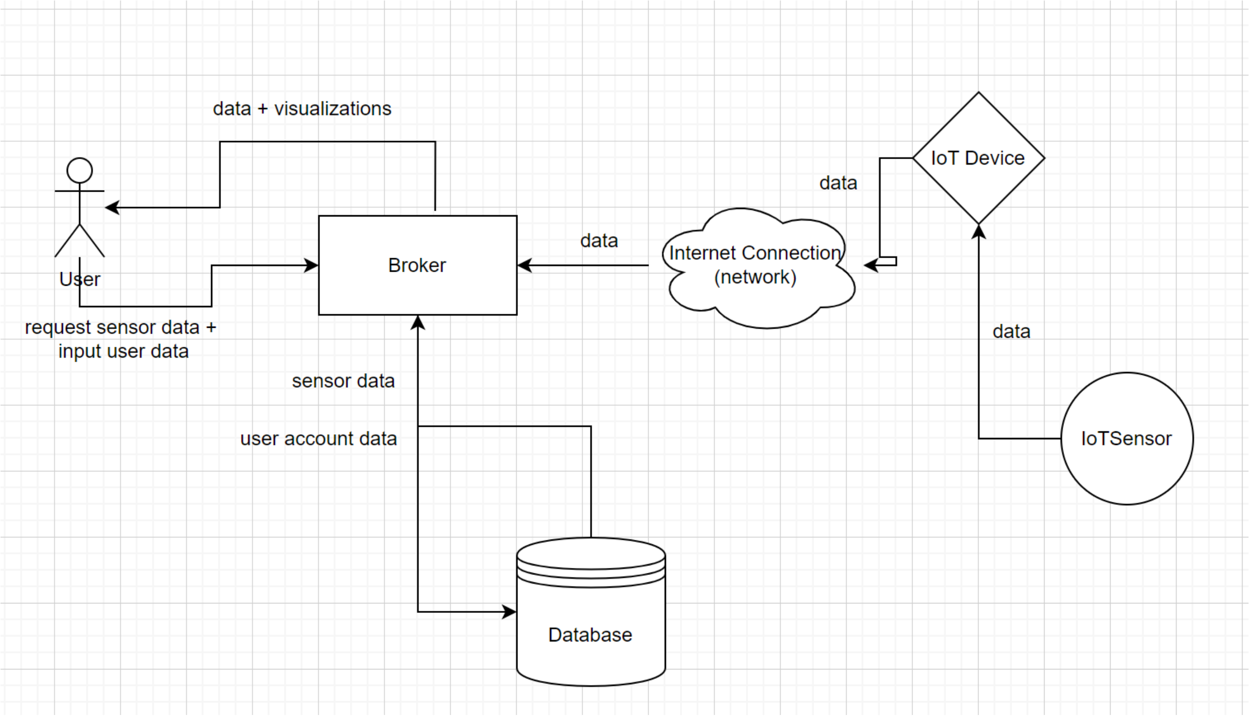
Milestone 3

Feedback M2

Architecture Diagram:



System functions:

The system is comprised of a user, a broker, a network, a database, IoT devices, and IoT sensors. The user interacts with a GUI provided by the broker. Here, the user enters account information to login and gain access to the desired data streams. The broker stores the account information in the database. Once logged in, the user can make a request for sensor data. The broker is connected to the IoT device through a network. The IoT is connected to the sensors through a network. The IoT sensors transmit data to the IoT device, which then transmits the data to the broker. The broker stores the data in the database. When a user requests a data stream, the broker will access the data from the database, perform data manipulations and visualizations to deliver the desired data streams that is useful to the user. A useful form would be tabular data, a plot, and the ability the look to access the raw data. This manipulation is done so that the data can be quickly interpreted by the user. Otherwise, a long stream of data values would need further cleaning and manipulation to deliver quantifiable results.

MQTT follows the Publish/Subscribe paradigm. The sender (Publisher) and receiver (Subscribers) of messages communicate via so-called topics and are decoupled from each other. The connection between them is handled by the broker. The task of the broker is to filter all incoming messages and distribute them correctly to the subscribers. A client doesn't have to pull the information it needs, the broker pushes the information to the client whenever something new is available.

Broker:

- connects to and receives data from IoT devices
- sends and receives data from IoT devices to database
- connects user to data
- provides GUI for user interaction
- sends and receives user information to database

Database:

- stores user data
- stores sensor data

IoT Device:

