University of Prince Muqrin bin Abdul-Aziz
College of Computer and Cyber Sciences
Department of Software Engineering

SE342 - Software Architecture and Design

Course Project – Semester II (Spring 2023)

Smart Traffic Control System (STCS)

Team Members:

Malak Sabouni 3910004

Samah Shamma 4010403

Salwa Shamma 4010405

Sana Shamma 4010404

Instructor:

Dr. Khalid Khankan

29/5/2023

Table of Contents

Abstract **3**

CHAPTER 1: INTRODUCTION **4**

1.1 Smart Traffic Control System (STCS) Overview 4

1.2 Selected Scenario: IoT Traffic Control System for City Government 4

CHAPTER 2: SOFTWARE REQUIREMENT **5**

2.1 List of Terms and Definitions in STCS 5

2.2 Functional Requirement 5

2.3 Non- Functional Requirements 6

CHAPTER 3: SYSTEM DESIGN **6**

3.1 System Architecture Diagram 6

3.2 Component Diagram 7

3.3 Class Diagram 8

CHAPTER 4: ANALYSES AND DISCUSSION **8**

4.1 Design Principles and Decisions Applied in Architecture 8

4.2 Architectural Analysis 9

4.2.1 Patterns in System Design 9

4.2.2 Middleware architectures and frameworks 11

4.2.3 Cloud containerization and visualization in STCS 12

4.3 Mapping System Design to Quality Attributes 12

4.4 How System Architecture Adapt to Handle Changes 13

4.5 Architecture Evaluation 14

CHAPTER 5: CONCLUSION **14**

5.1 Conclusion 14

5.2 References 15

APPENDICES **16**

1. Use Case Diagram 15

2. Diagrams Link 16

Abstract

This project report describes the design and architecture of a Smart Traffic Control System (STCS). It includes details about the main system requirements, both functional and non-functional, as well as a thorough discussion of the system analysis and design. This analysis covers system components and system architecture.

Additionally, the report includes an analysis and discussion chapter that covers the design principles used, architectural patterns chosen, and decisions made to ensure quality attributes were met. This section also covers the middleware and framework used and concludes with an evaluation of the software architecture, providing justifications for design choices.

CHAPTER 1: INTRODUCTION

1.1 Smart Traffic Control System (STCS) Overview

The congestion of traffic in cities around the world has become a growing problem as cities grow. In order to contain this issue, this project addresses a project where the city government launched a project to develop an Internet of Things (IoT) system that collects data on traffic volume, vehicle speed, and congestion in streets. Smart Traffic Control System (STCS) is the project name. The goal is to optimize traffic movements and reduce congestion in the city. Nevertheless, this solution must not compromise citizens' privacy of data, additionally, it must be scalable and modifiable to accommodate future needs and evolving trends. Furthermore, security against cyber-attacks is a very crucial factor to consider in the proposed solution.

This report describes the approach taken to design and implement the STCS System. It covers a selection of specifications of the system, including functional and non-functional requirements, the design and architecture of the system components, and an extensive analysis of the design principles and patterns selected. This is to guarantee a well-adaptable solution with all required quality attributes.

1.2 Selected Scenario: IoT Traffic Control System for City Government

A city government aims to improve traffic flow using IoT, so the city wants to develop an IoT system that can collect data on traffic volume, vehicle speed, and congestion, and use this data to optimize traffic signals and reduce congestion. The wanted solution should address the given functional requirements while ensuring privacy, scalability, and modifiability. In addition, the government wants to secure its IT infrastructure, including the required solution and other existing solutions against cyber-attacks (threat detection and response). Sensitive data and reports are confidential and must be protected from unauthorized access (access control).

CHAPTER 2: SOFTWARE REQUIREMENT

2.1 List of Terms and Definitions in STCS

The table describes the context and meaning of terms used in this report.

