12th Yam Final Report

Westerfield, Jonathan
jgwesterfield@tamu.edu

Campos, Abdul
acampos0297@tamu.edu

Mendiratta, Mannan
mannan1999@tamu.edu

Rodriguez, Ismael
ir3987@tamu.edu

Patlovany, Daniel
daniel.patlovany@tamu.edu

Todd, Aaron
altodd@tamu.edu

December 5, 2019

Abstract

This project was under development for CSCE 431 for the Fall 2019 semester. We will be working for the 12th Can Food Pantry to implement an inventory management system. The 12th Can previously tracked its inventory using differences in weight after adding/removing inventory. In addition, this was done in Google Sheets. This system does not specify individual types of inventory and does not give detailed analysis of the inventory supply or its trends. It can make it difficult to know how much of each item is stocked and doesn't help with determining other helpful statistics, such as seeing which items are more popular than others.Our new system allows them to keep inventory of individual items, add new items and manage members who have access to the system. The main objective is to make tracking inventory easier and faster as well as providing at-a-glance insight into the current inventory levels of the pantry. This specifies management of all items, not necessarily just food items but diapers, tote bags, etc. The Inventory Site will be hosted with Texas AM University Department of Student Affairs IT. The project was implemented using tools such as: PHP 7.3, Laravel 6.6.0, Heroku, Github, Dusk, PHPUnit, Pivotal Tracker and others. The final product was demonstrated and approved by the client.

Contents

1 Introduction

Our client, Vince Potter, represents the 12th Can Food Pantry in College Station, Texas. Our application will provide and easy to create items to track, add and remove inventory, and provide at-a-glance insight into the current supply of the system. Our goal is essentially to make their core processes easier and more meaningful.

2 Motivation

Our main motivation was to improve the workflow the 12th Can has to go through between pantry openings. Between openings, they must go through and see how much was taken from the opening, and then add inventory from donations and other sources. Since their current version of tracking inventory is done through Google Sheets and by weight, it is very difficult to derive any meaningful insight into the inventory they currently have stocked. In addition, it is also difficult to track changes and keep a readable audit history. There was a need to fix this and bring them the capabilities a university sponsored organization should have.

3 Stakeholders

Client: Vince Potter (Representative), David Chapa (End User)
Professor: Philip Ritchey
Team Members: Jonathan Westerfield, Abdul Campos, Ismael Rodriguez, Mannan Mendiratta, Daniel Patlovany, Aaron Todd

4 Lo-Fi Mockups and the Final Site

4.1 Login Page

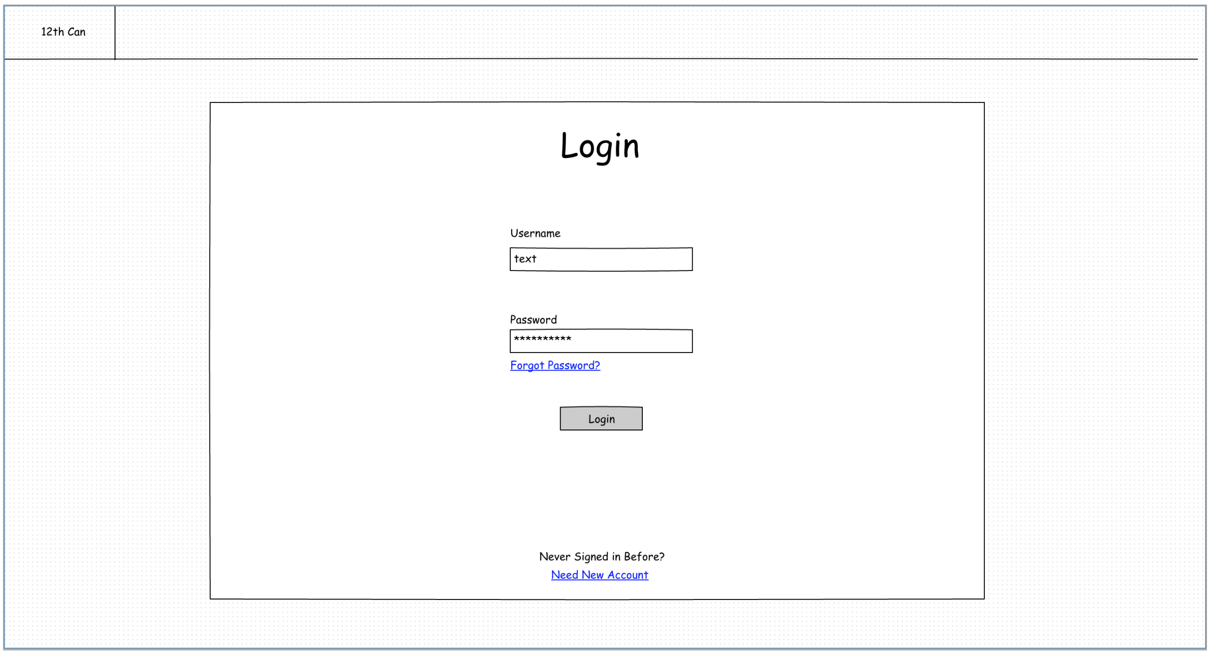
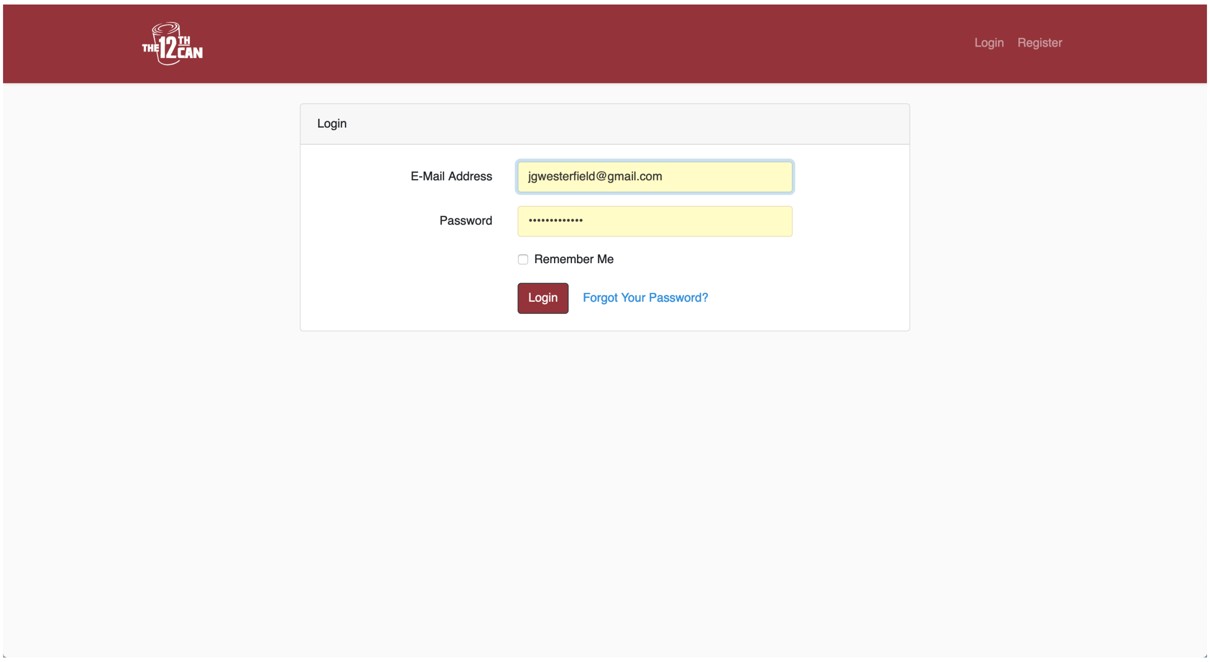


