Course Professor: Dr. Johnson
Team Members: Bree Betts, Maya Dahlke, Adriana Donkers, Henno Kublin, Lexi Weingardt
CMSI 401
04 November 2020

Daily Bites Requirements Document

5.0 Requirements Specification

5.1 Introduction

Currently, Coco is a nutritionist application that is designed to streamline diet data entry and redefine what a nutrition app has to be for its users. It is currently being worked on by Dr. Korpusik, our mentor to the project, and has had many people contribute along the way. At this stage in its development, we are looking at adding a handful of new features and doing a design overhaul for the iOS app that is currently in the app store. Additionally, we hope to perform extensive data analytics within the natural language processing data, as well as migrate the entire backend over to AWS. Since our goal is a great ways away from the current form of Coco, we decided to dub the project Coco 2.0 (edit: we later renamed it to "Daily Bites"), as it seems fitting that our final product would be a new-and-improved sequel.

5.2. CSCI Component Breakdown

CSCI Daily Bites is composed of the following CSCs:

5.2.1 Cloud Infrastructure CSC
5.2.2.1 RDS CSU -- AWS cloud database service
5.2.2.1.1 PostgreSQL Database -- The RDS database runs on PostgreSQL
5.2.2.1.1.1 Users Entity -- table to store user information (id, email, diet)
5.2.2.1.1.2 Meals Entity -- store all meals entered by user
5.2.2.1.1.3 Nutrition Entity -- Table that will store the nutrition information for a user's meal
5.2.2.1.1.4 Food Item Entity -- store information on the food items in a user's meal
5.2.2.1.1.5 Recipes Entity -- store all recipes recommended to users
5.2.2.1.1.6 Sessions Entity -- store information on each session a user has within the app
5.2.2.2 EC2 CSU -- AWS cloud compute service for the NLP
5.2.2.3 S3 CSU -- S3 Bucket with the audio clips of users and food images

5.2.2 Backend CSC
5.2.2.1 Flask App CSU -- service for all backend functionalities
5.2.2.1.1 Chatbot module -- bot that detects intent from user and responds like a nutritionist would
5.2.2.1.2 Data Analytics module -- collect data that will be used for recommendations and actionable insights

5.2.2.1.3 Recipe Recommendations module -- algorithm that will match a user to a food recipe using Spoonacular API, based on diet preferences and past meals

5.2.3 iOS Development CSC
5.2.3.1 Client Dashboard CSU -- page displaying user trends
5.2.3.1.1 GUIButton module -- day button to see tends throughout the day
5.2.3.1.2 GUIButton module -- week button to see tends throughout the week
5.2.3.1.3 GUIButton module -- month button to see tends throughout the month
5.2.3.1.4 Chart module -- breaks down Carbs, Fat, Protein eaten in a day/week/month into percentages
5.2.3.2 Client Chatbox CSU -- page displaying the chatbox
5.2.3.2.1 TextField module -- to type what a user has eaten
5.2.3.2.1 GUIButton module -- send button to send the text to the chat
5.2.3.2.2 GUIButton module -- barcode button to take a picture of a barcode
5.2.3.2.3 GUIButton module -- microphone button to listen and input a user's speech into the text field
5.2.3.2.4 GUIButton module -- camera button for food imaging
5.2.3.2.5 GUIButton module -- info button that tells more about the application
5.2.3.2.6 Label module -- to display how many calories a user has eaten
5.2.3.2.7 Label module -- to display the date and time a user ate
5.2.3.3 Client Recipe Recommendations CSU -- pages displaying a quiz for first time users, at most 5 recommended recipes for a user, information about a recipe, and saved recipes
5.2.3.3.1 Client Recipe Recommendations Quiz CSU -- page displaying a quiz for first time users
5.2.3.3.1.1 Label module -- to display a question
5.2.3.3.1.2 GUIButton module -- multiple buttons for each answer
5.2.3.3.1.3 GUIButton module -- done button when the user is done with the quiz
5.2.3.3.2 Client User Recipe Recommendations CSU -- a page displaying at most 5 recommended recipes for a user
5.2.3.3.2.1 GUIButton module -- get new recommendations button to get at most 5 new recommended recipes
5.2.3.3.2.2 TableView module -- table of recipe recommendations
5.2.3.3.2.2.1 Image module -- to display a picture of the recipe
5.2.3.3.2.2.2 Label module -- to display the title of the recipe
5.2.3.3.2.2.3 Label module -- to display the number of kcals of the recipe
5.2.3.3.2.2.4 Label module -- to display the meal type of the recipe
5.2.3.3.2.2.5 GUIButton -- heart button to save a liked recipe and to help improve the recommendation algorithm

