Group 8: Nicole Cao, Bowen Huang, Gina Jeon, Huyen Nguyen

Contributions:
Alpha Release - black
Beta Release - blue

Nicole (Release: 25%, Overall: 25%):

- I set up Firebase for meals collection, and then created the meal logging feature that is currently integrated with DeepSeek and Firebase. I set up a navigation interface with buttons for each meal category (Breakfast, Lunch, Dinner, Snack) that navigates the user to the meal log screen. In the meal log screen, there is an initial prompt to add a meal if no entries exist yet for the day. However, if there are existing meals logged under this Firebase user for today, then I fetch and display the food items for that category in a table view, sorted by entry time. The meal entries are generated programmatically using UITableViews, with each cell displaying the food name, quantity, and calories. When the user enters a food item, I format the input into a strict JSON prompt and call the DeepSeek API to get nutritional information. After processing the API response (around 10 seconds) and parsing the nutritional data, I upload the complete meal log to Firebase Firestore and refresh the UI. All user meal logs are stored in Firebase Firestore associated with each user.
- I set up a UICalendarView with custom styling and Auto Layout constraints in MealMentor, and implemented UICalendarViewDelegate for future customization.
    - LogDisplayViewController.swift
    - LogEntryViewController.swift
    - LogEntry.storyboard
    - CalendarPageViewController.swift
    - Meal.swift
    - Food.swift
- During this phase, I implemented a custom loading animation to indicate status during API processing. I made a circular spinner using Core Animation's CAShapeLayer, which activates when users submit food entries. The animation triggers on API requests, disabled user input during processing, and hides upon receiving API responses or errors, with main thread enforcement for UI updates.
- I have implemented a food journal for user to view past picture taken (currently hard coded picture, but for next phase I am going to try to save the photo into a CS server or firebase storage if it's free)
- To enable photo uploads as part of meal logging, I implemented a camera and photo library using UIImagePickerController. Users can tap the image view to launch the picker and either take a new photo or select one from their library. Once an image is chosen, it's displayed in the log entry page. I also tested the camera and uploaded photo functionality on my own device to make sure it works.
- Files:
    - LogEntryViewController
    - FoodGalleryViewController

    - LogEntry.storyboard
    - LoadingIndicatorView.swift

Bowen (Release: 25%, Overall: 25%):

- I set up the UI and functionality for the welcome, login, registration, onboarding, profile, and settings pages. I also set up the firebase authentication functionality for login/registration. For the login/registration, specific functionality includes creating an account and logging in. If the user doesn't log out, the application will automatically log in to their account and bring them to the home page. If the user has just created a new account, the registration page will segue directly to the onboarding pages(NOT to the login screen) where MealMentor will ask for specific information such as weight, gender, and height. The onboarding pages will also ask for user specific goals, allergies, etc.. The user can choose whether to answer any of the prompts in the onboarding pages. All this information will be stored in Firebase under the specific userid of the user's new account. The settings page is a page under the UITabController of the HomePage that will have functionality for changing the password, verbose chat responses in the chat page, and dark mode. The profile page is also a page under the UITabController of the HomePage that displays the user's personal information. The profile page also allows the user to log out and change their dietary restrictions, nutrition focuses, goals, and allergies.
    - Main.storyboard(login/registration)
    - LoginViewController.swift
    - OnboardingPage1ViewController.swift
    - OnboardingPage2ViewController.swift
    - ProfilePage.storyboard
    - ProfilePageViewController.swift
    - RegisterViewController.swift
    - SettingsPage.storyboard
    - SettingsViewController.swift
        - Empty Cocotouch swift file for now, will be useful in later phases
    - UserAccountInterface.swift
    - ChangePasswordViewController.swift
    - DietaryRestrictionsViewController.swift
    - WelcomePageViewController.swift
- This phase I mainly implemented the dark mode color theme to our application. I had to adjust some of the color settings in our various view controllers(all the view controllers in our app) and implement dark mode to be user account specific. I also adjusted some of the UI design in the onboarding and login/registration view controllers. Lastly, I adjusted the nutrition focus input to be category specific instead of a text field.
- Files:
    - Main.storyboard(login/registration)
    - LoginViewController.swift
    - OnboardingPage1ViewController.swift
    - OnboardingPage2ViewController.swift
    - ProfilePage.storyboard

