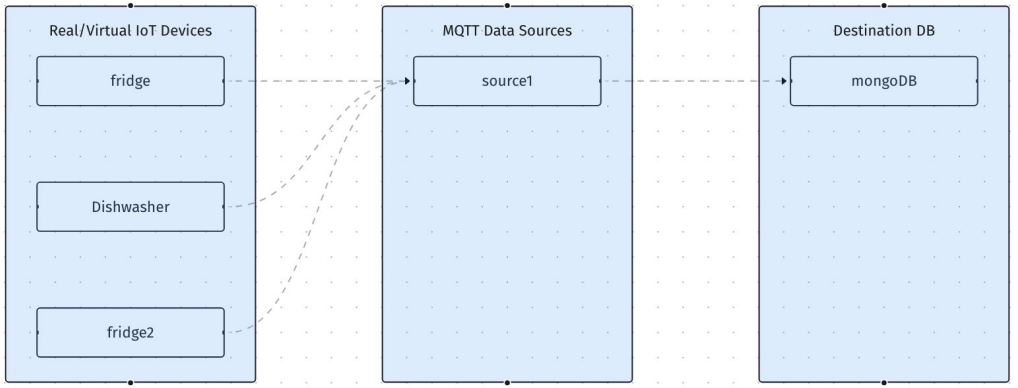
CECS 327: Building an End-to-End IoT System

Omar Youssef
Jeff Kim

Introduction

This project implements an end-to-end IoT system that integrates IoT sensor data, a MongoDB database, and a client-server architecture to process and analyze user queries. The system uses metadata from IoT devices to enhance functionality, and all queries are processed in PST and presented in imperial units.



Building Connection to Database

```
try:
    # Connect to MongoDB
    client = MongoClient(connectionURL, tlsCAFile=certifi.where())
    db = client[DBName]
    table = db[virtualTable]

    # Define time cutoff for the past 3 hours
    time_cutoff = datetime.now(timezone.utc) - timedelta(hours=3)
```

Query Database Q1

```
def query_database(query_type):
    try:
        # Connect to MongoDB
        client = MongoClient(connectionURL, tlsCAFile=certifi.where())
        db = client[DBName]
        table = db[virtualTable]

        # Define time cutoff for the past 3 hours
        time_cutoff = datetime.now(timezone.utc) - timedelta(hours=3)

        if query_type == "Q1":  # Average moisture in the fridge
            result = table.aggregate([
                {"$match": {"payload.asset_uid": fridge_asset_uid,
                            "time": {"$gte": time_cutoff}}},
                {"$group": {"_id": None,
                            "average_moisture": {"$avg": {"$toDouble": "$payload.Moisture Meter - moistureMeter1"}}}}
            ])
            # Safely return result or fallback
            result_list = list(result)
            return result_list[0]["average_moisture"] if result_list else "No data found for fridge moisture."
```

Query Database Q2

```
elif query_type == "Q2":  # Average water consumption in the dishwasher
    result = table.aggregate([
        {"$match": {"payload.asset_uid": dishwasher_asset_uid,
                    "time": {"$gte": time_cutoff}}},
        {"$group": {"_id": None,
                    "average_water": {"$avg": {"$toDouble": "$payload.WaterConsumptionSensor"}}}}
    ])
    # Safely return result or fallback
    result_list = list(result)
    return result_list[0]["average_water"] if result_list else "No data found for dishwasher water consumption."
```

Query Database Q3

```
elif query_type == "Q3":  # Device with the highest electricity consumption
    # Fetch fridge ammeter
    fridge_result = table.aggregate([
        {"$match": {"payload.asset_uid": fridge_asset_uid,
                    "time": {"$gte": time_cutoff}}},
        {"$group": {"_id": None,
                    "total_energy": {"$sum": {"$toDouble": "$payload.Ammeter"}}}}
    ])
    fridge_energy = list(fridge_result)
    fridge_energy_total = fridge_energy[0]["total_energy"] if fridge_energy else 0

    # Fetch dishwasher ammeter
    dishwasher_result = table.aggregate([
        {"$match": {"payload.asset_uid": dishwasher_asset_uid,
                    "time": {"$gte": time_cutoff}}},
        {"$group": {"_id": None,
                    "total_energy": {"$sum": {"$toDouble": "$payload.DishwasherAmmeter"}}}}
    ])
    dishwasher_energy = list(dishwasher_result)
    dishwasher_energy_total = dishwasher_energy[0]["total_energy"] if dishwasher_energy else 0

    # Compare and determine the highest consumer
    if fridge_energy_total > dishwasher_energy_total:
        return f"Fridge ({fridge_asset_uid}) consumed more electricity: {fridge_energy_total:.2f} kWh."
    elif dishwasher_energy_total > fridge_energy_total:
        return f"Dishwasher ({dishwasher_asset_uid}) consumed more electricity: {dishwasher_energy_total:.2f} kWh."
    elif dishwasher_energy_total == fridge_energy_total:
        return "Both devices consumed the same amount of electricity."
    else:
        return "No data found for one or more devices."

else:
    return "Invalid query type. Please use Q1, Q2, or Q3."
```