Term	Definition
IoT sensors	Internet of Things is a network of devices (such as sensors or cameras) that will collect and transmit data over the internet.
Traffic volume	The number of vehicles that pass through a given area over a period of time, usually measured in vehicles per hour.
key intersections	The main intersections (locations) that need to be monitored and managed to improve traffic flow in the city.
Congestion	The situation of having too many vehicles on a road or at an intersection, leading to slower traffic flow or even standstill.

Table 2.1

2.2 Functional Requirement

ID	**Functional Requirement Description**
FR1	The system shall be able to collect data on traffic volume from IoT sensors placed at key intersections.
FR2	The system shall be able to collect data on vehicle speed from IoT sensors placed at key intersections.
FR3	The system shall be able to collect data on congestion levels from IoT sensors placed at key intersections.
FR4	The system shall be able to analyze the collected data to determine optimal traffic signal patterns (the timing and sequencing of traffic signals at an intersection).
FR5	The system shall be able to update traffic signal patterns in real-time based on the analyzed data.
FR6	The admin shall be able to do emergency responding actions.
FR7	The system shall be able to provide reports on traffic flow and congestion for city officials.
FR8	The admin shall be able to display map

Table 2.2

2.3 Non- Functional Requirements

ID	Requirement	Description
NF1	Privacy	The system shall be able to handle user authentication with a false positive rate of no more than 2% to protect sensitive data such as traffic volume, vehicle speed, and congestion as well as the reports from unauthorized access.
NF2	Scalability	The system shall be able to handle at least 1000 data per second to accommodate the amounts of data as the city's traffic conditions change and expand.
NF3	Modifiability	The system should be designed to allow for modifications of its components within 3 months, with no more than a 10% impact on existing infrastructure.
NF4	Security	The system shall be able to detect at least 95% of all security threats within 10 minutes and respond to them within 3 minutes.

Table 2.3

* For further reference, you can see the use case diagram provided in the appendix section.