Figure 1: Login Screen Mockup



E-Mail Address jgwesterfield@gmail.com

Password ••••••••••••••

□ Remember Me

Figure 2: Final Login Screen

4.2 Inventory Dashboard

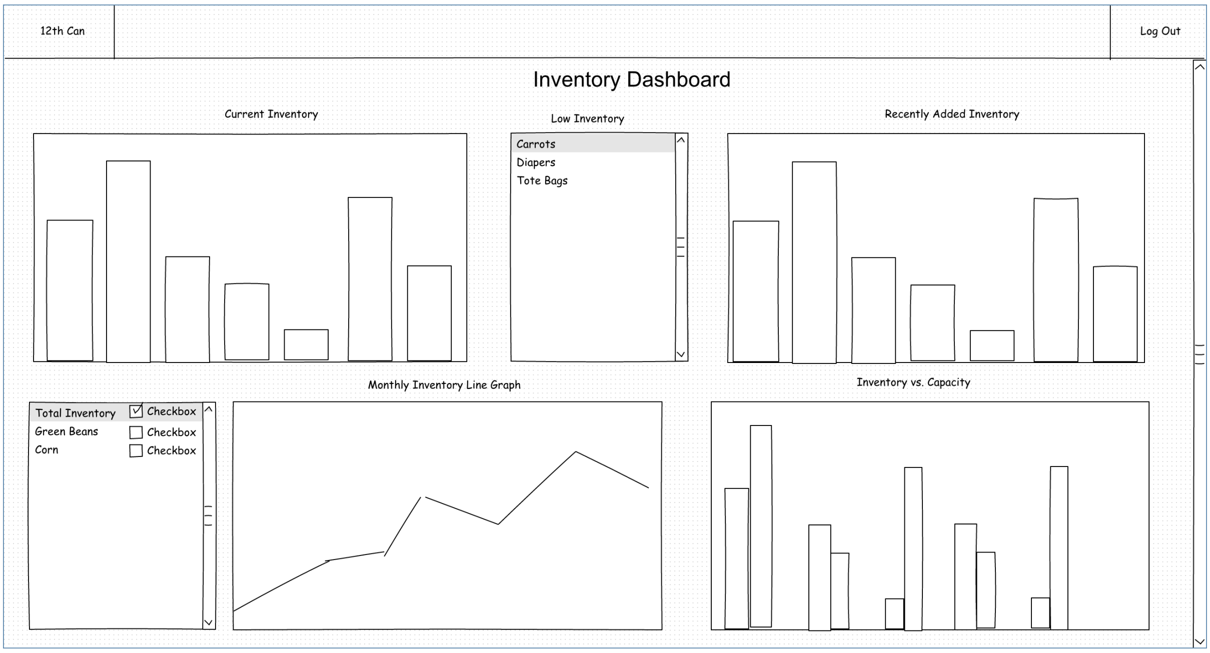
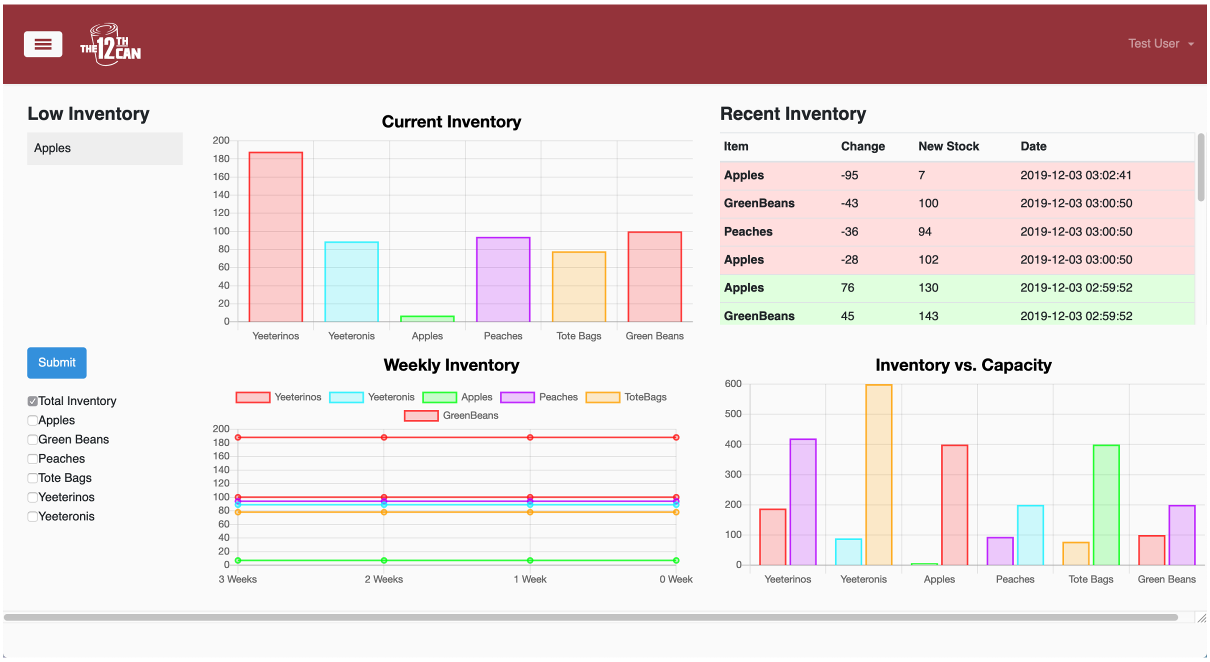


