Multi-Context Shopping Optimization

Design Document

Team Number: sdMay20-23

Clients/Advisors: Goce Trajcevski & Ashfaq Khokhar

Team Members:

Max Garton
Ethan Shoemaker
Jesrik Gomez
Karla Montoya
Nate Wernimont
Arnoldo Montoya-Gamez

Team Email: sdmay20-23@iastate.edu

Website: http://sdmay20-23.sd.ece.iastate.edu/

Revised: October 6, 2019 / v1

Executive Summary

Development Standards & Practices Used

The following development standards and practice apply to this project:
Electronic Circuits:

- Safety
- Reliability
- Durability
- Efficiency

Software Practices:

- Agile development
- Lean development (fail-fast)

Engineering Standards:

- Modularity
- Reliability
- Scalability
- Performance

Summary of Requirements

Functional Requirements

- A device measures and reports inventory status of household goods (such as milk) to a remote server
- The remote server tracks the status of inventory for each user
- A remote server computes the optimal destinations and routes for shopping given a variety of constraints (price, time, location, convenience, delivery speed)
- An android application allows users to retrieve optimal shopping routes from the remote server
    - The android application allows users to add or remove (and adjust quantity) of additional items to their personal shopping list

Environmental Requirements

- A device measuring and reporting inventory statuses would require wifi in the user's home (and potentially in their fridge)
- The device requires a power source
- The device must be able to operate in a refrigerator (down to 0 degrees Celsius)

Economic Requirements

- Microcontrollers, sensors, power sources and housings for the inventory measuring/reporting device
- Two central servers, one for data collection and one for recommendations, will be needed

- Smart phones (Android devices) will be needed to use the mobile app.

Applicable Courses from Iowa State University Curriculum

- COM S 227
    - Object-Oriented Design
- COM S 228 & COM S 311
    - Data Structures and Algorithm Efficiency
- COM S 309
    - Project Planning, and Versioning
- S E 339
    - Software Architecture
- S E 319
    - User Interfacing
- CPR E 288
    - Sensor Applications and Embedded Systems
- CPR E 388
    - Android Application Development
- ENGL 314
    - Reporting, Documenting, and Technical Communication
- STAT 330
    - Introduction to Statistics
- CPR E 308
    - Operating System basics, Inter-Process Communication, and File Management
- COM S 363
    - Database Management
- S E 329
    - Software Project Management
- CPR E 430
    - Information and Networking Security
- CPR E 489
    - Data Communication and Client-Server Paradigms

New Skills/Knowledge acquired that was not taught in courses

We expect to learn and research the following knowledge areas in order to be successful in this project:

- System architecture design
- Cloud computing
- Full-stack development
- Hardware design
- Requirements development

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Acknowledgement

The team thanks the Iowa State University department of Electrical and Computer Engineering for giving us a resources, guidance and expert consultation. We appreciate the Electronic Technology Group (ETG) for providing us with our team website server and hardware components for the project. Thank you to Goce Trajcevksi for meeting with us weekly to give us guidance and advice.

1.2 Problem Statement

Consumers are presented with many different ways to shop (going to a favorite store, going to the closest store, or going to the store where an item's price is lowest, taking the most efficient route to reach several stores). How can a consumer know that the choice they've made is the best choice? How can we provide an optimal shopping recommendation to a consumer using a combination of contextual information such as the user's current inventory, the user's current location, the user's shopping list, stores in proximity to the user, hours of the stores, and the stores' availability and prices for items?

Our solution is to develop a system that utilizes a microcontroller to automatically monitor the quantity of an item that a user has in their household and communicate the data to a remote server that tracks the user's inventory of several items. This inventory data would be accessible and modifiable on a smartphone app so that the user can view it and make any necessary updates. Then, the user would be able to utilize this application to create a shopping list (items can be added automatically when supply is low, or the user can manually add and remove items). We would then develop a remote server that creates recommendations for shopping based on the user’s current location, inventory data, the supply of items at nearby stores, and other contextual data to suggest an optimal way for the user to shop.

1.3 Operational Environment

The microcontroller device that measures the user’s inventory is expected to be stored within range of a WiFi connection (with internet service). If the device is stored inside a refrigerator, this may block or severely limit the connection. The device will also need a power source (either via a powered USB port or a power outlet) to function.

