Subway Inventory Management System

Design Analysis Report
SYDE 461 - Group 16

Andrei Grovu - 20831978
Archan Patel - 20834719
Moiz Khan - 20828670
Sayohn Ahilan - 20842344
Suyash Unnithan - 20821057

December 5th, 2023

Table of Contents

Lay Summary

This report outlines the Subway Automatic Inventory Management project being done by Systems Design Engineering students.

Inventory is a very tedious process that is conducted at most businesses. Automating this process is a project that could potentially save several hours of time per week. Ultimately, this will save businesses money and labor. For the scope of this project, we want to just focus on Subway inventory as it is a business that performs a tedious inventory management process.

This project is to be conducted from September 2023 to April 2024 and this report has been written at the halfway point, December 2023. Using resources provided by the University of Waterloo and purchasing additional equipment, the project has been developed to a medium fidelity stage.

The report also showcases the technological aspects of this project. The project can be broken down into 3 big technical components: Vision Model, Hardware Weight Sensing, and User Web Application. These 3 components work together to create an inventory management system that is 100% automatic.

Before April 2024, the team is planning to integrate all components together more efficiently and improve the performance of each component through rigorous testing and verification/validation of user requirements and engineering specifications.

I - Introduction

The Subway organization offers a franchising opportunity for entrepreneurs looking to start their own subway restaurant. This involves a comprehensive process that begins with an application and an evaluation of the potential franchisee's qualifications. Once approved, franchisees receive training on Subway's operational standards, menu preparation, and customer service. The Subway brand provides ongoing support, including marketing strategies, supply chain management, and continuous training to ensure the success of each franchise.

One of the primary tasks conducted by franchise owners is the management of their inventory. Inventory management refers to the concept of ordering, storing and using a company's raw supplies [1]. Typically, Subway owners store their raw supplies in either a dry storage room, refrigerator, drinks refrigerator, or freezer. Currently, the management process is done through the manual counting of raw supplies on storage shelves placed in the three described locations. The results of this counting are stored on a mobile application called SubVentory which is an application developed by Subway to help franchise owners along the inventory management process [2]. Every Tuesday, franchisees order their required products from distributors by manually entering the product and its quantities based on what is needed. They determine what is needed by viewing their live inventory count on the SubVentory app. Through conversations with industry connections, it was determined that owners spent approximately 4 hours per week counting their inventory, putting the data into SubVentory, and placing orders.

Subway franchise owners also find that manually tracking inventory is extremely prone to mistakes since a large variety of items must be counted and because predicting required inventory levels while accounting for stock perishability and one-off inventory requirements (holiday spikes, special offer spikes etc.) is an extremely complex process. The accuracy and efficiency of such operations can be increased by automating them while also saving a substantial amount of time and money spent.

With that being said, our team is designing an automatic inventory tracking platform to be used by Subway franchise owners. This design aims to ease the users inventory management process and enable subway franchise owners to utilize their time elsewhere. The platform should enable the franchisee to view their inventory or a subset of their inventory without having to do any manual counting, and should provide the user with a suggested inventory order based on previous demand, item perishability, and upcoming special events that in turn affect demand.

In order to develop the automated inventory tracking platform, multiple new approaches for using existing technologies are required to be invented. For example, a new system must be created that enables the interoperability between multiple weight sensors, cameras running AI vision algorithms and cloud front-end applications in order for the system to calculate and display the current inventory levels. Further, a new model is required to be created as to how to merge a series of existing inventory suggestion methods (that are specifically designed for varying product categories) in order to create a singular suggested inventory order for the user.

This platform creates positive economic and social impacts as Subway franchise owners are no longer required to manually conduct inventory counts, saving their own time or the salary of a manager to do this work item. Further, they also receive a positive economical benefit as manual inventory

tracking and prediction methods are prone to errors, leading to losses of inventory due to inventory punishments and losses of sales due to under-ordering. There are also positive environmental benefits that arise from the decreased amount of trashed inventory due to increased accuracy between supply and demand created by the platform. In addition, there will also be a decreased emissions created by inventory fulfillment trucks as the amount of orders may reduce.

However, it is important to note potential negative impacts of our system, such as the dependency on technology for users to be able to conduct inventory checks and orders. This may also create issues for its users if it is not adaptable to the continuously changing inventory requirements of Subway locations as menu items rotate and change. Other than the initial investment costs for purchasing and using the automated inventory platform, it may also lead to employee displacement as less of a manual load is required.

Overall, as the positive impacts seemingly severely outweigh the potential negative impacts, this platform stands as a promising potential improvement for Subway franchisees.

II - Engineering Methods of Design and Analysis

Table 1: Engineering Specifications

Specification	**Target Value**	**Justification**
Vision Model Accuracy	Accuracy >= 95% False Positives (FP) < 5%.	False Negatives (FN), means that our system predicts that we are missing an item when it actually is there. This will lead to overstocking an item which is better than understocking it. We can obtain this through a conservative bias in the system. Promote FN → over stock our inventory
Camera Resolution	At least 1000p x 585p	Camera Resolution is dependent on the number of pixels required for the small feature being analyzed. (see Appendix)
Camera Focus Field of View	165 degrees (horizontal), 91 degree (vertical)	Based on Camera Resolution and Camera distance calculation. (see Appendix, Camera Resolution calculation)
Shutter speed	At least 25 fps of a second	Subway inventory room is not a dynamically changing environment. Thus high shutter speed is not required. Low shutter speed == more light [3].
Light requirements	Color: Cool 5500K – 8000K	Lowest feature contrast: Brown box with black letter → increase contrast [4].
Camera Working Distance	1.3 m to 3.9 m away from Shelf	Optimal working distance should be 1-3x away from relative shelf size [5].

Specification	Target Value	Justification
System Processing Time	< 1 Minute	Inventory count will update every 30 minutes. Ideally we have at least 30 data points within each vision update to provide an accurate median reporting of inventory count.
Data Storage and Analysis	<512MB	MongoDB shared cluster configuration is free [6]. No image or video storage, so sufficient storage space.
Number of Human Operators Required	0	Fully-automated vision system. Once set up, no human intervention should be required besides troubleshooting.
Hardware working temperature	Min: -5 degrees	The minimum temperature in the freezer.
Weight Range for Load Cell	20-40kg	Subway box weight range is 5kg to 20kg. Increasing the cell's weight capacity beyond this ensures the weight sensor reliably manages unexpected loads and resists deformation.
Creep % for Load cell	<0.1	A low creep in our weight sensor is beneficial as it provides stable readings over time, reducing the need for frequent recalibration. Consistent creep allows our software to adjust readings accordingly, enhancing measurement reliability.
Accuracy for Load Cell	<+/- 0.1%	The minimum item package weight is 0.2kg, this is a relatively accurate load cell. We require accuracy for reliable inventory levels, waste prevention, and restocking optimization.
Load Cell Material	Aluminum	Aluminum is selected for the load cell material for its lightweight, corrosion resistance, and durability, essential for reliable performance in the variable conditions of a storage room environment.
Cost for Load Cell	<$30 each	A relatively cheap load cell is required as for a full size shelf we would need numerous weight sensors to be constructed

A) Design and Analysis

Starting the semester, our team had a plan of building a smart financial assistant for small businesses. The idea was to help businesses manage their finances and their financial goals easier through harnessing the power of AI. We decided to not finalize the scope of the project as we wanted to first get feedback and approval from our professors. Upon presenting our project to our professors and classmates, we were told that there was no clear scope and that we should focus on a problem that has more impact on the end-user. This led us to start thinking more about our end-user and what problems they face on a daily basis. This prompted a brainstorming session in order to both generate ideations that would help form the solution space and to understand the interests and skills of our team members. All responses were collected anonymously in order to encourage participation and reduce bias. Responses were also collected into groups of natural relationships following affinity diagramming in order to view connections between ideas and explore possible ways forward.

Figure 1: Brainstorming Session alongside Affinity Diagramming (A.G, M.K, A.P, S.U., S.A, 2023)

Conversations with our industry connection, a Subway multi-franchise owner, enabled us to understand the scope of our project better. They gave us store specific information that would help us validate our constraints and requirements. During our interview with them, they explained to us the 4 different sections of storage. They are, Freezer, Dry Storage, Drinks, Fridge. Conducting this interview with them gave the team much needed insight in order to empathize with our potential users. The interview also enabled us to figure out how we want to scope the project. They introduced us to their current inventory management system on Subventory. This led the team to conduct a literature review of existing inventory management solutions.

