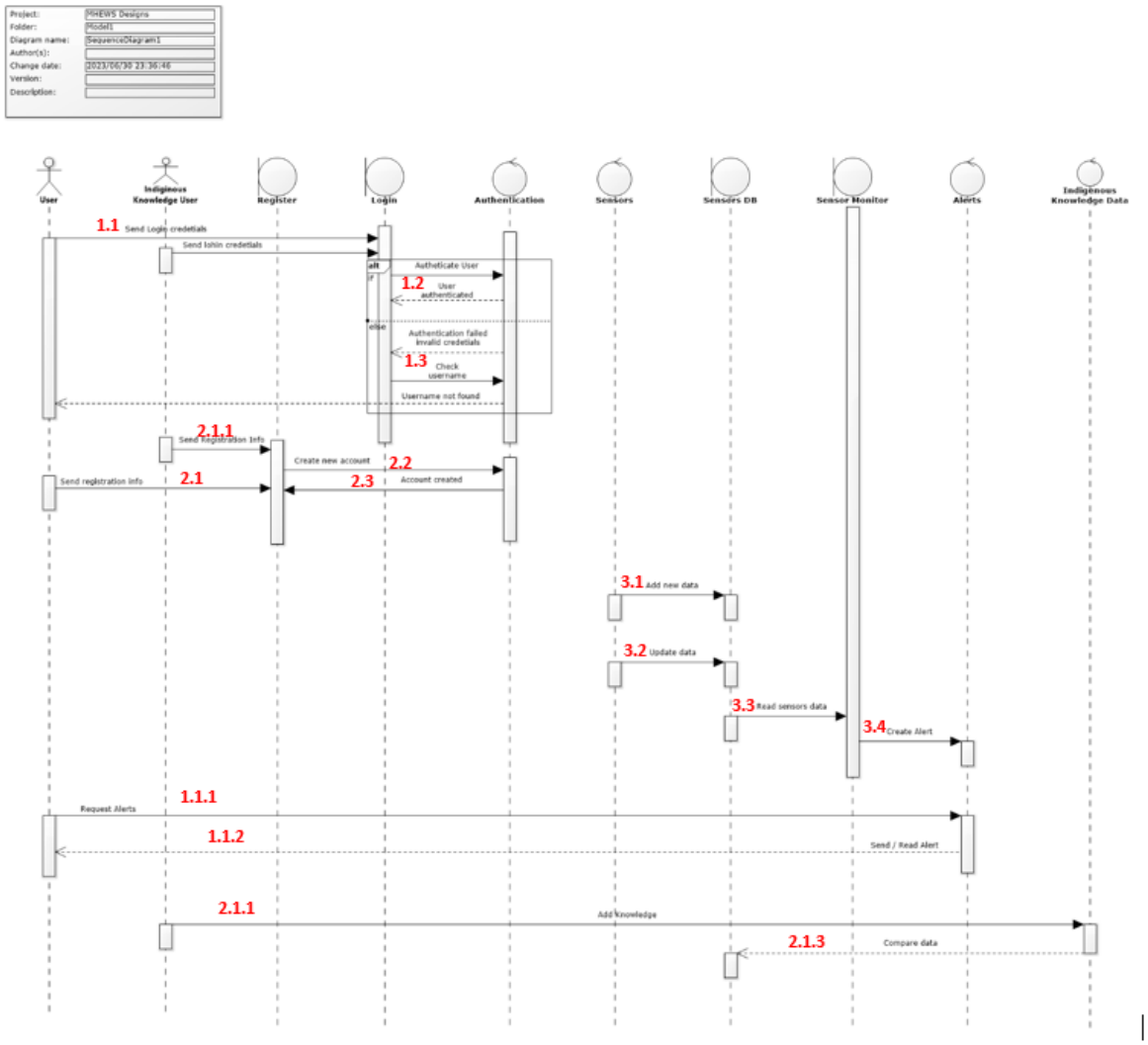


Figure 4.24: MHEWS Sequence Diagram Overview (Source: Author)

4.3 System Implementation

This section elaborates the transition from system design to implementation. The implementation phase focuses on developing a functional MHEWS system based on the design specifications. This includes containerizing the system using Docker, orchestrating it with Kubernetes, and incorporating semantic middleware for enhanced system interoperability.

4.3.1 Semantic Middleware

Semantic middleware is crucial for enabling seamless communication and data exchange between MHEWS components, especially in heterogeneous environments. By leveraging semantic technologies, the system ensures that data from disparate sources can be effectively

integrated, interpreted, and acted upon. The following steps outline the process of integrating semantic middleware using Docker, and Kubernetes.

4.3.1 Creating a Docker File and Docker Image

1. Creating the MHEWS Application

   The application is developed using Python and hosted on a service such as PythonAnywhere. The initial application is tested using Flask to ensure it runs locally before containerization. Figures 4-6 and 4-7 illustrate the application navigation and testing.

Figure 4.25: MHEWS Navigation Overview (Source: Author)

2. Test the App by running command: python -m flask run.

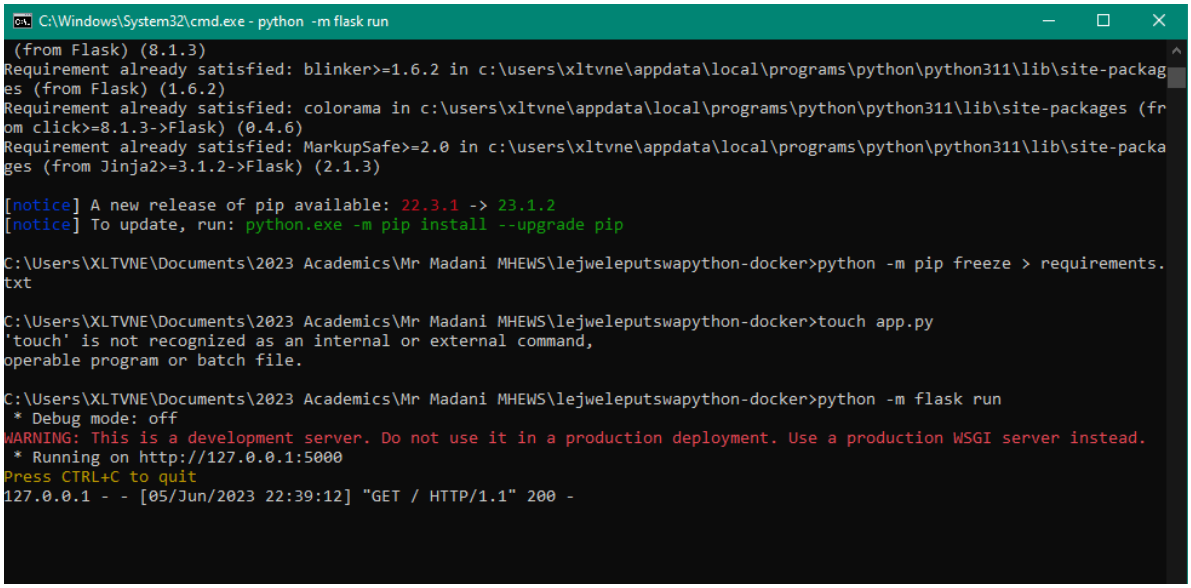


Figure 4.26: MHEWS Application Testing (Source: Author)

3. Application Results

A MHEWS python application: http://mhews.pythonanywhere.com/

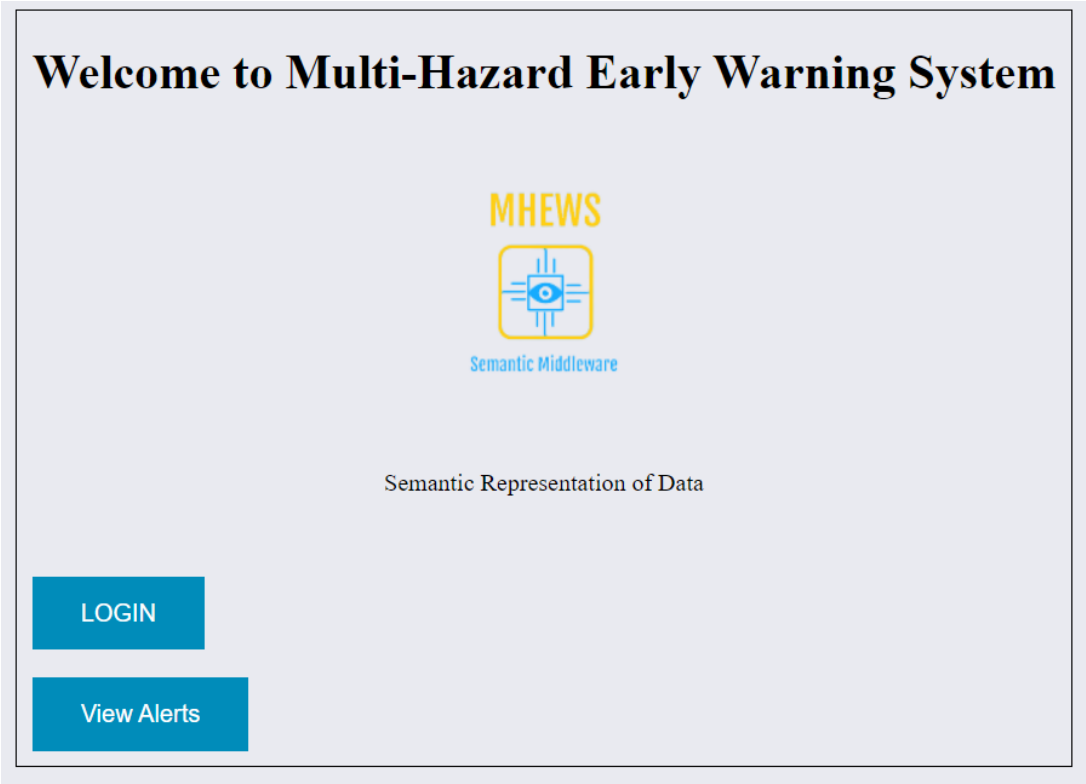


Figure 4.27: MHEWS-Welcoming Page (Source: Author)

4. Create a Docker File

   A Docker file is a file used to define the steps required to build a Docker image. This simplifies the process of building and running distributed applications inside containers. Defining one's application environment, and dependencies in a Docker file provides scalability, control, security, and infrastructure deployment.

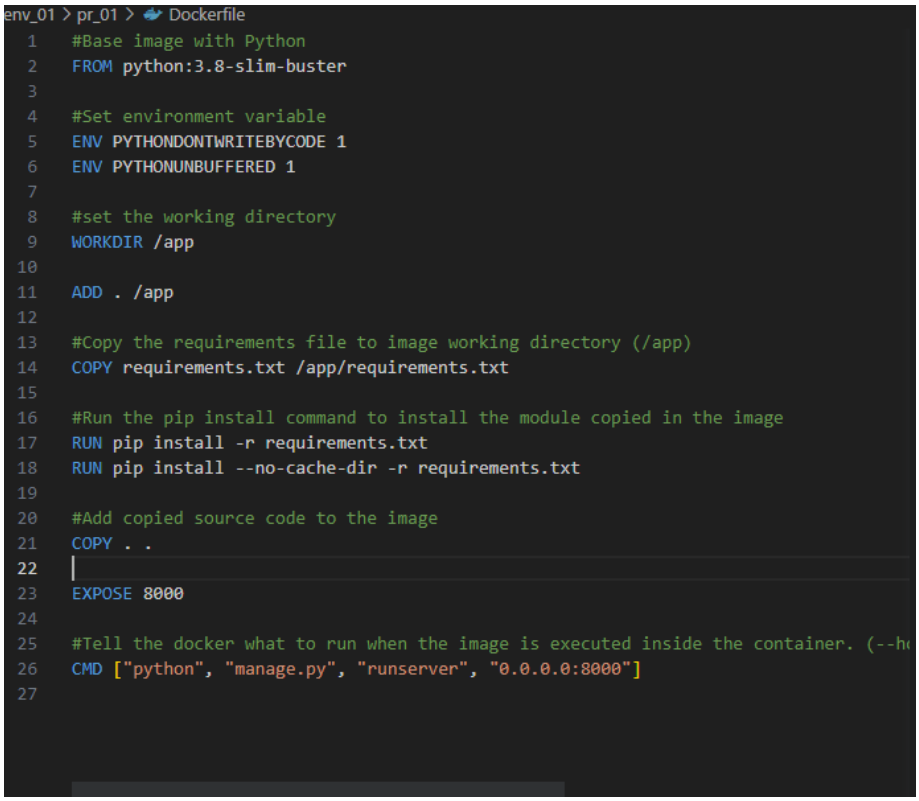


Figure 4.28: Docker File Overview (Source: Author)

5. Build a Docker Image\

   Docker images are a crucial component of modern software development, and deployment process. Docker images are built from Docker files, which define the instructions for building the Docker images. Docker images are highly portable, creating a Docker image makes it easy to distribute, and run the image in any computer with Docker installed, regardless of the underlying operating system. Figures 4.10, and 4.11 depict the building process and image deployment.

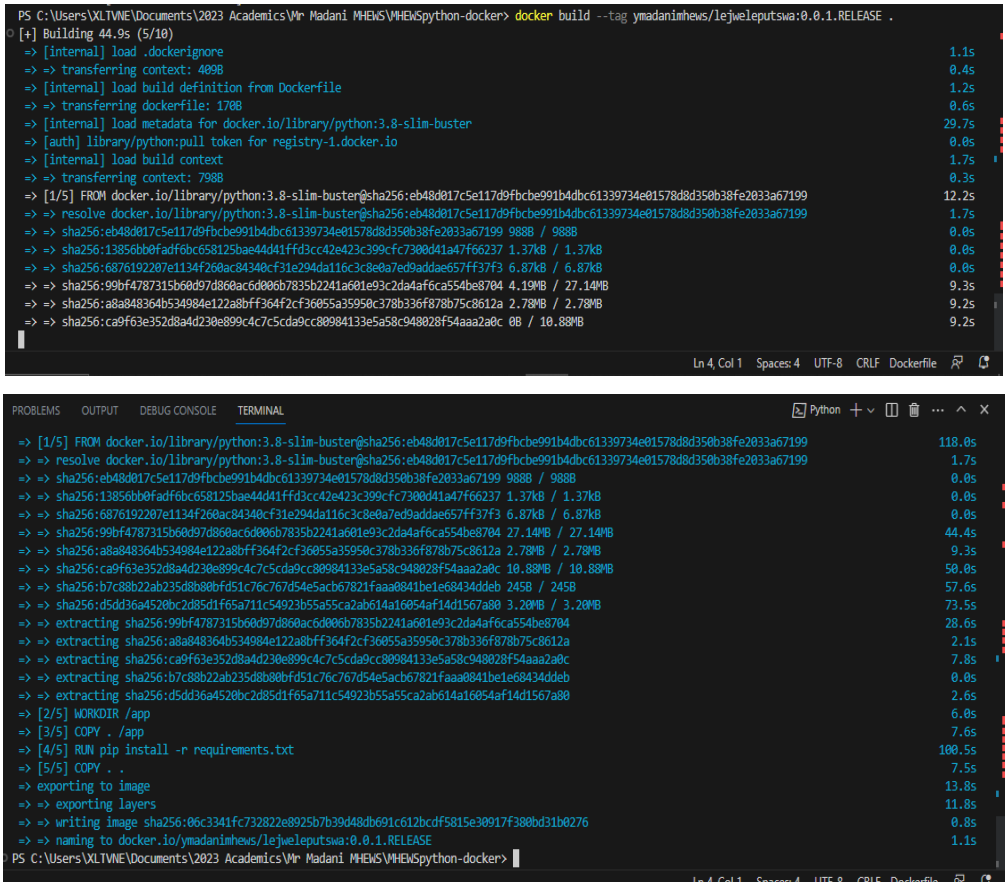


Figure 4.29: Build-Docker Image (Source: Author)

6. View the Image deployed to Docker.

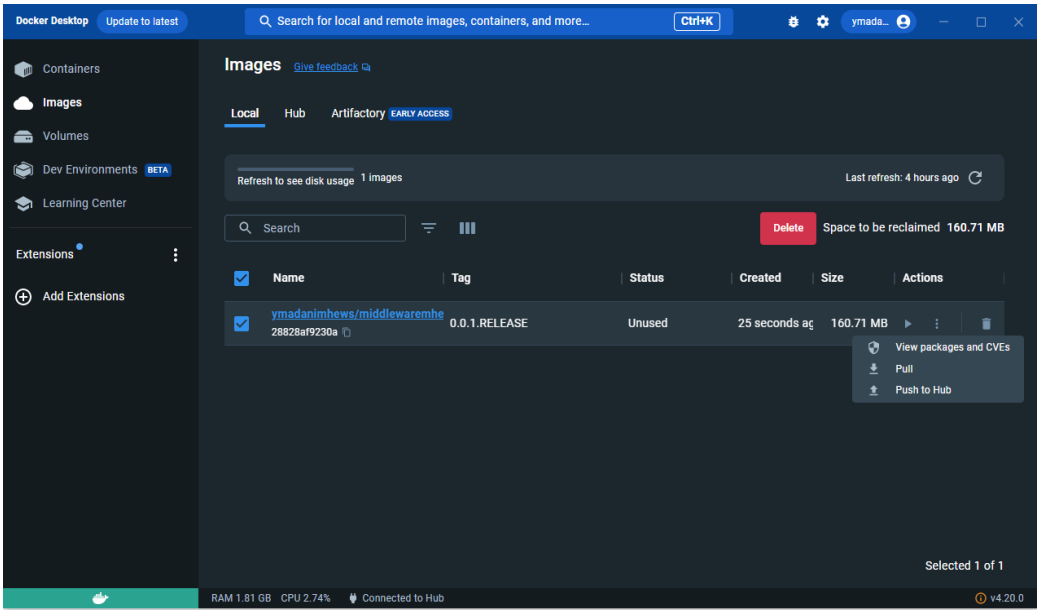


Figure 4.30: Docker Image Overview (Source: Author)

7. Create a container and run the container.

To deploy the application, Docker container is needed, Docker container is a running instance of a Docker image and can run on any system that has Docker installed. It makes the deployment process across multiple platforms easy and ensures consistency across different platforms or environments. Figures 4.12, 4.13, and 4.14 illustrate container creation and management.

