Requirement Document

## **Team-4**

**Team Members:**

PULKIT GUPTA 2021201037

SHREYASH AGRAWAL 2021201074

RAHUL KATYAL 2021201083

**INDEX**

**1. Overview**

**1.1Service Lifecycle Manager**

**1.2 Sensor Manager**

**1.3 Block Diagram**

**2.Intended Use**

**2.1 Intended Use**

**2.1.1 Service Lifecycle Manager**

**2.1.2 Sensor Manager**

**2.2 Assumption and Dependencies**

**3.System Features and Requirements**

**3.1 Functional Requirements**

**3.1.1 Service Lifecycle Manager**

**3.1.2 Sensor Manager**

**3.2 Non Functional Requirements**

**3.2.1 Fault tolerance**

**3.2.1.1 Platform**

**3.2.1.2 Application**

**3.2.2 Scalability**

**3.2.2.1 Platform**

**3.2.2.2 Application**

**3.2.3 Accessibility of data**

**3.2.3.1 Application**

**3.2.3.2 Sensors**

**3.3 Specification about the application**

**4. Primary Test Cases**

**5. Interaction between Components**

**5.1 Application deployer with Service Lifecycle manager**

**5.2 Request Manager with Sensor manager**

**6. Use Cases**

**1. Overview**

Basically Our team has been given the task of creating 2 components

a)Service Lifecycle Manager

b)Sensor Manager

**1.1Service Lifecycle Manager:**

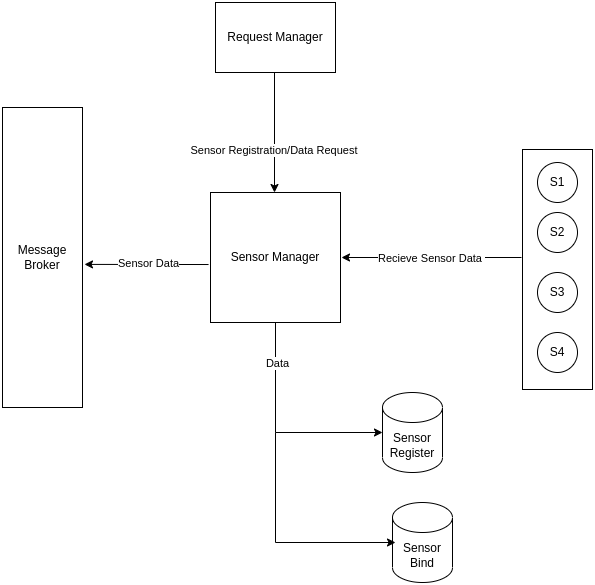
The Service Lifecycle Manager will be in charge of all service-related tasks such as starting a new service instance, recovering a service instance, terminating the entire service or terminating a service instance. It will also be in charge of monitoring all services.It is also in charge of providing an easy-to-use general-purpose logging system that records all events associated with a specific service.

**1.2 Sensor Manager:**

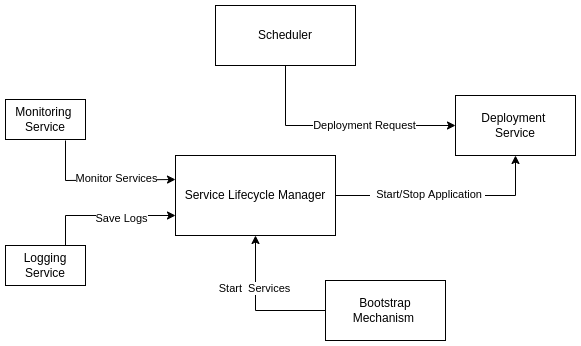
It will manage all the sensor related data like Sensor Type , Location (IP Address,Port number, Physical Location) and Assign a Sensor ID to it, which will be used when the end user will be using the app.It will also send these details to the Message Broker.

