IoT Elderly Care Solution

Design Document

DEC19-18

Optical Solutions
Daji Qiao

Robert Guetzlaff - Hardware and Software
Tyler Borchert - Hardware and Testing Engineer
Siyuan Zeng-Test and Logic Server
Nidhi Dalvi- Meeting facilitator and hardware
Jared Griffin - Web Application Engineer/GitLab Administrator

sddec19-18@iastate.edu
http://sddec19-18.sd.ece.iastate.edu

Revised: 2019-04-30

`

Table of Contents

`

List of figures/tables/symbols/definitions (This should be the similar to the project plan)

Pictures:

1 Introduction

1.1 Acknowledgement

Andrew Guillemette, the CEO of Optical Operations in Ames and our client, has helped significantly in guiding our group through this project. He has helped get us resources and provide us ideas when necessary. Daji Qiao, a professor in the ECpE department focusing in network engineering and our faculty advisor, has also given us welcomed advice on the ideas that we were presenting.

1.2 Problem and Project Statement

Today, there is a growing number of senior citizens, each wanting to maintain their independence. However, loved ones also want to ensure that they are in the best of health. This leads to a problem where senior citizens either end up in a retirement home, giving up their independence, or they have an at-home nurse, which can be expensive.

The goal of this project is to make improvements and advancements of a previous capstone project. This project is intended for elderly individuals who wish to have their health monitored through the use of sensors place around their place of residence. This project is driven by the need for elderly care today. There is a large number of elders who live alone, so we need to monitor their life quality and health status to prevent health issues from arising. This is important for us to do, but having a personal nurse is expensive, so we hope our project can provide a cheaper and more efficient way to achieve this purpose.

Our group's specific goal is to fix issues that arose in a previous solution, work on logic to detect when the elderly resident missed a meal based on data we collect, create and install a wireless sensor unit to replace the existing wired sensors that has a battery life of 1-2 years, and display sensor and logic data through a web application. The objective of this project is to determine if the data collected from a series of sensors can help predict or see early signs of health deterioration. Therefore, our group is trying to setup a monitoring environment at the elderly resident's residence to monitor their daily activity. The activity records will be passed to our logic server via a hub. Then the logic server will handle this data and determine the target's life quality. The logic server will then notify the web application about the elderly residents current life quality and health status.

1.3 Operational Environment

The sensing system will be housed within the elderly user's residence. The target audience of the product is elderly individuals who may not be interested in having visible sensors and wiring. To accommodate this, our sensors will need to blend in as much as possible with the environment and communicate wirelessly.

The database storing the user's data, the server acting as an interface to this database, the logic server, and the web application will be hosted in the Amazon Web Services (AWS) cloud environment. These components will require some level of security provided. However, as this is a prototype project, the standard protections AWS provides to systems hosted on its cloud will be sufficient.

1.4 Intended Users and Uses

There are two intended users for this project: one being the elderly individual who interacts with the system in the kitchen, and the other being family members or another concerned party monitoring the behavior of the elderly individual.

The elderly individual will be a passive user of the system, interacting with their residence as normal, and we will be able to record what they do. This system will need to be as unobtrusive as possible as to not change the habits of this user. The system should be able to accurately monitor the eating habits of this user to predict if normal eating habits have been disrupted.

The other users of the system will interact with the web application. The user interface of the web application should be simple enough that multiple people could be tracked with little effort by one person but also provide the level of detail required to know what exactly triggered a health alert. This user could be a relative or an employee of a elderly living center.

1.5 Assumptions and Limitations

Assumptions:

- Sensors should be able to track activities accurately.
- The data recorded by the sensors should be passed to server through an HTTP connection.
- The end product should be 1 sensor system per elderly individual.
- The web application should not have a maximum number of users.
- The product will not be used outside of the United States.
- Security will not be made a priority due to this being a prototype.

Limitations:

- The battery life for the sensors will last 1-2 years.
- The sensor will be wireless.
- The whole system should be indoor and requires Wi-Fi environment.
- The system must operate at 120 or 220 volts and 50 or 60 Hertz (the most common household voltages worldwide).

1.6 Expected End Product and Deliverables

The final product will consist of a system of sensors that will be placed throughout an elderly person dwelling that will report information back to a server. This server will process the data to determine if this person is maintaining their regular routine. The server will analyze the sensor input to monitor information such as water consumption as well as whether the target person has eaten. This data will be visible on a website viewable by the caretakers or loved ones of the target.

What we will provide to the client is a set of wireless sensors that will last at least one year on a battery, a hub that will receive the data from the sensors and pass it off to the server, a server application that will process data from the sensors and identify changes in the targets routine, and a web application that will allow interested parties to see the conclusions generated by the logic server alongside the raw sensor data.

