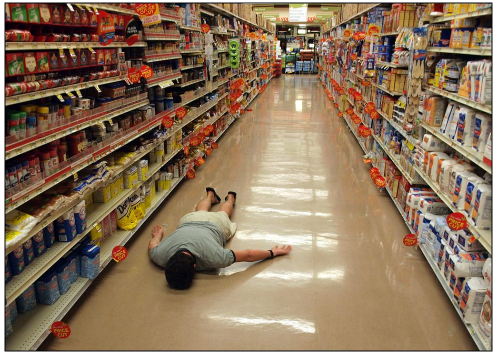
B6 Food Tracker

Jaeyoon Choi, Zhengze Gong, Keaton Drebes

Use case (review)

- You’re grocery shopping, and you forgot to make a list of things to buy.
- You don’t remember whether you have milk, or eggs, or that one ingredient for that one recipe.
- You bought something and completely forgot about it, leaving it in the fridge for ages

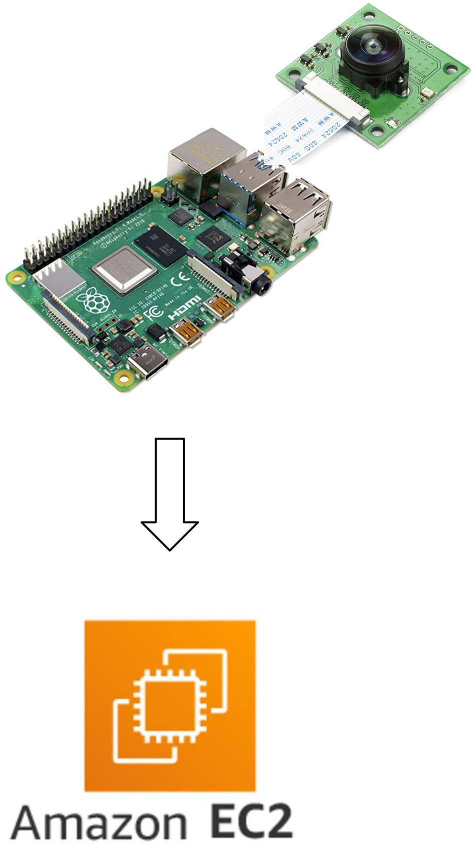


Use Case Requirements