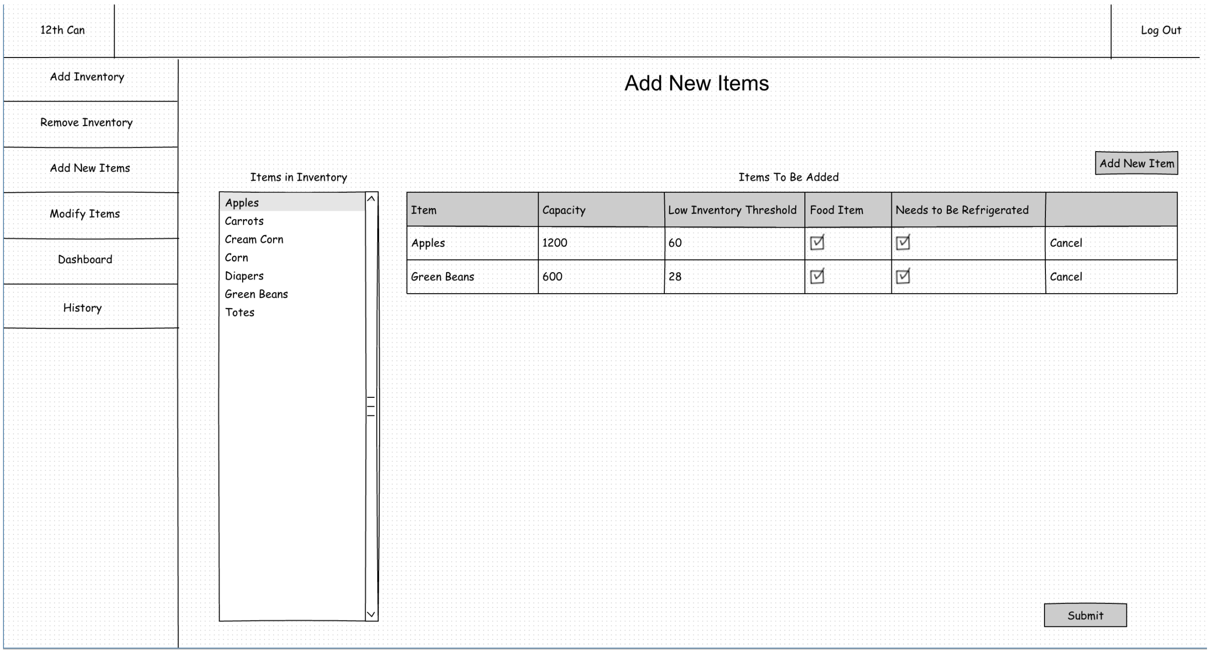
Figure 3: Inventory Dashboard Mockup



Item	Change	New Stock	Date
Apples	-95	7	2019-12-03 03:02:41
GreenBeans	-43	100	2019-12-03 03:00:50
Peaches	-36	94	2019-12-03 03:00:50
Apples	-28	102	2019-12-03 03:00:50
Apples	76	130	2019-12-03 02:59:52
GreenBeans	45	143	2019-12-03 02:59:52

Figure 4: Final Inventory Dashboard

4.3 Add New Items Page



Add New Items

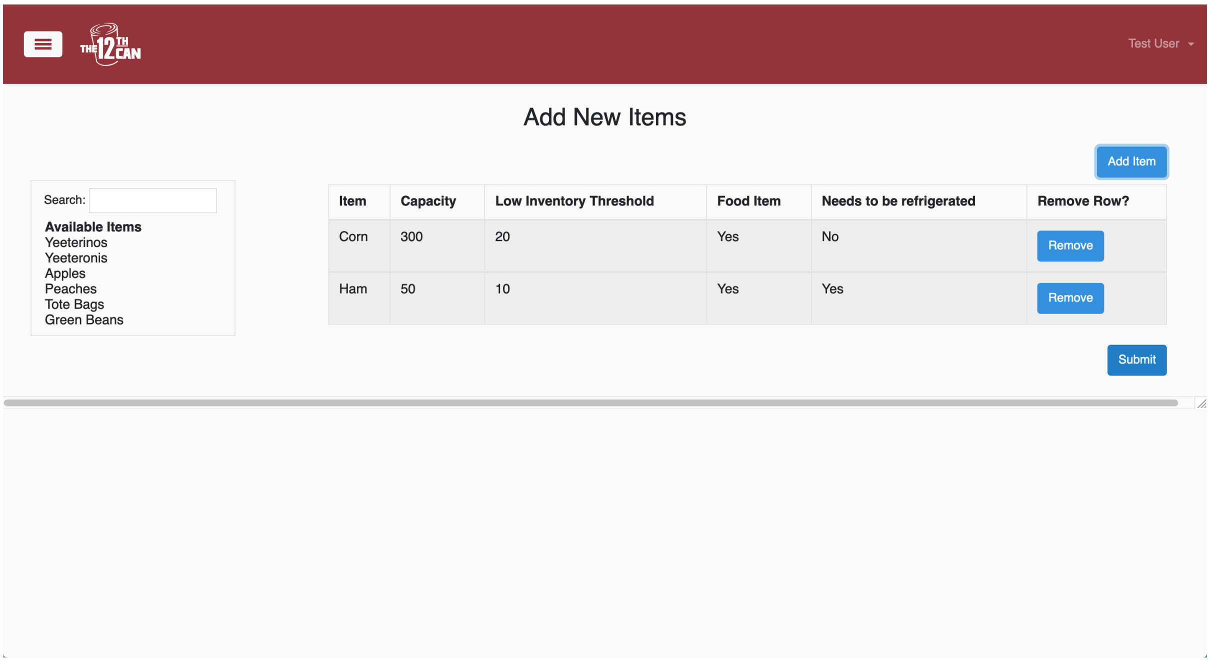
Items in Inventory

Apples
Carrots
Cream Corn
Corn
Diapers
Green Beans
Totes

Items To Be Added

Item	Capacity	Low Inventory Threshold	Food Item	Needs to Be Refrigerated	
Apples	1200	60	☑	☑	Cancel
Green Beans	600	28	☑	☑	Cancel

Figure 5: Add New Items Page Mockup



Add New Items

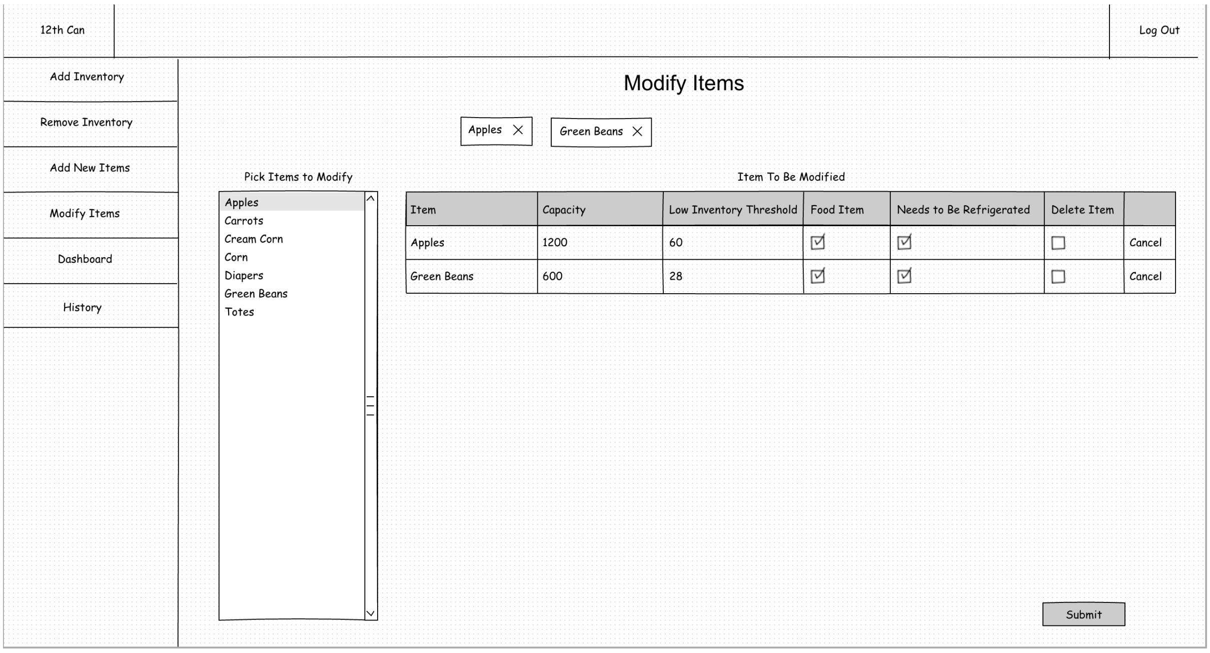
Search:

