WHERE'S THE MILK?

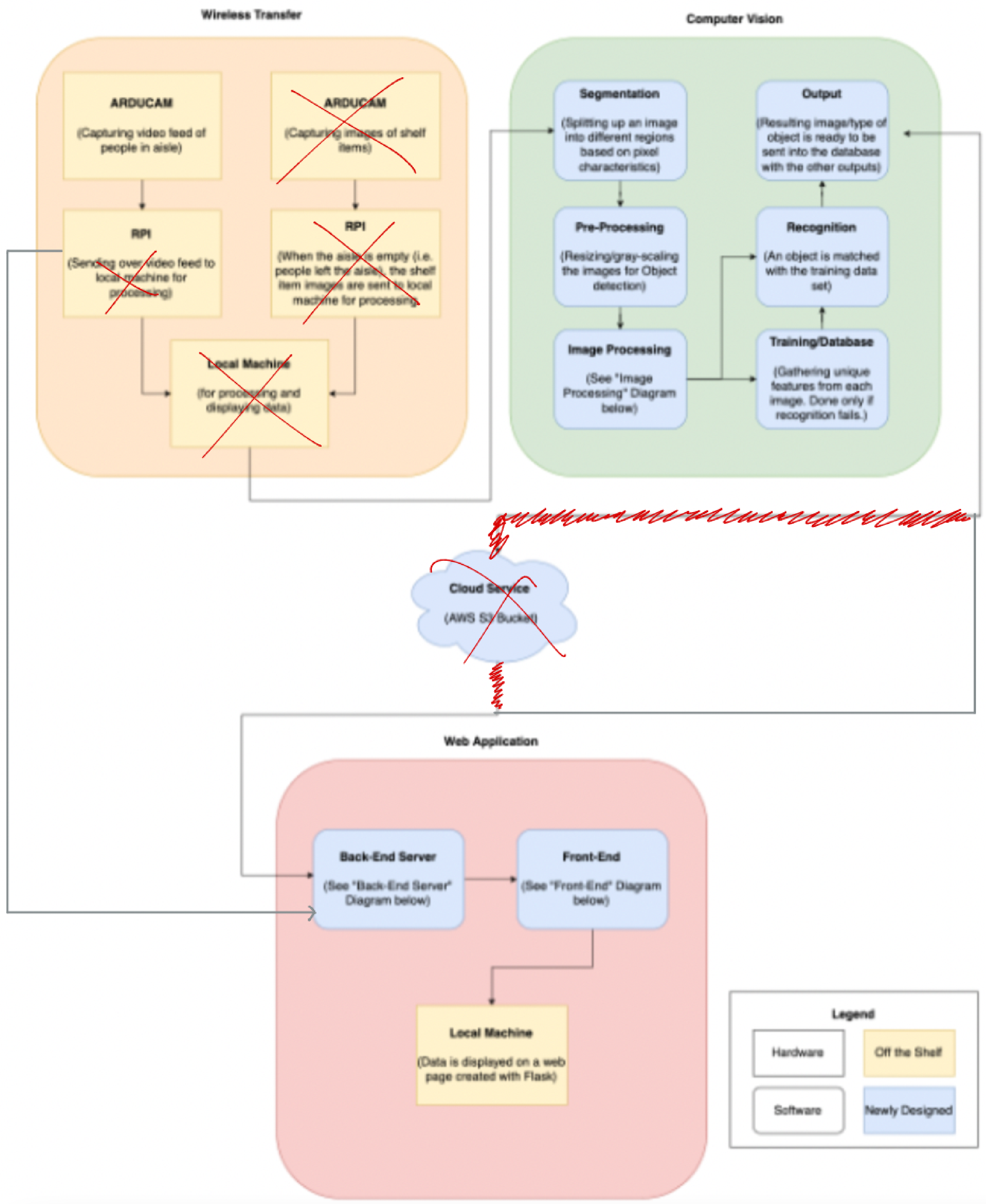
PRESENTED BY ALLEN DING
AUTHORED BY ALLEN DING, LUCKY WAVAI, TAKSHSHEEL GOSWAMI

RECAP

- **Problem**
  - Existing inventory management systems don't always have real-time analysis of information:
    - traffic detection
    - presence of items of a shelf
  - Current systems typically require expensive heavy infrastructure
- **Technology we propose**
  - Focus a CV camera system (Raspberry PI/ARDUCAM) to detect traffic, which then triggers a laptop to take a picture of a shelf, which is fed into an object detection algorithm, and the results are displayed on a web application (all done on the same laptop).