CHAPTER 3: SYSTEM DESIGN

3.1 System Architecture Diagram

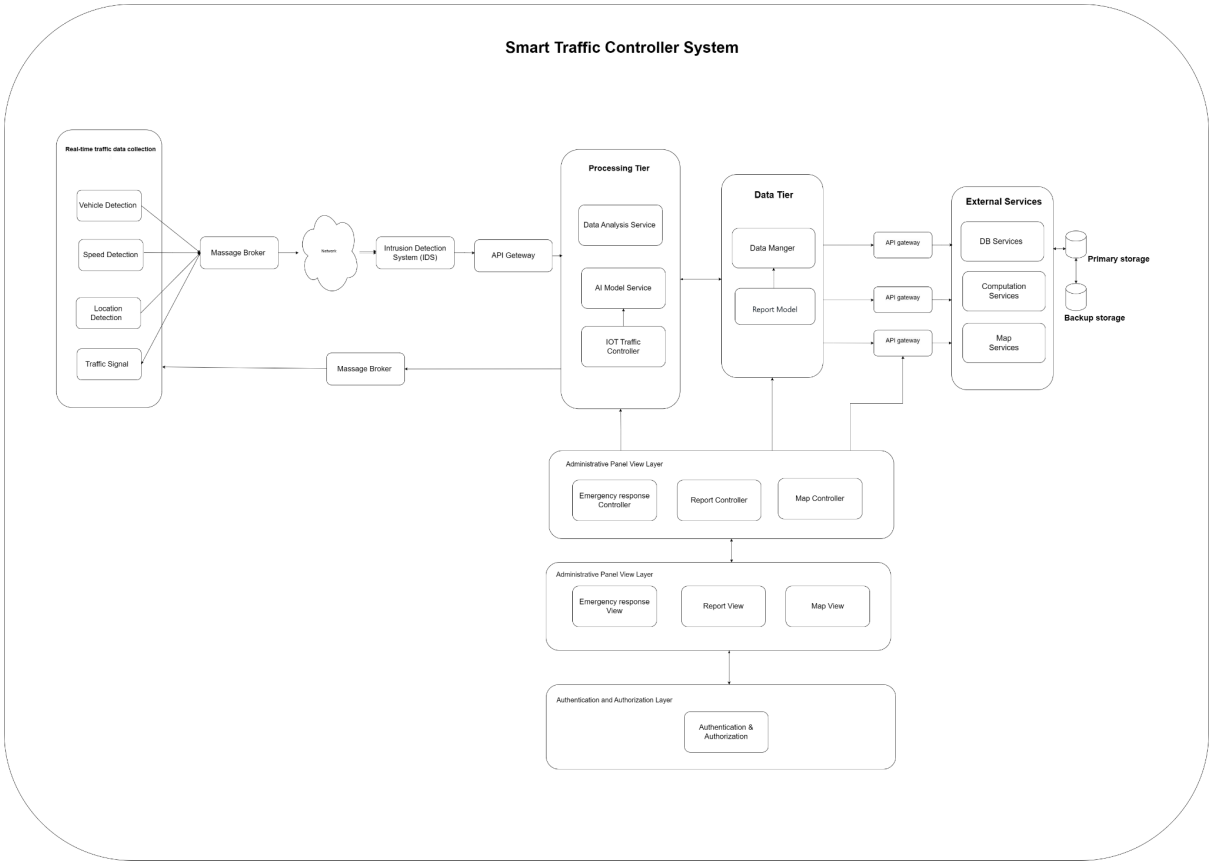

Figure 3.1.1- Overall Architecture

3.2 Component Diagram

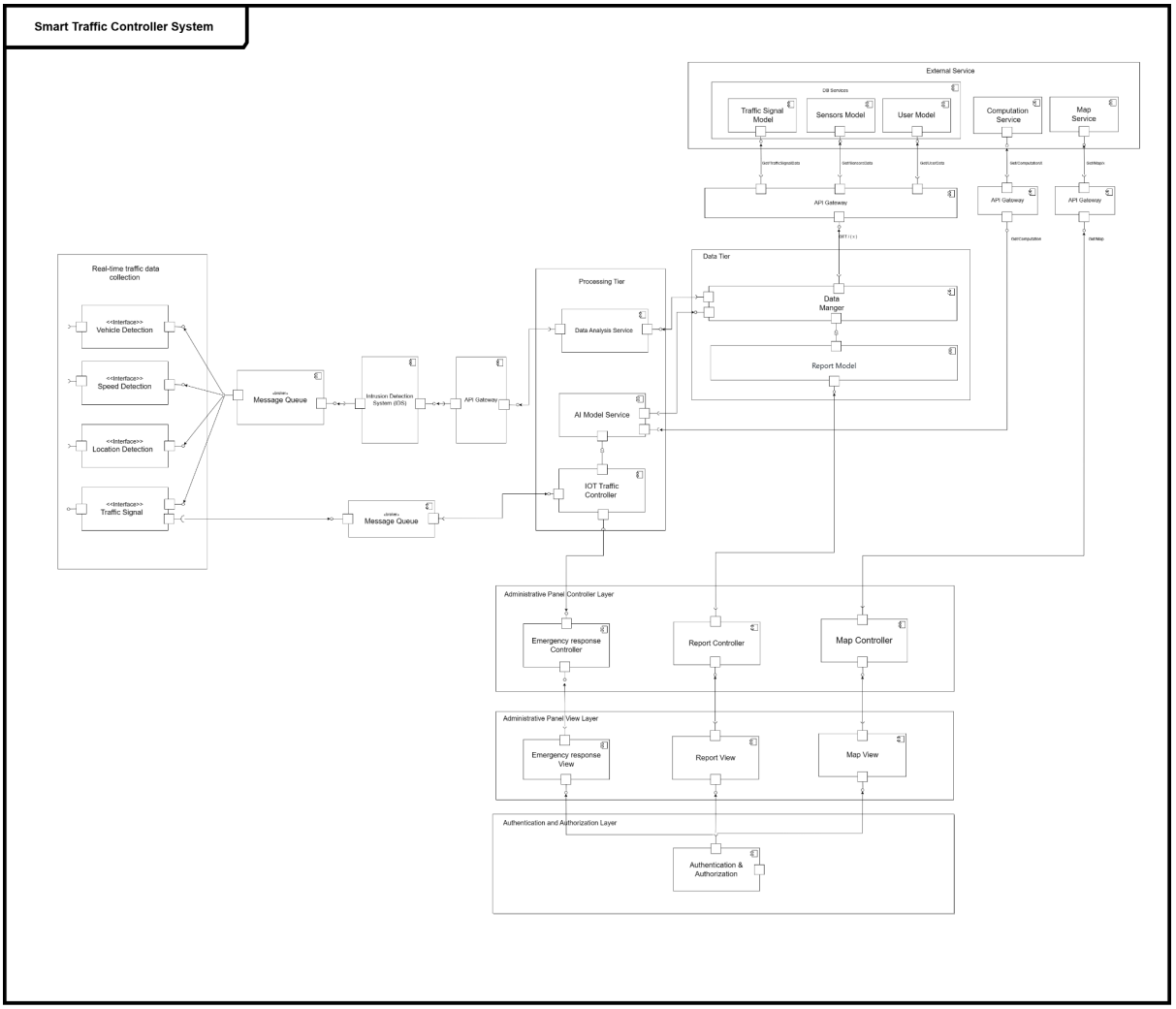

Figure 3.2.1 - Component Diagram

3.3 Class Diagram

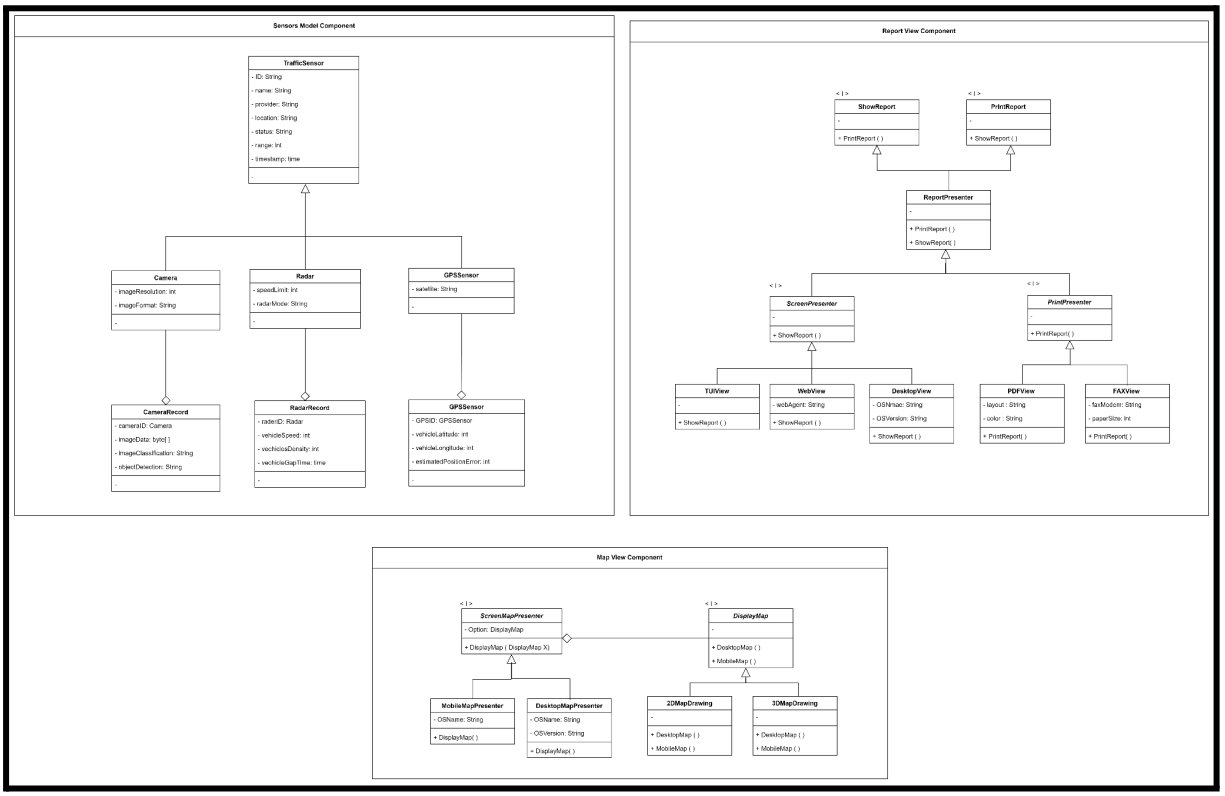

Figure 3.3.1- Class Diagram

CHAPTER 4: ANALYSES AND DISCUSSION

4.1 Design Principles and Decisions Applied in Architecture

The STCS follows three golden balls of design principles: separation of concerns, low coupling, and high coherence. To achieve separation of concerns, some of the system services are divided into view, controller, and model components, instead of having a single large component. For example, the report service has report view, report controller, and report model components, each with a specific responsibility.

Low coupling is achieved through the use of middleware to connect different parts of the system. Rather than having direct calls between components, the broker acts as a middleman, making the sensor side more independent from the rest of the system and allowing for easy plug-in and out.

Lastly, high coherence is achieved by implementing modular components with shared goals, instead of having a monolithic system with different modules that are affected by changes to unrelated modules or failures. Different modules are created for different services, such as the data tier, which includes all the components that communicate with the database.

4.2 Architectural Analysis

4.2.1 Patterns in System Design