Available Items
Yeeterinos
Yeeteronis
Apples
Peaches
Tote Bags
Green Beans

Item	Capacity	Low Inventory Threshold	Food Item	Needs to be refrigerated	Remove Row?
Corn	300	20	Yes	No	Remove
Ham	50	10	Yes	Yes	Remove

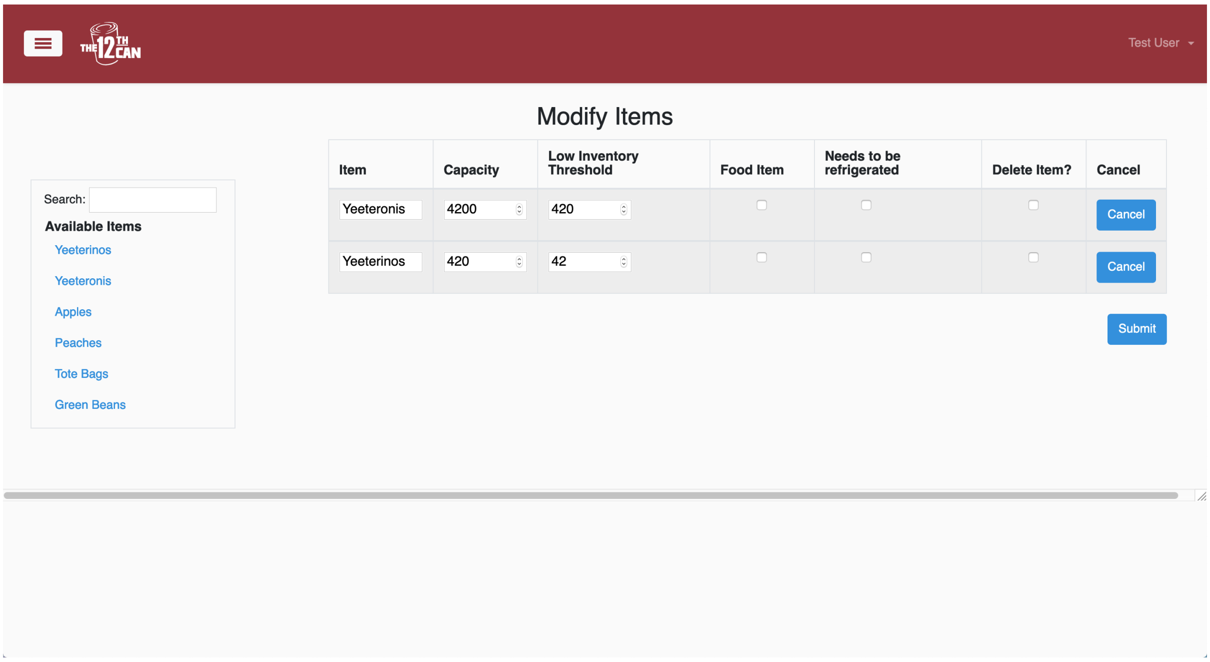
Figure 6: Final Add New Items Page

4.4 Modify Items Page



Item	Capacity	Low Inventory Threshold	Food Item	Needs to Be Refrigerated	Delete Item	
Apples	1200	60	☑	☑	□	Cancel
Green Beans	600	28	☑	☑	□	Cancel

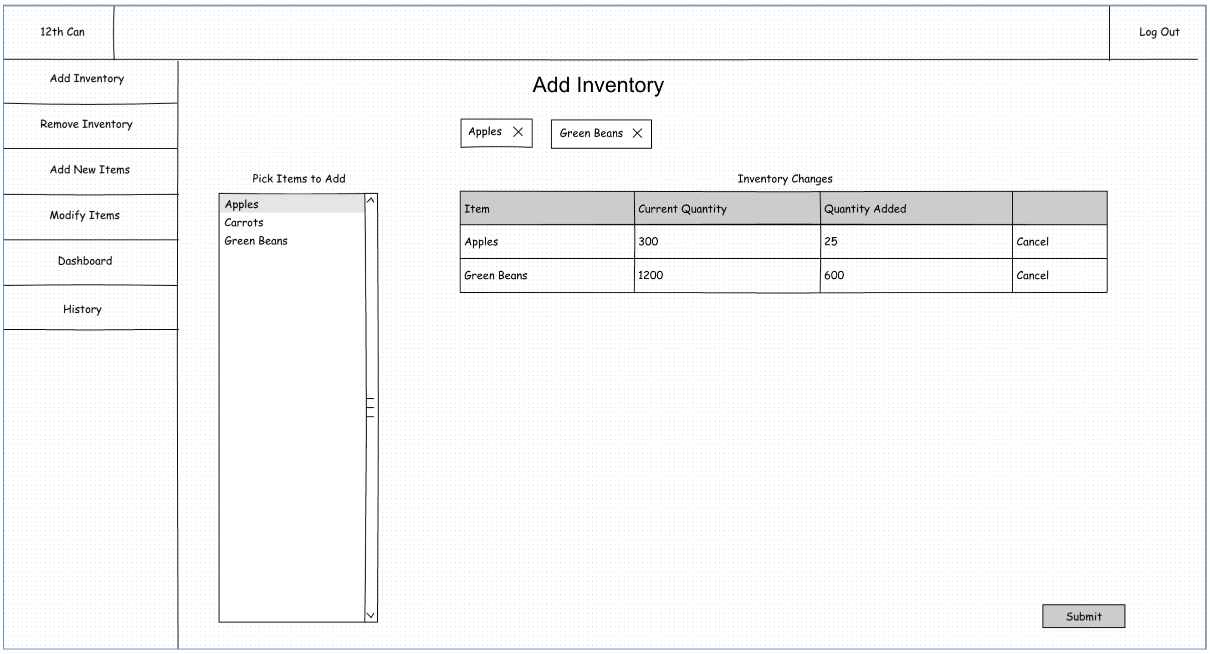
Figure 7: Modify Items Page Mockup



Item	Capacity	Low Inventory Threshold	Food Item	Needs to be refrigerated	Delete Item?	Cancel
Yeeteronis	4200	420	□	□	□	Cancel
Yeeterinos	420	42	□	□	□	Cancel