Subventory is Subway's current inventory management app. Employees are expected to perform inventory inputs on this app every week. The team reviewed the documentation provided by ZippyYum (the developers of Subventory), to grasp the current processes [7]. This documentation provides a quick and easy way for franchisees to transition from their previous pen and paper method to Subventory. This helped the team understand what aspects of the inventory management system are actually being handled by Subventory. After viewing this documentation and conducting the interview with our industry connections, we decided to develop our first iteration prototype on only dry storage items, but eventually make a working system for all ingredients in the 4 different sections mentioned above.

Another inventory management solution that our group took a look at would be Zoho Inventory. Similar to SubVentory, Zoho inventory is another software-based tool used to manage both retail, industrial, and manufacturing inventory. Zoho allows users to efficiently track stock levels, manage multiple warehouses, and automate reorder points to prevent stockouts [8]. The system provides real-time insights into inventory movement, allowing businesses to make informed decisions on restocking and avoid overstocking issues.

The biggest takeaway from looking at existing inventory management solutions was that despite the fact that these softwares features a multitude of automated features, all of these solutions still require human intervention for the system to work correctly. Whether that means the user has to manually input the inventory into the system or the user has to scan each inventory box with a scanner, the systems require a human to monitor and start the management processes. Upon discussing our findings with our industry connections, we learned that human intervention was a pain point for them in the management process. This is because having a user overlook the management process was a waste of vital resources. The overlooking of the inventory management process is a 3-4 hour process on a weekly basis which is time and money that can be spent on sales, marketing, and managing the store.

Taking the existing solutions and conversations with industry connections into consideration, our team was able to finalize the scope of our project. We wanted to build an inventory management system that encompasses the following design features.

1. Track inventory stock levels across all 4 storage locations
2. Software solution allowing user to view live inventory count and place inventory orders
3. No human intervention besides users placing boxes on current inventory shelves
4. Automatic ordering based on supply chain engineering concepts

Our preliminary design concept featured three sub-systems: vision, hardware, and software. The vision and hardware system would combine to remove human intervention of manually entering or counting inventory boxes. The human intervention in the original inventory management system is that they are required to see which inventory boxes exist on the inventory shelves and how much content is inside these boxes. In our new management system, the vision system will determine which boxes exist on the shelves through a camera facing the inventory shelf that is integrated with a machine learning model that is trained to classify boxes and labels.

The hardware system would be a weighing scale that measures the contents of an inventory box in the form of mass. The final sub-system is a client-facing web-application that allows Subway owners to view inventory and place orders.

Vision:

There presently are numerous Computer Vision platforms, many of which are open-source, free-to-use, and come with plenty of documentation and online support. For the purpose of this project, our team decided to test out different platform with specific Vision functions:

- Tesseract: OCR (Free)
- Roboflow: Object Detection, Position, OCR (Free/Paid Services)
- GPT Vision: Object Detection, Position, OCR (Paid Services)

The first two tools we tested out were Tesseract and Roboflow. These tools are widely used in the machine learning and computer vision industry, and are free to use for University students.

In a carefully monitored experimental environment, the Tesseract OCR model [9] that is included into the OpenCV framework underwent extensive testing. A methodically designed process was used to create a uniform setting for assessment. Fifty distinct photographs of subway ingredient boxes were captured, so that every picture had the same phone position and box location (i.e., distance from the lens). A Philips LED A19 E26 60W light bulb was used to replicate different illumination situations [10]. This evaluation's main goal was to determine how well the model could identify the inventory boxes when it was detected and then identify any text characters on the box after detection was successful.

Moreover, we expanded our analysis to include Roboflow, a substitute framework that has similar libraries to OpenCV in that it makes use of preset training models (Yolo, detectron3, etc.). The training set for Roboflow included the same 50 photographs that were now annotated (i.e., outlining boxes' shape) and were divided into three subsets, with a distribution of 65% for the training set, 20% for the validation set, and 15% for the test set. This allowed for robust model construction and evaluation in order to train the model efficiently. Roboflow and OpenCV provide a consistent image processing framework that includes uploading images, annotating them, and then training them. In order to explore a different way to perform computer vision analysis, the team decided to explore GPT's vision capabilities in our search for unique methods in computer vision.

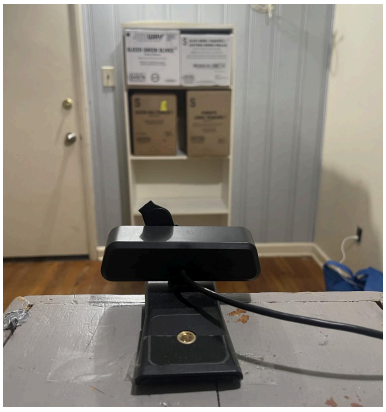


Figure 2: Shelf Set-up: Prototype 1 (M.K 2023)

At the same time, the team created a new bench arrangement (prototype 1) to improve our vision system. A Ymiko Webcam with a 2.54MP sensor with a maximum resolution of 2560 x 1440p was included in this configuration [11]. The webcam was placed exactly 1.41 meters away from a white shelf and was carefully centered around it. This shelf offered an organized space with four distinct areas for box placement. We started an empirical investigation into ChatGPT's vision capabilities. ChatGPT was given prompts to instruct what the camera is expected to observe , such as which boxes with different color profiles on the white shelf and where they exactly were in relation to the given coordinates (i.e., a 2D array, column vs row).

Hardware:

When deciding to make a weight sensor the most important aspect is to consider what load cell and load cell configuration will be used to collect all the weight data. While doing research on load cells available to purchase, it became very evident that the price constraint that we set would be quite difficult to meet as quality load cells were very expensive. There ended up being two options in the current price range, Option A: A four-wire full bridge strain gauge load cell, Option B: A three-wire half bridge strain gauge load cell. Comparing both of these based on our constraints and other factors was a useful tool to help decide which load cell would be more effective in our use-case.

Table 2: Decision Matrix for Weight Hardware

Specification	**Option A**	**Option B**	**Observations**
Required Load Cells for 1 Unit	1	4	Option B only has half wheatstone bridges and is balanced on a sphere, therefore a minimum of 4 cells can be used to create a fully balanced sensor plate.
Weight Range per Load Cell	20kg	25kg	Option B has a slightly higher range and with more cells it can be adjusted to be even higher, this may add a lot of complexity.
Creep % for Load cell	0.01%	0.1%	Option A creeps much less and therefore would need to be calibrated less often.
Accuracy	0.03%	0.2%	Option A is almost 10 times more accurate.
Temperature Range	-20~+65°C	-10~+60 ºC	Option A is able to withstand a slight larger temperature change. It can function more easily in cold temperatures than option B.
Cost	$15	$13	Option B is slightly cheaper
Material	Aluminum	Polycarbonate	Option A is made of a more quality material. Aluminum is known for high tensile strength (about 33% higher than PC) and durability [12].

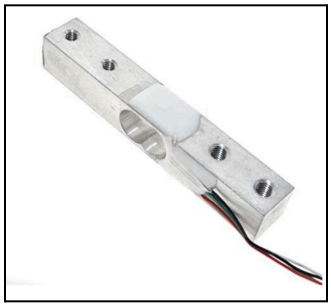


Figure 3: Option A Load Cell [13]

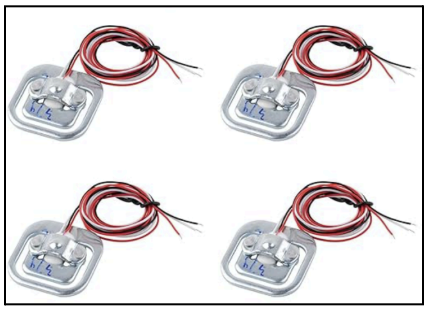


Figure 4: Option B Load cell [14]

Weighing all of these factors it becomes apparent that option A is more feasible for this design. It has a higher accuracy, better quality material and lower creep which means that weight data collected will be more accurate and consistent over time. It also only requires the use of one cell, as it has a full wheatstone bridge implemented unlike option B which requires four, which will significantly reduce the chance of human error in our setup and the lower the complexity of recreating the weight sensor. While option B is cheaper and is rated for more weight to be applied, the differences are reasonably negligible. Since option A is to be implemented, a customized 3D-printed component was created to maintain consistent pressure on the load cell and minimize instability, leading to a higher level of confidence in the precision and reliability of weight measurements. This component was created using SolidWorks and consists of a support that sits on the shelf and will stabilize the load cell, it also consists of a top plate that sits on top of the load cell and allows it to deform based on the weight applied.