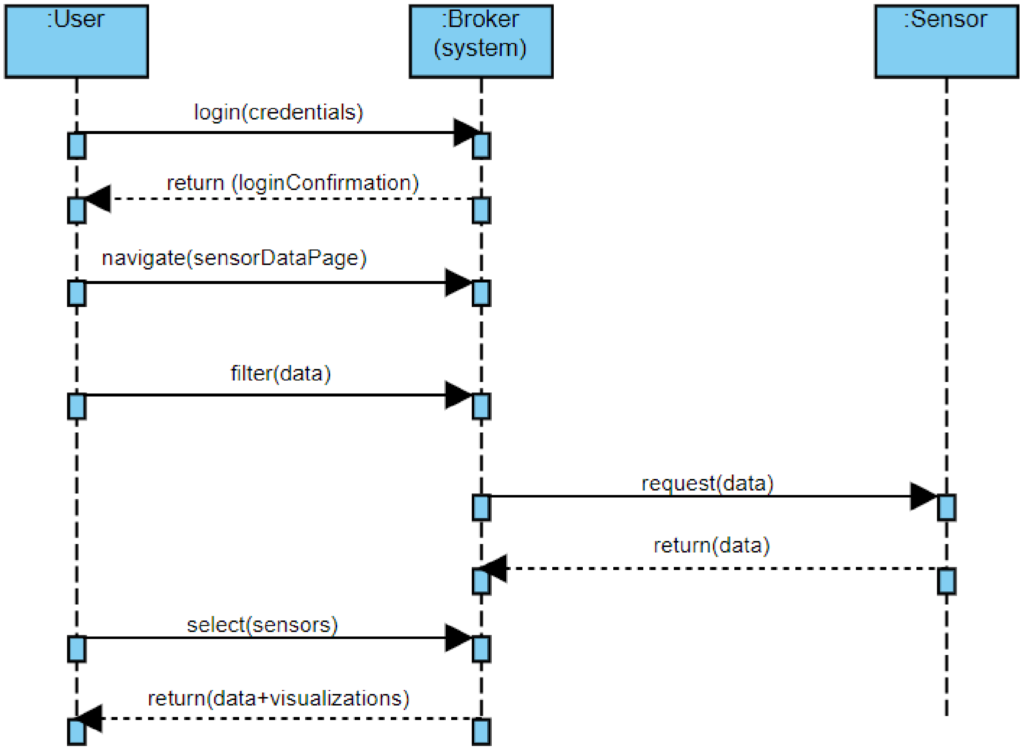
- central connector for multiple IoT sensors
- receives data from network of IoT sensors
- sends data to broker via the network

IoT Sensor:

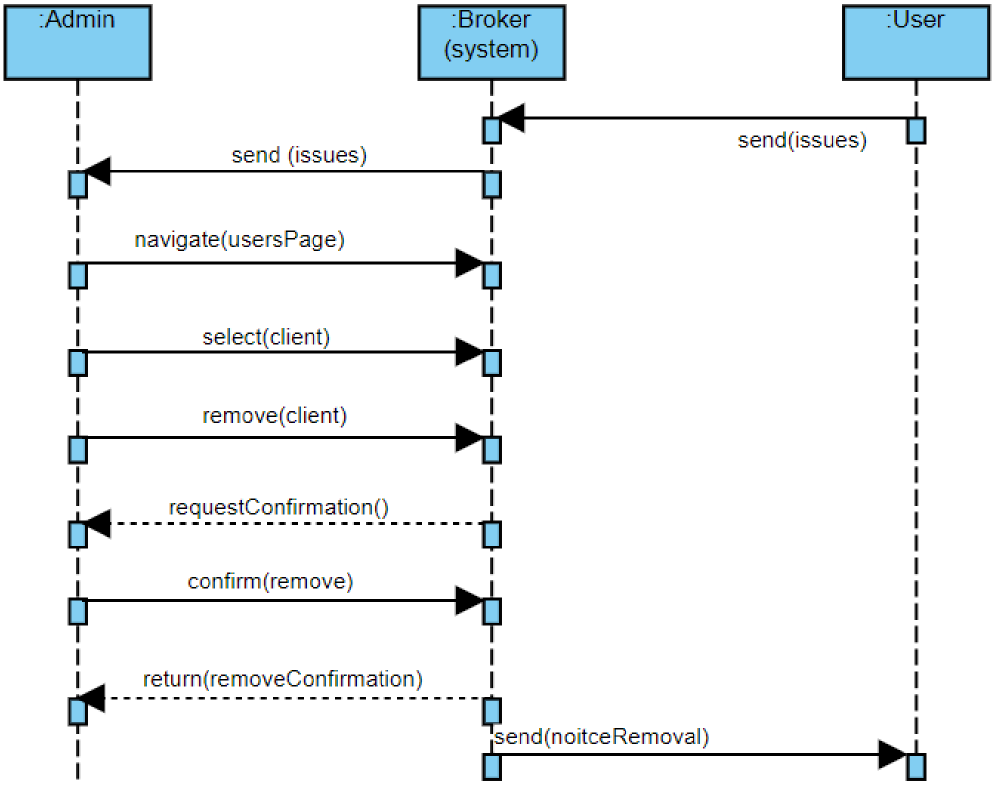
- collects source data
- sends data to IoT device in it’s network

Sequence Diagrams:

