To Online Shop, or To Not Online Shop

DESIGN DOCUMENT

Team Number: sdmay20-19
Adviser/Client: Goce Trajcevski

Team Members:
Amiah Gooding – Electrical Engineer
Matthew Martin – Report Manager/Scrum Master
Max Minard – Software Manager
Smruthi Sandhanam – Meeting Manager
Travis Stanger – Test Engineer
Yana Aleksandrova – Meeting Facilitator

Team email: sdmay20-19@iastate.edu
Team Website: http://sdmay20-19.sd.ece.iastate.edu

Revised: April 26, 2020 Final Revision

Table of Contents

List of Figures:

List of Tables:

List of Definitions:

- ETG - Electronics and Technology Group
- IEEE – Institute of Electrical and Electronic Engineers
- IoT – Internet of Things
- RFID – Radio Frequency Identification
- AWS – Amazon Web Services
- iOS – Operational system for Apple devices
- API – Application programming interface

Executive Summary

Abstract

To Online Shop, or To Not Online Shop is an end-to-end, IoT solution for efficient and effective management of the individual and household-based shopping experiences. The solution will: (1) monitor status of items in pantry or cabinet (2) generate list of items "to buy" from sensor (3) use Geolocation to detect user's proximity to a store location (4) notify users when items in the "to buy" list are offered in near-by store for reasonable prices. This project stems from the recent increase from online shopping compared to in-store shopping and hopes to provide an optimized shopping experience for users. This project is scheduled to be completed by May 2020 and is advised by Professor Goce Trajcevski.

Design Practices Used

- Agile software development
- Commenting code
- Following all safety protocols

Summary of Requirements

- Generation of data: The state of items in a cupboard
- Signal generator: Trigger to update in data
- Geolocation: Showing stores nearby and deals
- Mobile application: Receives notifications
- A signal from interface that an item has been purchased

Applicable Courses from Iowa State University Curriculum

- Computer Science 227: Object-oriented Programming
- Computer Science 228: Introduction to Data Structures
- Computer Science 309: Software Development Practices
- Computer Science 311: Introduction to the Design/Analysis of Algorithms
- Computer Engineering 288: Embedded Systems I
- Electrical Engineering 230: Electronic Circuits and Systems
- Electrical Engineering 330: Integrated Electronics
- Electrical Engineering 333: Electronic Systems Design

New Skills/Knowledge acquired that was not taught in courses

- Amazon Web Services
- Raspberry Pi 3 Model B
- Additional new skills/knowledge to be determined

1 Introduction

This section outlines the project constraints and goals as well as the expected timelines for the project.

1.1 Acknowledgement

The To Online Shop, or To Not Online Shop team would like to thank the Iowa State University College of Electrical and Computer Engineering for providing the student team a professional experience, resources, and consultation with experts. We would also like to thank Professor Goce Trajcevski for meeting with us weekly and guiding us through the development process of our product. The team appreciates the University's Electronics and Technology Group's (ETG) availability for providing hardware and server components for the project.

1.2 Problem and Project Statement

The goal of the project was to design a full-stack IoT-based solution that will help to strike a balance between online shopping and offline (in-store) shopping experiences for the consumers. The goal is to develop a project that will:

a. Monitor the status of the items in a shelf or cabinet
b. Generate a list of items "to buy"
c. Detecting when a customer is in a proximity of an actual store (e.g., Target, Walmart, etc.)
d. Automatically send notifications that certain items from the "to buy" list that are scheduled for delivery, could be obtained in the near-by store (and with discount coupons).

There are two main components for this project. There is the home IoT hardware component that retrieves the data regarding the status of the items from the cabinet or pantry. The data received gets populated into a database with the itemized pantry contents. The other component is the mobile application which retrieves the items that need to be bought from the home device. The algorithms determine if it is cheaper to buy an item online or in-store. Geolocation is used to find the best deal for a product.

Our solution was to use a barcode scanner along with a raspberry pi that can help to monitor status of the items located in the pantry or cabinet. A signal will trigger the sensors such that status of the items will be stored using an AWS database. Our mobile application will analyze the data and then generate a list containing the items that need to be bought. We wrote algorithms to determine if it's cheaper to buy items online or in-store. We used geolocation to figure out where the best deal for buying the products are.

1.3 Operational Environment

The sensors and microcontrollers used to monitor the status of the items is stored in a dry pantry/cabinet area with a Wi-Fi connection. The current device is a microcontroller-connected "Smart-Bin" that is expected to fit securely on the top of the shelf. All setup aside from the device is done through mobile application.

1.4 Requirements

Constraints:

- Budget of 200 dollars.
- Only viewing a few stores within a 0.5-mile radius.
- A user does not want to buy an item in person if it can be found for cheaper online.

Environmental Constraints

- All aspects of the project are tested in a simulated environment.

1.5 Intended Users and Uses

The main user for the project is anyone who is looking to save money and shop more efficiently. The user will initially have to set up the hardware in their cabinet or pantry. Afterwards, they will primarily interact with the product using the mobile application. The user can monitor the status of items in their pantry or cabinet and can generate a list of items that they need to buy. The app will, furthermore, analyze the data and figure out where the best places to buy the list are and whether it is cheaper to buy online or at the store.

1.6 Assumptions and Limitations

Assumptions:

- Pantry or cabinet monitoring devices have a power source
- Pantry or cabinet monitoring device can connect to internet
- Pantry or cabinet monitoring device is not at risk of water damage
- The customer properly sets up hardware

Limitations:

- Accuracy of automatic orders is completely based on the accuracy of the sensors to detect the status of an item (Ex. "Empty or full")
- The cost to produce the end product shall not exceed two hundred dollars
- The product will only be able to view the stores within a 0.5-mile radius

1.7 Expected End Product and Deliverables

The final product will be divided into deliverables for the first semester and the second semester. The final product deliverable will be at the end of second semester.

