Project Title: Kitchen Pantry Application (Pantry Panda)

Prepared by: Matt Holliday, Christian Averill, Ife Akindipe, John Vandermyde, Laura Bosakaite

Table of contents:

Introduction (Project description)

The goal of this course was to create an application which has a user login, user profile, home button, database, and is accessible through ISUNET. The web application we created was a Kitchen Pantry Application, called Pantry Panda.

This Kitchen Pantry is broken into dry goods, fridge, freezer, and spice rack to optimize the organization of the user's pantry. While adding items to the pantry, users can scan the item's barcode to make the process a bit quicker; the scanner autofill's the product name and the barcode. The user will have to adjust the quantity they will be adding, the expiration date, and which section of the pantry they would like it in. The application is also able to generate a grocery list for the user based on the threshold they set when adding the item. Whenever a pantry item falls below a set quantity, the item is added to the grocery list which the user can print as a pdf.

Project timeline(Gantt chart and planning)

As a team, we had a brainstorming session, and we worked on a Gantt chart to help start the planning process. In the brainstorming session, we determined what frameworks we would use to build the application. Once we submitted the proposal, we got to work right away without really discussing what we wanted to see out of the frontend or backend. We all trusted each other's judgements and worked together to build the front end and back end at the same time. Throughout the process, we would communicate with the customer to determine what he expected from the website and adjust accordingly. We would discuss ideas as a team to ensure there would be no issues encountered (hopefully). We built the site as if the user were using the site, so we started with the user in the back end and the login and sign-up page in the front end. We then continued to the pantry pages, which required the backend to be able to store the data for each item. We faced issues as a team, and sometimes other members would have to come along and help fix the issue that we were encountering.

One of the faults we encountered as a team was not planning enough. We created a general concept of what we wanted to create; we listed general tasks which would help guide us through our process. We did not hash out what we wanted to see out of our application, how we wanted to organize the user interface, or how we should have organized the code for our application.

Kitchen Pantry

Project lead

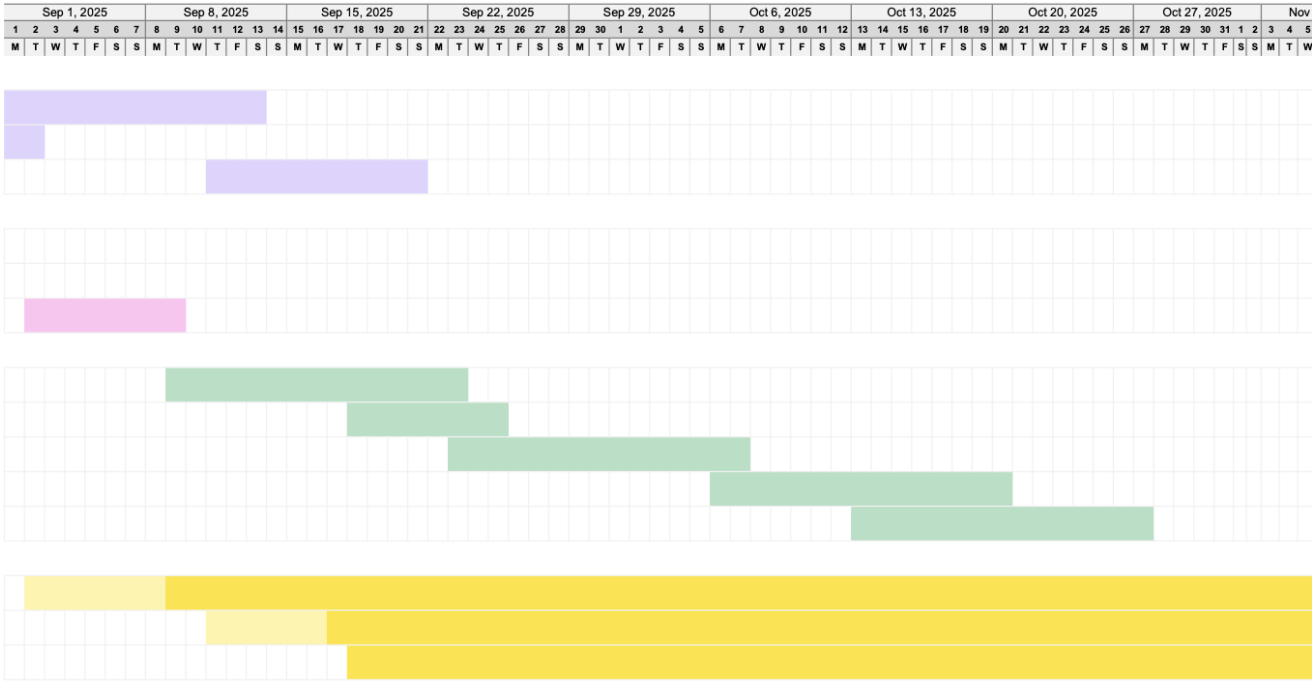
Project start: **Thu, 9/11/2025**

Display week: **0**

SIMPLE GANTT CHART by Vertex42.com
https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html

TASK	ASSIGNED TO	PROGRESS	START	END
Initiation				
Establish comms	Team	100%	8/28/25	9/13/25
Define goals	Team	100%	8/30/25	9/2/25
Generate VMs	Dean	100%	9/11/25	9/21/25
Planning and design				
Create schedule	Laura	100%	9/12/25	9/9/25
Identify deliverable	Ife	100%	9/14/25	9/9/25
Define scope	Matt	100%	9/2/25	9/9/25
Execution				
Build Database	Christian	0%	9/9/25	9/23/25
API Connections	Matt	5%	9/18/25	9/25/25
Frontend Developn	Ife	0%	9/23/25	10/7/25
Backend Developn	John	0%	10/6/25	10/20/25
Testing and validat	Laura	0%	10/13/25	10/27/25
Evaluation				
Connect with custo	Christian	10%	9/2/25	11/13/25
Evaluate progress	Matt	10%	9/11/25	11/13/25
Gather feedback	Laura	0%	9/18/25	12/2/25

Insert new rows ABOVE this one



Frontend Development

The frontend was built using the React framework, which is an open-source frontend development JavaScript library. We chose this framework because of its component-based architecture which allowed for reusable UI components; it used a declarative approach to designing the user interface which allowed for the development of the frontend to be more intuitive. React has a vast source of libraries and tools which are easy to incorporate into the application. React allows for easy scalability and provides flexibility for connecting to the backend; and is relatively easy to learn.

Before we could begin creating the various webpages for the site, we had to determine what pages would be necessary to effectively use the application. We determined we needed a minimum of 7 pages:

1. A home page that would contain a welcome banner, and a section describing the application's features.
2. A login page to allow users to access their pantry
3. A sign-up page to create their account
4. A user home page
5. An inventory page
6. A grocery list page
7. User settings/ security page

After deciding what was needed, we began by creating various blank pages; we decided to add two background images to use for the site, one for when a user is logged in vs a guest. The first major goal was to create a working login page and sign-up form which would connect to the backend and successfully add the user to the database. This was a bit difficult, but with some collaboration with the backend, we were able to determine the correct fetch URL which would successfully create a user. When a user goes to sign up, they are asked to provide their username, email, password, and to confirm their password. The login page was initially set up to only need the user's email and password to sign in. We modified the code of the login page to require an identifier which was the user's username OR email. This followed how many other sites create user accounts, and we wanted to align our sites' UI/UX with what others are doing. The sign-up and login pages require that all texts field be filled out, and the sign-up page checks that the email is valid and the user's password is of certain strength (8 characters, include uppercase and lowercase letters, a number, and a special character). Once the user successfully logs in, they are redirected to their pantry.