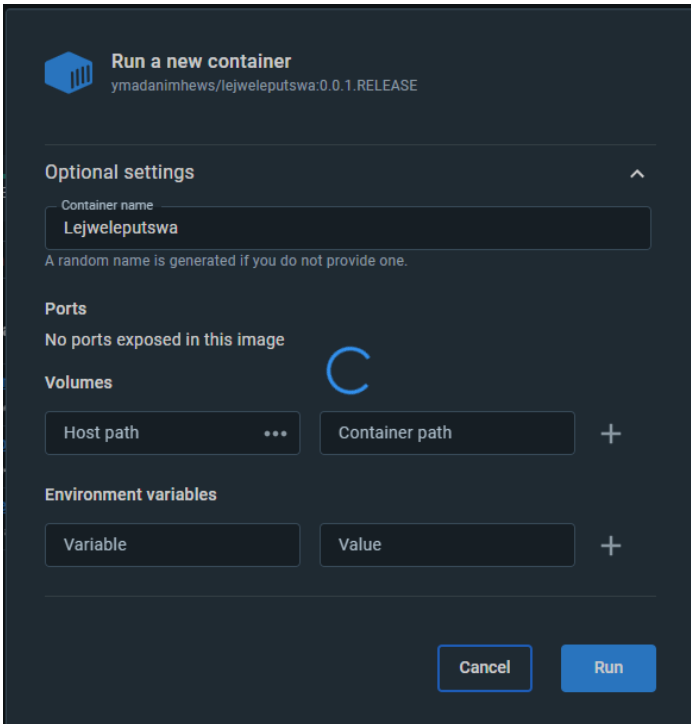


Figure 4.31 : Run-Docker Container (Source: Author)

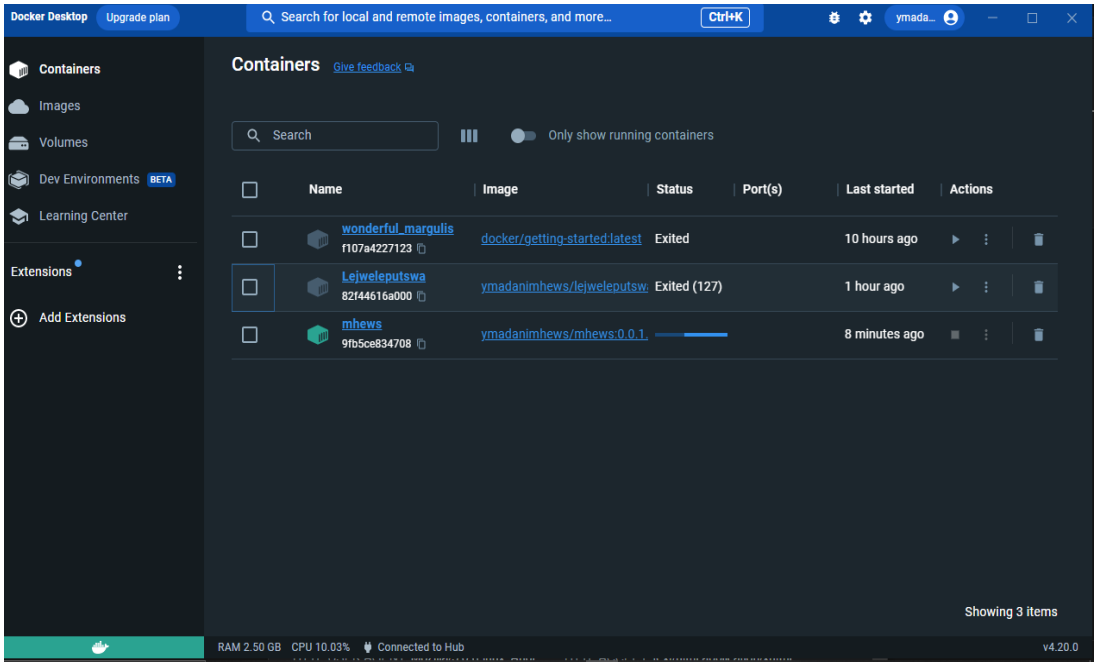


Figure 4.32: Docker Containers (Source: Author)

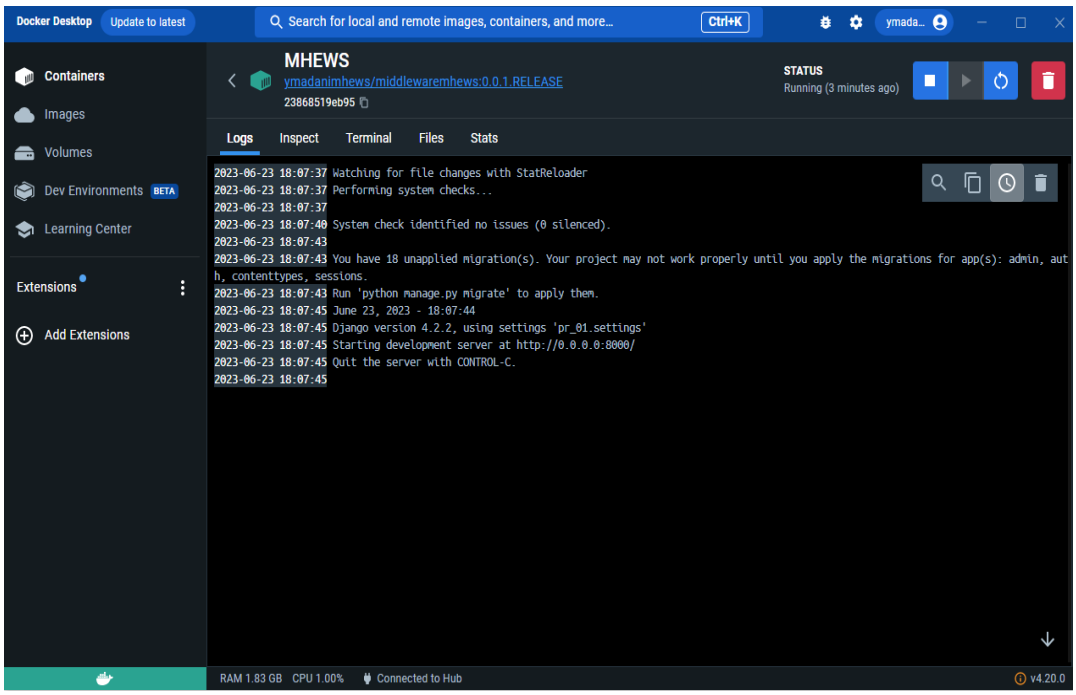


Figure 4.33: MHEWS-Docker Container Overview (Source: Author)

8. Push the Docker Image to Azure Container using the ‘docker push’ command.

   Once all the steps above are successfully implemented, push the Docker image that is inside the Docker container to the Azure Container Registry. Azure Container Registry provides secure and scalable image management, facilitates deployment, and offers seamless integration with Azure Services.

4.3.1.2 Azure Container Registry

To successfully push a Docker image to the Azure Container Registry (ACR), it is imperative to follow a precise and structured set of steps to ensure the seamless integration and deployment of containerized applications. This process requires careful attention to detail to guarantee the security, scalability, and efficiency of the deployment pipeline. The following steps outline the critical procedures involved. Figures 4.15, 4.16, 4.17, and 4.18 illustrate ACR creation.

1. Create Azure container Registry on Microsoft Azure cloud shell.

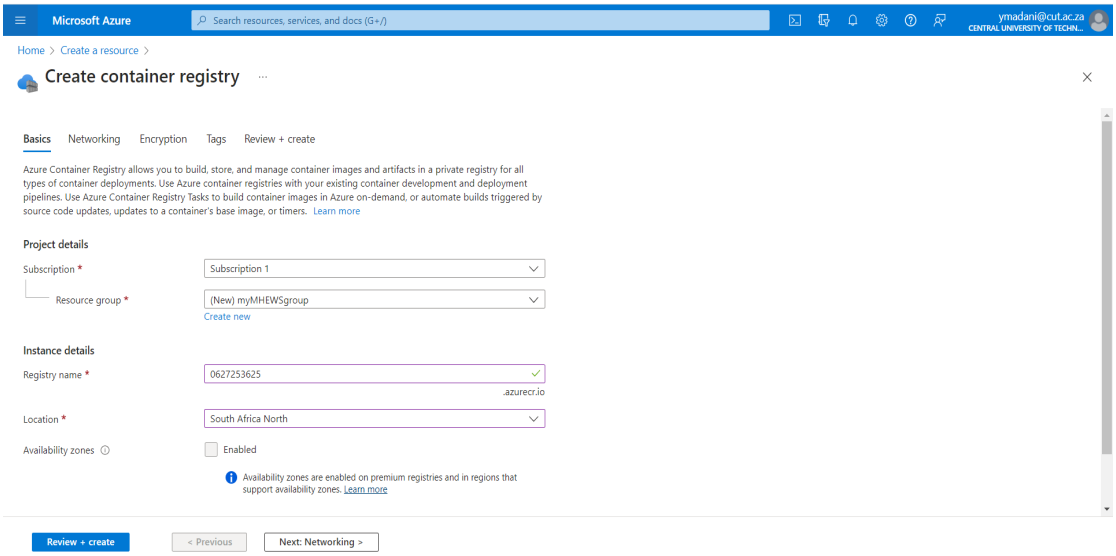


Figure 4.34: Creating MHEWS ACR in Azure (Source: Author)

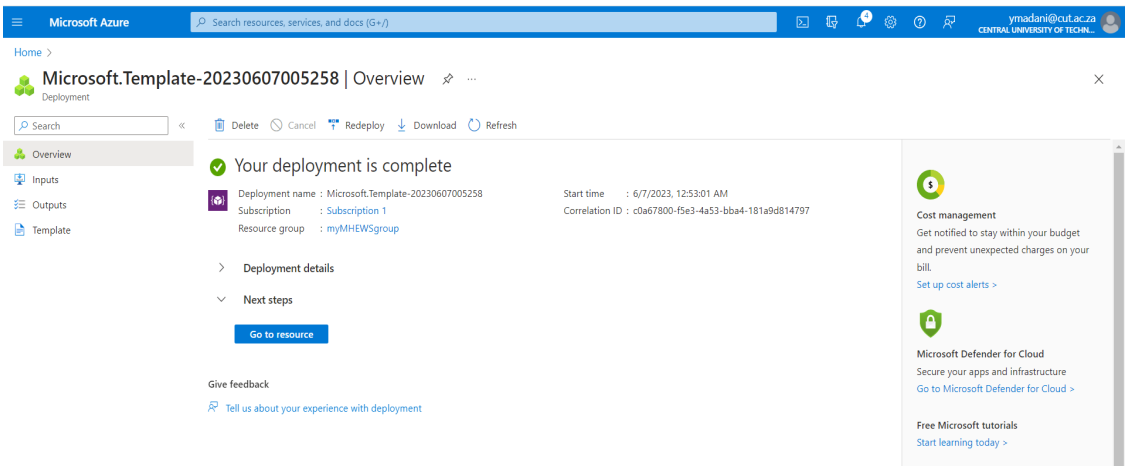


Figure 4.35: MHEWS ACR deployment in the Azure (Source Author)

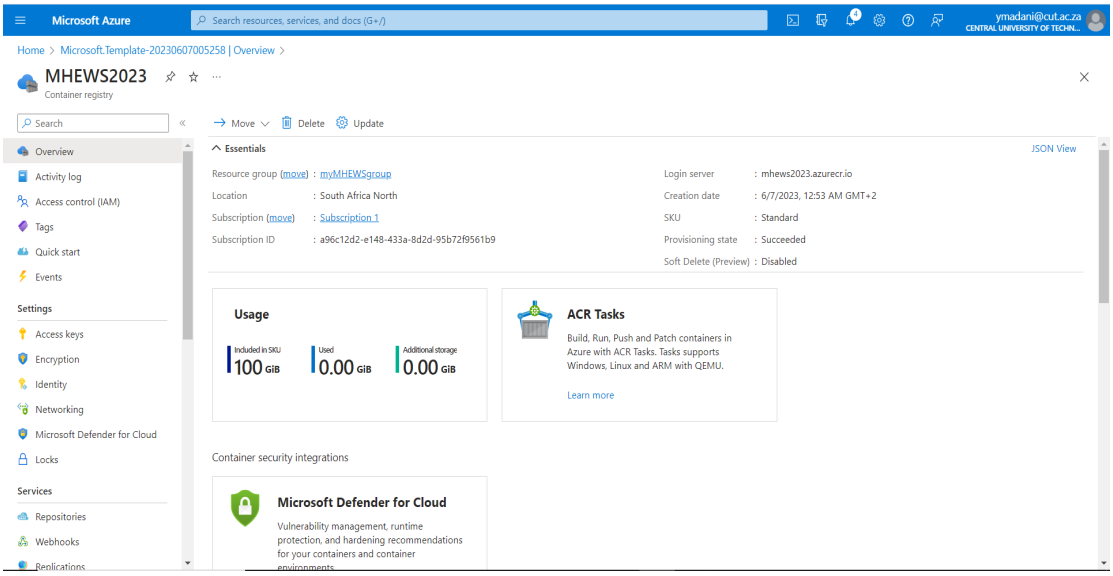


Figure 4.36: MHEWS ACR in the Azure overview (Source: Author)

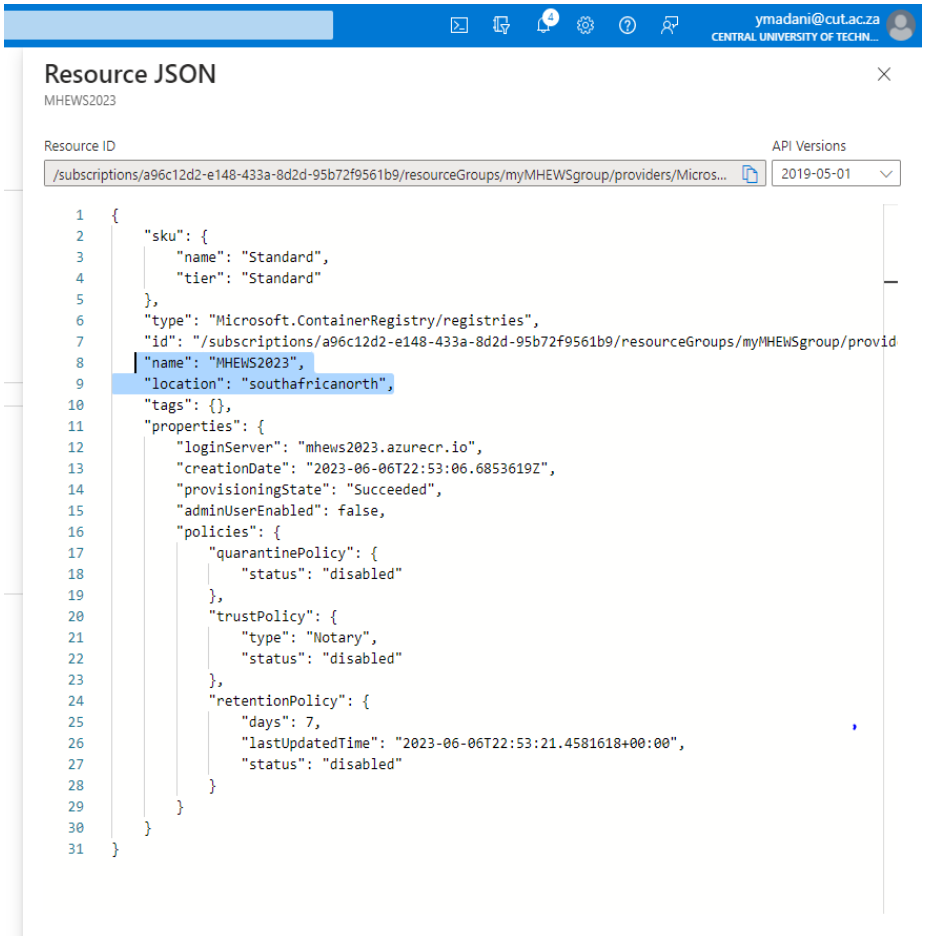


Figure 4.37: Code Snippet – A JSON REST API for accessing the MHEWS ACR in the Azure (Source: Author)

2. Login to the Azure Container Registry.

To initiate the process of pulling images, and pushing images into the Azure Container Registry; it is imperative to authenticate using an individual Azure Identity. This step is essential for establishing a secure connection between your local environment and Azure. Use the Azure CLI command 'az acr login --name <registry-name>' to authenticate. Authentication ensures that the user has the necessary permissions to push Docker images to the specified ACR instance. Figures 4.19, 4.20, and 4.21 illustrate authentication process.

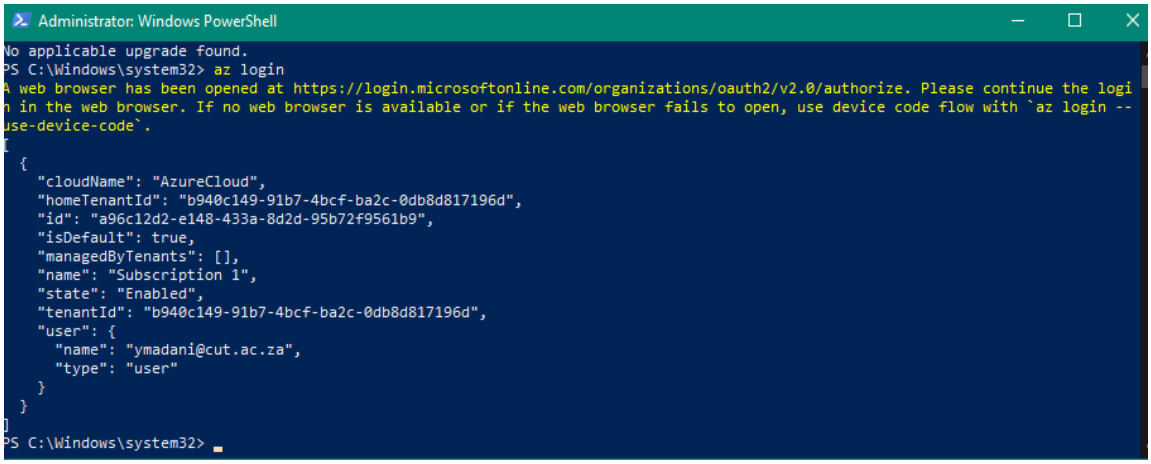


Figure 4.38: Azure Cloud Login using PowerShell (Source: Author)

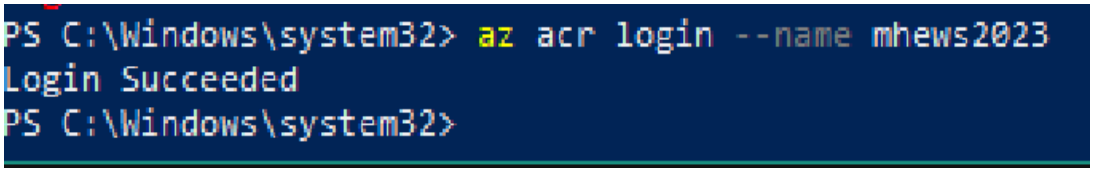


Figure 4.39: ACR mhews2023 Login Successfully (Source: Author)

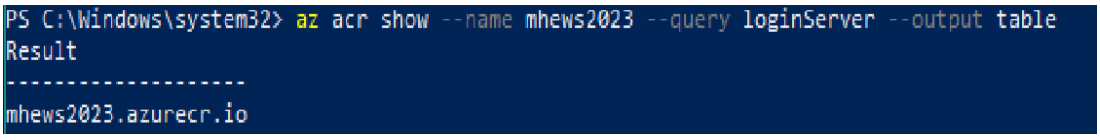


Figure 4.40: ACR mhews2023 Login Server (Source: Author)

3. Tag a Container.

Tagging a container is a critical step in the containerisation workflow, as it allows the Docker image to be uniquely identified and prepared for deployment to a specific container registry. This process involves associating the image with a specific tag that includes the address of the registry such as Azure Container Registry (ACR) to which the image will be pushed. Tags play a pivotal role in routing the container image to its intended destination

in the cloud or other deployment environments, ensuring seamless integration into the target infrastructure. The following steps are followed to tag a container.

