To Online Shop, or To Not Online Shop

DESIGN DOCUMENT

SDMay20-19
Goce Trajcevski

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

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

Revised: November 3, 2019 Revision 2

Executive Summary

Development Standards & 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
- Additional new skills/knowledge to be determined

Table of Contents

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

1 Introduction

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.

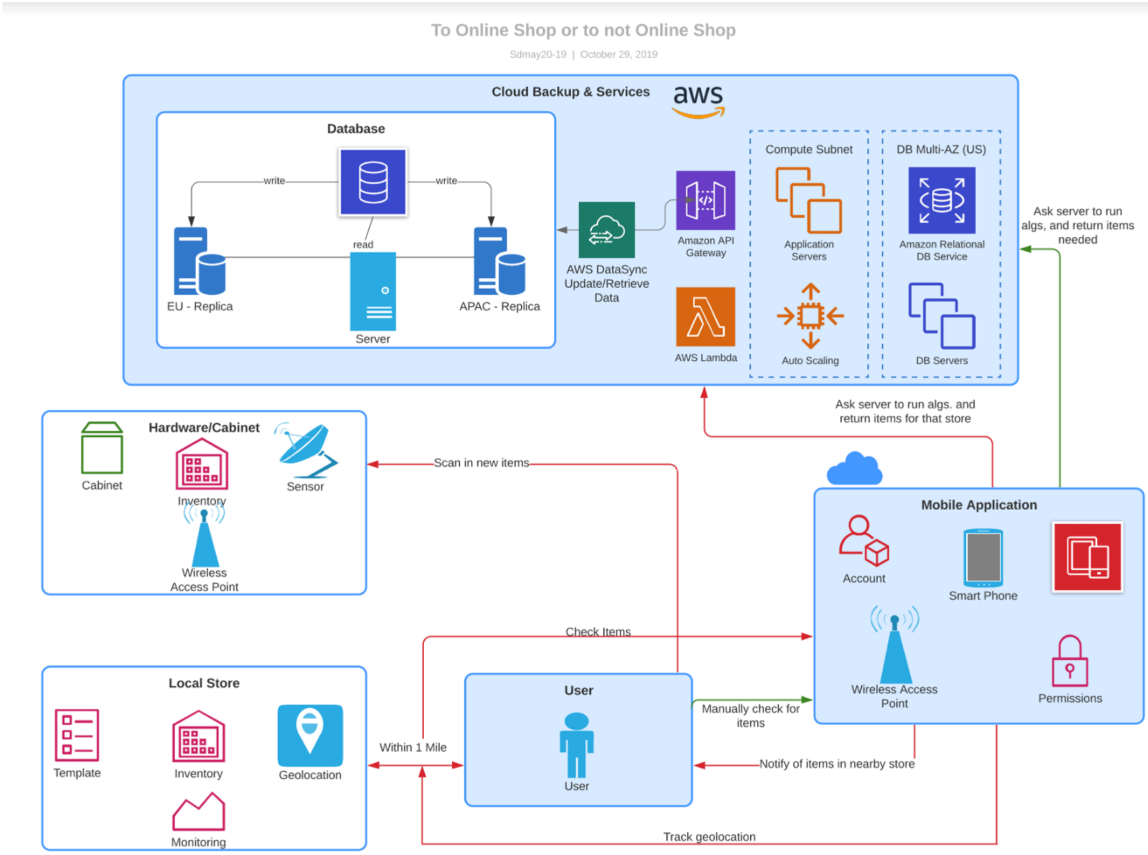
1.2 Problem and Project Statement

The goal of the project is 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" and prepare an online order
c. Detecting when a particular 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 actually 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 should determine if it is cheaper to buy an item online or in-store. Geolocation should also be used to find the best deal for a product.

Our solution is to use an RFID sensor or barcode scanner along with a microcontroller that can automatically monitor status of the items located in the pantry or cabinet. At midnight 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 will write algorithms to determine if it's cheaper to buy items online or in-store. We will also use geolocation to figure out where the best deal for buying the products are.



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 pantries or cabinet and installing 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 adding it to this list.