Semester 1:

1. Formalize the scope of the project and identify main components (September 20th, 2019)
    - In this deliverable we met with the customer and came up with our project requirements and finalized the components for our product. This is the initial planning phase.
2. Complete the survey of relevant literature and "off the shelf" products (October 10th, 2019)
    - In this deliverable we are decided what products to implement in our project. We also figured out how we plan to develop the mobile application is also important.
3. Decide upon use-case scenarios and main approaches to be investigated as "candidate-solutions" (November 1st, 2019)
    - In this deliverable we decided the use cases and approaches that we implemented in the product.
4. Converge on the design, and provide implementation(s) of some basic functionality (November 30th, 2019)
    - Complete the initial design for the product and start some basic functionality. This included work on some core features to show initial development.

Semester 2:

1. Revisit the design decisions (January 27th, 2020)
    - Continued from the development of the previous features and work on assigned tasks. Demo completed features to client and made necessary improvements

2. Define and start with the individual components testing (February 28th, 2020)
    - Based on completed components, made sure that individual parts meet all requirements are functioning completely.
3. Have working demo completed (March 9th, 2020)
    - We were accepted to PerCom IEEE International Conference on Pervasive Computing and Communication in Austin. TX.
    - Due to acceptance to the conference our timeline was sped up
    - Complete the end-to-end integration of all components and fine-tune anything was needed. At this point the project was completed and ready to present.
4. Complete necessary testing and integration before conference along with conference materials (March 22nd, 2020)
    - Completed all testing
    - Finalized demos and conference poster, presentation and deliverables
5. PerCom Conference (March 22nd-27th)
    - Selected team members presented at conference and attended virtually due to circumstances surrounding COVID-19.
6. Complete "dry-runs" of the demo and finalize the deliverable-version (April 26th, 2020)
    - Practiced the presentation and prepare demonstration for faculty and staff.

2. Proposed Approach, Specifications and Analysis

This section outlines the specifications and analysis of what was implemented. It also discusses observations on the proposed solution as well as giving justification, with use cases and requirements.

2.1 Constraints Considerations

To ensure that the necessary requirements are met, we have taken into consideration the following constraints. These constraints will include relevant non-functional requirements and standards to follow, such as IEEE standards.

2.1.1 Non-Functional Requirements

Availability: The system inventory is available to update automatically during 12 hours of the day and sends notifications whenever a user is near a store.

Data Integrity: The sensors provide accurate and consistent data, so the inventory is monitored correctly. A barcode sensor is employed to add items to pantry and determine when it is empty.

Deployment: The sensor network is easily installed in the pantry cupboard. For scalability we will make sure that this network is modular so it could be employed in other areas.

Resilience: The sensor network is in an environment of constant change with objects coming in and out of the cupboard. In order to protect the sensor, it is in a protected environment to avoid damage of the system and to avoid constant direct sunlight.

Scalability: We are leaving room for scalability in our project by making our design modular. That will leave the option of supporting different products in different areas (i.e. other pantry cupboards and more products).

Usability: Usage of the product should be intuitive as well as initial setup, with clear steps for the user to take for daily accessibility.

2.1.2 Engineering Standards and Design Practices

This project will abide to the following standards:

IEEE 1028-1997 - IEEE Standard for Software Reviews (References, IEEE Standards)

IEEE 802.11 - Wi-Fi between ESP8266 and Raspberry Pi (References, IEEE Standards)

These standards were chosen as our project is heavily software based so software reviews will be necessary. Our solution also involves communication of information between a local Wi-Fi and our own database, where the development has to follow appropriate networking standards.

IEEE 1532 In-System Configuration of Programmable Devices (References, IEEE Standards) is essentials since our product is an IoT technology involving a massive network of monitoring devices that record and communicate information to each other as well as an app component. This involves and initial configuration that needs to be implemented effectively and securely.

As far as design standards we follow the Agile software development model, working on two week sprints with constant iterations and feedback from our client. We also made sure to comment code for readability and usability, so different team members could work on different parts of the project easily. We made sure to follow all safety protocols set forward for us as well.

2.2 Proposed Design

The solution design has two main components: The Home IoT sensor network and the software mobile application. A full end-to-end behavior model can be seen in **Figure 1**, where further details are illustrated as an introduction to the overview of the system. We will come back to this model after introducing the two main components of **Figure 2**. The sensor on the IoT side satisfies functional and non-functional requirements for the monitoring of grocery inventory in the kitchen cupboard. The application analytics satisfy requirements of geolocation and optimization, using data from the sensor network.

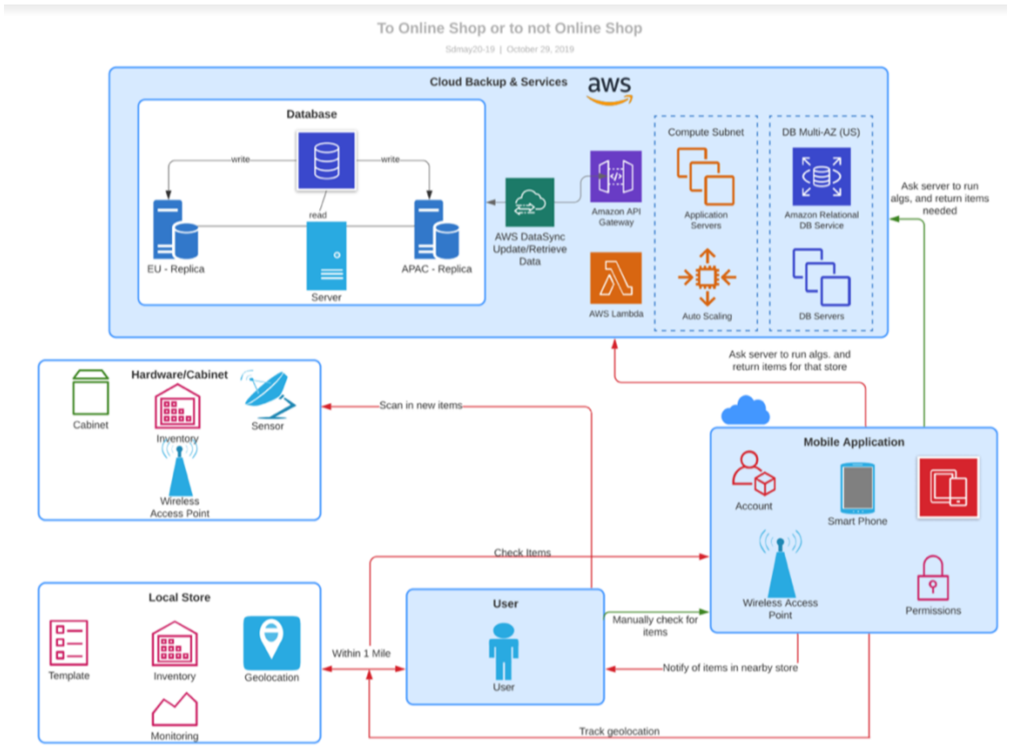

Figure 1: End-to-End Behavior

2.2.1 Functional Requirements – Home IoT Sensor Network

The sensory device employs sensors to monitor inventory levels of a single product. With scalability the client will be able to have multiple devices in order to monitor inventory of different cupboards. The sensor will receives data and sends communication to the database on the back-end. We implemented a barcode sensor that will communicate with AWS and our iOS application.