Step 1: Listing Available Container Images

To begin the tagging process, the first step is to identify the available Docker images on the system. This is accomplished by running the 'docker images' command. The output of this command provides a comprehensive list of all Docker images currently present, along with their associated image IDs, repository names, and tags. This information is necessary to ensure that the correct image is selected for tagging.

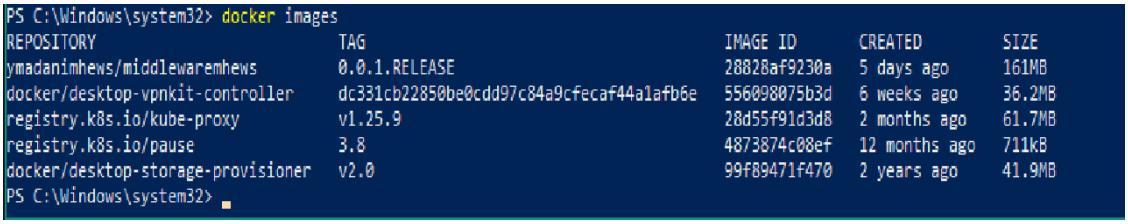


Figure 4.41: Viewing container Docker Images (Source: Author)

Step 2: Tagging the Docker Image with the Registry Address

Once the desired image is identified, the next step involves tagging the image with the login server address of the target registry. The tag serves as a unique identifier, specifying the destination for pushing the image.

Figure 4.42: Tagging Docker Images with the Login Server (Source: Author)

Step 3: Verifying the Tagging Process

After the image has been tagged, it is essential to verify that the tag has been correctly applied. This is done by running the docker images command once again, which will list all available images and their associated tags. The newly tagged image should now appear with the registry login server and repository information. This step ensures that the image is properly labelled and ready for the next phase of the deployment process.

Figure 4.43: Successfully Tagged Docker Image with the Login Server, verifying that the image is now ready for pushing to Azure Container Registry (Source: Author)

4. Docker Image into Azure Container Registry

To deploy a Docker image to the ACR, the 'docker push' command is employed. This command initiates the transfer of the locally built Docker image to the specified registry instance in Azure. The image is pushed to a repository within the ACR, allowing for centralized storage and version control of the container image. Figure 4.25 demonstrates the process of pushing the Docker image to the ACR, showcasing the command execution and the successful transfer of the image to the registry.

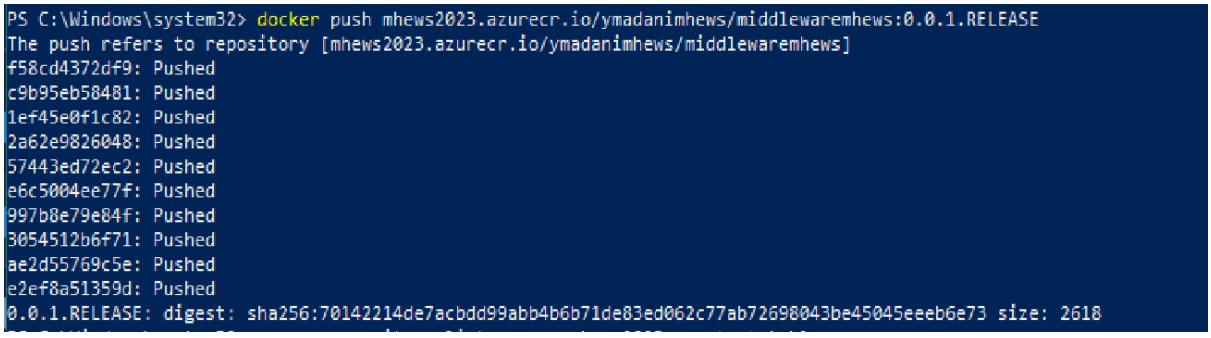


Figure 4.44: Push Docker Image to the ACR (Source: Author)

5. Return a list of images that have been pushed to your ACR instance

Once the image is successfully uploaded, it is crucial to verify the contents of the ACR repository to ensure that the images have been properly transferred. This is accomplished by using the az acr repository list command, which returns a list of all the container images stored in the registry instance. This command helps ensure proper image management and provides visibility into all stored images, allowing administrators to monitor and track the repository contents effectively. Figure 4.26 displays the output of the az acr repository list

command, listing all Docker images currently available in the ACR instance, providing validation of the image deployment process.

```
PS C:\Windows\system32> az acr repository list --name mhews2023 --output table
Result
--------------------------
ymadanimhews/middlewaremhews
```

Figure 4.45: List of Docker images pushed to the ACR (Source: Author)

4.3.1.3. Kubernetes Services

Kubernetes is employed to orchestrate the deployment of the containerised MHEWS application. It enables the system to achieve high availability, load balancing, and automatic scaling. Figures 4.27 through 4.30 detail the process of creating and managing the Kubernetes cluster.

1. Create a Kubernetes Cluster

   To enable an Azure Kubernetes Service (AKS) cluster to interact with other Azure resources, Azure automatically generates a cluster identity. This identity serves as a unique identifier for the AKS cluster, allowing it to securely access and manage other services and resources within your Azure subscription.

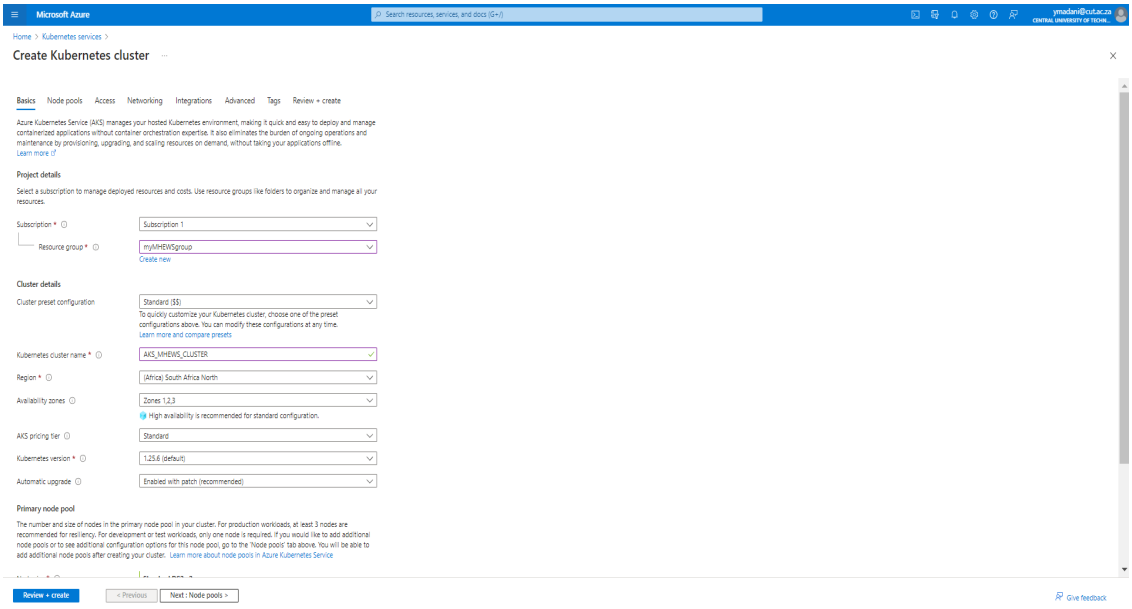


Figure 4.46: Creating a Kubernetes Cluster in Azure Cloud (Source: Author)

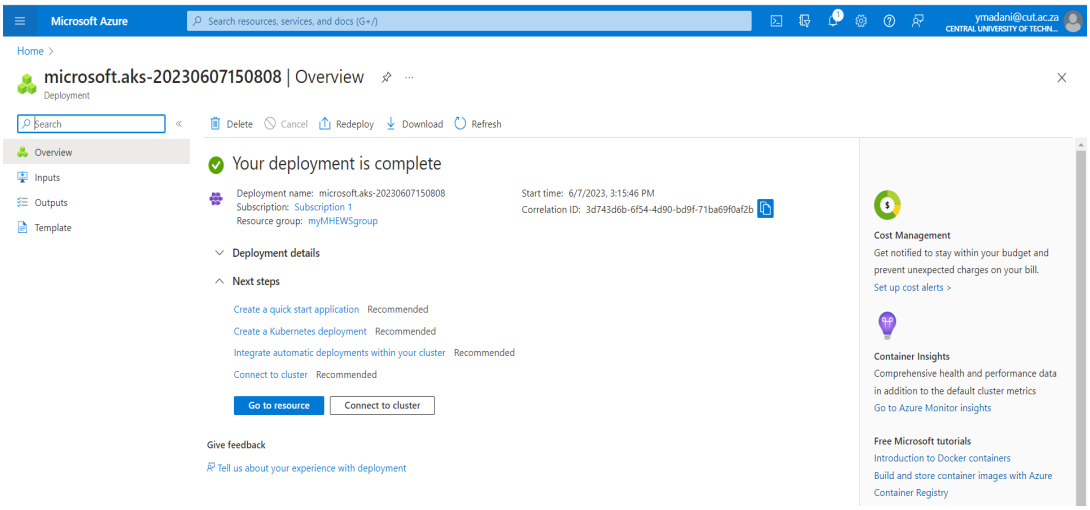


Figure 4.47: Kubernetes Cluster Overview in Azure Cloud (Source: Author)

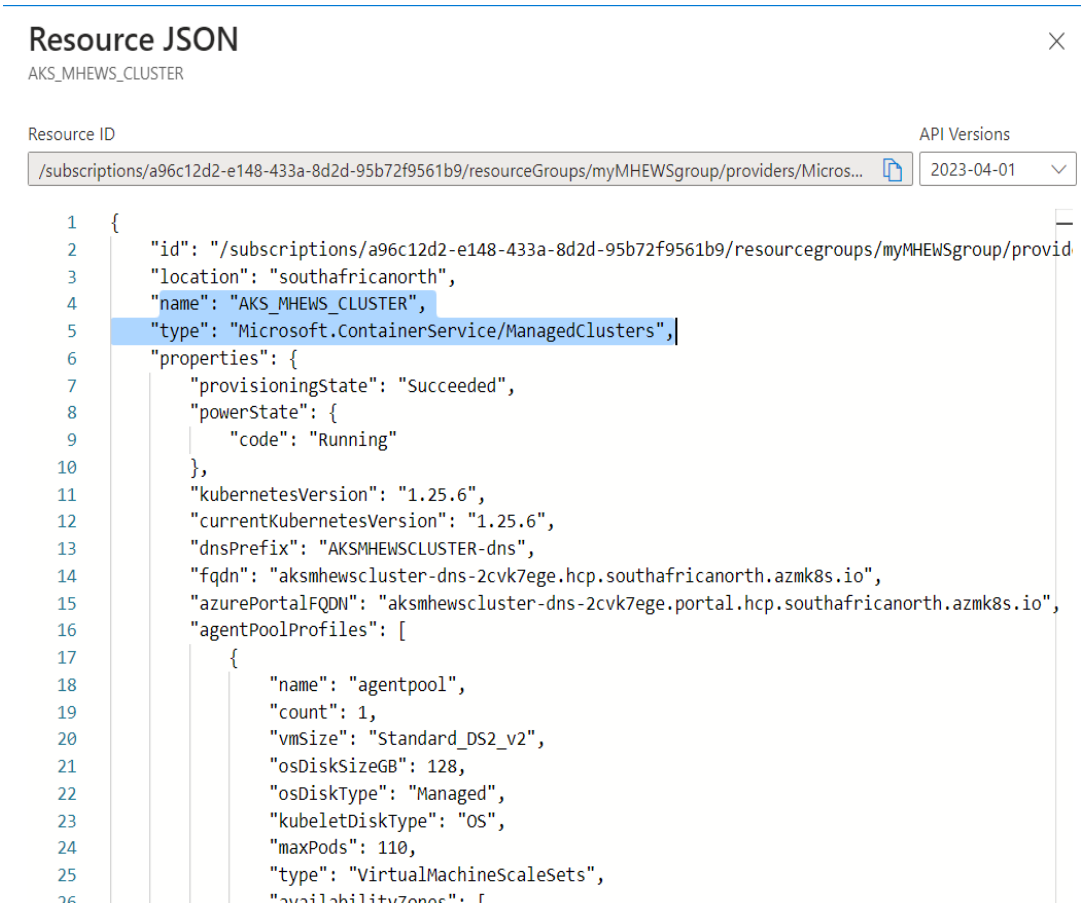


Figure 4.48: Code Snippet – A JSON REST API for accessing Kubernetes Cluster in Azure Cloud (Source: Author)

2. To establish a connection with the AKS Cluster, execute the command ‘kubectl get nodes’, this will verify the connection by returning a detailed list of the nodes within the cluster.

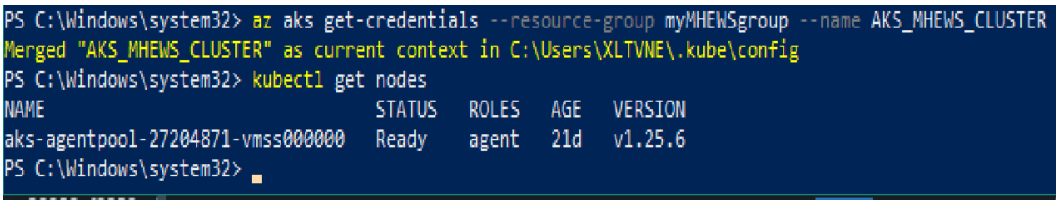


Figure 4.49: Nodes in Kubernetes Cluster in Azure Cloud (Source: Author)

4.3.1.4 Azure Kubernetes Cluster

Deploying an application to an Azure Kubernetes Cluster (AKS) involves several critical steps to ensure seamless integration, scaling, and deployment. This section outlines the process of updating the Kubernetes manifest file with the necessary configurations, including the Azure Container Registry (ACR) login server, to enable the deployment of containerised applications.

The Kubernetes manifest file plays a central role in defining the application's desired state within the cluster. This includes defining the deployment strategy, specifying the number of replicas, and detailing the container image to be used. Properly configuring these files is essential for ensuring a smooth and scalable deployment. Figure 4.31 to Figure 4.34 outline the deployment steps.

1. The Manifest File is named the deployment file (see Appendix A).

   The first step in deploying the application to AKS is updating the Kubernetes manifest file. The manifest file, commonly named (deployment.yml) must include the ACR login server in the image definition. The az acr list command is used to retrieve the appropriate login server details for your Azure Container Registry, ensuring that the cluster can pull the container image from the registry. Once the manifest file has been updated, the deployment process can be initiated by executing the ‘kubectl apply -f deployment.yml’ command, which ensures that the application is deployed with the configurations specified in the manifest. Figure 4.31 provides an illustration of the deployment process using the kubectl apply command.

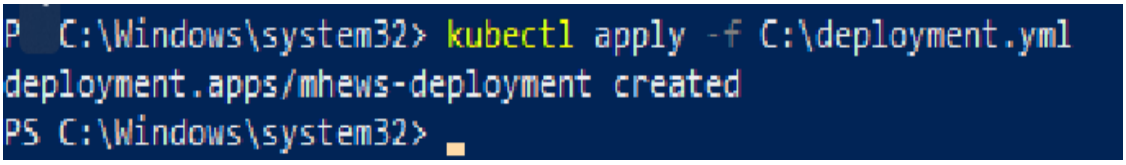


Figure 4.50: Deploying deployment.yml using Kubectl (Source: Author)

2. Create the Service File (see Appendix B).

After deploying the application, a Kubernetes service file must be created. The service file commonly named (mhews-service.yml) defines how the application will be exposed to the internet or other services within the cluster. A Kubernetes service provides an abstraction over the network, allowing communication between different applications or users and the cluster. Once the service file is properly configured, it is applied to the cluster using the 'kubectl apply -f mhews-service.yml' command, allowing external users to access the application. Figure 4.32 illustrates the use of the Kubernetes service file to expose the front end of the MHEWS application.

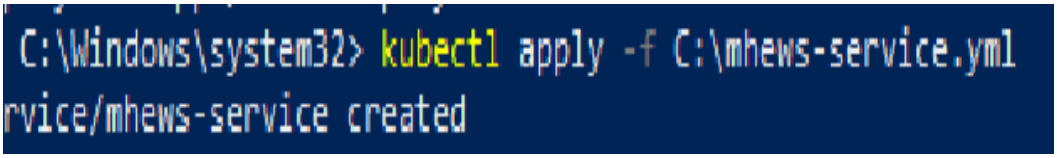


Figure 4.51: Using service file, mhews-service.yml to expose the application front end (Source: Author)

3. Create the Ingress File (see Appendix C).

To provide fine-grained control over HTTP and HTTPS routing within the cluster, an ingress file commonly named (mhwes-ingress.yml) is created. The ingress file specifies the routing rules that govern how external traffic reaches the internal services of the application. It provides a means for managing multiple services within the cluster through a single external IP address, which can simplify load balancing and traffic routing.