1.3 Operational Environment

The sensors and microcontrollers used to monitor the status of the items will be stored in a dry pantry or cabinet area with a WiFi 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 will be done through mobile application.

1.4 Requirements

Constraints

- Budget of 200 dollars.
- Only viewing a few stores within a 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 will be 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 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 five-mile radius
- The product will not select stores which total prices exceed user's budget

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 are meeting with the customer and are coming up with our project requirements and are finalizing 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 deciding what products we are implementing in our project. We are to purchase the sensors and microcontrollers that we will be implementing into the final product. Figuring out how we will be developing the mobile application is also important.
3. **Decide upon use-case scenarios and main approaches to be investigated as "candidate-solutions" (November 1th, 2019)**
    - In this deliverable we are deciding the use cases and approaches that we will be implementing 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 to demo to the panel. This should include work on some core features to show initial development.

Semester 2:

1. **Revisit the design decisions (January 30th, 2019)**
    - Continue from the development of the previous features and work on assigned tasks. Demo completed features to client and make necessary improvements
2. **Define and start with the individual components testing (February 28th)**
    - Not defined yet
3. **Complete the initial integration, define and start with integration-testing for plausible use-cases (March 30th, 2019)**
    - Not defined yet
4. **Complete "dry-runs" of the demo and finalize the deliverable-version (April 30th, 2019)**
    - Not defined yet

2. Proposed Approach, Specifications and Analysis

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 should be available to update automatically during 12 hours of the day.

Data Integrity: The sensors will need to provide accurate and consistent data so the inventory is monitored correctly. We will employ a barcode sensor and weight sensor to have auditing of sensor readings. There will be tests ran on sensors as well an error of margin determined.

Deployment: The sensor network will be 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 object coming in and out of the cupboard. In order to protect the sensor it will be in a protected environment to avoid damage of the system.

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.

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

2.1.2 Standard Protocols

Following our Standards reflection our team will specify which specific IEEE standards apply to our project.

2.2 Proposed Design

The solution design will have two main components: the Home IOT and Application side. The sensor on the IOT side will satisfy functional and non-functional requirements for the monitoring of grocery inventory in the kitchen cupboard. The application analytics will satisfy requirements of geolocation and optimization using data from the sensor network.

The sensory device will employ sensors to monitor inventory levels of single product. With scalability the client will be able to have multiple devices in order to monitor inventory of different cupboards. The sensor will receive data and send communication to the database on the back-end. We intend to implement an RFID sensor that will communicate with AWS and our IOS application.

The software will be an IOS application. We will use Swift to develop our application that will interact with our database. The front-end will use geolocation to send a notification when an individual is close to a grocery store and notify them of cupboard inventory and shopping options. We will use grocery store APIs and be able to compare to online shopping prices.

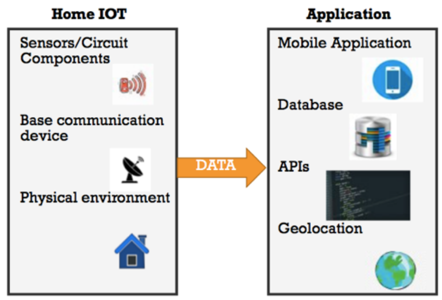


Figure 1: Deployment Diagram of Proposed Design

In addition to this, we also constructed a basic system diagram to modularize each component of the system and show how these sections are connected.