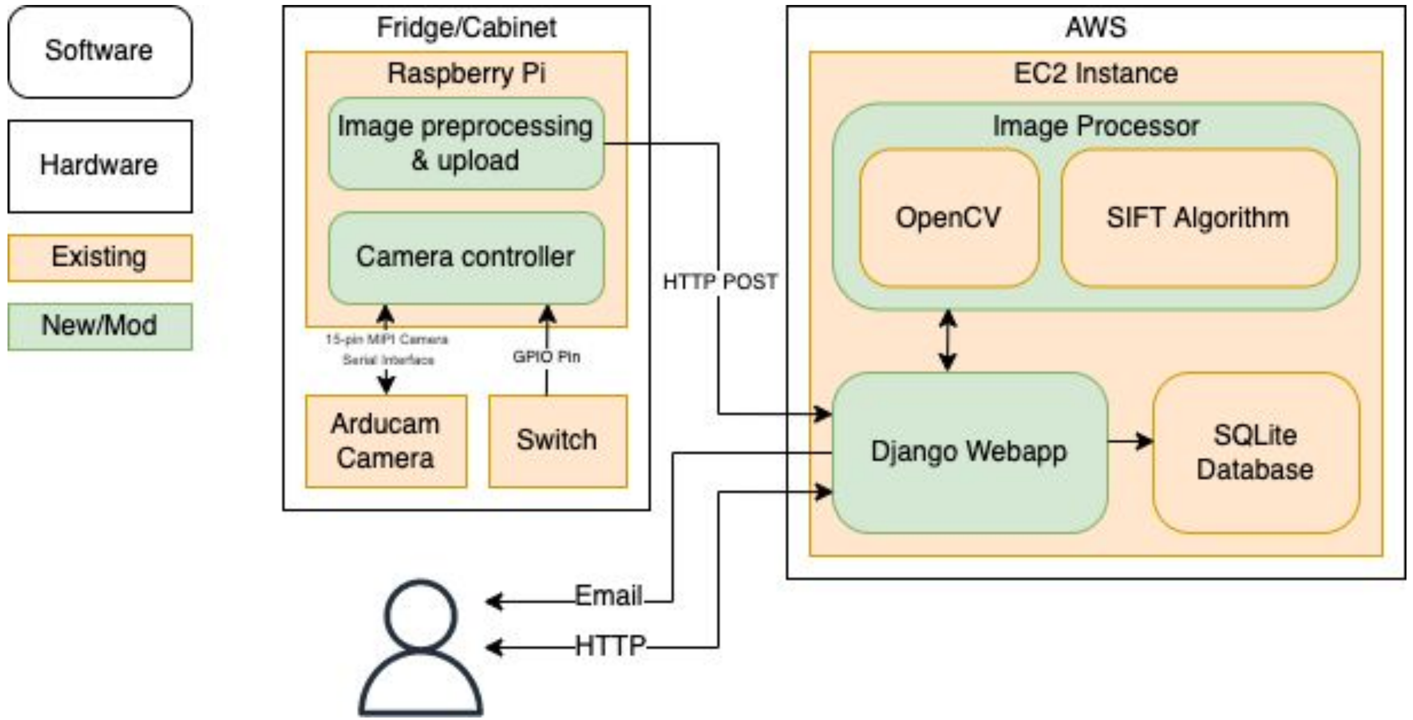
- Support 10 items: Milk, Eggs, Yogurt, Cheese, Butter, Orange Juice, Cereal, Canned Beans, Pasta, and Ritz Crackers
- Support multiple users
- Combined inventory from multiple source devices for any user
- Object identification accuracy > 85%
    - < 5% mis-identification (we prefer failure to identify over mis-identification)
- < 10s latency to update the inventory after the door closes
- < 1s latency between door close and taking the photo

