Sprint 2 Report (6/1/2024 - 6/30/2024)

What's New (User Facing)

- Working order menu, fetching item information from backend and sending receipt info to backend on checkout. Can categorize items on button click to only show that type e.g drinks.
- Login page with 2FA via email (currently configured in .env in Readme of main branch)
- Launchpad for navigation to other pages and logout. User on successful login is sent to this page, and other pages like menu can navigate back to it.

Work Summary (Developer Facing)

During Sprint 2, the team transitioned from planning to implementation, focusing on developing a functional fullstack for the web application, using techniques and languages learned in the previous sprint's preproject. These are the key features of the backend:

1. JWT cookies.
2. Email OTP handling.
3. Role based authentication.
4. PostgreSQL database.

Firstly, we'll discuss JWT cookie handling. There are two different JWT cookies created, one time for OTP, and the second time post OTP. The JWTs hold roles signed by the secret, after being decoded, the OTP cookie contains the role null and the post login JWT contains the role admin or employee. Admins can create new users and add stock to the database. Since the cookies are HTTP only, these cookies are not vulnerable to javascript injection, and because these signed JWT tokens are necessary to interact with any endpoint as a result of the implemented strategy covering every endpoint, a user must have a cookie to make meaningful changes to data in the database.

There are several relations in the database, a few for the user as well as many for inventory, recipes, and orders. The inventory, recipe, and order relations work together for inventory management. For each order taken, an aggregation of ingredients is taken from each item and subtracted from the inventory closest to expiration. To view current items, the aggregate sum of each inventory item is taken. To view items close to expiration, each item in the database is measured against the current date's distance to the expiration date. If the item is seven days from expiration, the item is added to a list as well as its corresponding inventory number to send to the frontend. Logic to view low quantity items is similar to the other two views; the aggregate sum of each item in inventory is taken, and for each item below the arbitrarily set 10 days, that item is returned to the frontend. While there is decent coverage of functionality for the database, more methods should be created to realize the architecture's potential in both revenue and inventory management.

The frontend was implemented with React and contains a fully functional menu and checkout ordering system that is dynamically scalable based off of the amount of items in the fetched menu item data from backend. A launchpad after login was also built, for navigating to the menu and other features like inventory. The frontend was tested at multiple view scales in desktop orientation and remained legible. Specific frontend struggles included getting menu items formatted in a grid correctly, setting up the cart sidebar, and having the shopping cart's checkout info be properly formatted (as the shopping cart initializes with all items at zero count).

Informal Review

Below is an informal summary and showcase video of what was developed:
https://drive.google.com/file/d/1fuTtd30FAswRuyAF__oAcY2_CUiO1bHp

Additional informal documentation of Samantha Brewer's frontend development between Github commits can be read here.

Unfinished Work

The revenue page remains to be implemented, the planned revenue page should only be accessible to admins and should contain multiple views. The conceptualized plan is a view with a graph of revenue over arbitrary periods of time. The backend is also unable to accommodate this view. The aggregated sum of each order over some arbitrary time period was not implemented; however, the database's architecture can accommodate this. With the implementation of a method, the revenue page can be created.

The frontend is functional, but still minimalist and uses placeholders at points. Given more development time, the design and look of the web pages will be improved with the incorporation of logos and dynamic animations. The reasons for the infeasibility of total feature completion prior to this sprint are the following:

1. Communication issues
2. Withdrawal of a teammate

While some setbacks were faced in this sprint, our team is confident of this project's completion.

Completed Issues/User Stories

Here are links to the issues that we completed in this sprint:

- Frontend - Implement grid Menu Ordering system: https://github.com/KleinMandolin/CptS322-Project/issues/21
- Frontend - Login Page: https://github.com/KleinMandolin/CptS322-Project/issues/22
- Issue 23 - Authentication for session initiation using JWT tokens: https://github.com/KleinMandolin/CptS322-Project/issues/23

- Issue 25 - Recipes, Ingredients, and RecipeDetails: https://github.com/KleinMandolin/CptS322-Project/issues/25
- Issue 26 - backend: orders orders-details: https://github.com/KleinMandolin/CptS322-Project/issues/26
- Finish authentication: https://github.com/KleinMandolin/CptS322-Project/issues/28
- Prototype alert page: https://github.com/KleinMandolin/CptS322-Project/issues/29
- Create authentication framework to modularly protect routes: https://github.com/KleinMandolin/CptS322-Project/issues/31
- Frontend - Add Backend API Calls to Menu: https://github.com/KleinMandolin/CptS322-Project/issues/33
- Frontend - Create Landing Page: https://github.com/KleinMandolin/CptS322-Project/issues/34
- Add expiration date check to inventory service: https://github.com/KleinMandolin/CptS322-Project/issues/40

Incomplete Issues/User Stories

Below are the uncompleted issues for this sprint:

- https://github.com/KleinMandolin/CptS322-Project/issues/46 - Issue 46 states a revenue view must be created with a graph of revenue over some arbitrary period of time.
- https://github.com/KleinMandolin/CptS322-Project/issues/45 - Issue 45 states SSL/TLS be implemented.
- https://github.com/KleinMandolin/CptS322-Project/issues/12 - Issue 12 states deployment of the frontend to Vercel.
- https://github.com/KleinMandolin/CptS322-Project/issues/11 - Issue 11 states deployment of the backend to Supabase

Our team was unable to complete these features for the reasons mentioned in the Unfinished Work section of this report: (1) the withdrawal of a teammate and (2) communication issues. Granted setback 2, our team has to reevaluate the scope of this project. Incompleted issue 46 is a necessity; however, the other issues are unlikely to be completed.

Code Files for Review

Please review the following code files in the main branch, which were actively developed during this sprint, for quality:

- Backend - services, entities, modules, and controllers.
  - Auth
  - Email
  - Ingredients
  - Inventory
  - Middleware

    - Order-Details
    - Orders
    - Recipe-Ingredients
    - Recipes
    - User-Management
    - *app*
    - *main.ts*
- Frontend
    - Auth
    - Inventory
    - Launchpad
    - Login
    - Menu
    - Revenue
    - Components
    - *App.tsx*
    - *App.css*
    - *Index.css*
    - *main.tsx*

Retrospective Summary

Here's what went well:

- Better division/sharing of tasks, with Kyle jumping between frontend and backend dev as the backend finished early compared to frontend.
- Better overall communication, with extra meetings that helped us understand who was struggling or what needed to be worked on.

Here's what we'd like to improve:

- Better time management, so that we could have a small working pipeline that components get incrementally added on to, instead of individually working on each component and trying to combine them all at the end.
- Better communication on non-committed contributions, so that we know if someone has already worked on a specific part before spending more time on it. A specific instance is the launchpad, where all three of us made a different prototype implementation of it.

Here are changes we plan to implement in the next sprint:

- Better frontend visuals. As said before, the frontend uses lots of placeholders and bland designs, with update with a more professional look going forward