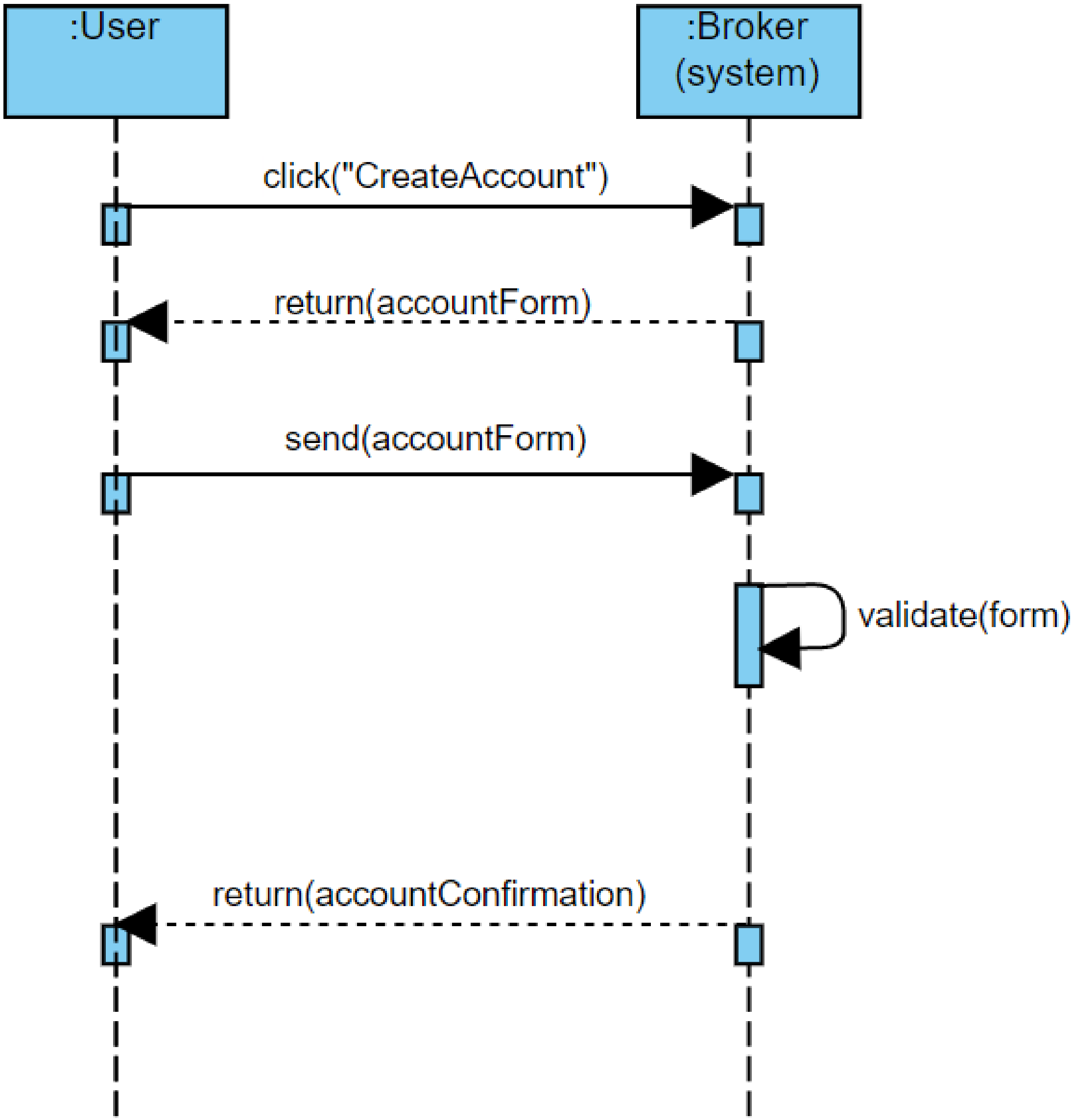
View Sensor Data



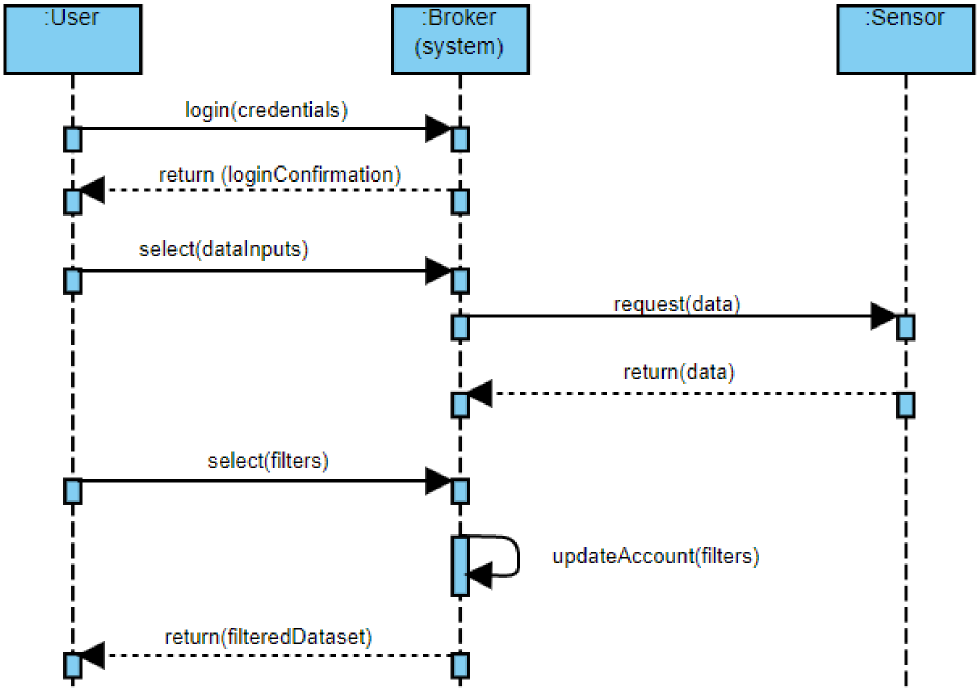
Remove Client



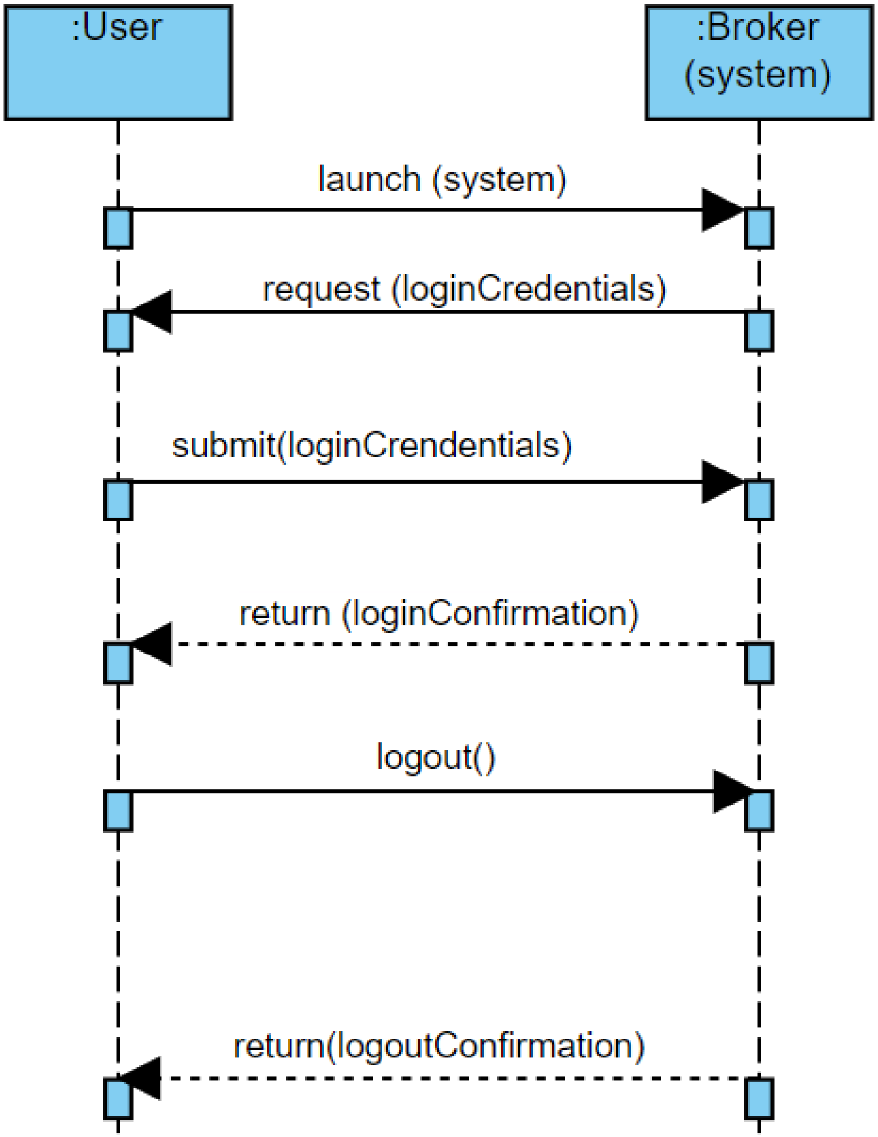
