BLACKBERRY IOT TECHNICAL TEST

- Amogh Adithya Bangalore

1. The Problem approach:

The technical requirement involves the creation of a web-based data visualization and simulation platform of Sensor data and storing the status of the sensor in cloud via a REST interface.

The first part of solving the task was to understand how many modules would be required in this project. An application that allows users to submit simulated sensor data and provides a dashboard interface for them to view the live updated data from the database..

The next stage of solving the problem was to understand that the sensor data points must be clear, concise, and readable. The users who use the web application must be able to submit the sensor data easily. The JSON based data structure requirement allows easier readability, database storage, and clear formatted structure when exposed in an API endpoint.

The third stage involved designing the major components: Database to store the data points, a backend server to handle the user data submission and dashboard data requests and a responsive front-end framework that would allow them input submissions and data visualization. The technology stack used for this project has been described in the section 2.

The final process to solving the task is to verify that all the components designed are interacting with each other as expected, the results displayed are correct, and the data points submitted make sense. The process also involves dockerizing the full application which makes creation of builds and running services easier and having version controls systems in place for code management.

2. Technology stack chosen:

Technology Stack and the reasons for choosing the same:

Technology Area	Technology Used	Reason
IDE and debugging tool	Visual Studio Code	VS Code has powerful code debugging, support for Git , built in terminal and multi language support.
Database	MongoDB Atlas (Cloud)	Free tier **cloud-based** Document DB for easy storage and access
Front-end Framework	ReactJS	Flexible Frontend framework with Virtual DOM support

Back-end Framework	Python Flask	REST API familiarity along with support for MongoDB and Routing
Container Technology	Docker and Docker-compose	Makes creation of build files, dependencies and running services easier via containers.
API Testing	POSTMAN	Easy HTTP API endpoint verification

3. Task Questions :

1. What testing approaches would you add to this prototype?
   The protype had mostly system level functional tests and API level tests done for this iteration. To improve the testing methodology, the following approaches can be undertaken:
   - Adding Unit testing using **Pytest** for functional, API and routes to test individual components.
   - Performing **Load** and **Performance** testing on the backend server using JMeter, Fiddler or Locust.
   - **Automation testing** will be a key feature to test successive builds with Jenkins Integration. It can be done using Selenium driver for python. This will improve build dependency and report bugs quickly.

2. Explain why you chose to implement the simulator as a web app or command line utility, etc.
   - The simulator having a graphical user interface as a web application allows users to be engaged in the application and provides them an easier way to use the application.
   - The frontend framework in React provides visually appealing and navigable UI for users to complete the CRUD operations easily.

3. If this design was to scale to 1,000,000 devices in the field, are there things you would do differently?
   - For scaling to 1,000,000 devices, the server should be capable of handling the large number of requests and the database must be able to store and process vast amounts of data.
   - Scaling Vertically would only be viable for smaller number of devices, I would approach this by horizontal hybrid scaling. Adding more servers to balance the load and increase server availability.
   - As python Flask server is highly modularized in the architecture for this project, adding new servers will be easier and more efficient.
   - The backend MongoDB database is non-relational and is designed to store compressed JSON data thereby saving storage space and is also highly available as a live cloud database.

4. Time Spent on Project:
The major time spent on the project can be depicted as follows:

Stage	Duration (in hrs)
Initial Environment and IDE Setup	0.5
Database Setup and Design	1.0
Front-end Framework design	5.5
Back-end Framework design	6.5
Connecting Frontend and Backend	2.5
Debugging, testing, CSS, and other changes	1.5
Containerization and deployment	4.5
Total Hours Spent	**22.0**

5. Future work and Deployment:
Since User Management and authentication were not a requirement, future work would involve adding user profile management, authentication, and access control. Securing the application via encryption methods like AES and Certificate management for authorization will be a key requirement.

For deploying the web application, I would go about it in two different ways. If a **cloud**-based deployment solution is required for scaling the application I would go with a PaaS software like Heroku along with AWS. If **security** is of higher priority along with using internal lab resources for the company, I would go with a linux based Apache server deployment solution.

5. GitHub Links for the Web application:

Project Repository : https://github.com/typemaster007/acme_iot.git

6. Platforms:
This application has been tested on the following systems:

1. Amazon AWS EC2 Cluster (Ubuntu 20)
2. Ubuntu 18.04 in Oracle Virtual Box
3. Linux Fedora System

7. Screenshots:

Not secure | ec2-18-217-233-37.us-east-2.compute.amazonaws.com:3000

ACME Sensor Data Home IOT Sensor Simulation Prototype Dashboard

ACME Sensor data visualization

This application allows users to Simulate incoming IOT data as well as view the live dashboard for device status .