The pantry page (user home page) displays the compartments of the user's pantry and an add Item button. The compartments are set up as a grid on the pantry page, and the user selects the button to see the items in that compartment. We then focused our efforts on getting a working add item form; where we asked the user to provide the product's name, barcode, quantity, the expiration date, and where the user wanted to store this product. Initially, we faced issues while trying to implement the barcode scanner in the add item form. We created a barcode scanner form which connected with the barcode scanner API (open food facts), where we fixed any camera issues and used the API to autofill the barcode for the user. The freezer, fridge, spice rack, and dry goods buttons display the pantry items for the user; these are all designed to display the items in a clear table. The user can delete items from their pantry directly from the list, or if they wanted to add items while in that section of their pantry, they could. When the user sets the quantity that they want to have on hand of a product, the grocery list now checks the user's pantry to see if there are items below that threshold. If the grocery list logic finds any items below that threshold, they are then added to the grocery list page for the user. The grocery list is displayed as a table with a white background, where it displays the product name, and the amount needed to buy to no longer be below that threshold. The user settings page displays the username once changed, and allows them to update their username, email, and password; Users can also delete their accounts. We had then planned on creating an inbox and allowing users to choose their notification preferences. We have the buttons and features, but they are not functioning currently.

The navigation bar at the top of the site allows users to easily navigate between the different pages while logged in and then log out when they are all done. When logged in, the user sees home, pantry, grocery list, user settings, inbox, and logout. When visiting as a guest, the user will only see home, log in, and sign up in the navigation bar. This was meant to make the site accessible and keep the look of the site cohesive.

Backend Development

We used NestJS as the backend framework for our project. NestJS is essentially NextJS but for TypeScript. TypeScript was chosen over JavaScript because of the typing as well as the fact that it’s a compiled language. This helps prevent any typing and runtime errors. NestJS has a default structure to create for each table in the database. It consists of a controller, module, service, entity, and DTOs. The controller controls the routing and is what the frontend actually calls to communicate to the backend and through backend to database. By routing, I mean routing the request to query the database. The service contains the actual logic of querying the database and returning a response to the frontend to render. Querying was made simple by using TypeORM, which allows us to query the

database without raw SQL queries. The entity just defines the entity based on the table in the database. DTOs are used primarily for creating and updating. They provide a way to pass a group of data to the backend rather than passing each individual parameter needed. The module helps resolve relationships between the different tables. This default structure that NestJS defines allowed us to create a structured way of mapping the database into our project. Each table that exists in the database had this structure.

Some pitfalls of this structure are that sometimes for simple tables/entities, the abstraction that is used is overkill. In other words, it may have been simple enough to not make all these files and folders for each entity. However, the reason is that the structure is used for scalability. While it might seem overkill now, the structure allows to scale with new additions or features easily.

Database Development

When it came to choosing the database that we wanted to use to support the operation that we had built up we decided on MySQL, given the ease of working with it as well the fast setup process on Linux so that we could hit the ground running as soon as possible. When it came to actually designing the database we had to look at the goals we had set out for our app such as having a user base, the ability to store products as well as make it so that users have an individual inventory and lastly make a grocery list. The hardest part of the development was actually trying to map out what was needed and how everything relates based on the relationships of a database. Once it was off the ground and running it worked as intended with the only modification needing to be done was a minor tweak to the necessity of each item needing a UPC given the fact that something like an apple doesn't have a barcode on it.

System Architecture and Data Flow

When a user requests some data whether they want to add items to pantry, retrieve the products added, and retrieve the grocery list, the frontend uses the fetch keyword with a URL using reverse proxy to route the backend to the database. When the frontend uses fetch with the proper routing defined by the controller, it sends all data necessary to the backend. The reverse proxy is set up in the server config. This helps simplify the process since it places the routing responsibility to the server. The server then finds the proper routing to the backend. When it finds the correct function in the controller, the controller calls the corresponding service function, which contains the actual logic to query the database to retrieve or insert data. The service function then returns these results so that the frontend can render the response for the user.

API's

When selecting the APIs that we were going to use we had set out to find two, one for recipes and one for barcode to product mapping however after project refinement we settled on just working on the grocery list so we settled on the Nutritionix API given that Christian had prior experience with it as well as a free tier all was settled. Fast forward a couple of months he got an email saying that the free tier was going to be revoked in the future so we hoped that it would stick around long enough for the project. Fast forward again to Friday before break and Christian got an email saying our token and key was revoked because we were on the free tier so we had to pivot to the OpenFoodFacts API which served the same functionality but had some minor changes we had to make to the code.

Testing

Initially testing was done on local machines, with each team member testing their changes individually before pushing to the main branch. However, when an independent server was set up to host the website, only a production branch was made. This combined with some internal changes meant that the database would only talk to the application deployed on the server and would not function on a local machine. While some front-end changes could still be tested by running the client on a local machine, anything involving the database would not work. Testing was done by team members pushing their changes to the main branch and then asking one of the backend developers to deploy the latest changes, as they were more familiar with the process. This obviously introduced some problems to our workflow. In the best circumstances, it was very inconvenient to have to ask another team member to deploy some changes to test them, as they had to take a minute to recompile and deploy the program. It wasn't uncommon to have to redeploy some changes several times, as the first attempt would sometimes have a coding error and need to be corrected. A far better approach would have been to have a testing branch for developers to test their changes and then teach each developer how to use it. Once those changes were stable, they could then be pushed to the production branch.

Basic User Guide

System requirements:

Web Browser: Latest version of Chrome, Firefox, Safari, or Edge (with JavaScript and cookies enabled)

Operating System: Windows 10/11, macOS 10.15+ (Catalina or later), or modern Linux distributions

Internet Connection: Required for accessing the web app and backend services, specifically ISUNET

Camera: For barcode scanning (if using the barcode scanner)

Printer: For printing or exporting the grocery list as a PDF

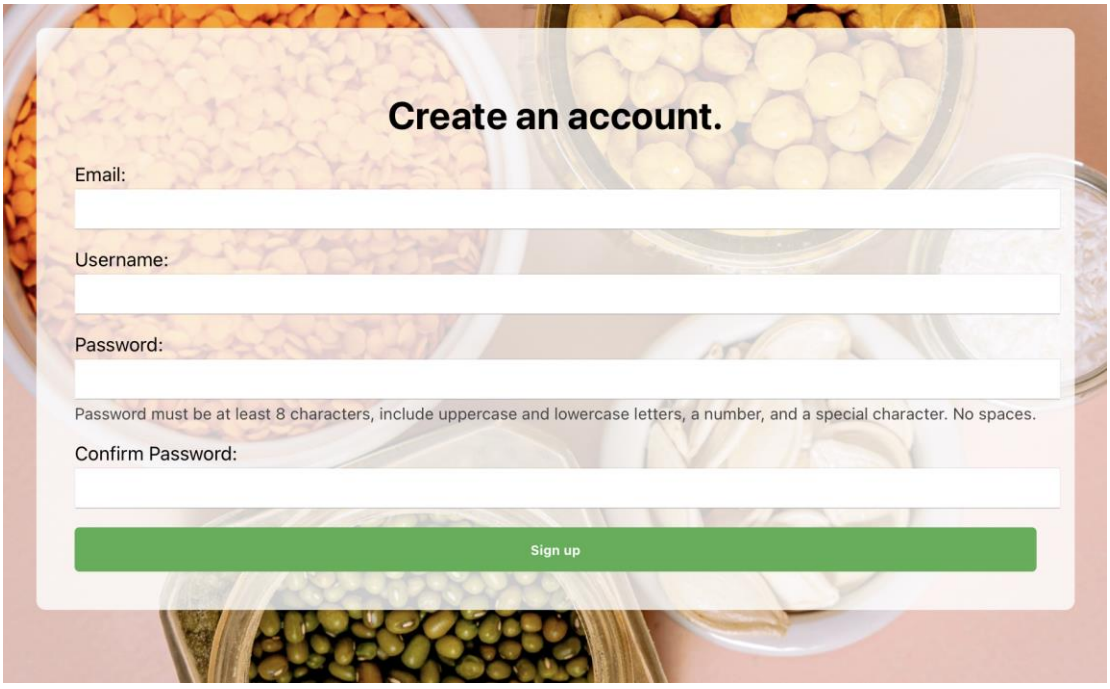
Getting Started:

How to access the website?