Updated Solution Approach

- RPI with embedded camera for hardware component ~~(originally Jetson Nano)~~
  - No need for dedicated hardware
  - Infrequent usage (≈ # of times you open a fridge)
  - Use case is lag tolerant
- Camera is 5 megapixel, Arducam OV5647
- Open CV for software (now runs in the cloud on dedicated EC2 instance)
  - Usinging SIFT for feature detection
- Django for web-app
  - SQLite backend
  - AJAX frontend
- Communication will just be posting JSON
  - To avoid malicious actors, we will use asymmetric cryptography

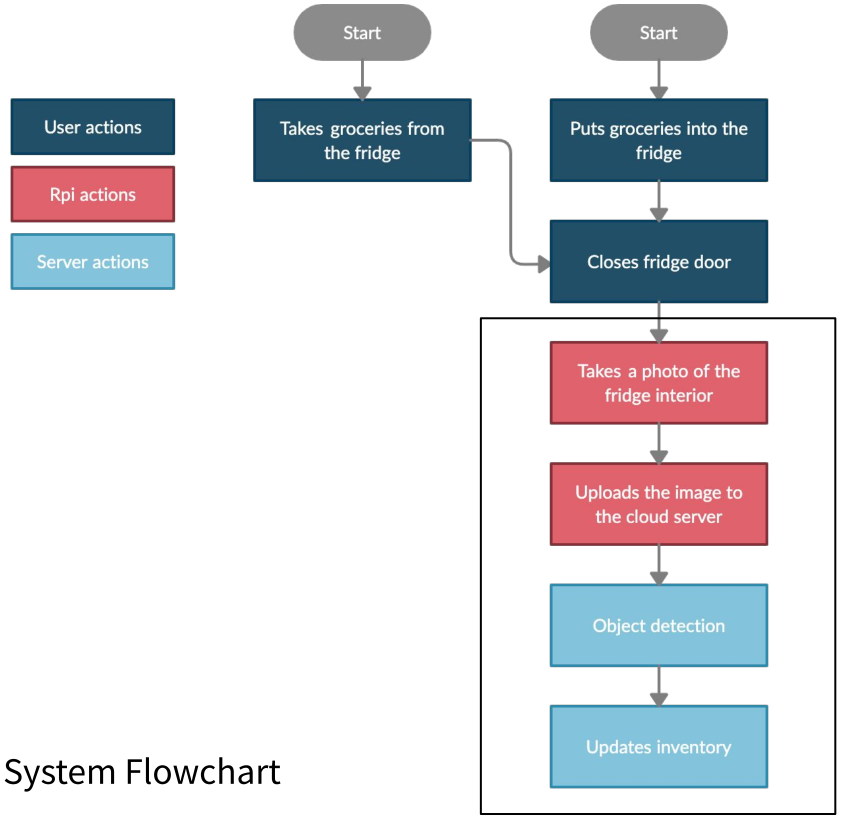


High level implementation plan

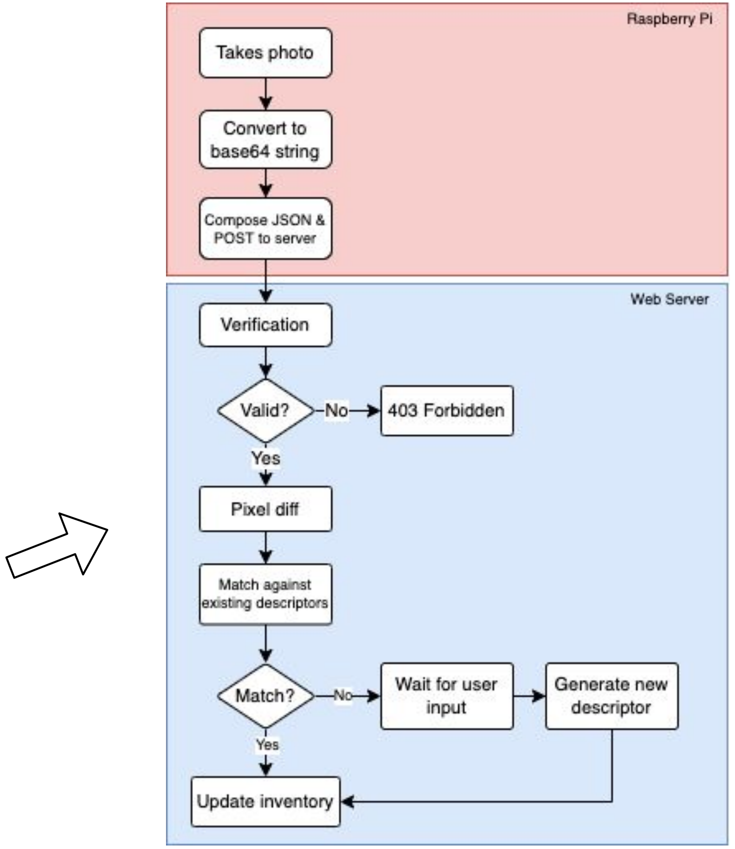


System Block Diagram

High level implementation plan



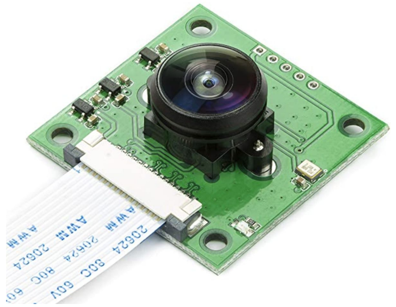
User actions
Rpi actions
Server actions
Start
Start
Takes groceries from the fridge
Puts groceries into the fridge
Closes fridge door
Takes a photo of the fridge interior
Uploads the image to the cloud server
Object detection
Updates inventory