Are you here to submit Sensor data?

Not secure | ec2-18-217-233-37.us-east-2.compute.amazonaws.com:3000/simulator

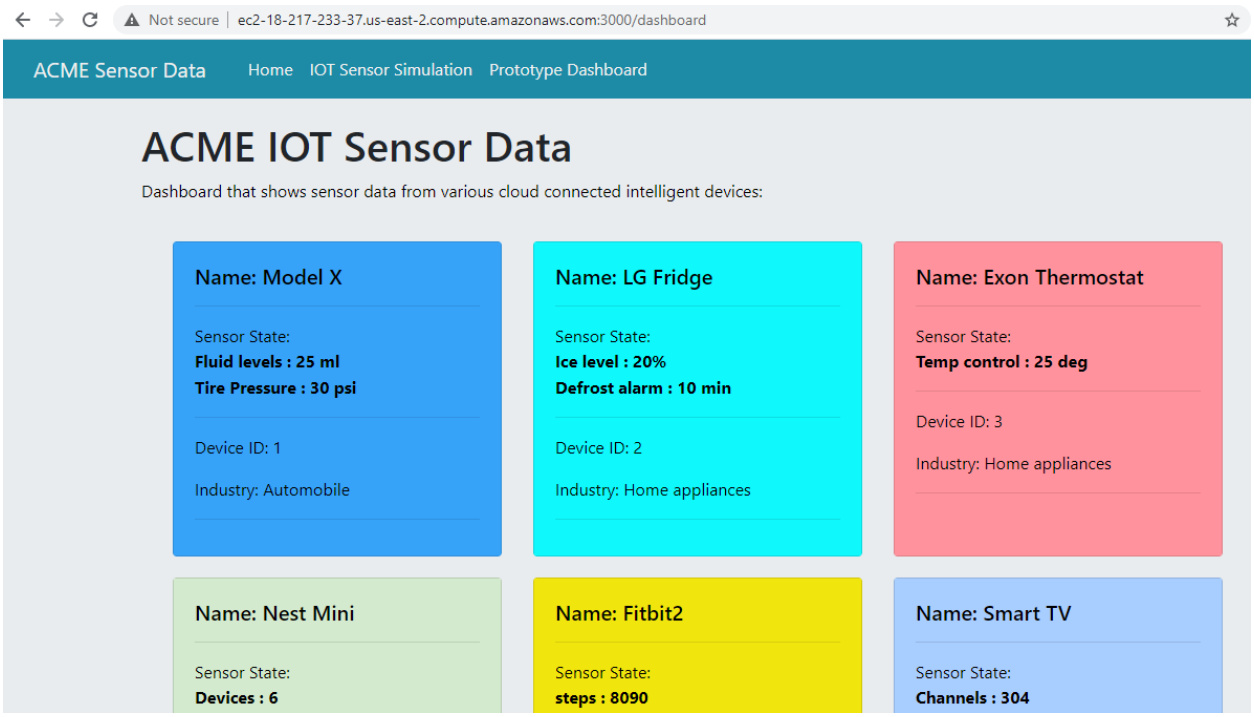
ACME Sensor Data Home IOT Sensor Simulation Prototype Dashboard

ACME Device Simulator

Welcome to ACME Sensor Data Simulator!

You can create a new sensor device, update it and delete it.

Create a new sensor Device Edit Device Delete Device


Not secure | ec2-18-217-233-37.us-east-2.compute.amazonaws.com:3000/dashboard
ACME Sensor Data
Home
IOT Sensor Simulation
Prototype Dashboard
ACME IOT Sensor Data
Dashboard that shows sensor data from various cloud connected intelligent devices:
Name: Model X
Sensor State:
Fluid levels : 25 ml
Tire Pressure : 30 psi
Device ID: 1
Industry: Automobile
Name: LG Fridge
Sensor State:
Ice level : 20%
Defrost alarm : 10 min
Device ID: 2
Industry: Home appliances
Name: Exon Thermostat
Sensor State:
Temp control : 25 deg
Device ID: 3
Industry: Home appliances
Name: Nest Mini
Sensor State:
Devices : 6
Name: Fitbit2
Sensor State:
steps : 8090
Name: Smart TV
Sensor State:
Channels : 304