Project NutrientMatch

Authors: Surya Chandramouleeswaran, Grace Liu, Steven Zeng
Affiliation: Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*— NutrientMatch is a system that seeks to track inventory of foods and calorie consumption in peoples' daily lives. The aim of this device is to improve the state of the art pantries and refrigerators which currently do not have inventory capabilities and/or calorie-tracking functions. The product consists of a wooden box with a scale at the center. Additionally, the box also has two cameras for scanning purposes as well as three light sources to illuminate the image for proper image recognition. The system is designed to improve the daily nutritional intake of consumers and provide a stress-free food tracking system that can be easily integrated into larger smart food appliances.**

Index Terms—Raspberry Pi Integration, Image Classification, Optical Character Recognition, SQL Database, Django Web Framework

1 INTRODUCTION

Calorie tracking can be a tedious chore for a wide range of people, from body builders to the average person trying to live a healthier lifestyle in this age where physical wellness is heavily emphasized. The conventional method to track calories is to weigh every ounce of consumed food and to use applications like MyFitnessPal to manually input and compute calories by hand. Likewise, food wastage concerns are significant in a time of limited resources and increasing demand. Constant tracking of food inventory and possession can be cumbersome, particularly how much food one has and also who that food belongs to, especially when living with a group of people. Being a health conscious group who has also experienced the frustrations of family members and friends taking food from a shared fridge or pantry, we wanted to create an inventory tracking product that accumulates daily caloric intake for users.

Our product incorporates a scale and camera into a physical box with the output forwarded to a website to display inventory and caloric intake. The expanded goal of our project beyond the scopes of the Capstone curriculum is to incorporate our product into pantries and refrigerators while implementing macro-nutrient tracking. However, for the scope of this course, we created a product that correctly distinguishes between two food categories: fruits and canned foods. Within the fruit category, our product can classify what type of fruit, obtain the weight in grams, and compute caloric values using the weight and online information. And within the canned food category, our product uses text recognition to obtain the name of the product and scan the nutritional label for the calories per serving and total calories. This data is then stored in a SQL database to keep track of inventory. When a user takes out an item to consume, it is removed from the database and following, the caloric value is accumulated into the user's daily calorie tally.

Our product is particularly valuable for two primary reasons. First, it is a minimalist inventory system that can be incorporated into any food storage apparatus. The only alternative currently on the market is Samsung smart fridges which seem to exhibit an extremely high price point and require users to manually enter the information of their food items. Likewise, people can manually keep a daily inventory, but this has been proven to be a tiring and annoying task. Second, our product allows for users to know roughly how many calories they consume a day from their pantry or fridge. Currently, there is no product that directly accumulates the amount of calories consumed for a given person. At best, most smart appliances have technology to display the caloric intake of various products and accumulate the sum for all users. Our product requires users to log in and track their own intake as well as their own food inventory. This provides a better system to track inventory ownership as well and improve the experience of sharing a smart appliance with others.

Overall, we were able to design an MVP that consists of image recognition and classification as well as database integration between a website, a Raspberry Pi (RPI), and periphery sensors (camera and scale). Our project incorporates the ECE areas of software systems and circuit design. Our group was very excited to learn more about the design process that goes into making a real-world product and the challenges that arise with it. Moreover, we were able to gain a more practical experience in circuit design, ML integration, and secure web development that will extend beyond our time here at Carnegie Mellon University and into industry and research for future use.

2 USE-CASE REQUIREMENTS

With an increase in physical wellness trends, we wanted to create a product that not only helps users keep count of their daily calorie intake in a simple manner but also allows for inventory tracking when a storage appliance is used among multiple people. As previously mentioned, problems arise when food tracking becomes a tedious task, especially for those with relatively busy lifestyles. This was the main factor that influenced the objectives of our product. Recent statistics have shown that 41.9% adults in the United States are obese. In addition to promoting a healthier lifestyle, the shockingly high amount of spoiled food items thrown out per week should also motivate more people to reduce waste during their daily consumption of foods and beverages.

The Use-Case requirements are delineated into a both a physical design experience and the intended effect on users of our product. This section is intended to provide an overview of such processes, and is more focused on the desired experiences from the consumer perspective; subsequent sections are dedicated to exploring design choices in greater detail.

2.1 Structural Overview

Our use-case requirements can be defined both from the design perspective and the user experience we want people to have with NutrientMatch. Structurally, our MVP is a wooden cubic structure with an edge length of 2 feet that holds all of the components of our design together. In the first case scenario when users want to log canned foods, they can hold up their item to a 2 megapixel, 15 frames per second (fps) video camera for image capturing and label reading. An ambient light source of 55 Watts, covered with a diffuse plastering material and dimming capabilities to enhance the user experience is placed around the camera and provides requisite illumination for the object for effective recognition. On the other hand, when users want to log fruits to their food inventory, they will follow the same steps but also place their item on the scale where the weight reading will be forwarded to the Django web application for backend calorie calculations. This reading can be done automatically once the scale senses weight on it. Both of these image recognition cases require users to hold their items around 12-18 inches away from the camera for accurate classification.

2.2 Accessibility and Design Implications

To ensure that our product meets the correct market and can target users who are highly interested in a more elevated food tracking experience, we explored various public safety, safety, and welfare, as well as global, cultural, social, environmental, and economic factors to help us initially identify our use-case requirements.

In regards to public welfare, our main goal is to help users improve their overall health through better calorie tracking habits. As people can become forgetful during busier days while tracking their food inventory, our product also aims to reduce user stress levels by keeping track of every item that enters their fridge. We have ambitious goals of reducing general food waste and improving people's lifestyles through a better food tracking system. With respect to the safety aspect, our product prioritizes the safety of our users by ensuring the base of our structure is built sturdy along with well built components of the camera, light, and scale attached to it through a series of vigorous testing plans. The light also has enough dimming capabilities so users are not blinded by such component.

In consideration of social factors, we created our product under the condition that it can be accommodating for people of different social groups, whether it is regarding dietary preferences or medical concerns, our algorithm performs with minimal bias during its classification process. This accessibility is achieved through a detailed image classification training process and a strong user interface that allows for the selection of various preferences, such as dietary restrictions and a transparent sourcing of nutritional information. We have built such a product to improve the average lifestyle of a user, but in addition to this, little systematic bias is key to our emphasis on being inclusive and creating something for everyone to comfortably utilize.

Lastly, with consideration to economic factors including production, distribution, or consumption of goods and services, concerns arose when we found out the food waste generated daily in the United States during our extensive research process. This does not only pose sustainability challenges economically, but also contributes to a substantial amount of landfill waste that would require extensive efforts to clean up and maintain. While the environment is being harmed, taxpayers are also paying for the damage. Our solution not only concerns the distribution of foods but also aims to help users reduce their waste during their process of food purchasing and consumption as well. By providing accurate nutritional information, from label reading to food classification, along with better control over food inventory, our solution helps users make better use of food items at hand and decrease the number of their food items going to waste.

We were able to build a product that directly addresses calorie tracking concerns but also provide diverse options for users and help make a positive impact on the environment. Careful consideration of various design decisions and their impact on the environment and consumer base we intend to target was key to developing a tailored product. Thus, surveys and incorporating active user feedback is an important aspect of our design process as we seek to optimize our product's performance and accessibility.

3 PRINCIPLE OF OPERATION

3.1 ABET Addition

Regarding engineering principles, we took into account a lot of ethics discussion. The first one is social implications of our design choices. We focused on creating a positive and encouraging community. At first, our product had the potential to cause poor eating habits and negative body image views in users. To combat this issue, we implemented features into the web application to create a supportive community. This can be seen by posting features that have liking and commenting capabilities. Likewise, there was a chat feature utilizing websockets that allowed for user-user interaction. Furthermore, we omitted any negative words and comments when it came to outputting caloric information.

Following ethical considerations, we also analyzed environmental and economic contexts. A major environmental concern that we had involved our product's intended goal of reducing food waste. To better alert users of their inventory, we had multiple website pages to fully display available items. Furthermore, we had to consider how our product could impact the smart appliance market sector. Likewise, there were patents that we researched beforehand to highlight the feasibility of our product and its future applications. This impacted our design choices. Likewise, cost of production really impacted our budget decisions. We wanted the most cost-effective plan which fully utilized our designated budget.

Another engineering principle that we utilized involved the fundamental principles of ML that consists of proper training, testing, and validation. We were cautious in designing an algorithm that had the right split amongst these categories to maximize both performance and applicability. Furthermore, we followed the principles of error-handling to prevent harmful and unintended actions. We were able to use the principle of component implementation followed by full-scale integration, especially on the ML side. The ML components were designed separately and tested locally before being integrated into the website and vice versa with the hardware. This allowed for proper optimization and helped in the process of debugging. This debugging process follows the principle of unit testing and sporadic print statements to find root causes. This helped us debug everything from the R-Pi to the Django web application, and we were able to speed up the verification and correction process.