5.2.3.3.3 Client Recipe Recommendation Information CSU -- a page that displays information about a recommended recipe
5.2.3.3.3.1 Image module -- to display a picture of the recipe
5.2.3.3.3.2 Label module -- to display the title of the recipe
5.2.3.3.3.3 Label module -- to display the meal type of the recipe
5.2.3.3.3.4 Label module -- to display the cuisine type of the recipe
5.2.3.3.3.5 Label module -- to display the nutrition information of the recipe
5.2.3.3.3.6 Label module -- to display the ingredients and instructions for the recipe
5.2.3.3.4 Client Saved Recipes CSU -- page displaying a list of saved recipes
5.2.3.3.4.1 TableView module -- table of liked recipe recommendations
5.2.3.3.4.1.1 Image module -- to display a picture of the recipe
5.2.3.3.4.1.2 Label module -- to display the title of the recipe
5.2.3.3.4.1.3 Label module -- to display the number of kcals of the recipe
5.2.3.3.4.1.4 Label module -- to display the meal type of the recipe
5.2.3.3.4.1.5 GUIButton -- trash button to unsave a recipe
5.2.3.4 Client Settings CSU -- page showing client's name and nutrition preferences
5.2.3.4.1 TextField module -- to enter the user's name
5.2.3.4.2 TextField module -- to enter the user's nickname
5.2.3.4.3 Reorderable TableView module -- to enter nutrition preferences
5.2.3.4.4 Picker module -- to choose daily calorie goal
5.2.3.4.5 GUIButton module -- to logout

5.3 Functional Requirements by CSC

5.3.1 Backend CSC
5.3.1.1 Flask App CSU
5.3.1.1.1 Chatbot module
5.3.1.1.1.1 Chatbot shall allow users to log their food items for a day.
5.3.1.1.1.2 Chatbot shall allow users to ask simple questions (i.e. about nutrition facts, such as "How much protein is in beef?") and shall respond appropriately to such questions.
5.3.1.1.1.3 Chatbot shall have a fallback mechanism in the case that it does not identify the user's input and specify which questions/input it does respond to.
5.3.1.1.1.4 Chatbot shall detect the user's intent (e.g., logging a new meal, logging an exercise, asking a question, correcting the system, greeting the system, etc.).
5.3.1.1.1.5 Chatbot shall ask follow up questions to the user if the question is unclear.
5.3.1.1.1.6 Chatbot shall allow users to log their exercise for the day.
5.3.1.1.1.7 Chatbot shall save user data to the database based on the conversation (e.g. user's preferred language, dietary restrictions).
5.3.1.1.2 Data Analytics module

5.3.1.1.2.1 Data Analytics shall use data stored in the RDS database.
5.3.1.1.2.2 Data Analytics shall make recommendations to a specific user.
5.3.1.1.2.3 Data Analytics shall make recommendations to a user based on the meal's logged by the user.
5.3.1.1.2.3.1 Data Analytics shall analyze the vitamins contained in a meal logged by a user.
5.3.1.1.2.3.2 Data Analytics shall analyze the cumulative vitamins contained in the meals logged by a user.
5.3.1.1.2.3.3 Data Analytics shall analyze the vitamins missing from the cumulative vitamins in a user's meals.
5.3.1.1.2.3.4 Data Analytics shall provide the missing vitamins data to the recipe recommender algorithm.
5.3.1.1.2.4 Data Analytics shall observe user's personalized goals (this will be determined in the Recipe Recommender quiz and stored in RDS)
5.3.1.1.2.4.1 Data Analytics shall detect if a user reaches one of their goals.
5.3.1.1.2.4.2 Data Analytics shall congratulate the user for reaching their goals.
5.3.1.1.2.4.3 Data Analytics shall keep track of how many times a user reaches their goals.
5.3.1.1.2.4.4 Data Analytics shall use this information to suggest adjustments to the user's goals.
5.3.1.1.2.5 Data Analytics shall look at each food item in a user's meal.
5.3.1.1.2.6 Data Analytics shall determine the calories in each food item in a user's meal.
5.3.1.1.2.7 Data Analytics shall store the calories in each food item in the RDS database.
5.3.1.1.2.8 Data Analytics shall calculate the cumulative calories consumed by the user.
5.3.1.1.2.9 Data Analytics shall analyze the cumulative calories consumed by the user.
5.3.1.1.2.10 Data Analytics shall look for a correlation between eating certain foods and exceeding calorie goals.
5.3.1.1.2.11 Data Analytics shall determine which days the user tends to consume more calories.
5.3.1.1.2.12 Data Analytics shall display this information on the Dashboard Page.
5.3.1.1.2.12 Data Analytics shall keep track of each session.
5.3.1.1.2.12.1 Data Analytics shall keep track of a user's clicks within the app.
5.3.1.1.2.12.2 Data Analytics shall determine where in the app a user terminates their session.
5.3.1.1.3 Recipe Recommendations module
5.3.1.1.3.1 Recipe Recommendations shall include an algorithm, in Python, to match a user to a recipe based on diet preference and past meals
5.3.1.1.3.1.2 Recipe Recommendations shall use Spoonacular as an API to implement the algorithm