    - ProfilePageViewController.swift
    - RegisterViewController.swift
    - SettingsPage.storyboard
    - SettingsViewController.swift
    - UserAccountInterface.swift
    - ChangePasswordViewController.swift
    - DietaryRestrictionsViewController.swift
    - WelcomePageViewController.swift
    - HomePage.storyboard
    - DarkModeViewController.swift
        - New viewcontroller that will be the parent for all viewcontrollers in order to implement dark mode

Gina (Release: 25%, Overall: 25%):

- I created the home page view controller and UI. This consists of a scrollable view, a weekly calendar collection view for displaying tracked days, a table view for displaying today's meals, and bar graph visualizations for nutrition statistics. In making the weekly calendar, I learned how to utilize Swift's Calendar and Date classes and also UICollectionView. In making the visualizations, I learned how to use the DGCharts and TinyConstraints frameworks.
- I created the visualizations page view controller and UI, which just consists of a segmented control (to navigate between nutrition categories) and a scrollable view for now.
- I created the tab bar controller and set up most of the navigation between our storyboards via segues and storyboard references (to log entry page, calendar page, settings page, profile page).
    - HomePage.storyboard
    - HomePageViewController.swift
    - VisualizationsPageViewController.swift
    - ThisWeekDayCell.swift
    - BarChartXAxisWeekdayValueFormatter.swift
    - BarChartYValueUnitValueFormatter.swift
- I wrote a fetch function that gets the total nutrition stats for the current day so that Huyen could use that information for the chat feature
- I improved the home page to make the UI structures populate with the user's real nutrition data. This includes:
    - Fetching all the meals for the current day to fill the "Today's Meals" table.
    - Fetching all the dates from the current week (Sun-Sat) that have trackes meals to fill the "This Week" weekly calendar (with the label that has the number of days #/7 tracked this week). Tracked days are filled in dark green, and untracked days are grayed out.
    - Fetching the number of dates from the current month that have tracked meals to fill the "# days logged this month" feature.
    - Fetching the streak of days starting from the current day that have trackes meals to fill the "# day streak" feature.

    - Fetching all the meals for each day of the current week, aggregating all of the nutrition facts from those meals, and using this data to fill the weekly nutrition graphs.
- Files:
    - HomePage.storyboard
    - HomePageViewController.swift
    - NutritionStats.swift

Huyen (Release: 25%, Overall: 25%):

- Alpha: I set up Firebase and Deepseek, and then created the AI chat feature that is currently integrated with Deepseek and Firebase. I set up the navigation bar to include a 'Chat' button that navigates the user to the chat page. In the chat page, there is an initial screen with sample prompts and an input bar. However, if there exists a chat history under this firebase user, then I don't show this screen and instead populate the chat with the chat messages in the last 24 hours in a scroll area, sorted by the time. The chat messages are created programmatically using a UIScrollView and a UIStackView. When the keyboard opens and closes, the chat bar will move up and down to fit the keyboard. The user can click the submit button or click return on the keyboard to chat to the AI. For every submission, I call the Deepseek API to produce a response. I also had to figure out how to store and hide the API key. I had to handle the response and upload this to firebase and populate it in the UI. I save all user chat history in Firebase firestore.
- Alpha:
    - ChatPage.storyboard
    - ChatPageViewController.swift
    - ChatBubble.swift
    - ChatMessage.swift
    - AppDelegate.swift (for firebase setup)
    - HomePage.storyboard (navigation to my storyboard)
- Beta: I improved the chat experience by auto scrolling down as you chat, expanding the chat bubble UI so the entire text doesn't get cut off anymore, and added prompt buttons at the start that you can choose from in addition to regular typing in the chat. Then I integrated the profile data, today's nutrition data, and their response length setting into the chat so it has catered answers. In the process I fixed a bug by chaining the asynchronous fetching from firebase data before calling deepseek to ensure most up to date logging data. Lastly, I implemented notification authorization and scheduled notifications throughout the day for breakfast, lunch, and dinner only if they haven't logged them yet for the day.
- Files:
    - ChatPage.storyboard
      ChatPageViewController.swift
    - SettingsPage.storyboard
    - SettingsPageViewController.swift
    - Profile.swift
    - ProfileLoader.swift
      ProfilePageViewController.swift

    - HomePageViewController.swift
    - Notifications.swift

Differences:

- We did not deviate from the original proposal in this phase. We have all of the features we proposed for both alpha and beta.