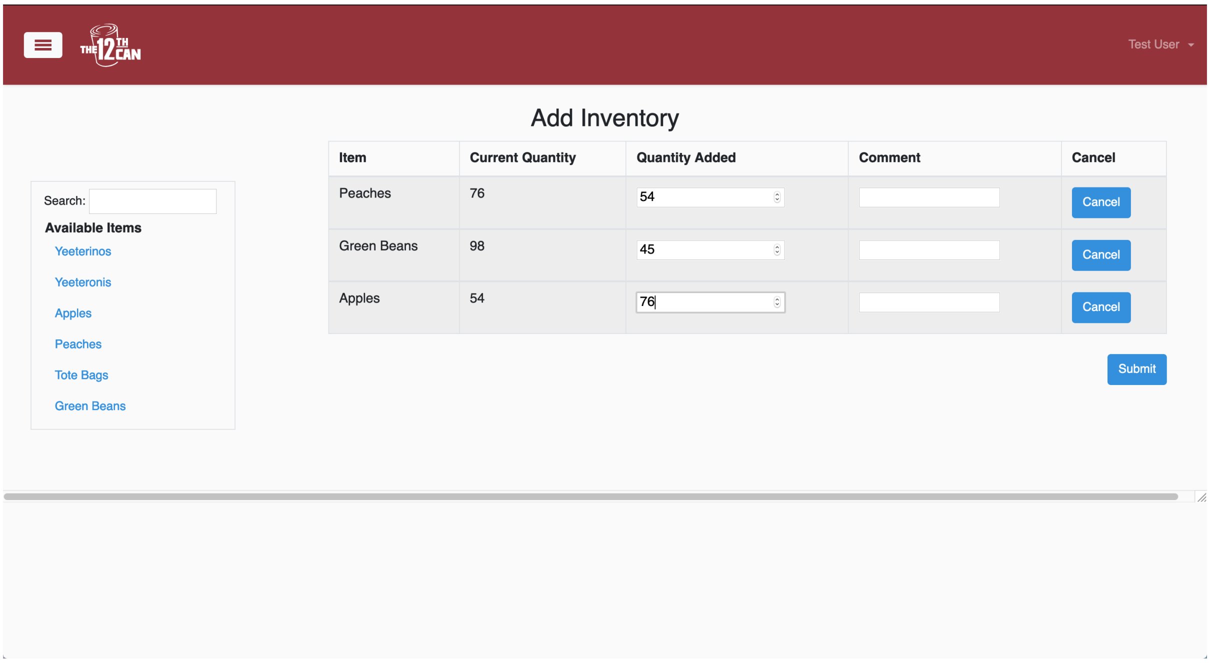
Figure 8: Final Modify Items Page

4.5 Add Inventory Page



Item	Current Quantity	Quantity Added	
Apples	300	25	Cancel
Green Beans	1200	600	Cancel

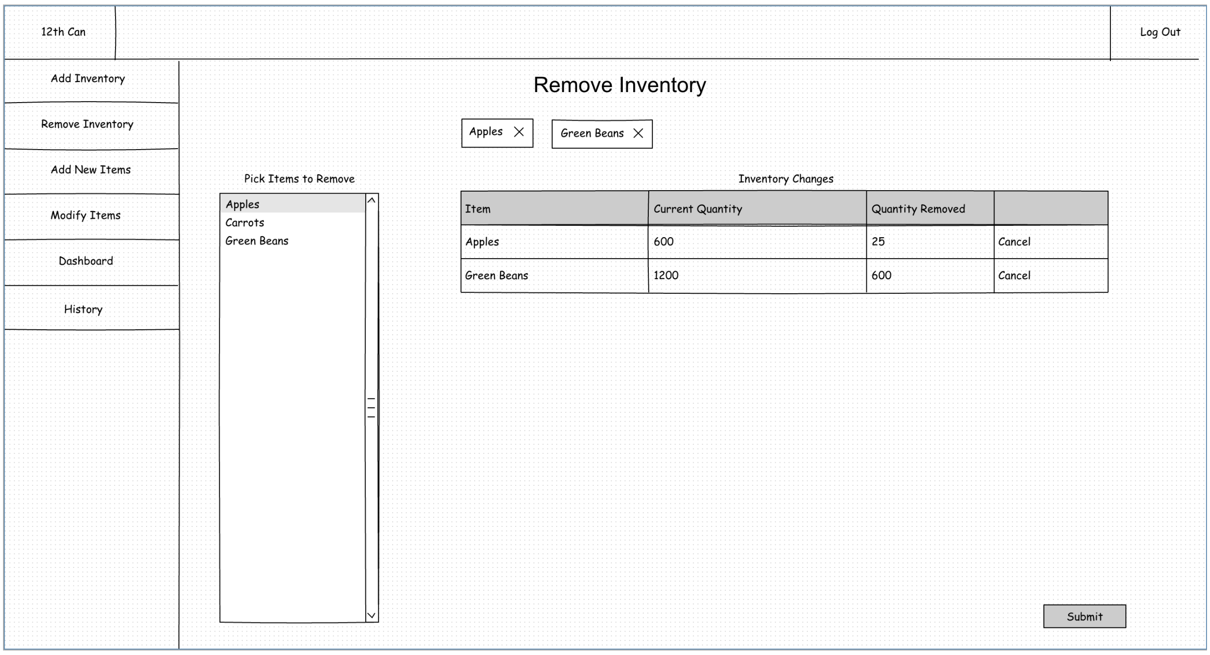
Figure 9: Add Inventory Page Mockup



Item	Current Quantity	Quantity Added	Comment	Cancel
Peaches	76	54		Cancel
Green Beans	98	45		Cancel
Apples	54	76		Cancel

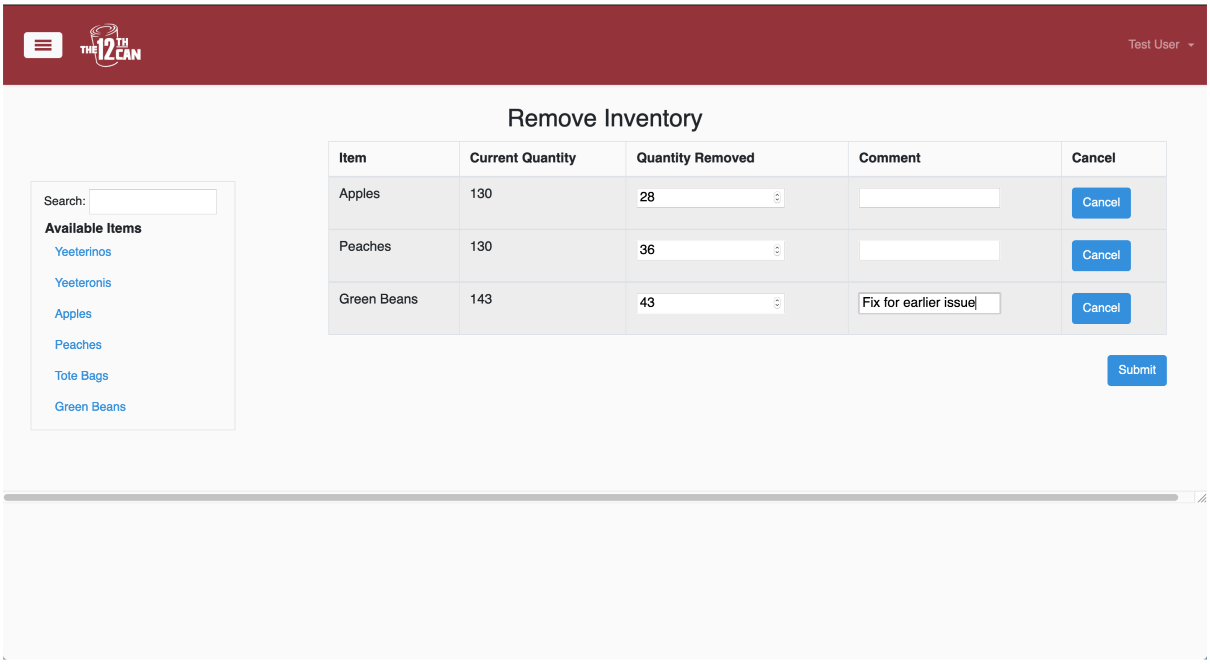
Figure 10: Final Add Inventory Page

4.6 Remove Inventory Page



Item	Current Quantity	Quantity Removed	
Apples	600	25	Cancel
Green Beans	1200	600	Cancel