1.4 Requirements

Functional Requirements

- A microcontroller measures and reports inventory status of household goods (such as milk) to a remote server
- The remote server tracks the status of inventory for each user
- A remote server computes the optimal destinations and routes for shopping given a variety of constraints (price, time, location, convenience, delivery speed)
- An android application allows users to retrieve optimal shopping routes from the remote server
    - The android application allows users to add or remove (and adjust quantity) of additional items to their personal shopping list

Environmental Requirements

- A device measuring and reporting inventory status would require wifi in the user’s home (and potentially in their fridge)
- The device requires a power source
- The device must be able to operate in a refrigerator (down to 0 degrees Celsius)

Economic Requirements

- Microcontrollers, sensors, power sources and housings for the inventory measuring/reporting device
- Smart phones (Android devices) will be needed to use the mobile app.

1.5 Intended Users and Uses

This solution is intended to be used by any consumer that shops regularly.

- Families in an urban and suburban setting
- Members of apartments who share kitchen and other household supplies

The main use cases of the solution are:

- Automatically tracking supply or inventory of a household item, such as milk, and updating the shopping list to reflect a low supply
- Maintaining a shopping list curated by the user

- Viewing optimal recommended shopping patterns based on contextual information and given priorities (the user can prioritize going to the closest stores, spending the lowest amount of money overall, getting the items as fast as possible, and other options)
- Viewing a family member’s shopping list (the above use case extended to multiple people)

1.6 Assumptions and Limitations

In order to narrow the scope of the problem statement, we introduce the following assumptions and limitations:

Assumptions

The users' home has wifi

For the project, we are assuming the refrigerator scale has a suitable WiFi connection. Without it being connected to WiFi, the inventory sensor’s data cannot be used to make shopping recommendations.

The user owns a GPS and Data enabled Android device

Because our whole project wraps around the idea of sending notifications to users, we will not be able to do this unless our user owns an Android device with the required technology.
Also, to figure out whether stores are nearby we will

Limitations

The product shall not be too expensive

The sensor devices should have a reasonable cost (less than $25 per sensor for the finalized product). The servers used for the project should be of minimal cost.

The product should have minimal impact on WiFi throughput

The sensor devices should only send data over WiFi and the internet when necessary (when the inventory has changed or during initial setup).

The product should not overuse the smartphone’s hardware (battery, RAM, CPU, storage, WiFi)

Any algorithmic computation (such as recommending shopping routes) should be done on a server and not on the user’s device. This is to ensure that the user interface on the Android application is responsive.

The only user interfaces of the project will be within the Android application.

Because Android is the most common mobile operating system, we will develop our mobile application on Android first. If time allows, a web application or iPhone application with the user interface may be developed.

1.7 Expected End Product and Deliverables

The final product delivery will be split into a proof-of-concept, minimum viable product, and a final product deliverable. The proof-of-concept and minimum viable product (MVP) will be delivered by the end of the Fall 2019 semester. The finalized product will be delivered at the end of the Spring 2020 semester. An overall design manual of the product and its architecture will also be provided for any future development.

1. Proof of Concept (December 1, 2019)
    a. The proof of concept prototype will demonstrate the capability of such a system to measure and track the supply of household items, communicate the data to a server, show the data to a user on a mobile device, and recommend an optimal way to shop based on the given data and user inputs.These may be individual working components, but the larger system may not be connected yet.
2. Minimum Viable Product (March 15, 2020)
    a. The minimum viable product is the overall system connected, where each component listed in the proof of concept is completed and integrated so that the system functions. Functionality may be limited at this point (the recommendation may not be based on all information or the user's input).
3. Finalized Product (April 15, 2020)
    a. The finalized product would have complete functionality, supporting all use cases described. The recommendation feature is extended to take into account other users' (family members) locations and shopping lists, optimizing the shopping experience for a whole household instead of one user.

2. Specifications and Analysis

2.1 Proposed Design

- Our current approach includes using simple analog sensors that can be interfaced to a widely available microcontroller in order to monitor the current contents in a user's home
    - The current plan is to use a Raspberry Pi as our microcontroller, and pressure or weight sensors to sense products in our fridge.