The last major engineering principle we used was the incorporation of mainstream technologies. This included the usage of open-source libraries like OpenCV and OpenAI. The popularity of these technologies allowed for proper documentation and we were able to seek out forums that helped us when we encountered errors. Likewise, there were multiple instances in which we posted questions that got answered by developers in a very timely manner. An example of this was when we had trouble instantiating the R-Pi on CMU wifi. We posted a question, and a developer helped us configure settings properly in the hardware and low-level software.

Regarding principles of science, the biggest idea we used was trial and error. When it came to testing the camera configuration and the lighting levels, we could only rely on intensive trial and error to figure which setup allowed for the best classification and processing. Likewise, there were many design trade offs we considered. Whenever one potential design choice did not work properly, we had to pivot to other choices that we planned beforehand.

Another popular scientific principle we followed which was on the more technical side was the Law of Reflection. We needed the digital reading on the scale to be caught properly by the camera. We did calibration in terms of camera angle and LED placement to minimize the amount of light reflected by the digital scale. Furthermore, we had to minimize object reflection. For example, there were issues with items like apples and canned objects reflecting light directly into the camera. We had to properly utilize the shadows and saturation to make the IC algorithms work properly. Furthermore, the image quality for text extraction was such a critical consideration that resulted in principles of frame rate and pixel refinement. We had to optimize latency with baseline image quality to help complement the ML models.

Shifting towards the mathematics side, we utilized a lot of linear algebra principles. The first was Principle Component Analysis which reduces an image with a high amount of parameters into a lower dimensional space. This uses eigenvectors and eigenvalues to find the most important features. This allowed us to store data utilizing less memory. Also, this helped us optimize the ML algorithms to only use the important features that were extracted from the camera captures. Another principle we used was singular value decomposition to speed up computations. We utilized a lot of numpy functions, and the latency was greatly reduced by these optimizations. The neural networks we used relied on recursive forward and back propagation which involved derivatives and gradients. In order to optimize the weights, gradient descent methods were used alongside error functions. This was used alongside difference functions like l2 norms.

Other principles of math that were used related to geometry and trigonometry. As mentioned previously, we used the law of reflection which utilized basic trigonometry to determine the optimal angles. Furthermore, shape recognition factored significantly into the IC algorithms. As a result, we needed to utilize algorithms that frame objects in a particular shape. Likewise, the box construction relied on structural integrity principles. This involved truss analysis which used trigonometric principles and the law of gravity to compute limits in terms of allowable weights. This allowed us to create more specific requirements that we mentioned in the previous sections.

The last principle of math that we used was integer and float manipulation. For the calories we used multiplication and sum functions to accurately compute caloric values. Likewise, we did type verification to ensure proper type conversion. This helped combat issues like integer overflow and negative numbers.

After carefully considering and understanding all these principles, we were able to construct our product with proper design choices. The next section will provide a more specific description of our product and the web application.

3.2 System Operation

Fig. 13 on the final pages of this design plan provides a block diagram of our product and its components. In summary, there are two peripheral sensors (the camera and the scale) that forward outputs to an R-Pi to synchronize with the Cloud and a SQL database. The web application interacts with the Cloud and the database to facilitate user interaction. We encourage readers of this project report to consult the back pages for a detailed view of the block diagram description of the project.

Fig. 1 below represents the physical product that have built to engage with the web application user interface and Cloud server.

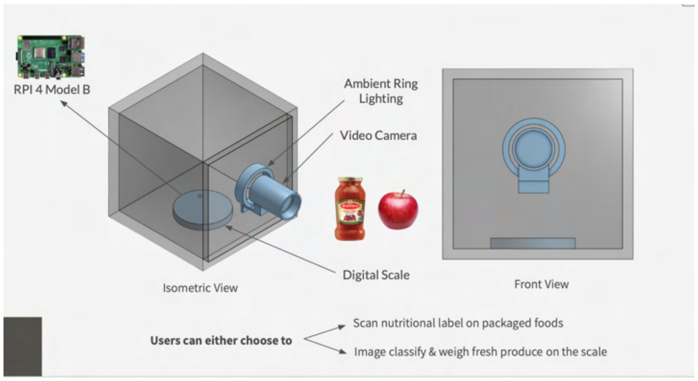
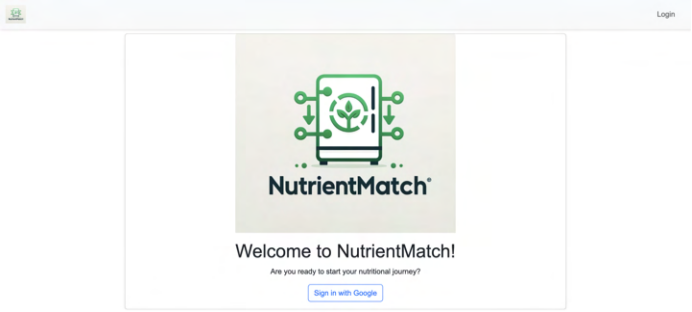


Figure 1: CAD Diagram

The R-Pi connects to the scale and camera to populate the database. After obtaining the scale reading and images of the item (front and back), the R-Pi stores these values in the database followed by computations of caloric value.

User authentication and management is handled through Google OAuth service that can be reflected on the user login page. To be able to login, users will need to first register using a registration page that also gives the option of logging in through OAuth as shown by Fig. 2:

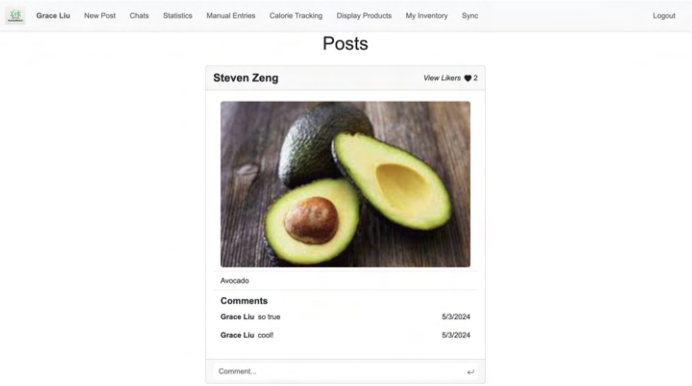


Welcome to NutrientMatch!

Are you ready to start your nutritional journey?

Figure 2: Registration Page

After users have successfully logged into the application using Google OAuth, they will be navigated to our Fig. 3, where they can see posts from other users using the same product and vice versa. As seen below, this page was scripted with asynchronous JavaScript and XML to ensure quick refreshes every 5 seconds.



Posts

Steven Zeng

Avocado

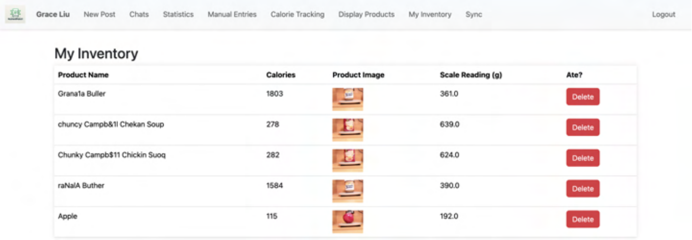
Comments

Grace Liu so true 5/3/2024

Grace Liu cool! 5/3/2024

Figure 3: Global Page

As shown by Fig. 5 under "My Inventory" on the navigation bar, users will be able to see an inventory list for their fridge as shown by Fig. 5 upon scanning and weighing their food items:



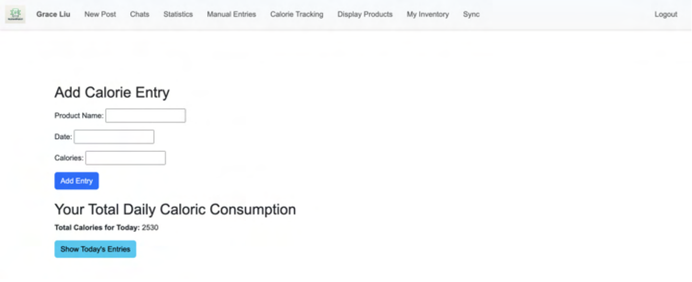
My Inventory

Product Name	Calories	Product Image	Scale Reading (g)	Ate?
Grana1a Buller	1803		361.0	Delete
chuncy Campb&1l Chekan Soup	278		639.0	Delete
Chunky Campb$11 Chickin Suoq	282		624.0	Delete
raNalA Buther	1584		390.0	Delete
Apple	115		192.0	Delete

Figure 4: Inventory Page

This page operates purely on database logic implemented through Django's MVC architecture. The database is accessed through the `models` framework, and `views.py` handles the processing and rendering of data to our front end to be displayed to users.

Lastly, users also have the option to manually enter food consumption with their respective caloric values as shown by Fig. 5. This is added to their daily total along with the entries that they delete from their person inventory.