The business started in this section, and our first step was to explore the design space we have and select the most promising architectural structures for our needs. To accomplish this, we examined a range of different views for each architecture and conducted thorough analyses to determine the most suitable one.

We have developed a table 3.2.1 that outlines the various views we would be considering with some justification to choose them as candidates or cancel them to ensure that we build a solid foundation for our project.

Architectural Structures	**Architecture Patterns**	**Result**	**Justification**
Component-and-connector (C&C) structures	Pipe-and-filter	Candidate	The Pipe & Filter pattern can be used to design the data processing and analysis component in a Smart Traffic Control System (STCS). The first filter could be responsible for collecting data from sensors, while the second filter could be responsible for cleaning and normalizing the data then can be sent to the AI model component to make decisions about traffic state.
	Model-View-Controller (MVC)	Candidate	The MVC can be used in STCS. The model can be used to store sensor data, while the controller can act as a management subsystem to facilitate the connection between the model and the view. The view can be used to display different types of information, such as report views, map views, and authentication views.
	Client-Server	Candidate	The Client-Server can be used in STCS. This approach provides a simple way to connect clients, such as administrators and workers, to servers that provide either local or remote services, such as report services and map services.
	Broker	Candidate	Brokers support guaranteed message delivery by using message queuing, which is important in STCS due to the

			large amounts of data being sent by several clients. This data includes traffic volume, vehicle speed, and congestion, which are sent to the server for processing.
Allocation structures	Map-Reduce	Not sure	The STCS system will rely more on historical data for computation rather than handling large volumes of data in real-time.
	Multi-Tier	Candidate	The Multi-Tier approach divides the system into multiple tiers, which can be distributed across several locations. This approach can work for the STCS system since the system is not centralized in a specific location. For example, the data collection tier could be located in one place, while the processing tier could be located in another, and the third tier for data tier and admin panel tier could be located elsewhere.
Module structures	Layer structure	Candidate	The admin panel in STCS system can be divided into layers, such as authentication layer, administrative view layer, and administrative controller layer.

Table 4.2.1.1

After considering multiple pattern options for our system, we have decided to work with the client-server, MVC, broker, layer, multi-tier, and semi-microservice patterns. Implementing these patterns will incorporate best practices into our system.

The layer and multi-tier designs improve security by grouping issues into tiers and levels that can be protected separately. As a result, system security is increased and sensitive data is protected from unauthorized access.