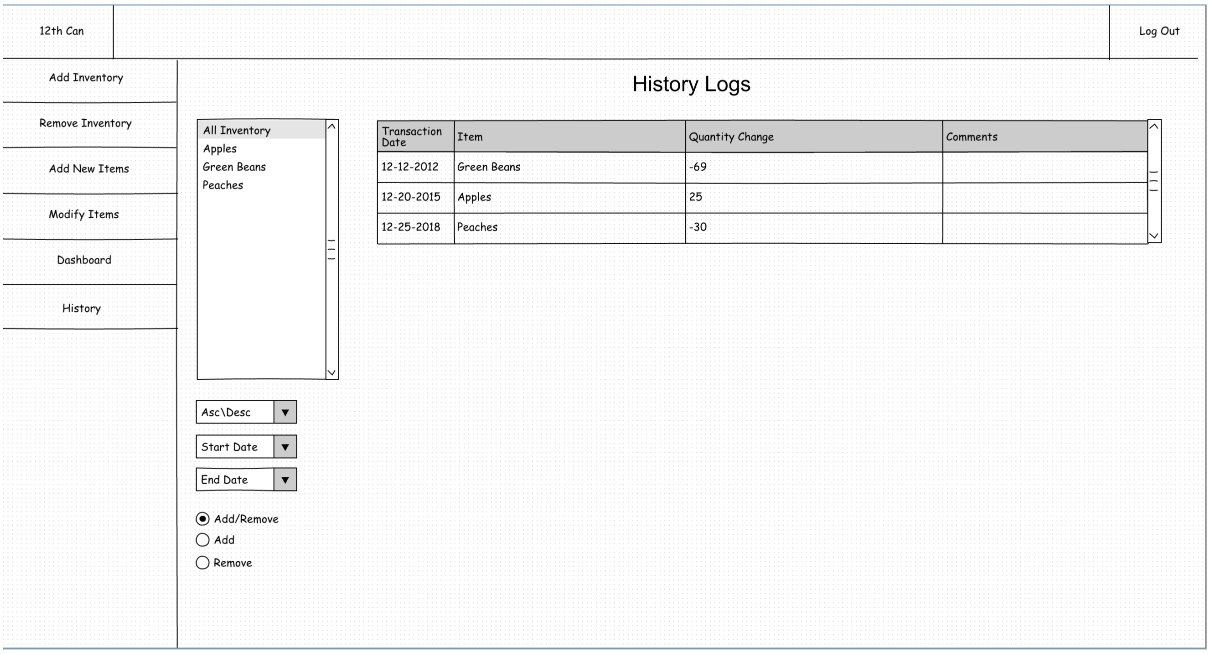
Figure 11: Remove Inventory Page Mockup



Item	Current Quantity	Quantity Removed	Comment	Cancel
Apples	130	28		Cancel
Peaches	130	36		Cancel
Green Beans	143	43	Fix for earlier issue	Cancel

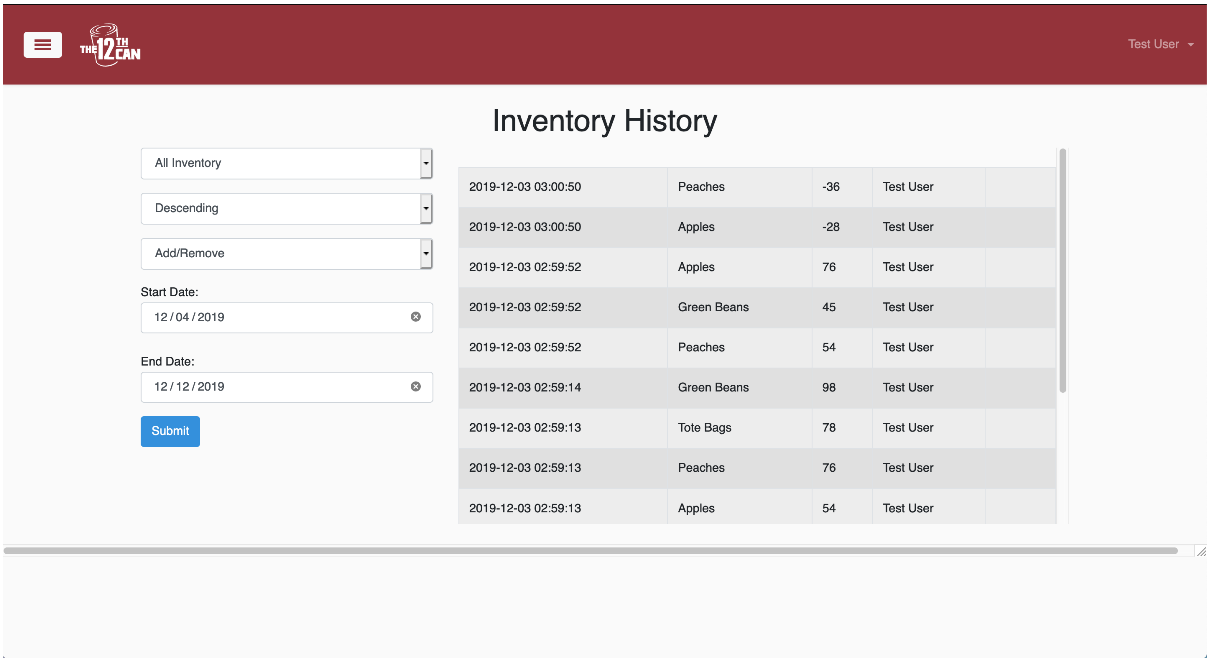
Figure 12: Final Remove Inventory Page

4.7 Transaction History Page



Transaction Date	Item	Quantity Change	Comments
12-12-2012	Green Beans	-69	
12-20-2015	Apples	25	
12-25-2018	Peaches	-30	