Add Calorie Entry

Product Name:

Date:

Calories:

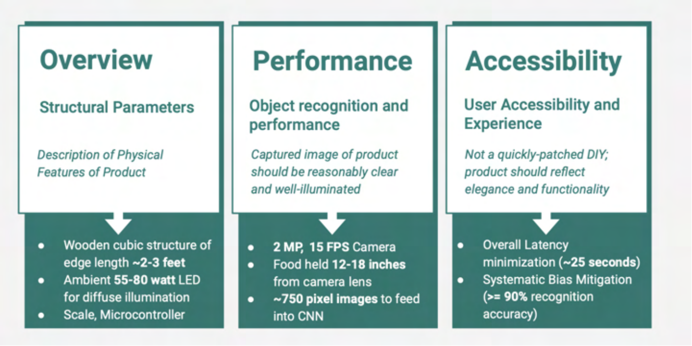
Your Total Daily Caloric Consumption

Total Calories for Today: 2530

Figure 5: Manual Entry Page

4 DESIGN REQUIREMENTS

Fig. 6 below summarizes and divides our overarching design requirements into three components: physical, computational performance, and user experience.



Overview

Structural Parameters

Description of Physical Features of Product

- Wooden cubic structure of edge length **~2-3 feet**
- Ambient **55-80 watt** LED for diffuse illumination
- Scale, Microcontroller

Performance

Object recognition and performance

Captured image of product should be reasonably clear and well-illuminated

- **2 MP, 15 FPS** Camera
- Food held **12-18 inches** from camera lens
- **~750 pixel images** to feed into CNN

Accessibility

User Accessibility and Experience

Not a quickly-patched DIY; product should reflect elegance and functionality

- Overall Latency minimization (**~25 seconds**)
- Systematic Bias Mitigation (**>= 90%** recognition accuracy)

Figure 6: Summarized Design Requirements

Design requirements, from both the hardware and software perspectives, are enumerated in this section and centered around the three implementation themes in this project below.

4.1 Physical Design Requirements

- **Size Constraints:** The scale and camera setup must fit within specified dimensions not exceeding 12 inches x 12 inches x 6 inches to ensure compatibility with food storage systems. This is centered around the dimensions of the Oxo scale that is placed within the apparatus.

- **Material Selection:** Since walnut wood has a consistent and monochromatic finish, it was our best option for the background to facilitate optimal image recognition.

- **Mounting Mechanism:** The hinge mechanism supports a weight of up to 2 kg and allows for adjustments with an angular precision of 5 degrees. This constitutes the class of food items we sought to classify in the project.

4.2 Functional Performance Considerations

- **Camera Specs:** The camera must capture images at a minimum frame rate of 30 fps with a resolution of at least 5 megapixels to support accurate image processing.

- **Database Interaction:** Database read and write operations must be completed within 25-30 seconds to ensure real-time data acquisition and user interaction. *This constitutes a full-cycle latency goal for each update operation in usage of our product.*

- **Calorie Tracking Accuracy:** The accumulated calorie amount should be within a 10% deviation of the actual consumption amount to meet the project's goal of promoting healthy eating habits.

- **Inventory Tracking:** The system must achieve 100% accuracy in inventory tracking, with a maximum allowable error rate of 1% per item.

- **Automatic Data Forwarding:** Upon detection of nonzero scale readings, the scale and camera inputs should be automatically forwarded to the database within 2 seconds.

4.3 User Experience Requirements

- **Website Responsiveness:** The web application must load within 3 seconds and maintain a response time of under 500 milliseconds for user interactions. This statistic is derived from social psychology experiments determining ideal latency times for users to not notice "lag" in application usage [5].

- **Cloud Deployment Statistics:** The web application must be deployed on a cloud infrastructure with a guaranteed uptime of 99.9% and supports concurrent user access of up to 10 users. This is a rather small number in comparison to the applications that modern databases are developed around but in our opinion strongly suffice for the project.

- **Security:** We have determined that password authentication must use bcrypt hashing with a minimum work factor of 12 and Google OAuth2 tokens must have an expiration time of 1 hour. These will be implemented through Django MVC and settings files.

- **Accessibility:** Users should be able to input corrections within 10 seconds of error detection and a confirmation prompt for data validation.

The statistics mentioned above are not arbitrary; they were developed as a function of market research and consultation with faculty and peers and are centered around a balance of optimal and achievable performance benchmarks for to help meet our design requirements.

5 DESIGN TRADE STUDIES

In this section, we will discuss the various design choices and evaluations made regarding the integration of various hardware and software components.

5.1 Camera Integration

Throughout the design process, we learned the importance of proper camera integration and lighting as the backbone of effective and accurate ML classification. The design pipeline as proposed relies on clear images to be passed into the recognition models which are hosted on the cloud. Several components, from resolution, frame rates, and lighting, were optimized only through active experimentation throughout the design process.

Experimenting with the MacBook camera (60 fps and 7MP), the resulting image worked well with the recognition model. However, the resulting image was extremely high in resolution which would eventually lead to ineffective memory usage. As a result, we began pivoting towards a simple Arducam IMX708 camera (12 MP). While this option worked well to test elementary performance on classification algorithms and offered seamless integration with our Raspberry Pi, the limited frame quality limited the maximum classification accuracy obtained on runs. We decided to purchase a RealTek 1080p USB camera; the image quality and integration with Raspberry Pi frameworks were seamless, and we vectorized our classification algorithms to compensate for latency concerns.

To reiterate, an important takeaway from our experiences with the testing process was the importance of proper lighting and its impact on object classification and scale reading. Our original design choice involved LED strip lights with its primary advantage being low wattage with functional lighting; however, the image quality was unexceptional and thus negatively impacted the label reading process. The seven-segment display of the scale is highly sensitive to glare, requiring a more diffuse, matte lighting structure with controlled wattage. With ring lights, the object became more clear with proper illumination. We experimented with angles, shadows, and object placement to find a functional routine for object placement and classification, and this process was optimized up until our final project demonstration.

An important drawback we considered in using a ring light is the need for more power, which is around 50 watts. However, for the scope of this project, we prioritized performance over power efficiency. In the long-run, we hope to maximize the energy usage of our product to achieve our overall goal of addressing environmental concerns.

5.2 Scale Selection and Communication

From measurement to communication with the cloud database, low-latency communication of scale readings is essential to user caloric tracking. Such a process was tightly-integrated with the classification process from the database perspective so that item-caloric content pairs are effectively established.

While evaluating options to achieve this, the most obvious approach to us was wiring a microcontroller chip into the USB/COM port of the scale hardware and writing scripts to sample and forward readings from such a device into the computer. However, an important consideration was the potential for irreparable scale damage through this approach if the soldering damaged nearby circuitry. This section aims to reiterate our options for scale integration (garnered through market research) and a justification for selecting the minimally-invasive optical character recognition option:

- **Aideepen ESP32-CAM W BT Board ESP32-CAM-MB:** The Aideepen ESP32-CAM board offered the convenience of both a camera module and Bluetooth functionality. Learning ESP32 protocol was a requirement for this project, and ultimately the limited ability to customize the board functionality with existing algorithms relegated this option in our preferences for implementation.

- **ESP-32 Thing Plus (SparkFun):** From our knowledge in previous electronics course experiences, SparkFun is a well-known and reputable manufacturer. This is considered an elementary option for integration: with a rich array of interfaces and expansion headers supporting I2C, SPI, and UART (these are all just jargon for interface protocols), the SparkFun option is most the flexible for incorporating various peripherals and sensors. However, since this option is general, we have found that the Aideepen option is much more tailored for our recognition regarding communication and update features that we seek to implement. We have included this product purely in consideration for its proof-of-concept potential and highly general application.

- **Amazon Essentials Kitchen Scale with Raspberry Pi (RPI) Model 4B:** Originally a stretch goal, the benefits of this approach are its wireless and functional integration with RPI technology, offering a simple solution for data transmission through optical character recognition (OCR). In particular, seven segment OCR (SSOCR) is well-documented for being optimized for the hex displays of simple kitchen scales. The most significant drawback, of course, is the somewhat limited accuracy of SSOCR and its sensitivity to lighting conditions. With the opportunity to optimize this algorithm over several weeks, we were eventually able to get an SSOCR model that recognizes measured scale weights within 10 percent of the actual weight, along with the added benefits of a wireless, aesthetic, and low-latency implementation that seamlessly works with our designed web application.

5.3 Database Management

Database operation is a critical component of this project as it constitutes the backbone of information recording and operation in the context of caloric tracking. During the evaluation of current database options to satisfy our MVP, we narrowed down these selections based on four factors: compatibility with existing hardware, scalability, performance, and familiarity. Among such valid options include MySQL, PostgreSQL, and MongoDB.

The following section evaluates the relative merits of each option and our rationale for selecting MySQL in the end as our database:

- **Reliability and Stability:** MySQL has a long-standing reputation for its reliability and stability in handling large datasets and concurrent transactions. Its robust architecture ensures consistent performance, making it generally more suitable for our project's needs.