The microservice enhances modifiability by decomposing the system into more manageable, independent services that can be created, tested, and deployed independently. Individual services can now be updated and modified more easily without affecting the system as a whole.

4.2.2 Middleware architectures and frameworks

The architecture of the Smart Traffic Controller system (STCS) incorporates essential elements of middleware architectures and frameworks to ensure efficient and secure traffic management. A key aspect of this architecture is the utilization of message brokers, which is a type of Message- Oriented Middleware (MOM), that acts as middleware components facilitating communication between different system modules and supports guaranteed delivery of messages.

The first message broker is positioned between the traffic hardware sensors and the analyzer module. This message broker collects and manages messages from the sensors, effectively decoupling the sensor hardware from the analyzer. By employing asynchronous communication and reliable message queuing mechanisms, this message broker enables seamless data flow, allowing the analyzer to process the collected data efficiently. Furthermore, a second message broker is integrated into the system to handle communication with the output hardware, specifically the traffic signals. This message broker serves as an intermediary between the analyzer module and the traffic signals, providing a centralized platform for transmitting the analyzed results. By leveraging message queuing and routing capabilities, this architecture ensures smooth and timely communication, enabling the traffic signals to respond appropriately based on the analysis outcomes. For message brokers in our system, we recommend using IBM WebSphere MQ, Apache ActiveMQ, or RabbitMQ. These frameworks are all acceptable options.

To enhance the system's security and authenticity, an API gateway is incorporated as a crucial middleware component. This API gateway is strategically placed before independent services within the architecture, such as the Map external services, DB service, and Data Analyzer module. By doing so, the API gateway acts as a primary entry point for all external requests, effectively safeguarding the system from unauthorized access and potential threats. Additionally, the API gateway plays a pivotal role in request routing, protocol translation, authentication, and request/response transformation. By enforcing secure communication protocols, verifying the authenticity of requests, and translating between different protocols, the API gateway ensures the integrity and confidentiality of data transmission. Additionally, it handles the transformation of requests and responses, enabling seamless integration between the STCS and external services, thus enhancing interoperability and extensibility.

4.2.3 Cloud containerization and visualization in STCS

In the proposed architecture, the utilization of independent services for different functionalities, such as the database, admin panel service, and analyzer module, aligns with the principles of cloud containerization and virtualization. By implementing containerization techniques, each service can be encapsulated within a lightweight, isolated container, ensuring consistency and portability across various cloud environments. Containers provide an efficient means of packaging and deploying these services, enabling seamless integration and scalability. Also, containerization allows for resource isolation and efficient resource utilization, as each service can be provisioned and managed independently. This modular approach enhances the system's flexibility, allowing for easier modification, scaling, and deployment of individual services, while ensuring the overall stability and performance of the system.

The use of cloud virtualization complements the architecture's independent services approach. Virtualization enables the creation of multiple virtual machines (VMs) on a single physical server, ensuring optimal utilization of resources. Each independent service can run within its own virtual machine, providing isolation and security between different components. Virtualization also enables efficient resource allocation and management, as the computing resources of the physical server can be dynamically divided among the virtual machines based on demand. This technique enhances the system's scalability, as additional VMs can be provisioned or existing VMs can be scaled up or down to accommodate changing workloads. The combination of containerization and virtualization techniques supports the architecture's modularity, scalability, and efficient resource utilization in the cloud environment, facilitating the effective operation of the Smart Traffic Controller system.[1][2]

4.3 Mapping System Design to Quality Attributes

The system must meet certain quality attributes in order to be deemed acceptable and effectively fulfill its purpose. These attributes as it states in requirement analysis are: privacy, scalability, modifiability, security, reliability, and performance. This section will discuss how the proposed system architecture tries to achieve them.

4.3.1 Security and Privacy

The proposed system prioritizes achieving a high level of security. One approach to achieve this is by implementing an API gateway as the primary entry point for all external requests. This design aims for effective protection of the system from unauthorized access and potential threats. The API gateway also plays a crucial role in ensuring authentication and

verifying the authenticity of requests, thus contributing to the overall security and privacy of the system.