The ingress file allows for the definition of specific routes, ensuring that traffic directed to the application's web interface is properly managed. The ingress controller is responsible for enforcing these routing rules, making it a crucial part of the deployment strategy. Once the ingress file is defined, it can be applied using the 'kubectl apply -f mhwes-ingress.yml' command, which ensures that the ingress controller applies the routing policies specified in the file. Figure 4.33 depicts the use of kubectl to apply the ingress file and implement the routing rules for the MHEWS application.

```
PS C:\Windows\system32> kubectl apply -f C:\mhews-ingress.yaml
ingress.networking.k8s.io/mhews-ingress created
PS C:\Windows\system32>
```

Figure 4.52: Using Kubectl to apply mhwes-ingress.yml to the Cluster (Source :Author)

4. Test the application deployment.

   The final step in the deployment process involves testing the application to ensure that it has been successfully deployed and is functioning as intended. This is done using the kubectl get services command, which lists all the services running in the cluster and provides details such as the Cluster-IP address, external IP addresses, and ports.

   This step verifies that the application is accessible and that all network configurations, including the service and ingress settings, are working correctly. Testing also ensures that the application can scale according to the specified parameters in the manifest file, providing resilience and load-balancing capabilities. Figure 4.34 shows the results of testing the cluster deployment, including retrieving the Cluster-IP address for accessing the deployed MHEWS application.

```
PS C:\Windows\system32> kubectl get service
NAME            TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
kubernetes      ClusterIP   10.0.0.1       <none>        443/TCP   22d
mhews-service   ClusterIP   10.0.176.241   <none>        80/TCP    12m
PS C:\Windows\system32>
```

Figure 4.53: Testing the cluster deployment and getting Cluster-IP address (Source: Author)

4.3.1.5. Semantic Integration and Representation Inside Docker

Integrating semantic technologies into Docker environments facilitates seamless communication and data exchange between containerized applications. To achieve semantic integration, and representation within Docker, Apache Jena is used as the primary tool for managing, and querying semantic data. Apache Jena is a powerful tool for Java, an extensible semantic web framework that provides tools for creating, querying, and managing SPARQL and RDF data. To integrate semantic representation with Docker, firstly must configure the

MHEWS containerized application to use Apache Jena for semantic data processing enabling seamless interaction with semantic data.

Achieving semantic representation and integration inside Docker with Apache Jena requires several steps. Figure 4.35 to Figure 4-43 detail the steps involved in setting up Apache Jena for semantic integration.

1. Firstly, the '**docker pull httpd**' command is initiated to pull the Apache Jena to the docker command.

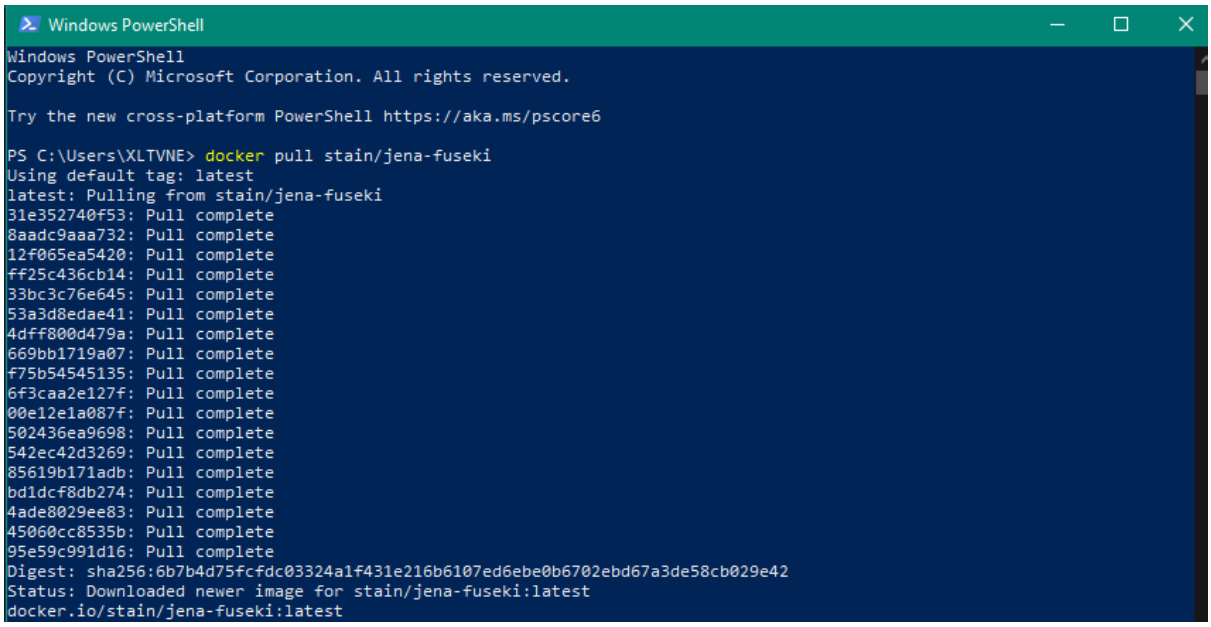


Figure 4.54: Pulling Apache Jena to Docker (Source: Author)

2. Choosing correct path to web application files and integrating web application to Apache Jena Port 8000 running on a local machine.

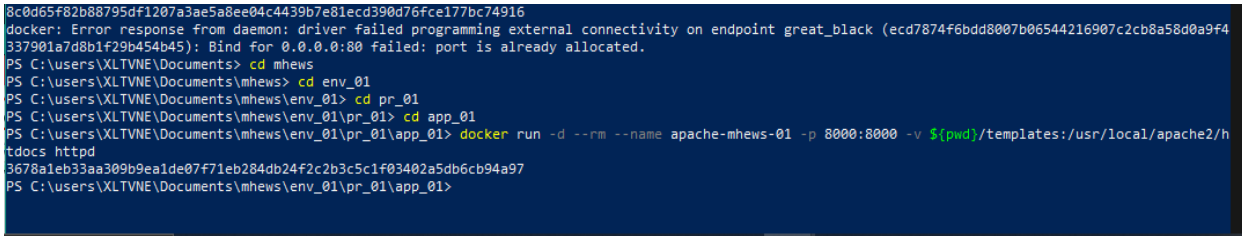


Figure 4-55: Integrating MHEWS application with Apache Jena port 8000 (Source: Author)

3. Success integration and Apache Jena container for web application created successful.

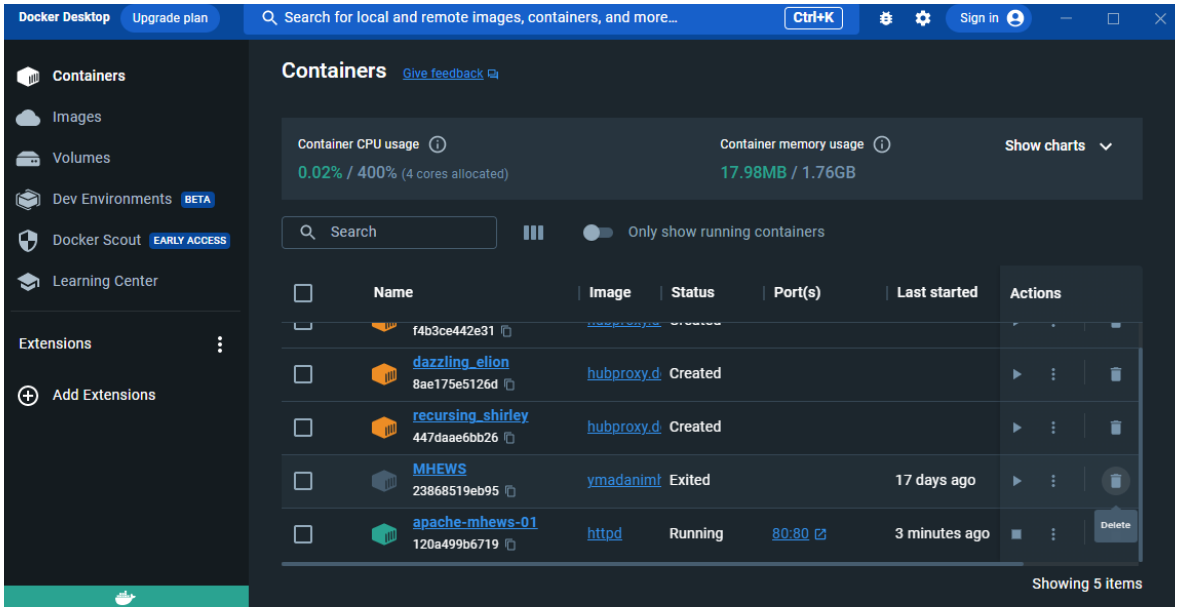


Figure 4.56: Successful Integration of MHEWS application with Apache Jena port 8000 (Source: Author)

4. Creating RDF files and SPARQL query.

Run Apache Jena Fuseki on local host by the following command.

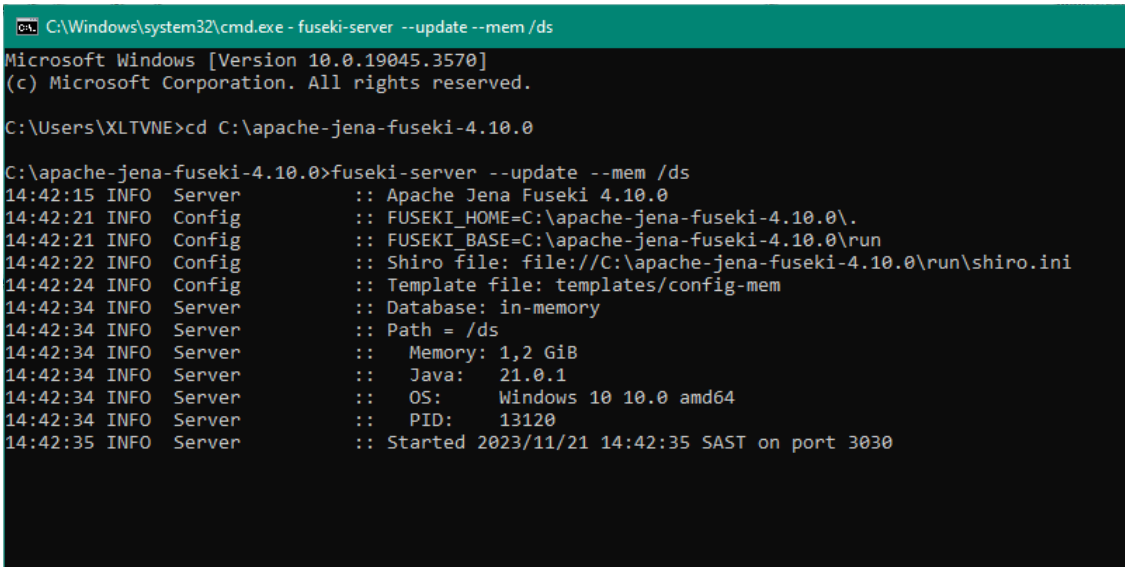


Figure 4.57: Running Apache Jena on a Local Host (Source: Author)

From the above command, a local port will be allocated to run the UI of Apache Jena

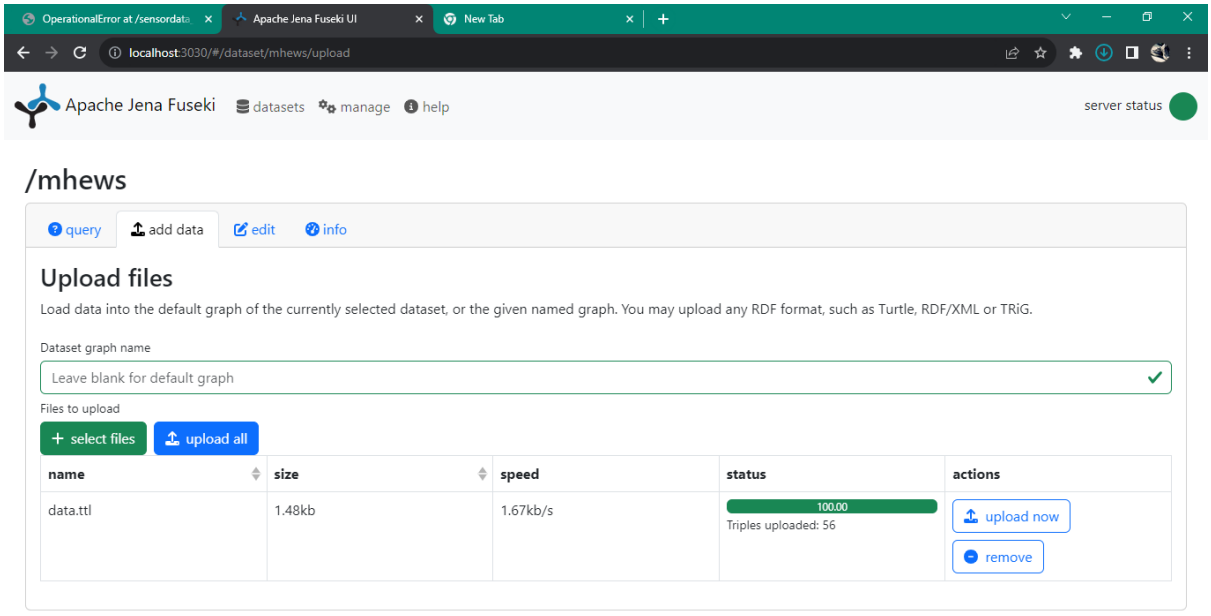


Figure 4.58: Apache Jena UI overview (Source: Author)

Creating Terse RDF Triple Language turtle data and save as ***data.ttl***, for expressing data in RDF data model.

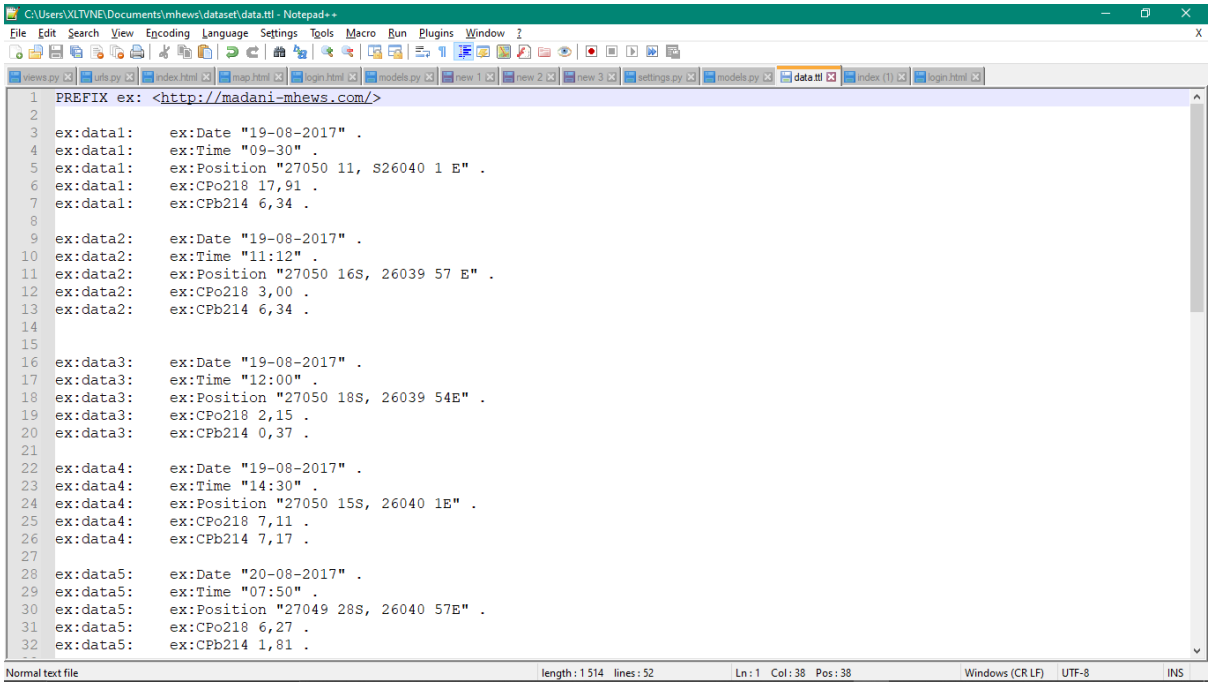


Figure 4.59: RDF-Turtle Data (Source: Author)

Uploading the **data.ttl** on Apache Jena**.**

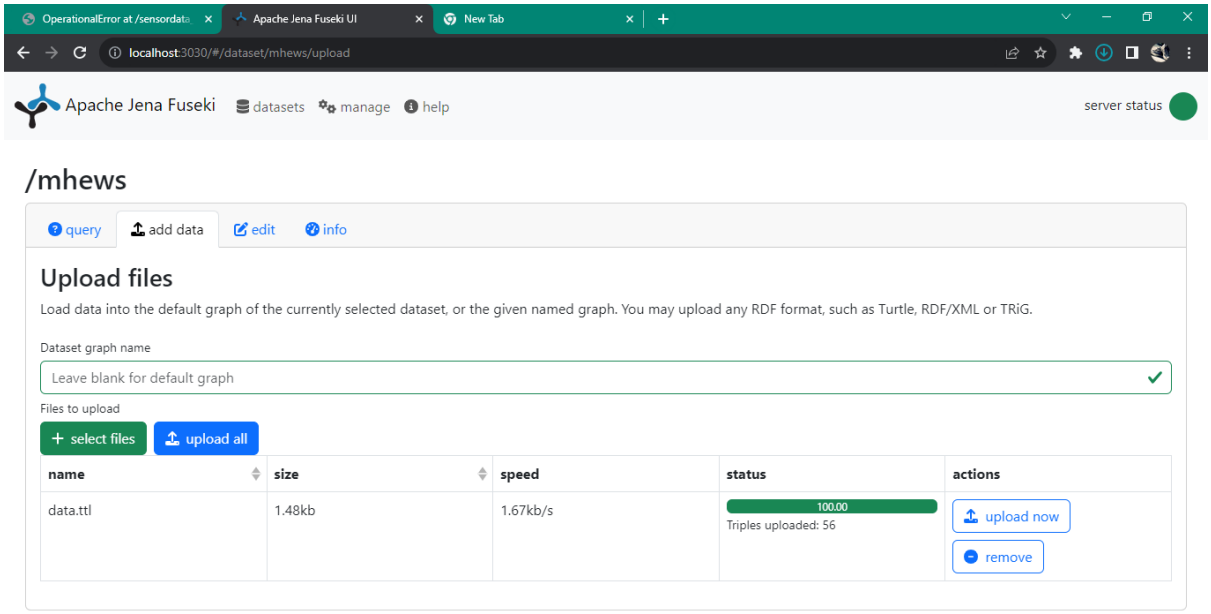


Figure 4.60: data.ttl uploaded (Source: Author)