Create Acccount



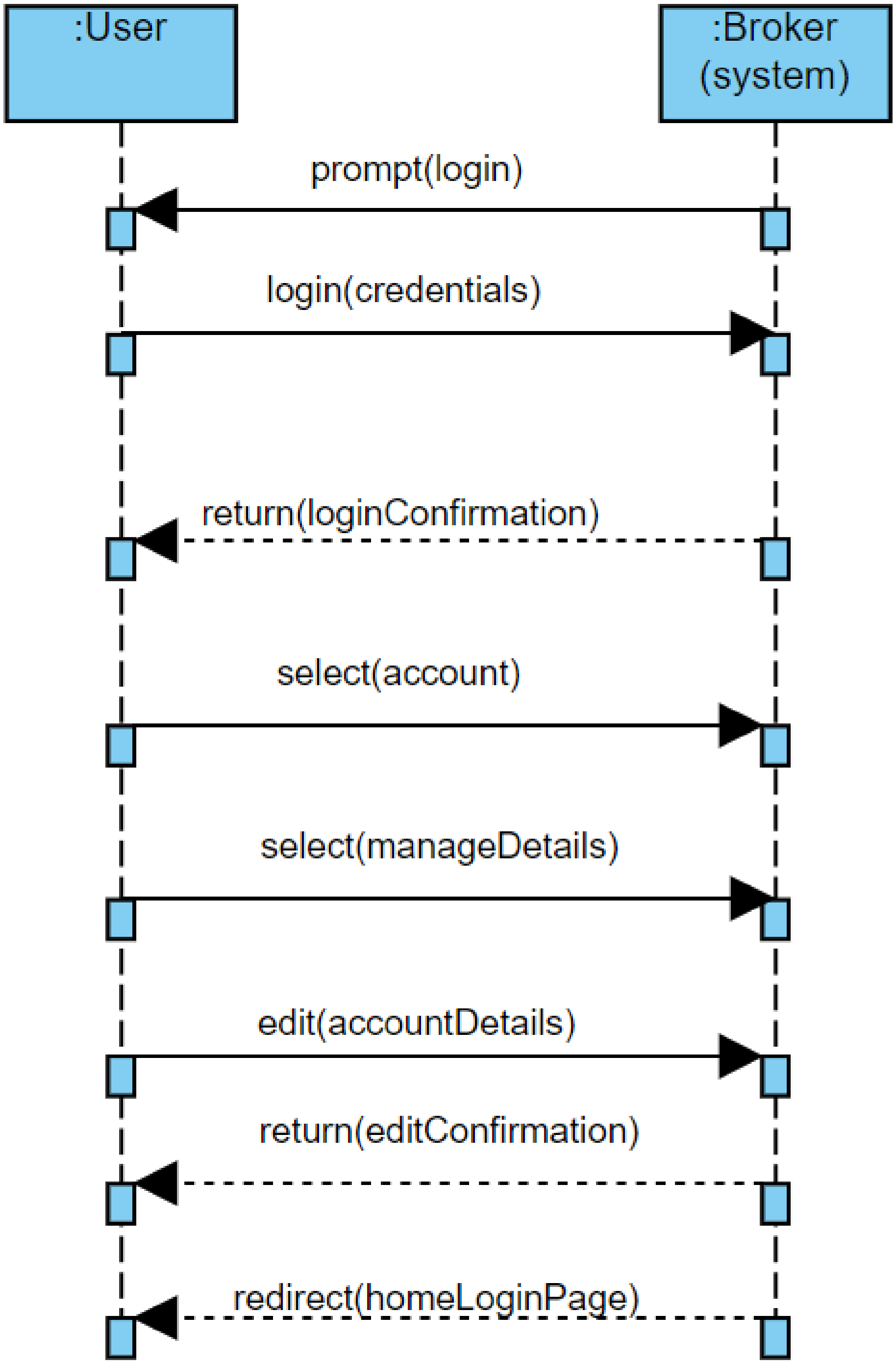
Filter Data


:User
:Broker (system)
:Sensor
login(credentials)
return (loginConfirmation)
select(dataInputs)
request(data)
return(data)
select(filters)
updateAccount(filters)
return(filteredDataset)