Furthermore, the proposed architecture suggests placing an Intrusion Detection System (IDS) at the most critical entry point, which is the point where all collected data is sent to the centralized data for analyzing. This strategic placement enables the system to effectively detect and prevent threats originating from hardware sensors, thereby mitigating risks such as spoofing attacks or distributed denial-of-service (DDoS) attacks. This additional layer of protection enhances the system's security and ensures the privacy of sensitive data.

It should be emphasized, nevertheless, that adding several layers to assist system security may have unintended consequences for the system. It limited the completion of the crucial task and reduced communication between system components, which might have an impact on the system's performance in general.

4.3.2 Scalability and Modifiability

This architecture's division into services promotes higher scalability and modifiability. By encapsulating functionalities within independent services in different layers (presentation/ processing/ storage) so separation of responsibility here pops up to help in a modifiability concept , the system becomes more flexible, allowing for easier modification and expansion of specific components without affecting the overall system, for concrete example about easy scaling is the following, the expansion in the server side or the client side doesn’t cause any damage as a result of having brokers . Furthermore, this approach enhances the system's resilience and fault tolerance, as the services can be scaled individually to meet varying traffic demands or accommodate additional features.

4.4 How System Architecture Adapt to Handle Changes

Our architecture design is open to change to market conditions and adapt to new trends, these can be done by adopting common design patterns and can be approved by following: in our project adding other type of sensor, camera will not generate a huge ripple to the rest of the system the broker and gateway will observes this effect by the low coupling. In addition, MVC design pattern allows different views to different roles in the system so we can add view for maintainer without in force minimum impact into the system. Also, we have a Tier pattern, in the future, if our system can support the mobile view, we can add new functionality to our system, such as (Alert), send alerts to drivers to avoid congestion. The separation of concerns that we get from the Tier pattern will help in this matter.

4.5 Architecture Evaluation

As the final stage of our trip, we evaluated our software architecture using the Tactics-Based Questionnaire. The evaluation focused on the following areas:

- **In term of privacy:**

- Is sensitive data only accessed by authorized users? Yes, the system enforces authentication and access control to restrict access to sensitive data.

- **In term of scalability and modifiability:**

- Can new features be added without affecting unrelated parts of the system? Yes, the broker reduces coupling between system parts, allowing the server and client sides to scale independently.

- Can the system scale up or down as needed? Yes, the system can be deployed on various cloud services.

- Can components be modified without affecting other components? Yes, the middleware was designed to support this.

- **In term of security:**

- Does the system support data encryption? Yes, there is an encryption feature in the broker.

- Can the system detect attacks? Yes, there is an IDS in place.

- Does the system resist attacks? Yes, only users with the right credentials can access the system.

- How does the system recover from attacks? The system has a standby database synchronized with the primary database.

Based on this checklist, we can confirm that our software architecture meets the business goals of being secure, modifiable, and scalable for smart applications.

CHAPTER 5: CONCLUSION

5.1 Conclusion

In conclusion, this report has presented the design and architecture specification of the Smart Traffic Control System (STCS). It described the main system requirements, both functional and non-functional aspects. Furthermore, it proposed a well-defined system architecture as a proposed solution of the problem, highlighting the necessary structures and design considerations for the system. The report also conducted a comprehensive analysis and provided justifications for the proposed solution and design decisions regarding the system architecture. It further explained how the selected design patterns and structures effectively

address and consider the required quality attributes needed in system requirement. Finally, it demonstrated how the proposed design accommodates changes in market conditions and technology trends.