USE-CASE REQUIREMENTS

- **Detect when there are people in the aisle**
  - From various research papers, we found that experiments with people detection achieved an accuracy rate of between 90-95%. Thus, **90%** will be our threshold.
- **Detect the presence of objects once people have left the aisle**
  - From various research papers, we found that experiments with SIFT achieved an estimated accuracy of 60-70% in 5-6 seconds. Thus, **60%** will be our threshold.
- **Information must be transmitted and displayed to the manager's local machine** within **15 seconds** of someone entering/leaving the aisle
  - From various research papers, we found that experiments with SIFT were on average 3-6 seconds per experiment. We are detecting people, then multiple objects, so we doubled the experiment time and added 3 seconds of leeway for the multiple object detection and information transfer.


Wireless Transfer
ARDUCAM
(Capturing video feed of people in aisle)
~~ARDUCAM~~
~~(Capturing images of shelf items)~~
RPI
(Sending over video feed to local machine for processing)
~~RPI~~
~~(When the aisle is empty (i.e. people left the aisle), the shelf item images are sent to local machine for processing)~~
~~Local Machine~~
~~(for processing and displaying data)~~
Computer Vision
Segmentation
(Splitting up an image into different regions based on pixel characteristics)
Pre-Processing
(Resizing/gray-scaling the images for Object detection)
Image Processing
(See "Image Processing" Diagram below)
Output
(Resulting image/type of object is ready to be sent into the database with the other outputs)
Recognition
(An object is matched with the training data set)
Training/Database
(Gathering unique features from each image. Done only if recognition fails.)
~~Cloud Service~~
~~(AWS S3 Bucket)~~
Web Application
Back-End Server
(See "Back-End Server" Diagram below)
Front-End
(See "Front-End" Diagram below)
Local Machine
(Data is displayed on a web page created with Flask)
Legend
Hardware
Off the Shelf
Software
Newly Designed