Access the website by going to the URL: https://10.111.16.231/ while connected to ISUNET

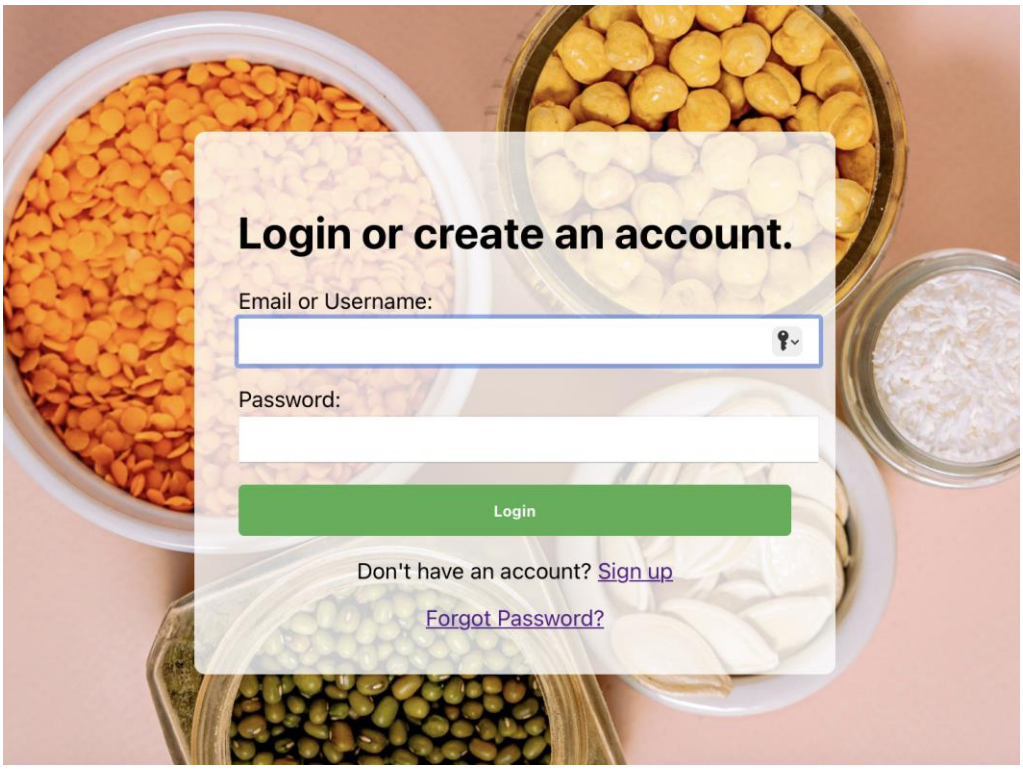
How to create an account?

Select Sign up from the navigation bar, which redirects the user to the sign-up form. The user will provide their email, username, password, and confirm that password. After successfully completing the form, the user will be redirected to the login page.



How to log in?

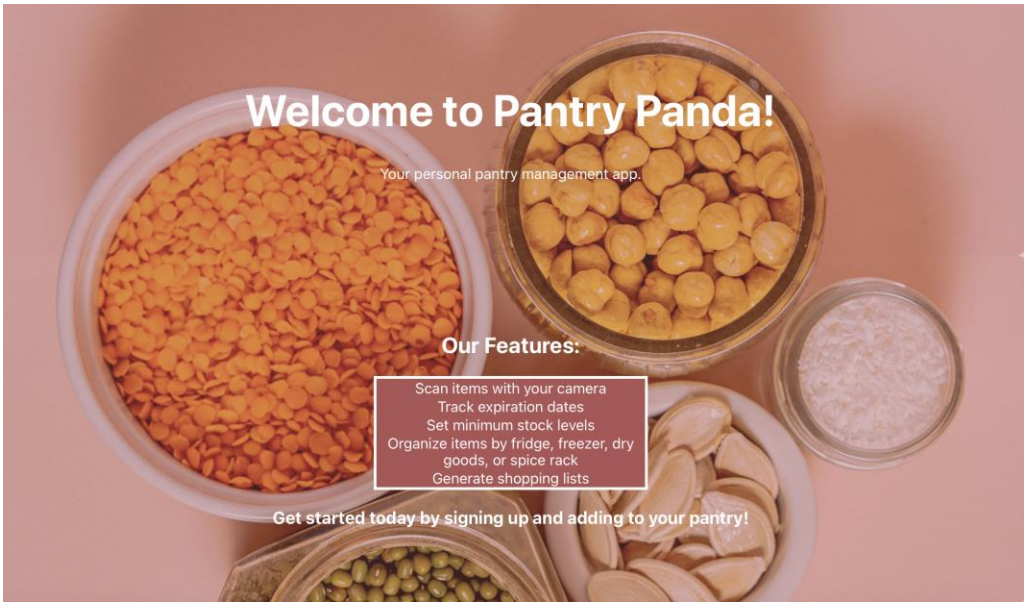
Select Login from the navigation bar, or after signing up you should be redirected here. The user will provide their username or email and password. After successfully logging in, the user will be redirected to their pantry page.



Main features:

Home:

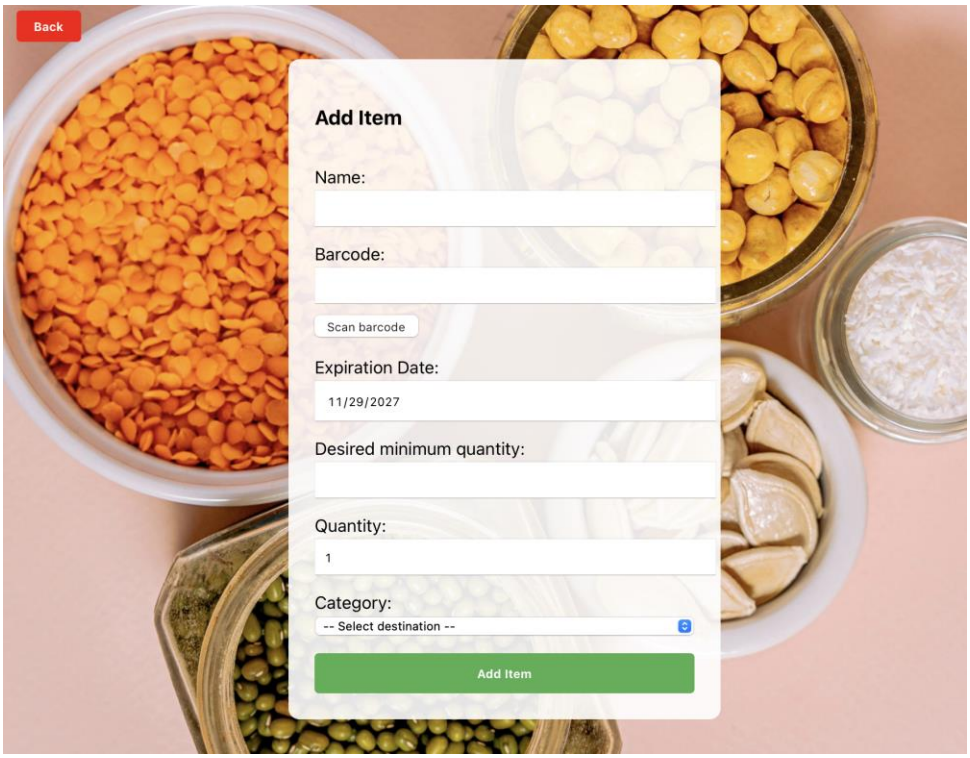
When accessing the site, the default landing page is the home page. This simply displays the main features.