- **Resources and Support:** MySQL is an extensively used relational database management system (RDBMS) with strong sources of support, both online and through CMU faculty and coursework from a Web Applications course (17-437) we have all previously taken. This ensures access to comprehensive documentation, support forums, and readily available resources for troubleshooting and optimization.
- **Scalability:** MySQL offers excellent scalability options, allowing us to accommodate for future growth and increased data volume without having to make significant architectural changes. Features like replication, sharding, and clustering demonstrate how MySQL can efficiently handle a growing user base and data-intensive operations.
- **Compatibility:** MySQL is compatible with various programming languages, frameworks, and platforms, which would allow for seamless integration with our software systems. This compatibility extends to our chosen technologies for web development and RPI programming, ensuring smooth database integration across the project. Note that mySQL also demonstrates compatibility with ESP8266 and Raspberry Pi microcontrollers as well.
- **Performance:** MySQL is renowned for its performance optimization capabilities, including indexing, query caching, and storage engine options. By leveraging these features, we can achieve optimal performance for data retrieval, storage, and manipulation, which is important to achieve our real-time image recognition and classification tasks.

PostgreSQL and MongoDB were also well-established options for database management in our project, but there were some considerations that limited them from being viable competitors to MySQL in the end:

- **PostgreSQL:** Although PostgreSQL offers advanced features such as support for JSON data types and robust transactional capabilities [1], it is more suitable for complex data structures and applications requiring advanced querying capabilities. For our MVP, which primarily focuses on image recognition, classification, and basic data storage, the additional complexity of PostgreSQL may not be warranted. Furthermore, although PostgreSQL is built into deployment options such as Heroku through Nginx for static file management, our familiarity for deployment through Amazon Elastic Cloud (EC2) better-serves us to pursue integration with mySQL as opposed to PostgreSQL.
- **MongoDB:** MongoDB is a document-oriented NoSQL database known for its flexibility and scalability, particularly for unstructured or semi-structured data [6]. While MongoDB could potentially offer benefits in terms of flexibility and schema-less design, it may also introduce complexity in terms of maintaining data consistency and integrity, especially when dealing with relational data models or transactional requirements. Given the structured nature of our data and the relational aspects of our project, MySQL's strong support for Atomicity, Consistency, Isolation, Durability (ACID) transactions and relational integrity constraints align better with our needs.

5.4 ML Design Choices

There are 3 options we have analyzed for the ML design portion of this project. The first was the soft-margin SVM formulation with the following optimization problem:

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{1}$$

$$\text{subject to} \quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall i = 1,\ldots,n \tag{2}$$

$$\xi_i \geq 0, \quad \forall i = 1,\ldots,n \tag{3}$$

The goal was to utilize this formula to create a decision boundary between canned foods and fruits. The slack variable, ξ_i, allows for data points that are either misclassified or between the minimum margin and decision boundary. The primary reason we planned to use this for the binary classification was because the SVM method will not be feasible for the other ML design steps. The primary benefit of using soft-margin SVM was due to that fact that it did not rely on all the training data but rather on support vectors which are closer to the decision boundary (on the margin or outliers). Likewise, soft-margin SVM was more resistant to outliers due to the introduction of slack variables, and canned food and fruits are such broad categories, thus resulting in the use of anticipating a moderately large number of outliers in our training set. Additionally, using kernels allowed for non-linear classification, which was why we decided to choose SVM over logistic regression and Naive Bayes classification. The goal was to optimize the weight variables and intercept variable to correctly classify an image based on its position with regards to the decision boundary created by the solution to the optimization problem. Likewise, we planned to use knowledge from the course ML for Engineers (18-661) and assistance from professors to effectively utilize the benefits of this approach. The reason we decided not to integrate this method into our final product was for accuracy reasons. Online libraries have significantly more training data leading to more representative decision boundaries that our SVM model could not replicate. Likewise, we tried using radial kernels to model non-linear representations, and the accuracy never satisfied our previous requirements. Despite all this, it was a great learning experience in terms of applying previous knowledge, and the experience helped us understand other ML models better in terms of complexity.

The second ML algorithm we planned to use was either GoogLeNet or ResNet-18. The primary purpose of using either one of these was to differentiate fruit (mainly between bananas, oranges, and apples). This tested our abilities in understanding neural networks and fine-tuning them to classify specific items. Using a library like this has benefits in accuracy and efficiency due to its extensive documentation and testing by professionals in the field. But the primary drawback of this approach involved complications with library compatibility and difficulty in altering pre-existing code. Both these models had the same concerns, and it ultimately came down to performance and latency tests that pushed us to use ResNet-18. As a result, we used ResNet-18 primarily for image classification, and its scope in our project was to do fruit classification and to allow differentiation between whole foods and foods with nutrition labels. There were some preliminary accuracy concerns, but proper tuning allowed us to address these. Overall, it was a good experience testing out two commonly used libraries, and ResNet-18 was powerful enough to fulfill our design and use-case requirements.

The last algorithm we experimented with was ChatGPT4 API which was by far the most powerful tool to implement. It required familiarity with using the API in which we gained a lot of experience with. We expected this method to yield the highest accuracy results due to the vast training data and network that ChatGPT4 utilizes. The primary drawbacks were speed and usability. We planned to use this primarily for label reading by inputting an image into the API to output values from labels. However, this API was really difficult to work with in regards to accepting images. Likewise, the execution speed fluctuated tremendously due to network traffic. Lastly, this method involved a lot high-level knowledge with limited understanding in how the neural network worked. This made debugging very difficult and time-consuming, and this led to a stressful learning process.

We experimented with all three of these approaches with the expectation of using all of them throughout various components of our classification and text extraction pipeline. Each of these choices had their own benefits and drawbacks, and we ultimately decided upon using ResNet-18 with Tesseract and Seven Segment OCR.

An honorable mention that we explored was an online library to extract valuable information off nutrition tables created by Open Food Facts [9]. The issue with this approach boiled down to its low accuracy and buggy design. This library correctly read the correct protein values but was very ineffective in extracting the calorie amount. Likewise, the training data was quite limited and unrepresentative. The primary benefit of trying this design was being able to obtain valuable knowledge using various libraries like OpenCV and Tesseract. Likewise, Steven got practice fine-tuning parameters to improve results slightly. Despite reaching a dead end in this approach, we obtained invaluable information that will go a long way in the design process.

6 SYSTEM IMPLEMENTATION

The architecture for this project is centered around a modular technology approach. We focused first on software development, including web development and classification, and then moved onto scale hardware integration with our cloud-based database.

6.1 ML Algorithms

There were a sequence of steps that each consisted of varying ML algorithms to correctly classify and store values for an image in the database.

The first step was IC which categorizes an image into two groups: fruits or canned foods. To do this task, the product was intended to use either ResNet-18 or soft-margin SVMs [3] to properly classify images into these two groups. To find the correct decision boundary, the ML algorithm took into account various features to best formulate the respective boundary. An example of what this consisted of can be seen below in Fig. 7.

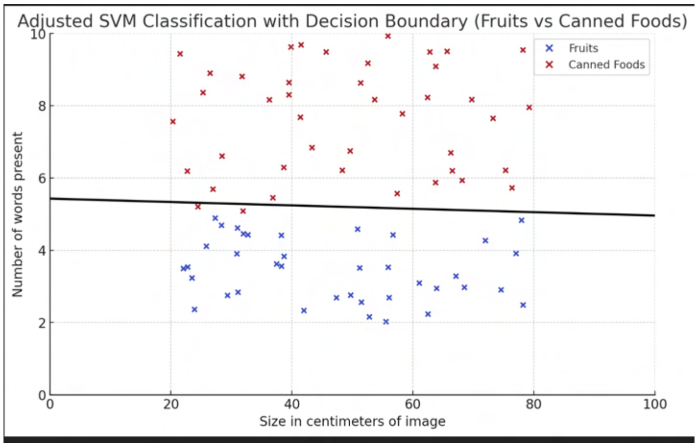


Figure 7: Classification Decision Boundary

After classifying these images into two groups, each group has its own stage of required computations. For the fruit group, the image was then offloaded to a program that utilized GoogLeNet, a twenty-two layer CNN, to classify the fruits into either apple, banana, or orange. The twenty second layer was fine-tuned using OpenCV and color recognition to better differentiate between apples (red), bananas (yellow), and oranges (orange). After completing these computations, the classified result was forwarded to the database with a corresponding weight and caloric value which will be obtained using ChatGPT's API call.

The second group, which is canned foods, was redirected into a label reading algorithm. The algorithm to conduct this functionality was intended to be ChatGPT's Image Reading API which would take in an image and output a formatted response that can easily be forwarded to the database. However, we decided to use Tesseract OCR to do the extraction process instead. The response then consisted of the number of servings, serving size, and caloric information alongside the name of the canned food product.