Figure 5: 3D printed components to build weight sensor (A.G 2023)

Software:

This aspect of the project will allow the user to interact with the live inventory management system through the app, SubwAI.

The initial wireframes for this app were done on Miro. Miro is an easy to visual workspace for innovation. Low fidelity and medium fidelity wireframes were constructed to figure out the layout and navigation of the app.

The actual development of the app took place using a React web framework called, Next.js. It is used by some of the world's largest companies, Next.js enables you to create full-stack Web applications by extending the latest React features, and integrating powerful Rust-based JavaScript tooling for the fastest builds. It offers great performance and optimized developer experience. It also integrates well with Mongo.db and Google Cloud API. As a bonus, Next.js has one of the best documentations for a framework.

Agile methodologies are practiced using Github. Github is responsible for SubwAI's version control and collaboration. Referencing Next.js' documentation was pivotal to overcoming roadblocks and furthering the development of the app.

III - Project Outcomes to Date

A) Progress and Outcomes from Design and Analysis

Vision:

WEIGHT	13%	20%	33%	33%	100%
OPTIONS	Easy to Use/Set-up	Variability of CV Libaries	Collobarative Environment	Vision Train Efficency	
OpenCV	1	5	1	2	2
Roboflow	5	3	5	4	4
ChatGPT	5	1	5	1	3

Figure 6: Decision Matrix of Computer Vision tools (M.K, A.P. 2023)

The Tesseract OCR model in OpenCV had similar performance to the Roboflow's OCR model. While Roboflow is an online platform, Tesseract is a defined model part of the OpenCV library which requires a lot of initial set-up work to run the OCR model. On the other hand, Roboflow's platform has the ability to run multiple different vision models and integrate with our software backend with an API call. Roboflow offers fewer options for different varieties of vision models, but it requires less integration work and the ability to train the model on an external VM without any additional setup work. Thus, training models through Roboflow is relatively faster than the Tesseract OCR model.

Mid-way through the semester, OpenAI released GPT vision which is a complex vision model based on GPT4. We decided to test using this model for our vision system as it would be much easier to tune it for OCR and box placement. Using the OpenAI API routes, we tuned the model using a series of different prompts (can be found in appendix) and tested it with a different combination of images from our testing dataset. We found that the GPT vision model was extremely accurate in reading the labels on boxes, however, it did not reach 95% accuracy in terms of classifying the position of the box on the shelf. We determined that in order to move forward with the GPT model, we would require the hardware component to determine the box position on the shelf which is completely doable.

Our team also determined that some form of pre-processing on images improved the accuracy of all our models. Processing steps included cropping based on predefined coordinates of the assumed location of each box and then adjusting the contrast on the image (code located in the appendix). After some preliminary testing, it was observed that adjusting the contrast on the image showed promising results. Future tinkering is required to come to a consensus on the pre-processing of the image.

Hardware:

For our final weight sensor iteration a single point strain gauge load cell with an HX-711 amplifier and arduino was used. This load cell is constructed from aluminum, is rated for 20kg and has a high accuracy (±0.03%)[15]. Due to it meeting all of our originally listed constraints it was chosen to be implemented in our weight scale system. This device operates by applying a load to one side of the cell, causing deformation that is detected by strain gauges. The deformation creates an imbalance in the load cell's embedded Wheatstone bridge circuit, outputting a corresponding voltage output. This voltage is fed into an HX-711, which converts it into a readable weight figure based on what units we chose to calibrate it in. Finally, the Arduino processes and displays this weight measurement with the help of some code.

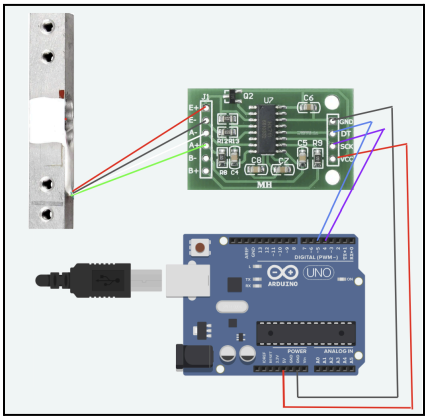


Figure 7: Circuit diagram for load cell (A.G 2023)

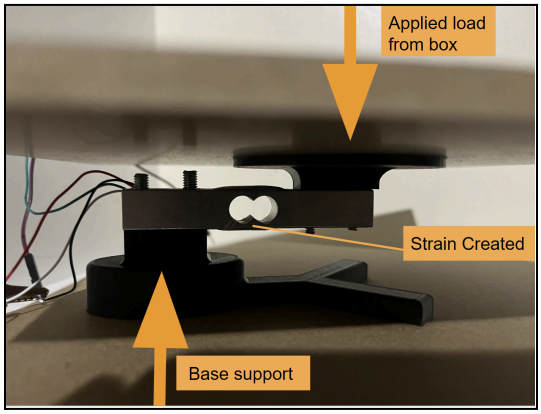


Figure 8: Picture of load cell in place on our shelf (A.G 2023)

As shown in the figure below the weight sensor component is placed under a thin wooden plank, which is slightly smaller than the shelf area it is covering. This ensures the weight is evenly applied on the load cell as long as the box is pushed fully into our shelving unit.

Using code in the arduino we are able to calibrate our load cell to accurately measure the weight of our boxes. This calibration works based on two functions provided in the arduino library *HX711_ADC* that is compatible with our HX-711 amplifier [16]. LoadCell.tareNoDelay() is a function that allows us to tell the load cell what deformation represents zero weight, this helps us ignore any initial weight on the top plate that isn't the actual box. LoadCell.getNewCalibration() is a function that allows us to calibrate the load cell based on a known mass, which we have chosen to be a 5kg metal weight. Using these tools we are able to obtain accurate measurements within +/-t 1.5g, which can be ignored. While this error slowly rises as the weight sensor is left in use for a longer period of time, routine calibration will ignore this creep. This weight data in our serial output is captured and saved in a CSV file that is routinely transferred to our vision hardware server which will eventually relay the data to our software backend.

Vision Hardware Server:

In order for our system to coordinate the process of capturing images, running the vision model, reading weight-sensor values, and communicating with our software layer, it was decided that a server would be built. A light-weight framework that allowed us to send HTTP POST requests, HTTP GET requests, schedule functions, and run Python libraries was required and hence, a decision to use Python's Flask framework was made. Using this framework and libraries such as APScheduler, a server was built that captured a photo and ran the vision model every minute. This server also reads the weight sensor values.

Software:

The first consideration for the designing of this app was whether to make it a mobile app or web application. Mobile apps are installed and downloaded via an app store on the users phone and they run specifically on that device. Web apps are hosted on the internet and don't need to be downloaded. They can be accessed on computers and mobile devices [17]. Ultimately, proceeding with a web app was the decision that was made. The weighted decision matrix [18] can be found in the Appendix.

The framework selected is Next.js as that is an up and coming framework. As seen in the Appendix under 'Architecture,' API routing is a crucial component of the project. A perk of using Next.js is that it provides a built-in way to create your own API Routes and endpoints [19].

The app is at a low fidelity stage right now. The user interface is functional and has many key features developed. A user survey was taken to determine the perception of the user interface from users. The questions and submission results of the survey are in the Appendix.