The system is a master-slave system, with the master component being the Raspberry Pi and the slave component being the barcode sensor. When the Raspberry Pi has received data it is directly sent it to the database. The sensor to Pi communication is done through a Wi-Fi chip on the Pi.

2.2.2 Functional Requirements – Software Mobile Application

The software is an iOS application. We used Swift to develop our application that interacts with our database. The front-end uses geolocation to send a notifications when an individual is close to a grocery store and notify them of cupboard inventory and shopping options. The back-end performs data processing in helping the user make a shopping decision.

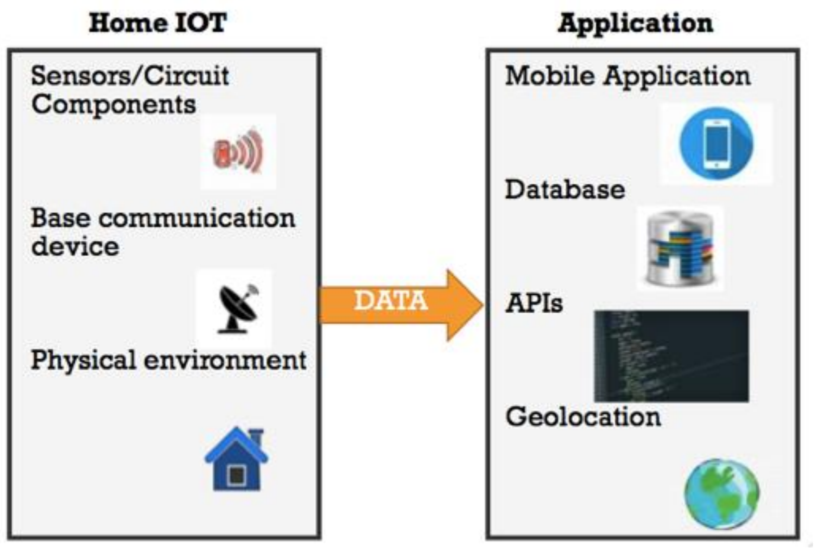

Figure 2: Deployment Diagram of Proposed Design

2.2.3 End-to-End Behavior

(Refer to **Figure 1** for scenarios listed below)

Geolocation Scenario:
The first use-case scenario revolves around the geolocation aspect of our system design. If the user is within a mile of a certain store, the mobile application will pick up on this and send a request to the server to run our algorithms on the needed list of items that the user previously had. If any items return saying that the best prices can be found at that specific store, then it will notify the user of those items. The user can then decide to go to that store and purchase those items. Whichever items they purchase, the user will then head back home and register each one with the barcode scanner before placing it in the cabinet. This sensor will transfer this data to the Raspberry Pi, which will then communicate with the server to update the information on the database.

Manual Request Scenario:
The second use-case scenario provides the option for the user to manually request the best prices for each of their needed items with a push of a button on the mobile application. You could imagine the user is at home and wants to know which locations or which websites to purchase an item. They will hit a button on the mobile app which will send a request to the server to gather the information from the database and conduct the price evaluation algorithms and return all the results. The user will be shown a list of each item and the corresponding online or local store that offers the best deal. The user can then filter through this list and find which items they are looking for. After this, they can purchase the items they need, whether it be online or from a local store, and then once the user has the item, they can scan in on the barcode scanner to register the item and update the database.

Installation:
Before the user can use the product, they must first setup the given hardware in their pantry or cabinet and install the mobile application on their phone. They would also need to scan in all the items they already have so the database knows which items the user initially has after setup. After this, they are free to use the application and service using one of the two use-cases stated above.

Item tracking:
The items that the user needs can be generated through several ways. The user can manually add the item to their list, or an item can be added after being scanned out of the cabinet for being empty. A stretch goal for our solution is to allow the user to monitor their current items to see if any are almost empty and add it to this list.

2.3 Design Analysis

There are two primary component systems for continued development on the project. The Home IoT system is dependent on the stream of data from the sensor network. We ensure that there is continuous transmission through the process. The application is reliant on the sensors and hardware being implemented and functioning properly. Issues in the network directly affect components on the application side and the ability to geolocate as well as compare prices and make decision whether to online shop or not. In **Figure 1** it is clear to see all the components relying on each other for transmission and proper operation.

We are using a barcode sensor and Raspberry Pi as our sensor network. There were several options to consider as we chose Arduino versus Raspberry Pi for its analog-to-digital conversion or Pi Wi-Fi/Bluetooth capabilities. Since we will be gathering a variety of data the Raspberry Pi seemed like the right choice for the implementation. The Pi provides ease of use for development, even though it is a trade off to the Arduino Analog-to-Digital Conversion abilities. The focused data for the project is that from sensors and the Pi has the capabilities of chip add-ons that are a cheap solution to provide the Arduino capabilities. The built in Pi OS helps makes the decision as replacing and altering sensors allows scalability, which is a non-functional requirement of the project.

There were several options to consider as we chose between barcode and RFID as a dependable form of sensing items. RFID provided an ease in entering items as they are easily put into the cabinet as they don't need to be scanned. But added difficulty in tagging/un-tagging each individual item. A barcode scanner will be incorporated as items will be scanned as they enter and exit the pantry, with subtracting or adding to current amount in inventory database.

