HoMePIT- Final Presentation

By Erik Anderson, Jerome Chestnut, and Maxwell Fugette

Introduction

- HoMePIT will assist with planning a household's menu and tracking its supply, usage, and replacement of ingredients and other foodstuffs.
- Its primary features are the Pantry, Recipe Book, Meal Planner, and Shopping List
- This presentation will serve to summarize the current state of the project, its structure, design, and known issues.

Tools and Tech

Platform: Firefox, Chrome, Safari

OS: Windows 10 & 11, Android

IDE: VSCode - abandoned Replit & VisualStudio

Languages: JS, TS, HTML, CSS, PL/pgSQL - Abandoned C#

3rd party tools, libraries, & services: GitHub, SupaBase, React, Bootstrap, MaterialUI, FullCalendar

Abandoned ASP.NET

Server software & services: Node.JS, Google Firebase, SupaBase

Database Design

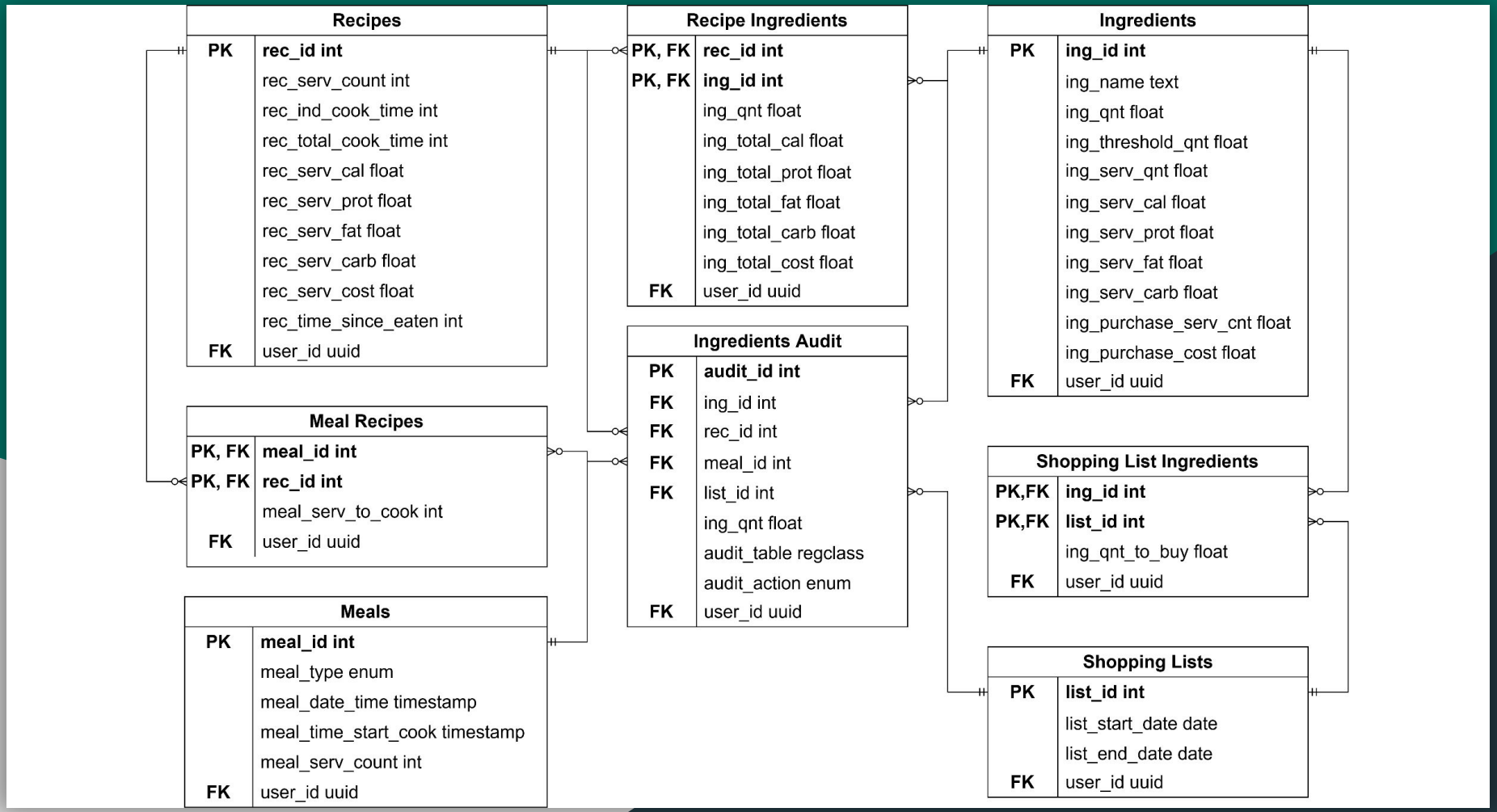
Designed with 3NF in mind - no functional dependencies only on the primary key

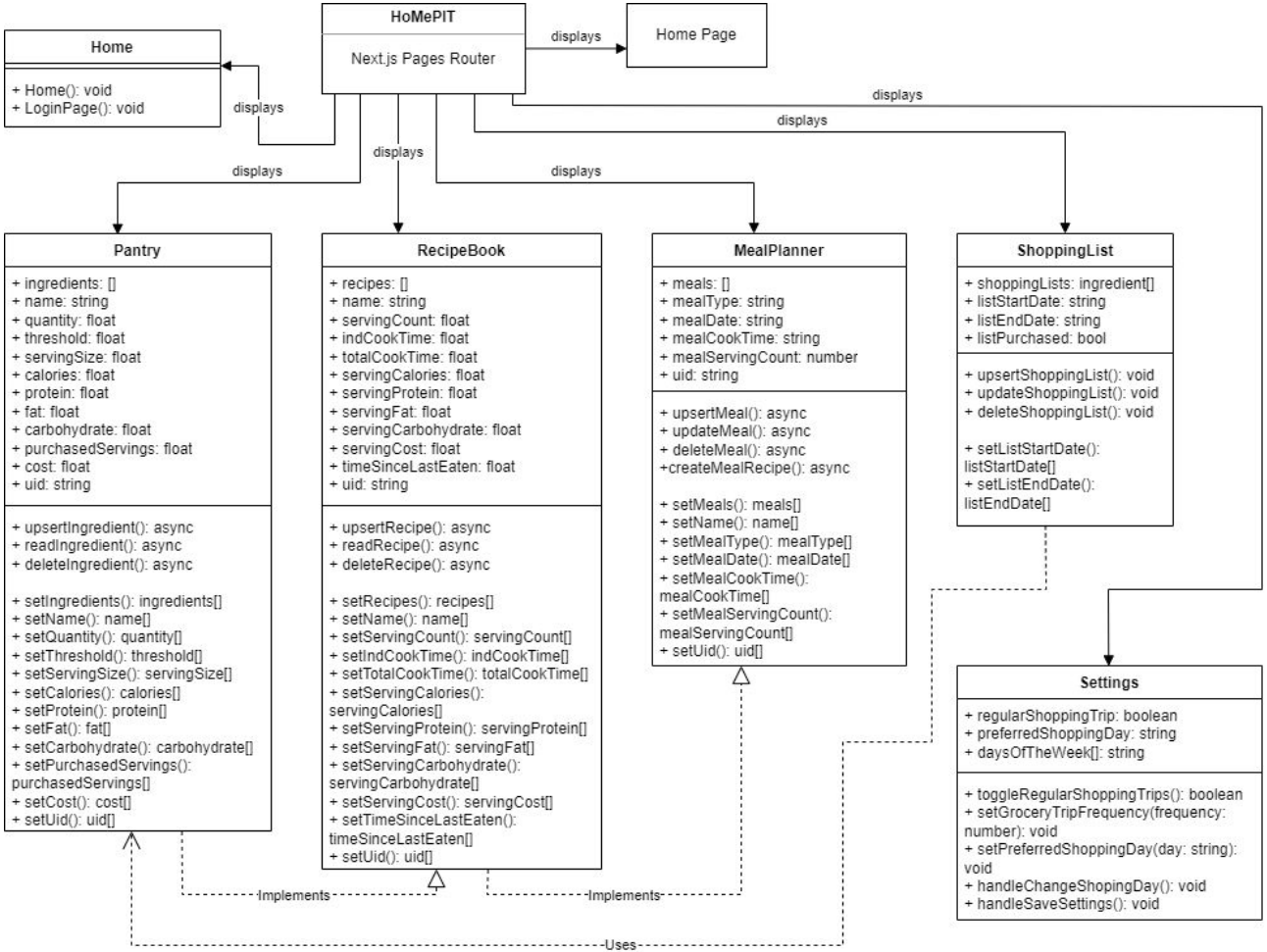
Some data duplication exists as artifacts from earlier development assumptions

- Has been left for compatibility

No null entries

- Exception: Ingredients Audit table
- Resolves null-value issues


Recipes
PK rec_id int
rec_serv_count int
rec_ind_cook_time int
rec_total_cook_time int
rec_serv_cal float
rec_serv_prot float
rec_serv_fat float
rec_serv_carb float
rec_serv_cost float
rec_time_since_eaten int
FK user_id uuid
Recipe Ingredients
PK, FK rec_id int
PK, FK ing_id int
ing_qnt float
ing_total_cal float
ing_total_prot float
ing_total_fat float
ing_total_carb float
ing_total_cost float
FK user_id uuid
Ingredients
PK ing_id int
ing_name text
ing_qnt float
ing_threshold_qnt float
ing_serv_qnt float
ing_serv_cal float
ing_serv_prot float
ing_serv_fat float
ing_serv_carb float
ing_purchase_serv_cnt float
ing_purchase_cost float
FK user_id uuid
Ingredients Audit
PK audit_id int
FK ing_id int
FK rec_id int
FK meal_id int
FK list_id int
ing_qnt float
audit_table regclass
audit_action enum
FK user_id uuid
Meal Recipes
PK, FK meal_id int
PK, FK rec_id int
meal_serv_to_cook int
FK user_id uuid
Meals
PK meal_id int
meal_type enum
meal_date_time timestamp
meal_time_start_cook timestamp
meal_serv_count int
FK user_id uuid
Shopping List Ingredients
PK,FK ing_id int
PK,FK list_id int
ing_qnt_to_buy float
FK user_id uuid
Shopping Lists
PK list_id int
list_start_date date
list_end_date date
FK user_id uuid


HoMePIT
Next.js Pages Router
displays
Home Page
Home
+ Home(): void
+ LoginPage(): void
displays
displays
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
+ uid: string
+ upsertIngredient(): 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[]
+ setUid(): uid[]
RecipeBook
+ recipes: []
+ name: string
+ servingCount: float
+ indCookTime: float
+ totalCookTime: float
+ servingCalories: float
+ servingProtein: float
+ servingFat: float
+ servingCarbohydrate: float
+ servingCost: float
+ timeSinceLastEaten: float
+ uid: string
+ upsertRecipe(): 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[]
+ setUid(): uid[]
MealPlanner
+ meals: []
+ mealType: string
+ mealDate: string
+ mealCookTime: string
+ mealServingCount: number
+ uid: string
+ upsertMeal(): async
+ updateMeal(): async
+ deleteMeal(): async
+createMealRecipe(): async
+ setMeals(): meals[]
+ setName(): name[]
+ setMealType(): mealType[]
+ setMealDate(): mealDate[]
+ setMealCookTime(): mealCookTime[]
+ setMealServingCount(): mealServingCount[]
+ setUid(): uid[]
ShoppingList
+ shoppingLists: ingredient[]
+ listStartDate: string
+ listEndDate: string
+ listPurchased: bool
+ upsertShoppingList(): void
+ updateShoppingList(): void
+ deleteShoppingList(): void
+ setListStartDate(): listStartDate[]
+ setListEndDate(): listEndDate[]
Settings
+ regularShoppingTrip: boolean
+ preferredShoppingDay: string
+ daysOfTheWeek[]: string
+ toggleRegularShoppingTrips(): boolean
+ setGroceryTripFrequency(frequency: number): void
+ setPreferredShoppingDay(day: string): void
+ handleChangeShopingDay(): void
+ handleSaveSettings(): void
Implements
Implements
Uses

Frontend Design

- Upon opening HoMePIT, the user will be presented with a Sign In prompt where they will be able to login with their google account.
- The top “App Bar” displays a menu icon, HoMePIT’s logo, and a settings icon.
  - Upon clicking the menu icon, the side navigation bar will appear.
  - Clicking the logo will take the user back to the Home page.
- The sidebar is how the user will navigate throughout HoMePIT, with tabs for Home, Pantry, Recipe Book, Meal Planner, Shopping List, and Settings.

Page Structure

- The Pantry Recipe Book, and Shopping list are structured very similarly, with each consisting of the following components:
  - A table displaying data from the given database table.
  - An “Add” button above the table to add to the database.
  - Options on each row to edit or delete the entry (or in the case of Shopping List, only delete).
  - A modal for adding and editing data that appears when the respective button is clicked.
- The Meal Planner consists of an “Add Meal” button, which when clicked opens a modal in the same manner as above for adding meals.
  - These meals appear in a column directly below this button.
- Directly to the left of this column is the Calendar component.

Code Structure

- All major pages have the same basic structure from top to bottom, those being:
  - Import statements and function declarations
  - State variables
  - Core logic
  - Return statement for visual display
- An upsert function is used for the creation and editing of data objects.
- All visual elements are defined in a functions return() statement, which is characteristic of Next.js projects.

Third Party Libraries

- Being a Next.js project, React libraries are heavily used throughout HoMePIT.
  - React state variables made storing data relatively easy.
- Supabase's auth-helpers-react library was used to create the Supabase Client for the project.
- In place of native HTML, Material UI was opted for in most instances.
  - This had the effect of making a nicer-looking user interface that suited our needs more
- Google Firebase was used to host the site, initially through the free Spark plan, however as HoMePIT grew an upgrade to the Blaze plan was necessary.

Known Bugs

Database

- Invalid shopping list insertions and updates return null instead of an error

Frontend

- Ingredient nutrition displays improperly
- Dragging meals to the calendar crashes
- Duplicate meals can be generated
- Ingredient and recipe editing doesn’t populate with existing information
- UI does not adapt sufficiently to screen resolution

Future Work

4 months of future work

Twice as much as was available

Increased familiarity with HoMePIT

Future Work

More shopping list insertions allowed

Allow shopping list date range updates

Additional shopping list ingredients updates

Purchase location functionality & optimization

Leftover functionality

Expiring ingredient functionality

Greater past data input functionality

Purchase location, leftovers, & ingredient expiry UI

Shopping list confirmation UI

Real time database subscription

Improved general UI & UX

Table sorting & search UI & UX

Tagging UI & UX

UI & UX for non-g/mL units

Better Android compatibility

Settings page functionality, UI, and UX