    - Because we plan on using an analog pressure sensor, the raspberry pi will decide to update the server if we only have "2/12 eggs", or we are "Almost out of milk". This will require the user to calibrate an "empty" and "full" weight for the item being measured by the system.
- The microcontroller will connect to wifi, and will interface directly with our server, sending it the quantity of tracked items we have on our fridge.
    - As previously mentioned, if the raspberry pi decides we are low on some product, it will update the server through the internet.
    - The Raspberry Pi will communicate with the remote server via an encrypted TCP/IP channel to guarantee confidentiality
- After our server receives product quantity data, if the quantity is below a certain threshold we will notify our users they are running low.
- There will be an Android application that allows the user to manually input products that they need to shop for
    - The application displays the current shopping list, which includes items that were automatically added based on the sensor data
    - The user can add or remove items to the list, or update quantities needed.
- Our database will be a simple SQL server instance and our data will be stored in simple key-value pair format.
- The Android application will also be the interface the user will use to compute optimal routes for what to purchase.
    - The optimal routes are computed by remote server and displayed on the application's user interface.
    - The user can choose whether to focus on time or cost.
    - These optimal routes will initially be perfect, which means that we will be using NP-hard algorithms initially (weighted set cover, TSP)
        - This will later be relaxed to be approximate versions of these algorithms

2.2 Design Analysis

Our proposed solution, explained in section 2.1 above, with the plan for each component explained in section 2.4, will theoretically address our problem statement. At this point in the project, everything is based on research, hence the details of each component or interfaces within the system are subject to change as we prototype and improve on the current design. The current design's strengths are its simplicity and low cost (minimal hardware costs). However, the current design's two biggest weaknesses are that it may not be user friendly (this is an area where we are still working to improve the design) and that it might have limited functionality for recommendations (which will be built more in the second semester of the project).

We plan to iteratively design each component. This means that the finalized designs may be completely different than the prototype designs that we have documented as of yet. For example, we may use a smaller, lower power microcontroller (the current design uses a full-sized raspberry pi) in a later design.

Our proposed solution does not address any security concerns. This includes encrypted communication; encrypting customer sensitive information, both in transit and at rest; preventing people from using a man in the middle attack with our devices; preventing tampering with our Raspberry Pis; and securing our databases. We acknowledge that this project does introduce security risks and concerns, and there are methods of mitigation, but our priority is to build the functionality for our use cases first.

2.3 Development Process

Throughout the Fall semester, we will be following more of a waterfall development process. This will and has allowed us to aggregate requirements, develop initial plans, and gradually provide plans at a finer level. Then, in the Spring semester, we will be relying on an Agile approach. An Agile approach will allow us to build an initial workable solution, then iterate on it to provide more and more features.

2.4 Design Plan

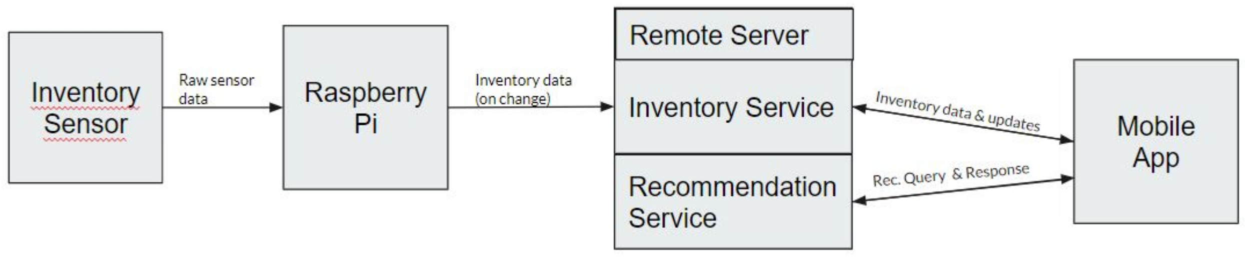


Figure 1: System Component Diagram

The planned handling of each use case is explained below:

- Automatically tracking supply or inventory of a household item, such as milk, and updating the shopping list to reflect a low supply
    - A pressure sensor (or scale sensor) is connected to a raspberry pi microcontroller. The microcontroller is constantly monitoring the measurement from the scale. When this component is first set up by the user, the user calibrates the sensor by setting the “empty” and “full” weight values. When the measurement from the scale reaches a threshold set by the user, the microcontroller sends an update to the remote inventory server with the new data.
- Maintaining the shopping list

    - The inventory server automatically adds items to a user’s shopping list when the quantity of that item (indicated by the inventory sensor data sent by the raspberry pi) falls below a certain threshold set by the user from the Android application.
    - The user can manually add or remove items to the shopping list from the Android application. They can also update the quantity of the items needed.
- Viewing optimal recommended shopping patterns based on contextual information and given priorities (the user can prioritize going to the closest stores, spending the lowest amount of money overall, getting the items as fast as possible, and other options)
    - The user requests shopping recommendations from the Android application. The user selects their desired cost to optimize (money, time, convenience, etc). The application then sends API requests to the recommendation service in a remote server. That server then computes and sends the application shopping recommendations. The recommendations are displayed in the user interface of the Android application. The user can accept a recommendation and the application will update the user’s inventory, assuming that the user completed the recommended shopping route and that they purchased the items.
- Viewing optimal recommended shopping patterns for multiple users (multiple people who can shop)
    - The implementation is similar to the above use case, but the location of all users who are shopping is included. The recommendation service finds an optimal combination of shopping routes so that all items on the shopping list are purchased and the routes are optimized in some way.

3. Statement of Work

3.1 Previous Work And Literature

A senior design team at Iowa State University worked on creating a “smart bin” device that measures the quantity of an item based on the measured weight, combined with software that helps businesses restock products more efficiently [1]. While this project aims to solve a similar problem, the user that we are targeting is a general consumer or shopper, rather than a business. The project had a higher cost than our anticipated cost (their project required $80 minimum purchase of microcontrollers and sensor modules, while our anticipated cost is under $60). In addition, the Multi-Context Shopping Optimization project includes the possibility of extending the optimization to multiple shoppers, which differentiates it from the past project.

3.2 Technology Considerations

Strengths and weaknesses different technologies that may be used for the project are listed below. Bolded technologies indicate that the team chose to utilize them.

- Microcontroller
    - **Raspberry Pi 4 (and Raspberry Pi Zero W)**
        - Strengths: The team has experience with projects using it. It is well documented and supported. It allows flexibility in programming languages. The board has all of the
        - Weaknesses: It's expensive compared to other microcontroller options. It has high power consumption and features that would not be used by this solution.
    - Other Linux-based microcontrollers
        - Strength: Most raspberry pi alternatives are cheaper than the raspberry pi. They also allow flexibility in programming languages.
        - Weaknesses: These boards aren't documented or supported as much as the raspberry pi or arduino.
    - Arduino Nano IOT
        - Strengths: The team has experience with projects using it. It is well supported and documented. It costs less than most other microcontrollers
        - Weaknesses: It requires us to write programs in C. The board may be tied to Arduino's IOT platform (it may not allow us to easily connect to our own services).
- Web Server
    - **NodeJS**
        - Strengths: The team has experience with NodeJS. It also supports open source libraries. Highly scalable.
        - Weaknesses: NodeJS has relatively new, immature tooling. Coding using it can get messy with callbacks.
- Mobile Application
    - **Android**
        - Strengths: The team has experience developing Android applications. Android is well-documented as a development platform and supports many open source libraries.
    - iOS
        - Strengths: It would allow more users to utilize this project's product.
        - Weaknesses: No one on the team has experience developing iOS applications.
- Data Storage
    - SQL
        - Strengths: Most of the team is familiar with SQL databases and querying. It is also fairly well supported.
        - Weaknesses: Constrains us to using relational data models.
- Cloud Computing & Modularization
    - **Microsoft Azure**
        - Strengths: The team has experience using Azure to deploy microservices and web applications. Azure also has a free tier for services that should

        satisfy our needs for this project. It has many options for app hosting features. Azure is well documented and has many tutorials for creating and deploying various web applications and services.
        - Weaknesses: Azure isn't as open-source friendly as other cloud computing options.
    - Amazon Web Services (AWS)
        - Strengths: AWS is more open source friendly than Microsoft Azure. AWS has many services and is a well documented platform.
        - Weaknesses: Team isn't as familiar with AWS. Encountered bugs in Aurora.
    - Docker
        - Strengths: The team has experience using Docker. There are also readily available container images.
        - Weaknesses: Can be somewhat confusing for anyone who hasn't used Docker.