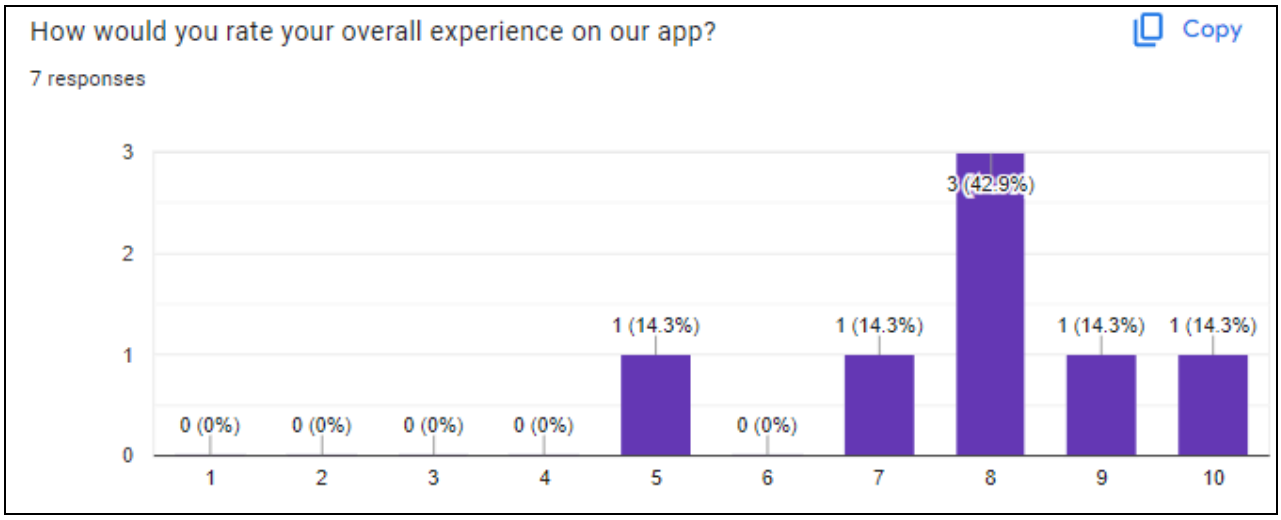


Figure 9: Result of last question on user experience survey (A.P 2023)

This question aimed to gauge the overall experience of the users. The app got a respectable mode rating of 8, and a mean of 7.86. This rating is below the team's satisfaction threshold. User interface changes are required.

B) Promoted to 462

Vision:

Going into the next semester, we are going to promote our GPT Vision model design and our Roboflow Vision Model. Our plan will be to continually test both models and determine the best plan of action as we scale our hardware. We will try our best to meet the engineering specifications that were defined for the vision system.

As previously mentioned, we have only been testing our vision models with dry storage items. During our gallery walk and conversations with our supervisor, a great point was brought up which is that we need to expand our system to work with storage items from all storage locations (dry, freezer, refrigerator, drinks refrigerator).

Hardware:

The focus for developing our hardware next term will involve weight sensor placement and hardware organization. While the current configuration of the load cell works well, one challenge is that it is quite tall from base to top plate making box placement slightly awkward. A raspberry pi will also need to be added to our shelf during 462, so the placement and connectivity of that piece of hardware still needs to be tested. Also more testing in cold conditions would need to be done to ensure that fridge placement is viable in practice.

Software:

The focus for SubwAI in Syde 462 is to improve the user interface. The overall rating of experience for our app, as referenced in Figure 9, should be greater than a mean of 9 for the team to be satisfied.

Interface concepts learnt in Syde 452 Interface Design, such as Fitts Law will be applied. Based on user feedback the app has room to improve. Utilizing tools like moodboards, icon design, typography and storyboarding will enhance the application.

The largest component to tackle in the upcoming semester will be ensuring that all subsystems are able to communicate with one another. As of now, the hardware and vision server connects the vision model and weight sensors together on one platform. Our next step will be to build API routes on the software layer such that we can transfer data from the hardware and vision server to the backend server. This will allow our software layer to store any necessary data, perform business logic, perform inventory computations, and display live data to our clients through the frontend layer.

IV - Consideration of Safety, Ethics, Standards, Regulations, Sustainability

Adhering to the Canadian Standards Association's (CSA) guidelines, namely C22.2 No. 61010 [20] and C22.2 No. 0:20 [21], is essential to ensuring that electrical and fire safety regulations are strictly followed during the system's development. For example, when designing the weight sensor device, great care has been taken to ensure that wire insulation, grounding, and protection against external electrical hazards. Moreover, during system testing and design, precautions for adequate ventilation and fire suppression mechanisms were taken. These procedures are essential for the safe use of electrical components and the avoidance of possible fire hazards. This is a critical factor to take into account as the system moves from testing dry storage items to other diverse item types in the upcoming months.

On the software side, we must consider data security, reliability and privacy as this data is confidential and extremely vital to the users daily business. One such regulation is PIPEDA [22] which describes how the platform must obtain an individual's consent when they collect, use or disclose that individual's personal information. This influenced our design towards making use of a series of proven open-sourced and pre-built systems that provide extreme amounts of functionality. For example, authentication methods such as Google Authentication for the application and hosted MongoDB authentication on the database were chosen in order to secure user data. Further, hosted MongoDB databases ensured that user data was reliably accessible whenever required. For sensor reading, we have embraced Arduino boards to strengthen our methodology even more. These adaptable, affordable instruments facilitate comprehensive assessment of sensor capabilities, guaranteeing precision and anticipatorily mitigating any vulnerabilities. This decision also enables us to secure sensitive data all the way from the beginning at the sensor level.

A fundamental component of our design methodology is the deliberate incorporation of sustainability principles. The utilization of modular and interchangeable components guarantees

minimal downtime and cost-effectiveness while also enabling quick repairs in the event of errors or damage. Choosing hosted cloud systems over proprietary servers also helps achieve sustainability objectives because they minimize energy consumption, have a smaller environmental impact, and provide scalable, energy-efficient solutions. This strategy ensures proper lifecycle management, reduces the need for hardware, and encourages energy conservation — all of which are essential elements for developing sustainable practices in small business operations.

There is a lot of potential for the automated inventory tracking system engineering design that Subway franchises are implementing to have a variety of positive effects on the economy, society, and environment. Its social effect is that it relieves franchise owners of laborious manual inventory duties, giving them more time to concentrate on customer service or other business-related activities, which may improve work-life balance and job satisfaction. Economically speaking, this invention could greatly reduce operating expenses related to mistakes in inventory management, increasing franchise profitability. Furthermore, the platform could support environmental sustainability by minimizing discarded inventory and reducing waste through accurate prediction-based inventory optimization, which would lessen the platform's negative environmental impact.

The automated inventory tracking platform's design and solution strongly align with the Professional Engineers Ontario (PEO) Code of Ethics [23]. Our goal to save time and money for Subway franchisees through process streamlining is a clear indication of our dedication to the public welfare. Furthermore, the platform's emphasis on reducing waste and improving supply chain management is a perfect fit with our obligation to uphold environmental standards. Throughout the design process, we ensure to treat stakeholders and experts with respect as we have a personal connection to the project (as one of our team member’s parents are our industry connection). This tight relationship also allows us to maintain integrity by enabling us to frequently reach out and seek advice from expert users when required and conduct user testing.

V - Conclusions Drawn From Outcomes to Date

Based on the design outcomes described above, the team is just short of meeting the goals set for SYDE 461. The integration between all aspects of the project still needs to be done. At this point the individual parts of the project are complete at a medium fidelity level. With a week's worth of time we may be able to meet our initial goals set at the start of the term.

Based on the work spent building the vision model, we believe that training a vision model to accurately classify the position of the box on a shelf in grid coordinates is a very hard challenge to meet. This is because vision models often lack the spatial awareness to guess a position on a shelf. We build an extremely large training dataset (1000+ images), it may be possible to achieve this, however, it would only work on the shelf that the model is trained on. When we plan on using an industrial size shelf during our evaluation period, our model will most definitely fail in correctly assigning position. With that being said, we think that it would be a better design for us to use the weight sensors to classify box positioning. This is ideal as our weight sensors will already exist on the shelf in a grid setup (X sensors

per row). Since we know the position of each sensor on grid, we can combine this information with our vision model output to determine each box's positioning.

Through practices such as wireframing and iterative designing, the SubwAI app is at a point where it is functional with a mean score of 7.86 average user satisfaction. Using concepts learnt in SYDE 542 Interface Design, the app is on track to meet the goals set out for SYDE 461. However, for next semester this rating must be greater than or equal to a mean rating of 9.

VI - Planning for SYDE 462

The main focus for our group during SYDE 462 is harmoniously connecting all of our software and hardware that has been developed this term. A RaspberryPi 4 is being implemented as the point of contact for our vision and weight sensing systems. This choice allows for efficient communication and data exchange between these components, forming an interconnected system. Since we are currently using a personal laptop to do all of our data processing and software management, the RaspberryPi will offer a compact and highly process-capable solution.

Another high focus for our group will be to finalize some characteristics of our vision model solution. This includes determining whether to use a combination of the vision system and hardware system to determine inventory contents and position or whether we should just use the vision system.

We also plan to emphasize more heavily on the supply chain and inventory management logic behind our application. Currently, suggestive orders do not account for item perishability and/or special cases of demand requirements (i.e. holiday deals driving demand for specific products, storms causing reductions in demand, etc.). We also want to build out the ability to edit the suggested order and save this in a meaningful format that can be sent off to Subway's inventory systems.