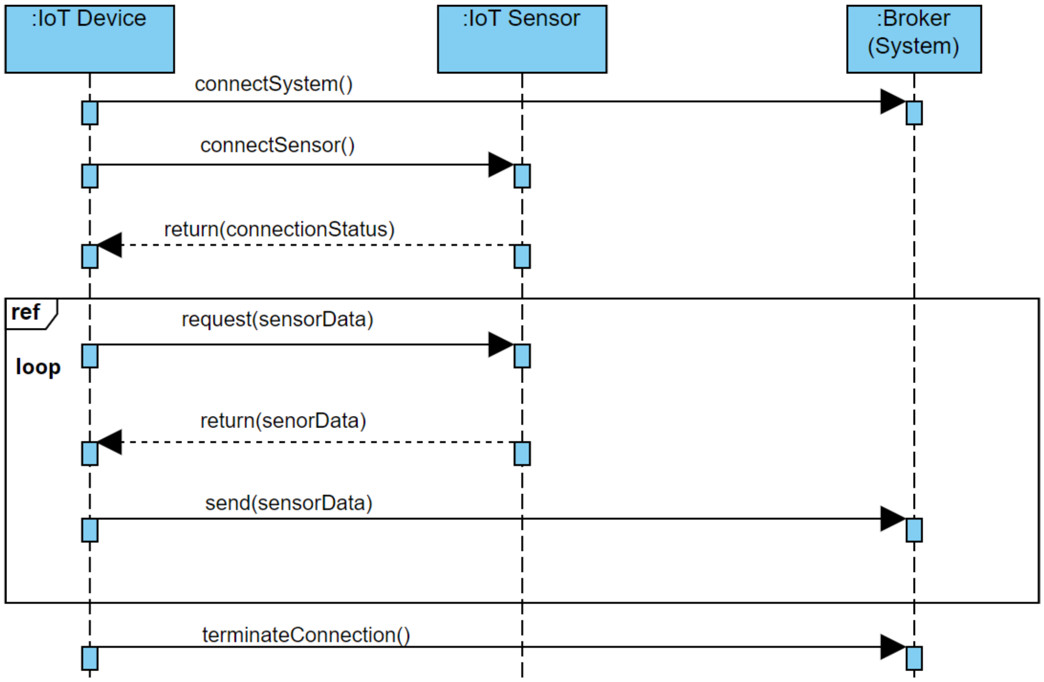
Login/Logout

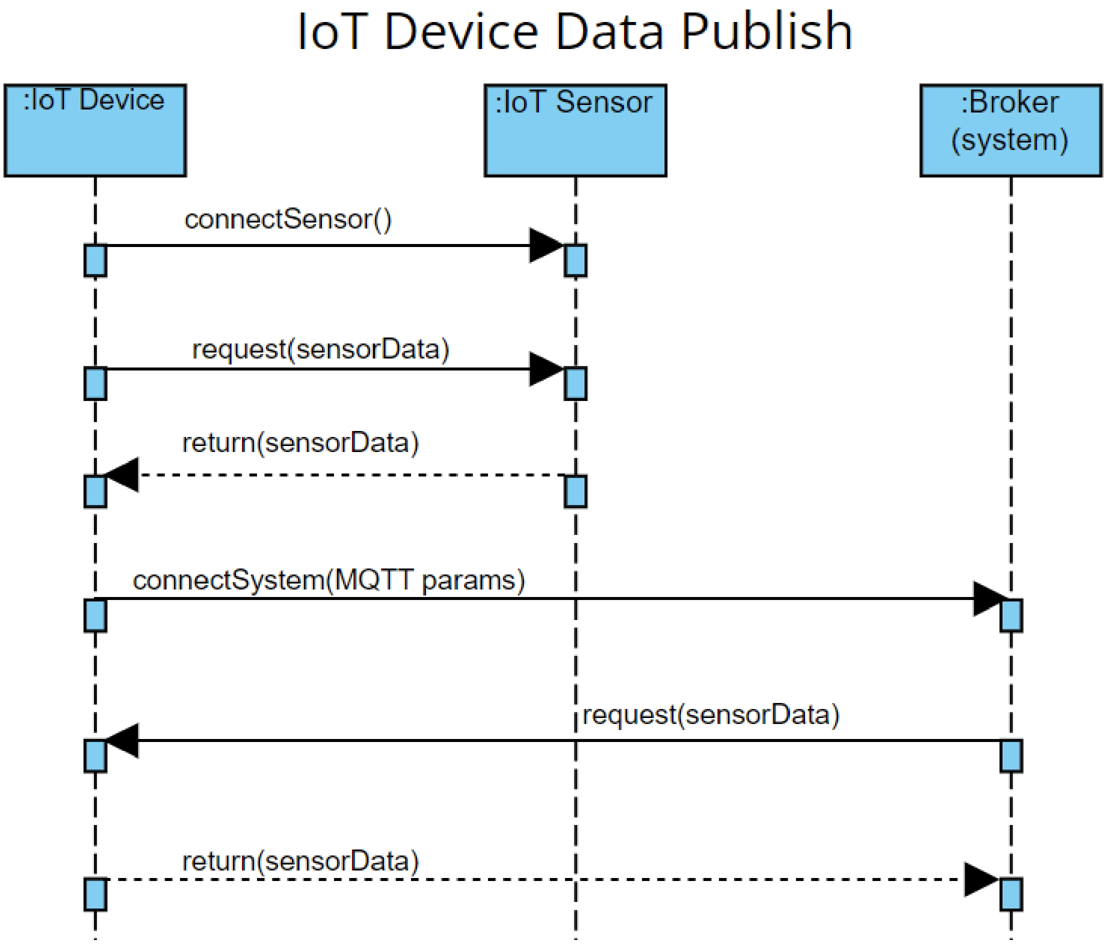

:User
:Broker (system)
launch (system)
request (loginCredentials)
submit(loginCrendentials)
return (loginConfirmation)
logout()
return(logoutConfirmation)

Account Management



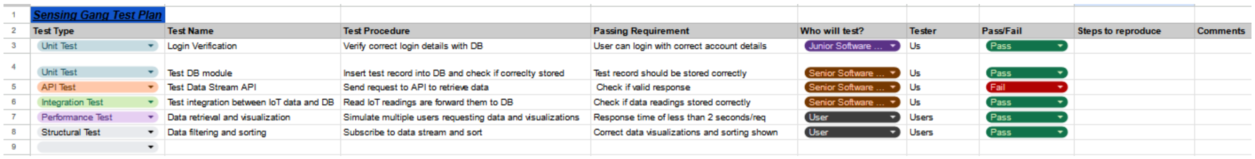
IoT Device Connection





Test Plan

Testing plan with sample Login, database, data stream, data retrieval, and data filtering verification tests, . We can use this spreadsheet to track Unit tests, API tests, integration tests, and performance tests. This will be continually added to throughout the development process using a TDD philosophy.