The front end is built with Swift and XCode for an iOS application, because of the team familiarity with the framework as well as preexisting testing equipment for those frameworks. All of our data is sent to a central database and accessed in our application. For the back-end component we used AWS to monitor database information and make decisions based of specifications and prior information. Since the framework is maintained by Amazon there is continued support as well as plenty of resources. The user of our application will interact with a registration to register the sensory device and have input for the products being monitored as well as the conditions for when the product need to be ordered. Communication between the front-end and back-end will be done via API calls.

2.4 Development Process

We used the Agile developmental method. This way we were allowed to iterate often and get feedback from our client often as well. It is the framework that fits best as we met with our client each week to discuss development. Through these frequent meetings we were able to discuss sprints and next steps in the development process. If there were steps that did not meet our requirements we were able to go back and iterate on certain parts rather than starting over. Since we had a very software heavy project once initial hardware was setup, the flow into agile software development was seamless as two week sprints become the standard work flow. In agile we were able to test during development producing a higher quality product as well as defining and elaborating requirements in time keeping our client active and engaged.

2.5 Design Plan

Our design plan included the five pillars design thinking: empathize, define, ideate, prototype and test. In empathize we met with our client to see what was expected of our final project. In define we established the problem and constraints, and determined the requirements and goals of our project. Ideate is where we brainstormed solutions, analyzed time and costs, assigned roles, and made a plan for going forward. For the prototyping section of our design process we split our development into hardware and software. We developed the two separate parts and tested them on their own and then integrated them back together for final testing. The illustration below shows our overall design process with the ability to go back and iterate to fix or improve designs.

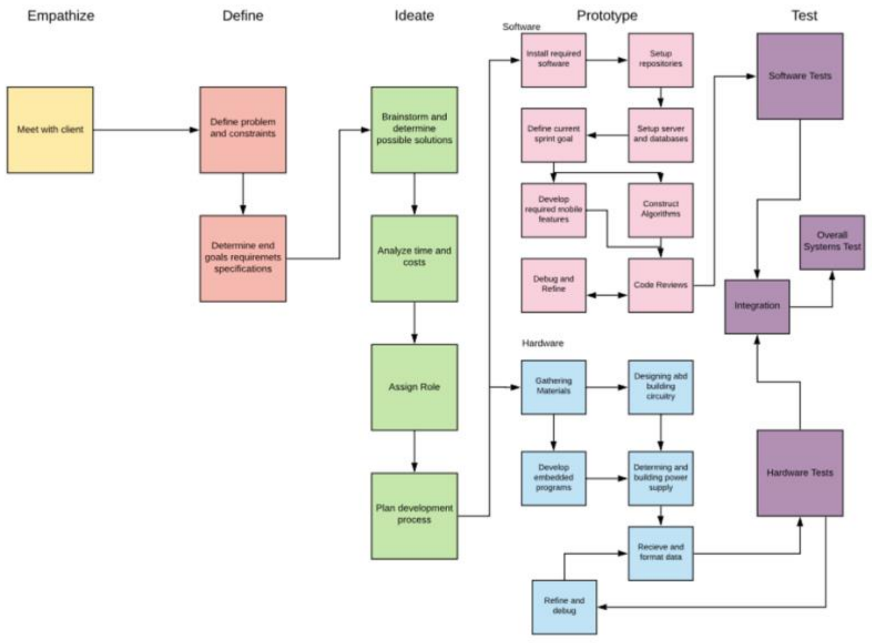
Empathize
Define
Ideate
Prototype
Test
Software
Meet with client
Define problem and constraints
Determine end goals requiremets specifications
Brainstorm and determine possible solutions
Analyze time and costs
Assign Role
Plan development process
Install required software
Setup repositories
Setup server and databases
Define current sprint goal
Develop required mobile features
Construct Algorithms
Debug and Refine
Code Reviews
Software Tests
Integration
Overall Systems Test
Hardware
Gathering Materials
Designing abd building circuitry
Develop embedded programs
Determing and building power supply
Recieve and format data
Refine and debug
Hardware Tests

Figure 3: Design Diagram

3. Statement of Work

This section will cover what is currently on the market, our technical considerations, our task break down, our risk and solutions, our expected milestones and expected results.

3.1 Previous Work And Literature

Automatic inventory management is not a new concept and has been used in the past. Companies such as Rolls Royce are implementing sensors to track their inventories locations. In addition, to them many other companies have implemented RFID tags to track what is moving from room to room or warehouse to warehouse. In all of the examples provided the companies are using RFID tags which are physical stickers placed usually on containers that hold the products. Though RFID scanners have a large set of industry examples, they are expensive and the tagging each item would be cumbersome. Due to the scale of our project and user satisfaction using RFIDs fails to accomplish our desired outcome. So, we have moved to a less expensive option that still has similar capabilities.

From further research we discovered another automated inventory management company named Barcodes Inc, they use barcode scanners to manage inventory. Employees scan the barcodes and the scanners used update the inventory data on the company database. In comparison to RFID they are cheaper and provide easy implementation. Though they provide similar results to RFID, barcode scanners are less passive and will need the user to remember to scan the item.

However, the portion of our project that checks the prices is not as commonly used in a specific company. There are a few websites that do price checking between different companies or sitewide for example with, Amazon where they show similar products from different vendors. Another pricing product available is from Rucksack which is titled Compare Prices with Barcode Scanner- for Amazon and eBay. This application allows the user to scan a barcode on any item and the app will pull it up on Amazon and eBay at the same time to see the cost differences. The prices evaluations that need to be set up for our project need to be for multiple stores and items. Additionally, they will be geographically based for which store is the cheapest.

3.2 Technology Considerations

For sensor tracking there are several different variations available on the market. However, for our project we were able to reduce sensor option to two different kinds, barcode scanner or RFID.

Originally, we had considered RFID tags, however, due to the additional costs of buying RFID tags and scanner, we felt that a barcode scanner would be better for the project. With RFID, the issue of placing and removing the RFID tag for each item would be an additional thing to do.

The barcode scanner proved to be a better choice as the barcodes are already on each item. Since the only thing we need is the ID of the item, barcode scanner will accomplish this. However, it’s important to keep in mind that items need to be scanned one at a time. There is also potential for wear and tear as the scanner is continually used.