`

2. Specifications and Analysis

2.1 Use case diagram

`

2.2 Proposed Design

We have three distinct design areas we will be working on, each of them will work together but be designed separately. The areas are the sensors, server logic, and web application.

Sensor

The first portion of this project is fixing a previous implementation done by the group that proceeded ours. Their implementation has sensors on most drawers and doors in the testing environment. These sensors would go to a Raspberry Pi and send the collected information off to a hub Raspberry Pi. Then this hub will send the information to a EC2 instance on a AWS cloud server. Currently what we are working on is fixing this solution to send off sensible data. This portion involves understanding the current implementation and identifying points of weakness with the current code. This fix needs to produce proper data and run in a timely manner. We currently have an implementation for this and it will be discussed in the following section.

A sub-portion to the sensor part of this project is reimplementing the wired sensor array with wireless sensors. These sensors have to have a battery-life of 1-2 years, be non-invasive, and off the shelf. Currently we are conducting research on what solutions would be best to implement. There are two types of sensors we are looking into. The first solution being looked into is a magnetic reed switch. The current implementation uses a wired version of this. The second solution is a gyroscopic sensor that was suggested by our client. This takes gyroscopic and accelerometer data to produce the same data.

Logic

The logic for this product will exist on an EC2 instance on AWS. By hosting this in the cloud we will maximise our service uptime while reducing expenses incurred. The logic service will look at the data in the database and determine if specified targets have been met. These targets will include eating specific meals and drinking heated drinks. If an expected event is missed, a flag will be raised, and the logic system will notify SDMAY19-36's android app. This flag will also be viewable on the web application. This notification will allow those monitoring the elderly individual to know that the user's standard routine has been disrupted.

To accomplish this, the logic server will need to know what different meals the elderly user may consume and how often these meals may take place outside the kitchen. Each regular meal time will be analysed to determine what sensors have been tripped and if they match with any known meals. If it matches a known meal, we can mark that meal time as complete. If it appears to have been skipped, we would send our notification.

Web Application

The web application will serve as the user's interface with the system. They will be able to monitor the elderly individual's behavior to confirm that there are no irregularities. In the case that the elderly individual were to break from their regular routine, the web application would display this information to the users monitoring the specific elderly individual.

The web application will be built using React, a JavaScript framework for single page web applications (SPA) designed by Facebook. The web application will be hosted from AWS S3, a simple storage service that will serve as a web server for this project. Communication between the web application and other systems will be done through REST APIs built by the team.

`

2.3 Design Analysis

Sensor

Fixing the old implementation was the first major task that needed to be accomplished. To make this fix possible, our client gave us a prototype to test on. This prototype has all of the currently implemented features. We ran the original code and observed what it was doing as we were working with the protype. When running it, we noticed some major issues such as not completing if statements at the correct times and generating unusable data. We decided that it was best to fix this code instead of starting over. For this fix to be considered, it had to reproduce proper data and report it in a timely manner to the server. This fix involved creating a flag and changing the response times of the triggered events. To ensure that this fix was successful, we conducted many test while observing the output. Once we knew that the output was good, we pushed this code to the testing environment.

The wireless implementation of this project has reached the end of the research process. We have chosen a wireless sensor tag solution. It is called th CC265STK SimpleLink BLE TI wireless tag. This has may sensors available and we will primarily use the gyroscope and accelerometer. This will solve many of the issues that we have with the magnetic reed switches. This also makes the future implementation more versatile and non-invasive.

Logic

The database used for storing the sensor data was created by a previous group and has been running for the six months. Our first step was to gain access to this database so we can review the data that was being stored. We have implemented a Php based interface for this data that allows us to currently view the sensors that are triggered for breakfast over the past week. We have also built a tool for viewing sensor data under a given time threshold to identify any erroneous data that is sent to the database. The server side provides a filter to show identified sensor data through user id and/or resident id. We can use this to find a logic for the user's daily routine, and then we can use this logic to determine if a certain activity is taken.

Web App

Up until this point, most of the development on the web application has been focused on getting a development flow setup. Part of this revolved around setting up continuous integration for the web application. We decided to use GitLab's Pipelines for this. However, ETG, the group that hosts the class' GitLab instance, did not have this feature properly configured, and it took a couple of weeks communicating with ETG to get the issue resolved.

Aside from waiting on continuous integration, significant progress was made on bootstrapping the web application. The continuous integration pipeline is properly configured, and we have added client-side routing and a CSS framework. We have also began to mock out pages of the web application, which you can see below.

Figure 2.3.1: Food Consumption Overview - Web Application

Below is an evaluation of the strengths and weaknesses of the solutions we are pursuing::

- Sensors don't produce incorrect data (Fix old solution)

  Strengths:
  - No error data now. All data looks good and ready for use.
  - Data are well managed in our database.