5.3.2 iOS Development
5.3.2.1 Client Dashboard
5.3.2.1.1 Pie chart shall display information about total calories eaten and percentages of fat, carbs, and proteins in the day, week, or month previous.

5.3.2.1.2 Chart displaying percentages of other vitamins, sugars, etcetera consumed in the day, week, or month previous.
5.3.2.1.3 Page shall contain a menu bar which allows users to navigate to the chat page, the recipes page, or the settings page.
5.3.2.2 Client Chatbox
5.3.2.2.1 Chatbox shall allow users to input food items they have consumed.
5.3.2.2.2 Chatbox shall allow users to ask for recipe recommendations.
5.3.2.2.3 Chatbox shall contain an info button which tells users more about the app.
5.3.2.2.4 Chatbox shall tell users the time and date.
5.3.2.2.5 Chatbox shall tell users how many calories they have consumed and how many they have left.
5.3.2.2.6 Chatbox shall allow users to input their food items through voice recognition by clicking the microphone button.
5.3.2.2.7 Chatbox shall allow users to scan the barcode of their food items by clicking the barcode button.
5.3.2.2.8 Chatbox shall allow users to take a picture to log their food items by clicking on the camera button.
5.3.2.2.9 Chatbox shall allow users to send their food items using the send icon to the right of the textbox.
5.3.2.2.10 Chatbox shall allow users to see their messages and the chat agent's responses in the chat window.
5.3.2.2.11 Page shall contain a menu bar which allows users to navigate to the chat page, the recipes page, or the settings page.
5.3.2.3 Client Recipe Recommendations
5.3.2.3.1 Recipe Recommendations shall ask users to answer questions about their diet preferences and lifestyle.
5.3.2.3.2 Recipe Recommendations shall provide a table of at most five recipes based on diet preference or history of meals.
5.3.2.3.2.1 Recipe Recommendations shall depict an image, title, kilocalorie number, and meal type for each recipe listed.
5.3.2.3.2.2 Recipe Recommendations shall allow users to like a recipe, and, therefore, save the recipe to view later.
5.3.2.3.2.3 Recipe Recommendations shall contain a button for users to get at most five new recipes based on diet preference or history of meals.
5.3.2.3.2.4 Recipe Recommendations shall bring users to another page to view more information about a recipe when a tableViewCell is clicked.
5.3.2.3.2.4.1 Recipe Recommendations shall display an image, title, meal type, cuisine type of a recipe.
5.3.2.3.2.4.2 Recipe Recommendations shall also display nutrition information and provide ingredients and instructions of a recipe.

5.3.2.3.3 Recipe Recommendations shall provide a table of all recipes a user liked.
5.3.2.3.3.1 Recipe Recommendations shall allow a user to view each recipe in a similar way to what is stated above.
5.3.2.4 Client Settings
5.3.2.4.1 Textbox shall allow users to modify their name/nickname.
5.3.2.4.2 Drop down menu shall allow users to input their nutrition preferences.
5.3.2.4.3 Drop down menu shall allow users to input their daily calorie goal.
5.3.2.4.4 Log out button shall allow users to log out of the application.