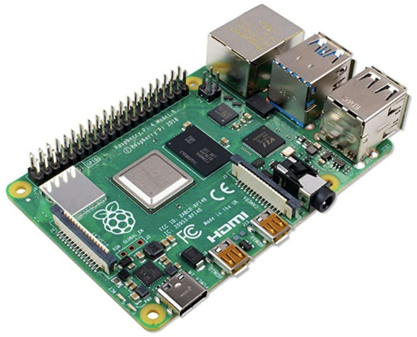


System Flowchart

Implementation plan, hardware component

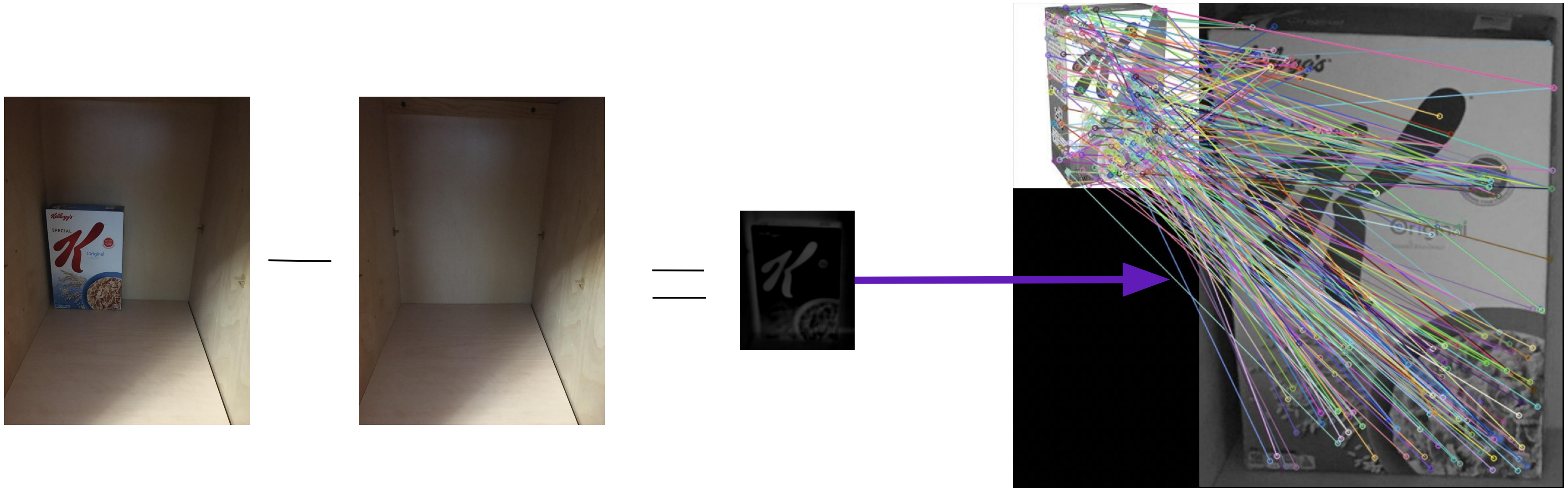
- Arducam OV5647
  - ¼'' sensor, 5 Megapixel
  - Connect through 15-pin MIPI Camera Serial Interface
  - Control with the picamera library
- RPI 3B
  - RPI will know its serial ID, and private key
- POST serialized JSON containing:
  - “Contents”: image encoded with base64
  - “Serial ID”: 64 bit serial_ID
  - “Validation string”: (Some agreed upon string, encrypted)
- Use validation string and public-private key to avoid malicious actors
- Send POST requests with empty “contents” field periodically as heartbeat messages to indicate its active status





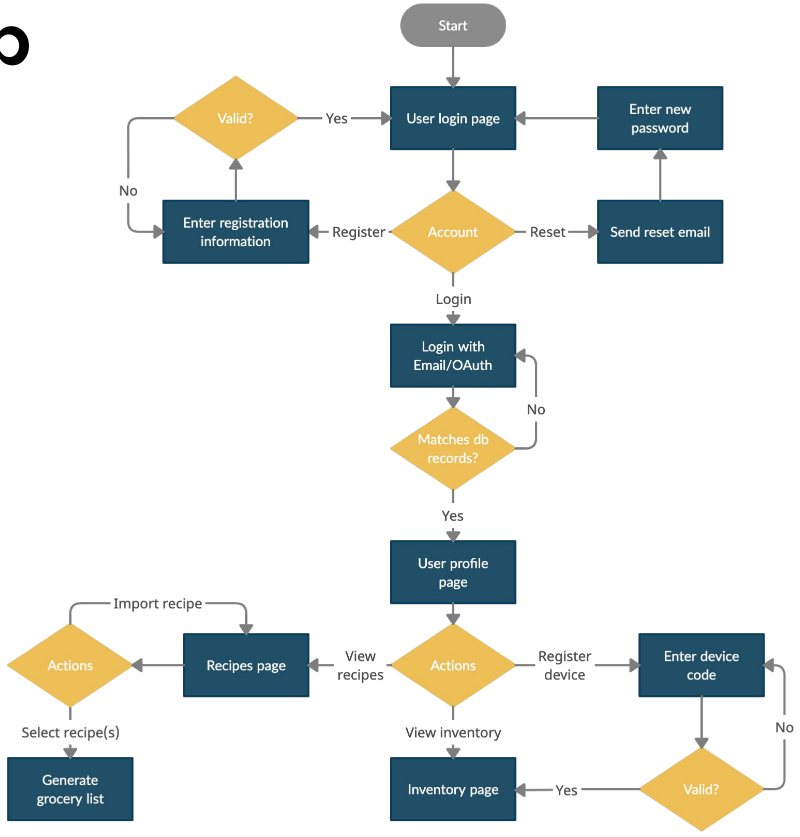
Implementation plan, CV component

- Perform a pixel difference to localize object
- Classify object with SIFT using pre-generated descriptors
- If classification fails, prompt user, and perform new object registration