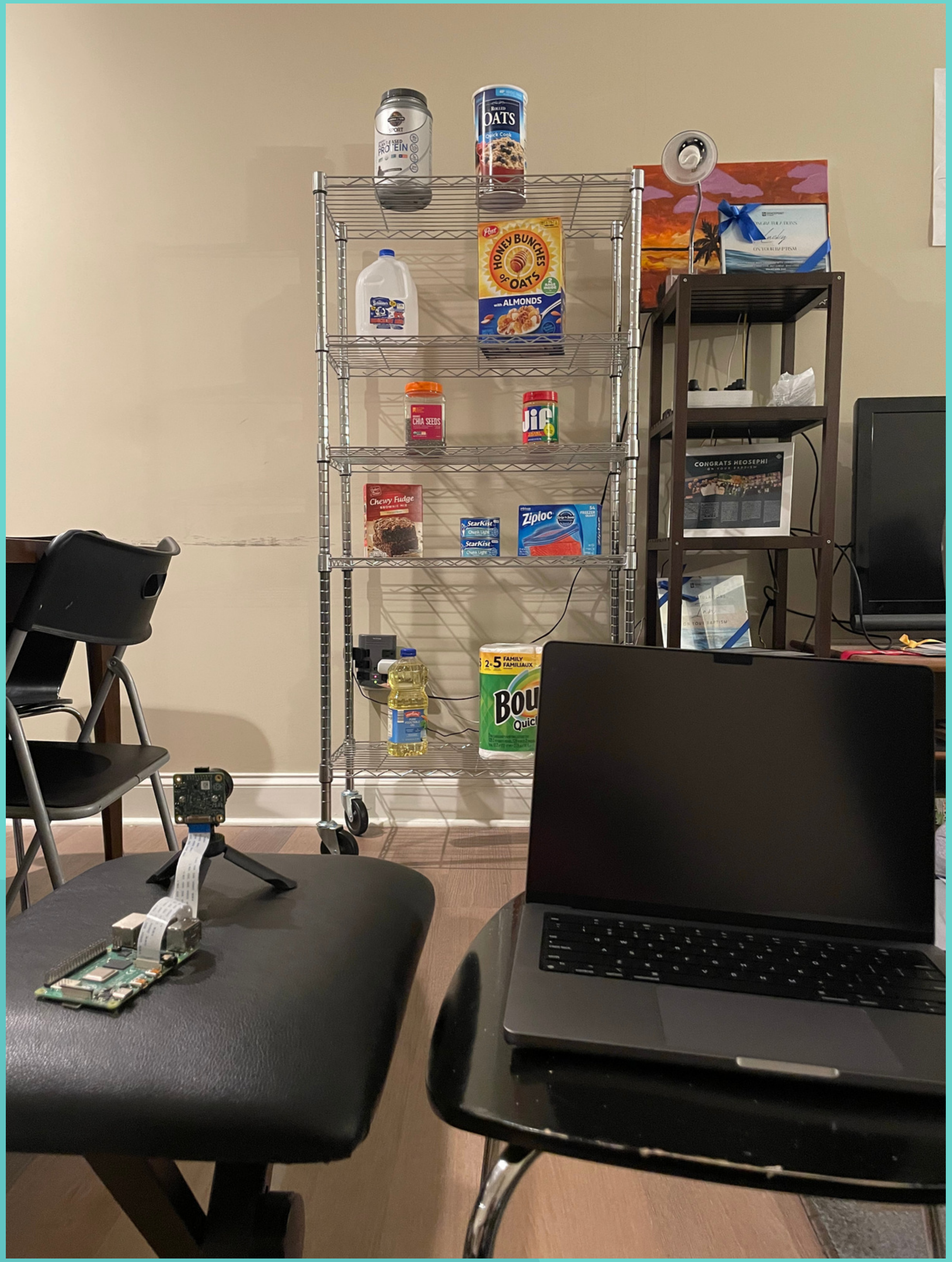
BEFORE (DURING DESIGN REVIEW PRESENTATION)