  Weakness:
  - A few extremely small data are caused by the bounce of door. But we code to ignore them so everything good.

- Detect when the resident has missed a meal from data stored on the logic server

  Strengths:
  - The server and sensor hub interface with the same database, making interaction with the data easier.
  - Can response to many requests from the web app.
  - Event category are implemented, meaning that it can now decide what activities happened through the sensor data.

  Weakness
  - Still needs more data to implement the event decisionmaking to ensure the decisions are accurate.

- Create and install wireless sensors

  Strengths:
  - Magnetic Sensor: Code is already written; Easy to set up; More solutions are available on the market.

`

- Bump Switch Sensor: No possible magnetic interference; Little change in the code to implement.
- Gyro Sensor: Easy installation; Much more versatile

Weakness:

- Magnetic Sensor: Has issues with magnetic interference; Depending on sensor, a hub to interface to.
- Bump Switch Sensor: Has a motorized element the can break or degrade over time; More room for error with installation location.
- Gyro Sensor: Have to rewrite a lot of the code; Calibration depending on location; Possible gyro drifts that can mess with calibration.

Display resident data through a web application.

Strengths:

- Clearly displays the daily events with graphs.
- Can visibly confirm that the sensing system is operating as expected easily.
    - Weakness:
- Adding a column diagram for drawers may help show the daily logs more clear.
- Is not yet feature complete.

3 Testing and Implementation

3.1 Interface Specifications

Our project consists of three different components communicating with each other over the internet. Our testing environment will communicate with the cloud server using HTTP POST requests to transmit four pieces of information relevant to the sensor: the time the sensor was triggered, the id of that sensor, the resident id for that apartment, and the duration that sensor was triggered. This data is received by the server and stored in the database to be processed at a later time. The web application will then use HTTP GET requests to pull both raw sensor data as well as our processed events to be displayed.

System wide integration testing will be done through the web application verifying the flow of the data through the system. Because each of our sections are independent and only pass completed processed data, this testing should be sufficient.

3.2 Hardware and software

Postman

Postman [1] is an application for interacting with HTTP APIs. We use it to test our server to verify the backend APIs works as expected. The strength of this is we do not have to use our frontend to test the server.

RealVNC

RealVNC [2] is a client and server programs that allow users to remote into their device. It enables us to remotely access and securely monitor and change code on the sensor Pi's and hub.

Jest

Jest [3] is a unit testing framework built by the software engineers at Facebook for testing JavaScript projects. It has deep integration to React, the JavaScript framework we are using for the web application, and also provides a test coverage tool for confirming that the web application is completely covered by tests.

GitLab Pipelines

GitLab Pipelines [4] is a continuous integration solution that allows us to build our web application. Along with that, we are able to run the web application's test suite prior to building the application. This allows us a final check that the web application is ready to deploy.

Apache

Apache [5] is an HTTP server that interprets PHP code and serves HTML pages to users. Our current database interface is using this server.

React

React [6] is a JavaScript library produced by Facebook for designing single page web applications. This is the framework that serves as the backbone of our web application.

Spring Boot

Spring Boot [7] is a application framework for java platform. It helps to build, develop and manage a java based project.

3.3 Functional Testing

Below we explain the different methods we use to test the individual parts of the system along with the whole integrated system.

Unit Testing:

- Sensors produces expected data
    - Access Raspberry Pi with a monitor to see if data received are correct. If not, then check the code.
    - Check the database frequently to ensure data stored is the same as the data sent by the hub.

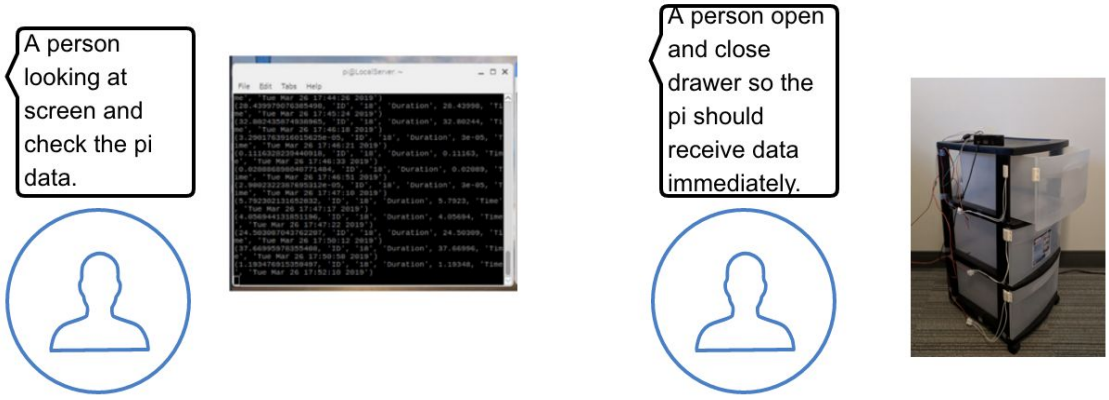

Figure 3.2.1: Unit test for sensor

- Detect when the resident has missed a meal from data stored on the logic server
    - Using Postman to check the server is running correctly.
    - Check the database to ensure data is up to date.
    - If all the above is fine, check the server code.