Scaling to a full shelf will be the most crucial aspect of our working during 462. Multiple weight sensors need to be created to fill a full mock-shelf. This setup allows us to optimize the calibration and placement of the sensors to ensure uniform and accurate weight measurements across the entire shelf. Fine-tuning the sensors in this way is crucial for achieving precision in inventory management. The camera placement will also need to be re-evaluated based on this more realistic and larger shelf case. This expansion is crucial for simulating real-world scenarios where shelves are fully stocked, enabling us to test and refine the performance of our vision and weight sensing systems in a more realistic setting.

Acknowledgements

We want to express gratitude to our capstone supervisor, Jim Bookbinder, for his insight and knowledge into inventory management. We also want to express gratitude to our industry connections, Mehul Patel and Julie Patel.

References

[1] A. Hayes, “The Ins and Outs of Inventory Management,” *Investopedia*, Mar. 28, 2023. https://www.investopedia.com/terms/i/inventory-management.asp

[2] ZippyYum. (2023, September 20). *Subventory*. Google Play. https://play.google.com/store/apps/details?id=com.zippyyum.inventoryapp&hl=en_CA&gl=US&pli=1

[3] R. Chera and N. Freeman, “Using your camera as a pro webcamRicci Chera,” Nikon School, https://nikonschool.co.uk/hints-and-tips/using-your-camera-as-a-pro-webcam (accessed Dec. 5, 2023).

[4] J. Thrailkill, “A practical guide to machine vision lighting,” Advanced Illumination, https://www.advancedillumination.com/a-practical-guide-to-machine-vision-lighting (accessed Dec. 5, 2023).

[5] W. Spurgeon, “Optical parameters for machine vision applications, part 1,” Clearview_Wordmark_RGB, https://www.clearview-imaging.com/en/blog/optical-parameters-for-machine-vision-applications-part-1(accessed Dec. 5, 2023).

[6] “Pricing,” MongoDB, https://www.mongodb.com/pricing (accessed Dec. 5, 2023).

[7] “Getting started guide V4 - Subventory,” Subventory , https://www.subventory.com/GettingStartedGuide.pdf (accessed Dec. 5, 2023).

[8] P. Jothi, “Complete Inventory Management Features: Zoho Inventory,” Zoho Inventory, https://www.zoho.com/ca/inventory/features (accessed Dec. 5, 2023).

[9] A. Rosebrock, “OpenCV OCR and text recognition with Tesseract,” PyImageSearch, https://pyimagesearch.com/2018/09/17/opencv-ocr-and-text-recognition-with-tesseract (accessed Dec. 5, 2023).

[10] “Smart LED Bulb,” Philips, https://www.usa.lighting.philips.com/consumer/p/smart-led-bulb-88w--eq.60w--a19-e26/046677555474/specifications (accessed Dec. 5, 2023).

[11] “Webcam, built-in microphone USB high definition computer camera, 30 fps for online class with Bracket Online Conference Multi-system support: Walmart Canada,” Walmart.ca, https://www.walmart.ca/en/ip/Webcam-Built-In-Microphone-USB-High-Definition-Computer-Camera-30-FPS-For-Online-Class-With-Bracket-Online-Conference-Multi-system-Support/PRD2

20P14MENYMG?fbclid=IwAR1DJDFDsqcyizWYg3-dhSGFzQ72gh9r1CDcqAm17PlV7b1Vpb2IhD7TyGU (accessed Dec. 5, 2023).

[12] "Aluminium alloy specifications ," Aaloc Metals, https://www.aalco.co.uk/datasheets/Aalco-Metals-Ltd_Aluminium-Alloy-Specifications_42.pdf.ashx (accessed Dec. 5, 2023).

[13] "Bolsen Tech digital load cell weight sensor," Amazon, https://www.amazon.ca/Bolsen-Tech-Portable-Electronic-Weighing/dp/B07N5KTQ2L/ref=sr_1_13_sspa?crid=5XRUYBLXABTN&keywords=arduino%2Bload%2Bcell&qid=1701815446&s=industrial&sprefix=arduino%2Bload%2Bcell%2Cindustrial%2C93&sr=1-13-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9tdGY&psc=1 (accessed Dec. 5, 2023).

[14] "Geekstory 4pcs 50kg half-bridge strain gauge load cell," Amazon, https://www.amazon.ca/Geekstory-Half-Bridge-Weighting-Amplifier-Raspberry/dp/B0CF4J1C1G/ref=sr_1_11?crid=TIG4BKMA4EM2&keywords=arduino%2Bload%2Bcell&qid=1701815525&s=industrial&sprefix=arduino%2Bload%2Bcell%2Cindustrial%2C97&sr=1-11 (accessed Dec. 5, 2023).

[15] "114990097," DigiKey, https://www.digikey.ca/en/products/detail/seeed-technology-co-ltd/114990097/5487616 (accessed Dec. 6, 2023).

[16] O. Kallhovd, "HX711_ADC," HX711_ADC - Arduino Reference, https://www.arduino.cc/reference/en/libraries/hx711_adc (accessed Dec. 5, 2023).

[17] "Web App vs Mobile App – Which to Develop First? [2023]," *brainhub.eu*. https://brainhub.eu/library/app-vs-website-which-to-develop-first#:~:text=Mobile%20apps%20are%20better%20when (accessed Dec. 05, 2023).

[18] "Free Decision Matrix Templates," *Smartsheet*. https://www.smartsheet.com/decision-matrix-templates (accessed Dec. 5, 2023).

[19] "Next.js vs React: The Difference and Which Framework to Choose," *ninetailed.io*. https://ninetailed.io/blog/next-js-vs-react/ (accessed Dec. 05, 2023).

[20] "CAN/CSA-C22.2 NO. 61010-1-12 (R2022)," CSA Group, https://www.csagroup.org/store/product/2421901 (accessed Dec. 5, 2023).

[21] “CSA C22.2 NO. 0:20,” CSA Group, https://www.csagroup.org/store/product/CSA%20C22.2%20NO.0:20 (accessed Dec. 5, 2023).

[22] Office of the Privacy Commissioner of Canada, “The Personal Information Protection and Electronic Documents Act (PIPEDA),” Office of the Privacy Commissioner of Canada, https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda (accessed Dec. 5, 2023).

[23] “Code of ethics,” Professional Engineers Ontario, https://www.peo.on.ca/licence-holders/code-ethics (accessed Dec. 5, 2023).

Appendix

[A1] 461 Individual Contributions

Team Member	Project Contribution
Moiz Khan	● Helped scope out project/interview stakeholders with the entire team ● Completed competitor research ● Computer Vision Testing/Training/Analysis ● Computer Vision Research ● Aided in creating and testing mock-shelf and camera setup
Archan Patel	● Helped scope project with direct communication to industry connections ● Acquired subway standard boxes for project ● Project Management with team members ● Built full stack app using Next.js, Mongo.db and Google Cloud API
Andrei Grovu	● Completed competitor research and evaluated current adjacent solutions ● Constructed weight sensor hardware ● Programmed arduino code to calibrate weight sensors and capture weight data based on ● Recorded meeting takeaways
Suyash Unnithan	● Helped scope out project/interview stakeholders with the entire team ● Completed competitor research ● Aided in creating and testing mock-shelf and camera setup ● GPT Vision Model Testing/Training/Analysis ● Built Vision Hardware Server using Flask
Sayohn Ahilan	● Project Management ● Safety and Ethics Compliance ● Created Database Models for Predictive Orders ● Create low fidelity predictive orders functionality