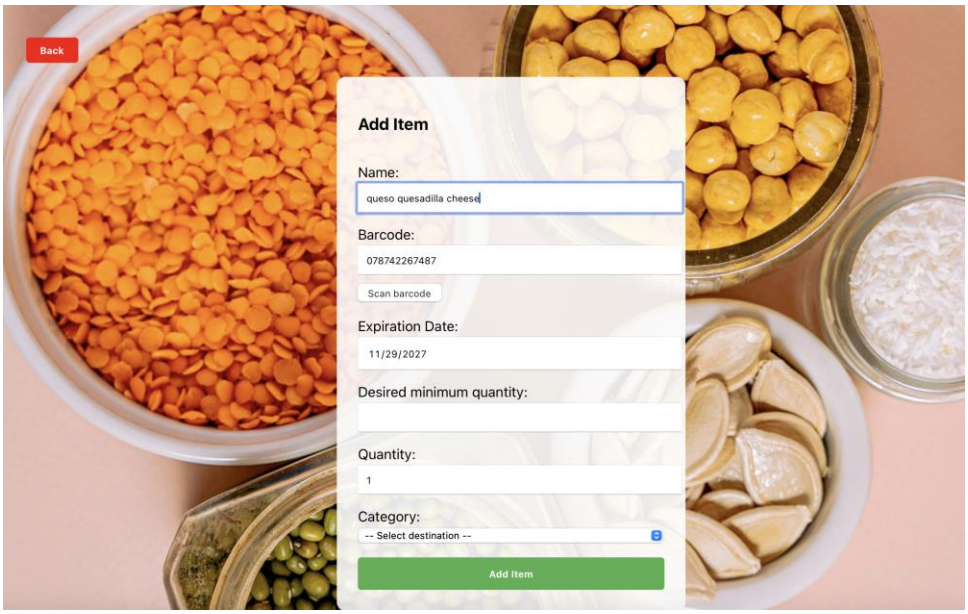
Pantry Management:

Adding an item to the pantry:

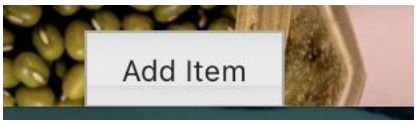
Users will land on the pantry page after logging in, if they want to add items to their pantry then they will need to select the add item button which redirects the user to the add item form.



The user can then select the scan barcode, and the camera will turn on; the scan will take the barcode of the item and autofill the barcode of the product. The user will need to modify the products name, quantity, and expiration date. If there is no expiration date, then the form calculates a default date(two years from entry date).

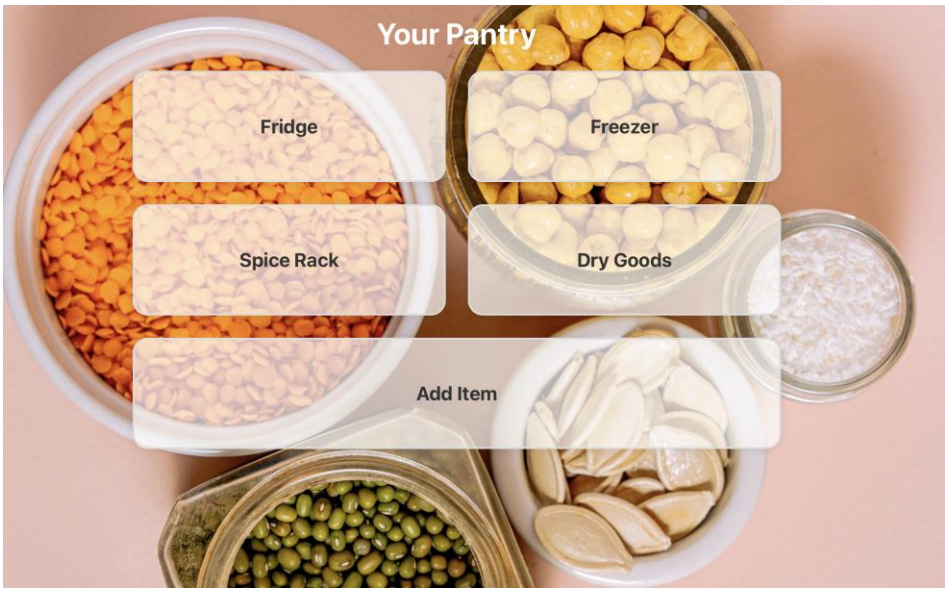


Users can also decide to add items while visiting the various pantry sections by selecting the add item button.

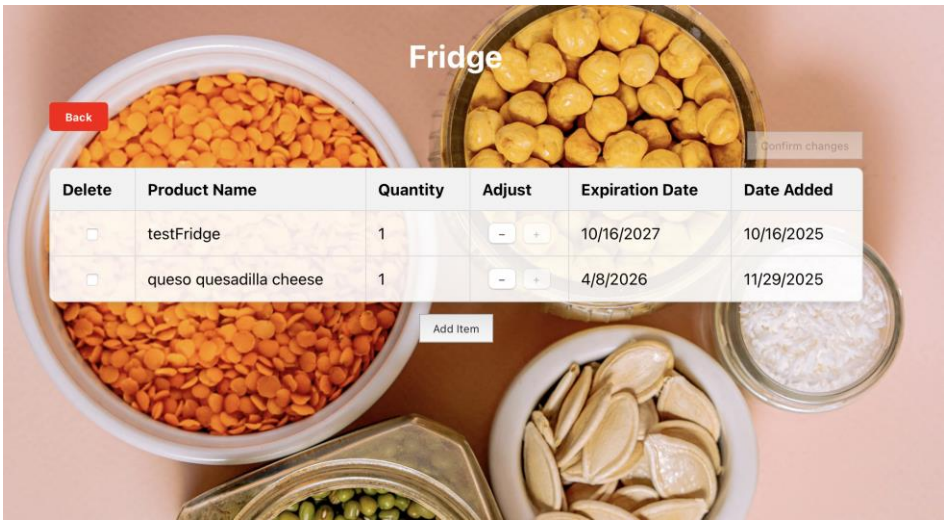


Viewing pantry by storage location:

Users can view their pantry by selecting which section they want to view. Users will select either fridge, freezer, spice rack, or dry goods.

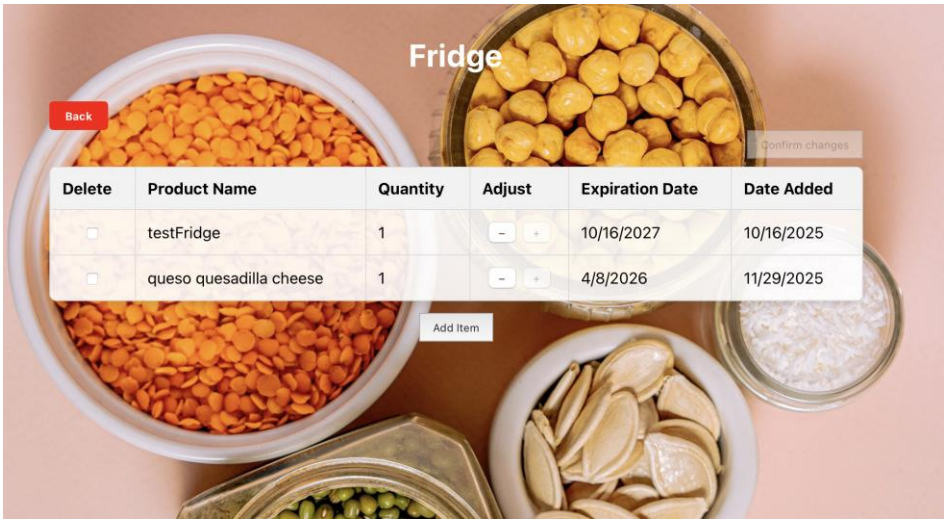


The items in each section are displayed as a table, where the user can see the quantity and expiration date.



Removing items from pantry or marking items as used:

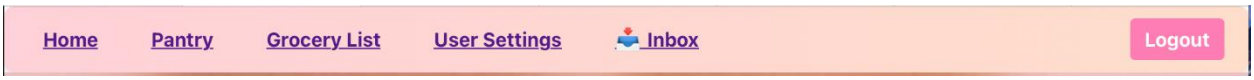
The user can also choose to remove items from their pantry directly by choosing the checkbox in delete column or selecting the minus sign in the adjust column. The choice is determined by how much the user has used, if its all gone then select delete. If marking just one unit as used then use the adjust feature.