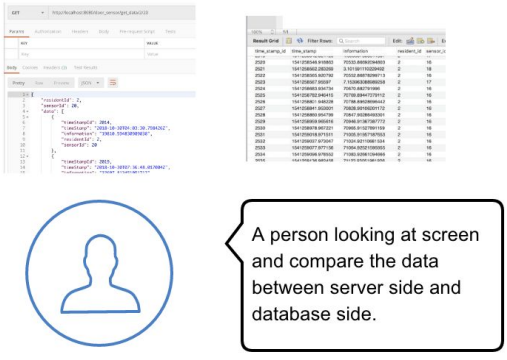

Figure 3.2.2: Unit test for the server.

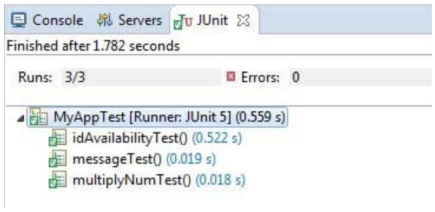

Figure 3.2.3: Junit test for the single class.

- Display sensor event status through a web app.
    - Using a unit testing suite that covers the features in the web application

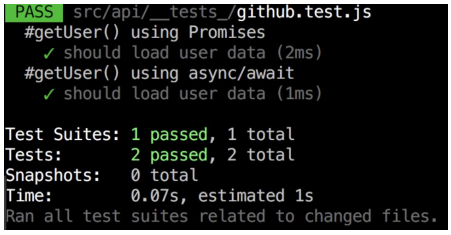

Figure 3.2.4: Jest test for the web app js class test.

Integration Testing:

- Sensing System and Database: Compare the data received in Raspberry Pi to the data stored in the database and review for discrepancies.

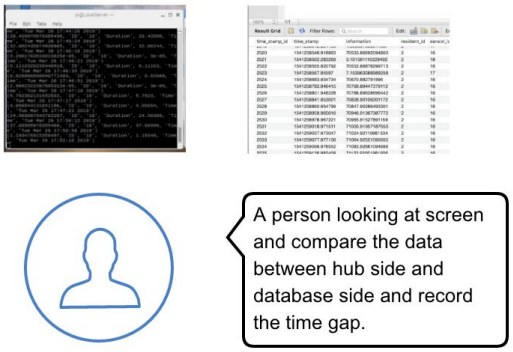

Figure 3.2.5: Sensor - Database integration test.

- Logic Server & Web Application : Compare the web application results and graph information to the logic algorithm we build and verify the results.

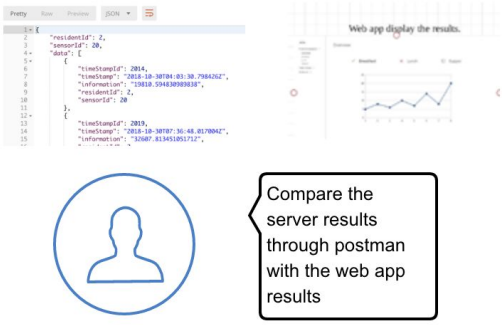

Figure 3.2.6: Server - Web App integration test.

- Web Application to Database Server: Interface with the database server to setup test data for the integration tests. Then, use a web driver to use the web application to go through the user's flow, confirming that the web application updates correctly.

System Testing:

To check that the entire system is functional, we can check the web application for the elderly individual's behavior before calling them to confirm their previous actions.

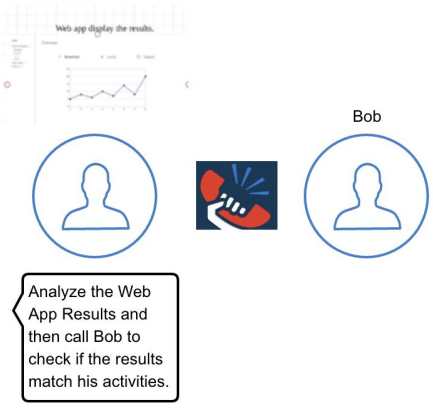

Figure 3.2.7: While system integration test.

3.4 Non-Functional Testing

Timely manner: We will need to make sure data is being sent in a timely fashion along with making sure all data being sent over the wireless connection is not being lost or manipulated. To test this we will have to add a timer to count hold long it takes for communication to read from one end to another.