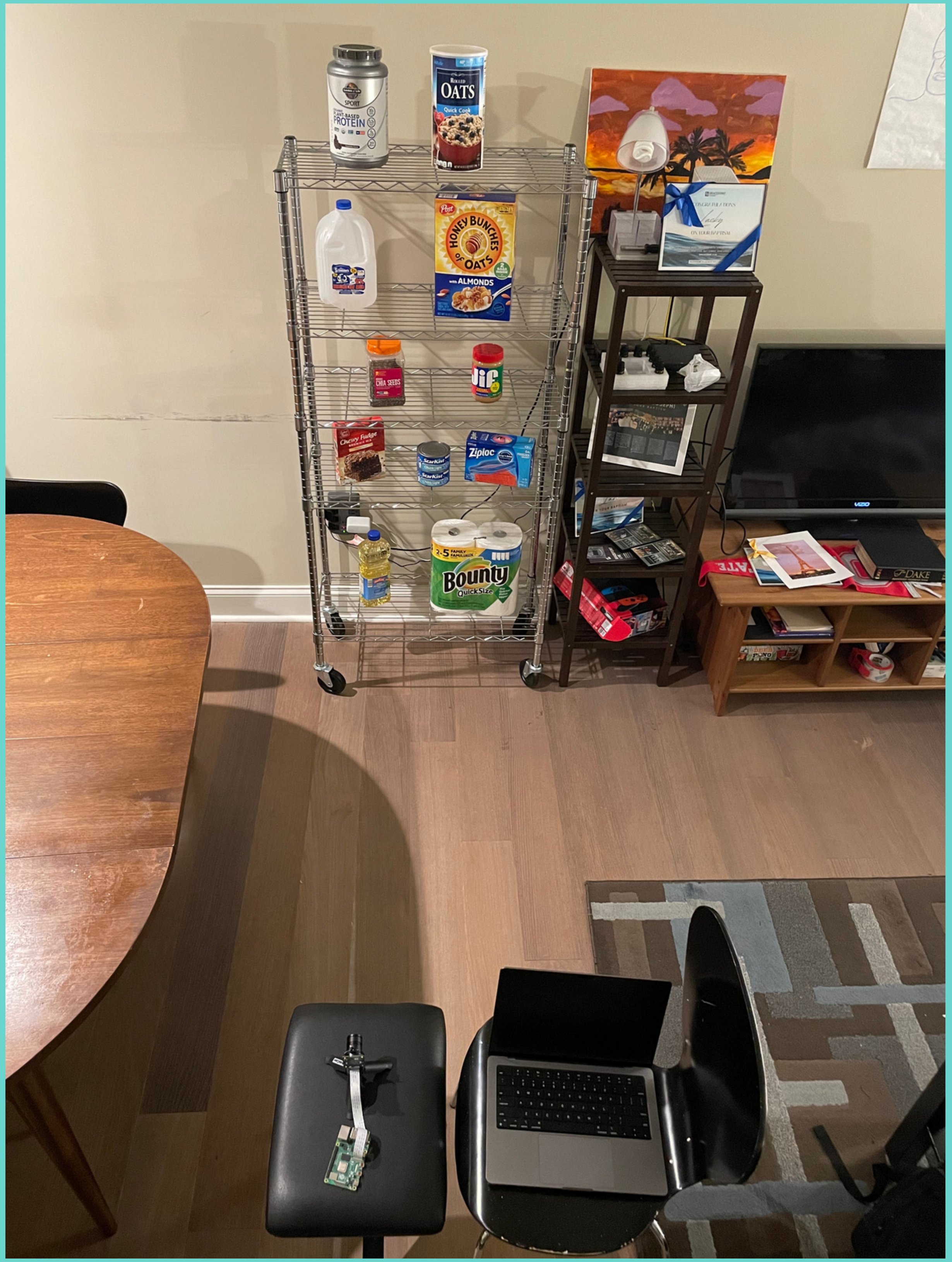
- **Wireless Transfer**
  - Two sets of Raspberry Pi's (RPI) with an ARDUCAM
    - People-detection System
      - Detects for people and triggers the object detection system (Lucky's laptop) once a person is found
    - Object-detection system
      - Sends images to Lucky's laptop for processing
- **Computer Vision (Object Detection)**
  - Takes in images from the RPI to process image
  - Sends results to web application
- **Web Application**
  - Takes in data from the object detection algorithm

CURRENT DESIGN

- **Wireless Transfer**
  - One set with an RPI and an ARDUCAM
    - People-detection System
      - Detects for people and triggers the web application to run the object detection algorithm, which then will display the results on the Front-End
- **Computer Vision (Object Detection)**
  - Lucky is in the process of turning his subsystem into a library for me to integrate with the web application.
- **Web Application**
  - Listens for an trigger from the wireless transfer system
  - Calls on Lucky's object detection code that was imported
  - No longer hosted on AWS (reach goal if we finish testing/integration before the final demo)

COMPLETE SOLUTION





- Here, we see a makeshift portrayal of what our final presentation would look like.
    - We have the shelf of items roughly 5ft spaced from the laptop (which will have a webcam attached to it) and the RPI next to it.
    - (The laptop and RPI will be placed on a table during demo day)
- What will be displayed on the laptop will be the web application, showing the presence of items on the shelf retrieved from the object detection code.

TESTING, VERIFICATION, AND METRICS

- In the upcoming slides, we will discuss the following points for the three aforementioned subsystems (wireless system, object detection, web application):
    - Quantitative measurements of the goodness and functionality of the design of the subsystem
    - How well did our implementation perform?
    - Trade-offs?