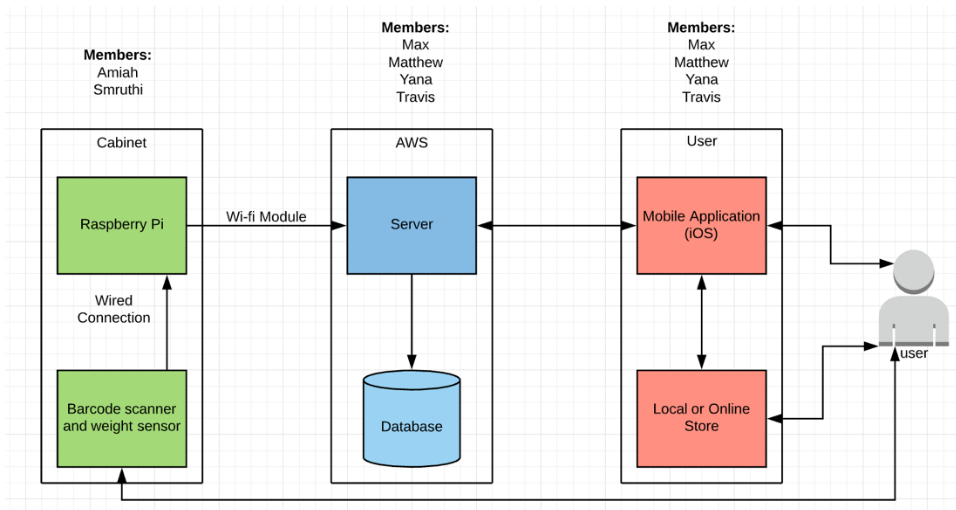


Figure 2: System Diagram with Member Assignments

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 will receive data from either user input or geolocation tracking of the user to access data from the server. The server will access 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 will be transferred to the raspberry pi which will communicate 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 have 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 will be the rest of the team featuring Matt, Max, Travis, and Yana. This part of the team will have specific delegated tasks as well as an overall knowledge of the components.

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 have to 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.

We are using a barcode sensor and Raspberry Pi as our sensor network. There were several options to consider as we chose between barcode and RFID as a dependable form of sensing items, and the Arduino versus Raspberry Pi for its analog-to-digital converter as well as its built in Wi-Fi/Bluetooth capabilities. Since we will be gathering a variety of data the Raspberry Pi seemed like the right choice for the implementation.

A barcode scanner will be incorporated as items will be scanner as they enter and exit the company's pantry, with subtracting or adding to current amount in inventory database. A weight sensor is an option for scalability of the project; being able to tell by weight whether there is more than one of an item that would have the same barcode.

For analytics we will have our application front-end and database/server back-end. The front end is going to be built with Swift 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 will be using AWS to monitor database information and make decision based of specifications and prior information. Since the framework is maintained by Amazon there will be 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 HTTPS calls.

2.4 Development Process

We will be using the Agile developmental method. This way we are allowed to iterate often and get feedback from our client often as well. It is the framework that fits best as we are meeting with our client each week to discuss development. Through these frequent meetings we will be able to discuss sprints and next steps in the development process. If there are steps that do not meet our requirements we will be able to go back and iterate on certain parts rather than starting over. Since we have a very software focused project once initial hardware is setup this will flow rather nicely, and the same steps can be applied across systems on our project.

2.5 Design Plan

Our design plan includes the 5 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 requirements and goals 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 will develop the two separate parts and test them on their own and then integrate 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.

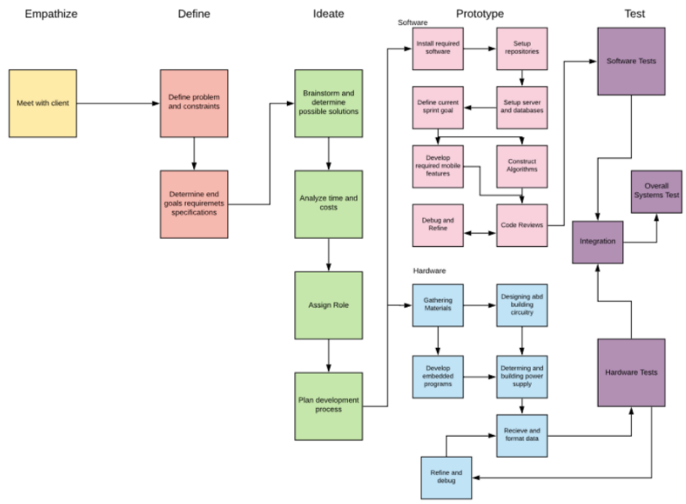


Figure 3: Design Diagram

3. Statement of Work

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 will accomplish that. However, it's important to keep in mind that items need to be scanned one at a time and that the items are in line of sight with the barcode. There is also potential for wear and tear as the item is continually used.