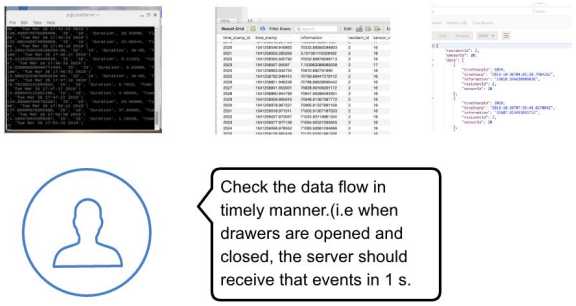

Figure 3.3.1: Non-Functional test: timely.

Performance: The performance of the devices we are implementing is not a huge factor in our project. We would like the devices to be working correctly before we focus on how we can send data faster. However, with that being said, some of the devices are slow. To test this, we will need to run some kind of analysis on the code we write to determine it's execution time.

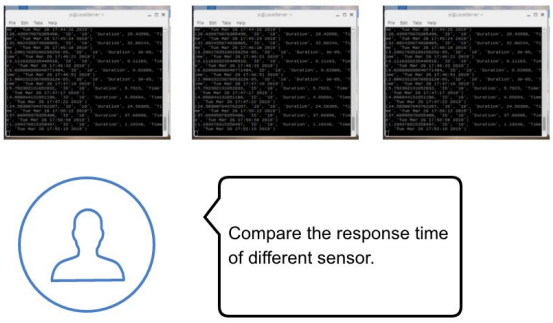

Figure 3.3.2: Non-Functional test: performance.

3.5 Process

Sensors

The process for testing the sensors is observing the output of the code while manipulating the prototype. This is accomplished by working outside of the actual test environment with the prototype only. This will ensure that the test environment does not receive faulty code.

Logic

The process for testing the logic will be to generate a set of test data and pass it into the server. This will be done through Postman. Postman will then watch for any returned data and verify that the correct response is being generated by the server.

Web Application

The process of executing tests on the web application is to execute the testing suite each time before new code is merged into the web application's master branch. This is accomplished through a GitLab Pipeline executing each respective testing suite on a GitLab runner. If there is some kind of failure in any of the testing suites, the deployment of the web application will fail, and every member of the team will be notified.

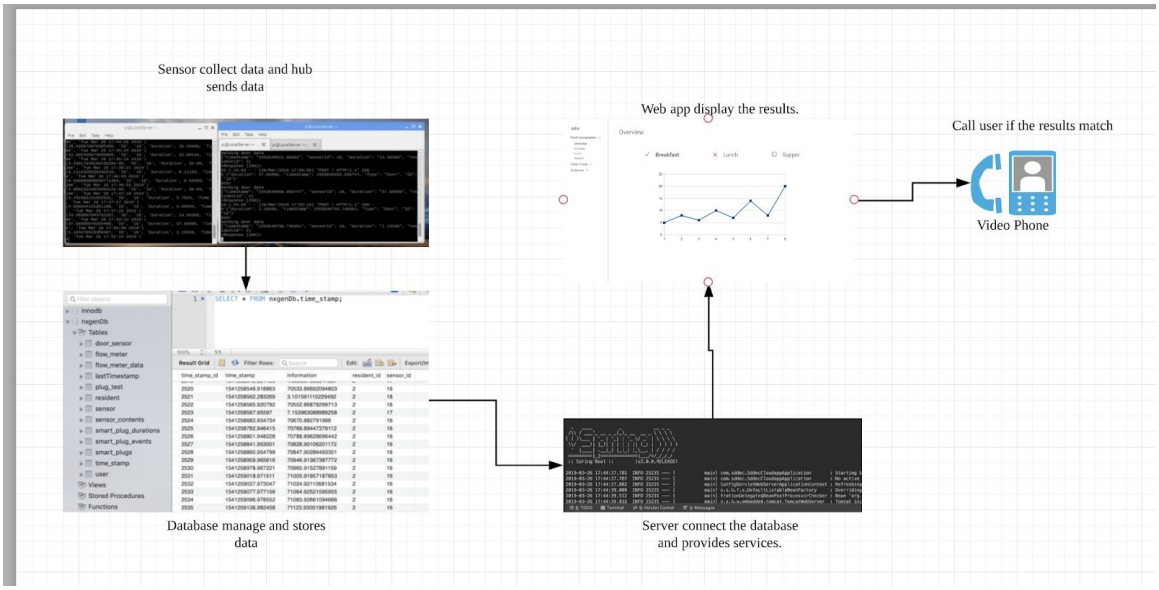

Figure 3.4.1: Flow Diagram for Data Transmission and Consumption

3.6 Results

So far from the beginning of the project, we have achieved the following:

- Sensors are collecting correct data and the Hub sends the data to server and database in a timely manner. During this phase, we learned new knowledge about the embedded system such as Raspberry Pi OS, Pi's structure, and sensor's code.