Implementation plan, webapp

- User login
  - Django native library django-social-auth, works with Google, Facebook, Twitter
- Bootstrap for CSS
- Ajax for dynamic updating of storage information
- Send push notification via browser to the user if they request a grocery list, if their device is inactive/offline, or if an item cannot be classified

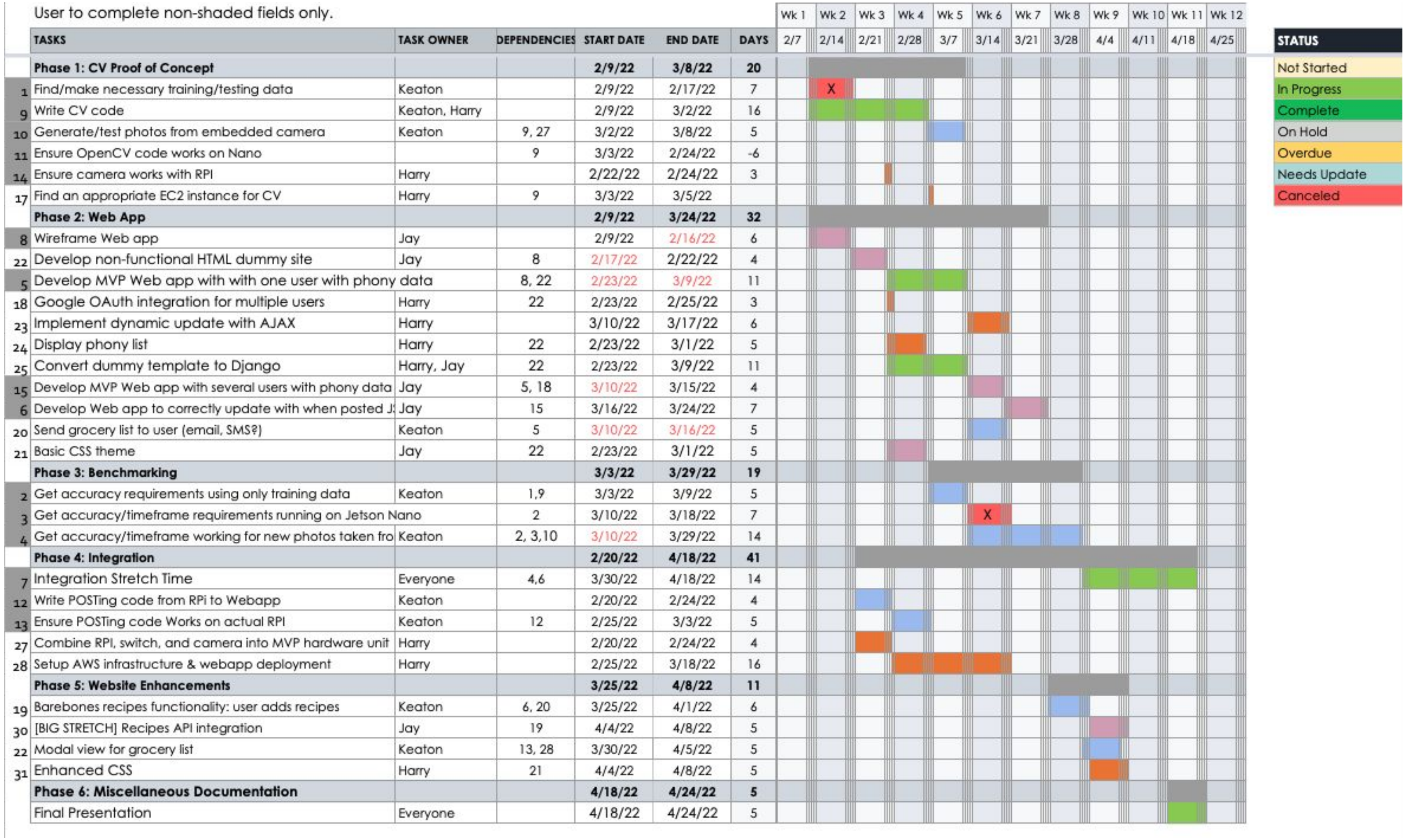


Metrics and Validation

Requirement	Metric	Verification Method
System latency	< 10s	Update fridge inventory, record the time between closing the fridge door and receiving the AJAX data from the server
Hardware latency	< 1s	Update fridge inventory, record the time between closing the fridge door and the image being saved in the RPi file system, verify that the image taken matches the current inventory
CV accuracy	> 85% accuracy < 5% mis-ID	Accuracy on test dataset of fridge interior photos taken with the RPi camera
Support multiple users	100%	Test with two users, each registered with one fridge. Update items for one fridge and check if only the owner's inventory is updated
Support multiple devices	100%	Register one user with two fridges, update items for both fridges, and check if the user's inventory is updated accordingly

Risk and Mitigation

Risk	Mitigation
Cannot achieve required CV accuracy/mis-ID rate	Tweak with the confidence threshold and image pixel difference threshold
Cannot achieve required system latency	Optimize CV algorithm, upgrade EC2 instance, reduce image size



User to complete non-shaded fields only.