Server processing user request.

```
# Process the query
if data == "Q1":
    response = f"Average moisture (RH%): {query_database('Q1')}"
elif data == "Q2":
    response = f"Average water consumption (gallons): {query_database('Q2')}"
elif data == "Q3":
    response = f"Device with highest electricity consumption: {query_database('Q3')}"
else:
    response = "Invalid query. Please use Q1, Q2, or Q3."
```

Metadata used to enhance the system

1. Device Identification

Each IoT device in the dataset had a unique asset_uid stored in the payload section of the MongoDB documents:

- Kitchen Fridge: uz9-9mr-391-mfq
- Dishwasher: 2w3-l58-e05-a5c
- Second Fridge:

These asset_uids were used to distinguish between devices and ensure that queries fetched data only for the intended device. For example:

- When calculating average moisture (Q1), only data with payload.asset_uid matching the fridge asset_uid was considered.
- Similarly, water consumption (Q2) focused on data with the dishwasher's asset_uid.

2. Time Filtering

Metadata included a time field for each record, which was used to filter data for the past three hours. This ensured that only recent and relevant data was considered in calculations. For example:

- {"time": {"$gte": time_cutoff}}

Challenges Encountered

1. **Aggregation Query Complexity**

MongoDB aggregation pipelines needed to handle:

- Time-based filtering (last 3 hours)
- Data type conversions (ex: strings to numbers)
- Comparison of devices' data (especially for Q3)

2. **Metadata Utilization**

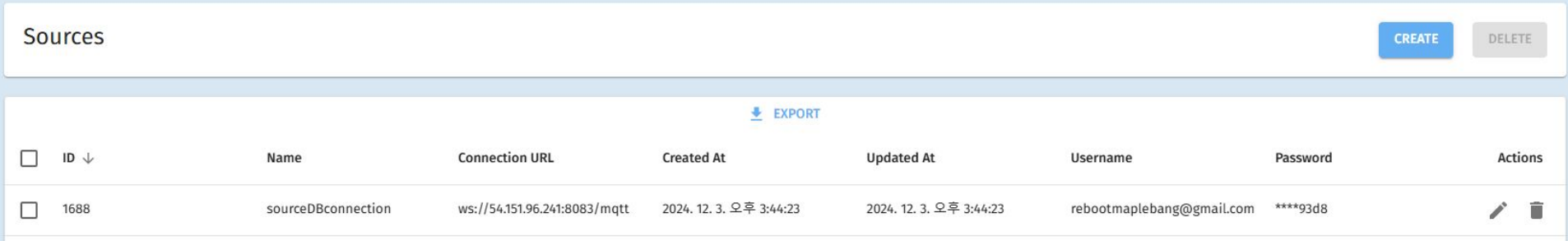
Effectively incorporating metadata (e.g., asset_uid, time, board_name) required a clear understanding of its structure and relevance to queries.

Could not filter by sensor type, had to use each sensor's name.

3. **Integration Testing**

Integrating multiple components (IoT devices, MongoDB, TCP server on Google Cloud VM, and client) required rigorous testing to ensure seamless communication and accurate results.

Dataniz Feedback



Dataniz Feedback

Allowing Metadata removal from the website

Metadata

EXPORT

Device	Latitude	Longitude
Dishwasher	32	59
Refrigerator One	48	23
Refrigerator Two	58	43

Data

Device	Sensor	Timestamp	Topic	Value
Refr...	parent_asset_uid	2024. 12. 10. 오후 11:41:05	conne...	8jo-135...
Refr...	asset_uid	2024. 12. 10. 오후 11:41:05	conne...	f98-3n...
Refr...	Moisture Meter - R...	2024. 12. 10. 오후 11:41:05	conne...	21.3655
Refr...	Thermistor	2024. 12. 10. 오후 11:41:05	conne...	47.5030
Refr...	Ammeter	2024. 12. 10. 오후 11:41:05	conne...	13.2403

Create a Virtual Sensor

BMP180

MPU6050

LM386

MPR121

Thank you