Another area we need to decide was the micro controller to gather all of the sensor data. The two controllers we were initially deciding between 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 2 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 will divide it into 3 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 will be 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 will be tested and validated for refinement and approval.

3.4 Possible Risks and Risk Management

Risks:

- Accuracy of automatic orders is completely reliant on the accuracy of the 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 a orders list and an 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 will be posting a weekly status report to track our progress made during the week. We will also be documenting our meeting notes to record our thoughts and concerns that don't make it onto the weekly report. Having separate meeting notes will also serve as a way to validate what is discussed and decided during meetings.

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

3.7 EXPECTED RESULTS AND VALIDATION

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

Our product will take a sensor data to collect the amount of a product in the pantry. The sensor will then send 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

4.1 PROJECT TIMELINE

Task #	Milestone/Activity	Start Date	End Date
1	Formalize Scope/ Identify main components	9/6/19	9/20/19
2	Research	10/1/19	10/10/19
3	Use-case scenarios and main approaches	10/7/19	11/1/19
4	Converge on design/ Start implementing basic functionality	11/1/19	11/30/19
5	Revisit design decisions while continuing to develop the solution	1/13/20	1/30/20
6	Begin Component Testing	1/30/20	2/28/20
7	Complete initial integration w/ testing	2/28/20	3/30/20
8	Complete "dry-runs" of demo	3/30/20	4/30/20

Task #	Sep-19	Oct-19	Nov-19	Dec-19	Jan-20	Feb-20	Mar-20	Apr-20
1	X							
2		X						
3		X						
4			X					
5					X			
6						X		
7							X	
8								X

As you can see from the tables 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 table gives an overview of what is to be expected and by when each part should be completed. It also accounts for winter break, which explains the gap in the month of December. We feel like these steps highlight the main tasks that will be tackled on a monthly basis, and it gives plenty of time for the actual construction of the solution. We will follow an agile-like approach to the software development, and make sure to constantly test the hardware and software separately before integrating and testing the final, complete system. We have even given a month after the final integration to test, demo, and possible expand upon our solution.

The timeline also allows for an adequate amount of time for preparation before we start creating a solution. It accounts for research, brainstorming, and analysis of requirements and constraints before constructing a prototype. This will ensure enough thought is put into what is expected as a final result and what steps can be taken to accomplish this. Overall, this timeline distributes a sufficient amount of time to each component of our project and, if followed correctly, will allow our team to finish our project while meeting quality expectations and time constraints.

4.2 Feasibility Assessment

The expected result of the project will be 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 projection, our team believes we will be able to successfully construct a proof of concept, especially since it will be narrowed down to a single location in the home. Any foreseen challenges will primarily reside 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. It will require more research to use this technology since none of our team members have experience with it.

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.

This table represents the main tasks involved in the construction of our IOT solution. Each task is given a description and estimated time to complete it, as well as an explanation behind these approximations. As of now, these are just estimates of how much time each task may take and they are subject to change during the process of development throughout the year.

4.4 Other Resource Requirements

The specific materials needed to complete the project are:

- Raspberry Pi 3 Model B
- Barcode scanner
- Weight Sensor
- Wires and circuit components
- Power supply
- AWS server and database
- Makeshift cabinet and items for test simulation

4.5 Financial Requirements

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

- Raspberry Pi: $35-40
- Weight Sensor: Potentially free?
- Barcode scanner: $50-$100

As stated, the price of the weight sensor could potentially be free since the sensor from previous groups might be handed down to us. After a quick look at these prices, we can assure that we will meet the $200 limit that has been given to us.

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 will be crucial to test the sensors we use in our IoT network. With an agile operating style, we will consistently be testing our sensors to ensure accurate data is being pulled into the database. To guarantee the sensors work for an increased number of items, tests will be performed on various quantities and varieties of items. The sensor modules will be 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 will then be compared to the database to ensure the mobile application is able to run the right algorithms and notify the customer. This testing ensures the Raspberry Pi can function properly when accepting data from sensors and sending the information to the database.