Grocery list:

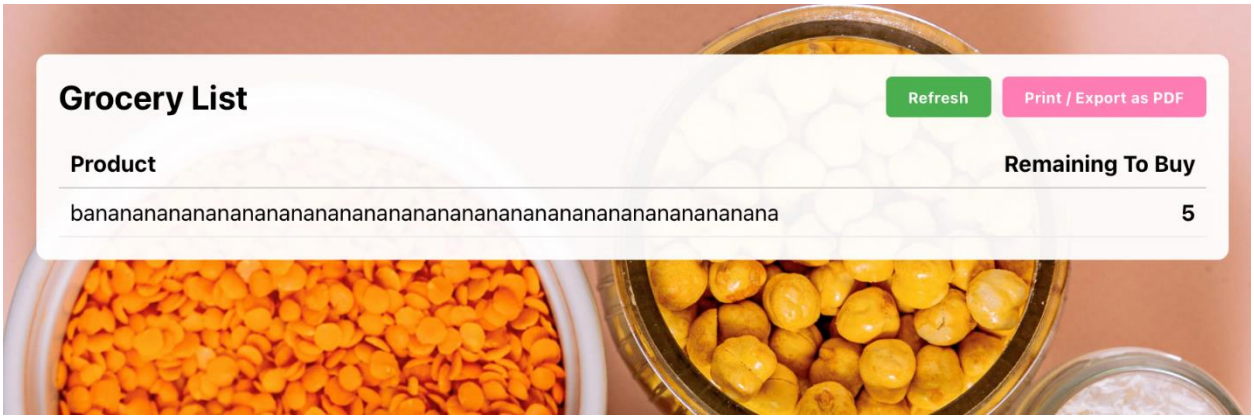
How to view grocery list:

To view the users generated grocery list, the user will select grocery list from the navigation bar.

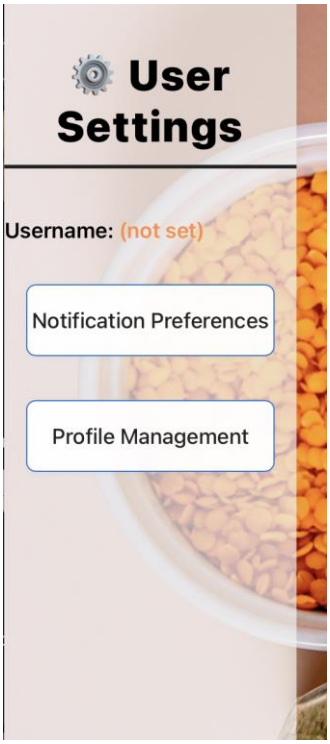


How to print grocery list:

To print the grocery list, the user will select the print/export button which will print the entire page.

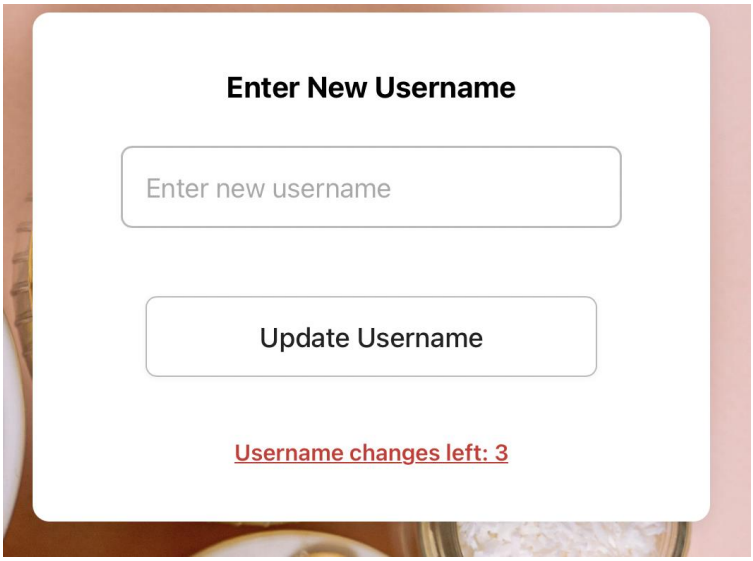


User settings:



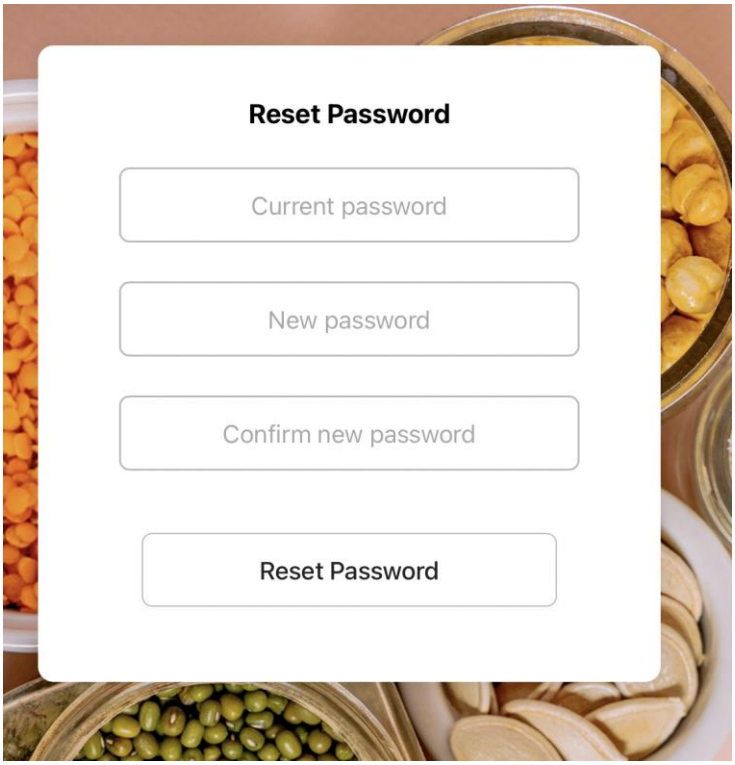
How to change username:

Select profile management then update username



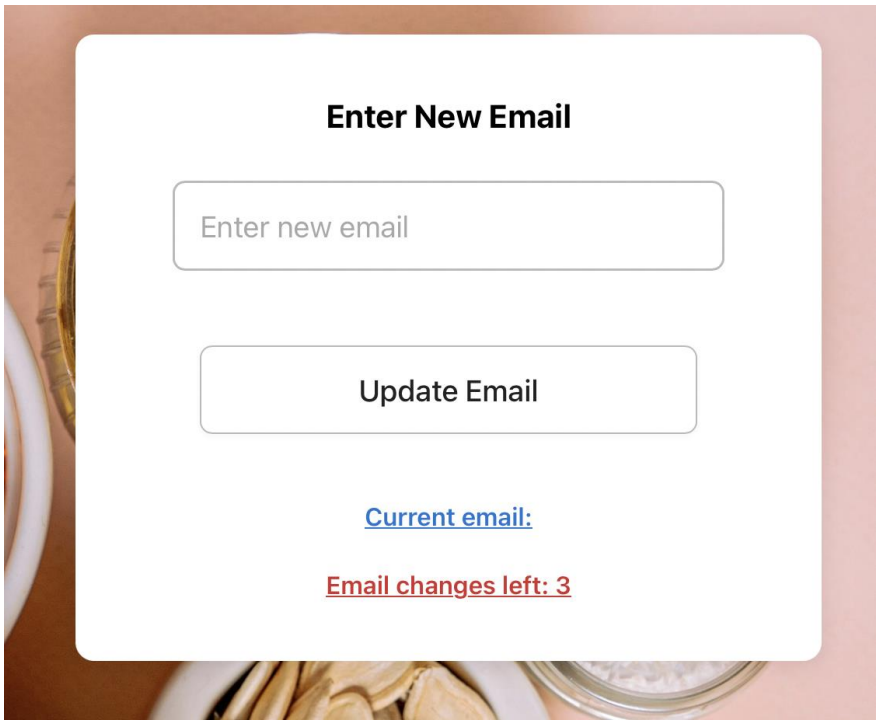
How to change password:

Select profile management then update password



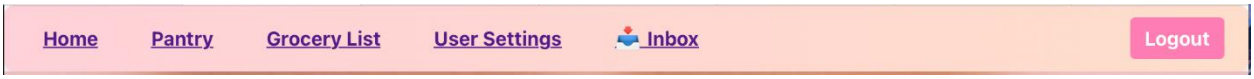
How to change email:

Select profile management then update email



Logging out:

To logout the user will select the logout button from the navigation bar.