SPARQL query execution…

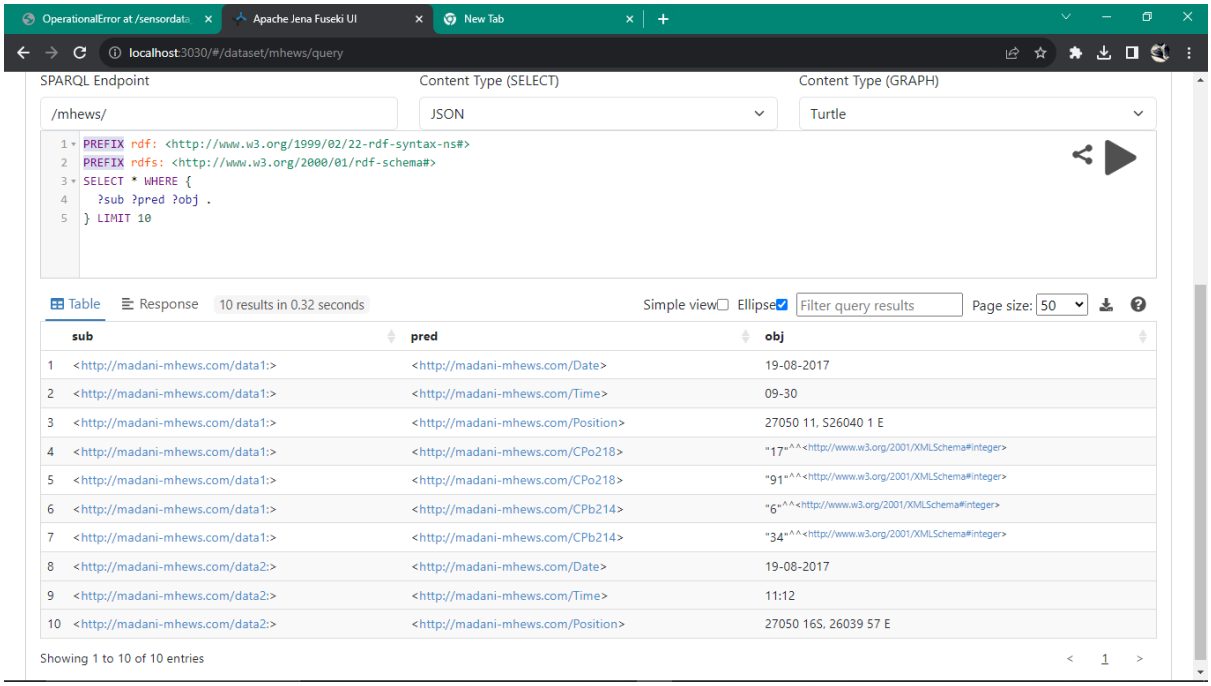


Figure 4.61: Running SPARQL query (Source: Author)

Figure 4.43 below shows the status of every action and execution happening on Apache Jena.

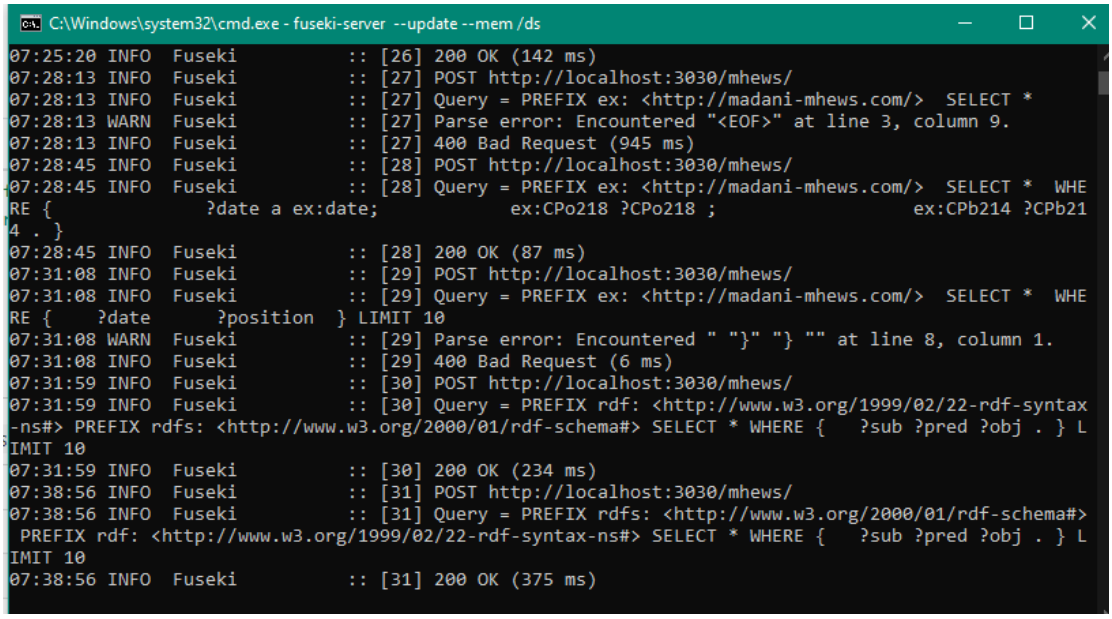


Figure 4.62: Apache Jena Status (Source: Author)

4.4 Data Analysis

This section shows how Statistical Package for the Social Sciences (SPSS) was used as a reliable guide to explore and understand air pollution data obtained through Wireless Sensor Networks on specific dates and times in different positions. Analyzing air pollution data collected by WSN is essential to identify trends, patterns, and potential sources of pollution and develop informed strategies to mitigate the impacts air pollution has on environmental health. The data was then imported into the SPSS, ensuring that the Randon (Rn) concentration values, and GPS coordinates were correctly recognized as numeric data.

4.4.1 Case Study Overview and Data Collection Context

The case study was conducted in the Lejweleputswa district, a region heavily influenced by mining and agriculture, which are key drivers of environmental pollution. The district's socio-economic structure and its reliance on these industries made it essential to gather both qualitative and quantitative data to assess the impact of pollution on public health and the environment. The study's dual approach aimed to integrate indigenous knowledge with real-time environmental data, ensuring a comprehensive understanding of pollution patterns and their broader implications. The data collection context in Lejweleputswa was defined by the intersection of traditional practices and modern technological solutions. The local community's

knowledge of environmental shifts, particularly concerning air quality and pollution, was considered invaluable for supplementing the data captured by wireless sensor networks. These networks, deployed strategically across urban, rural, and industrial zones, gathered key environmental variables such as particulate matter, temperature, and air quality, providing real-time data essential for the MHEWS prototype.

A mixed-methods approach was employed, combining both qualitative and quantitative data collection techniques to ensure a robust dataset for system development. The qualitative data primarily obtained through interviews with local community members and mining workers. These individuals, selected for their in-depth knowledge of the region's environmental conditions, shared insights into the observable signs of pollution and traditional methods of environmental monitoring. The aim was to capture local, experiential knowledge that could complement sensor data and identify pollution indicators that may not be immediately detectable through technology. In parallel, quantitative data collected through wireless sensor networks installed across the district and mines. These sensors continuously monitored environmental variables and provided objective, real-time data on pollution levels, which were essential for feeding into the MHEWS framework. The combination of indigenous knowledge and sensor data offered a multifaceted approach to understanding pollution and its effects on public health, contributing significantly to the MHEWS development.

The data collected through this case study directly contributed to the development of the Semantic MHEWS by providing a rich, contextually grounded dataset that aligned with the system's objectives. Integrating both indigenous knowledge and quantitative sensor data, the system was able to capture a more complete and nuanced view of environmental health risks in the Lejweleputswa district. This dual approach not only ensured the relevance of the system to the local community but also enabled the MHEWS to respond to both immediate pollution events and longer-term environmental trends. The inclusion of indigenous knowledge was particularly significant in framing the data collection process, as it provided insights into pollution indicators that sensors alone might not detect. Furthermore, the quantitative data collected via the sensor networks complemented these qualitative insights, creating a holistic dataset that could be used to generate accurate, real-time pollution warnings for the community. Thus, the case study’s data collection practices were integral to the system’s design, ensuring that the MHEWS was not only technologically advanced but also contextually relevant to the specific environmental challenges faced by the Lejweleputswa district.

4.4.2 Data Description and Analysis

The analysis focused on providing both descriptive and inferential insights into the pollution levels, with a particular emphasis on understanding their implications for the development of the MHEWS. The selection of statistical metrics, including mean, median, mode, and standard deviation, is guided by the need to capture a comprehensive understanding of the air quality data. These metrics are chosen based on their ability to highlight both central tendencies and data variability, which are essential for identifying trends and anomalies in pollution levels. Specifically:

- Mean: The mean is used to identify the average pollution levels across different monitoring locations. This measure provided an overall sense of the typical pollution level within the region and helped in understanding whether pollution consistently exceeded acceptable thresholds.
- Median: The median is used to assess the distribution of pollution data, particularly in cases where the data was not symmetrically distributed. The median offered insight into the "middle" of the dataset, allowing for a more accurate understanding of pollution levels when extreme values might distort the mean.
- Mode: The mode is included to identify the most frequently occurring pollution levels. This metric is particularly useful in pinpointing specific ranges of pollution that occurred most often, which could indicate recurring pollution events or problematic hotspots.
- Standard Deviation: Standard deviation is critical in assessing the variability in pollution levels. A high standard deviation indicated that pollution levels fluctuated significantly across locations or time periods, suggesting the need for a more adaptive warning system capable of responding to changes in air quality.

The chosen metrics provided important insights that directly influenced the design and functionality of the Semantic MHEWS. For instance, the mean and median values helped in determining baseline pollution levels, which were essential for setting thresholds for pollution alerts. If the mean pollution level exceeded a certain threshold, the system would trigger an early warning to the community, informing them of potentially hazardous conditions. The mode was particularly valuable in identifying recurring pollution hotspots, enabling the MHEWS to be more focused in its warnings for areas that consistently experienced high pollution levels.

This allowed the system to prioritize regions most in need of monitoring and intervention, optimising resource allocation and enhancing the effectiveness of the system.

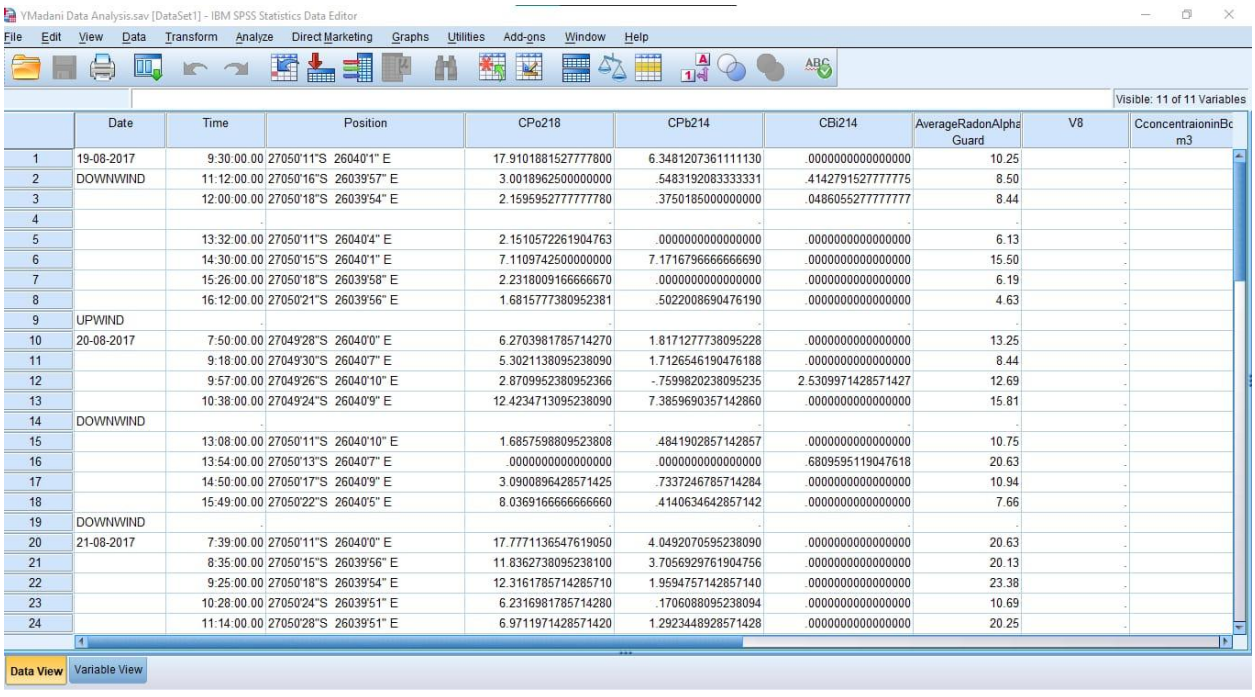
Additionally, the standard deviation provided insights into pollution variability, which was crucial for developing adaptive algorithms within the MHEWS. If pollution levels were highly variable, the system could be designed to issue more frequent or dynamically adjusted alerts based on real-time data fluctuations. This adaptability is key for ensuring that the system remains responsive to both immediate and long-term environmental changes, improving public safety. Incorporating these statistical methods, the data analysis not only helped describe the current state of air pollution but also played a pivotal role in informing the MHEWS design. The metrics guided the decision-making process, shaping how the system would interpret and respond to data in real-time, and ensuring that the MHEWS was both contextually relevant and technologically robust in addressing the pollution challenges faced by the Lejweleputswa district.

- Data obtained for this study represented the measurement of Randon (Rn) concentration [C (Po-218), C (Pb-214), and C (Bi-214)] in Becquerels per cubic meter (Bq/m^3).
- Each measurement was associated with specific GPD coordinates, which indicated the geographic location where the measurement was taken.
- Data was collected over several days, and it appeared to alternate between UPWIND and DOWNWIND measurements.
- Average Randon (Alpha Guard) was found to be an average of the Rando concentrations for each set of measurements on a particular date or day.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Date	Time	Position	C (Po-218)	C (Pb-214	C (Bi-214)	Average Radon (AlphaGuard)		C = concentraion in Bq/m^3			
2	19-08-2017	09:30	27°50'11"S 26°40'1" E	17,91018815	6,348120736	0	10,25					
3	DOWNWIND	11:12	27°50'16"S 26°39'57" E	3,00189625	0,548319208	0,414279153	8,5					
4		12:00	27°50'18"S 26°39'54" E	2,159595278	0,3750185	0,048605528	8,44					
5												
6		13:32	27°50'11"S 26°40'4" E	2,151057226	0	0	6,13					
7		14:30	27°50'15"S 26°40'1" E	7,11097425	7,171679667	0	15,5					
8		15:26	27°50'18"S 26°39'58" E	2,231800917	0	0	6,19					
9		16:12	27°50'21"S 26°39'56" E	1,681577738	0,502200869	0	4,63					
10	UPWIND											
11	20-08-2017	07:50	27°49'28"S 26°40'0" E	6,270398179	1,817127774	0	13,25					
12		09:18	27°49'30"S 26°40'7" E	5,30211381	1,712654619	0	8,44					
13		09:57	27°49'26"S 26°40'10" E	2,870995238	-0,759982024	2,530997143	12,69					
14		10:38	27°49'24"S 26°40'9" E	12,42347131	7,385969036	0	15,81					
15	DOWNWIND											
16		13:08	27°50'11"S 26°40'10" E	1,685759881	0,484190286	0	10,75					
17		13:54	27°50'13"S 26°40'7" E	0	0	0,680959512	20,63					
18		14:50	27°50'17"S 26°40'9" E	3,090089643	0,733724679	0	10,94					
19		15:49	27°50'22"S 26°40'5" E	8,036916667	0,414063464	0	7,66					
20	DOWNWIND											
21	21-08-2017	07:39	27°50'11"S 26°40'0" E	17,77711365	4,04920706	0	20,63					
22		08:35	27°50'15"S 26°39'56" E	11,83627381	3,705692976	0	20,13					
23		09:25	27°50'18"S 26°39'54" E	12,31617857	1,959475714	0	23,38					
24		10:28	27°50'24"S 26°39'51" E	6,231698179	0,17060881	0	10,69					
25		11:14	27°50'28"S 26°39'51" E	6,971197143	1,292344893	0	20,25					
26	DOWNWIND											

Active data | Passive data

Figure 4;63: WSN Data Collection in Microsoft Excel-Format (Source: Author)



	Date	Time	Position	CPo218	CPb214	CBi214	AverageRadonAlpha Guard	V8	CconcentraioninBc m3
1	19-08-2017	9:30:00.00	27050'11"S 26040'1" E	17.9101881527777800	6.3481207361111130	.0000000000000000	10.25	.	
2	DOWNWIND	11:12:00.00	27050'16"S 26039'57" E	3.0018962500000000	.5483192083333331	.4142791527777775	8.50	.	
3		12:00:00.00	27050'18"S 26039'54" E	2.1595952777777780	.3750185000000000	.0486055277777777	8.44	.	
4		.		.	.	.	.	.	
5		13:32:00.00	27050'11"S 26040'4" E	2.1510572261904763	.0000000000000000	.0000000000000000	6.13	.	
6		14:30:00.00	27050'15"S 26040'1" E	7.1109742500000000	7.1716796666666690	.0000000000000000	15.50	.	
7		15:26:00.00	27050'18"S 26039'58" E	2.2318009166666670	.0000000000000000	.0000000000000000	6.19	.	
8		16:12:00.00	27050'21"S 26039'56" E	1.6815777380952381	.5022008690476190	.0000000000000000	4.63	.	
9	UPWIND	.		.	.	.	.	.	
10	20-08-2017	7:50:00.00	27049'28"S 26040'0" E	6.2703981785714270	1.8171277738095228	.0000000000000000	13.25	.	
11		9:18:00.00	27049'30"S 26040'7" E	5.3021138095238090	1.7126546190476188	.0000000000000000	8.44	.	
12		9:57:00.00	27049'26"S 26040'10" E	2.8709952380952366	-.7599820238095235	2.5309971428571427	12.69	.	
13		10:38:00.00	27049'24"S 26040'9" E	12.4234713095238090	7.3859690357142860	.0000000000000000	15.81	.	
14	DOWNWIND	.		.	.	.	.	.	
15		13:08:00.00	27050'11"S 26040'10" E	1.6857598809523808	.4841902857142857	.0000000000000000	10.75	.	
16		13:54:00.00	27050'13"S 26040'7" E	.0000000000000000	.0000000000000000	.6809595119047618	20.63	.	
17		14:50:00.00	27050'17"S 26040'9" E	3.0900896428571425	.7337246785714284	.0000000000000000	10.94	.	
18		15:49:00.00	27050'22"S 26040'5" E	8.0369166666666660	.4140634642857142	.0000000000000000	7.66	.	
19	DOWNWIND	.		.	.	.	.	.	
20	21-08-2017	7:39:00.00	27050'11"S 26040'0" E	17.7771136547619050	4.0492070595238090	.0000000000000000	20.63	.	
21		8:35:00.00	27050'15"S 26039'56" E	11.8362738095238100	3.7056929761904756	.0000000000000000	20.13	.	
22		9:25:00.00	27050'18"S 26039'54" E	12.3161785714285710	1.9594757142857140	.0000000000000000	23.38	.	
23		10:28:00.00	27050'24"S 26039'51" E	6.2316981785714280	.1706088095238094	.0000000000000000	10.69	.	
24		11:14:00.00	27050'28"S 26039'51" E	6.9711971428571420	1.2923448928571428	.0000000000000000	20.25	.	