Another area we decided was the microcontroller to gather all of the sensor data. The two controllers we were initially deciding between were Arduino and Raspberry Pi. We finalized on a Raspberry Pi 3 Model B+ because of its built-in wireless module and an on-board Linux OS. The wireless module is important as it will handle communication from the sensors to the database.

3.3 Task Decomposition

Our project is broken into two key components: the sensor network and the software application. For the best result the two parts will be created simultaneously, we have decided to have two members on the sensor network side and four on the software application side. Design plans and changes on either side that can impact the whole team will be made as a team. This is to ensure transparency and team success to minimizes inefficient work. For the implementation of our project we divided it into three phases, each with a focus on developing a prototype to final product: The objective of each phase is outlined below:

Phase 1

- Data collection from sensors
- Data is being transmitted to database from sensor
- Front-end can visualize data from the database

Phase 2

- Integrate system for multiple sensors to be registered to a specific product (assemble sensor array)
- Demand for new item purchase can be generated based on pantry contents

Phase 3

- Complete integration of sensor arrays for multiple products
- Purchase suggestion for multiple products can be determined

Each phase was treated as an Agile sprint to ensure continuous improvement upon the planned architecture. For a deeper breakdown of the project refer to the timeline in 4.1. In each phase, our work was tested and validated for refinement and approval.

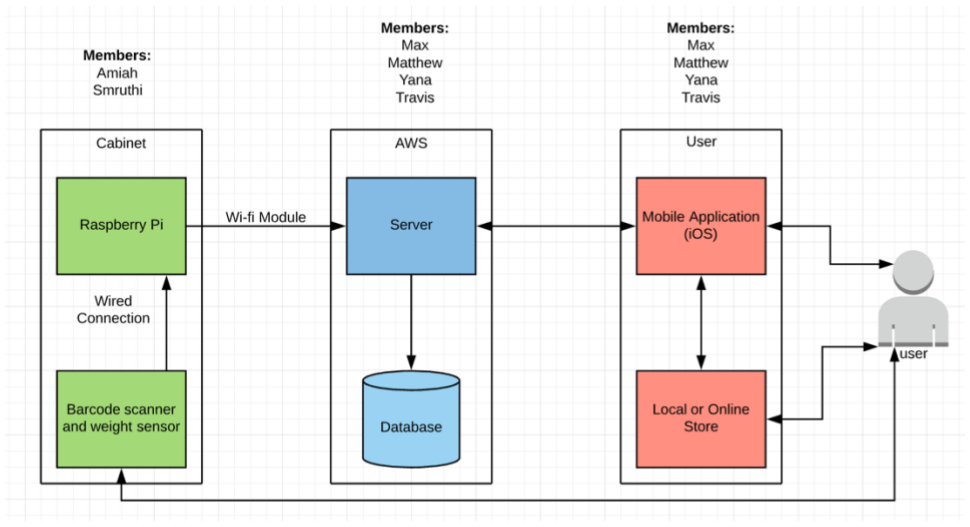

Figure 4: System Diagram with Member Assignments

As you can see from the figure above, the main system components can be divided further into different subsections. These sections include the cabinet (which contains the hardware), Amazon Web Services, and the user (which contains the mobile application and online stores). The mobile application receives data from either user input or geolocation tracking of the user to access data from the server. The server accesses this information and the server will run specific algorithms to determine the best prices and locations for each item. The mobile application will return to the user these prices and locations for the best deals. After an item is bought, the user can register the item in their cabinet at home by scanning the barcode of the item on the sensor. This sensor data is transferred to the raspberry pi which communicates with the server to register the item and update the database. Overall, the system diagram above shows the modularization and connections between each main component in our system. As far as role assignments we had two members focused on the hardware implementation working on the sensor network, Amiah and Smruthi. Working on the database/server side and user application side was the rest of the team featuring Matt, Max, Travis, and Yana. This part of the team had specific delegated tasks as well as an overall knowledge of the components.

3.4 Possible Risks and Risk Management

Risks:

- Accuracy of automatic orders is completely reliant on the accuracy of the items sensors used in the sensor network
- Can only make correct orders if the sensors properly establish a connection with the application which will then run an algorithm to tell which the cheapest option is
- Unfamiliarity with technology

Risk Management:

- Frequent testing of sensor network

- Ensure to test pre-made database for good price differences
- Securing outside sources that are familiar with the technology and being sure to compile information and placing in one drive for other to use

3.5 Project Proposed Milestones and Evaluation Criteria

- Proof-of-Concept
  - The phone application component should be set-up and ready for user to log in and view data being collected by the inventory monitoring device.
  - The inventory managing device should be connected and communicate with the phone application's database and should be sending information to the base
- Minimum Viable Product
  - The monitoring devices should be properly calibrated to measure what is being stored. The phone application component should be able to monitor product levels of multiple products and be able to produce an orders list and optimized pricing evaluation systems for online and in-store purchasing.
- Beta-Testing
  - The device and application will be constantly tested with real products over a period to time to find bugs, issues, and areas to improve.
  - We will physically scan in products and use phone application as if we are the client. This way of testing will provide feedback and bring attention to issues that were not previously found
- Integration
  - The phone application is modified so it is ready with the pantry's database
- Finalize Product
  - The finalize product will have all bugs fixed while the phone application's monitoring and price checking system is optimized. The entire project will be set up to handle multiple products from multiple types of food.

3.6 Project Tracking Procedures

We posted a weekly status report to track our progress made during the week. We also documented our meeting notes to record our thoughts and concerns that didnt make it onto the weekly report. Having separate meeting notes also served to validate what is discussed and decided during meetings. For communication and collaboration, we created a Slack channel so we could share documents and post deadlines in a central location for all of us.

Gitlab was heavily used to keep track of weekly progress. This was an easily accessible way to know where we were on more specific tasks. It was also a way to look back on more specific tasks that were generalized on the weekly report.

3.7 Expected Results and Validation

The end goal was to have a fully functional, autonomous pantry inventory management system. Client's validation will act as a major component in building our sensor network. The product will meet all requirements discussed in this document.