5.2 References

[1] The University. (1978). *What is Virtualization?*. Amazon. https://aws.amazon.com/what-is/virtualization/

[2] Virtualization in cloud computing - javatpoint. www.javatpoint.com. (n.d.). https://www.javatpoint.com/virtualization-in-cloud-computing

APPENDICES

1. Use Case Diagram

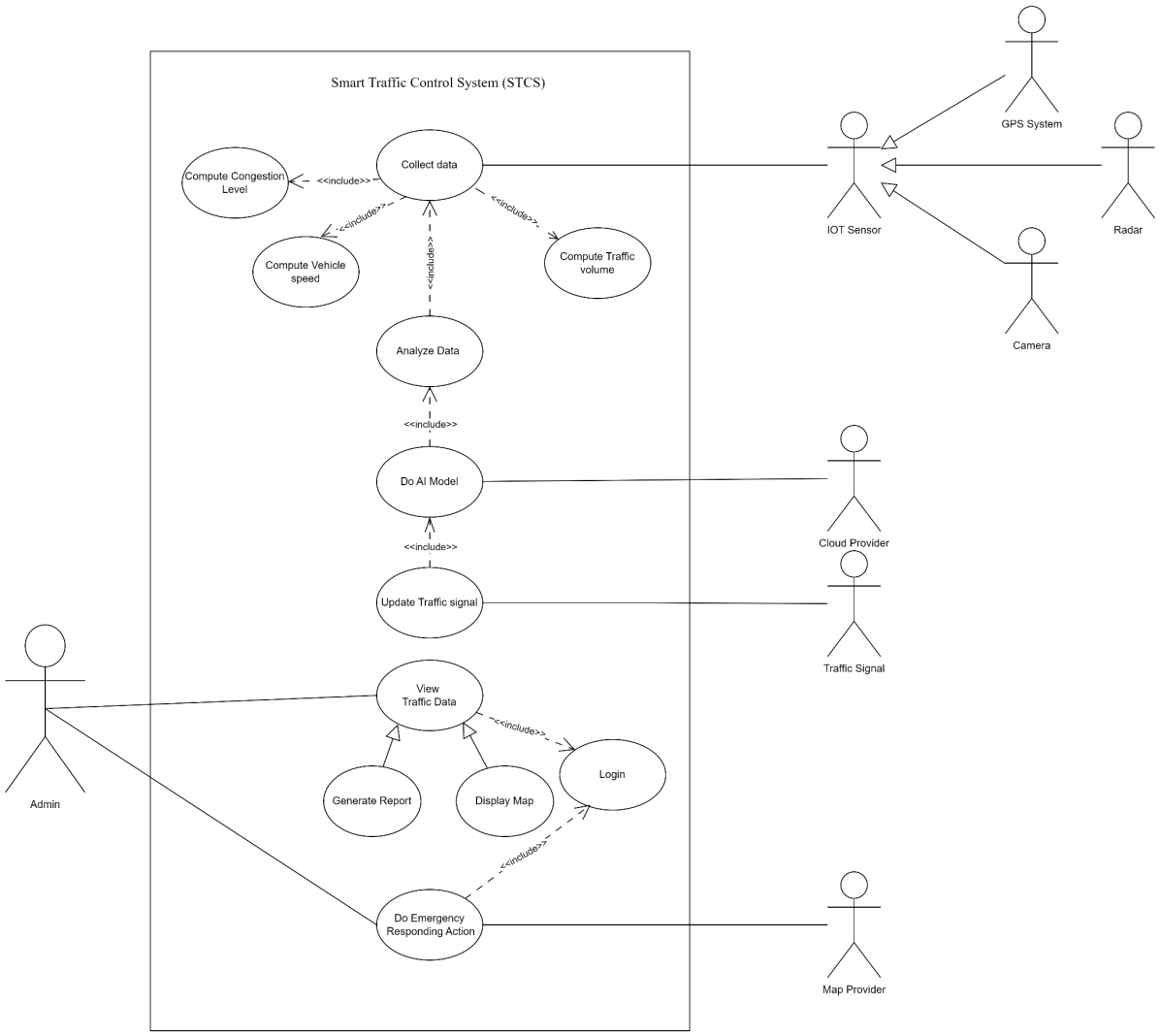

Figure 1- Use Case Diagram

2. Diagrams Link

[Copy of Arch_Project.drawio - Google Drive]