Figure 4.64: WSN Data Collection in SPSS (Source: Author)

Clustered Bar Chart

The chart below shows changes in different hazards on different dates. From the data received, C (Po-218) was found as the highest hazard at all different dates the data was collected.

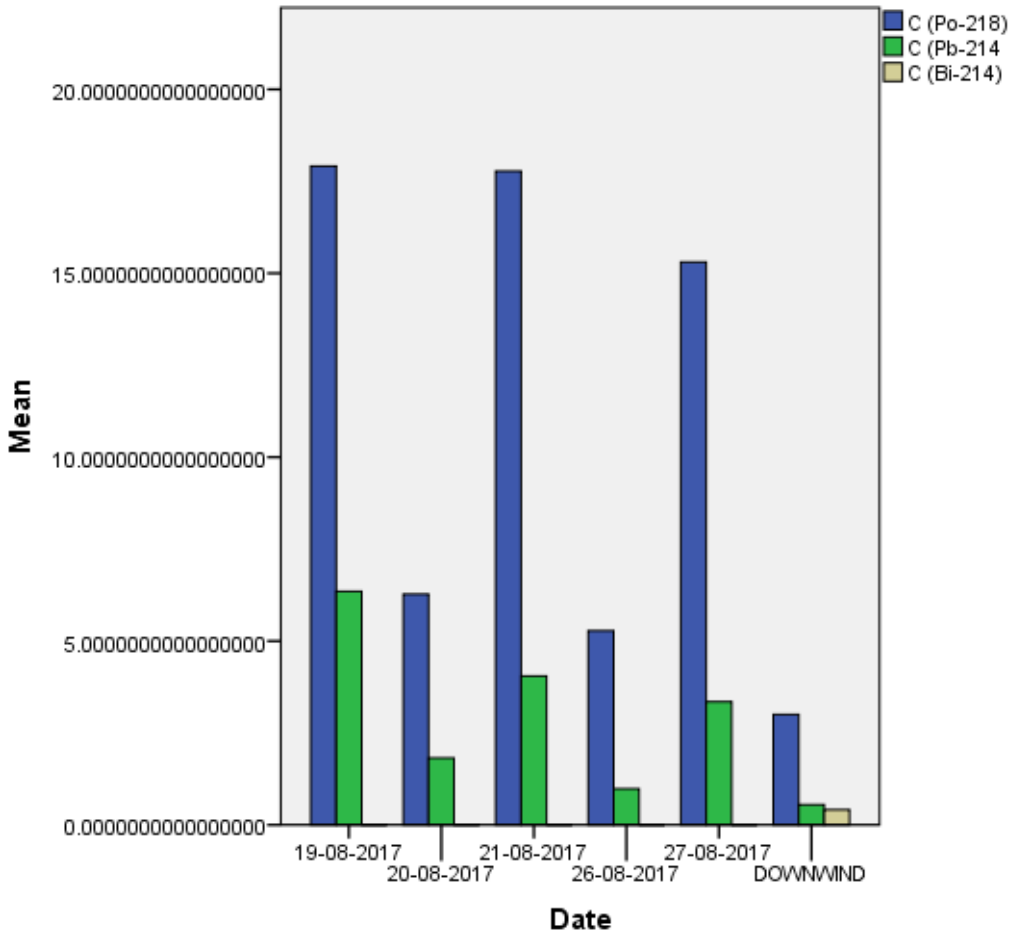


Figure 4.65: Clustered Bar Graph of WSN Data (Source: Author)

Due to some factors (i.e., environmental surroundings, objects, and air ventilations), the level of C (Po-218) decreased because of the factors provided. With this level of hazard, the second hazard was found to have high concentration of C (Pb-214). The figure below 4-47, depicts how the level of the hazards decreased at different locations (positions. CPb-214 at Position 27049'26"S 26040'10" E, which shows the lowest level of -.7599820238095235.

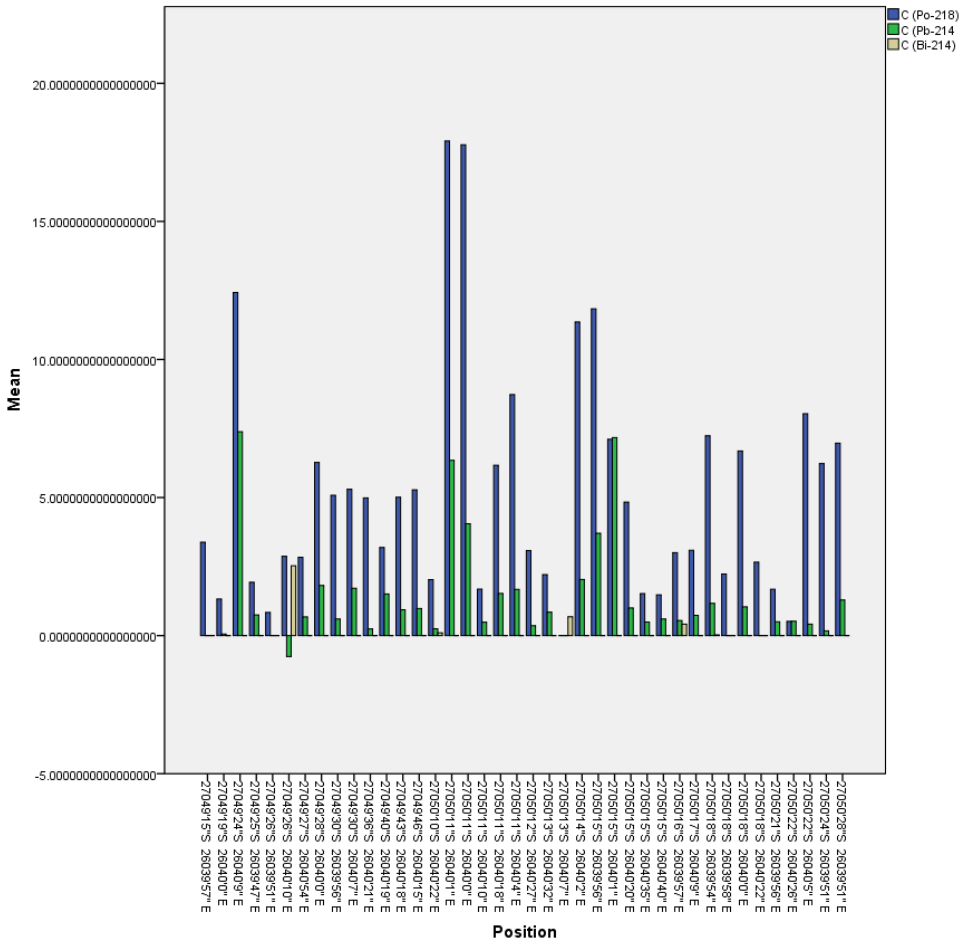


Figure 4.66: Bar-Graph of WSN Data (Source: Author)

The table below shows the mean of three hazards based on data collected, the mean was calculated based on the dates the data was collected.

	Date							
		19-08-2017	20-08-2017	21-08-2017	26-08-2017	27-08-2017	DOWN WIND	UPW IND
	Mean	Mean	Mean	Mean	Mean	Mean	Mean	Mean
C (Po-218)	4.36790442339065 25	17.9101881527777 800	6.27039817857142 70	17.7771136547619 050	5.28108726190476 20	15.3029350833333 300	3.00189625000000 00	.
C (Pb-214	1.07242645370370 38	6.34812073611111 30	1.81712777380952 28	4.04920705952380 90	.98177034523809 54	3.34881574999999 95	.54831920833333 1	.

C (Bi-214)	.093438301036155 2	.000000000000000 0	.000000000000000 0	.000000000000000 0	.000000000000000 0	.000000000000000 0	.414279152777777 5	.

Table 4.2: Mean of the three Hazards (Source: Author)

In addition, a frequency table depicting the calculation of Mean, Median, Mode, Standard Deviation, Skewbess and Kurtosis is presented below.

Statistics

		Average Radon (Alpha Guard)	C (Pb-214	C (Po-218)	C (Bi-214)
N	Valid	42	42	42	42
	Missing	10	10	10	10
Mean		10.8133	1.326207457294029	5.304480424366968	.089953761668556
Median		8.9050	.641065761904762	3.286104029761905	.000000000000000
Mode		8.44	.000000000000000	.000000000000000[a]	.000000000000000
Std. Deviation		5.38722	1.871960168344260	4.572490812509376	.404681303622688
Skewness		.824	2.169	1.413	5.703
Std. Error of Skewness		.365	.365	.365	.365
Kurtosis		-.147	4.371	1.370	34.292
Std. Error of Kurtosis		.717	.717	.717	.717

Sum		454.16	55.700713206349210 0	222.78817782341267 00	3.7780579900793647
Percentiles	25	7.1875	.24860544047619 0	2.120843604166667	.000000000000000
	50	8.9050	.64106576190476 2	3.286104029761905	.000000000000000
	75	13.2825	1.575184458333333	6.757716669642857	.000000000000000

a. Multiple modes exist. The smallest value is shown.

Table 4-3: Frequency Table (Source: Author)

4.5 Conclusion

In conclusion, containerizing the MHEWS application with Docker, and orchestrating it with Kubernetes as well as incorporating semantic middleware components has proven to be a solid approach to system deployment. This modernization effort addresses various aspects of system analysis and design including gathering functional requirements and non-functional requirements, and defining the required hardware and software components. Additionally, data modelling is required through the development of use cases, entity-relationship diagrams (ERD), and class diagrams. By leveraging these comprehensive methodologies, MHEWS is now better equipped to meet scalability, portability, and maintainability requirements, providing users with great efficiency, and flexibility. Importantly, semantic integration and representation were achieved inside Docker by utilising Apache Jena to enhance system interoperability, and improve data understanding between systems.

CHAPTER FIVE

CONCLUSION AND DISCUSSION

5.1 Introduction

This chapter highlights the findings and core issues from the previous chapters of this study. Throughout this research, the development, implementation, and potential impact of semantic middleware systems designed to improve communities' ability to manage and respond to MHEWS have been explored. This chapter summarizes the key findings, discusses the implementation, and outlines possible future directions for research and development in this critical area.

5.2 Summary

This section presents the summary of previous chapters of this study.

5.2.1. Chapter One

The first chapter introduced the background, context, motivation and objectives of this study and the importance of MHEWS to mitigate the impact of pollution in communities surrounded by mining operations in Lejweleputswa district, South Africa.

5.2.2. Chapter Two

Chapter two provides a comprehensive literature review on early warning systems, environmental monitoring systems, cloud computing, microservices, semantic technologies and their application in disaster risk reduction or management. Semantic interoperability, data integration, and the role of the semantic middleware layer in increasing the effectiveness of PEWS concepts were also explored.

5.2.3. Chapter Three

In Chapter Three, the research methodology adopted in this study was discussed in detail. This included data collection methods, system design and semantic technologies. We also discussed the development process, data modelling, ontology, and system integration.

5.2.4. Chapter Four

Chapter four focused on the system design and implementation of semantic middleware for a MHEWS. We discussed the system architecture, containerization of the MHEWS application using Docker and Kubernetes and the system integration into existing infrastructure.

5.3. Discussion

This section presents the key research objectives identified at the beginning of our study, and summarize the key conclusions that emerged from our study. These objectives led the way in the development of semantic middleware for South Africa's MHEWS.

RO1: Develop a semantic framework for a MHEWS to and improve the automatic configuration of the EWS for data integration and system interoperability.

- Semantic Framework provided basis for system interoperability by establishing common semantics for data exchange.
- The development of a semantic framework enabled a common understanding of the data in the MHEWS components.
- The development of ontology played an important role in standardizing data representation and facilitating automatic configuration.

RO2: How can the integration of the systems through semantic middleware enhance or improve the decision-making process for disaster risk reduction and response by providing actionable insights based on the integrated data?

- Enhanced data interoperability: Integration through semantic middleware facilitated seamless communication between disparate systems, which improved the sharing and interoperability of data from various sources.
- Scalability and flexibility: The middleware was adaptable to various data sources and disaster types, allowing for a scalable system that could be applied to future disaster scenarios.
- Increased accuracy and relevance of warnings: The system integration allowed for more precise data aggregation, leading to the generation of more accurate and localised early warning signals.
- Informed response strategies: Semantic middleware provided a comprehensive view of potential hazards, which aided in developing more informed and timely response strategies to mitigate risks.

RO3: Develop and test a prototype middleware layer that semantically represents data from different EWS in the context for pollution monitoring in Lejweleputswa district, Free State.

- The prototype middleware layer was developed and successfully represents data from various sources.
- The middleware process and contextualize data for MHEWS components.
- Integration of heterogeneous data was achieved through semantic federation enabling the middleware layer to understand and process data.

RO4: Evaluate to what extent does the application of semantic middleware in the context of MHEWS facilitates systems heterogeneous EWS integration and interoperability.

- The application of semantic middleware within MHEWS substantially enhances the integration and interoperability of heterogeneous EWS holistically by standardizing data representation, enabling effective communication, and supporting system scalability.
- Increased interoperability was observed, allowing the various components of the system to communicate effectively, regardless of the technologies on which they are based.

In summary, data integration and system interoperability improved with the use of semantic middleware. The semantic framework and middleware developed in this research serve as essential tools to improve the accuracy and effectiveness of the EWS holistically by contributing to disaster risk reduction and community resilience. The research hypothesis was supported by a semantic middleware layer for storing semantic data federation, and solving problems related to data integration and system interoperability in a heterogeneous system.

5.3.1 Discussion of Findings and Broader Significance

The findings of this study offer valuable insights that extend beyond the immediate development of a Semantic Multi-Hazard Early Warning System (MHEWS) for the Lejweleputswa district. By integrating indigenous knowledge with advanced sensor-based technologies, the study bridges critical gaps in the current understanding and practice of disaster risk management and early warning systems.

The incorporation of indigenous knowledge demonstrates that local expertise and traditional practices can enhance the precision and cultural acceptability of warning systems. This approach challenges the predominantly technology driven paradigms in MHEWS by emphasising the value of blending modern and traditional knowledge systems. Such an integrative framework could inspire other regions to adopt similar methodologies, especially in areas where indigenous communities maintain a strong connection to their environment. Additionally, the findings highlight the potential of wireless sensor networks in improving real-time data collection and analysis for disaster preparedness. Demonstrating the successful deployment of these technologies in a region with varied environmental and industrial challenges, the study provides a replicable model for implementing cost effective and scalable early warning systems. This innovation could be particularly transformative for low resource settings, where technological infrastructure is often limited.

The study advances the theoretical understanding of disaster risk management by proposing a multi-dimensional risk assessment model that incorporates environmental, social, and technological factors. The integration of indigenous knowledge within a semantic framework challenges existing disaster management theories that often marginalize non-technical perspectives. This contribution enriches the conceptual foundations of disaster risk reduction, advocating for a more inclusive and participatory approach. Furthermore, the use of semantic technologies in organizing and analysing heterogeneous data contributes to the growing field of disaster informatics**.** Demonstrating the utility of semantic middleware for managing complex datasets, this research provides a novel perspective on how data can be structured and interpreted to support decision-making in disaster risk scenarios. The findings also carry significant policy implications. Highlighting the importance of localised, real-time data in addressing pollution and disaster risks, the study advocates for policies that prioritise community engagement and the integration of indigenous knowledge in early warning systems. Policymakers and practitioners can draw from this study to design systems that are not only technologically sophisticated but also socially inclusive and environmentally relevant.

Ultimately, this research contributes to advancing the global discourse on disaster resilience and environmental sustainability. It demonstrates how region-specific solutions, informed by both local knowledge and modern technologies, can serve as scalable models for addressing complex environmental challenges. The insights gained from this study have the potential to influence the design and implementation of MHEWS in diverse contexts, fostering a more adaptive and resilient approach to disaster risk management worldwide.

5.4. Implementation Challenges

The implementation of containerizing a software application with Docker and Kubernetes offers many benefits, but it also presents some challenges that organizations need to address. The challenges include the learning curve, application decomposition, managing network communication, and cultural shift.

The learning curve associated with Docker and Kubernetes technologies is due to their complexity, and represents a significant learning curve for us to master these technologies. These technologies require a fundamental understanding of the principles of containerization. This learning curve potentially slowed down the implementation process as understanding of these technologies was required. This challenge has been addressed through training programmes that provide the opportunity to gain practical experience. This has helped shorten the learning curve and accelerate implementation by investing time and providing access to educational resources. The decomposition of a monolithic application into smaller container services is a crucial step in the containerization process. Identifying the appropriate boundaries of these containers and making the application to work in the new paradigm is a very complex task. It requires careful planning, architectural decisions, and precise execution.

Managing network communications for an application in a container environment involves a set of methods. To establish secure and efficient communication between containers and services, network components such as the ingress file and the service file must be configured correctly. Tasks such as configuring service files and ingress files, managing them, and ensuring the integrity of network protocols can be difficult. One of the most crucial challenges is the cultural shift or change. Adopting containerization often requires adopting new practices such as microservices. To address this challenge, developing a culture of collaboration and learning DevOps methodologies for successful implementation was necessary. Addressing the challenges of containerizing applications with Docker and Kubernetes requires a combination of education, technology adoption, system implementation practice and cultural transformation. By investing in training, automation and compliance controls, organizations can be able to overcome these challenges and achieve positive results in their containerization efforts.

5.5. Key Findings

Semantic Middleware development and application containerization with Docker and Kubernetes orchestration platforms combine two technology areas. Semantic Middleware is

intended to improve data processing, system interoperability, and containerization to efficiently deploy and scale MHEWS applications. This approach offers significant advantages and insights.