3.3 Task Decomposition

The project is broken down into the following tasks:

- Develop android application
- Create inventory service
- Create inventory tables in database
- Create user service
- Create user tables in database
- Create store service
- Create store tables in database
- Configure raspberry pi
- Connect raspberry pi to inventory service
- Develop raspberry pi application

3.4 Possible Risks And Risk Management

Possible risks and mitigation methods for this project are listed below:

- Security: sending user's data over the internet (and over WiFi) can present a risk that the user's data could be exposed to others.
    - Mitigated by encrypting the data before sending it over the internet. This prevents the likelihood of someone sniffing the data and seeing sensitive user information.
- Authenticity: if attackers know the data format of the recommended stores, they can spoof the identity of the server and recommend non-existent or invalid stores to the user

    - Mitigated by encrypting the data and by using digital signatures to verify the integrity of the message as well as the identity of the source

3.5 Project Proposed Milestones and Evaluation Criteria

Key milestones for the project will be based on the use cases or features listed below. Each milestone is completed by the verification of the use-case functionality. Once we have reached all of these milestones, we will focus on refining and improving non-functional requirements, such as security, reliability, performance, and scalability.

- Usability tests
    - Android app allows user to modify checklist
    - Android app syncs checklist with server so it is saved across devices
    - Android app can display establishments having items from their list
- Functional tests
    - In-Home monitoring system can process sensor data
    - In-Home monitoring system can send data to remote server
    - In-Home monitoring system updates checklist when inventory runs low
    - Android app can receive updates from remote server
    - Server can process inventory data to find stores which have items on the list
    - Server can process location of user and determine if shortest path to nearby establishments

3.6 Project Tracking Procedures

We will be posting a weekly status report about every 10 days to track our progress made during that period. We will also be documenting our meeting notes after every meeting 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 those meetings.

Gitlab will also be heavily used to keep track of weekly progress through the issues board. 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

Expected Results

The desired outcomes include a system and a user guide which a user can use in order to accurately track and be reliably notified about items that are running low in their home. They must also be informed of nearby establishments which have the lowest prices and are in closest proximity to their current location.

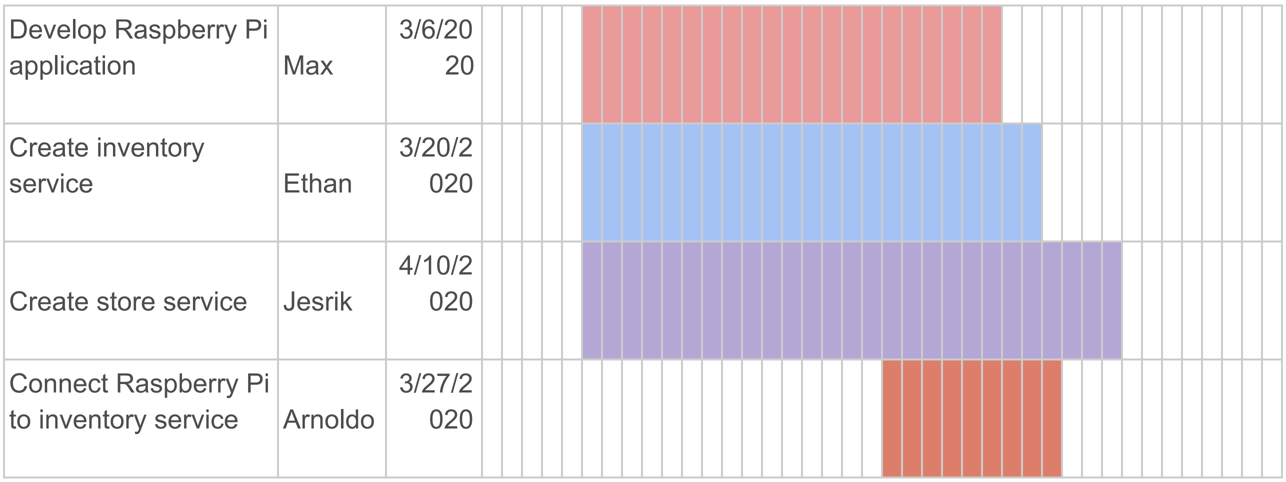
Validation

We will confirm our solutions work at a high level by providing test cases with actors performing actions that should trigger information to be sent to the user, we can test our product. We can describe an in depth use case that details each individual step that must be made. This includes actions that the user doesn't specifically have to take in order to demonstrate the automation of the product