5.4 Performance Requirements by CSC

5.4.1 Backend CSC
5.4.1.1 Flask App CSU
5.4.1.1.1 Chatbot module
5.4.1.1.1.1 Chatbot shall detect user's intent and respond in less than 5 seconds.
5.4.1.1.2 Data Analytics module
5.4.1.1.2.1 Data Analytics recommendations and insights returned in 10 seconds.
5.4.1.1.3 Recipe Recommendations module
5.4.1.1.3.1 Recipe Recommendations shall provide users with at most 5 recipe recommendations in less than 5 seconds.
5.4.1.1.3.2 Recipe Recommendations shall display information about a recipe or liked recipes in 3 seconds.

5.5 Project Environment Requirements

5.5.1 Development Environment Requirements

The following are the software requirements for Daily Bites:

Category	Requirement
Operating System	MacOS Catalina
Compiler	Swift
Interpreter	Python
Integrated Development Environment	Xcode Version 12.0 or higher

Cloud Computing Platform	Amazon Web Services
Micro Web Framework	Flask

5.5.2 Execution Environment Requirements

The following are the hardware requirements for Daily Bites:

Category	Requirement
Processor	GHz Intel Core i5 CPUs
Disk Storage	256 GB
Disk Space	8 GB
RAM	8 GB
Display	2560 x 1600, 256 colors

5.5.3 Deployment Environment Requirements

The following are the deployment requirements for Daily Bites:

Category	Requirement
Device	iPhone 10 or iPhone 11
Application	Daily Bites

Gantt chart:
https://docs.google.com/spreadsheets/d/12XPNBduizmIBxl9WhNN4iLDbLpAqgWoRF0UXSBD9SC8/edit?usp=sharing

6.0 Architecture Specification

6.1 Introduction

This document presents the architecture and detailed design for the software of the Daily Bites project (formally known as Coco 2.0). The project is a nutritionist application designed to streamline diet data entry and redefine what a nutrition app has to be for its users. The app performs data analytics, recipe recommendations to the user, and natural language understanding in a conversational chatbot. The app also will be on AWS, and its GUI will be through an iOS app.

6.1.1 System Objectives

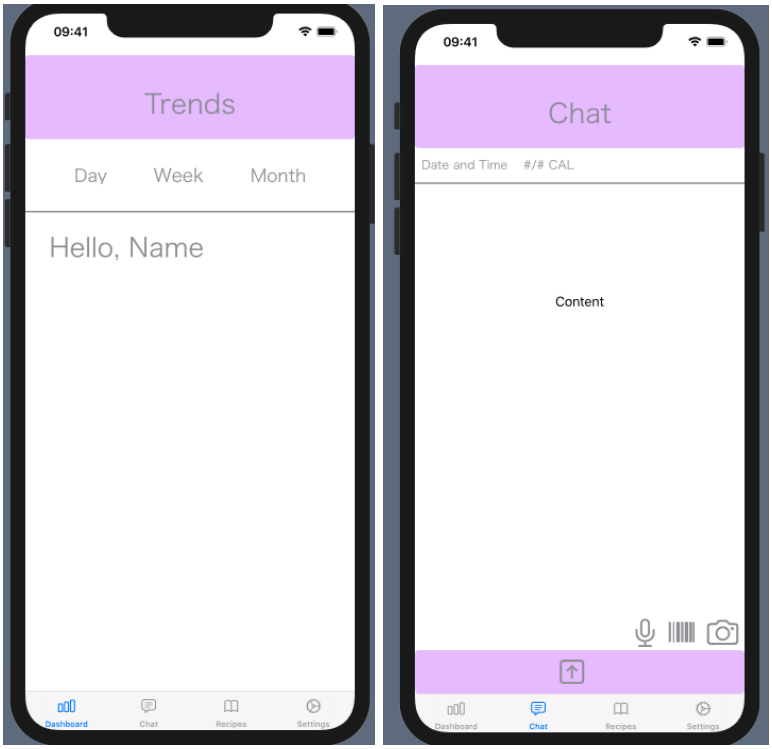
The objective of this application is to provide a new user interface, database, chatbot, and server for the existing Coco Nutrition application. This is done with the goal of making nutrition tracking simple and intuitive for users. Insights into diet patterns are provided to the user on the Dashboard page of the application. Users can save, search and view recipes recommended to them based on their diet patterns. Additionally, users can track their meals by having conversations with the chatbot.

6.1.2 Hardware, Software, and Human Interfaces

6.1.2.1 One interface will be the chatbot. Humans can interact with this interface as if it were an actual nutritionist via text or voice input. For voice input, they will press a button on the chatbot page. Users will be able to use the chatbot interface either with voice or by text to log their meals, ask basic questions about nutrition, change their personal settings such as dietary restrictions, and inquire for recipe recommendations. The user interface for this chatbot will be created through the iOS app.