[A2] 462 Planned Individual Contributions

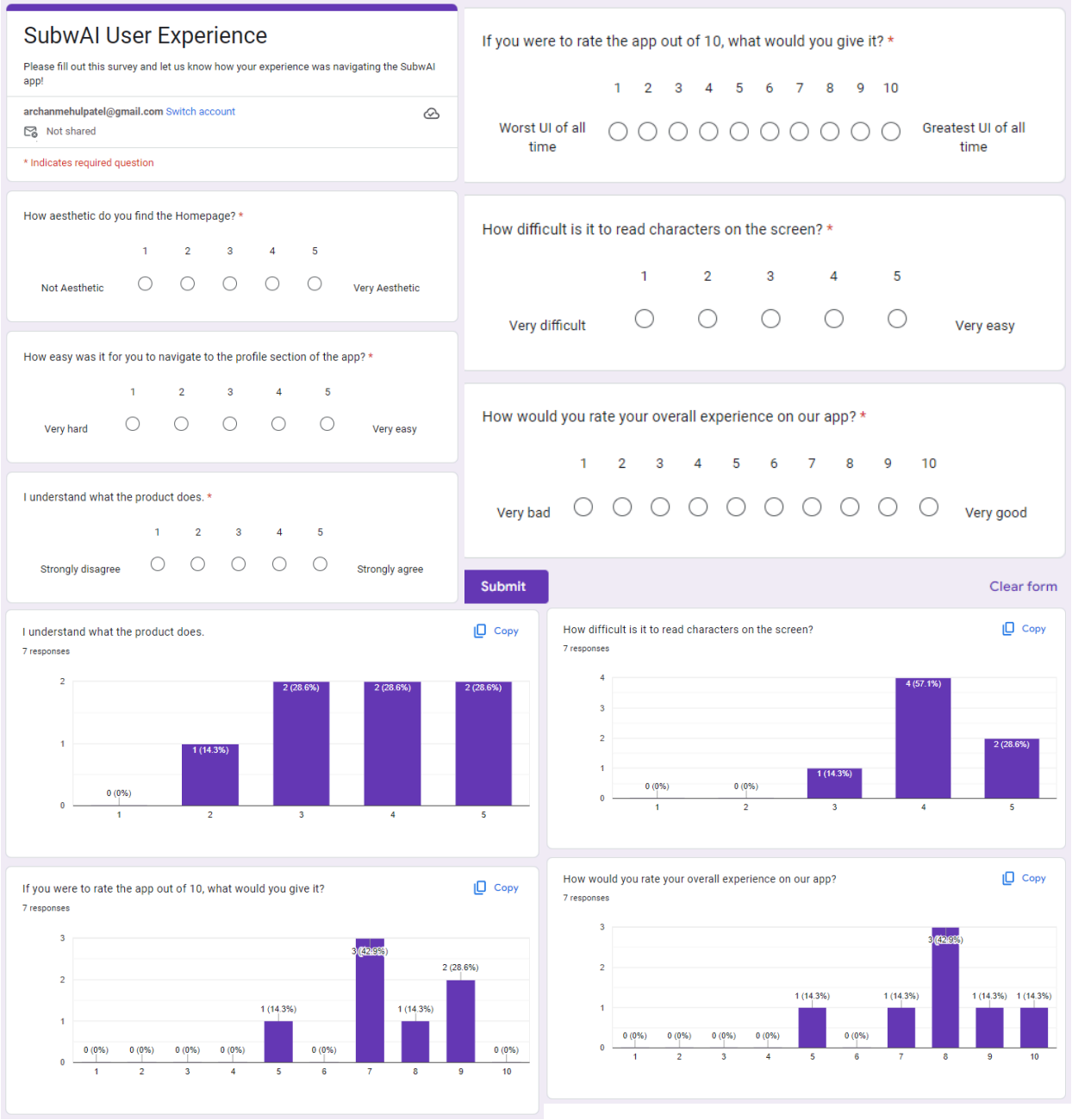
Team Member	Project Contribution
Moiz Khan	● Improve vision object and location detection ● Integrate the weight sensor hardware with vision hardware
Archan Patel	● Make application more user friendly ● Deploy and host the application ● Implement statistics and profile page
Andrei Grovu	● Design a more sleek version of current 3d-printed weight sensor ● Build out 8 more weight sensors ● Implement the weight-data transfer python script into the raspberry pi environment
Sayohn Ahilan	● Conduct item demand-supply analysis ● Build out suggested order functionality ○ Handle one time special inventory requirement ○ Handle perishable items ○ Handle edit suggested orders
Suyash Unnithan	● Improve vision OCR ● Build out API endpoints and requests on web-app and vision server ● Deploy Vision Server ● Help out with software development and inventory engineering backend logic

[A3] Engineering Specification

Camera Resolution Spec:

- Small Feature: “S” letter in “”spicy jalapenos” , 13mm x 15mm in real-life
- Detail Level: 6 x 6 pixels for the smallest feature
- Pixel required: P = Small Feature/Number of pixels covering the feature (P=13/10=1.3 mm per pixel)
- FOV: Shelve size → 1.3m x 0.76m → Rw = W/P and Rh = H/P → Rw=760/1.3=584.615p and Rh = 1300/1.3 = 1000p
- Camera resolution: at least 1000p x 585p

[A4] User Interface Survey



SubwAI User Experience

Please fill out this survey and let us know how your experience was navigating the SubwAI app!

archanmehulpatel@gmail.com Switch account

Not shared

* Indicates required question

If you were to rate the app out of 10, what would you give it? *

1 2 3 4 5 6 7 8 9 10

Worst UI of all time — Greatest UI of all time

How aesthetic do you find the Homepage? *

1 2 3 4 5

Not Aesthetic — Very Aesthetic

How difficult is it to read characters on the screen? *

1 2 3 4 5

Very difficult — Very easy

How easy was it for you to navigate to the profile section of the app? *

1 2 3 4 5

Very hard — Very easy

How would you rate your overall experience on our app? *

1 2 3 4 5 6 7 8 9 10

Very bad — Very good

I understand what the product does. *

1 2 3 4 5

Strongly disagree — Strongly agree

Submit Clear form

I understand what the product does.

7 responses

Copy

0 (0%) 1 (14.3%) 2 (28.6%) 2 (28.6%) 2 (28.6%)

How difficult is it to read characters on the screen?

7 responses

Copy

0 (0%) 0 (0%) 1 (14.3%) 4 (57.1%) 2 (28.6%)

If you were to rate the app out of 10, what would you give it?

7 responses

Copy

0 (0%) 0 (0%) 0 (0%) 0 (0%) 1 (14.3%) 0 (0%) 3 (42.9%) 1 (14.3%) 2 (28.6%) 0 (0%)

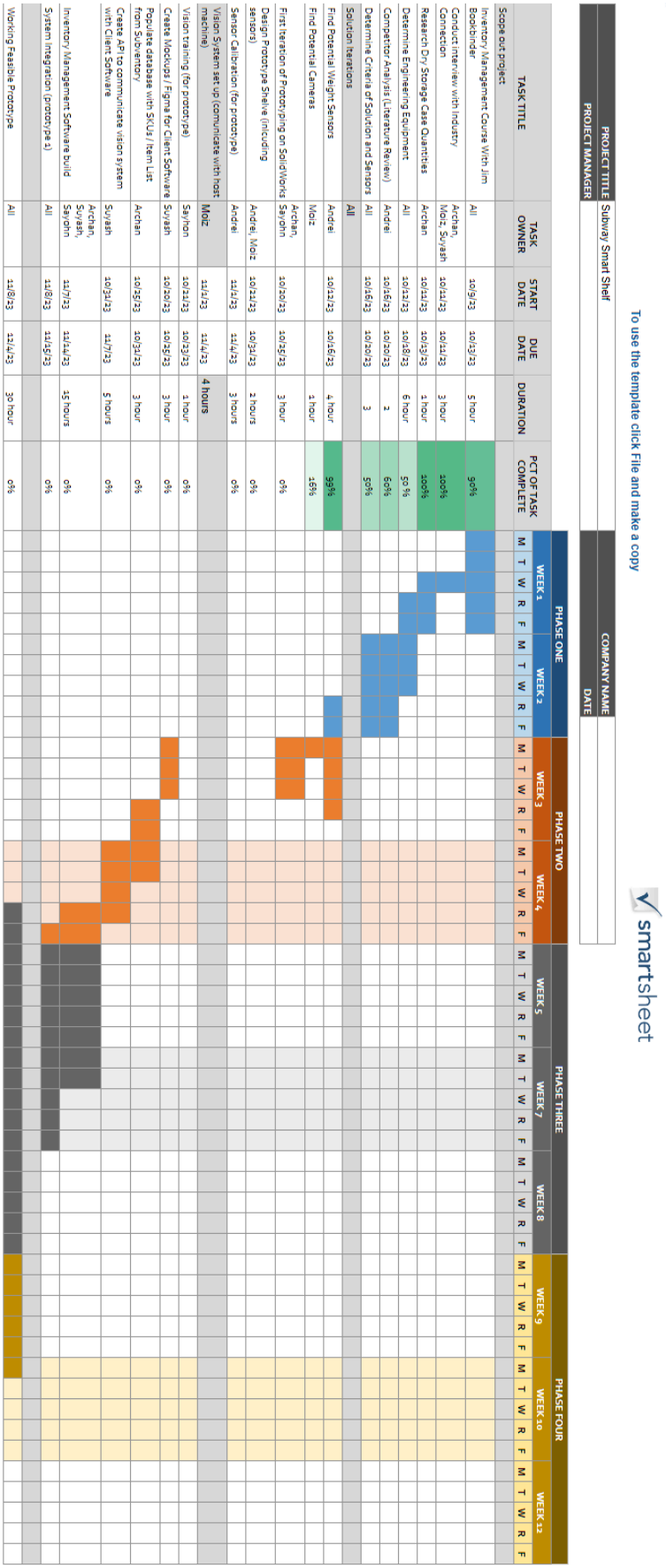
How would you rate your overall experience on our app?