**1.3 Block Diagram:**

**Sensor Manager:**

****

**Service Lifecycle Manager:**

****

**2.Intended Use**

**2.1 Intended Use:**

**2.1.1 Service Lifecycle Manager:**

The Service Lifecycle Manager will be in charge of all service-related tasks such as starting a new service instance, recovering a service instance, terminating the entire service or terminating a service instance. It will also be in charge of monitoring all services.It is also in charge of providing an easy-to-use general-purpose logging system that records all events associated with a specific service. Service Lifecycle Manager acts as a deployment manager. It receives details of service from the scheduler. It communicates with the server lifecycle manager for getting the address of the node/server on which the service is to be run.

**2.1.2 Sensor Manager:**

During application execution (which is initiated by the user), data from the sensor is required. Sensor manager will communicate with node management via message broker in this situation, and node manager will transfer sensor data to the relevant node that is running the application.It will keep track of all sensor-related information, such as Sensor Type, Location (IP Address, Port Number, Physical Location), and assign a Sensor ID to it, which will be utilized when the end user uses the app. These details will also be sent to the Message Broker.

**2.2 Assumption and Dependencies :**

* Initially we assume that all the essential services are running.
* We are assuming that platform admin is providing correct information about the sensor that are available on the platform
* We are also assuming that the sensor data is only provided when the application demands for it.
* We are assuming that there is no redundancy in the database.
* After start of any service, the component should tell the Service lifecycle manager so that it can add it to the services database.

**3.System Features and Requirements**

**3.1 Functional Requirements:**

**3.1.1 Service Lifecycle Manager:**

* It is responsible for all the service related tasks.
* Starting a new service instance is done by Service LCM.
* Terminating an entire service is done by. it
* Recovering a service Instance is also done by it.
* It is also responsible for terminating a service instance.
* Monitoring of all the services is also done by Service LCM.
* It also records all events associated with a specific service through a logging system.
* It also receives the details of service from the scheduler so that it can start the service.
* It communicates with the server lifecycle manager for getting the address of the node/server on which the service is to be run.

**3.1.2 Sensor Manager:**

* It will add all the types of sensors supported by the platform to the Sensor Register database.
* Controller registration is performed in the same way as Sensor registration
* If the sensor is registered successfully then a success response is sent to the platform admin else not.
* It will bind all the sensors by the port number, IP and location provided by the end user of the app.
* If a sensor is binded for that particular application then that sensor cannot be binded again for the same application.
* If the sensors are binded successfully then a success response is sent to the platform admin else it will send an error message.
* All these steps will be followed for controllers data binding.

**3.2 Non Functional Requirements**

**3.2.1 Fault tolerance:**

**3.2.1.1 Platform**

To check whether all the components (like scheduler, deployer, application manager etc) of the platform are working properly, and in case of failure reinitialize the component.

**3.2.1.2 Application**

To check if the application instances are working properly or not. If they are not

working then re-initialize another app instance.

**3.2.2 Scalability:**

**3.2.2.1 Platform**

Any number of AI models can be deployed and sensors can be registered to the platform.

**3.2.2.2 Application**

If an application instance is consuming resources above a specified threshold, then that instance will be shifted to a new node.

**3.2.3 Accessibility of data:**

**3.2.3.1 Application**

The application will specify in the configuration which AI models, sensors it will use at the time of deployment. Based on the entry of the model and sensor in the configuration file it will be able to use the services of the platform.

**3.2.3.2 Sensors**

Sensors will be able to access data based on the requirements of the application developed.