Technical guide

To start out you need to have installed the packages Nginx, MySQL, NodeJS and npm. Each of these components are responsible for running different parts of the webserver Nginx actually serves the webpage, NodeJS and npm are responsible for running the sites JavaScript/typescript and MYSQL allows databases to be built and calls to be made from the site. After installing the packages, you need to edit the firewall on Linux to allow for each of these applications to be allowed out as well as for people to be able to make calls into the site so for Nginx you need to run ‘sudo ufw allow 'Nginx Full’. After that navigate to a web browser and plug in your IP to verify an index html page is running and viewable. After you have Nginx up and running were now going too look at configuring MySQL. After MySQL is up and running you login to the server as root from there you run the command “ALTER USER 'root'@'localhost' IDENTIFIED BY ‘password’; with password being the password you will set for the root user. The last thing to do to get the barcode scanner to work is you need to use OpenSSL to self-certify the website to allow for camera permissions to work we used this tutorial for our site. Another crucial element to the project was use of the NPM package ‘PM2’. Using PM2 allowed for us to ensure the NodeJS application ran 24/7 in the background without the need for the terminal to be open. This is a simple tool all you would need to get it running is ‘npm install pm2 -g’ the -g allows for a global install on the server you are using. Then you need to use the command “PM2 start ‘app’ ” which starts the app then you are off the ground and running with NodeJS fully daemonized and off of mind.

Clone the repository, run the build script to deploy to the server. Will have to change the server to scp into.

What we learned

Matt:

Throughout the project, I learned a lot about backend development. Going into this project I was already familiar with JavaScript. However, JS has a very complex ecosystem with many frameworks to work with, both front-end and backend. So going into the project, I took the opportunity to learn a new framework. I decided to work with NestJS since during my research, it seemed to be a very popular choice among TypeScript developers. I first started with the NestJS documentation, but honestly, the docs were not very thorough and ended up relying on many third-party sources. A lot of the ideas with NestJS such as dependency injection and modularity are things that I already knew about, but I never used those concepts in this context before. Some specific things about writing code with this framework that I learned were about the pieces of modularity that the framework utilizes, such as DTOs. DTOs provide a nice scalable way to pass data around. For example, when creating or updating an entity, if you need a new attribute or to remove an attribute, you just adjust the dto file rather than adjusting all the function calls. The DTO is simply just a regular TypeScript file that has the attributes and their types in it. So, whenever the client calls, say createUser as an example, it creates a User object and initializes the values of the DTO. Afterwards, the client just passes the DTO to the function. There are a couple of other ways NestJS offers modularity such as controllers, service files, etc., however, these concepts were not new to me. They essentially just create a way to keep the structure of an entity organized, clean, and easy to change if needed.

Working on the backend, you learn all of the little things that come along with it. One thing is routing. This was something that was almost completely new to me. Routing is just the process of the frontend communicating to the backend and to the database. When the frontend calls a function to modify the database, the backend is called through reverse proxy. You don't need to use reverse proxy, as there are other ways, but this is the case for our project. The reason I chose to use reverse proxy is because as I started to try and create users in the database, the routing seemed complicated. That's when I found out how to reverse proxy. The idea behind it was to define the reverse proxy in the nginx config file. So, when the frontend calls for the backend, the server actually routes that call the backend. In essence, the frontend and backend are completely independent of each other. The reverse proxy makes the routing simplified. For example, the frontend could do something along the lines of "fetch('/api')", where api is the reverse proxy. The routing that the frontend utilises is defined in the controller. Whenever you have multiple of the same method i.e., GET, POST, etc., the routing will just do the first one of that method. So, in this case you need to define a route such as /allUsers and /oneUser. This is a very simple

example that we didn't use but showcases that both of these use the GET method. The front-end call would be /api/allUsers or /api/oneUser.

One thing everyone needs to learn when working with new tools is how to debug. For me, a lot of my debugging was where requests were not being handled correctly to the database. I used pm2, which allows us to keep our server running at all times, even when the terminal session is closed. Pm2 comes with logs, both regular and error. The pm2 error logs showed exactly where things went wrong. For example, there could be an insert statement, but it's passing a DEFAULT value, which is not always allowed. This was an issue that I faced a couple of times. It also has the simplest solution. Sometimes when there are big changes with changing backend and how it maps with the database, the server needs to just restart. This wasn't the only issue that I came across. Another issue was simply just inserting wrong value types, or missing a value, etc. The pm2 error logs would highlight this. It will show the exact statement it tried to execute. So, if we were performing an insert statement, it would show the insert statement it executed. This along with the error message tied to it helped identify the exact issue.

Something that I didn't necessarily learn but got more experience with is working on a shared repository. An issue that I always had with this was merge conflicts. Before this project, it was always a mess and I never really knew what to do. This time around, I took the time to try and learn about it. I learned that git has a mergetool, where you simply type the command "git mergetool." This opens a window where you have the original and the remote version side by side. Then you can choose to keep certain sides or adjust it yourself. This was a helpful tool that I learned during this project.

Laura:

Before starting this project, I was very nervous because I felt like I had no idea where to start. I focused on front end development which I had a very basic understanding of; I knew it had to do with the user interface, and I was pretty sure it involved HTML coding which I had used previously. When I was assigned to front end development, I was frazzled; I had no idea what to do, I stared at my laptop for a while with VS Code open.

I was unfamiliar with the react framework and I felt uncomfortable working with JavaScript. I had to learn the basics of creating a webpage, which I relied on w3 Schools and the vs code copilot to get me started. I had to learn how to travel between the various pages we would be creating, to test if the webpages I was creating were valid. As a cybersecurity major, I feel like my coding skills are minimal, and it is a skill I'd like to get more comfortable with. This project allowed me to gain some understanding of the react

language, which is a declarative language. That means I tell it what state I want observed, in our case one of the states we tracked was whether a user was logged in, which altered their navigation bar. I was able to sharpen my skills with HTML, which I had used to create a website on a vulnerable web server in IT 350. I learned how to build separate compartments, and how to use the language. I wasn't familiar with the styling elements that go along with HTML. HTML is great for the foundation of a website, but it can look very basic without CSS styling elements. There are a few different ways to use CSS; you can either use in-line styling, CSS modules, or CSS stylesheets. I focused mostly on in-line styling because I was able to learn the various commands a bit better through repetition. However, it would have been a lot neater to use CSS modules. Since the goal is to keep the site looking as cohesive as possible, a lot of the inline styling we used was similar if not the same, which means we could have created code snippets to add to a CSS module we had preinstalled with the React files. I spent a lot of time copying, pasting, and searching for the CSS elements to ensure I was using similar colors and styling attributes throughout. If I had planned better, I would have spent some time creating the code necessary for the module, and I would have been able to have the whole team use those elements as well to keep the entire site cohesive. For the most part, I think we did a great job keeping the site cohesive, I would fix any styling elements that the team said they didn't like as much and occasionally John would help with the styling. I felt like after a certain point the job of editing the site UI was my sole responsibility. Matt would tell me specifically that something looked off or that Christian didn't like something (the navigation bar) and I would go ahead and fix it for the team.

When I was building the login and signup pages at the beginning of the process, I was really struggling with finding how to correctly format the links for the fetch statements to connect with the database. I wasn't sure how to find the correct way to traverse the backend, and I think as a whole we weren't sure as a team. Matt helped find the correct links which allowed us to successfully create users and login with those user accounts. I think part of the reason I was having trouble with it on my own was because I had set up a mock user to test the login and sign-up forms before the database and backend were completely ready. Even after I included the fetch statement, the form was storing the data as a mock user. I began working on the add item form, which would ask the user for the product name, barcode, expiration date, quantity, and where it should be stored in the pantry. This was relatively simple for me since it mostly required HTML code; however, the add item page required a connection to the backend. If I remember correctly, I created a basic code which once again stored the data for a mock user but did not send the data to the backend, so the pantry data was not being saved for each user. John ended up taking over this page to get the barcode actually working at one point, and he modified the page a

bunch from my version. I don't think I correctly tracked the data which we had gathered from the add item form. We had created a remove item form and page as well, which we ended up not using because we moved the delete item feature to the individual pantry pages. I initially added it because I thought it would make sense to scan an item right before you toss it, but my other team members mentioned it seemed a bit redundant and that the user may not type the product name or barcode correctly which could cause issues. I created pages for each of the pantry pages and started brainstorming how we would want the inventory to show up. A table made the most sense to me, so I created a table to show the product name, quantity, expiration date and date added, we then added the select/delete feature so users could adjust their inventory. I copied and pasted this table to all the pages; eventually I had to add a background to the table because the font was black, and the background clashed with it.