Sensing Gang Test Plan

Test Type	Test Name	Test Procedure	Passing Requirement	Who will test?	Tester	Pass/Fail	Steps to reproduce	Comments
Unit Test	Login Verification	Verify correct login details with DB	User can login with correct account details	Junior Software ...	Us	Pass		
Unit Test	Test DB module	Insert test record into DB and check if correclty stored	Test record should be stored correctly	Senior Software ...	Us	Pass		
API Test	Test Data Stream API	Send request to API to retrieve data	Check if valid response	Senior Software ...	Us	Fail		
Integration Test	Test integration between IoT data and DB	Read IoT readings are forward them to DB	Check if data readings stored correctly	Senior Software ...	Us	Pass		
Performance Test	Data retrieval and visualization	Simulate multiple users requesting data and visualizations	Response time of less than 2 seconds/req	User	Users	Pass		
Structural Test	Data filtering and sorting	Subscribe to data stream and sort	Correct data visualizations and sorting shown	User	Users	Pass		

Scope:

- Any software involved in the project’s product, including but not limited to:

    - Components
    - Modules
    - Functions
    - Databases
    - User-facing UI
    - Data streams and quality from inputs

Out of scope:

- The following items do not fall in the scope of testing:
    - The IOT devices themselves

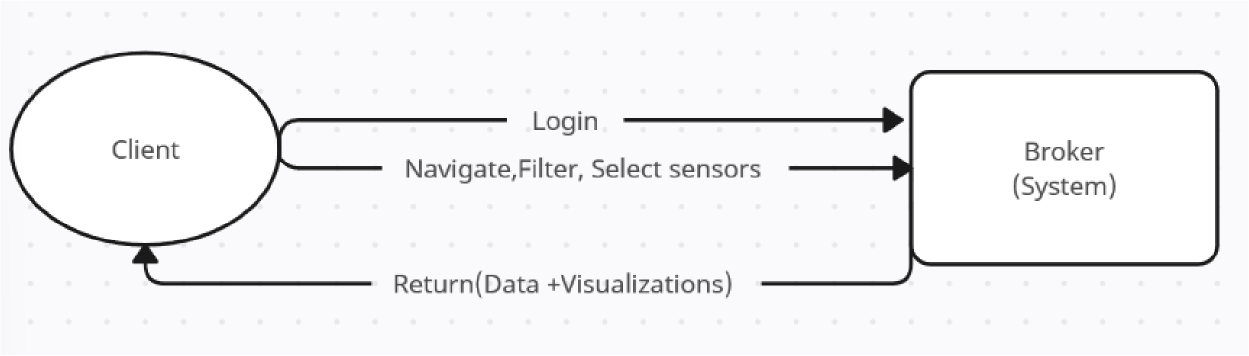
Implementation of Agile testing methodology and TDD

Testing will be continuous through the development cycle. Per the TDD (Test Driven Development) principle, unit tests will be written before code when implementing a new feature. Additional tests will be created to test different structures, larger components, etc. Tests are flexible, and can continuously be updated throughout the development process. Unit tests map out the requirements of a given feature, driving the development process.

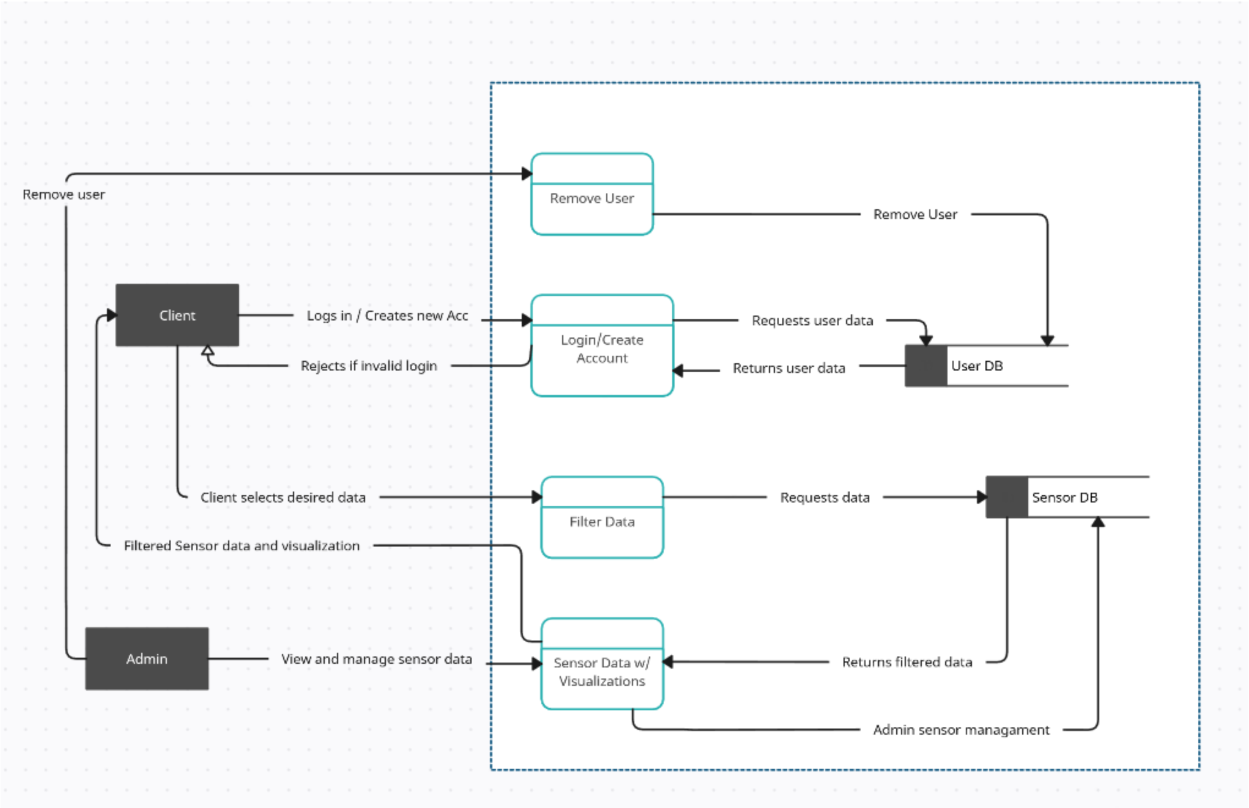
The collective is responsible for testing, meaning that everyone involved in the project will have testing responsibilities. The developer writing the code for a new feature will be responsible for preempting it with a unit test, which will have to be passed in order to implement the code. Tests will first be conducted by the developer who wrote the code in question, but also by other members of the team, as well as users, focus groups, customers, etc. as it becomes relevant.