7 responses

Copy

0 (0%) 0 (0%) 0 (0%) 0 (0%) 1 (14.3%) 0 (0%) 1 (14.3%) 3 (42.9%) 1 (14.3%) 1 (14.3%)

[A5] Capstone Gantt Chart

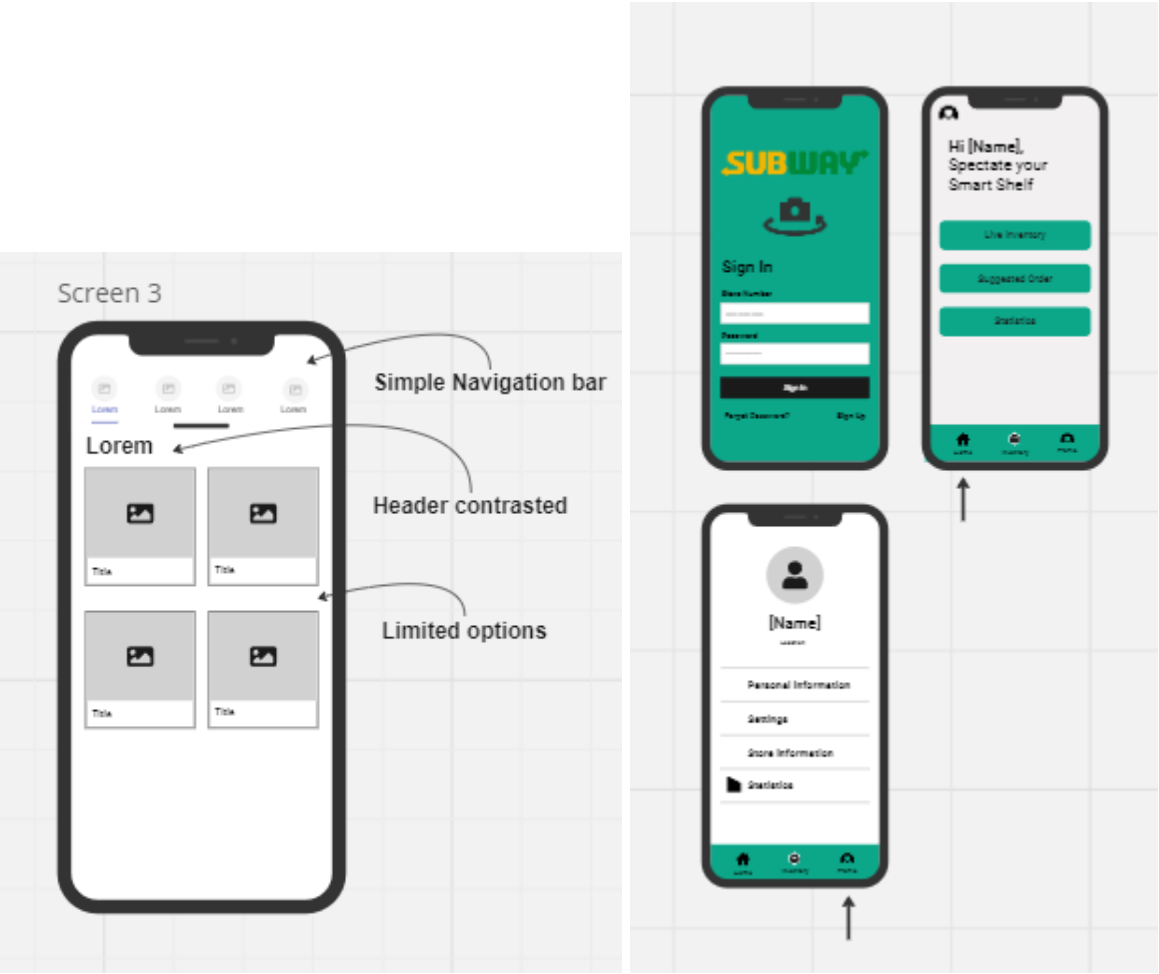


[A6] Web App vs Mobile App Decision Matrix

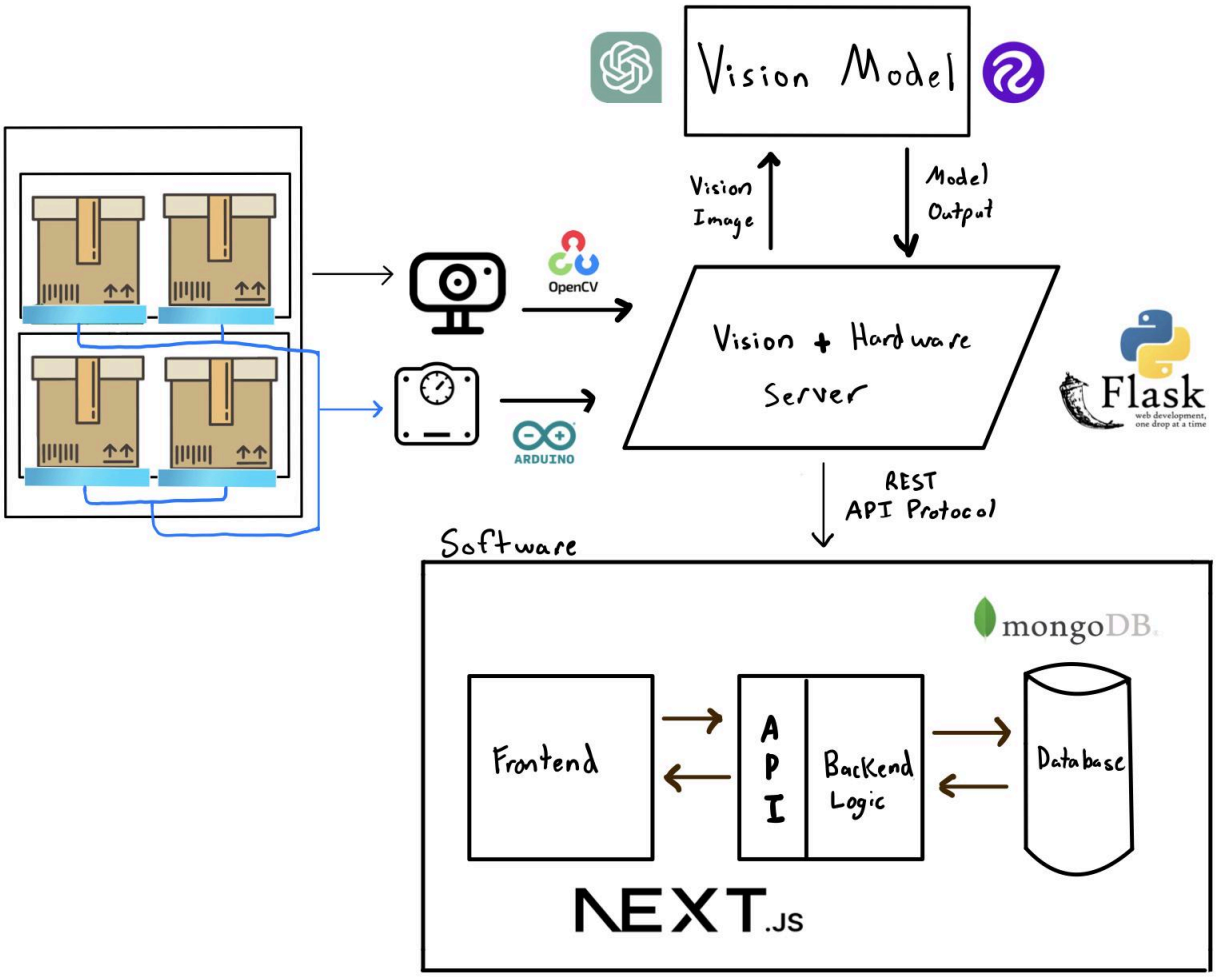
DECISION MATRIX - WEIGHTED

	Criteria 1	Criteria 2	Criteria 3	Criteria 4	Criteria 5	
CRITERIA DESCRIPTION	User Market	Speed of development	Performance speed	Cost	continuous integration and continuous delivery	
	Criteria 1	**Criteria 2**	**Criteria 3**	**Criteria 4**	**Criteria 5**	**WEIGHTED SCORE**
WEIGHT	1	5	4	3	2	15
	7%	33%	27%	20%	13%	100%
OPTIONS	**Criteria 1 SCORES**	**Criteria 2 SCORES**	**Criteria 3 SCORES**	**Criteria 4 SCORES**	**Criteria 5 SCORES**	
Web Application	5	4	3	5	4	4
Mobile Application	3	3	5	2	3	3