Early on I added a navigation bar that was relatively simple; it was just text separated by '|' and set to link to the necessary pages. I thought this was fine and looked okay, but Christian thought it looked very novice. So, I added some spacing to the headers, changed the color of the bar, and changed the look for him. This required me to learn a bit about the CSS styling and Reacts link module. This helped set up routes for the pages in our application and helped me separate what a logged in user vs a guest sees.

I learned a lot about what it takes to build a website this semester… I honestly thought it would be much simpler. I had to put work into the website every week during class time and sometimes I would work a bit out of class to ensure that I was getting the results I wanted from myself. There were some nights I thought I spent way too much time staring at a screen. But I truly appreciate the experience; it gives me some ideas as to what other websites I may want to build in the future. There are times I wished I had worked on the backend and gained some experience there, but that is something I can focus on once I graduate. I should feel comfortable enough with front-end development to be able to successfully build something without as much stress (if I am using React again). I was impressed during the demo day by how many tools and resources there are to build the components of a website in case you want some help. I would be interested in using some of those tools in the future if they could help me out.

Christian:

When I first started working on this project, I thought there's no way in the world I could contribute to this project I'm only a cybersecurity major I have limited coding experience and a limited background building web applications. However, in the end it turns out that the background I had picked up in passing and various classwork turned out

to be instrumental to helping get this off the ground as well as picking up some new skills and tools in passing during this project.

The first way that I was able to help the team out massively was building out the database and tables within it that we needed in order to support the project we wanted to do and achieve. The first and most fun step was drawing up and writing out the tables that we needed such as one to support the users and another to support all the items the users are storing. A big part of the development for the database was also developing the primary and foreign keys for each table and ensuring the 'one-to-many' and 'many-to-many' relationships made sense in practice as well as on paper. Our database heavily relied on many-many relationships such as users having many pieces of inventory or the grocery list having many different products in inventory. Once all of that was mapped out it was a matter of remembering all the nuances of SQL and ensuring things got up and running stably it the only thing that was left was making minor tweaks to the DB based on the ongoing changes with the project such as realizing that not every product needs a barcode because something like a banana doesn't have a barcode on it so I had to drop the unique constraint on the table for UPC to allow for full optimization of the grocery list and ensure that every item a user might need is included in the list. I also was constantly making sure that whatever data was being sent from the site was slotting into the DB normally and that everything was on the up and up and the DB was normalized to the proper standard.

Something that I learned along the way was how to set up a Nginx web server and have it served content. This was something that I had never done before so I was a little bit nervous about setting it up properly because at the end of the day this was what everyone was going to see as well as what you would grade us on. Luckily for me, there were enough tutorials online plus a little bit of help from AI I was able to get the server up and running. The one thing that I had wished I had figured out was how to serve the site with an actual domain name as opposed to the IP to give it a more professional look, but I'll take functionality over looks any day of the week especially when it's something as tedious as the url.

Something that I had to relearn on the fly for this project was self-certifying a website using OpenSSL. The reason I had to do this was because without HTTPS the camera functionality of our barcode scanner wouldn't work due to obvious privacy concerns that lied within using a camera in a plaintext protocol. After refreshing my memory with a DigitalOcean article I was able to get the website up and running as well as getting the Nginx tuned to only listen on port 443 and successfully use camera permissions on every device from mobile to laptop. The only issue with this process is every web browser serves an error with the certificate as self-signed which might discourage people from using the site. However,

this was the best free way to make this happen with the way everything we had set up as well as the constraints we had in place in terms of time.

The last part that was a part of my responsibilities was finding the proper API's to fit our use cases and integration of the API. Luckily, I had experience with the initial API we set with which was a simple barcode to product API from Nutritionix which met our cases with a simple end point and allowed for us to make just enough calls to do the project with the number of users that we had set out with. Unfortunately, about halfway through the project I got an email saying that they were revoking the free tier which was what we were relying on given the fact that everything we did had to be for zero cost. But we were hopeful as a group that we could hold on long enough to finish the project. We got another bad dose of news when on Friday before we went on break, I got another email from them saying that our access had officially been revoked which means that we had to change to a new provider on-the-fly. Thankfully, Matt came in massively with one he found from OpenFoodFacts which allowed us to make calls to an endpoint no registration required. So, it was a simple change the API URL and rework what it was specifically pulling from that call.

John:

I learned a fair deal about front-end development during the project. Part of my work was working with REACT framework which was new to me. Learning how to use the framework was significantly different from learning a new language. While knowing a programing language will give you a good intuition for nearly every other language, it doesn't help you very much for frameworks. Learning the framework involved a lot of looking through documentation and searching the internet. There was a lot to learn and when starting it was difficult to know where to start. Even after finishing this project, I don't feel particularly confident in my ability to work with this framework.

I also learned how to function in a team with this project. There are many differences from working in a team compared to being alone. The biggest thing is that not everyone has the same knowledge as you. While I may know exactly what information I need from the backend to perform some sort of calculation on the frontend, Others don't have my though process and don't know that. This was largely apparent when communicating with group members that were majoring in cyber security and not computer science like I am. Cyber security majors don't have the same courses I do and didn't fully understand what I needed. Another thing I learned was how to find work for myself to do. In solo projects or small group projects you usually have some set

requirements to fulfill and can easily divide the work up. However, when working on a project on this scale sometimes things don't work out so cleanly. There were times I had to wait for someone else to finish their work and find something to do. Usually, I would use this time to work on a minor feature for the application, such as decrementing stock directly from the shelfs. There were also times when I was unsure what needed to be done next. Sometimes I just had to look at the somewhat vague requirements we had laid out and take a stab at it. This did not always work out; one time I started working on one of the web pages only to find out that none of the backend logic for that page existed, so I had spent a while working only to have little more than a stylized blank page to show for it.

During the project I got some hands on experience interacting with a database. I had done some basic stuff with it before, but this was the first time I really got down to it. One of my previous classes had me working with Springboot, which has its own way of doing databases where it's part of Springboot. That felt more like I was learning how to interact with a database in Springboot and not how to interact with one in general. This was much more straightforward and more importantly, wasn't me and one other guy trying to do a five-man project by ourselves. This time I had the time to sit down and see how things worked rather than searching the internet at 2 AM begging for a working solution so I could go to sleep.

I also learned how to properly use GitHub to manage a project. This was the first time I had ever properly used GitHub to manage a project before. It was a significant improvement compared to the last project I was on where it was me and one other person sharing files over discord. At first it was rather confusing with the commands and version history, but I eventually got the hang of it. Merge conflicts occurred sometimes and were always painful to figure out. Sometimes it was easier for me to discard my changes, fetch the new version of the file, and add my changes to that than try to figure it out. I would like to learn how to properly utilize the merge conflict handler in the future, as I didn't take the time to sit down and properly learn it.

The last thing I learned from this project was working with HTML and CSS. It had been a long time since I worked in these languages and at first, I had no idea how to do many things. This was also the largest project I have ever done with HTML. My previous projects were rather small in size and often didn't have more than a single page. It didn't help that I didn't like working in these languages either, as they don't usually tell you what you did wrong. It is very annoying to find out that you're trying to set an invalid attribute in CSS after twenty minutes of trying because there's no error message saying what you're doing is wrong. I had to relearn a few things to get back up to speed. I also had to help

Laura with various issues, as well as provide feedback on styling. After this project I can that that I am more knowledgeable in achieving a desired effect with CSS.

lfe:

Throughout this class I would say that I learned quite a bit, both technically and personally. One of the biggest and most meaningful areas of growth for me came from learning how to work effectively in a team setting. In most of my previous classes and projects I have either worked alone or taken on work that felt more individual than collaborative. This class required a very different approach, and it pushed me to understand how valuable teamwork can be when everyone is committed to the same goal.