WIRELESS SYSTEM

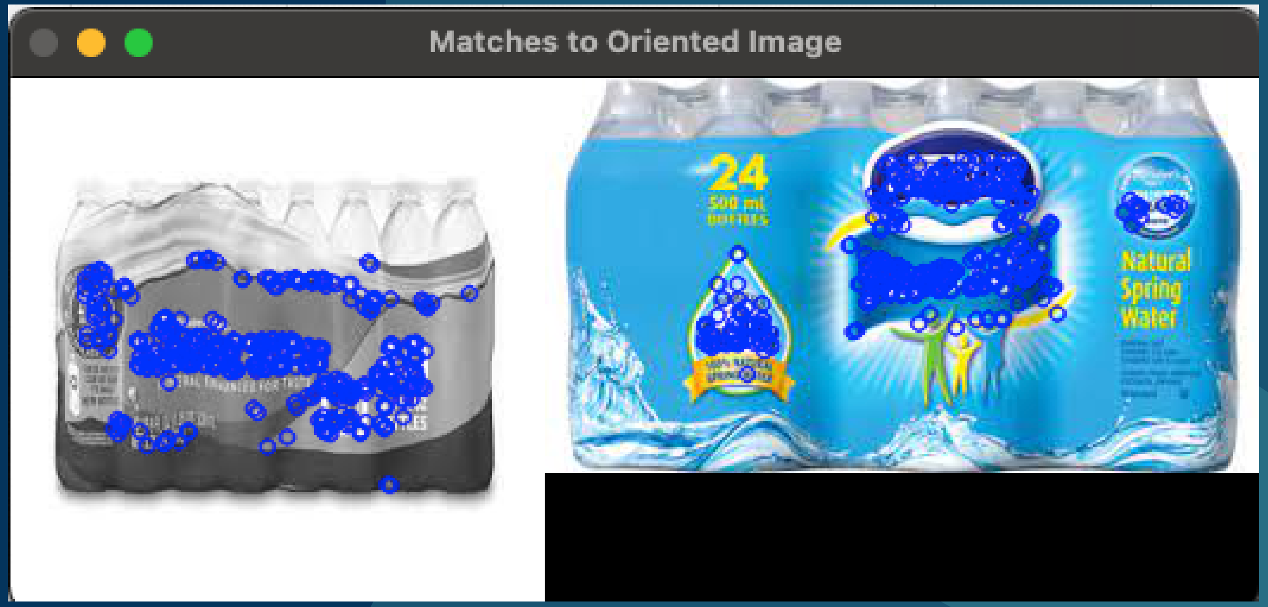
- The way our people detection algorithm works is that it treats people as objects, just like it would for any other object. That way, any motion across the aisle will be detected.
  - A passing test would be detecting a human as an object (so far, that has not failed)
- In short, our user requirements for detecting a person at an **accuracy rate of 90% or higher was met**
- Transmission to trigger the web application to run the object detection code is still being worked on
  - Have tried SSH, HTTP Requests, and we are likely sticking with HTTP Requests
  - Transmission is usually instant (will time the speed once we integrate the wireless system with the web application)