6.2 Scale Integration

Our project required seamless integration of scale readings with the MySQL database to enable efficient inventory tracking and caloric consumption accumulation. Leveraging the Raspberry Pi Model 4B in conjunction with a simple Bluetooth scale such as the *Amazon Essentials Simple Kitchen Scale* [2], we developed a robust solution for capturing weight measurements and transmitting them to the database:

- **Hardware Setup**: The RPI board is controlled and booted through a headless SSH setup, and a simple recognition algorithm is implemented on this board to focus the cameras and prepare for the capture. The camera ports are configured with the `V4L2` camera driver, and the autofocus implementation runs to focus the capture on the scale reading frame alone.
- **Software Implementation**: Upon receiving weight measurements from the scale, the captured image is sent to the SQL database through a series of SQL operations implemented on the RPI. The table, if not already generated, is built from scratch and the new data entry associated with the logged-in user and scanned food item is feteched and updated with the parsed weight. The `mySQLconnector` package, in concert with the `cv2` library, are key to our accurate and low-latency transfer of scale readings to the database with the help of the RPI.
- **Polling and Data Transmission**: To facilitate efficient inventory tracking, the RPI code implements a polling mechanism that periodically samples weight readings at predefined intervals. This aligns with faculty discussion which discussed the desire for a real-time busy-polling feature for weight sampling. The sampled data is then packaged into structured JSON or CSV format and transmitted to the MySQL database via HTTP POST requests for seamless integration with the backend system.
- **Error Handling**: Addressing failure cases is essential in any human system which is prone to failure. Error handling mechanisms within the firmware to mitigate data loss or corruption due to network instability or hardware malfunctions are implemented thorugh a backup procedure that runs to save the latest state to the on-board SD card of 128 GB capacity. Furthermore, error logging and retry mechanisms were implemented to ensure that failed data transmissions are logged and reattempted to maintain data integrity and reliability. And finally, a ”sync” option is present in the event the scale reading is unable to be performed properly; given that this happens roughly 1/4 times due to the difficulty of accurate SSOCR, the ability to quickly scan and resync is important for both debugging purposes as well as usability on the part of our audience.

6.3 Web Application

We envisioned a strong web application to play a pivotal role in the accessibility of our product. Our vision for such an application was to provide intuitive access to inventory tracking, caloric consumption accumulation, and data visualization functionalities. Built on the Django MVC framework and utilizing MySQL for the database backend, our web application successfully delivered a seamless and responsive user experience:

- **User Authentication and Authorization**: Our web application implements robust user authentication and authorization mechanisms, leveraging Django's built-in authentication system to authenticate users and manage access control. We require users to register accounts and log in securely to access personalized features and data.
- **Real-time Updates and Notifications**: To enhance user engagement and provide timely feedback, our web application integrates real-time updates and notifications functionality. Python WebSockets, coupled with Django Channels, enables bidirectional communication between the server and clients, facilitating instant notifications for inventory changes, weight measurements, and system updates. This offered an opportunity to build on similar concepts covered in our web applications course (17-437).
- **Front-end Compatibility**: Responsive layout principles ensured optimal viewing and usability across various home devices, including laptops, smartphones, and tablets. With the help of React and Bootstrap CSS frameworks, consistent and visually appealing user interfaces across different platforms was achieved.

7 Test, Verification, and Validation

Our exhaustive testing plan was centered around a modular unit-testing structure for each of the sub-components of the project, followed by comprehensive final product testing, as reflected in our planned schedule on the last page. This section elaborates on the specifics of our testing structure.

7.1 Tests for Scale and DB Design

With regards to validating the scale readings, we tested two metrics: latency and accuracy. Latency was measured with an RPI Latency Measurement Kit that we obtained online. The expected time for the database to fully update and retrieve information to be printed on the RPI shell was set to 20 seconds. Our goal was to reach that time and reduce latency as much as possible. The test included 100 measurements of various weights. Then, the accuracy of the scale reading that is displayed on the terminal shell from the SQL database was analyzed for correctness with the goal of 90% accuracy.

The latency resulted in a time of **14.6 seconds** which exceeded our original expectations. Likewise, the accuracy of the scale reading was **93**%. This tested the Seven Segment OCR algorithm that we coded ourselves. Surya used knwoledge from his computer vision course to implement this algorithm.

7.2 Tests for Image Classification

Regarding the ML portion of the design, there were multiple components to execute and test. The first component was the differentiation between fruits and canned food using SVMs. The desired result was 90% which was obtained through careful analysis of similar ML projects [11] and the limitations of SVMs. After training the soft-margin SVM model, Fig. 8 below demonstrates the optimal C value which was used in determining how much slack our model gives to outliers or points that fall between the decision boundary and the support vectors [3]. These points could have been misclassified data values or points that are not as confidently classified. As seen by Fig. 8, the optimal C value appears to be 0.005 with around 86% accuracy. We were able to fine-tune the algorithm by adjusting the various weights given to the two features as well as experiment with different features like weight and light exposure. However, we were only able to obtain an accuracy of **68%**.

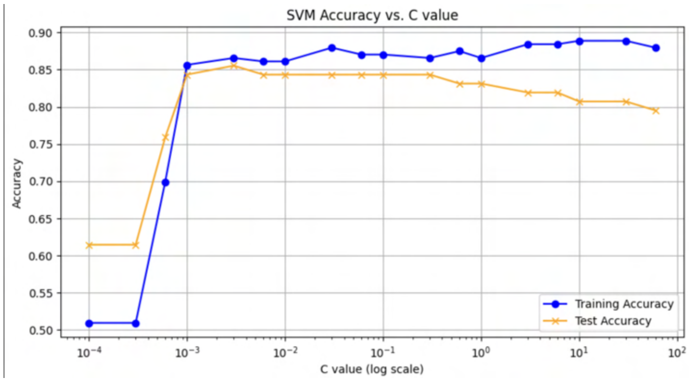


Figure 8: Optimization of C Parameter

These results were not optimal for our model requirements, so we decided to test pre-trained models, Fig. 9 demonstrates the accuracy and latency results of three popular pre-existing models. Note, the accuracy was consistently better than our SVM implementation. We decided to use ResNet-18 due to its optimization of accuracy and latency.

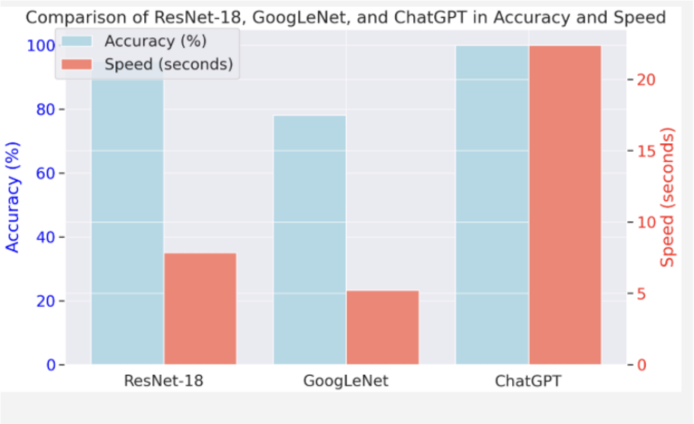


Figure 9: Comparison Amongst Different IC Modles

The next series of tests after determining the right model involved testing the ResNet-18 model's classification accuracy. Fig. 10 shows the resulting confusion matrix for a sample of 400 fruit and canned food images obtained from RPI captures. As seen by Fig. 10, the accuracy turned out to be 90%. For the purpose of testing, we only used four categories, but the classification worked with other objects as well.

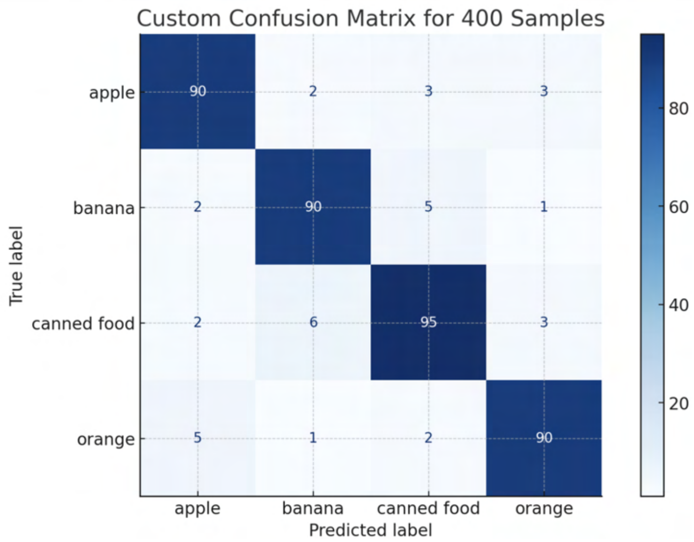
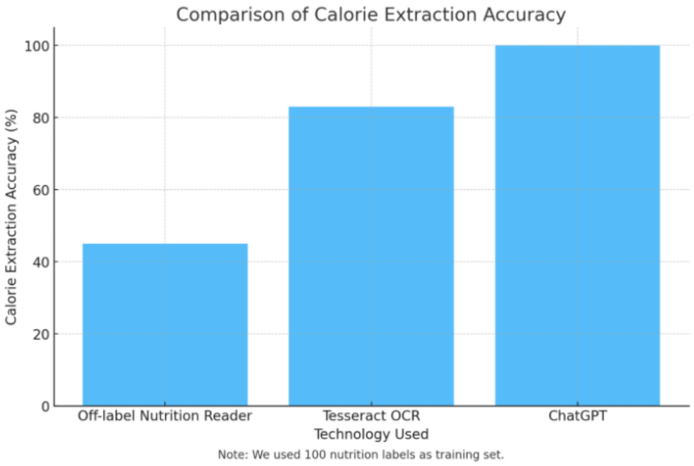


Figure 10: Confusion Matrix

7.3 Tests for OCR Text Extraction

Another series of ML tests involved label reading accuracy tests. OpenAI has a reputation of having around 97% accuracy regarding text-reading capabilities. As a result, the desired result we sought was 95% accuracy when it comes down to reading the nutritional label of canned foods [8]. The test consisted of 100 images of various canned food images online. The accuracy was determined solely by the amount of calories per serving. These were conducted manually to better monitor the results and identify potential issues. Fig. 11 below demonstrates the results. Due to usability concerns with ChatGPT, we decided to go with Tesseract OCR as we deemed its accuracy reasonable.