The use of semantic middleware has improved data integration by providing a common semantic framework to understand data sources in a standardized manner and improve data quality. Semantic middleware facilitates interoperability between different components and systems, ensuring that data and services can communicate and work together effectively, even if they come from different environments. Semantic middleware also allows the application to understand the context of the data, which is essential for making informed decisions. This is useful in MHEWS where accurate interpretation of data is essential. Finally, semantic middleware integrates knowledge-based reasoning and allows both adopted systems to infer knowledge based on existing knowledge, and this reasoning ability improves prediction accuracy and decision support.

Docker containers provide a consistent environment for applications across all platforms, from development to production. This consistency ensures that applications behave the same in every environment. Kubernetes organizes containers and optimizes resource usage by scaling and automatically loading applications as needed. This efficiency leads to cost savings and improved efficiency. Containerization makes it easier to deploy and scale applications. Kubernetes simplifies the deployment, management, and scaling to achieve high availability through redundancy. In summary, combining semantic middleware for data integration, and understanding of containerization using Docker and Kubernetes to effectively deploy and manage applications is a powerful approach to modernizing and improving software systems.

5.5.1. Implications of Findings for Policy, Practice, and Future Research

The findings of this study extend beyond the technical development of the Semantic MHEWS, providing actionable insights that can shape policies, influence practical applications, and guide future research in disaster risk management and early warning systems. The integration of indigenous knowledge and advanced technological tools, as demonstrated in this study, offers a compelling case for the formulation of inclusive and community focused policies in disaster risk reduction. Policymakers are encouraged to:

- Adopt frameworks that incorporate indigenous knowledge: Recognising and formalising the role of local knowledge in early warning systems can ensure that

disaster management strategies are culturally relevant and widely accepted by affected communities.

- Invest in decentralised data systems: The findings support the implementation of decentralised data collection and analysis systems, such as wireless sensor networks, which are adaptable to diverse environmental contexts.
- Explore blockchain technology for data security and transparency: The potential of blockchain to ensure the integrity and traceability of critical data could be transformative for disaster risk management, fostering trust among stakeholders and ensuring data authenticity.

Practical Implications

For practitioners in the field of disaster risk management, the study's findings highlight several practical considerations:

- Enhanced real-time decision-making: The integration of semantic middleware and real-time sensor data provides a scalable model for rapidly interpreting environmental changes and issuing timely warnings. This model can serve as a blueprint for improving decision making processes in other regions.
- Localised and adaptable systems: The region-specific focus of this study emphasises the need for early warning systems that can be tailored to local environmental and socio-economic conditions. Practitioners should prioritise systems that are flexible and adaptive to regional variations.
- Community engagement and training: Ensuring that local communities are actively involved in both the development and implementation of MHEWS is essential for long-term success. Training programs should be designed to build capacity and enhance the community's ability to respond effectively to warnings.

Implications for Future Research

The findings open several avenues for further exploration in the field of MHEWS and disaster informatics:

- Multi-dimensional risk modelling: Building on the framework proposed in this study, future research could focus on developing advanced multi-hazard models that integrate

socio-economic, environmental, and technological dimensions to predict and mitigate risks more effectively.

- Scalability and adaptability of semantic frameworks: Investigating how semantic technologies can be adapted to other regions with varying environmental and cultural contexts would contribute to the broader applicability of MHEWS.
- Impact evaluation: Longitudinal studies assessing the impact of integrating indigenous knowledge and advanced technologies into early warning systems could provide critical insights into their effectiveness and areas for improvement.

5.6. Recommendations and Future Scope

Incorporating blockchain technology into the development of semantic middleware for South Africa MHEWS is a fundamental approach that can enhance the system's efficiency, security, and transparency. This study hereby provides some recommendations for integrating blockchain technology into existing early warning systems. This will help to address critical issues related to data authentication, data integrity, secure data exchange and decentralized information management. The use of decentralized systems can effectively prevent unauthorized manipulation and changes to critical information and create a solid foundation for trust and reliability. Additionally, the study recommends the development of blockchain-based smart contracts to facilitate secure data exchange and collaboration between different users involved in disaster management. These smart contracts can streamline the process of sharing critical data during natural disasters by ensuring those involved have timely access to verifiable information. In addition, the study also proposes the implementation of a decentralized network of IoT sensors that record data, and transmit it directly to blockchain technology for reducing the risk of data manipulation, and providing reliable real-time information.

To ensure the successful integration of blockchain technology, interoperability with MHEWS is important. This interoperability ensures a coherent and efficient information ecosystem where data can flow seamlessly between systems, maximizing the utility of a blockchain-based early warning system. The adaptation of blockchain technology in the South African MHEWS can improve disaster preparedness and response capabilities. The above-mentioned recommendations represent a comprehensive approach to blockchain integration, and emphasize data security and interoperability to ensure long-term stability and suitability. The semantic middleware developed for this study serves as a foundation for future advancements

in disaster risk reduction, and mitigation of pollution on communities surrounded by mining companies. Such development is crucial as we move forward collaboration, innovation, and commitment towards mitigating pollution, and safeguarding communities, which remain our core efforts.

5.7 Contribution to the Literature and Comparative Analysis

This research contributes to the growing body of knowledge in Multi-Hazard Early Warning Systems (MHEWS) and disaster risk management by addressing specific gaps in the existing literature:

1. Integration of Indigenous Knowledge with Advanced Technologies: While previous studies have largely focused on technology driven early warning systems, the integration of indigenous knowledge in this study highlights the value of community-specific insights. This approach not only enhances the cultural relevance of MHEWS but also bridges the gap between traditional practices and modern technological frameworks.
2. Semantic Middleware for Multi-Hazard Systems: The deployment of semantic middleware in this study provides a novel framework for organising and interpreting heterogeneous data. Unlike existing systems that often rely on monolithic architectures, the semantic middleware ensures seamless data integration and enhances the system's adaptability to dynamic environmental conditions.
3. Focus on Regional Context: Much of the current research in MHEWS overlooks the importance of tailoring systems to specific regional contexts. This study addresses this gap by developing and implementing a region-specific MHEWS for the Lejweleputswa district, providing a scalable model for similar regions.

Filling these gaps, the research establishes a foundation for creating more inclusive, efficient, and adaptable early warning systems, contributing to the broader discourse on sustainable disaster risk management.

Comparative Analysis with Existing Frameworks

A comparison with existing early warning systems and frameworks highlights the unique contributions of this study:

1. Traditional Early Warning Systems: Conventional systems such as those used in meteorological forecasting often rely solely on real-time sensor data and deterministic models. While effective for single hazards, these systems are limited in their ability to handle the complexity of multi-hazard scenarios. This study's MHEWS leverages semantic technologies to manage multi-hazard datasets, enabling comprehensive risk assessments and more nuanced alerts.

2. Community-Based Disaster Risk Reduction (CBDRR): Existing community-based frameworks emphasize local engagement but often lack the technological integration needed for large scale or dynamic hazards. This study bridges that gap by incorporating indigenous knowledge into a technologically robust framework, ensuring that local insights are complemented by advanced data analysis tools.

3. Blockchain-Integrated Systems: Emerging studies have explored the use of blockchain in data security for early warning systems. While this study identifies blockchain as a potential enhancement, its current focus on semantic middleware prioritises operational efficiency and scalability, offering a pragmatic solution for regions with limited resources.

4. UNDRR (United Nations Office for Disaster Risk Reduction) Guidelines: The study aligns with global frameworks such as the Sendai Framework for Disaster Risk Reduction as mentioned in Chapter2. This alignment ensures that the study's contributions are both globally relevant and locally applicable.

Combining these unique features, this study situates itself at the intersection of technology, community engagement, and environmental sustainability. It provides a replicable model for integrating advanced technologies with localised knowledge, offering valuable insights for both academic research and practical implementation in disaster prone regions.

5.8 Chapter Summary

Chapter 5 serves as the culmination of this study, bringing together the findings from previous chapters and synthesising the core results of the research on developing and implementing a Semantic Multi-Hazard Early Warning System (MHEWS). The research aimed to improve the ability of communities, specifically those affected by pollution in the Lejweleputswa district, South Africa, to manage and respond to hazards through the integration of semantic middleware technologies. The chapter is structured to offer a thorough summary of the study's

objectives, methodologies, findings, challenges, implications, and contributions, while also offering recommendations for future research and development.

In conclusion, this research presents a comprehensive framework for developing a Semantic Multi-Hazard Early Warning System that integrates semantic technologies, containerization, and community knowledge to improve disaster risk management. Addressing the technical, social, and environmental dimensions of early warning systems, the study contributes to the ongoing efforts to create more resilient communities. The findings not only advance the theoretical understanding of disaster informatics but also offer practical solutions for policymakers, practitioners, and researchers in the field of disaster risk reduction. The insights gained from this study have the potential to influence the design and implementation of early warning systems in diverse regions, contributing to global efforts in building adaptive and resilient disaster management frameworks.

REFERENCES

Akanbi, A. K., & Masinde, M. (2015, December 7). Towards semantic integration of heterogeneous sensor data with indigenous knowledge for drought forecasting. *Proceedings of the 12th Middleware Doctoral Symposium, Middleware Doctoral Symposium 2015 - Co-Located with the ACM/IFIP/USENIX 15th International Middleware Conference*. https://doi.org/10.1145/2843966.2843968

Akanbi, A. K., & Masinde, M. (2018). *IKON-OWL: Using Ontologies for Knowledge Representation of Local Indigenous Knowledge on Drought Completed Research*.

Akanbi, A., & Masinde, M. (2020). A distributed stream processing middleware framework for real-time analysis of heterogeneous data on big data platform: Case of environmental monitoring. *Sensors (Switzerland)*, *20*(11), 1–25. https://doi.org/10.3390/s20113166

Al Jawarneh, I. M., Bellavista, P., Bosi, F., Foschini, L., Martuscelli, G., Montanari, R., & Palopoli, A. (2019). Container Orchestration Engines: A Thorough Functional and Performance Comparison. *IEEE International Conference on Communications*, *2019-May*. https://doi.org/10.1109/ICC.2019.8762053

Alonso, L., Barbarán, J., Chen, J., Díaz, M., Llopis, L., & Rubio, B. (2018). Middleware and communication technologies for structural health monitoring of critical infrastructures: A survey. *Computer Standards and Interfaces*, *56*, 83–100. https://doi.org/10.1016/j.csi.2017.09.007

Aslam, A., & Curry, E. (2021). A Survey on Object Detection for the Internet of Multimedia Things (IoMT) using Deep Learning and Event-based Middleware: Approaches, Challenges, and Future Directions. *Image and Vision Computing*, *106*. https://doi.org/10.1016/j.imavis.2020.104095

Badica, C., Vidaković, M., Ilie, S., Ivanović, M., & Vidaković, J. (2019). Role of Agent Middleware in Teaching Distributed Systems and Agent Technologies. *Journal of Computing and Information Technology*, *27*(1). https://doi.org/10.20532/cit.2019.1004464

Bamhdi, A. (2021). Requirements capture and comparative analysis of open source versus proprietary service-oriented architecture. *Computer Standards and Interfaces*, *74*. https://doi.org/10.1016/j.csi.2020.103468

Coker, A., Sangodoyin, A., Sridhar, M., Booth, C., Olomolaiye, P., & Hammond, F. (2009). Medical waste management in Ibadan, Nigeria: Obstacles and prospects. *Waste Management*, *29*(2), 804–811. https://doi.org/10.1016/j.wasman.2008.06.040

Census, 2011. Statistical release (Revised). [Online] Available at: http://www.statssa.gov.za/

DWAF. (2001). *DEPARTMENT OF WATER AFFAIRS AND FORESTRY CHIEF DIRECTORATE : WATER SERVICES "FREE BASIC WATER" Implementation Strategy Document Version 1*.

Emmerich, W. (1997). *From 1996 to 1997 he was a Lecturer*.

Espí-Beltrán, J. V., Gilart-Iglesias, V., & Ruiz-Fernández, D. (2017). Enabling distributed manufacturing resources through SOA: The REST approach. *Robotics and Computer-Integrated Manufacturing*, *46*, 156–165. https://doi.org/10.1016/j.rcim.2016.09.007

Farsi, M., Badawy, M., Abdelmoneim, N., Arafat Ali, H., & Abdulazeem, Y. (2019). QoS-Aware Framework for Performance Enhancement of SOA in Enterprise IT Environments. *IEEE Access*, *7*, 107699–107715. https://doi.org/10.1109/ACCESS.2019.2932683

Fashola, M. O., Ngole-Jeme, V. M., & Babalola, O. O. (2016). Heavy metal pollution from gold mines: Environmental effects and bacterial strategies for resistance. In *International Journal of Environmental Research and Public Health* (Vol. 13, Issue 11). MDPI AG. https://doi.org/10.3390/ijerph13111047

Feris, L., & Kotze, L. (2015). The regulation of acid mine drainage in South Africa: law and governance perspectives. *Potchefstroom Electronic Law Journal/Potchefstroom Elektroniese Regsblad*, *17*(5), 2104. https://doi.org/10.4314/pelj.v17i5.07

Gaglio, S., Re, G. Lo, Martorella, G., & Peri, D. (2014). A lightweight middleware platform for distributed computing on wireless sensor networks. *Procedia Computer Science*, *32*, 908–913. https://doi.org/10.1016/j.procs.2014.05.510

Gbangou, T., VAN SLOBBE, E., Ludwig, F., Kranjac-Berisavljevic, G., & Paparrizos, S. (2021). Harnessing local forecasting knowledge on weather and climate in Ghana: Documentation, skills, and integration with scientific forecasting knowledge. *Weather, Climate, and Society*, *13*(1). https://doi.org/10.1175/WCAS-D-20-0012.1

Goundar, S. (2012). *Research Methodology and Research Method Methods Commonly Used By Researchers*.

Gwaze, P., & Mashele, S. H. (2018). South African Air Quality Information System (SAAQIS) mobile application tool: Bringing real time state of air quality to South Africans. In *Clean Air Journal* (Vol. 28, Issue 1, pp. 3–4). National Association of Clean Air. https://doi.org/10.17159/2410-972X/2018/v28n1a1

Hopping, K. A., Yangzong, C., & Klein, J. A. (2016). Local knowledge production, transmission, and the importance of village leaders in a network of Tibetan pastoralists coping with environmental change. *Ecology and Society*, *21*(1). https://doi.org/10.5751/ES-08009-210125

Igwenagu, C. (2016). Fundamentals of Research Methodology and Data Collection. *LAP Lambert Academic Publishing*, *May*.

Jiang, L., Yue, P., Kuhn, W., Zhang, C., Yu, C., & Guo, X. (2018). Advancing interoperability of geospatial data provenance on the web: Gap analysis and strategies. *Computers and Geosciences*, *117*. https://doi.org/10.1016/j.cageo.2018.05.001

Kubernetes. Available online: https://kubernetes.io (accessed on 6 January 2019).

Justin Dhas, Y., & Jeyanthi, P. (2019). A review on internet of things protocol and service-oriented middleware. *Proceedings of the 2019 IEEE International Conference on Communication and Signal Processing, ICCSP 2019*. https://doi.org/10.1109/ICCSP.2019.8698088

Luther, J., Hainsworth, A., Tang, X., Harding, J., Torres, J., & Fanchiotti, M. (2017). World Meteorological Organization (WMO)—Concerted International Efforts for Advancing Multi-hazard Early Warning Systems. In *Advancing Culture of Living with Landslides*. https://doi.org/10.1007/978-3-319-59469-9_9

LEJWELEPUTSWA DISTRICT MUNICIPALITY. Profile and analysis. District development model. (2020). Available at https://www.cogta.gov.za/ddm/wpcontent/uploads/2020/11/Lejweleputswa-DM-October-2020.pdf. Accessed 14 May 2021.

LEJWELEPUTSWA DISTRICT MUNICIPALITY. DISTRICT DEVELOPMENT MODEL DRAFT 2. (2021). Available at https://www.tswelopele.gov.za/index.php/notices/publicknotices?download=997:ldm-draft-district-development-model-25-aug-2021. Accessed 10 July 2021.

Maneo Ntseliseng Ramahlosi, Yolo Madani and Adeyinka Akanbi. (2023, October). A Blockchain-based Model for Securing Data Pipeline in a Heterogeneous Information System. Published Online by the SAICSIT 2023 Organising Committee Potchefstroom: South African Institute of Computer Scientists & Information Technologists ISSN:2959-8877 (electronic) (p. 167). ACM.

Malucelli, A., Palzer, D., & Oliveira, E. (2006). Ontology-based Services to help solving the heterogeneity problem in e-commerce negotiations. *Electronic Commerce Research and Applications*, *5*(1), 29–43. https://doi.org/10.1016/j.elerap.2005.08.002

Masinde, M., & Bagula, A. (2011). ITIKI: bridge between African indigenous knowledge and modern science of drought prediction. *Knowledge Management for Development Journal*, *7*(3). https://doi.org/10.1080/19474199.2012.683444

Masinde, M., & Thothela, P. N. (2019). ITIKI Plus: A Mobile Based Application for Integrating Indigenous Knowledge and Scientific Agro-Climate Decision Support for Africa's Small-Scale Farmers. *2019 IEEE 2nd International Conference on Information and Computer Technologies, ICICT 2019*. https://doi.org/10.1109/INFOCT.2019.8711059

Mayowa Morakinyo, O., Stephen Adebowale, A., Ingrid Mokgobu, M., & Stanley Mukhola, M. (2017). Health risk of inhalation exposure to sub-10 µm particulate matter and gaseous pollutants in an urban-industrial area in South Africa: an ecological study. *BMJ Open*, *7*, 13941. https://doi.org/10.1136/bmjopen-2016

Mbele, M. J., Masinde, M., & Ngowi, K. I. (2017). *Development of an Adaptive Environmental Monitoring System for Lejweleputswa District : A Participatory Approach through Fuzzy Cognitive Maps Masters Dissertation by Mpho Josephine Mbele Supervised by Prof . Muthoni Masinde and Itumeleng Kgololo-Ngowi Thi*.

Mbele, M., Masinde, M., & Kgololo-Ngowi, I. (2016). Adaptive environmental management system for lejweleputswa district: A participatory approach through fuzzy cognitive maps. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, *171*(1), 225–226. https://doi.org/10.1007/978-3-319-43696-8

Mesmoudi, Y., Lamnaour, M., El Khamlichi, Y., Tahiri, A., Touhafi, A., & Braeken, A. (2020). A Middleware based on Service Oriented Architecture for Heterogeneity Issues within the Internet of Things (MSOAH-IoT). *Journal of King Saud University - Computer and Information Sciences*, *32*(10), 1108–1116. https://doi.org/10.1016/j.jksuci.2018.11.011

Microsoft, 2017. Microsoft Azure Cloud Computing Platform & Services. Microsoft Website [WWW Document]. https://azure.microsoft.com/en-us/. (Accessed 24 July 2017).