- Our product took sensor data to collect the amount of a product in the pantry. The sensor then sends a signal to our software through a Raspberry Pi. Then the data will help send an order request for the product.

We will consistently test our sensor network to ensure that all of our sensors record an accurate product count by comparing the calculated total with the true value. Validation will occur with a variety of products at various quantities to ensure our measurement algorithms return accurate data on what the product is. We will be sure to compare all data stored in the database with the measured data to check that the information given to our users is as accurate as possible.

4. Project Timeline, Estimated Resources, and Challenges

This next section covers the time for each specific task and overall costs for our project. We needed to make sure our project didn't exceed the time limit or budget constraints that our team was given. Additionally, we list the challenges we faced and what our final result consists of.

4.1 PROJECT TIMELINE

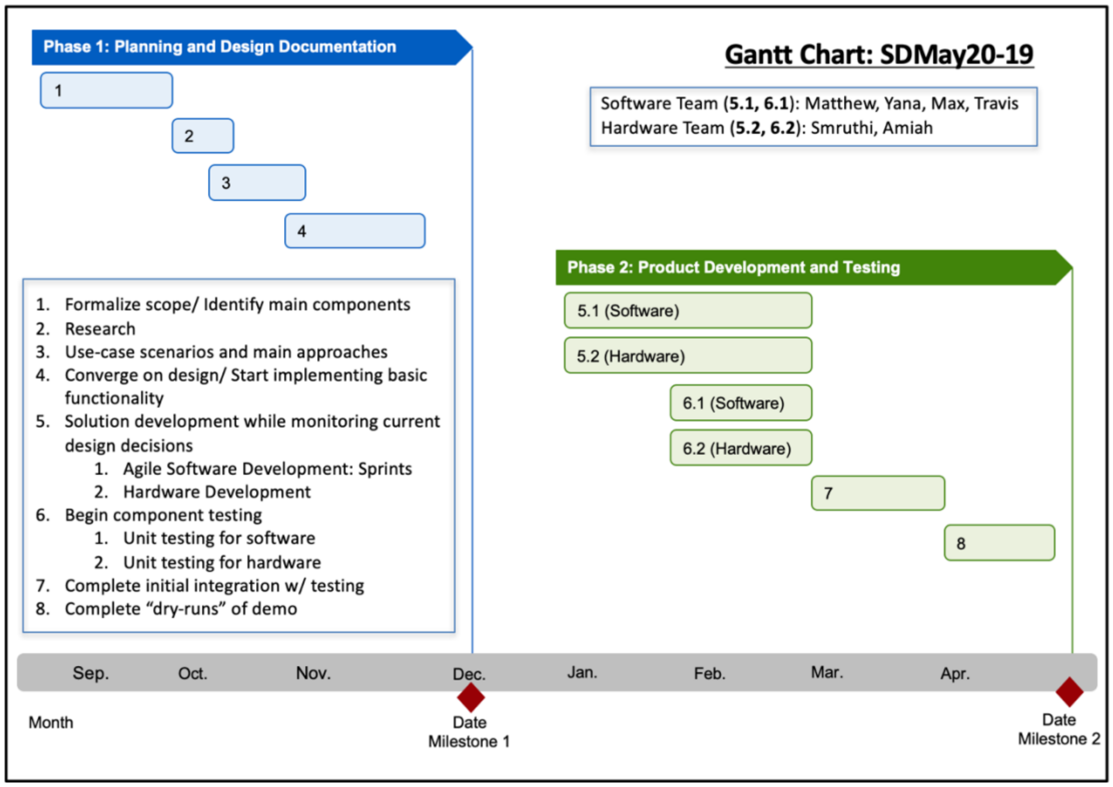

Figure 5: Gantt Chart

As you can see from the chart above, we have separated out our project into 8 main components that span the time allotted for the project. From the initial scope to the final demos, the chart gives an overview of what was expected and by when each part was completed. It also accounts for winter break, which explains the gap in the month of December. We feel like these steps highlighted the main tasks that were tackled on a monthly basis, and it gave plenty of time for the actual construction of the solution. We followed an agile-like approach to the software development, and made sure to constantly test the hardware and software separately before integrating and testing the final, complete system. We even gave a month after the final integration to test, demo, and possibly expand upon our solution.

The timeline also allowed for an adequate amount of time for preparation before we started creating a solution. It accounted for research, brainstorming, and analysis of requirements and constraints before constructing a prototype. This ensured enough thought was put into what was expected as a final result and what steps were taken to accomplish this. Also, it should be noted that we prepared for the PerCom conference in late March, so this coincided with our planned timeline with being towards the end of step 7, integration testing. Overall, this timeline distributed a sufficient amount of time to each component of our project and allowed our team to finish our project while meeting quality expectations and time constraints.

4.2 Feasibility Assessment

The initial expected result of the project was a proof of concept IoT solution for a specific area in the house, with hopes of potentially expanding to other parts of the home as well. As for a realistic finalization, our team believes we were able to successfully construct a proof of concept, especially since it was narrowed down to a single location in the home. Any previous foreseen challenges resided in tackling new technologies that our teammates might not have had experience with prior to the project. One example of this might be using Amazon Web Services to set up our server and database, or Xcode for those who haven't done iOS development. It required more research to use these technologies, since some of our team members had no experience with them.

4.3 Personnel Effort Requirements

Task	Description	Approx. hours
Set up AWS server and database	Research and set up the AWS server and backend services, as well as database design.	40+ After further research is done, a better estimate can be given on time.
Develop front-end user interface	Design and construct the iOS mobile application that the user will interact with.	100 This task will take up a lot of time since it handles the majority of the mobile application that needs to be built.
Algorithms and data analysis	Perform any required analytics and algorithm formulation to effectively compare the prices between online and local stores and whether the user's distance away from the stores might affect this decision.	50 This task will require a decent amount of time since the algorithms we produce will influence the choices made by the user. Therefore, constant analysis of these algorithms will be needed.
Construct circuits and integrate hardware components.	Effectively plan out and construct the overall hardware involved with the project.	80 This job will require a lot of work from the hardware team since it must be mapped out and calculated perfectly.
Embedded programming	Program the Arduino to correctly transfer the data collected from the sensors to the AWS server.	80 This task can be tedious and require some research to pursue, so it has been given a decent amount of time.
Testing and Integration	Consistently test the software and hardware separately before integrating them into one system. Perform subsequent tests and demos on overall system.	150 We made sure to leave a lot of time for testing to ensure that the quality of our solution is the best it can be. This helps alleviate the

		possibility of errors and future confusion after integration has been completed.