Comparison of Calorie Extraction Accuracy

Note: We used 100 nutrition labels as training set.

Figure 11: Calorie Extraction Accuracy

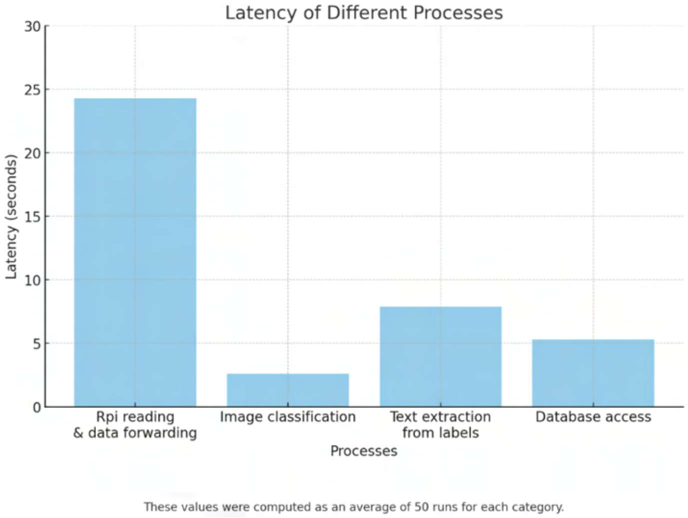
The second to last series of ML tests involved proper product name extraction from the front label. This was a test that allowed for error in terms of word-for-word accuracy. As long as the name was something roughly similar to the actual product name, then we considered it accurate. With this lose definition of accuracy, we were able to see an accuracy of 87% with a sample of 100 canned food images (online and RPI captures).

Lastly, the database was be consistently monitored to see if our algorithm properly stored names and values from the various ML computations as described in the previous tests.

7.4 Website Test and Validation

The website was tested in two separate phases: local and cloud. The local phase was ran locally on a computer, and the primary concerns with it was usability and functionality. Usability tests involved surveys in which we asked volunteers to interact with our website and report different components regarding their satisfaction levels. The two primary goals our design was to achieve easy navigation and well-formatted display of information. The registration, login, inventory, and caloric intake pages were extensively tested as a volunteer interacted with it for roughly around 5 to 10 minutes. Then, a populated database with dummy data was displayed on the inventory page as well as the caloric intake page. Likewise, tests regarding security of the website were conducted. Attacks like SQL injections and privacy attacks were monitored and analyzed to ensure a secure web application.

After passing the local tests, the next step involved cloud deployment. Latency was tested to ensure a good user experience. AWS EC2 instance was the designated platform used for cloud deployment. Since the ML algorithms was computed on the cloud server, the *Route 53* console was used to conduct latency analysis. Likewise, accuracy tests regarding database reads and writes was conducted with accuracy of **100%**. Finally, an exhaustive test was conducted at the end which combines the hardware input with the ML algorithm. Fig. 12 shows the whole process latency that was present in our final product.


Latency of Different Processes

These values were computed as an average of 50 runs for each category.

Figure 12: Process Latency

8 PROJECT MANAGEMENT

8.1 Schedule

The detailed work schedule is shown in full in the back closing pages of this report in Fig. 14. The following aims to enumerate the key milestones that we followed to MVP completion:

- Milestone 1 was the creation of the physical box which has already been completed
- Milestone 2 was forwarding scale input to the RPI which then forwards it to the database
- Milestone 3 was completing the ML classification process with input from our system's camera
- Milestone 4 was the completion of our website design including cloud deployment
- Milestone 5 was reaching our MVP by combining all the components into a cohesive product

8.2 Team Member Responsibilities

The following work distributions amongst us were as follows:

Grace worked primarily on the web applications portion. This consisted of front-end design to create a user-friendly website. The website was composed of various pages using the MVC template from Django to display values in our database. She created a login and registration page that allows users to create accounts using Google

OAuth and monitor their inventory and caloric intake. Furthermore, she worked with Surya to accept user-input in the form of user IDs and buttons on the website to be forwarded to the other hardware components when writing to the database. Lastly, she helped with the creation of the physical product and the integration between ML computation and the back-end database.

Surya led efforts in integrating physical components with the computational elements of the design. To be clear, although this was primarily a software project, it relied heavily on an elegant and effective interplay of sampling, transmission, and updating that had to be done in a concurrent and timely manner. Towards this goal, he worked with Steven on microcontroller scripting and with both Steven and Grace on database management, which was the most challenging aspect of this project. Additionally, he worked with Steven on the camera integration and the "autocapture" feature which filtered and rerouted captured images of food items to directories for the classification algorithm to read from.

Steven primarily focused on the ML computation and classification component of the product. This consisted of gathering the data, training the model, and testing it on verification and validation data. Likewise, he worked with using online libraries and APIs to perform the remaining computations. Lastly, he extensively worked on connecting the results to the database in addition to receiving input from the hardware components. As a result, Steven worked closely with Surya in terms of communication protocols between the peripheral sensors, the RPI, and the computer.

8.3 Bill of Materials and Budget

Please find on the following page, Table 1, which displays the necessary resources used for our project and their corresponding prices.

8.4 Risk Mitigation

Risk Mitigation was an important field of consideration which consisted of the various categories below.

- **USB Camera Synchronization:** There was difficulty in running the image classification camera and the scale reading camera simultaneously. The cameras were higher quality than a Pi Camera which resulted in slow and tasking parallel execution. The RPI would reach unsafe temperature levels, and the cameras reaching deadlock at times. To combat these issues, we decided to do sequential execution in which the first camera takes an image capture then the second camera takes the second capture. This resulted in no noticeable differences in execution speeds and safer execution on the hardware.
- **Mismatched baud rates** were a significant (but thankfully easily-resolved) issue in the context of micro-controller-computer communication. In the context of our project, the "polling feature" required logic on the baud rate which can be controlled in the 'Serial Monitor' of the micro-controller IDE. Through `Serial.begin()` functions, the synchronization ensured that data was transmitted and received at the same rate, which ensured proper sampling.
- **Buffer overflow** and **data loss** posed challenges in micro-controller-based systems that had to interface with external devices. This occurred when incoming data from the scale exceeded the capacity of the micro-controller `receive` buffer, which caused data corruption or subsequent instability. To address this issue, our team implemented control mechanisms in the controller scripting, following XON/X-OFF software flow control protocols. We also incorporated interrupt-driven processing for asynchronous data management and effective buffer forwarding.

  In the context of data loss, we logged incoming data to non-volatile storage options. These involved various hardware backups which the database can read from in the event of system crashes.

The database was the most vulnerable component in this entire project. It was by far the most important, yet held the most potential for bottleneck-related issues and crashing, which resulted in potential user frustration. Several steps were taken in the context of efficient database access and retrieval, and protection against concurrency, injection, and CSRF attacks:

- **Transactions for concurrency:** In multi-threaded and concurrent environments such as the ones employed in this project, where the database can be simultaneously updated by inventory and calorie values, preventing race conditions and data consistency were of paramount interest. Thus, we employed Atomicity, Consistency, Isolation, and Durability (ACID) design properties when implementing transactions. In particular, we implemented a rolling 2-phase locking approach through Django's `Transactions` package to help maintain data integrity and reliable operation of microcontroller-based systems interacting with our MySQL databases.
- **Data Sanitization**: SQL injection attacks and Cross-Site Scripting (XSS) could occur due to a failure to properly sanitize and validate input data when constructing SQL queries. To mitigate this risk, we employed data sanitization on request inputs to separate data from commands, ensuring security and integrity.
- **Database Manipulation**: In relational databases like MySQL, the storage and retrieval of data were crucial for efficient operations. One common method for organizing data was through tree-like structures,

Table 1: Bill of materials

Description	Model #	Manufacturer	Quantity	Cost @	Total
Raspberry Pi Model 4B	Model 4B - 64Bit	RPI Foundation	1	\$49.99	\$49.99
Amazon Essentials Digital Kitchen Scale	N/A	Amazon	1	\$9.99	\$9.99
Arducam 5MP Camera for RPI	IMX708	Arducam	2	\$14.99	\$29.98
Biioones 2K Full HD USB Webcam	PC02 Pro	Biioones Semiconductor	2	\$25.99	\$52.98
Load Cell Amplifier	HX711	Sparkfun	1	\$5.99	\$5.99
Adafruit Jumper Cables	Male-Female	TechSpark Inventory	1 pk	FREE	FREE
RS232 to TTL Converter	MAX3232	Texas Instruments	1	\$9.99	\$9.99
					\$158.92

such as B-trees or B+ trees. These structures enabled logarithmic access and retrieval times; namely, the balanced hierarchy of nodes allowed for efficient searching, insertion, and deletion operations by maintaining a balanced hierarchy of nodes. However, there were scenarios where performance can degrade. For example, query calls using the reverse accessor, which involved traversing the tree in the opposite direction from its typical orientation, could have a worst-case time complexity of $O(n)$, where n is the number of elements in the tree. This degradation in performance could impact the overall efficiency of data retrieval operations.