4. Project Timeline, Estimated Resources, and Challenges

4.1 Project Timeline

Task	Main Contributor	Due Date	October	November	December	January	February	March	April	May
Develop Android Application	Karla	4/10/2020	🟩🟩🟩🟩🟩	🟩🟩🟩🟩🟩	🟩🟩🟩🟩🟩	🟩🟩🟩🟩🟩	🟩🟩🟩🟩🟩	🟩🟩🟩🟩🟩	🟩🟩⬜⬜⬜	
Create inventory tables in database	Nate	10/25/2019	🟦🟦🟦🟦🟦							
Create user tables in database	Nate	10/25/2019	🟦🟦🟦🟦🟦							
Create store tables in database	Nate	10/25/2019	🟪🟪🟪🟪🟪							
Configure Raspberry Pi	Ethan	10/25/2019	🟥🟥🟥🟥🟥							
Create user service	Nate	11/22/2019	⬜⬜⬜🟩🟩	🟩🟩🟩🟩⬜						



Develop Raspberry Pi application	Max	3/6/2020
Create inventory service	Ethan	3/20/2020
Create store service	Jesrik	4/10/2020
Connect Raspberry Pi to inventory service	Arnoldo	3/27/2020

Figure 2: Project Timeline Gantt Chart

The above diagram demonstrates the time frames associated with the various tasks as well as the person that will lead the effort for the given task. October is spent primarily setting items up while we continue to iterate on designing our hardware and software. The only items in October are setting up tables in the database, configuring the Raspberry Pi, and beginning work on the Android application. Starting in November, more complicated services and applications begin to be worked on, such as the software on the Raspberry Pi, the inventory service, and the store service. We plan to connect the Raspberry Pi to the inventory service once both services have been sufficiently developed independently to be able to support being hooked up. We expect to have all tasks done by late April, so that we can focus on preparing presentation documents prior to the end of the Spring semester.

4.2 Feasibility Assessment

We expect that our project will include all of the features outlined in our design section.

The routing algorithm is seen as a significant challenge to our project. It has to route users, which could be viewed as a traveling salesperson problem, but then it also needs to conditionally go to certain stores. We can simplify the problem by narrowing the range of possibilities so that the algorithm doesn’t need to consider every store. In order to choose which stores, it has to weigh the additional travel times of going to a store against the possible savings by going to that store. In order to view the possible savings of a store, there can be some subset of items that are cheaper at a given store, compared to another store. There are a lot of factors in play. Once a “perfect” algorithm (that always gives the correct, exact answer) is developed, it should be fairly straightforward to develop an approximate certain parts of the algorithm in order to lower the complexity of the problem from NP-hard to P. An additional option would be to make some sort of naive local hill climbing AI agent that could be expanded to have a higher chance of giving the user a “good” selection.

4.3 Personnel Effort Requirements

Task	Description	Time (person-hours)
Develop android application	This includes allowing users to create users, sign in, manually update items needed, and fetch the ideal route and items to buy along the route.	100
Create inventory service	Create a service that has APIs to update and fetch items needed for a given user.	50
Create inventory tables in database	Make the inventory tables in the database.	1
Create user service	Create a service that has APIs to update and fetch user information.	50
Create user tables in database	Make the user tables in the database.	1
Create store service	Create a service that can fetch pricing information for a variety of stores.	150
Create store tables in database	Create the tables needed to store the information about physical retail stores.	2
Connect raspberry pi to inventory service	Connect the raspberry pi to the internet and get it to send usage statistics to the inventory service	20
Configure raspberry pi	Make the actual physical device that goes in the fridge and reads data from a weight sensor.	50
Develop raspberry pi application	Write the software that the raspberry pi runs to communicate with the weight sensor and android	70

	application	

Table 1: Personnel Effort Requirements

4.4 Other Resource Requirements

In order to complete this project, we will need a Raspberry Pi. On top of a Raspberry Pi, we will need a weight sensor. We will be using the free service tier for Microsoft Azure. We don't expect to need to upgrade to a paid tier, but this issue may need to be revisited in the spring when we are testing the recommendation service with large amounts of data.

