Senior Capstone Project Update

Project Name

Household Menu Planner and Ingredient Tracker (HoMePIT)

Team Members

- Erik Anderson
- Jerome Chestnut
- Maxwell Fugette

Update Description

1. Changes

A number of changes have been made to HoMePIT over the past few weeks, many of them relating to the database. The first change made would be the addition of a user ID column and row-level-security policy to every table to ensure that users can only ever access their own data. Furthermore, the Recipe Ingredients table had columns added to track the total nutritional information and cost associated with each ingredient of a recipe, as doing so simplified and atomized some database functions. Some of these functions included those that were added to automatically update a recipe's nutritional and cost information when relevant information changed in other tables. A table to audit changes to ingredient quantities was added, as it was realized that this was necessary to reasonably track ingredient quantities over time. This table will also be paired with functions and triggers to log the audits as necessary. The table of measurement units has been removed, as tracking and accounting for any given ingredient's specified measurement unit was determined to be excessive and unnecessary when such conversions can occur in the client before sending this information to the database. Instead, the database solely tracks how many grams or milliliters of any ingredient exists in a user's Pantry. Similarly, the decision was made to at least temporarily cut the tracking of substitutes, sub-recipes, leftovers, and expiration dates, as these are minor quality-of-life features that require outsized quantities of development time to realize. Given current time constraints, this decision was seen as necessary. Likewise, the previously mentioned functionality of optimizing shopping list routes between multiple purchase locations is somewhat likely to be changed from a "guaranteed" to a "nice to have" or "future" feature. Should any significant setbacks occur, this change of priority may also be made to the purchase location tracking feature as a whole, given its minor auxiliary role in the function of HoMePIT.

In addition to various changes to HoMePIT's backend, numerous changes have been made to the frontend as well. Early in the development process it was decided that using Replit as the primary IDE was not desirable, given that the team overall was more familiar with VSCode and found Replit to be slow and had difficulties with version control. Initially Replit was chosen due to various features, such as the possibility of free hosting, ease-of-access, plugins for Supabase, and even a mobile app which would allow work to be completed when no team member had access to a computer. Despite these benefits, Replit is a newer platform than VSCode and as such is not as reliable and widely used. In addition, there exists little documentation for using Replit with the two main technologies of HoMePIT, Supabase and Next.js. Overall, the simplicity and familiarity of VSCode made it the obvious choice for an IDE within the context of this project.

As stated above, one of the initial reasons for using Replit was the possibility of free website hosting. After the decision was made to switch to VSCode as the primary IDE, it became clear that the team would have to find another way to host HoMePIT. The possibility of using Google Firebase for hosting was discussed, but fortunately it was discovered that Replit has a feature that allows projects to be hosted after being imported from GitHub, which would be ideal for the purposes of this project. For the time being hosting is only necessary for the project demo; at a later date this will become a more pressing issue.

Perhaps the most glaring change to the frontend so far is the structure of the different pages in code. Initially, much of the functionality for creating, reading, updating, and deleting things such as ingredients or recipes were handled in the frontend, without ever stopping to look at how this would interact with the backend. Various components such as the "Add Ingredient" form were once broken up into separate files, which made the code appear scattered and difficult to understand. Upon overhauling the primary components and paying attention to how things interact with the database, the code became much more streamlined with the primary pages consisting of roughly the same components. The Pantry, Recipe Book, and Meal Planner all consist of state variables (imported from the React useState library) for handling data such as ingredient name, quantity, etc, CRUD (create, read, update, delete) functions that allow the user to interact with the database, and an HTML component that is the visual display for the user.

One aspect of the project that was not explicitly stated, but was expected, was the potential use of third party libraries to create the calendar component in the Meal Planner page. As it stands, HoMePIT is currently using a React calendar extension called FullCalendar to serve as the visual display of the users meal plan. The extent to which this library will be used is currently unknown, but the team intends to keep third party code to a minimum. The intent is to only use such libraries when the team's attention would be better focused elsewhere in the project, given HoMePIT's lengthy requirements list and the impending final submission date.

2. Accomplishments

a. Complete

HoMePIT as a whole is currently in a partially complete state, although some individual elements within it are nearing or have achieved completion. With respect to the database, previously specified tables have been implemented and provided with row level security policies to ensure that only the owner of any given data is allowed to access, modify, or delete it. As mentioned in Changes, the Ingredients, Recipes, and Recipe Ingredient tables have had functionality implemented to auto-update nutritional and cost information as necessary

With respect to HoMePIT's frontend, various components are in a completed state. Upon opening the application, the user is taken to a home page which presents the option to sign in with Google. Once clicked, the user will be taken to an authentication page where their account will be granted an access token and stored in the database. At the top of every page the navigation bar will be found, with tabs to access the Pantry, Recipes, Meals, Lists, and Settings pages. The Pantry page functions as it should, with the user being able to create, update, and delete ingredients as they wish. The page consists of a list of ingredients created by the user, displaying the ingredients name and quantity, with options to edit or delete. Above this list there is a button that when clicked will prompt the user for information regarding adding an ingredient, and a final "Add Ingredient" button which will log the inputted information in the database. Changes made here will then be reflected in the Pantry's ingredient list. Following the Pantry, the Recipes page functions nearly the same, with the only notable difference being that Recipes are the object in question rather than ingredients.

b. Partially Complete

Much of the functionality of the HoMePIT database is partially complete. This is primarily because of the addition of the Ingredient Audit table, as ingredient audits can be prompted by operations performed on the tables of Ingredients, Recipes, Recipe Ingredients, Meals, Meal Recipes, and Shopping List Items. Each operation performed on each table has a different impact on the Ingredient Audit table, and thus it is largely impossible to generalize the behavior of the functions that maintain the audit table.

Much of HoMePIT's frontend has work that still needs to be done. There is currently an unresolved issue with user authentication, where upon signing in the user is still unable to alter the contents of the database due to the current RLS policy. The Pantry page, while still functional, lacks any methods for data validation to ensure that the user is limited in what can be added as an ingredient. The Recipes page is in a similar state, and lacks functionality beyond simply creating, updating, or deleting a new recipe in the database. There is currently no way to add individual ingredients to a recipe, and no way to handle unit conversion, which will be discussed in section 3.

3. Still to be Implemented

A number of elements still require implementation. The most notable such element in the database would be the generation of shopping lists and optimization of purchase locations, should that remain an element that is to-be-delivered. Outside of shopping list generation and related functionality, there are some utility functions that require implementation in various parts of the database in order to simplify calls to the database from the client.

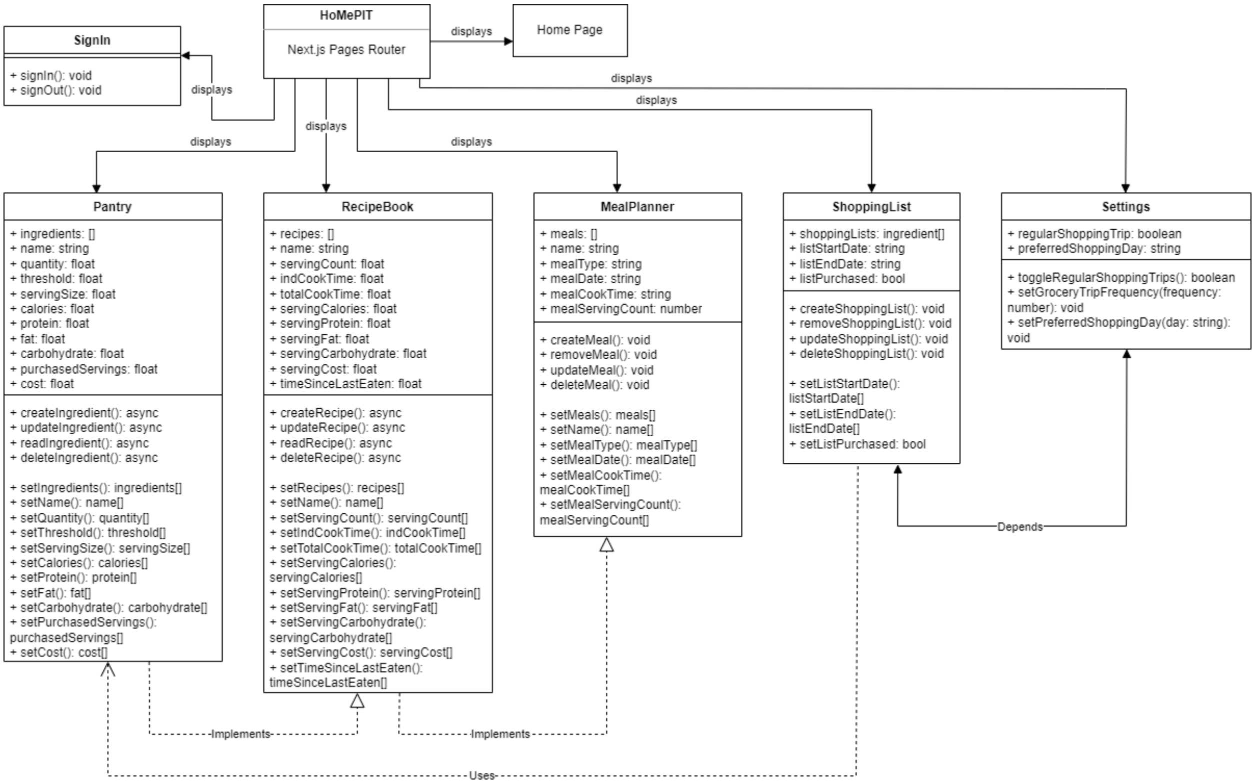
Aside from those database-specific elements, functionality for allowing users to specify units aside from grams and milliliters still needs to be implemented in the client. There currently exists no methods for unit conversion, which is a core component of this project. The Meal Planner's calendar component is present, but has no functionality beyond displaying the current month. Work still needs to be done to be able to add meals and mealtimes to a given day, along with methods for calculating total cost and nutrition data over a given date range. The Shopping List and Settings components have yet to be worked on, and their respective pages display nothing but a blank screen. As stated previously, data validation has yet to be implemented in the frontend, and if the user inserts data that does not match its definition in the database, an error will result.

4. Issues

As mentioned before, some or much of the Shopping List functionality may end up having to be reduced or cut due to time constraints. This would cause a reduction in the total functionality of HoMePIT, but it wouldn't ultimately have a major impact on the overall ability to deliver the core functionality of HoMePIT. Likewise, the sub-recipe, substitution, leftover, and expiration date functionalities have already been cut from HoMePIT, which has a minor impact on its general quality of life features, but minimal impact on core functionality. Finally, storage of measurement units and interpretation of that information has been removed from the database, which has a minor positive impact on the project by simplifying the database's contents.

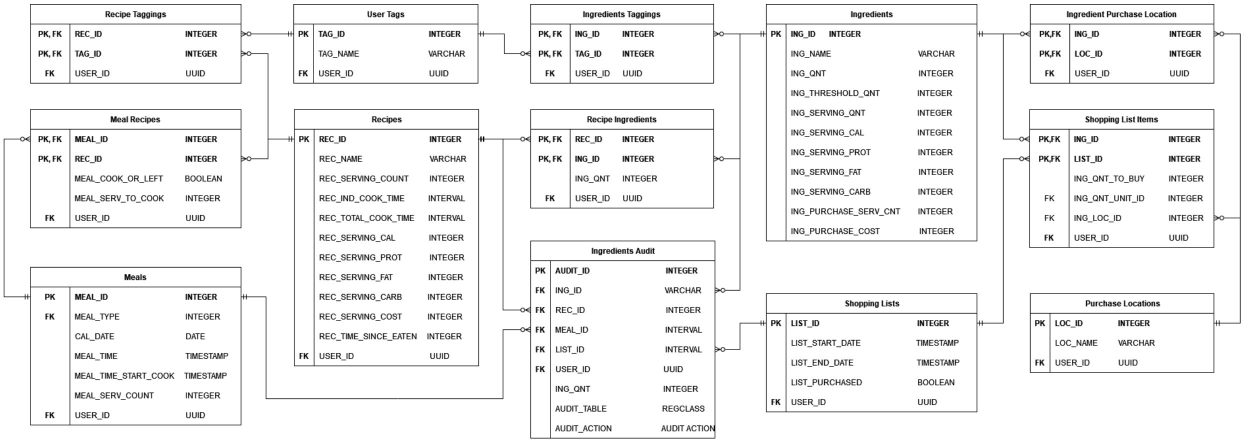
Updated Diagrams

1. Components Diagram



HoMePIT
Next.js Pages Router
displays
Home Page
displays
displays
SignIn
+ signIn(): void
+ signOut(): void
displays
displays
displays
displays
Pantry
+ ingredients: []
+ name: string
+ quantity: float
+ threshold: float
+ servingSize: float
+ calories: float
+ protein: float
+ fat: float
+ carbohydrate: float
+ purchasedServings: float
+ cost: float
+ createIngredient(): async
+ updateIngredient(): async
+ readIngredient(): async
+ deleteIngredient(): async
+ setIngredients(): ingredients[]
+ setName(): name[]
+ setQuantity(): quantity[]
+ setThreshold(): threshold[]
+ setServingSize(): servingSize[]
+ setCalories(): calories[]
+ setProtein(): protein[]
+ setFat(): fat[]
+ setCarbohydrate(): carbohydrate[]
+ setPurchasedServings(): purchasedServings[]
+ setCost(): cost[]
RecipeBook
+ recipes: []
+ name: string
+ servingCount: float
+ indCookTime: float
+ totalCookTime: float
+ servingCalories: float
+ servingProtein: float
+ servingFat: float
+ servingCarbohydrate: float
+ servingCost: float
+ timeSinceLastEaten: float
+ createRecipe(): async
+ updateRecipe(): async
+ readRecipe(): async
+ deleteRecipe(): async
+ setRecipes(): recipes[]
+ setName(): name[]
+ setServingCount(): servingCount[]
+ setIndCookTime(): indCookTime[]
+ setTotalCookTime(): totalCookTime[]
+ setServingCalories(): servingCalories[]
+ setServingProtein(): servingProtein[]
+ setServingFat(): servingFat[]
+ setServingCarbohydrate(): servingCarbohydrate[]
+ setServingCost(): servingCost[]
+ setTimeSinceLastEaten(): timeSinceLastEaten[]
MealPlanner
+ meals: []
+ name: string
+ mealType: string
+ mealDate: string
+ mealCookTime: string
+ mealServingCount: number
+ createMeal(): void
+ removeMeal(): void
+ updateMeal(): void
+ deleteMeal(): void
+ setMeals(): meals[]
+ setName(): name[]
+ setMealType(): mealType[]
+ setMealDate(): mealDate[]
+ setMealCookTime(): mealCookTime[]
+ setMealServingCount(): mealServingCount[]
ShoppingList
+ shoppingLists: ingredient[]
+ listStartDate: string
+ listEndDate: string
+ listPurchased: bool
+ createShoppingList(): void
+ removeShoppingList(): void
+ updateShoppingList(): void
+ deleteShoppingList(): void
+ setListStartDate(): listStartDate[]
+ setListEndDate(): listEndDate[]
+ setListPurchased: bool
Settings
+ regularShoppingTrip: boolean
+ preferredShoppingDay: string
+ toggleRegularShoppingTrips(): boolean
+ setGroceryTripFrequency(frequency: number): void
+ setPreferredShoppingDay(day: string): void
Depends
Implements
Implements
Uses

2. ERD



Recipe Taggings		
PK, FK	REC_ID	INTEGER
PK, FK	TAG_ID	INTEGER
FK	USER_ID	UUID

User Tags		
PK	TAG_ID	INTEGER
	TAG_NAME	VARCHAR
FK	USER_ID	UUID

Ingredients Taggings		
PK, FK	ING_ID	INTEGER
PK, FK	TAG_ID	INTEGER
FK	USER_ID	UUID

Ingredients		
PK	ING_ID	INTEGER
	ING_NAME	VARCHAR
	ING_QNT	INTEGER
	ING_THRESHOLD_QNT	INTEGER
	ING_SERVING_QNT	INTEGER
	ING_SERVING_CAL	INTEGER
	ING_SERVING_PROT	INTEGER
	ING_SERVING_FAT	INTEGER
	ING_SERVING_CARB	INTEGER
	ING_PURCHASE_SERV_CNT	INTEGER
	ING_PURCHASE_COST	INTEGER