Namiot, D., & Sneps-Sneppe, M. (2014). On Micro-services Architecture. *International Journal of Open Information Technologies*, *2*(9).

Pearson, L., & Pelling, M. (2015). The UN Sendai Framework for Disaster Risk Reduction 2015–2030: Negotiation Process and Prospects for Science and Practice. *Journal of Extreme Events*, *02*(01). https://doi.org/10.1142/s2345737615710013

Petcu, D. (2021). SERVICE DEPLOYMENT CHALLENGES IN CLOUD-TO-EDGE CONTINUUM. *Scalable Computing*, *22*(3). https://doi.org/10.12694/SCPE.V22I3.1941

Pham, N. M. (2020). A proposal for a cloud-based microservice architecture for the Skolrutiner system. . . *Uppsala Universitet*, 1–28.

Pradeep, P., Krishnamoorthy, S., & Vasilakos, A. V. (2021). A holistic approach to a context aware IoT ecosystem with Adaptive Ubiquitous Middleware. *Pervasive and Mobile Computing*, *72*. https://doi.org/10.1016/j.pmcj.2021.101342

Rashid, A., & Chaturvedi, A. (2019). Cloud Computing Characteristics and Services A Brief Review. *International Journal of Computer Sciences and Engineering*, *7*(2). https://doi.org/10.26438/ijcse/v7i2.421426

Maneo Ntseliseng Ramahlosi, Yolo Madani and Adeyinka Akanbi. (2023, October). A Blockchain-based Model for Securing Data Pipeline in a Heterogeneous Information System. Published Online by the SAICSIT 2023 Organising Committee Potchefstroom: South African Institute of Computer Scientists & Information Technologists ISSN:2959-8877 (electronic) (p. 167). ACM.

Reis, J. Z., & Gonçalves, R. F. (2018). The role of internet of services (IoS) on industry 4.0 through the service-oriented architecture (SOA). *IFIP Advances in Information and Communication Technology*, *536*. https://doi.org/10.1007/978-3-319-99707-0_3

Rimayi, C., Odusanya, D., Weiss, J. M., de Boer, J., & Chimuka, L. (2018). Contaminants of emerging concern in the Hartbeespoort Dam catchment and the uMngeni River estuary 2016 pollution incident, South Africa. *Science of the Total Environment*, *627*, 1008–1017. https://doi.org/10.1016/j.scitotenv.2018.01.263

Rivett, U., Champanis, M., & Wilson-Jones, T. (2013). Monitoring drinking water quality in South Africa: Designing information systems for local needs. *Water SA*, *39*(3), 409–414. https://doi.org/10.4314/wsa.v39i3.10

Rodrigues, C. M., Pereira, L., Rocha, A., & Dias, Z. (2019). Image Semantic Representation for Event Understanding. *2019 IEEE International Workshop on Information Forensics and Security, WIFS 2019*. https://doi.org/10.1109/WIFS47025.2019.9035102

Rumiński, D., & Walczak, K. (2020). Large-scale distributed semantic augmented reality services – A performance evaluation. *Graphical Models*, *107*. https://doi.org/10.1016/j.gmod.2019.101027

Schneider, B., Hoffmann, G., & Reicherter, K. (2016). Scenario-based tsunami risk assessment using a static flooding approach and high-resolution digital elevation data: An example from Muscat in Oman. *Global and Planetary Change*, *139*. https://doi.org/10.1016/j.gloplacha.2016.02.005

Srivastava, P., & Khan, R. (2018). A Review Paper on Cloud Computing. *International Journal of Advanced Research in Computer Science and Software Engineering*, *8*(6), 17. https://doi.org/10.23956/ijarcsse.v8i6.711

Sun, L., Li, Y., & Memon, R. A. (2017). An open IoT framework based on microservices architecture. *China Communications*, *14*(2). https://doi.org/10.1109/CC.2017.7868163

Tengö, M., Austin, B. J., Danielsen, F., & Fernández-Llamazares, Á. (2021). Creating Synergies between Citizen Science and Indigenous and Local Knowledge. *BioScience*, *71*(5), 503–518. https://doi.org/10.1093/biosci/biab023

Wortmann, F., & Flüchter, K. (2015). Internet of Things: Technology and Value Added. In *Business and Information Systems Engineering* (Vol. 57, Issue 3, pp. 221–224). Gabler Verlag. https://doi.org/10.1007/s12599-015-0383-3

World Meteorological Organization (WMO) (2012) Institutional Partnerships in Multi-Hazard Early Warning Systems: A Compilation of Seven National Good Practices and Guiding Principles. Edited by M. Golnaraghi. Heidelberg Dordrecht London New York: Springer.

Yoo, S. K., & Kim, B. Y. (2018). A decision-making model for adopting a cloud computing system. *Sustainability (Switzerland)*, *10*(8). https://doi.org/10.3390/su10082952

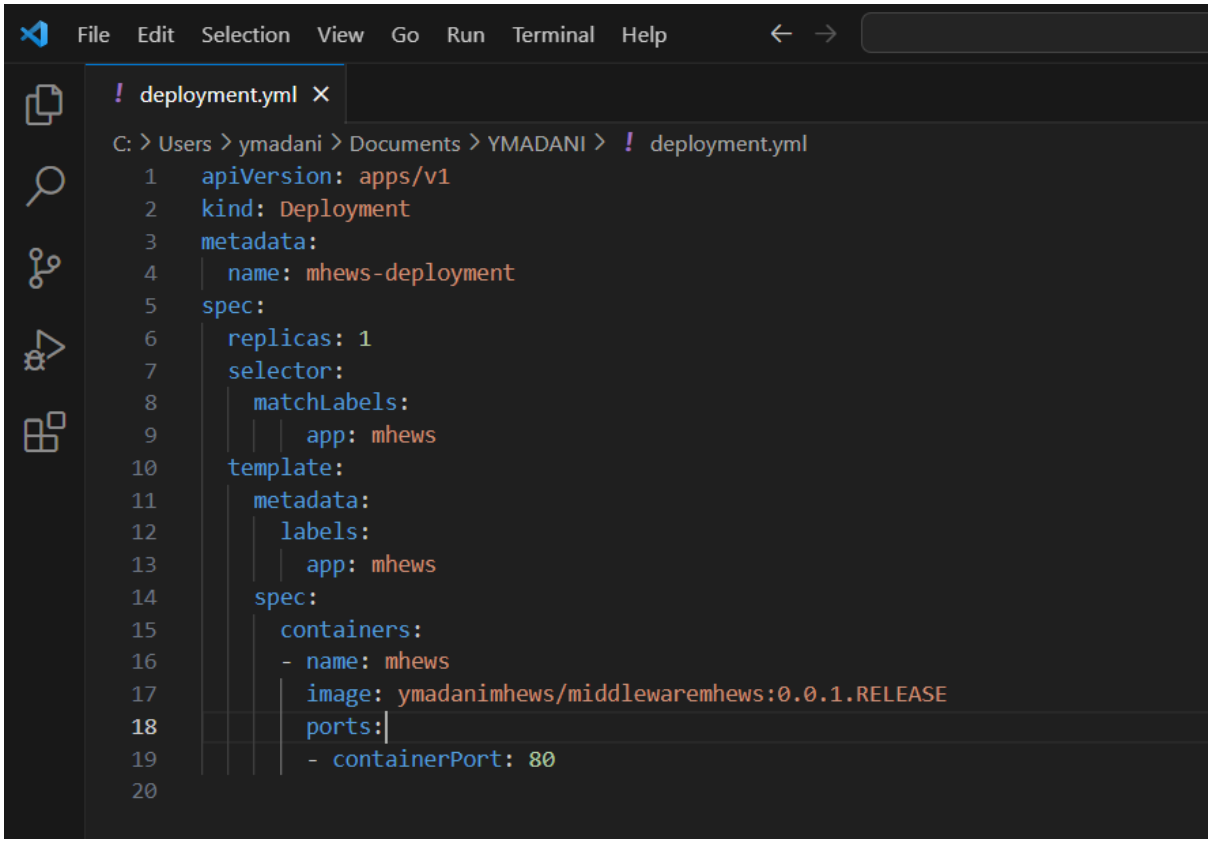
Younan, M., Houssein, E. H., Elhoseny, M., & Ali, A. A. (2020). Challenges and recommended technologies for the industrial internet of things: A comprehensive review. *Measurement: Journal of the International Measurement Confederation*, *151*. https://doi.org/10.1016/j.measurement.2019.107198

Zgheib, R., Conchon, E., & Bastide, R. (2019). Semantic middleware architectures for IoT healthcare applications. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 11369 LNCS* (pp. 263–294). Springer Verlag. https://doi.org/10.1007/978-3-030-10752-9_11

Zhang, J., Ma, M., Wang, P., & Sun, X. dong. (2021). Middleware for the Internet of Things: A survey on requirements, enabling technologies, and solutions. In *Journal of Systems Architecture* (Vol. 117). Elsevier B.V. https://doi.org/10.1016/j.sysarc.2021.102098

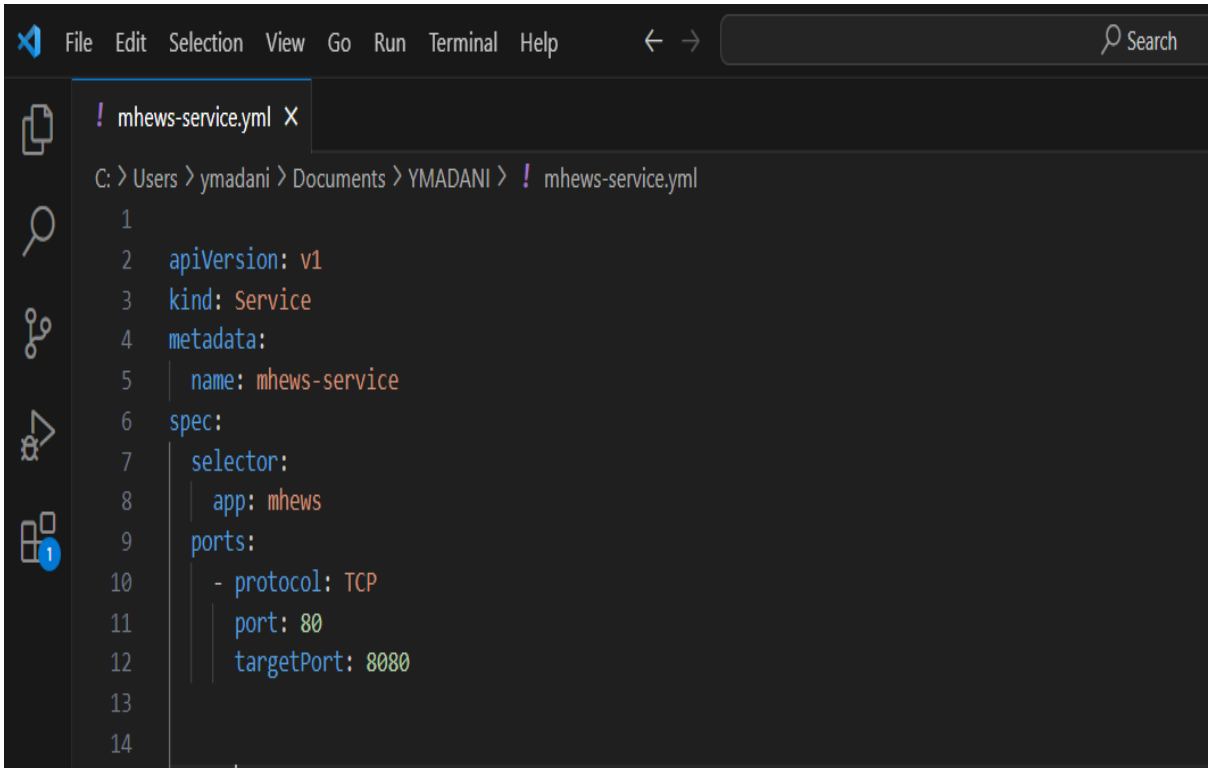
APPENDICES

APPENDIX A



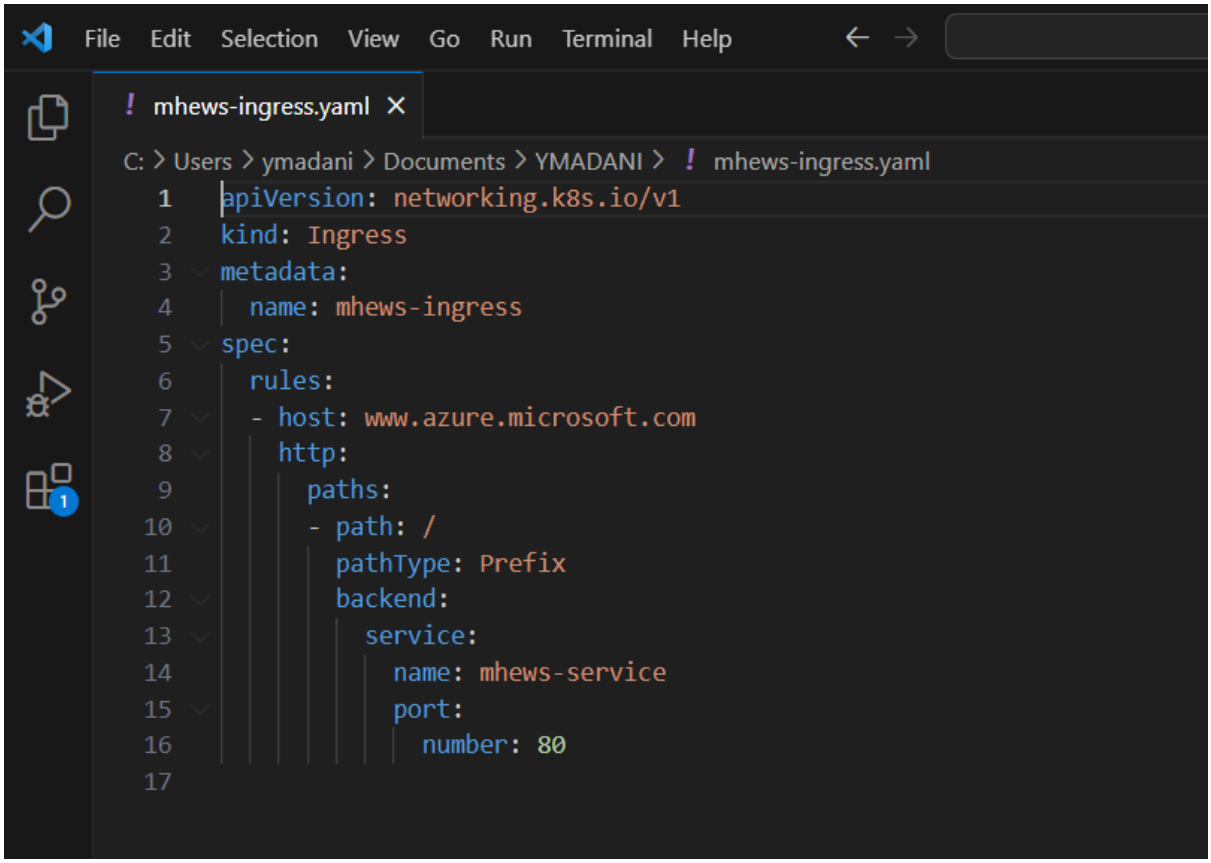
Using manifest file, deployment.yml for Kubernetes Deployment (Source Author).

APPENDIX B



Defining a Kubernetes Service Resource (Source Author).

APPENDIX C



Using mhwes-ingress.yml for Http and Https routes (Source Author).

APPENDIX D

Thank You for taking the time to complete this questionnaire. Your input is greatly appreciated in understanding and addressing the concerns related to air pollution caused by mining activities in your community.

RESEARCH QUESTIONNNAIRE

SECTION A – Demographic Information

1. Please specify your gender
   ☐ Male
   ☐ Female
   ☐ Prefer not to say
2. Please select your age group.
   ☐ 18 – 24 Years
   ☐ 25 – 39 Years
   ☐ 40 – 60 Years
   ☐ 61 Plus
3. Occupation?
   ☐ Farmer
   ☐ Small Business Owner
   ☐ Student
   ☐ Not Working
   ☐ Miner
4. If Mine worker, for how long have you been working at the mines?
   ☐ 0 – 5 Years
   ☐ 6 – 10 Years
   ☐ 11 – 20 Years
   ☐ 21 Plus
5. Yeas of residence in the Area?
   ☐ 0 – 5 Years
   ☐ 6 – 10 Years
   ☐ 11 – 20 Years
   ☐ 21 Plus

SECTION B – Impact of Mining Activities.

Please rate the following statements based on your observations and experiences:

6. Mining activities have provided economic benefits to our community.

☐ Strongly Agree
☐ Agree
☐ Neutral
☐ Disagree
☐ Strongly Disagree

7. Mining activities have affected water quality in our community.
   ☐ Strongly Agree
   ☐ Agree
   ☐ Neutral
   ☐ Disagree
   ☐ Strongly Disagree
8. Mining activities have caused changes in local vegetation and soil quality.
   ☐ Strongly Agree
   ☐ Agree
   ☐ Neutral
   ☐ Disagree
   ☐ Strongly Disagree
9. Mining activities have led to displacement or disruption of local communities.
   ☐ Strongly Agree
   ☐ Agree
   ☐ Neutral
   ☐ Disagree
   ☐ Strongly Disagree

SECTION C – Air Pollution.

10. Have you noticed changes in air quality since mining activities began in the area?
    ☐ Yes
    ☐ No
    ☐ Not sure

11. If yes, please describe any changes you have observed:

    [_____]

SECTION D – Pollution Early Warning System.

12. Are you aware of any early warning systems or initiatives related to environmental pollution in your community?

☐ Yes

☐ No

13. If yes, please describe how effective you believe these systems are:

[_____]

SECTION E – Request for additional information.

14. Would you be interested in receiving more information about environmental impacts and early warning systems related to mining activities?

☐ Yes

☐ No

15. Any additional comments or information you would like to share:

[_____]

T H A N K Y O U FOR YOR PARCITIPATION! YOUR INPUT IS VALUABLE TO OUR RESEARCH

--- End ---