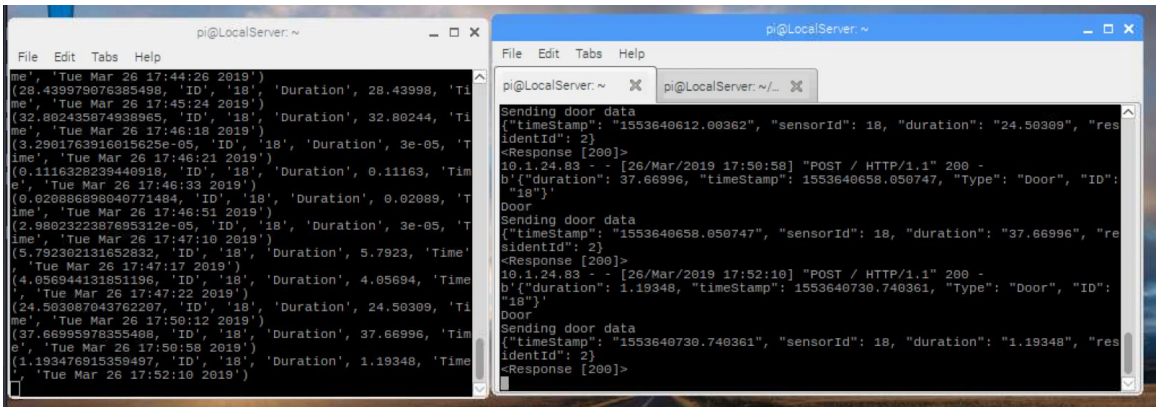

Figure 3.5.1: A Raspberry Pi Running at our User's Residence

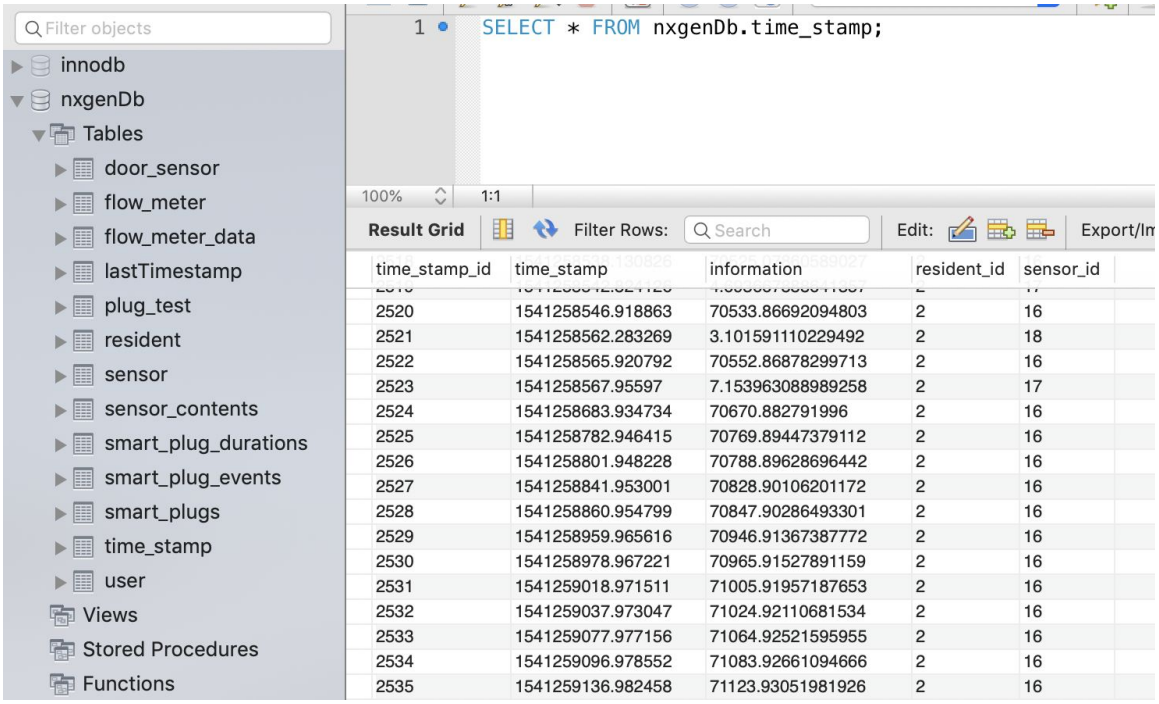

Figure 3.5.2: Data Stored in our Database

1553040000

Number	Entry ID	Time	Durration	Resident	Sensor
0	84686	Mon Mar 18 08:27:42 2019	3.17306	Robert Kern	Left refrigerator door
1	84685	Mon Mar 18 08:27:10 2019	3.15835	Robert Kern	Left refrigerator door
2	84684	Mon Mar 18 08:26:58 2019	135.05425	Robert Kern	Pantry
3	84683	Mon Mar 18 08:26:04 2019	3.7172	Robert Kern	Glasses, Mugs, Cups, Bowls, Plates
4	84682	Mon Mar 18 08:24:30 2019	4.51163	Robert Kern	Silverware
5	84681	Mon Mar 18 08:23:26 2019	9.64607	Robert Kern	Pantry
6	84680	Mon Mar 18 08:21:04 2019	6.04711	Robert Kern	Left refrigerator door
7	84679	Mon Mar 18 08:18:44 2019	12.46453	Robert Kern	Pantry

1553040000

Number	Entry ID	Time	Durration	Resident	Sensor
0	84665	Sun Mar 17 08:09:02 2019	3.11072	Robert Kern	Silverware
1	84664	Sun Mar 17 08:08:32 2019	5.46287	Robert Kern	Silverware
2	84663	Sun Mar 17 08:05:26 2019	4.93539	Robert Kern	Silverware
3	84662	Sun Mar 17 08:04:49 2019	4.47763	Robert Kern	Glasses, Mugs, Cups, Bowls, Plates
4	84661	Sun Mar 17 08:02:20 2019	14.75616	Robert Kern	Pantry
5	84660	Sun Mar 17 08:00:42 2019	16.56169	Robert Kern	ceramic bowls, Fancy glasses, Cooking spray

1553040000