4.5 Financial Requirements

We need to purchase some raspberry pis for developing and testing prototypes. 2 Raspberry Pi 4 boards and 2 Raspberry Pi Zero W boards should be sufficient for our purposes. Additional circuitry parts or plastic housings may be required for the microcontrollers and weight sensors. We need to buy the weight sensor, which is typically available for around $10. The overall estimated hardware cost of this project is $200.

5. Testing and Implementation

Testing and test results are a critical indicator of the success of this project.

Testing of this project will be conducted as follows:

1. Types of Tests
    a. User acceptance testing will be conducted to validate that requirements are met. Functional requirements will be tested.
    b. System testing will be conducted to ensure that all of the components integrate with each other properly. Non-functional requirements such as performance, reliability and scalability will be tested.
2. Items to be Tested
    a. Raspberry pi & weight sensor device
    b. Android application
    c. Remote server
        i. Inventory service
        ii. Store service
        iii. Recommendation service
3. Test Cases
    a. Specific test cases will be created during the second semester of the project. General test cases are described in sections 5.3 and 5.4.

4. Test Documentation
    a. The testing process and procedures will be standardized. Once finalized, the testing process will be documented.
    b. Results of tests will be documented (problems or areas of improvement will be noted so that the next revision can address them).

5.1 Interface Specifications

Sensor to Raspberry Pi

The weight or pressure sensor and the raspberry pi will be connected via the microcontroller's GPIO.

Raspberry Pi to Remote Server

Communicate via an encrypted TCP/IP port and exchanging raw bytes. The format of the bytes sent over the port could be specified by us, or by utilizing a transfer language such as Protocol Buffers for compressing and packaging the data in distinct, unambiguous ways that can be processed by many languages.

Remote Server to Android App

Using a similar method as those specified between the Raspberry Pi and the Remote Server.

5.2 Hardware and software

We will use the following hardware to test and develop the project:

- Android phone to test the Android application.
- Raspberry pi to connect to the weight or pressure sensor and relay data to our remote server

The following software will be used:

- Android Studio provides an emulator for any team members without an Android phone
- Postman provides a way for us to test the REST API calls on our server
- MySQL Workbench will be used to test storage is taking place on our databases

The end product of this project will result in one instance of the remote server (running multiple services) and many instances of sensor devices (many users having many sensors). This also includes many mobile devices interacting with the system.

5.3 Functional Testing

Functional testing will be conducted by user acceptance testing for each use case, as described below:

- Automatically tracking supply or inventory of a household item, such as milk, and updating the shopping list to reflect a low supply
    - Testing method: set up the scale with the raspberry pi. Calibrate the full and empty weight settings, and then simulate the quantity of the item being depleted (by using lower weight than the previous weight). The inventory data in the remote inventory server should reflect the change when the user views their current inventory in the app. When the simulated weight falls below the set threshold, the item should be automatically added to the shopping list.
- Maintaining the shopping list
    - Testing method: test that the user can add, remove, or update the quantity of items on their shopping list. This list should be stored on the inventory server and viewable in the android application.
- Viewing optimal recommended shopping patterns based on contextual information and given priorities (the user can prioritize going to the closest stores, spending the lowest amount of money overall, getting the items as fast as possible, and other options)
    - Testing method: test that a user can request shopping recommendations via the android application. The user should be able to view recommended shopping routes in the user interface, and when they complete a route, the inventory should automatically be updated to reflect the newly bought items.
- Viewing optimal recommended shopping patterns for multiple users (multiple people who can shop)
    - Testing method: test that multiple users can be given recommendations in the android application that optimize the shopping routes for multiple shoppers, and not just a single person.