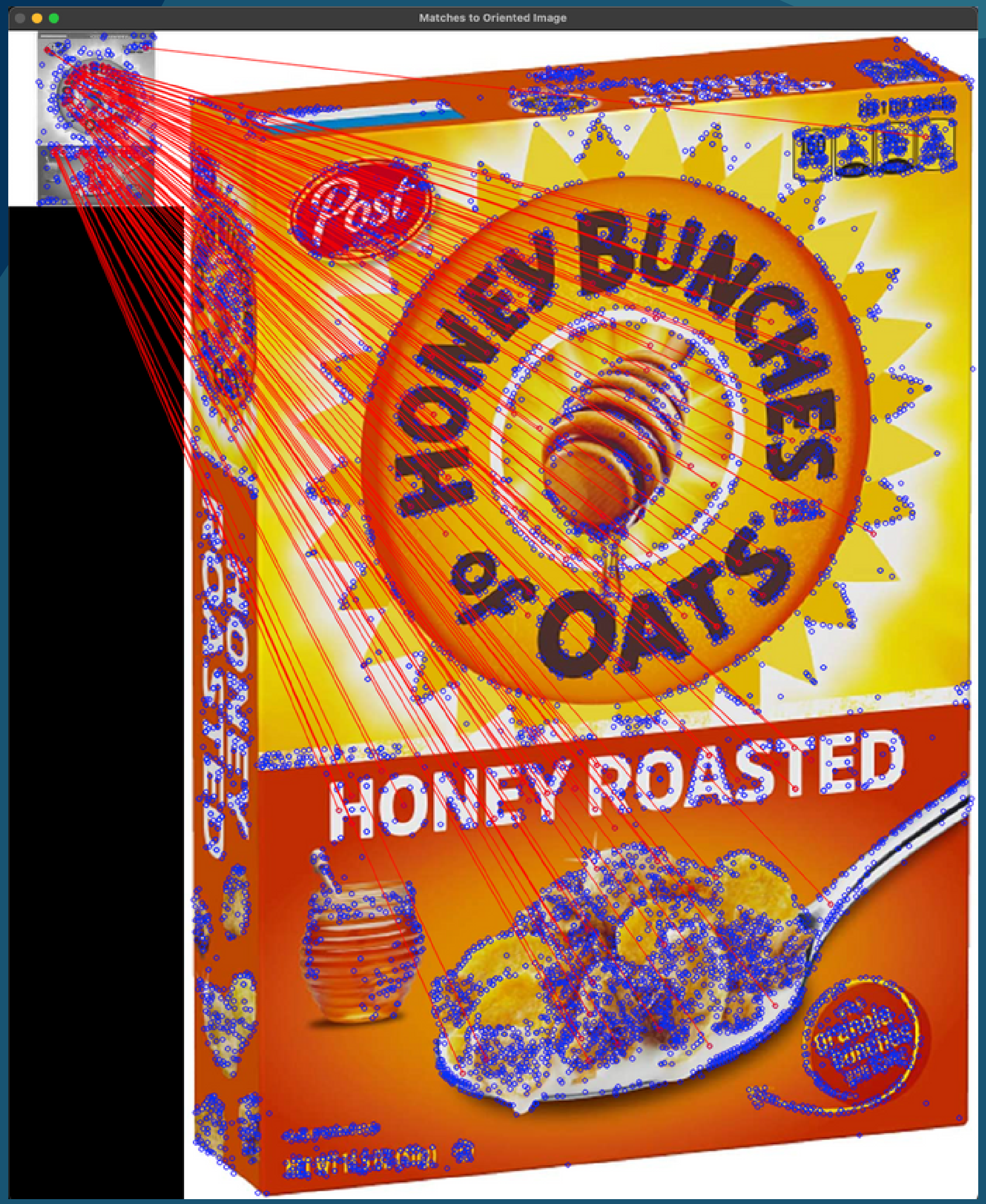
WIRELESS SYSTEM (CONT.)

- Some things that we have to consider are:
  - Security of transmission of data to the web application
    - Currently, through HTTP Requests we use the IP address of the host laptop as the endpoint, which of course poses security issues if our code base is hacked
  - Sensitivity of motion through the aisle
    - Currently, the algorithm to detect for moving objects is very sensitive. Therefore, we will have to determine a threshold value (i.e. the number of times the algorithm detects a moving object) so that the object detection algorithm is not triggered multiple times when a user is still walking/standing in an aisle.
      - That threshold value is being tested right now and will be finalized this week.