Number	Entry ID	Time	Durration	Resident	Sensor
0	84642	Sat Mar 16 08:31:35 2019	7.19455	Robert Kern	Pantry
1	84641	Sat Mar 16 05:59:52 2019	4.90231	Robert Kern	Silverware
2	84640	Sat Mar 16 05:59:43 2019	3.75242	Robert Kern	Left refrigerator door
3	84639	Sat Mar 16 05:58:56 2019	3.66861	Robert Kern	Left refrigerator door
4	84638	Sat Mar 16 05:58:44 2019	67.38551	Robert Kern	Pantry
5	84637	Sat Mar 16 05:56:42 2019	4.18911	Robert Kern	Glasses, Mugs, Cups, Bowls, Plates
6	84636	Sat Mar 16 05:52:39 2019	8.26387	Robert Kern	Freezer door
7	84635	Sat Mar 16 05:51:38 2019	10.93095	Robert Kern	Pantry

1553040000

Number	Entry ID	Time	Durration	Resident	Sensor
0	84632	Fri Mar 15 07:38:54 2019	6.1491	Robert Kern	Pantry
1	84631	Fri Mar 15 07:35:18 2019	847.06622	Robert Kern	Pantry
2	84630	Fri Mar 15 05:43:16 2019	13.50819	Robert Kern	Silverware
3	84629	Fri Mar 15 05:42:48 2019	4.37977	Robert Kern	Left refrigerator door
4	84628	Fri Mar 15 05:41:39 2019	3.85422	Robert Kern	Left refrigerator door
5	84627	Fri Mar 15 05:40:43 2019	75.99464	Robert Kern	Pantry
6	84626	Fri Mar 15 05:39:06 2019	5.56643	Robert Kern	Glasses, Mugs, Cups, Bowls, Plates

1553040000

3.5.3: Sensor Data Monitoring Tool

- Logic Server provides a sensor(s) filter service to the frontend. In this phase, we learned new experience on spring boot framework and logic algorithm.