Table 1: Personnel Requirements

This table represents the main tasks involved in the construction of our IoT solution. Each task is given a description and an estimated time to complete it, as well as an explanation behind these approximations. These are rough estimates of how much time each task took to complete throughout the year.

4.4 Other Resource Requirements

The specific materials needed to complete the project were:

- Raspberry Pi 3 Model B
- Barcode scanner
- Wires and circuit components
- Power supply
- AWS EC2 instance
- AWS RDS instance
- Makeshift cabinet and items for test simulation

4.5 Financial Requirements

The financial resources provided to us was 200 dollars, mainly intended for hardware components. After researching specific products, we have listed the prices of all components that we needed to purchase below:

- Raspberry Pi, Power Supply, NOOBS loaded SD card: $52
- Barcode scanner: $29
- Total: $ 81

Total cost coming to less than our $200 limit given to us by our client.

5. Testing and Implementation

This section outlines and discusses the step by step processes taken when testing each component of our project.

5.1 Interface Specifications

In any project, comprehensive testing is a major component to having a successful outcome. It was crucial to test the sensors we use in our IoT network to ensure the accuracy of the data. With an agile operating style, we were able to consistently test our sensors to ensure accurate flow of data from the sensors to the database. To guarantee the sensors work for an increased number of items, tests were performed on various quantities and a variety of items. The sensor modules were extensively tested with various scripts to ensure the data that is retrieved from the sensors are transferred to the database. The data pulled by the sensors was then compared to the database to ensure the mobile application would be able to run the created algorithm and notify the user. This testing ensured the Raspberry Pi can function properly when accepting data from sensors and sending the information to the database.

5.2 Hardware and software

The sensor we decided to use for the project was a barcode scanner. The main purpose of the barcode scanner is to be able to keep track of which items are currently in the pantry, as mentioned in section 2.3. When an item is scanned into the pantry, the barcode scanner will send the data about the device back to the Raspberry Pi. The Raspberry Pi will then send this information to the database to add the item to correct table.

The transfer of information from the barcode scanner to the Raspberry Pi was initially tested by scanning physical objects. When an object is scanned by the barcode scanner, the expectation was for the value of the barcode to be received by the Raspberry Pi and then sent to the server. The Raspberry Pi then sent this information to the database being stored in AWS using the built API. Initially this was tested by creating a few scripts to test each part. The first would be a script that would take a barcode as input and send the data to the Raspberry Pi. The second script would take the data from the Raspberry Pi and use the API to add the information to the database. After this was completed, the whole process was tested by scanning a physical object and ensuring the data flows from the barcode scanner all the way to the database. This testing ensures the Raspberry Pi can make successful calls to the database after receiving the signals from the barcode scanner.

When an item is detected as no longer being in the cabinet, the database will be updated for this item. The Raspberry Pi will receive the signal that an item needs to be removed from a user's table and send the information to the database. Testing for this functionality was completed in the same fashion as the sending of information from the Raspberry Pi to add an item. Additional testing for this was done from the application side to verify the proper tables have been updated.

The testing of the software will be completed in a variety of way. First, the algorithms and code were tested through automated tests in GitLab using pipelines. This ensured the functionality of the code is always stable. Testing of the frontend UI was completed with the built-in testing framework, XCTest. Additionally, the back-end API calls will be tested through a software called Postman. Postman is a program that provides a GUI for testing REST API calls, which are being used in this project. This will help with the automation of feeding data into the database.

5.3 Functional Testing

Throughout the project many tests will be completed to ensure the functionality is performing as expected. In the application itself, unit tests will be created and ran for each feature. These tests can be created directly within XCode as the software has a built-in testing framework called XCTest. After the unit tests are complete, integration testing will be completed where the various features will then be tested as a group. To follow up this testing, the functionalities of the application will be tested from front to end. This will be done to ensure there is a direct flow from items being detected in the cabinet to individuals receiving alerts to either go buy the item or that it has been bought online for them. Lastly, we will test the acceptance of the application and make sure the system follows the requirements set forth by the customer.

FR.1: A user can add multiple scanners to their home system to track products in various cabinets. The inventory of sensors should be sent to the database and see all active devices.

> **Test Case:** For this requirement, we want to register a scanner with the application to ensure a full connection between the scanner and database.
>
> **Test Steps:**

1. Add scanner to network via the mobile application setup page.
2. Send data to database.
3. Ensure new sensor is saved accurately into the database for correct user.

Acceptance Criteria: Database successfully shows activated sensor.

FR.2: The user should be able to scan items into the pantry and have an inventory of items in their pantry.

Test Case: For this requirement, we want to test that the scanner can accurately send a new item that is being placed into the pantry to the database.

Test Steps:

1. Scan a new item and place in the pantry.
2. Send data to database.
3. Ensure new item data is sent to the database and is listed under the correct user.

Acceptance Criteria: Database shows new item for the correct user.

FR.3: The user should be able to see the list of items on the application that are needing to be purchased.

Test Case: For this requirement, we want to test that the database is updating to the inventory of items that are available at the user's home.

Test Steps:

1. Login as a valid user to the application.
2. Open list of items needing to be purchased
3. Compare to actual inventory in house.

Acceptance Criteria: Database shows an accurate inventory of items needing to be purchased for user.

FR.4: The user should receive a notification to ask if they would like to purchase an item on their shopping list when they are in a 0.5-mile radius of a store. After interacting with the notification, the database is updated according to their response

Test Case: For this requirement, we want to test that the database sends a notification to a user when they are close to a store that sells an item they need. Database reflects their response.

Test Steps:

1. Add a user with an item needed to be purchased that is sold by a local store into the database.
2. Login as that user and have their location be within 0.5 miles of the store.
3. Check that a notification was sent.
4. User responds that they are picking the item up from the store or to ignore deal.
5. Response is sent to the database

Acceptance Criteria: Database accurately sends the notification when the user is within the 0.5-mile radius. Additionally, if the user picks up the item from the store, that item is removed from their list.

