Assignment Documentation 2

1. Conceptual Architecture

This assignment is centered around improving the distributed application designed to simulate energy consumption sensors, process their data, and monitor energy usage events. The application is implemented to act like smart meters and involves two new components: the **Sensor Simulator** and the **Monitoring and Communication Microservice (MCMS)**. Data is communicated between these components using RabbitMQ as the message broker.
The system's core functions include:

- Collecting energy data in 10-second intervals from a CSV file.
- Computing hourly energy consumption.
- Logging or notifying users when energy consumption exceeds predefined thresholds.

2. Components

Sensor Simulator

The sensor simulator is a standalone app that is used to read the consumption data from a csv file called "sensor.csv", which contains rows of values corresponding to consumption data.
After these values are read, the app sends the necessary information in the form of a JSON formatted message through a RabbitMQ queue every 10 seconds to the MCMS.

```
{
   "timestamp": 1570654800000,
   "device_id": "5c2494a3-1140-4c7a-991a-a1a2561c6bc2",
   "measurement_value": 0.1
}
```

This app is autonomous meaning it doesn't depend on any other microservices to work.

Monitoring and Communication Microservice (MCMS)

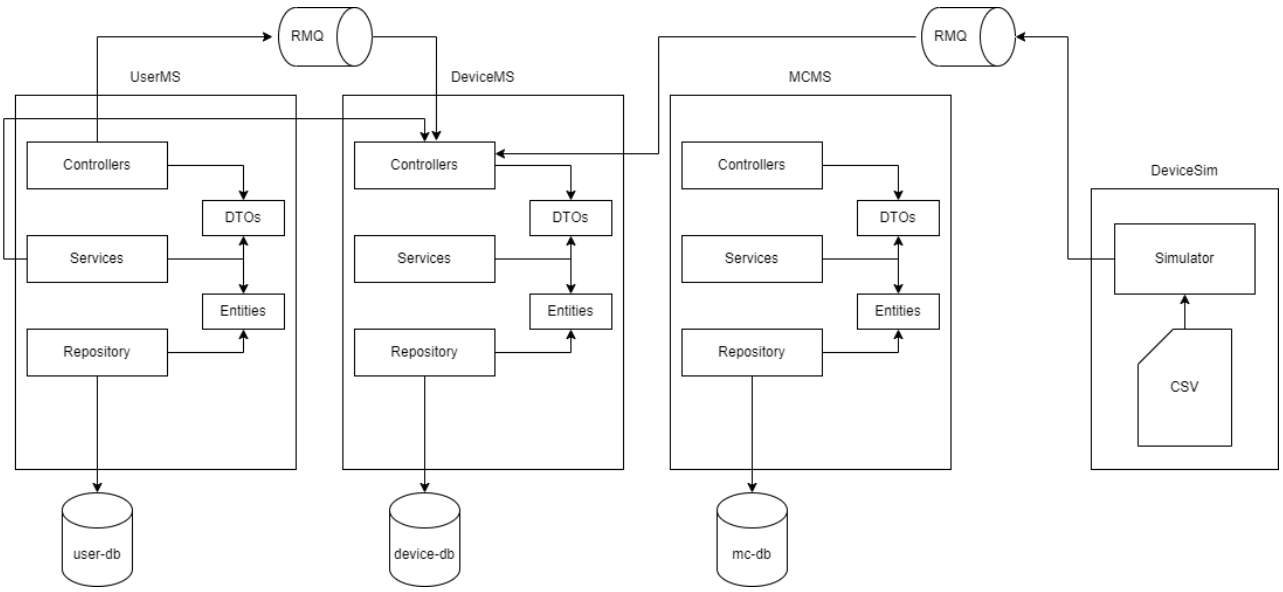
The microservice processes the data and performs the following tasks:

1. **Data Consumption**: Consumes JSON messages from RabbitMQ sent by the Sensor Simulator and the DeviceMS.
2. **Hourly Consumption**: Computes hourly energy usage by summing up the measurements received within 60 seconds.
3. **Threshold Monitoring**: Checks if the hourly energy consumption exceeds a configured limit. When the threshold is exceeded, it logs a notification in the console.
4. **Data Storage**: Stores processed hourly energy consumption in a database.
5. **Frontend Integration**: Provides an endpoint that enable users to view energy consumption data for specific devices and dates.

3. Communication and Data Flow

1. **Message Broker (RabbitMQ)**:
   RabbitMQ acts as middleware to ensure delivery of messages between the Sensor Simulator and the backend as well as between DeviceMS and MCMS. It supports message queuing, so the data is consumed even if the backend experiences downtime.

2. **Data Format (JSON)**:
   All communication between the components is standardized using JSON.



4. Deployment Architecture

Sensor Simulator :

- Deployed as independent instance.
- Autonomously reads sensor.csv and sends JSON data to RabbitMQ.

RabbitMQ :

- Serves as the message broker.
- Queuing and delivery of data between the Sensor Simulator and the backend microservice.

Monitoring and Communication Microservice :

- Deployed as containerized instance.
- Consumes messages, processes energy data, and stores results in a database.
- Logs notifications when consumption exceeds thresholds.

Traefik Proxy :

- Handles traffic routing and load balancing.
- Secures communication through SSL termination.
- Enables service discovery and health monitoring for containerized components.