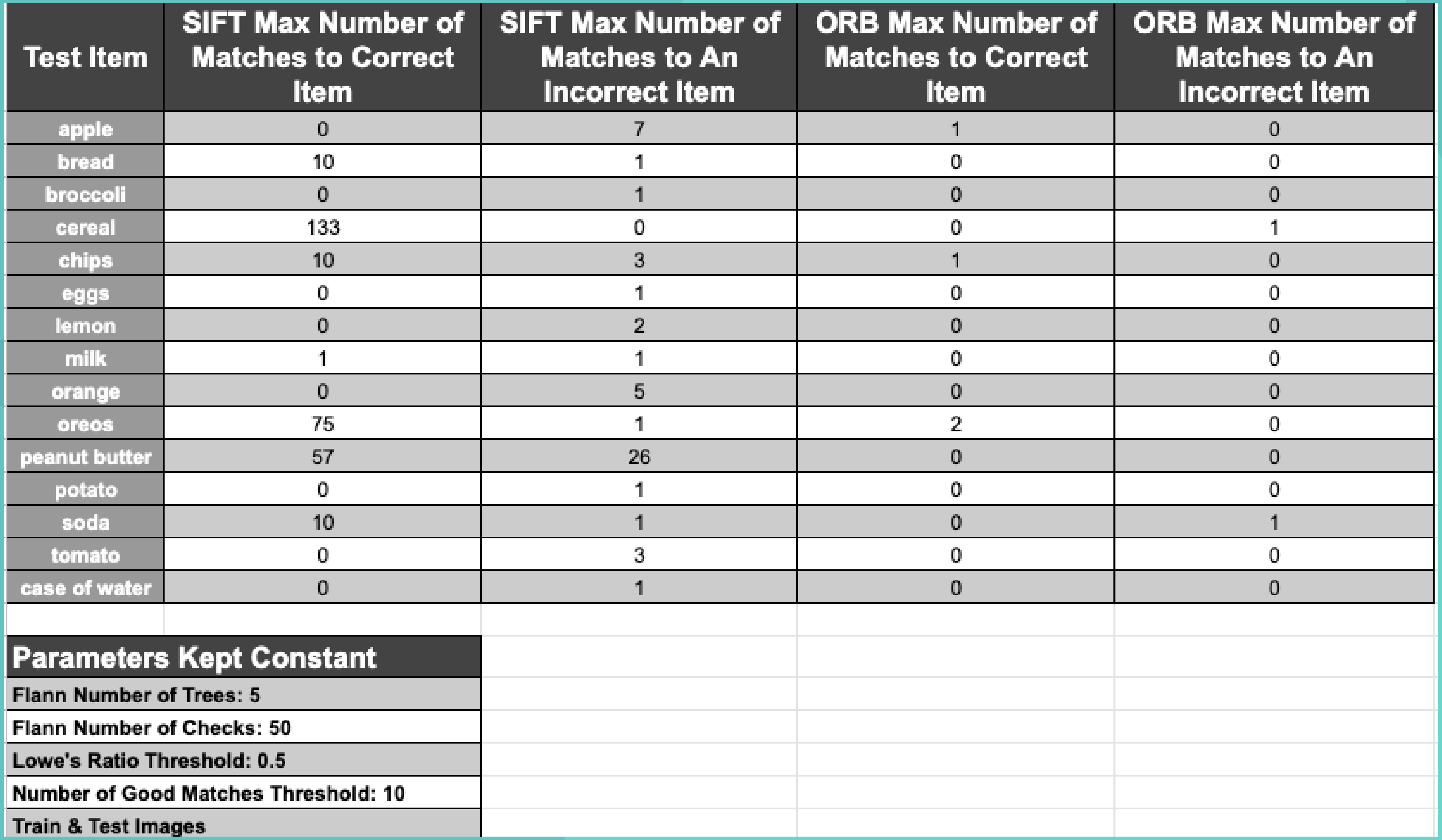
PRESENCE TEST

- As discussed in the design review, one of our testing areas of focus was in the object detection subsystem
- We proceeded with development of the subsystem in incremental testable stages following the Presence Test model we described in the outline
  - Test One parameter at a time i.e. object detection algorithm, FLANN Matcher Parameters, Lowe's Ratio Threshold, and number of good matches
    - Do multiple tests for that one parameter with all other parameters held constant and change the other parameters incrementally
  - We repeated these tests, and continue to in order to decide on optimal settings for accuracy and speed





EXAMPLE PRESENCE



Test Item	SIFT Max Number of Matches to Correct Item	SIFT Max Number of Matches to An Incorrect Item	ORB Max Number of Matches to Correct Item	ORB Max Number of Matches to An Incorrect Item
apple	0	7	1	0
bread	10	1	0	0
broccoli	0	1	0	0
cereal	133	0	0	1
chips	10	3	1	0
eggs	0	1	0	0
lemon	0	2	0	0
milk	1	1	0	0
orange	0	5	0	0
oreos	75	1	2	0
peanut butter	57	26	0	0
potato	0	1	0	0
soda	10	1	0	1
tomato	0	3	0	0
case of water	0	1	0	0

