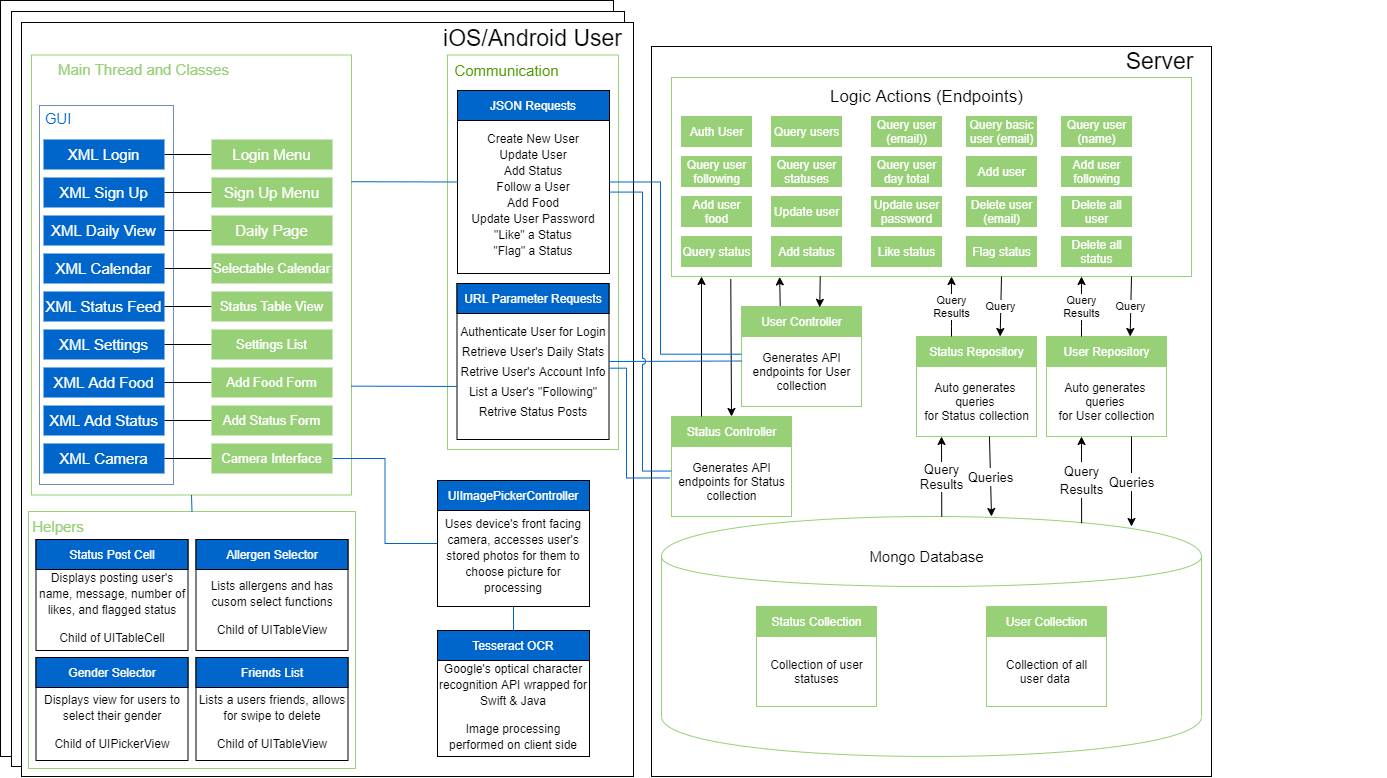
Block Diagram

HV\_3:

Sam Massey, Tyler Johnson, Max Van De Wille, Max Wilson

FoodBuddy



Design Descriptions

iOS/Android User

The iOS application uses a “Main” thread to execute all GUI changes/updates/etc. Each class has functions that alter the GUI in some form, and these functions are dispatched to the Main thread for these updates. In Android, these pages all have an individual XML file associated with them, whereas in iOS the XML file encompasses many pages or “View Controllers” in one storyboard file. The view controllers or activities contain the logic processing for accepting user button presses, text fields, and more and sends/receives information to the server via the established endpoints.

Helpers

Since we are developing two front-end applications, the helpers will look different for each iteration, but the core functionality is essentially the same. We have a helper for a status post, which allows for uniform distribution of the fields within a status object. This helper is used to display a list of all statuses posted by a following user. Another helper is the allergen selector, which lists all supported allergens and allows the user to choose which ones pertain to them. This is used when the user first creates an account, and later in the allergen settings also. Similar to the allergen list is the following list, which lists all users that the logged in user is following. This is found in the settings menu, where the user can add or remove other users as following. The last helper is the gender selector, which on iOS takes the form of a UIPickerView, which is a sort of scrolling selector with set options. On Android this takes the form of a dropdown menu. This is used on account creation and in the user’s account settings.

Communication

The created endpoints for our server are a mix of POST requests that require a JSON body and GET requests that only accept URL parameters. Almost every page in our application sends or receives data or both, since we are attempting to save as little as possible on the user’s device. For user passwords we are first encrypting with SHA512 before sending to the server.

Server-side Controllers

Our Mongo database prefers to use embedded tables, so many of our original table designs are embedded within users, meaning we need less controllers overall. Therefore our only two needed controllers are the User controller, which performs all actions involving user data, and the Status controller, which performs actions related to retrieving and posting statuses.

Database Field List

User Table:

email: String – Stores the user’s email

name: String – Stores the user’s full name

password: String – Stores the user’s SHA512 encrypted password

userType: String – Determines if the user is a moderator or admin or regular

height: Int – User’s height in inches

weight : Int – User’s weight in pounds

calorieLimit: Int – User’s daily calorie limit for tracking

age: Int – User’s age in years

gender: String – User’s selected gender

lifestyle: String – User’s lifestyle choice from Active, Moderate, Inactive

allergens: List<String> – List of what the user is allergic to

following: List<String> – List of which user’s emails the user is following

foods: List<Food> – List of food objects added by that user

name: String – Food’s name

calories: Int – Number of calories in that food

sodium: Int – Milligrams of sodium in that food

carbs: Int – Grams of carbohydrates in that food

protein: Int – Grams of protein in that food

fat: Int – Grams of fat in that food

cholesterol: Int – Milligrams of cholesterol in that food

amount: Int – How many servings the user consumed of that food

calendar: Map<String, Day> – List of days where the user added food

Status Table:

id: Long – A unique ID for that status

email: String – The email associated with the account that posted the status

timestamp: String – A timestamp of when the status was created

flagged: Boolean – Whether the status has been flagged as inappropriate

message: String – The user’s post or message displayed as the status

likes: Int – Number of total likes for that status