I realized how important it is to communicate constantly and make sure everyone stays on the same page. It is not enough to assume people understand your ideas or your progress. You have to talk about it, check in with each other, and be open about challenges and changes. That kind of communication ended up being a major part of why our group was able to stay organized and move forward steadily.

On the technical side I learned how to work with several new tools. React was the biggest one, and getting comfortable navigating it and applying it to our system took time. Since I focused completely on the front end I also spent a lot of time working with HTML and making sure that whatever I added to our website looked presentable and functioned the way it was supposed to. This combination of design and technical problem solving helped me understand what front end development actually feels like in a real project.

Even with all the technical lessons I still think my biggest takeaway is the importance of group communication. Throughout the class I heard about other groups struggling because certain teammates were absent or the group dynamic was completely dysfunctional. In our group we made a real effort to avoid that. Everyone showed up, everyone talked, and everyone participated. That level of involvement from every member made the whole experience smoother and more enjoyable. It also played a big role in why our final project turned out the way it did. By the end we had something we could all be proud of, and it felt like the result of genuine teamwork rather than a set of individual pieces forced together.

Appendix

Update 1:

1) What customer interactions did you have this week?

This week we have no interactions planned with the customer; last week we had a meeting where we finalized the project plan and figured out which components are best to include in our application. This week we plan on creating the database and finalizing the server which our application will operate on.

2) General project overview of status.....how are things going, etc.

Our project currently is on track with the GANTT chart; we have completed the planning and design phase. Next, we need to start on the execution phase, in which we will begin actually creating the tool. We have been meeting every week at least once in order to stay on track.

3) Issues - Not resolved at this time. List the things that are causing the project problems.

Currently the issues we are encountering are a bit trivial since we have not begun on the backend development yet. We have not decided what we want our application name to be, this is something that can be resolved over time as we finalize the application and figure out our team chemistry.

We still are experiencing some issues with finalizing which APIs we should use for our application, but this is simply because of the constraints we may experience with using various APIs, most do not allow for an unlimited search option.

4) Issues - Resolved since last update. List the things that caused the project to stall and that you've worked thru since the last update.

Virtual machines are up and running now, which was slowing us down last week. We had no way of creating the environment and began the work of building our database, which is step 1.

5) Schedule status - Behind, ON, Ahead

As of this week we are currently on schedule with our progress. We have begun execution on building our database and our API connections alongside pretty much having our planning phase all complete.

6) Projected status for next update in 2 weeks - Behind, ON, Ahead

In two weeks, were projected to still be right on track and starting on our front-end development.

7) What customer interactions are planned for next week?

We plan on setting up a meeting with our customer next week

Update 2:

Our group did not have any customer interactions this week. Overall, our project is going well and any snags we hit during the last week have been resolved. The only issue that we have not resolved yet is figuring out how to get the API to interact with the website and how we can also pull the barcode reader information into the API. The main issue we resolved from last week to this week is the ability to transfer files, which was a problem that was resolved as well as figuring out the interaction between user and database. So far our testing with the database and code works, and a user should be able to be created and used on the site once we get a sign-up page going. Our project at the time of writing is well on schedule and we hope to have a working prototype soon as we are nearing a point where we can do some real-world testing with real users and barcodes. We project to be on pace as well in the upcoming weeks as nothing has come up or we have any projected problems in the next few weeks. In terms of customer interaction, that we have planned, we don't have anything right now, however we plan to discuss that on Thursday when we meet and see if there are any concerns that we need to address with the customer.

Update 3:

Previous Team Lead - Laura, Previous Customer Liaison - Christian

New Team Lead - Matt, Previous Customer Liaison - John

1) What customer interactions did you have this week?

We did not have any customer interactions this week

2) General project overview of status.....how are things going, etc.

Things are going well. Frontend is being built at a good pace as well as the backend. We are almost finished with building all of the entities in the database in our codebase.

3) Issues - Not resolved at this time. List the things that are causing the project problems.

Connection with the frontend and backend.

4) Issues - Resolved since last update. List the things that caused the project to stall and that you've worked thru since the last update.

We resolved the issue with the barcode scanner. We have figured out how to get it to work, and now we just have to implement in our app.

5) Schedule status - Behind, ON, Ahead

On schedule.

6) Projected status for next update in 2 weeks - Behind, ON, Ahead

On schedule.

7) What customer interactions are planned for next week?

John plans on meeting with the customer sometime this week.

<u>Update 4:</u>

1) What customer interactions did you have this week?

None.

2) General project overview of status.....how are things going, etc.

Things are going well, it's mostly just connecting everything together.

3) Issues - Not resolved at this time. List the things that are causing the project problems.

Not any issues at the moment. Again, it's just connecting certain pieces such as barcode scanner creating the products.

4) Issues - Resolved since last update. List the things that caused the project to stall and that you've worked thru since the last update.

We were able to create a product manually, and fixed all current database issues with updating.

5) Schedule status - Behind, ON, Ahead

ON

6) Projected status for next update in 2 weeks - Behind, ON, Ahead

ON

7) What customer interactions are planned for next week?

None

8) List each Team Member's name and what they did.

Matt - User and Product logic for database

Christian - Database/api testing

Ife - Frontend work such as settings

John - various frontend work

Laura - various frontend work

<u>Update 5:</u>

1) Demo presented to customer

2) Things are going well.

3) Grocery List isn't implemented completely, still working on api communication to ensure everything works as expected, just minor things.

4) We fixed all of the user inventory fetching to display on correct pages.

5) ON

6)ON

7) No customer interactions this week

8) Matt - Backend for user inventory logic

John - Various frontend work for adding items

Laura - Various frontend work in terms of the looks of it

Ife - User settings in frontend

Christian - api connection, barcode fixes

Update 6:

1) What customer interactions did you have this week?

This week we did not have interactions with our customer. After last week's all hands-on meeting, it was decided that there would be no more customer interactions.

2) General project overview of status.....how are things going, etc.

Overall, the project's status is very positive. Right now, we have all of the base functions working that we set out to do, such as a working barcode scanner, working pantry/fridge/freezer pages that accurately track a user's inventory and push out to the grocery list for shopping purposes. We are also actively working on getting a printable version of the list for those that still prefer to have things done on paper or have a physical copy. The group is also working on documentation for the site as well as the scripts that the customer requested to rebuild the database if wanted.

3) Issues - Not resolved at this time. List the things that are causing the project problems.

Currently, we are experiencing an issue with our user profile/ user settings page. Ife, Laura, and John had been working on the frontend and realized last week that the fetch links in the settings page were causing issues.

4) Issues - Resolved since last update. List the things that caused the project to stall and that you've worked thru since the last update.

We have resolved any issues regarding the barcode scanner, made sure it works on both PCs and Mobile devices.

5) Schedule status - Behind, ON, Ahead

We are basically on schedule compared to our GANTT chart, we are supposed to be in a testing and validation phase. This week the team is testing the usability of the web

application and seeing if we need to add a better UI/UX, and we are making sure that all our features are working as expected.

6) Projected status for next update in 2 weeks - Behind, ON, Ahead

In the next update, the final presentation will be complete with our application. The goal for the next 2 weeks is to complete our final report, final presentation, and fine tune any remaining issues we find during the next two weeks.

7) What customer interactions are planned for next week?

No customer interactions are planned next week. The team has had very few questions for the customer lately and has not had a need for customer interaction.

8) List each Team Member's name and what they did.

Laura: Added grocery list navigation, CSS updates to pantry and user settings

John: frontend grocery list, user settings page

Ife: finalized Inbox page and added it to the NAV bar

Matt: Backend grocery list, user settings page

Christian: Database changes for grocery list and barcode scanner