5.2 Hardware and software

The first of the sensors we will be using is 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 be sending 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 the user's items table.

The transfer of information from the barcode scanner to the Raspberry Pi will be tested initially by scanning physical objects. When an object is scanned by the barcode scanner, we expect the value of the object to be successfully sent and received by the Raspberry Pi. The Raspberry Pi will then need to send this information via API to the database being stored in AWS. Initially this will be tested by creating scripts that would input the values of items sent by the barcode scanner and call the functions on the Raspberry Pi that send information to the database. After this is completed, the whole process will be tested by scanning a physical object and ensuring the data flows from the barcode scanner all the way to the database. This testing ensure 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 will be completed in the same fashion as the sending of information from the Raspberry Pi to add an item. Additional testing for this will be 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 will be tested through automated unit tests. This will ensure the functionality of the code is always stable. This testing will be 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 can be created directly within XCode as the mobile application will be created for iOS users. 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.

To test the validity of the sensors we are using for this project, various items of different quantities will be scanned and placed in the cabinet. The data sent to the Raspberry Pi will be analyzed before and after scanning and placing the items. The two data sets will then be compared to ensure the accuracy of the sensors are capable of being used for the project. This style of testing will be completed on both the barcode scanner and weight sensor.

The ability to add or remove items from the pantry from the sensors and updating the system will be evaluated on pass or fail metric scale. For the test to pass, the Raspberry Pi must accurately retrieve item data from the sensors and send the information to the database. If the Raspberry Pi cannot update the database accurately, the tests will be marked as a fail and focused on for remediating issues.

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. There are many platforms that can be used to test these areas. For example, we can use BlazeMeter to simulate additional traffic on our application. This will help us explore the scenario of the number of users using the application increasing dramatically. We will be using services like BlazeMeter to test the availability, scalability, and reliability of our application.

In addition to automated tests on the server, manual tests will be conducted to test for scalability, data integrity, availability, and usability. These tests will be rate on a pass or fail metric based on the outcome of the procedures for each task. For scalability, a manual test of adding more devices and users to the application will be conducted. It will be important for users to continuously add devices and accounts to the application. Data integrity will be tested to ensure data is being transferred to the database without being altered. A test of looking at two files will be completed for this test. The first file will be from the Raspberry Pi before it sends the data to the database and the second file will be from the database. These files need to be identical as it is important items are only purchased that the user wants. The availability of the application is important as the user needs to always know they are getting the best price for their items. A successful test will indicate the user receives notification for the items that they need to purchase or items that will be purchased online. Finally, usability will be tested. This non-functional test will ensure the user saves money and time by using the application. The automation the application provides will be compared to data from manual completing the tasks.

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

We are currently closing in on the first half of this two-semester project and do not have enough results currently. The following are a few of the tests that will be conducted to ensure all requirements are met.

1. Sensor data retrieval testing
    a. No tests completed yet

2. Item purchase testing
    a. No tests completed yet
3. In-person purchase route optimization testing
    a. No tests completed yet

6. Closing Material

6.1 Conclusion

Thus far, our team has learned about all of the expectations of our client and gathered the necessary requirements to discuss at our meetings we have conducted so far this semester. Additionally, our time has done a good job at working with each other in order to come up with a solution for our problem. We are currently in the early development stages and are still ironing out the final details before we begin prototyping, although we have decided on specific parts that we will be using for our project. We have split into two main subgroups, one with two people and the other with four people, in order to organize the work for this project. The two-person subgroup will primarily work on the Home IoT, including sensors, circuit components, the base communication device, and the physical environment. The four-person subgroup will work primarily on the mobile application, geolocation, database, and all necessary APIs.

6.2 References

Pricing Reference Websites

https://www.microcenter.com/product/460968/3_Model_B

https://www.adafruit.com/product/1203?gclid=Cj0KCQjw9fntBRCGARIsAGjFq5GLrJspq7wXrja1O0B7KF24Z4V7MrMxNpKAF50HEncGgWSauBxKqTUaA0m9EALw_wcB

6.3 Appendices

Raspberry Pi 3 Model B Documentation

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