- **Classification Concerns**: A final consideration pertained to classification accuracy and text extraction. Regarding the classification accuracy, we had the algorithm output the top three classification results instead of classifying an object with the top value. The risk mitigation involved using an algorithm to double-check if the classification result was a food item. In addition to that, there was bound checking done with regards to calorie and weight extraction using the various OCR algorithms. We standardized weights that were extracted incorrectly to a reasonable amount after bound checking to reduce incorrect calorie calculations. These mitigation efforts greatly reduced the risk in the inaccuracies of the various ML models we employed.

9 RELATED WORK

There are two Capstone projects that are closely related to our idea. The first being Food Tracker (team B6 Spring 2022) [10]. This group implemented a food tracking system to keep track of inventory for grocery items using very similar features as us such as camera recognition and database display. Likewise, A Smart Kitchen Assistant (team D3 Spring 2021) also completed a project involving inventory tracking [4]. These projects have provided us with good inspiration and advice as we seek to develop our product.

In addition to these projects, Samsung has created smart refrigerators that do inventory tracking. These have costs of a couple thousand dollars due to their extensive AI technology, significantly beyond the planned design costs for our design. Nonetheless, the blueprint of Samsung's modern refrigerator technology informs and inspires much of the features we aim to highlight in our MVP.

Overall, our product has similarities with both the Capstone projects and Samsung smart appliances[7]. However, there is novelty in our calorie-tracking feature which we hope to highlight in our design. Likewise, we aim to leverage versatility in product operation; our vision for this product is integration into smart appliances in the kitchen setting, with interfacing capabilities from a mini-fridge to a pantry.

10 SUMMARY

NutrientMatch was a novel project that involved efficient inventory tracking and accumulation of caloric consumption that can easily be integrated into food storage systems and displayed on laptops and smartphones. We sought to incorporate camera and scale readings with ML classification and recognition algorithms to store data that can easily be accessed via a website. In the end, we were successful in fulfilling the design and use-case requirements for our MVP goals while fulfilling some stretch goals that involved user-user interaction on the web application.

The primary limitations of our performance included less automation than we initially expected. To take images, we needed to manually run a script in a linux shell of the RPI. In addition to that, our text extraction accuracy was limited by the Tesseract OCR algorithm which led to poor label extraction. Lastly, our product fell short in terms of some of the website features we hoped to implement. For example, the graphing feature involved manual inputs rather than a fully automated logging and graphing process that we had envisioned.

We could improve our system by implementing buttons into the website that run scripts on the RPI rather than having a shell open to run directly on the RPI. Likewise, we could improve our text extraction by using different algorithms or model averaging to use multiple forms of OCR. Lastly, since web design was not the primary focus of our project, we could have simplified the web application to enhance the primary features of our product more.

As much as we would have loved to continue working on this project beyond graduation, we have decided to not pursue this idea more. We plan to optimize various features like the ML capabilities as well as complete some of the stretch goals we had. These initiatives will be primarily for learning purposes rather than for the sake of pursuing this project more.

10.1 Lessons Learned

One lesson we learned was the importance of organization and proper documentation. Our project consisted of hundreds of git commits within each group member. There were many instances where we needed to rollback commits to return to a previous version. The lack of specificity in the commits resulted in misunderstandings between group members as well as difficulty in differentiating what was changed. There was a wide variety of code changes being submitted every hour (i.e. RPI code, Django code, ML), so better documentation would have resulted in less confusion and less time spent debugging.

Another lesson we learned was the importance of slack in scheduling. There were multiple instances of hardware components breaking and compatibility issues with the integration of various software components that complicated our schedule. However, we were able to address a majority of these issues in a timely manner due to proper planning and the usage of slack during planning.

Lastly, we realized that the beginning design process is such a crucial part in making a good product at the end. All the assignments and guides helped steer us in the right direction. By being specific in the beginning and considering multiple implementation plans, we reduced a lot of the workload towards the end of the project. Also, when we eventually ran into issues, we were able to pivot to other design choices easily. This provided us flexibility in the end which we greatly valued.

Overall, we learned so much through this project, and we are glad to have applied knowledge spanning all our time here at CMU into a real-world application. Also, we were lucky enough to have great faculty and peer support at every step. Ultimately, we were proud of what we created and appreciative for the whole 18-500 Capstone experience.

Glossary of Acronyms

Please find below frequently-referenced acronyms used throughout this design report:

- API - Application Programming Interface
- CMU - Carnegie Mellon University
- CNN - Convolutional Neural Network
- FPS - Frames per Second
- GPIO - General Purpose Input/Output
- IC - Image Classification
- ML - Machine Learning
- MP - MegaPixels
- MVP - Minimum Viable Product
- MVC - Model-View-Controller Architecture
- OCR - Optical Character Recognition
- RPI - Raspberry Pi
- SSOCR - Seven-Segment Optical Character Recognition
- SVM - Support Vector Machines
- TTL - Transistor-Transistor Logic
- XSS - Cross-Site Scripting

References

[1] URL: `https://www.postgresql.org/docs/current/datatype-json.html`.

[2] URL: `https://www.bbcgoodfood.com/review/best-kitchen-scales`.

[3] Dishaa Agarwal. URL: `https://www.analyticsvidhya.com/blog/2021/04/insight-into-svm-support-vector-machine-along-with-code/#:~:text=Soft%20margin%20SVM%20maximizes%20the,maximizes%20the%20margin%20between%20classes.`.

[4] A Smart Kitchen Assistant. URL: `http://course.ece.cmu.edu/~ece500/projects/s21-teamd3/`.

[5] Jonathan Deber. “How Much Faster is Fast Enough? User Perception of Latency & Latency Improvements in Direct and Indirect Touch”. In: *Conference on Human Factors in Computing Systems* (2015).

[6] Alexander Gillis. URL: `https://www.techtarget.com/searchdatamanagement/definition/MongoDB`.

[7] Team HomeServe. URL: `https://www.homeserve.com/en-us/blog/buying-guide/samsung-smart-refrigerator/`.

[8] Jeremy Hsu. “ChatGPT outperforms humans at labelling some data for other AIs”. In: *The Scientist* (2023).

[9] OpenFoodFacts. URL: `https://world.openfoodfacts.org`.

[10] Food Tracker. URL: `http://course.ece.cmu.edu/~ece500/projects/s22-teamb6/`.

[11] Yue Yuan. “Vegetable and fruit freshness detection based on deep features and principal component analysis”. In: *Current Research in Food Science* (2024).