5.4 Non-Functional Testing

System testing, via unit tests, stress tests, and monitoring, will be conducted as described for each area below:

Performance Testing

- Monitor the network use while the In-Home system is active
- Monitor the network, battery, RAM, CPU, and storage usage on the smartphone when the app is running and when the app is in the background

Security Testing

- Use a packet sniffer to try to gain access to user information

- Use SQL injection and buffer overflow attacks to try and break both client and server code

Usability Testing

- User testing on outsiders who are unfamiliar with our project can indicate whether the project and the user interfaces are usable enough or if they need additional explanation.

Scalability Testing

- We are initially focusing on smaller data sets to simplify the scope of the recommendation problem (otherwise we are dealing with an NP-complete algorithm problem). Later on, we can introduce more data and possibilities for recommendations and test the algorithms that we've designed. We can also simulate a large number of sensors or users to see how our system scales.

5.5 Process

When we begin testing prototypes, we will iteratively test the prototype and make improvements until we reach a finalized design. This allows us to build confidently on our design and utilize our past successful work. A diagram of our design and testing process is attached below.

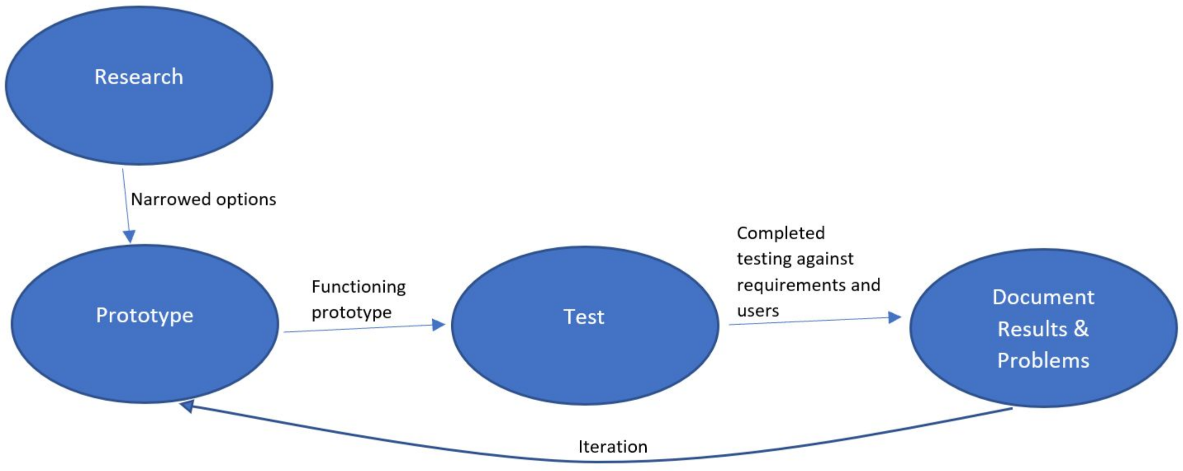

Research
Narrowed options
Prototype
Functioning prototype
Test
Completed testing against requirements and users
Document Results & Problems
Iteration

Figure 3: Project Testing & Iterative Development Process Diagram

5.6 Results

We are currently in the research phase of the project, so we have not conducted testing. However, soon we will begin prototyping and testing iteratively. The results with each iteration will be documented.

6. Closing Material

6.1 Conclusion

Up to this point we have done extensive research regarding tools and technologies that we can use to help us both develop a solution to our problem statement as well as aid us in communication and technical discussion.

We have begun drafting the architecture of our whole system including communication methods between the different components as well as the data collection paradigm that the sensor network will use in the user’s home.

We have laid out some deadlines for different aspects of the subsystems of our project to help drive our development and force us to meet deadlines.

From here, we plan to begin some rudimentary prototyping to approach our goal of a proof of concept by the end of the semester. Our hope is that in prototyping we can come up with shortcomings of our approaches before placing too much of our effort in solving a problem which is intractable.

6.2 References

[1] A. Hauge, D. Bis, S. Guenette, H. Moser, N. Bix and B. Gruman. *Automating Inventory Management: Routing through Sensor Networks*. Iowa State University: 2019. http://sdmay19-29.sd.ece.iastate.edu/