**3.3 Specification about the application**

* Scalability
* Reliability
* Security
* Usability
* Maintainability

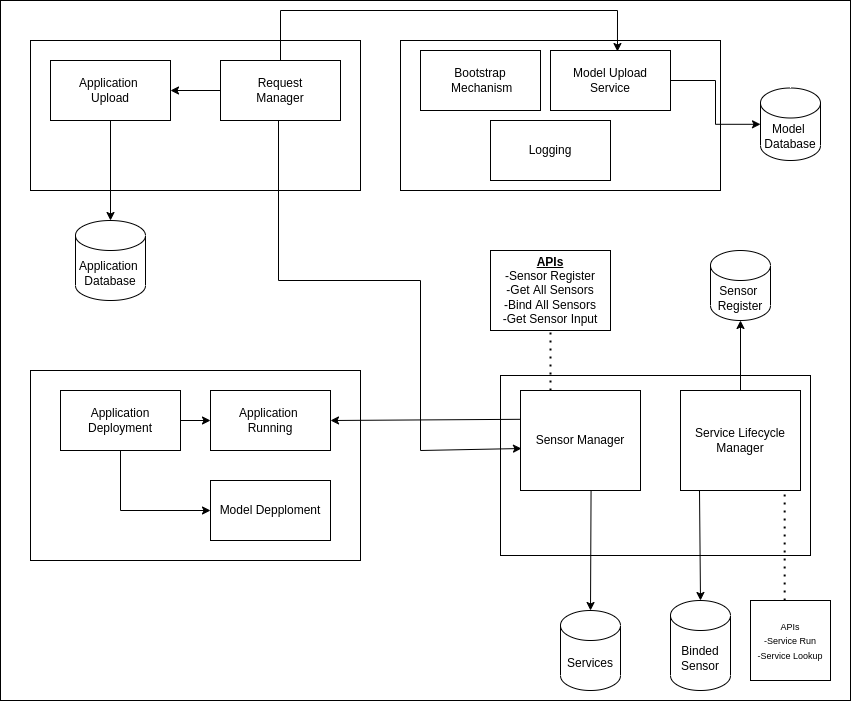
**4. Primary Test Cases:**

* **Platform Admin registers the sensor type and sensor name:**
* If the sensor type is already registered on the platform then an error message is returned.
* If the sensor type is not registered then the sensor type is added to the database and a success message is returned.
* **End user Adds the sensor to be binded through IP, Port and Location:**
* If the sensor is present at that location then it is binded and a success message is returned.
* If the sensor is not present at that location then an error message is returned.
* **App deployer sends the details of service to Service Lifecycle Manager**
* If the service details are valid then the Service Lifecycle manager adds the service to the database and starts the service.
* If the service details are not valid then the Service Lifecycle manager sends an error message.
* **App deployer sends the service name to Service Lifecycle Manager**
* If the service name is correct and the service is in running state then the Service Lifecycle manager will return the port number of the service.
* If the service name is not correct or the service is not in running state then Service Lifecycle manager will return an error message.

**5. Interaction between Components:**

**5.1 Application deployer with Service Lifecycle manager:**

* The Service Lifecycle Manager sends an error message if the service details are invalid.If the service information is correct, the Service Lifecycle Manager registers the service in the database and begins it.
* The Service Lifecycle Manager will return the port number of the service if the service name is correct and the service is functioning. Service Lifecycle Manager will return an error message if the service name is incorrect or if the service is not operating.

****

Interaction of our modules with other teams

**5.2 Request Manager with Sensor manager:**

* The request for registering the sensor will come through request manager to sensor manager.
* Request of getting sensor data for the application will also come from Request Manager
* Request of sensor binding for the application will also come from Request Manager

**6. Use Cases:**

* **Fish swarm detection for efficient fishing**

We may use streamed data from an underground submarine in this use case to determine whether locations are suitable for fishing. This can also be used to determine whether fishing for a species is profitable based on its market rate and other complex criteria.

# **Smart Traffic Management System**

# The traffic signal, which is at the heart of the traffic system, is one of the most critical places in need of development. Traditional traffic signals have a number of flaws, including poor time management at road intersections, vulnerability to weather conditions such as rain, and the inability to provide priority to emergency vehicles.We can use IOT to manage traffic more efficiently.

# **Smart Grids:**

# A smart grid is one of many beneficial IoT examples. It is a holistic solution that uses a wide variety of Information Technology resources to reduce electricity waste and price on current and new gridlines. Electricity efficiency, reliability, and economics will all benefit from a future smart grid.