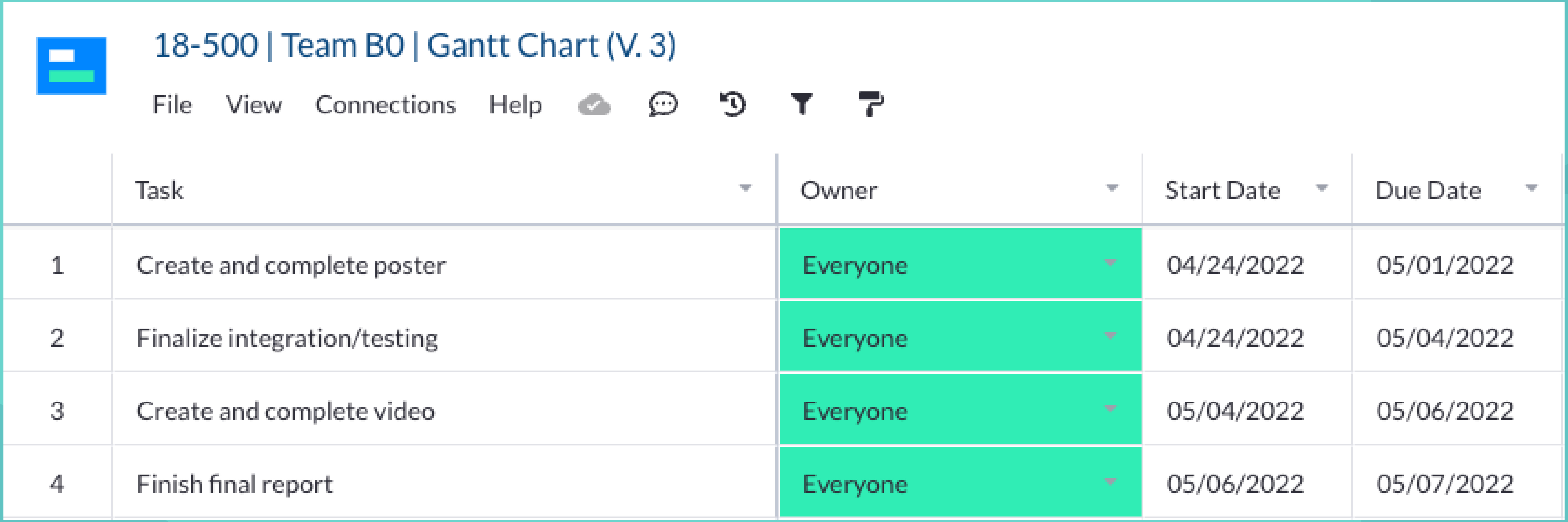
Parameters Kept Constant
Flann Number of Trees: 5
Flann Number of Checks: 50
Lowe's Ratio Threshold: 0.5
Number of Good Matches Threshold: 10
Train & Test Images

- In this example we show one of the tests we performed when deciding which detector to use between ORB and SIFT
- We found through our tests one intended result and one unintended
- Intended Result: we found the detector that produced the most correct good match disparity over incorrect good match (false positives) => SIFT
- Unintended Result: We found that some items prove difficult for all of the tested algorithms
    - Items with labels resulted in more accurate results i.e. cereals over items without labels i.e. fruits

WEB APPLICATION

- Aspects that have been tested along the way are:
  - Ensure all links work properly (i.e. no HTTP errors, links route where they should)
  - User authentication
    - Username/password are cross-reference with the database before sign-in/registration
  - Password encryption
    - The passwords are hashed with a randomly generated "salt" (randomly generated word) that is placed somewhere in the password (dependent on the algorithm/user (if they created their own encryption method)) to mitigate hackers' efforts
  - Visibility
    - Only signed-in users can view certain web pages
    - Only the "admin" user can view the database as a UI in the app
  - Security of inputs on web pages
    - form submissions are cross-referenced with data in the backend to ensure that even if a user tampers with the submission (i.e. through the inspect tool), the inputs will not be processed/saved in the database
- Results:
  - Everything thus far has been successful, but more testing will be done once integration is complete
- Trade-off:
  - We chose the Flask framework to prioritize speed over the breadth of functionality that comes with frameworks such as Django. Flask is very "bare-bones", meaning the user has full control on how complex the application can be.

Project Management (Final Schedule)



18-500 | Team B0 | Gantt Chart (V. 3)

	Task	Owner	Start Date	Due Date
1	Create and complete poster	Everyone	04/24/2022	05/01/2022
2	Finalize integration/testing	Everyone	04/24/2022	05/04/2022
3	Create and complete video	Everyone	05/04/2022	05/06/2022
4	Finish final report	Everyone	05/06/2022	05/07/2022