A group of test cases will be created to test the software created. The testing suite will test various cases such as expected data and unexpected data. This will ensure all edge cases are covered when creating new versions of the software. These tests will begin with unit tests which will ensure any new features are operating as expected. Most of these tests will be completed using XCTest in XCode. After unit tests, integration testing will occur to ensure the new features do not cause issues to other modules. This will make sure all requirements are being met. The third step in the functional testing will be user testing. This phase will guarantee the experience the user has with the application is good. This test case will help in identifying how a user will use the IoT device and mobile application.

After completion of the functional tests, the solutions decided on for this project can be verified as successful.

5.4 Non-Functional Testing

Testing the non-functional features is important for the user experience. As more individuals begin to use the application, parts of the program can possibly not work as well as expected. This is where the non-functional testing comes into play. With our application being hosted on AWS and using its various services, it makes testing for these cases easier. Manual tests will be conducted to test for scalability, data integrity, and usability. These tests will be rate on a pass or fail metric based on the outcome of the procedures for each task.

Scalability

Procedure:

1. Begin with x users registered to application and able to receive notifications from the application.
2. Add a user to the application with a location within 0.5 miles of a store
3. Check logs for x+1 users connected to the application and receiving a notification.

Success Criteria: x+1 users are receiving notifications.
Failure Criteria: Fewer than x+1 users receive a notification.

Data Integrity:

Procedure:

1. Begin monitoring data being sent between the Raspberry Pi and the server on AWS. Have data saved to a log file.
2. Send information from Raspberry Pi to the database. Have data saved on databased in log file.
3. Compare log files at end of testing phase to determine accuracy

Success Criteria: Log files are equivalent.
Failure Criteria: Log files are not equivalent.

Usability:

Procedure:

1. Track the time is takes to manually find an item from a user’s shopping list on a nearby store’s website.

2. Track the time it takes for a notification to be sent to a user to decide to buy an item from their shopping list after arriving within 0.5 miles of a store.
3. Compare manual and automatic times.

Success Criteria: Automatic time is less than manual time.
Failure Criteria: Manual time is less than automatic time.

5.5 Process

The main process for testing the sensors will be completed by comparing the data collected by the sensors and stored in the database to the actual quantity of items in the cabinet. This testing will ensure the data being sent to the server with customer visibility is accurate.

The algorithms will face unit tests and user tests to confirm the expected output of the application. This will involve using the pre-populated table as mentioned in section 5.2. Additionally, users will test the application to test the given features, give feedback, and report any bugs found in the program.

The testing of the primary features included in the application will be tested separately from the sensors. To test these features which rely on data from the sensors, a pre-populated database will be used to simulate different scenarios. This type of testing will be helpful when trying to have granular testing of the features. Each feature laid out for this part of the project will have automated tests created so that the implementation can be continuously tested as new features get applied. This setup will ensure regression errors are prevented at a high rate.

To ensure testing is completed at a high level, an agile development process will be followed. All functional requirements, seen in section 3, will be added to Trello for assignment of work. Weekly meetings with the client will continue throughout the year to ensure features are added in the correct priority. The development of these features will be completed in sprints of 2 weeks. Each sprint will consist of features being committed to. After each sprint, the product will be demoed to the client for input after the thorough testing. Any feedback from the client will be prioritized for the completion in the next sprint. Phase 1 of the project, section 3.3, will be retrieving data from the sensors and ensuring the application can visualize the data. Phase 2 will consist of assembling the sensor array and generating new item purchases for the user. Phase 3 will have the complete sensor array functioning and purchase suggestions for multiple items functional.

5.6 Results

The following sections indicate the results of the tests mentioned in sections 5.3 and 5.4.

5.6.1 Functional Testing

Seen below is a table with the results of our functional requirements from section 5.3.

Test Number	Test Title	Comments/Results	Pass/Fail
FR.1	Adding Scanners	Scanner was successfully able to be added using the registration page on the application and be seen in both the application and database.	Pass
FR.2	Add Items to Pantry	Pantry item was added to both the database and the application.	Pass
FR.3	View List of Items Needed	The frontend UI shows the list of items needed purchased and maps correctly to the database.	Pass

FR.4	Notification When Near Store	A notification was successfully shown during simulation on an item that was needed.	Pass

Table 2: Testing Results of Functional Requirements

5.6.2 Non-Functional Testing

For testing the non-functional requirements of our project, we decided to manually test each of the requirements. With the running of our application on AWS, a lot of these non-functional requirements were easy to complete. For scalability, we wanted the ability to keep on adding users without a limit. One of the features within AWS is to be able to continuously monitor and add data to the database. This allowed for the scalability of adding users to the database successfully. For data integrity we wanted to ensure the flow of data was accurate throughout the whole process. In this section we outputted the data at the Raspberry Pi into a log file to be able to compare the information all the way to the database. Our tests were successful here and the data was completely accurate. The last non-functional requirements that was vital for our project was usability. For this test, we manually tested the application to ensure notifications were received on the application as soon as the user was in the given radius of the store. The results of these tests showed that the usability of this application was successful.

6. Closing Material

The concluding section summarizes the current status of our project at the end of the second semester of development, including the completed work and the possible future expansion.

6.1 Conclusion

Following our second semester working on the project, our team has completed all of the necessary goals as discussed with our client, as well as identified some possible areas for future development. The project was successfully divided amongst members into two smaller groups to complete both the hardware and software aspects, and we were able to complete all needed components before presenting at the PerCom conference in March. Future project expansion may include things such as a weight sensor to track multiple of the same item, a modular design to scale to different parts of the kitchen, or an in-app online purchase option to purchase everything directly from the app.

6.2 References

Pricing Reference Websites

1. https://www.microcenter.com/product/460968/3_Model_B
2. https://bit.ly/3571O1m

IEEE Standards:

1. https://standards.ieee.org/standard/1028-1997.html
2. https://standards.ieee.org/standard/802_11-2016.html
3. https://standards.ieee.org/standard/1532-2002.html

6.3 Appendices

Raspberry Pi 3 Model B Documentation

https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf