HoMePIT’s Design

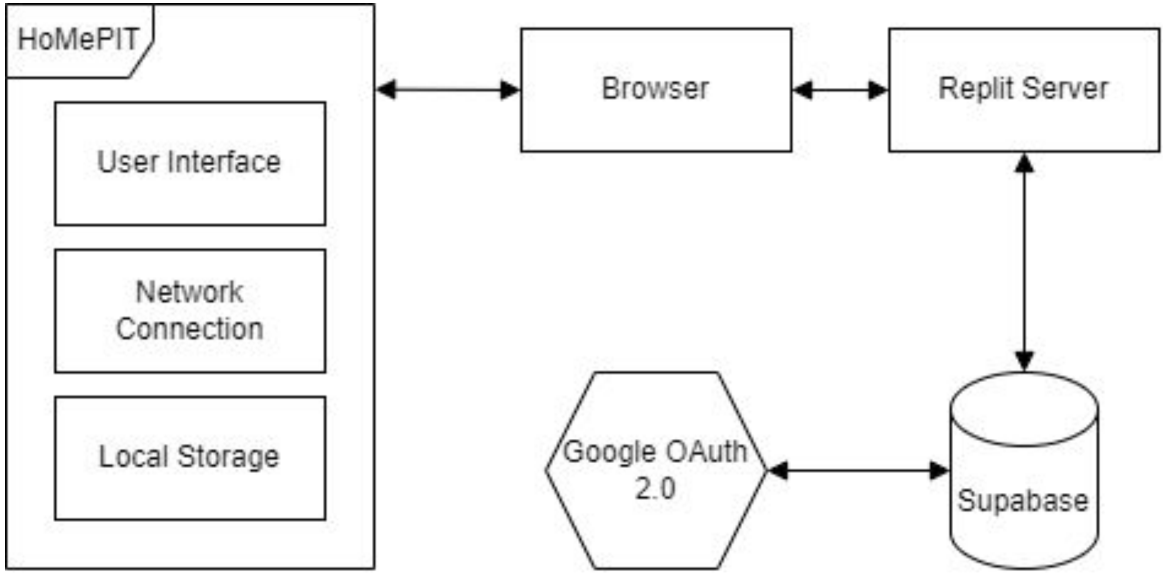
By Erik Anderson, Jerome Chestnut, and Maxwell Fugette

Introduction

- HoMePIT's purpose
  - HoMePIT will assist with planning a household's menu and tracking its supply, usage, and replacement of ingredients and other foodstuffs.
- Outline
  - Block Diagram
  - Component Design
  - Database Design
  - UI Design

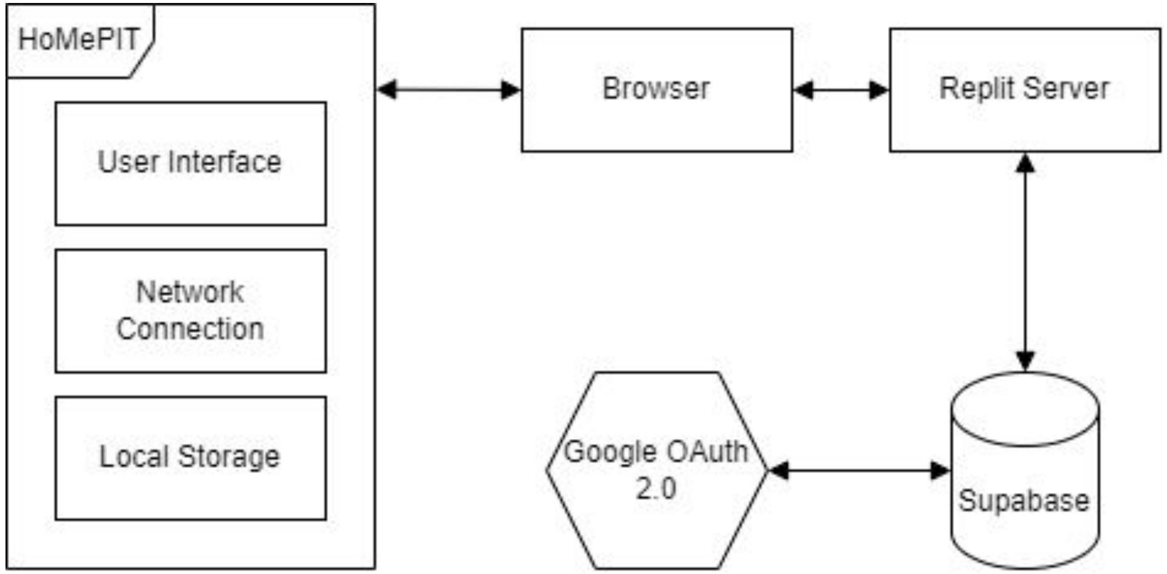
Block Diagram

- At the core is HoMePIT, orchestrating user interactions.
- It interfaces with three key sub-components: Local Storage, Network Connection, and User Interface.

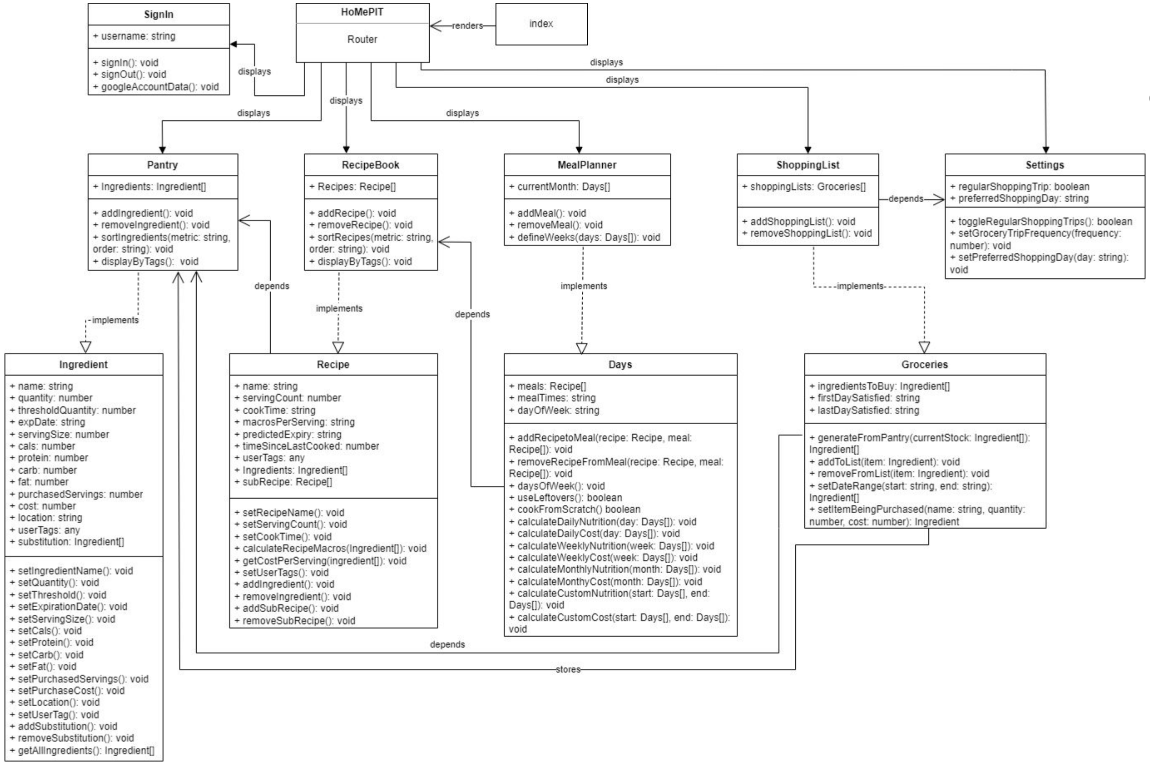


Block Diagram(continued)

- HoMePIT is viewed on a web browser, connecting it to the Replit Server.
- The Replit Server handles various backend operations and communicates with the Database.
- Data from the Supabase Database is securely connected to the Google OAuth 2.0 component.



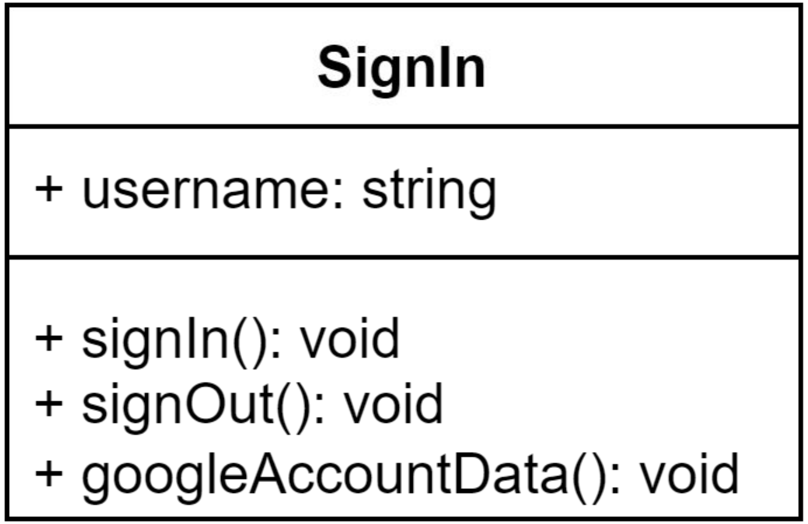
Components: Overview


SignIn
+ username: string
+ signIn(): void
+ signOut(): void
+ googleAccountData(): void
HoMePIT
Router
renders
index
displays
displays
displays
displays
displays
displays
displays
Pantry
+ Ingredients: Ingredient[]
+ addIngredient(): void
+ removeIngredient(): void
+ sortIngredients(metric: string, order: string): void
+ displayByTags(): void
RecipeBook
+ Recipes: Recipe[]
+ addRecipe(): void
+ removeRecipe(): void
+ sortRecipes(metric: string, order: string): void
+ displayByTags(): void
MealPlanner
+ currentMonth: Days[]
+ addMeal(): void
+ removeMeal(): void
+ defineWeeks(days: Days[]): void
ShoppingList
+ shoppingLists: Groceries[]
+ addShoppingList(): void
+ removeShoppingList(): void
depends
Settings
+ regularShoppingTrip: boolean
+ preferredShoppingDay: string
+ toggleRegularShoppingTrips(): boolean
+ setGroceryTripFrequency(frequency: number): void
+ setPreferredShoppingDay(day: string): void
implements
depends
implements
depends
implements
implements
Ingredient
+ name: string
+ quantity: number
+ thresholdQuantity: number
+ expDate: string
+ servingSize: number
+ cals: number
+ protein: number
+ carb: number
+ fat: number
+ purchasedServings: number
+ cost: number
+ location: string
+ userTags: any
+ substitution: Ingredient[]
+ setIngredientName(): void
+ setQuantity(): void
+ setThreshold(): void
+ setExpirationDate(): void
+ setServingSize(): void
+ setCals(): void
+ setProtein(): void
+ setCarb(): void
+ setFat(): void
+ setPurchasedServings(): void
+ setPurchaseCost(): void
+ setLocation(): void
+ setUserTag(): void
+ addSubstitution(): void
+ removeSubstitution(): void
+ getAllIngredients(): Ingredient[]
Recipe
+ name: string
+ servingCount: number
+ cookTime: string
+ macrosPerServing: string
+ predictedExpiry: string
+ timeSinceLastCooked: number
+ userTags: any
+ Ingredients: Ingredient[]
+ subRecipe: Recipe[]
+ setRecipeName(): void
+ setServingCount(): void
+ setCookTime(): void
+ calculateRecipeMacros(Ingredient[]): void
+ getCostPerServing(ingredient[]): void
+ setUserTags(): void
+ addIngredient(): void
+ removeIngredient(): void
+ addSubRecipe(): void
+ removeSubRecipe(): void
Days
+ meals: Recipe[]
+ mealTimes: string
+ dayOfWeek: string
+ addRecipetoMeal(recipe: Recipe, meal: Recipe[]): void
+ removeRecipeFromMeal(recipe: Recipe, meal: Recipe[]): void
+ daysOfWeek(): void
+ useLeftovers(): boolean
+ cookFromScratch() boolean
+ calculateDailyNutrition(day: Days[]): void
+ calculateDailyCost(day: Days[]): void
+ calculateWeeklyNutrition(week: Days[]): void
+ calculateWeeklyCost(week: Days[]): void
+ calculateMonthlyNutrition(month: Days[]): void
+ calculateMonthyCost(month: Days[]): void
+ calculateCustomNutrition(start: Days[], end: Days[]): void
+ calculateCustomCost(start: Days[], end: Days[]): void
Groceries
+ ingredientsToBuy: Ingredient[]
+ firstDaySatisfied: string
+ lastDaySatisfied: string
+ generateFromPantry(currentStock: Ingredient[]): Ingredient[]
+ addToList(item: Ingredient): void
+ removeFromList(item: Ingredient): void
+ setDateRange(start: string, end: string): Ingredient[]
+ setItemBeingPurchased(name: string, quantity: number, cost: number): Ingredient
depends
stores

- HoMePIT will consist of eleven main components, six of which represent the primary pages accessible by the user. The rest consist of dependent classes, and the HoMePIT Router.

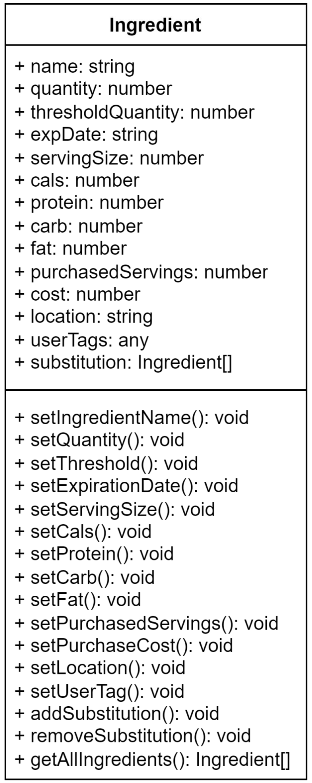
Components: SignIn

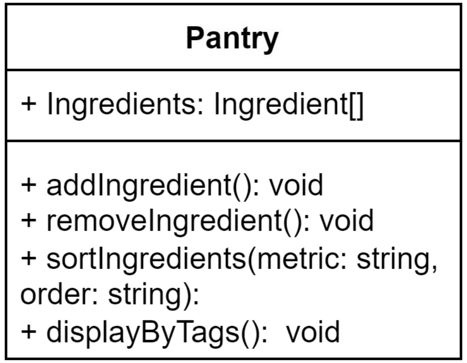
- The SignIn component handles everything related to Google authentication through Supabase. It will handle information such as the username, as well as methods for signing in and signing out.



Components: Pantry & Ingredient

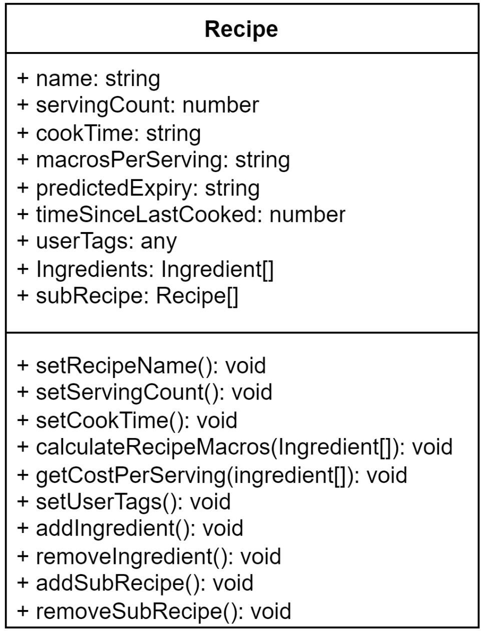
- The Pantry page will serve to define a list of Ingredient objects from the Ingredient class. From here, the Pantry will deal with the addition, removal, sorting, and display of Ingredient lists.
- The Ingredient class itself will define all information pertaining to individual ingredients.

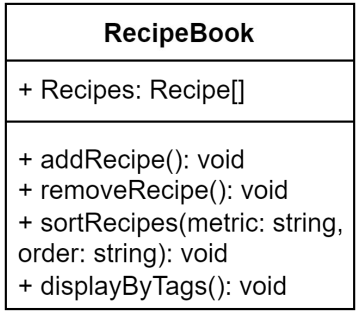



Pantry
+ Ingredients: Ingredient[]
+ addIngredient(): void
+ removeIngredient(): void
+ sortIngredients(metric: string, order: string):
+ displayByTags(): void

Components: RecipeBook and Recipe

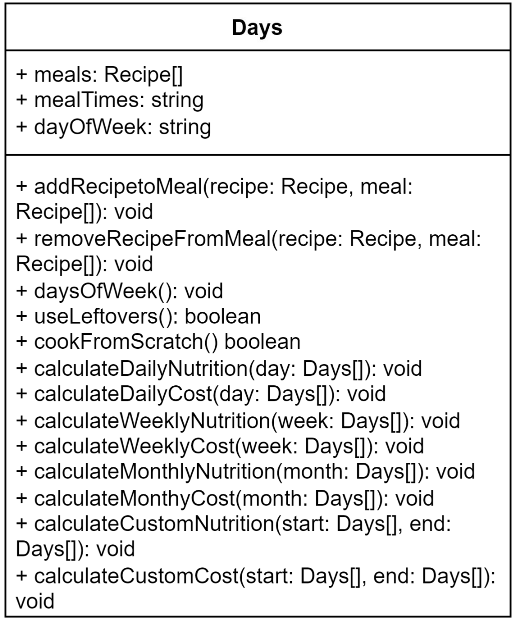
- The RecipeBook page will handle the creation of new Recipe lists, their removal, sorting, and display to the user.
- The Recipe class will define information pertinent to a given recipe, such as its name, nutritional info, and cost per serving, with all being pulled from each recipes Ingredient list.


Recipe
+ name: string
+ servingCount: number
+ cookTime: string
+ macrosPerServing: string
+ predictedExpiry: string
+ timeSinceLastCooked: number
+ userTags: any
+ Ingredients: Ingredient[]
+ subRecipe: Recipe[]
+ setRecipeName(): void
+ setServingCount(): void
+ setCookTime(): void
+ calculateRecipeMacros(Ingredient[]): void
+ getCostPerServing(ingredient[]): void
+ setUserTags(): void
+ addIngredient(): void
+ removeIngredient(): void
+ addSubRecipe(): void
+ removeSubRecipe(): void

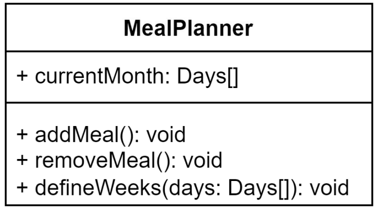


Components: MealPlanner & Days

- The MealPlanner page will consist of a list of Days in the form of a calendar.
- The Days class will define all meal and recipe information that a given day consists of, while also calculating nutrition and cost data from the individual days, weeks, month, or a custom date range.



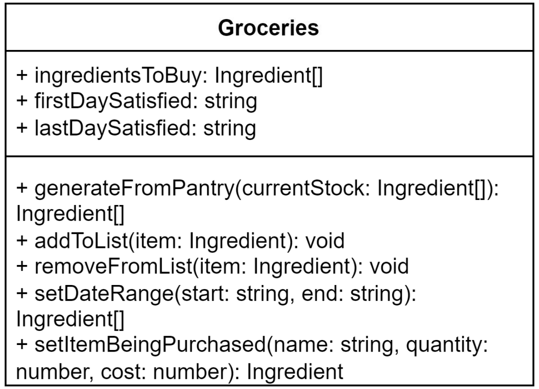
```
Days
----
+ meals: Recipe[]
+ mealTimes: string
+ dayOfWeek: string
----
+ addRecipetoMeal(recipe: Recipe, meal:
Recipe[]): void
+ removeRecipeFromMeal(recipe: Recipe, meal:
Recipe[]): void
+ daysOfWeek(): void
+ useLeftovers(): boolean
+ cookFromScratch() boolean
+ calculateDailyNutrition(day: Days[]): void
+ calculateDailyCost(day: Days[]): void
+ calculateWeeklyNutrition(week: Days[]): void
+ calculateWeeklyCost(week: Days[]): void
+ calculateMonthlyNutrition(month: Days[]): void
+ calculateMonthyCost(month: Days[]): void
+ calculateCustomNutrition(start: Days[], end:
Days[]): void
+ calculateCustomCost(start: Days[], end: Days[]):
void
```



```
MealPlanner
----
+ currentMonth: Days[]
----
+ addMeal(): void
+ removeMeal(): void
+ defineWeeks(days: Days[]): void
```

Components: ShoppingList and Groceries

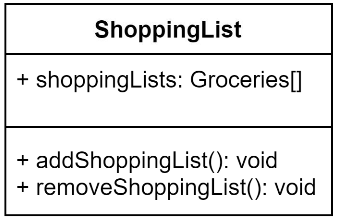
- The ShoppingList page will consist of a list of Groceries, while also handling their addition and removal.
- The Groceries class will define what ingredients the user needs to buy on their next shopping trip, as well as handling the date range satisfied by the single trip.



Groceries

+ ingredientsToBuy: Ingredient[]
+ firstDaySatisfied: string
+ lastDaySatisfied: string

+ generateFromPantry(currentStock: Ingredient[]): Ingredient[]
+ addToList(item: Ingredient): void
+ removeFromList(item: Ingredient): void
+ setDateRange(start: string, end: string): Ingredient[]
+ setItemBeingPurchased(name: string, quantity: number, cost: number): Ingredient



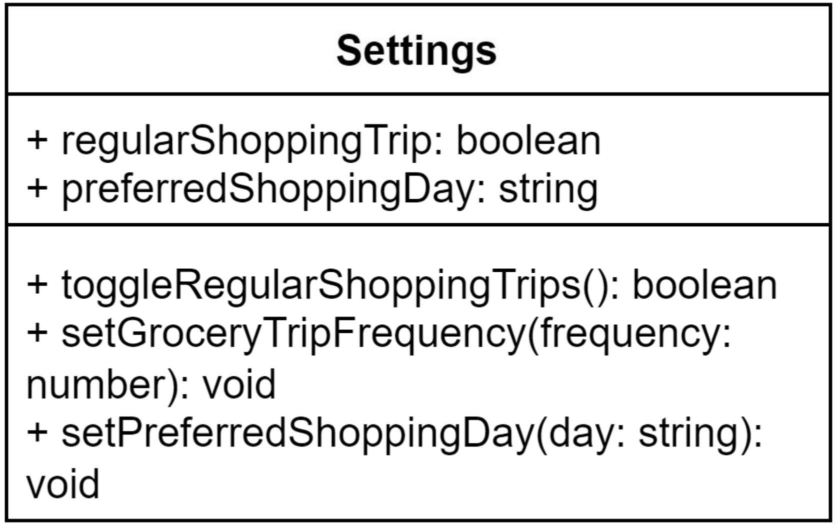
ShoppingList

+ shoppingLists: Groceries[]

+ addShoppingList(): void
+ removeShoppingList(): void

Components: Settings

- The Settings page will be the final page accessible by the user, and will primarily handle the settings regarding shopping list generation.
- This page will allow the user to set regular shopping trips, preferred days, and define frequency of said trips.

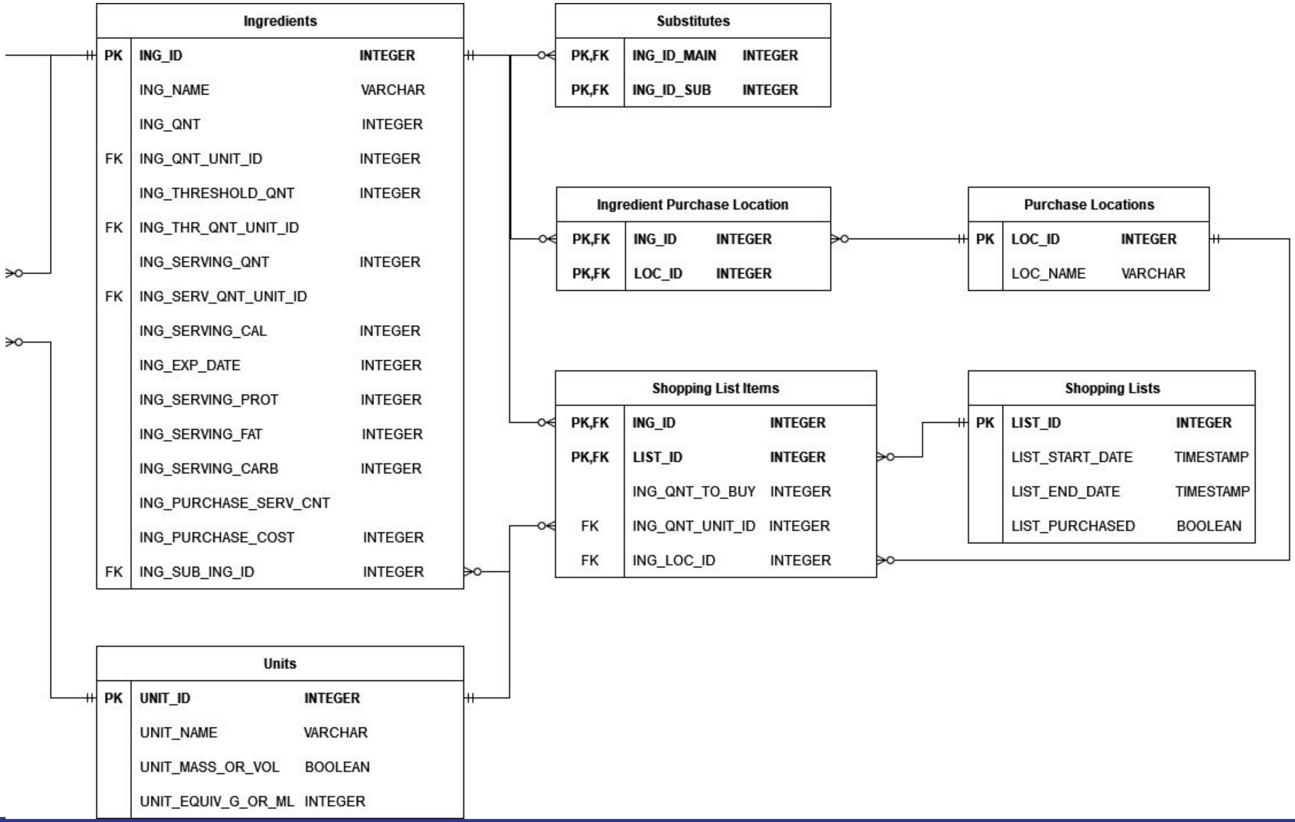


Settings

+ regularShoppingTrip: boolean
+ preferredShoppingDay: string

+ toggleRegularShoppingTrips(): boolean
+ setGroceryTripFrequency(frequency: number): void
+ setPreferredShoppingDay(day: string): void

Database Design



Ingredients		
PK	ING_ID	INTEGER
	ING_NAME	VARCHAR
	ING_QNT	INTEGER
FK	ING_QNT_UNIT_ID	INTEGER
	ING_THRESHOLD_QNT	INTEGER
FK	ING_THR_QNT_UNIT_ID	
	ING_SERVING_QNT	INTEGER
FK	ING_SERV_QNT_UNIT_ID	
	ING_SERVING_CAL	INTEGER
	ING_EXP_DATE	INTEGER
	ING_SERVING_PROT	INTEGER
	ING_SERVING_FAT	INTEGER
	ING_SERVING_CARB	INTEGER
	ING_PURCHASE_SERV_CNT	
	ING_PURCHASE_COST	INTEGER
FK	ING_SUB_ING_ID	INTEGER

Substitutes		
PK,FK	ING_ID_MAIN	INTEGER
PK,FK	ING_ID_SUB	INTEGER

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

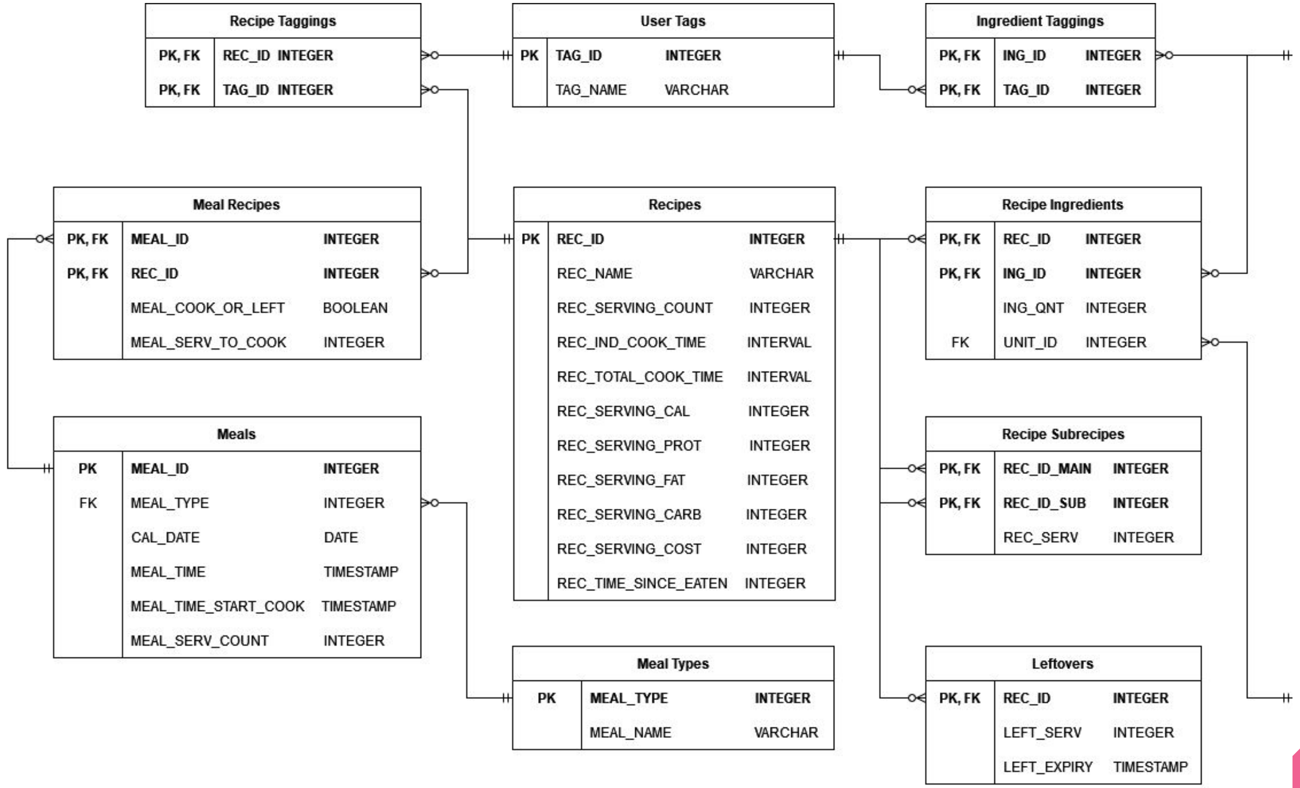
Purchase Locations		
PK	LOC_ID	INTEGER
	LOC_NAME	VARCHAR

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

Shopping Lists		
PK	LIST_ID	INTEGER
	LIST_START_DATE	TIMESTAMP
	LIST_END_DATE	TIMESTAMP
	LIST_PURCHASED	BOOLEAN

Units		
PK	UNIT_ID	INTEGER
	UNIT_NAME	VARCHAR
	UNIT_MASS_OR_VOL	BOOLEAN
	UNIT_EQUIV_G_OR_ML	INTEGER

Database Design



Recipe Taggings

PK, FK	REC_ID	INTEGER
PK, FK	TAG_ID	INTEGER

User Tags

PK	TAG_ID	INTEGER
	TAG_NAME	VARCHAR

Ingredient Taggings

PK, FK	ING_ID	INTEGER
PK, FK	TAG_ID	INTEGER

Meal Recipes

PK, FK	MEAL_ID	INTEGER
PK, FK	REC_ID	INTEGER
	MEAL_COOK_OR_LEFT	BOOLEAN
	MEAL_SERV_TO_COOK	INTEGER

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

Recipe Ingredients

PK, FK	REC_ID	INTEGER
PK, FK	ING_ID	INTEGER
	ING_QNT	INTEGER
FK	UNIT_ID	INTEGER

Meals

PK	MEAL_ID	INTEGER
FK	MEAL_TYPE	INTEGER
	CAL_DATE	DATE
	MEAL_TIME	TIMESTAMP
	MEAL_TIME_START_COOK	TIMESTAMP
	MEAL_SERV_COUNT	INTEGER

Recipe Subrecipes

PK, FK	REC_ID_MAIN	INTEGER
PK, FK	REC_ID_SUB	INTEGER
	REC_SERV	INTEGER

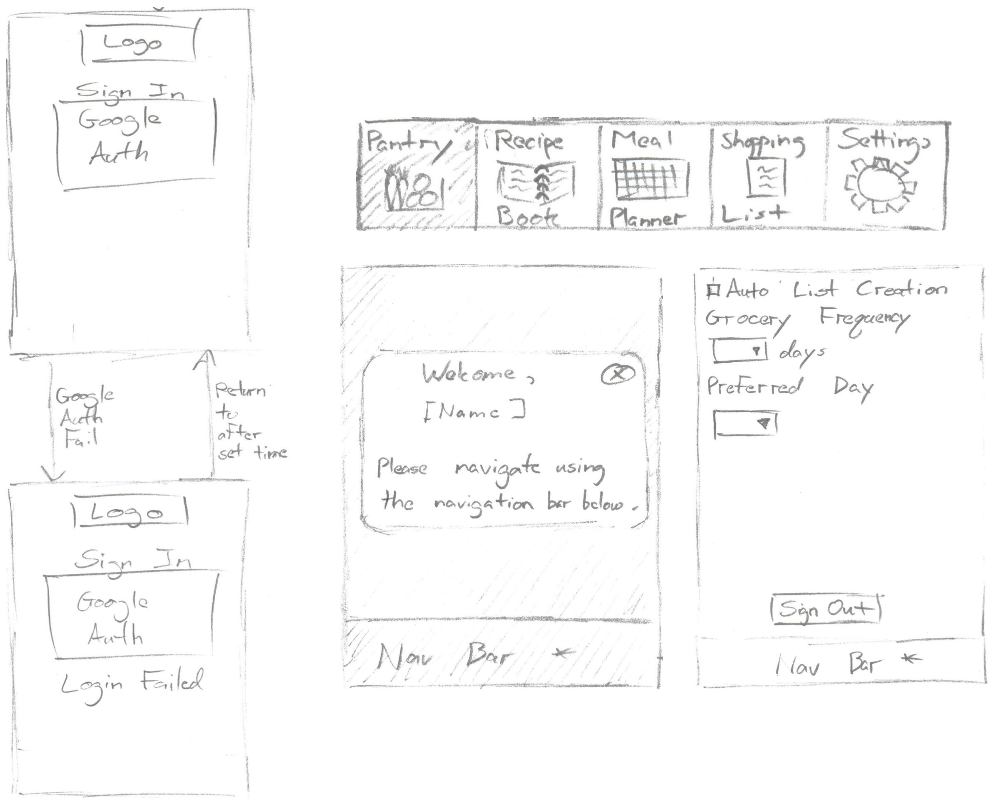
Meal Types

PK	MEAL_TYPE	INTEGER
	MEAL_NAME	VARCHAR

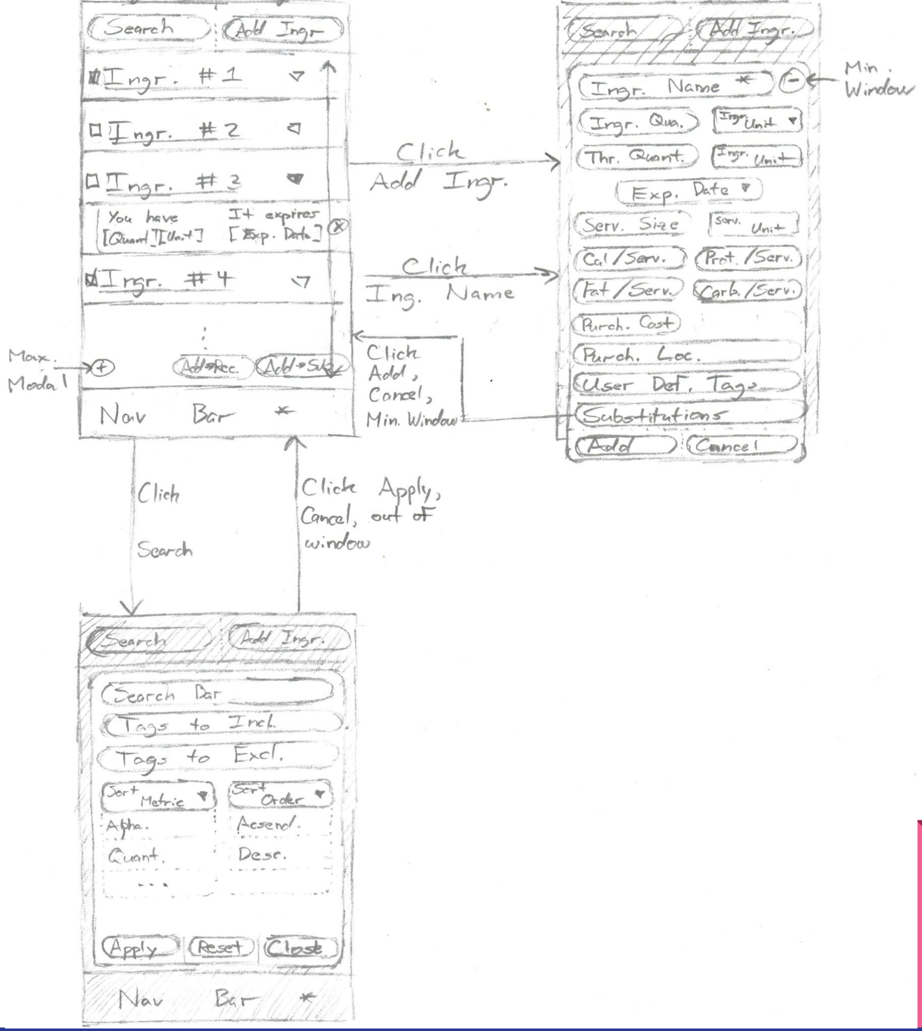
Leftovers

PK, FK	REC_ID	INTEGER
	LEFT_SERV	INTEGER
	LEFT_EXPIRY	TIMESTAMP

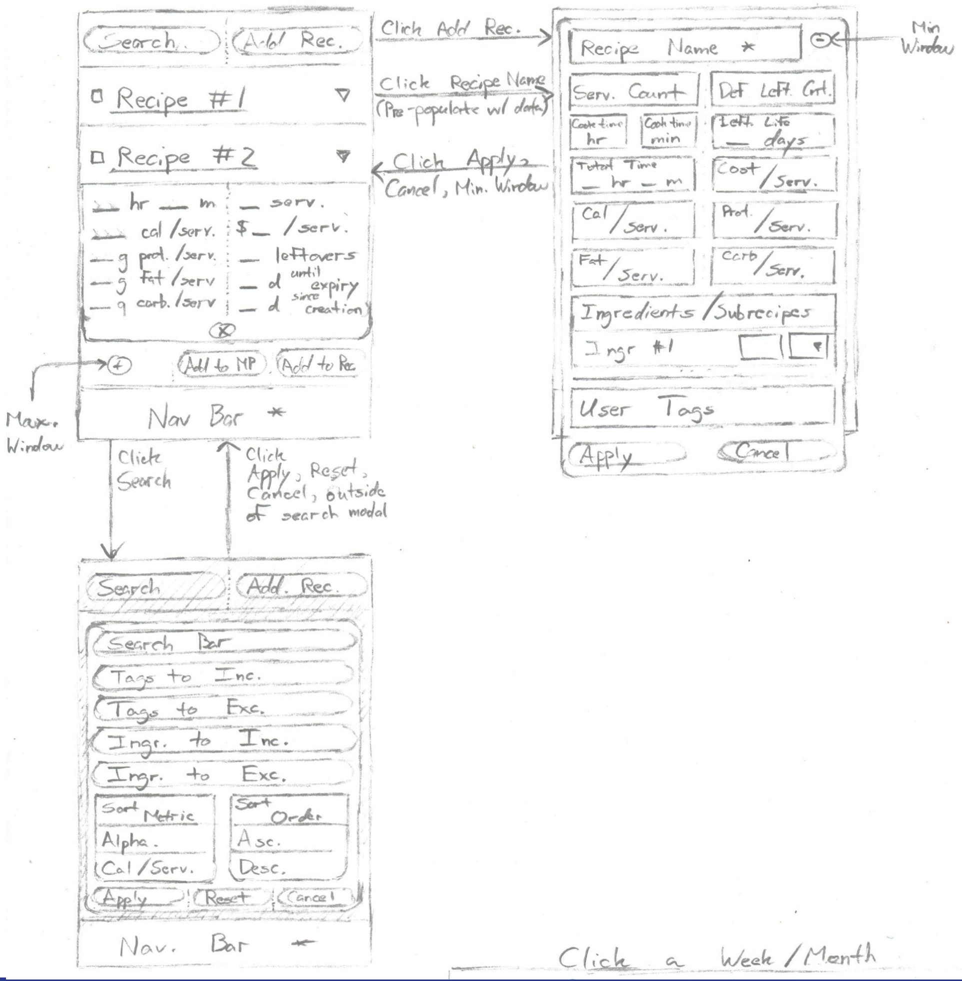
UI Storyboard: Sign in, Nav Bar, Settings



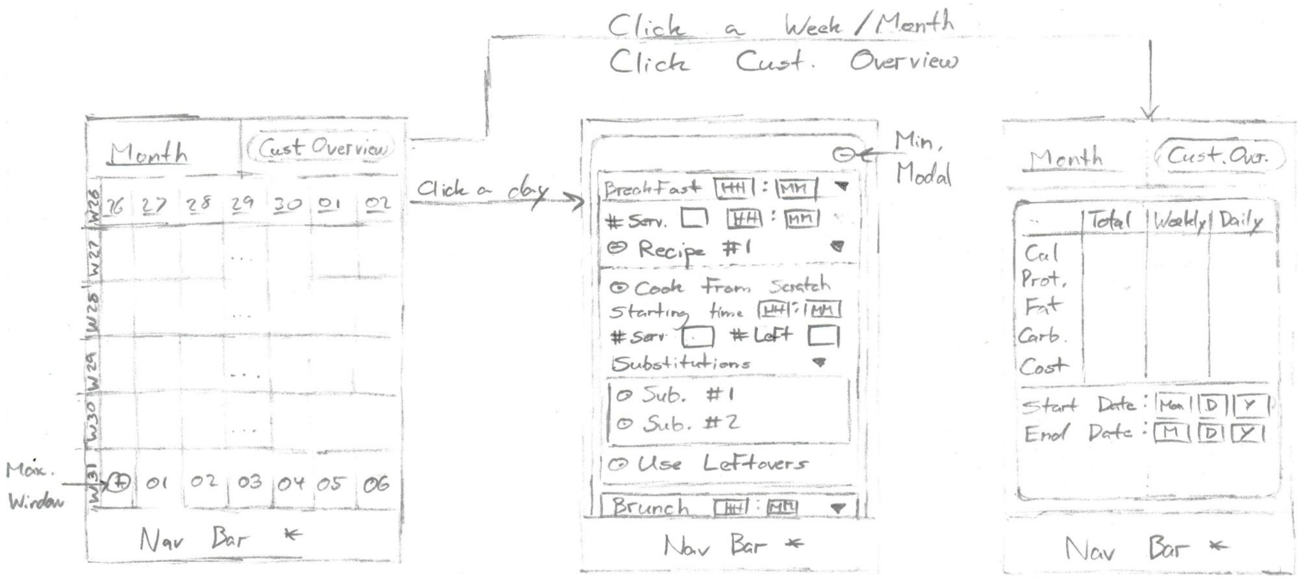
UI Storyboard: Pantry



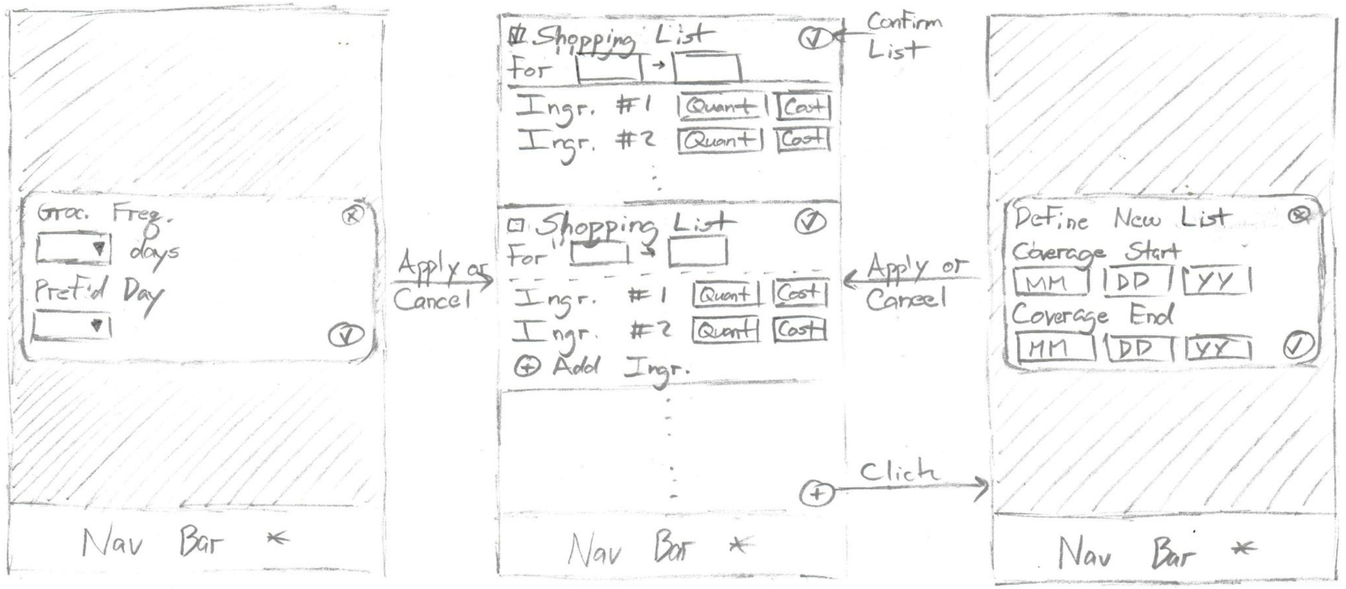
UI Storyboard: Recipe Book



UI Storyboard: Meal Planner



UI Storyboard: Shopping List



Conclusion

- What have we covered?
- Thanks for listening.
- Any questions?