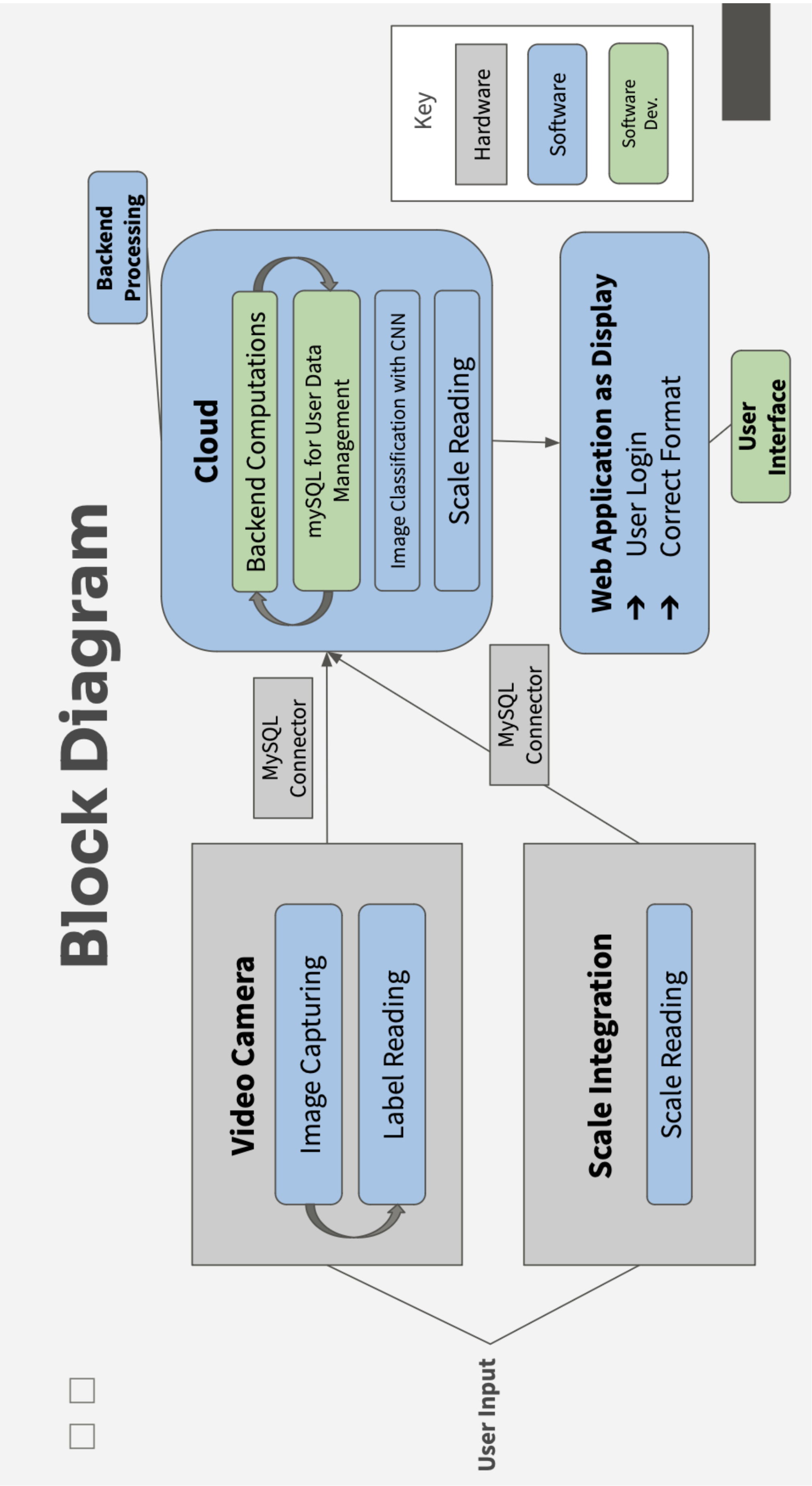
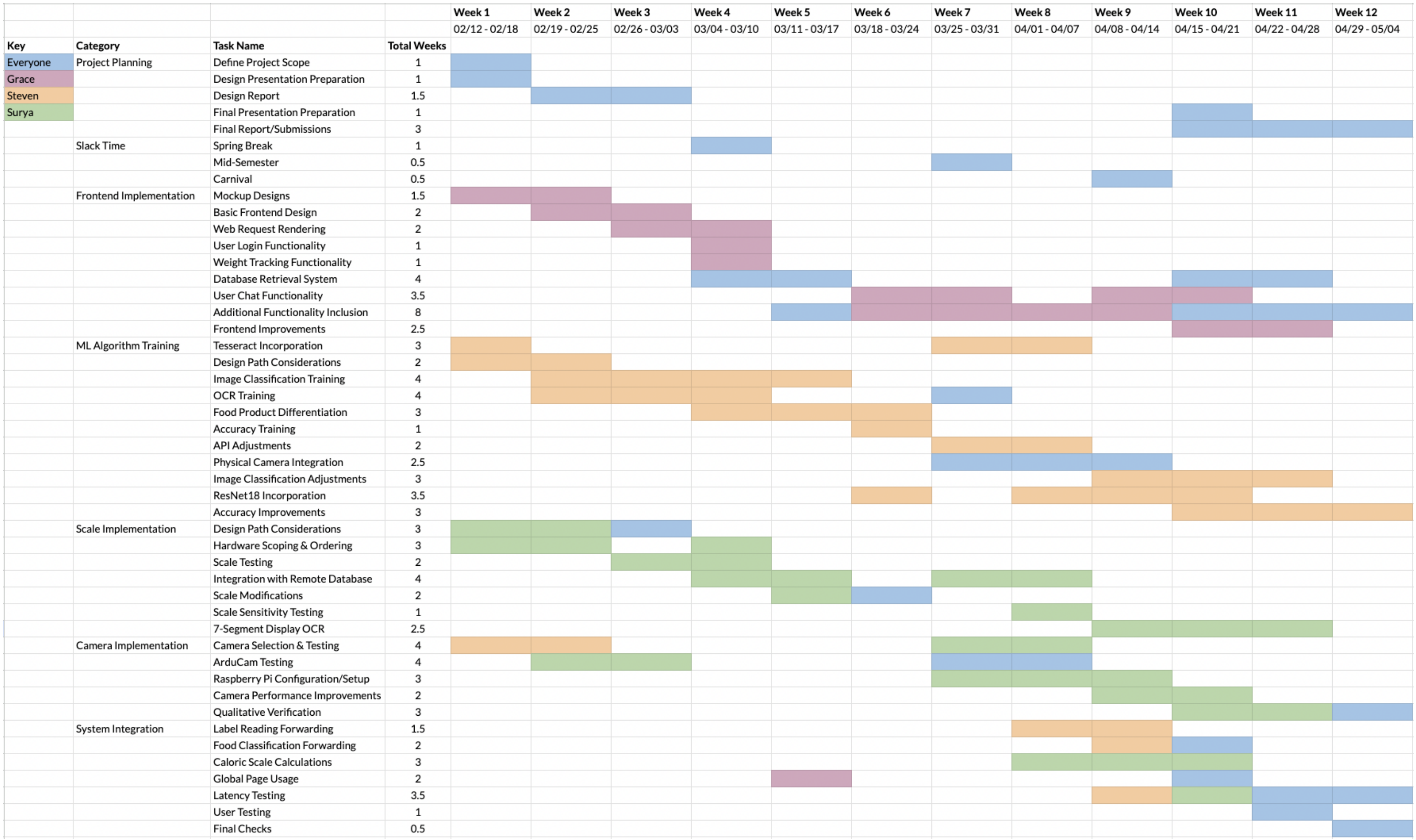


Figure 13: Full-page version of the system block diagram as described earlier.



				Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12
				02/12 - 02/18	02/19 - 02/25	02/26 - 03/03	03/04 - 03/10	03/11 - 03/17	03/18 - 03/24	03/25 - 03/31	04/01 - 04/07	04/08 - 04/14	04/15 - 04/21	04/22 - 04/28	04/29 - 05/04
Key	**Category**	**Task Name**	**Total Weeks**												
🟦 Everyone	Project Planning	Define Project Scope	1	🟦											
🟪 Grace		Design Presentation Preparation	1	🟦											
🟧 Steven		Design Report	1.5		🟦	🟦									
🟩 Surya		Final Presentation Preparation	1										🟦		
		Final Report/Submissions	3										🟦	🟦	🟦
	Slack Time	Spring Break	1				🟦								
		Mid-Semester	0.5							🟦					
		Carnival	0.5									🟦			
	Frontend Implementation	Mockup Designs	1.5	🟪	🟪										
		Basic Frontend Design	2		🟪	🟪									
		Web Request Rendering	2			🟪	🟪								
		User Login Functionality	1				🟪								
		Weight Tracking Functionality	1				🟪								
		Database Retrieval System	4				🟦	🟦					🟦	🟦	
		User Chat Functionality	3.5						🟪	🟪		🟪	🟪		
		Additional Functionality Inclusion	8					🟦	🟪	🟪	🟪	🟪	🟦	🟦	🟦
		Frontend Improvements	2.5										🟪	🟪	
	ML Algorithm Training	Tesseract Incorporation	3	🟧						🟧	🟧				
		Design Path Considerations	2	🟧	🟧										
		Image Classification Training	4		🟧	🟧	🟧	🟧							
		OCR Training	4		🟧	🟧	🟧			🟦					
		Food Product Differentiation	3				🟧	🟧	🟧						
		Accuracy Training	1						🟧						
		API Adjustments	2							🟧	🟧				
		Physical Camera Integration	2.5							🟦	🟦	🟦			
		Image Classification Adjustments	3									🟧	🟧	🟧	
		ResNet18 Incorporation	3.5						🟧		🟧	🟧	🟧		
		Accuracy Improvements	3										🟧	🟧	🟧
	Scale Implementation	Design Path Considerations	3	🟩	🟩	🟦									
		Hardware Scoping & Ordering	3	🟩	🟩		🟩								
		Scale Testing	2			🟩	🟩								
		Integration with Remote Database	4				🟩	🟩		🟩	🟩				
		Scale Modifications	2					🟩	🟦						
		Scale Sensitivity Testing	1								🟩				
		7-Segment Display OCR	2.5									🟩	🟩	🟩	
	Camera Implementation	Camera Selection & Testing	4	🟧	🟧					🟩	🟩				
		ArduCam Testing	4		🟩	🟩				🟦	🟦				
		Raspberry Pi Configuration/Setup	3							🟩	🟩	🟩			
		Camera Performance Improvements	2									🟩	🟩		
		Qualitative Verification	3										🟩	🟩	🟦
	System Integration	Label Reading Forwarding	1.5								🟧	🟧			
		Food Classification Forwarding	2									🟧	🟦		
		Caloric Scale Calculations	3								🟩	🟩	🟩		
		Global Page Usage	2					🟪							
		Latency Testing	3.5									🟧	🟩	🟦	🟦
		User Testing	1											🟦	
		Final Checks	0.5												🟦

Figure 14: Gantt Chart