Figure 13: Transaction History Page Mockup



All Inventory

Descending

Add/Remove

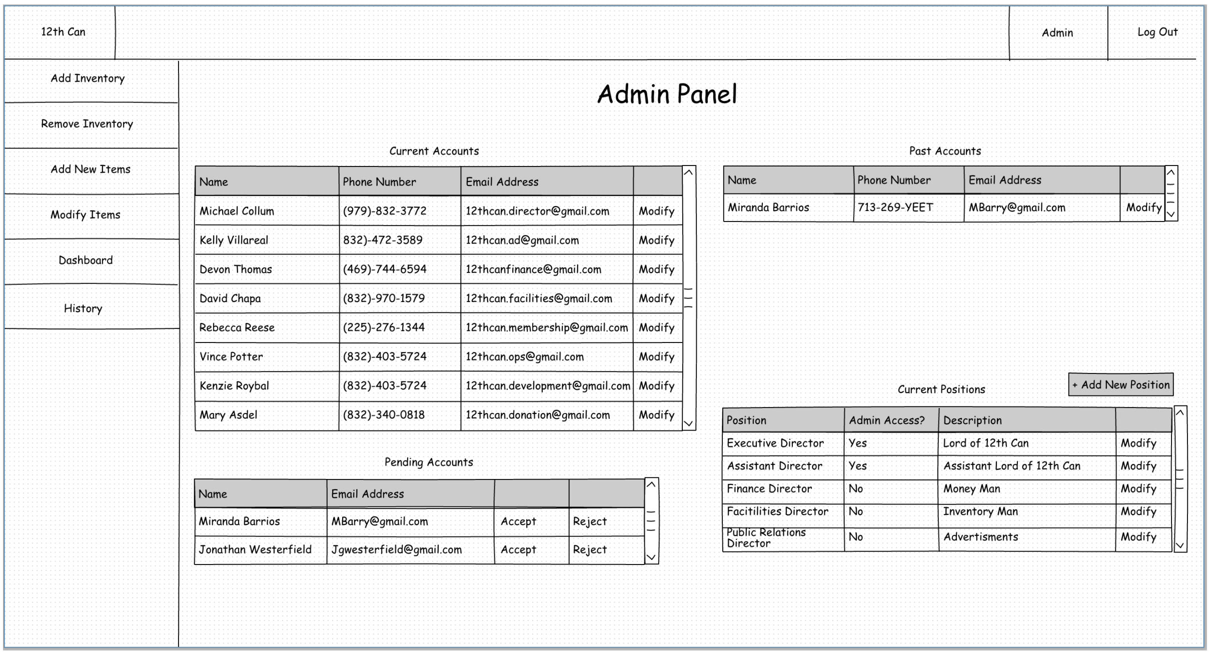
Start Date: 12 / 04 / 2019

End Date: 12 / 12 / 2019

2019-12-03 03:00:50	Peaches	-36	Test User
2019-12-03 03:00:50	Apples	-28	Test User
2019-12-03 02:59:52	Apples	76	Test User
2019-12-03 02:59:52	Green Beans	45	Test User
2019-12-03 02:59:52	Peaches	54	Test User
2019-12-03 02:59:14	Green Beans	98	Test User
2019-12-03 02:59:13	Tote Bags	78	Test User
2019-12-03 02:59:13	Peaches	76	Test User
2019-12-03 02:59:13	Apples	54	Test User

Figure 14: Final Transaction History Page

4.8 Admin Panel



Admin Panel

Current Accounts

Name	Phone Number	Email Address	
Michael Collum	(979)-832-3772	12thcan.director@gmail.com	Modify
Kelly Villareal	832)-472-3589	12thcan.ad@gmail.com	Modify
Devon Thomas	(469)-744-6594	12thcanfinance@gmail.com	Modify
David Chapa	(832)-970-1579	12thcan.facilities@gmail.com	Modify
Rebecca Reese	(225)-276-1344	12thcan.membership@gmail.com	Modify
Vince Potter	(832)-403-5724	12thcan.ops@gmail.com	Modify
Kenzie Roybal	(832)-403-5724	12thcan.development@gmail.com	Modify
Mary Asdel	(832)-340-0818	12thcan.donation@gmail.com	Modify

Past Accounts

Name	Phone Number	Email Address	
Miranda Barrios	713-269-YEET	MBarry@gmail.com	Modify

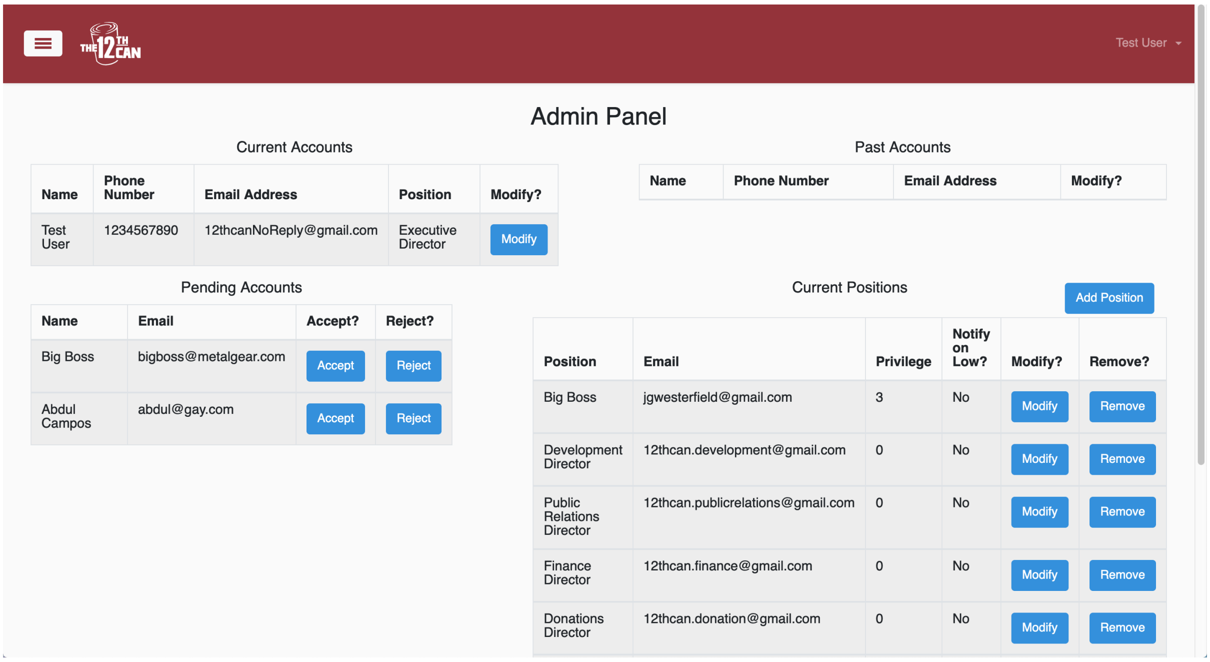
Pending Accounts

Name	Email Address		
Miranda Barrios	MBarry@gmail.com	Accept	Reject
Jonathan Westerfield	Jgwesterfield@gmail.com	Accept	Reject

Current Positions

Position	Admin Access?	Description	
Executive Director	Yes	Lord of 12th Can	Modify
Assistant Director	Yes	Assistant Lord of 12th Can	Modify
Finance Director	No	Money Man	Modify
Facitilities Director	No	Inventory Man	Modify
Public Relations Director	No	Advertisments	Modify

Figure 15: Admin Panel Mockup



Admin Panel

Current Accounts

Name	Phone Number	Email Address	Position	Modify?
Test User	1234567890	12thcanNoReply@gmail.com	Executive Director	Modify

Past Accounts

Name	Phone Number	Email Address	Modify?

Pending Accounts

Name	Email	Accept?	Reject?
Big Boss	bigboss@metalgear.com	Accept	Reject
Abdul Campos	abdul@gay.com	Accept	Reject

Current Positions

Position	Email	Privilege	Notify on Low?	Modify?	Remove?
Big Boss	jgwesterfield@gmail.com	3	No	Modify	Remove
Development Director	12thcan.development@gmail.com	0	No	Modify	Remove
Public Relations Director	12thcan.publicrelations@gmail.com	0	No	Modify	Remove
Finance Director	12thcan.finance@gmail.com	0	No	Modify	Remove
Donations Director	12thcan.donation@gmail.com	0	No	Modify	Remove

Figure 16: Final Admin Panel

5 Team Roles

Scrum Master: Jonathan Westerfield
Product Owner: Jonathan Westerfield
Front End: Mannan Mendiratta, Daniel Patlovany, Ismael Rodriguez
Back End: Jonathan Westerfield, Abdul Campos, Aaron Todd

Everyone contributed to the code base. Product Owner and Scrum Master positions were not rotated.