Ingredient Purchase Location		
PK,FK	ING_ID	INTEGER
PK,FK	LOC_ID	INTEGER
FK	USER_ID	UUID

Meal Recipes		
PK, FK	MEAL_ID	INTEGER
PK, FK	REC_ID	INTEGER
	MEAL_COOK_OR_LEFT	BOOLEAN
	MEAL_SERV_TO_COOK	INTEGER
FK	USER_ID	UUID

Recipes		
PK	REC_ID	INTEGER
	REC_NAME	VARCHAR
	REC_SERVING_COUNT	INTEGER
	REC_IND_COOK_TIME	INTERVAL
	REC_TOTAL_COOK_TIME	INTERVAL
	REC_SERVING_CAL	INTEGER
	REC_SERVING_PROT	INTEGER
	REC_SERVING_FAT	INTEGER
	REC_SERVING_CARB	INTEGER
	REC_SERVING_COST	INTEGER
	REC_TIME_SINCE_EATEN	INTEGER
FK	USER_ID	UUID

Recipe Ingredients		
PK, FK	REC_ID	INTEGER
PK, FK	ING_ID	INTEGER
	ING_QNT	INTEGER
FK	USER_ID	UUID

Shopping List Items		
PK,FK	ING_ID	INTEGER
PK,FK	LIST_ID	INTEGER
	ING_QNT_TO_BUY	INTEGER
FK	ING_QNT_UNIT_ID	INTEGER
FK	ING_LOC_ID	INTEGER
FK	USER_ID	UUID

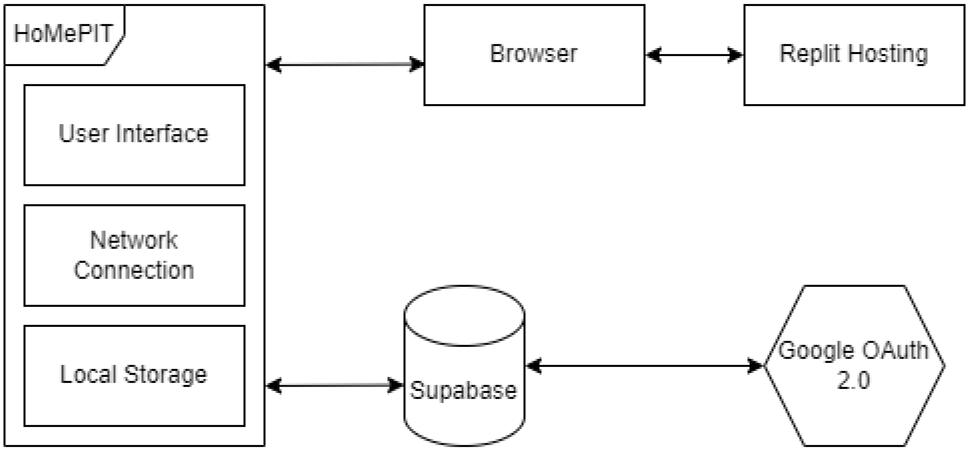
Meals		
PK	MEAL_ID	INTEGER
FK	MEAL_TYPE	INTEGER
	CAL_DATE	DATE
	MEAL_TIME	TIMESTAMP
	MEAL_TIME_START_COOK	TIMESTAMP
	MEAL_SERV_COUNT	INTEGER
FK	USER_ID	UUID

Ingredients Audit		
PK	AUDIT_ID	INTEGER
FK	ING_ID	VARCHAR
FK	REC_ID	INTEGER
FK	MEAL_ID	INTERVAL
FK	LIST_ID	INTERVAL
FK	USER_ID	UUID
	ING_QNT	INTEGER
	AUDIT_TABLE	REGCLASS
	AUDIT_ACTION	AUDIT ACTION

Shopping Lists		
PK	LIST_ID	INTEGER
	LIST_START_DATE	TIMESTAMP
	LIST_END_DATE	TIMESTAMP
	LIST_PURCHASED	BOOLEAN
FK	USER_ID	UUID

Purchase Locations		
PK	LOC_ID	INTEGER
	LOC_NAME	VARCHAR
FK	USER_ID	UUID

3. Block Diagram


HoMePIT
User Interface
Network Connection
Local Storage
Browser
Replit Hosting
Supabase
Google OAuth 2.0