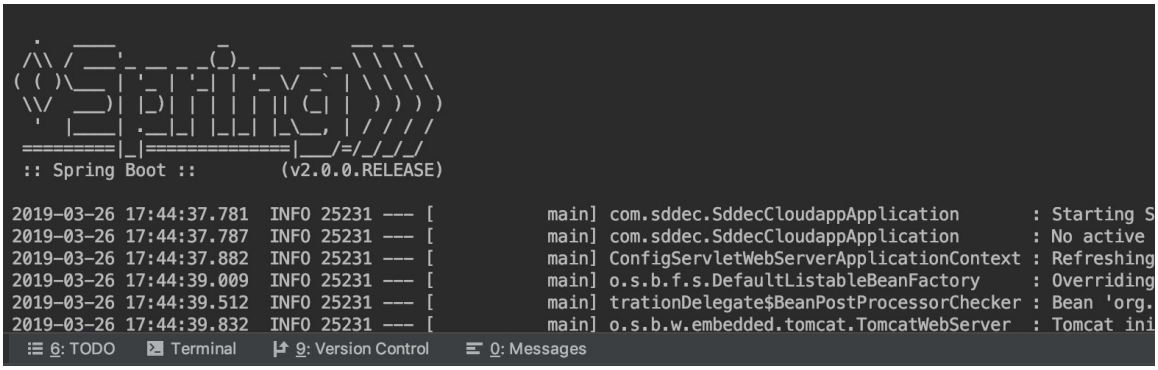

Figure 3.6.4: The Behavioral Logic Server Running

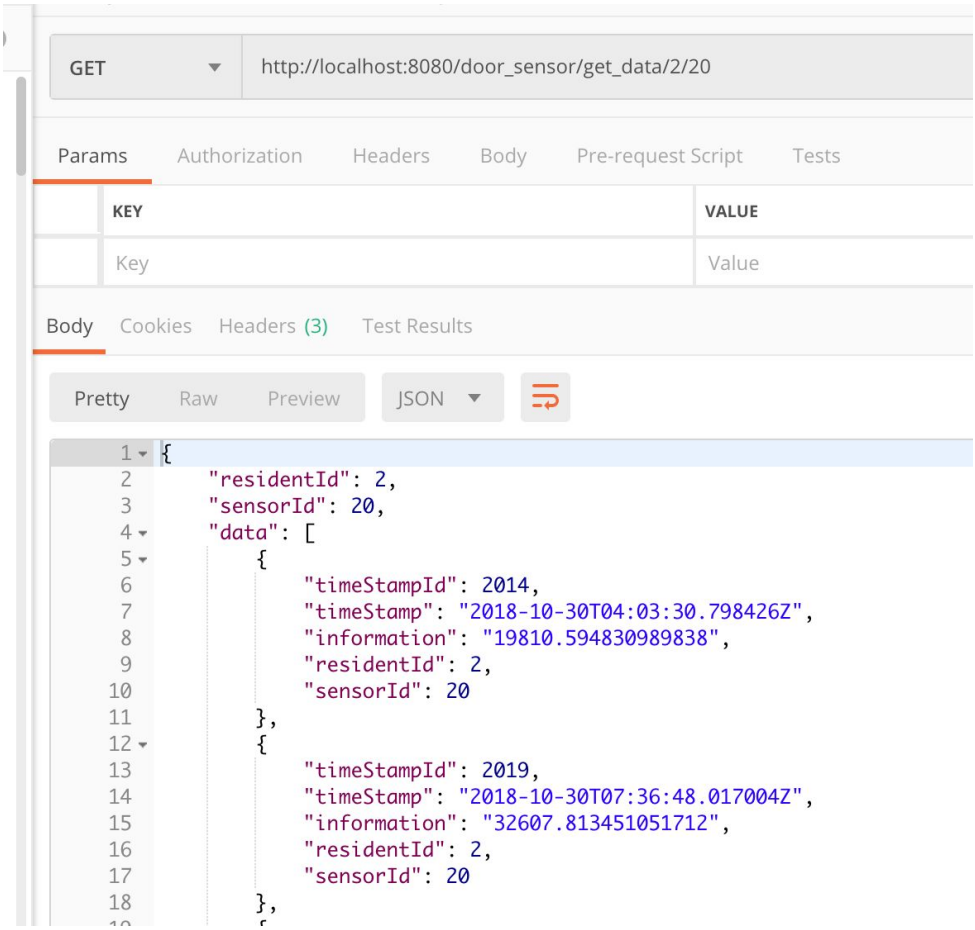

Figure: 3.6.5: A Response from Retrieving Drawer Sensor Data

- The web application is not yet in a state where it can display the elderly user's behavior due to delays stemming from issues with GitLab Pipelines.

`

Although we still have features not achieved yet and things not completed yet, we don't want to say them failures because we still have time to work on them and we believe we can figure them out. They things we need to continually work on are (As the template's introduction indicates: failure and future plan.):

- Find a wireless solution to replace the current wired sensors.
- Keep implementing more services on the logic server for our web application, such as more accurate logic to determine daily activity.
- Adding onto the web application so it can display more information and find a clear approach to display the results.

4 Closing Material

4.1 Conclusion

Our project aims to provide a cheaper and efficient solution than getting an in-home nurse for elder care problems. So far we figured out the sensor data problems the previous group struggled with, data delivery problems, data handling and analysis problems, and web application problems. For the elder care problems, we found two possible solutions: getting an in-home nurse, and the project design we are doing now. Our project is much cheaper than hiring an in-home nurse while it is also being more efficient. Therefore, we will keep working and implement our design with the best efforts and honor.

5. References

[1] https://www.getpostman.com/products
[2] https://www.realvnc.com/en/raspberrypi/
[3] https://jestjs.io/
[4] https://docs.gitlab.com/ee/ci/
[5] https://httpd.apache.org/
[6] https://reactjs.org/
[7] https://spring.io/projects/spring-boot