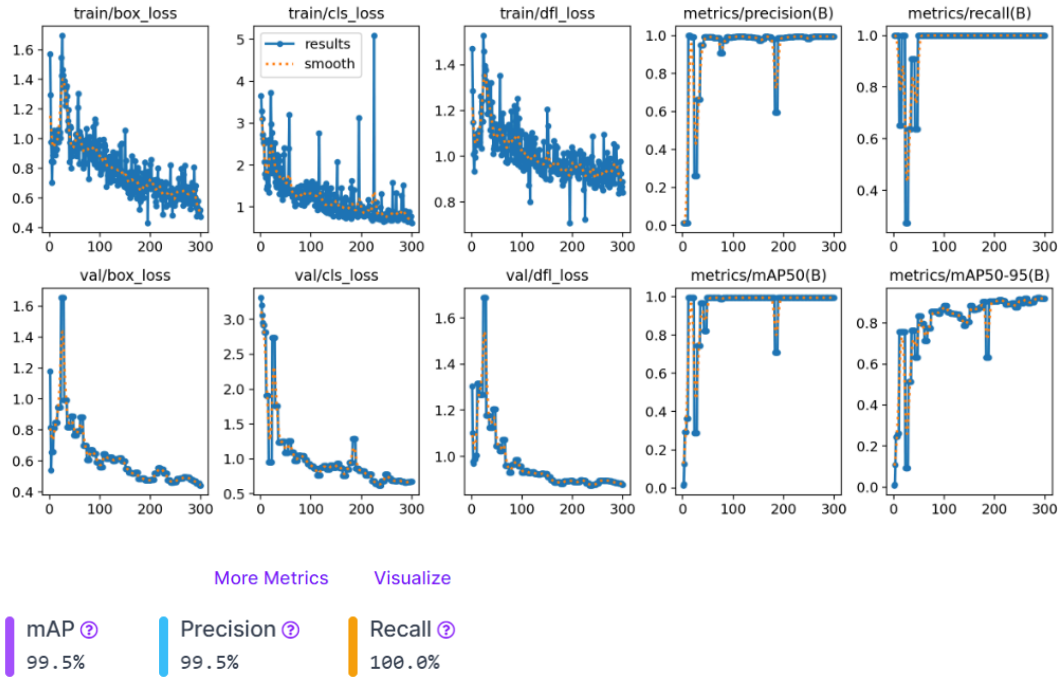
[A7] Wireframes of User Interface



[A8] System Architecture (End to End)



[A9] Roboflow Object Detection Training Statistics: (Similar performance to Tesseract OCR model)



More Metrics Visualize

mAP	Precision	Recall
99.5%	99.5%	100.0%

[A10] Vision Model Example Prompt

```
You will be given an image of an inventory shelf with 4 rows and 2 columns.
    Your task will be to classify the contents of the boxes of the shelf. Some Boxes are white and some are brown.
    You will do this by reading the labels on the boxes and placing them in a 2D array to indicate its position.
    Think of the shelf as a grid system with 4 rows and 2 columns.
    Boxes on the left side of a row are in the first column.
    Boxes on the right side of a row are in the second column.
    The shelf is a white in color.
    Sometimes rows and columns can be empty.
    Typically, positions that are empty on the shelf are denoted with a green 'X' on the backdrop.
    This 'X' is made with green tape and may not look perfect.
    If you see a green 'X' on the right side of a box, that means the box is in the first column.
    If you see a green 'X' on the left side of a box, that means the box is in the second column.
    The following box labels exist on the boxes:
    1. Sliced Black Olives
    2. Sliced Green Olives
    3. Sliced Dill Pickles
    4. Hot Sliced Banana Peppers
    5. Sliced Jalapeno Peppers
    # An example output of your response will be:
    # [['Sliced Black Olives', 'Sliced Green Olives'], ['', ''], ['Sliced Dill Pickles', 'Sliced Jalapeno Peppers'], ['', 'Hot Sliced Banana Peppers']]
    # You only need to return the array.
    This means that Sliced Black Olives exists on the first row of the shelf in the first column and Sliced Green Olives exist in the second column.
    This also means that the second row is completely empty.
    This also means that the third row has Sliced Dill Pickles on the first column and Sliced Jalapeno Peppers in the second column.
```

The final row is empty in the first column and has Hot Sliced Banana Peppers in the second column.

Try your best!

[A11] Layout.jsx (SubwAI)

```jsx
import '@styles/globals.css'
import Nav from '@components/Nav';
import Provider from '@components/Provider';

export const metadata={
  title: "Subwai",
  description: 'Viewing live Subway inventory'
}


export default function RootLayout({ children }) {
 return (
  <html lang='en'>
    <body>
      <Provider>
        <div className='main'>
           <div className='gradient' />
        </div>

        <main className='app'>
          <Nav/>
            {children}
        </main>
        </Provider>
    </body>
  </html>
  )
}
```

[A12] Page.jsx (SubwAI)

```jsx
import Options from "@components/Options";

const Home = () => {
  return (
    <section className='w-full flex-center flex-col'>
      <h1 className='head_text text-center'>
        Hello,
        <br className='max-md:hidden' />
      {/* <span className='green_gradient text-center'> Automated Inventory</span> */}
      </h1>
      <p className='desc text-center'>
      Spectate your Subway Smart Shelf:
    </p>
    <div className="Home" style={{ marginTop: '80px' }}/>

    <Options />
    </section>
  )
}

export default Home;
```

[A13] Computer Vision Pre-Possessing Image Code

```
from PIL import Image, ImageEnhance
import os

def adjust_contrast(image_path, output_path, contrast_value):
    """Adjusts the contrast of an image."""
    with Image.open(image_path) as img:
        enhancer = ImageEnhance.Contrast(img)
        enhanced_img = enhancer.enhance(contrast_value)
        enhanced_img.save(output_path)

def process_images(folder_path, output_folder):
    """Processes all images in a folder."""
    for image_name in os.listdir(folder_path):
        if image_name.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp', '.gif')):
            image_path = os.path.join(folder_path, image_name)
            output_path = os.path.join(output_folder, image_name)
            adjust_contrast(image_path, output_path, 0.75)  # Lowering contrast

input_folder = 'C:\\Users\\moizk\\Documents\\SYDE\\4A\\461\\trainingset'
output_folder = 'C:\\Users\\moizk\\Documents\\SYDE\\4A\\461\\modified_set'
os.makedirs(output_folder, exist_ok=True)
process_images(input_folder, output_folder)
```

[A14] QFD: Engineering Specs

	Functional Requirements (How's) → Customer Requirements - (What's) ↓	Steps to step up (Steps ↓)	Steps to Input Inventory items (steps/item ↓)	Min Temp Working range (Celcuis ↓)	Inventory reading interval (days ↓)	User interface is easy to use (Satisfaction score ↑ [0-5])	Retail Cost (CAD ↓)
1	1.Calculation of inventory			9	9		
2	2. Provide suggested order				9	9	
3	3. Reduce at least 70% of time required to do inventory management	1	9	9	3	3	1
4	4. Visually appealing user component					9	
5	5. Costs less than $3000 to install [4].	9			3		9
	Technical importance score	0	0	0	0	0	
	Importance % (Weighted Score)		28%	16%	23%	15%	13%
	Priorities rank	6	1	3	2	4	5
	Current performance (competition)	50	5	-20	7	2	Free - $10000
	Target	1	0	-10	0.021	5	<3000
	Benchmark	Subinventory	Subinventory	Cognex	Subinventory /Zoho	Subinventory	Subinventory /Cognex