DFDs:

Level 0 Client-Broker Interaction:

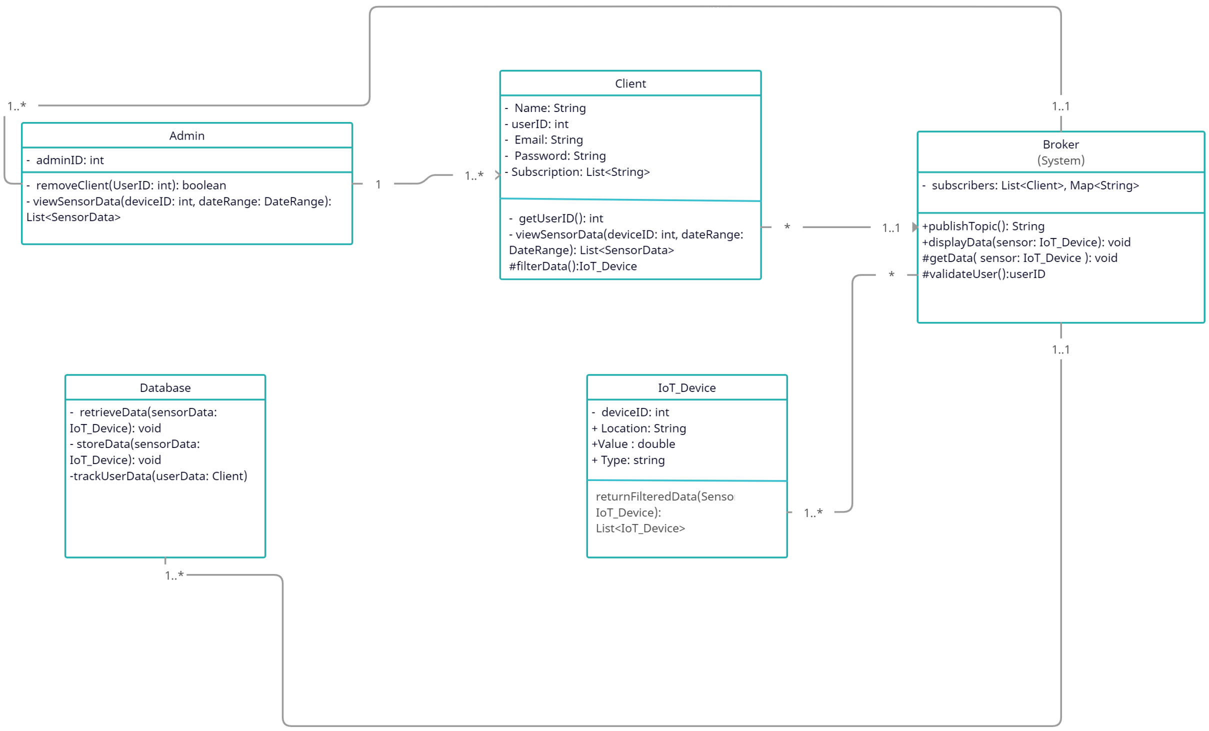


Level 1 Overview Client-Broker Interaction:



This DFD models the data flow of the Client and Broker interaction. In addition to this, the broker establishes a connection with the IoT sensors that we provide data for, and through these connections IoT device data will be transmitted to the broker, which then populates the Sensor database.

UML Class Diagram


1..*
1..1
Client
- Name: String
- userID: int
- Email: String
- Password: String
- Subscription: List<String>
- getUserID(): int
- viewSensorData(deviceID: int, dateRange: DateRange): List<SensorData>
#filterData():IoT_Device
Admin
- adminID: int
- removeClient(UserID: int): boolean
- viewSensorData(deviceID: int, dateRange: DateRange): List<SensorData>
1
1..*
*
1..1
Broker
(System)
- subscribers: List<Client>, Map<String>
+publishTopic(): String
+displayData(sensor: IoT_Device): void
#getData(sensor: IoT_Device): void
#validateUser():userID
*
1..1
Database
- retrieveData(sensorData: IoT_Device): void
- storeData(sensorData: IoT_Device): void
-trackUserData(userData: Client)
IoT_Device
- deviceID: int
+ Location: String
+Value : double
+ Type: string
returnFilteredData(Senso
IoT_Device):
List<IoT_Device>
1..*
1..*