6 Scrum Iterations and User Stories

6.1 Iteration 0

- Set up a meeting with the customer to take their interview video
- Negotiated customer requirements
- Got user stories needed for the project
- Setup Github repo and Pivotal Tracker for the project
- Submitted Iteration 0 Report

6.2 Iteration 1

We setup PHPUnit so that we could write our test and also use it to generate code coverage reports. We also submitted the Iteration 1 Report.

Implemented the following stories:

Feature: **Log In**

As an inventory manager I want a page that will authenticate allowed users so they can modify the inventory

Feature: **Add New Item Page**

As an inventory manager So that I can keep up with new products I want to add new items to the database

Feature: **Inventory Dashboard**

As an inventory manager I want a way to easily view the inventory in the system so that I don't have to click through a whole bunch of stuff to see our inventory.

Feature: **Navigation Sidebar**

As a user I want a way to easily navigate the pages So I don't have to do extra steps to navigate the website.

6.3 Iteration 2

Submitted the Iteration 2 Report and also implemented the following user stories:

Feature: **Add Inventory**

As an inventory manager
I want a way to easily and accurately add inventory from the system
so that we can easily keep up with changes in our inventory.

Feature: **Remove Inventory**

As an inventory manager
I want a way to easily and accurately remove inventory from the system
so that we can easily keep up with changes in our inventory.

6.4 Iteration 3

While we had already implemented the Inventory Dashboard, we determined that the quality was not high enough and had to be redone. We also submitted the Iteration 3 Report.

Feature: **Inventory Dashboard**

As a customer
I want a way to easily view the inventory in the system
So that I don't have to click through a whole bunch of stuff to see our inventory.

6.5 Iteration 4

Submitted the Iteration 4 Report and also implemented the following user story:

Feature: **Admin Panel**

As a site admin
I want a way to have control of user account info
So that I can add, remove, and modify user information.

7 Customer Meetings

7.1 Iteration 0

Sept 5, 2019 @6pm in the Zachry Building

- Discussed the current workflow for tracking inventory
- Determined the scope of the project and the user stories needed
- Discussed and approved the mockups of the site

7.2 Iteration 1

October 17, 2019 @6pm in the Zachry Building

- Discussed revised mockups
- Demoed user stories implemented up to this point

7.3 Iteration 2

October 31, 2019 @6pm in the Zachry Building

- Showed the user more of the features
- Reconfirmed user stories and verified we are making what they want

7.4 Iteration 3

November 14, 2019 @6pm in the Zachry Building

- Displayed progress on user stories up to this point
- Customer was very please since we essentially had a working product by this point
- Discussed the handoff of the project to the IT department

7.5 Iteration 4

December 1, 2019 @6pm in the Zachry Building

- Displayed full working product
- Customer was very pleased at the polish and ease of the website
- Customer was also pleased they had so much control over the website
- Discussed the handoff of the project to the IT department more in depth since more info was available

8 Testing (BDD/TDD)

Our BDD/TDD process was haphazard at first. While we were following behavior driven development the entire time, we weren't always following test driven development. At the beginning, the only tests we had were unit tests for the backend functionality and these were only written *after* the function had been implemented. Later on, we were able to get browser tests working so we could do feature tests. However, these tests were still implemented *after* the features had been implemented. It wasn't until the end, when we had grasped how to use Dusk (the browser test framework) that we started to write our browser tests *before* implementing the features. However, our unit tests were run using PHPUnit and these tests, along with the browser tests, were always run to verify a code change actually worked. PHPUnit and Dusk were used in place of Cucumber and Rspec due to our project being a PHP/Laravel project.

9 Configuration Management

9.1 Version Control

Since we are implementing a system that is not only large, but also going to be put into production for our customers to interact with, it is imperative that we use a version control system to host our code base. We had several options to choose from with the most notable being Github, Bitbucket, Gitlab. Of these, we decided to choose Github due to familiarity from the rest of our team, the ability to host a static website from the repo, and having built in bug tracking features.

We decided to use the Gitflow branching stategy in order to split up work amongst the team. The gist of the Gitflow strategy is that there are two distinct branches: master and develop (dev for our team). The master branch is maintained solely as a major release branch that is only changed on major releases and during hotfixes. The dev branch is for integrating and testing features. For each release, the dev branch would be merged into master. This allows for the branching history to look clean and crisp on master while still having the freedom to make a mess on dev.

To make changes to dev and develop new features, new branches are needed. The Gitflow stategy entails that each new feature that the project needs will get its own branch. We named these new feature branches after the feature they were created to implement. Once the feature was completed and tested, that feature branch would then be merged into dev. This offered many advantages. First Git allows us to see the user that pushed changes by default so creating a branch for each developer was not necessary. Second, if a branch is named after the feature to be implemented, it removes all ambiguity of the work taking place in that branch. Finally, these feature branches are very helpful when reverting changes that break the code base. For example, once the Add Inventory feature was implemented and merged into dev, it suddenly and inexplicably broke the login capabilities of the application. Luckily, we were able to fix this by simply remerging the Login branch into dev again, overwriting the broken code from the Add Inventory branch. This allows us to keep certain snapshots of the project that are known to work in case of failure or loss in the dev branch.

More info on the Gitflow branching strategy can be found at `https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow`.

This strategy eventually led us to have 5 releases, one for each iteration and one for the final website. It also led to the creation of 19 branches, one for each feature implemented (which is not the same as the user stories implemented).

9.2 Database

We only had one version of the database throughout the project. We used a SQLite database that was manipulated purely through the database seeds and database actions from the website. Even if there were breaking changes made to the database, it was simple to easily delete, migrate, and reseed the database.

10 Issues in the Production Release Process to Heroku

In the end, we were able to successfully deploy to Heroku, but not without some issues. First, figuring out how to push to heroku was initially difficult due to our team implementing our site in PHP/Laravel. Secondly, we discovered a bug in our database when deployed to Heroku. Heroku has a bug where it will delete and reseed the database after any period of unuse if the database is a SQLite database. However, this will not be an issue in production as the Department of IT utilizes a MySQL database for their production environments.

11 Implementation Environment

The entire project was developed in a *nix environment. We had 3 members with MacOS computer and the Windows users ran their dev environments in a Ubuntu VM or Linux Subsystem. The MacOS Unix environment was used to demo to customers.

12 Tools Used

- PHP 7.3
- Laravel 6.6.0
- PHPUnit 8.4.3
- Laravel Dusk 5.5.0
- Chrome Driver 72 (for Dusk testing)
- SQLite3
- Mailgun
- Heroku
- Github

PHPUnit was used for both unit testing and also for generating code coverage reports.

13 Important Links

13.1 Github Repo

https://github.com/JonathanGWesterfield/12th-Can

13.2 First Customer Interview

https://youtu.be/VeL_BOOuVnE

13.3 Pivotal Tracker

`https://www.pivotaltracker.com/n/projects/2397334`

13.4 Heroku Site

`https://limitless-tor-35321.herokuapp.com`

Figure 17: QR Code for the Heroku site.