	TASKS	TASK OWNER	DEPENDENCIES	START DATE	END DATE	DAYS
	Phase 1: CV Proof of Concept			**2/9/22**	**3/8/22**	**20**
1	Find/make necessary training/testing data	Keaton		2/9/22	2/17/22	7
9	Write CV code	Keaton, Harry		2/9/22	3/2/22	16
10	Generate/test photos from embedded camera	Keaton	9, 27	3/2/22	3/8/22	5
11	Ensure OpenCV code works on Nano		9	3/3/22	2/24/22	-6
14	Ensure camera works with RPI	Harry		2/22/22	2/24/22	3
17	Find an appropriate EC2 instance for CV	Harry	9	3/3/22	3/5/22	
	Phase 2: Web App			**2/9/22**	**3/24/22**	**32**
8	Wireframe Web app	Jay		2/9/22	2/16/22	6
22	Develop non-functional HTML dummy site	Jay	8	2/17/22	2/22/22	4
5	Develop MVP Web app with with one user with phony data		8, 22	2/23/22	3/9/22	11
18	Google OAuth integration for multiple users	Harry	22	2/23/22	2/25/22	3
23	Implement dynamic update with AJAX	Harry		3/10/22	3/17/22	6
24	Display phony list	Harry	22	2/23/22	3/1/22	5
25	Convert dummy template to Django	Harry, Jay	22	2/23/22	3/9/22	11
15	Develop MVP Web app with several users with phony data	Jay	5, 18	3/10/22	3/15/22	4
6	Develop Web app to correctly update with when posted J	Jay	15	3/16/22	3/24/22	7
20	Send grocery list to user (email, SMS?)	Keaton	5	3/10/22	3/16/22	5
21	Basic CSS theme	Jay	22	2/23/22	3/1/22	5
	Phase 3: Benchmarking			**3/3/22**	**3/29/22**	**19**
2	Get accuracy requirements using only training data	Keaton	1,9	3/3/22	3/9/22	5
3	Get accuracy/timeframe requirements running on Jetson Nano		2	3/10/22	3/18/22	7
4	Get accuracy/timeframe working for new photos taken fro	Keaton	2, 3,10	3/10/22	3/29/22	14
	Phase 4: Integration			**2/20/22**	**4/18/22**	**41**
7	Integration Stretch Time	Everyone	4,6	3/30/22	4/18/22	14
12	Write POSTing code from RPi to Webapp	Keaton		2/20/22	2/24/22	4
13	Ensure POSTing code Works on actual RPI	Keaton	12	2/25/22	3/3/22	5
27	Combine RPI, switch, and camera into MVP hardware unit	Harry		2/20/22	2/24/22	4
28	Setup AWS infrastructure & webapp deployment	Harry		2/25/22	3/18/22	16
	Phase 5: Website Enhancements			**3/25/22**	**4/8/22**	**11**
19	Barebones recipes functionality: user adds recipes	Keaton	6, 20	3/25/22	4/1/22	6
30	[BIG STRETCH] Recipes API integration	Jay	19	4/4/22	4/8/22	5
22	Modal view for grocery list	Keaton	13, 28	3/30/22	4/5/22	5
31	Enhanced CSS	Harry	21	4/4/22	4/8/22	5
	Phase 6: Miscellaneous Documentation			**4/18/22**	**4/24/22**	**5**
	Final Presentation	Everyone		4/18/22	4/24/22	5

Wk 1	Wk 2	Wk 3	Wk 4	Wk 5	Wk 6	Wk 7	Wk 8	Wk 9	Wk 10	Wk 11	Wk 12
2/7	2/14	2/21	2/28	3/7	3/14	3/21	3/28	4/4	4/11	4/18	4/25

STATUS
Not Started
In Progress
Complete
On Hold
Overdue
Needs Update
Canceled

Updated Gantt Chart