6.1.2.2 One interface is the iOS application, which will be created using Xcode. Xcode is an IDE for iOS which contains software development tools created by Apple. The application will be used on iPhones with iOS 13 or higher. Users will be able to interact with this interface by navigating through the different pages of the application using the tab bar. They will also be able to interact with components on each page, including the recipes page, the chatbot, and the settings.

6.1.2.3 iOS Interface Screenshots



6.1.2.4 One interface will be the recipe recommendations tab. Humans can interact with this interface to get a list of up to five recipes, along with information about each recipe. Users will be able to click a button at the top of the screen labeled 'Get New Recommendations' to get a list of up to five new recipes at a time. The human can click the heart button next to the recipe to add it to their Saved List, and they can click on a recipe to get nutrition information, ingredients, and instructions for the recipe.

6.2 Architectural Design

6.2.1 Major Software Components

6.2.1.1 The Flask Application for Daily Bites will be hosted with an EC2 instance on AWS.

6.2.1.2 The Chatbot subsystem of the Daily Bites application will consist of a conversational AI agent using the Rasa open source machine learning framework.

6.2.1.3 The Recipe Recommendation subsystem of the Daily Bites application will use APIs created in the AWS Flask Service which will involve the use of the Spoonacular API created by a third party.

6.2.1.4 The data analytics portion of the Daily Bites application will query and store data in the RDS stored on AWS.

6.2.1.5 The Daily Bites application will be built using SwiftUI in Xcode. This application will be used on an iPhone which must have iOS 13 or higher.

6.2.2 Major Software Interactions

6.2.2.1 The iOS application will interact with the chatbot, the data analytics, and the user information. When the user logs in for the first time and enters their information, that information is stored in the RDS through an endpoint in the Flask Application.

6.2.2.2 The Flask Application will serve as the link between the iOS application and the RDS. A new user will log their information on the iOS interface, which will pass the information to the Flask Application endpoint on the server, which will then use a postgres insert statement to store the user in the database.

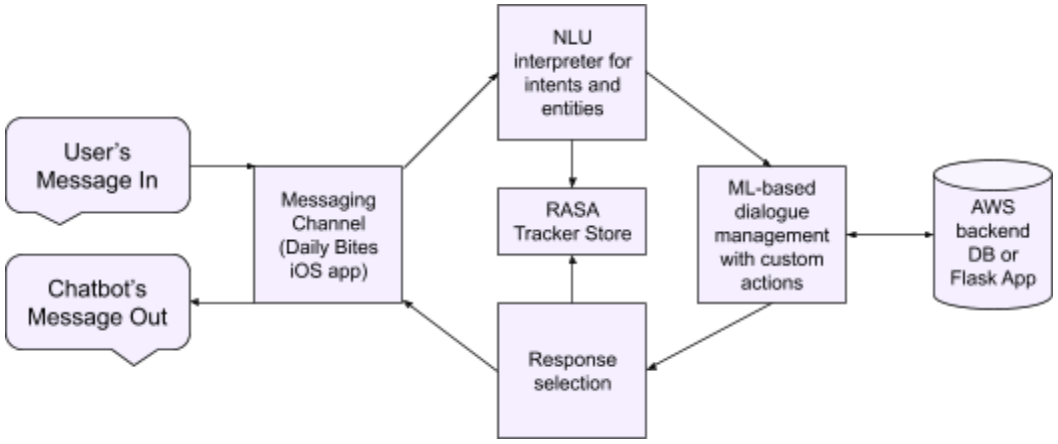
6.2.2.3 The chatbot will be integrated into the system by adding the chat components into the IOS portion of the application, and integrating some of the functions with the flask backend app. The chatbot will also make use of Rasa custom actions to make changes to the AWS database system depending on the input from the user. The Rasa custom actions may also make API calls to certain functions in the flask app. To explain the Rasa system in more depth: The iOS application will receive a text or voice message from the user, which will then be sent to the Rasa code that is in a separate application. The Rasa interpreter then receives that piece of input and uses natural language understanding (NLU) to interpret the English words and structure of the input and caches the message in the Rasa Tracker. The Rasa code then parses through this message to recognize specific intents (i.e. a greeting from the user; or a user logging a meal they just ate) and entities if applicable (i.e. the specific food item and quantity of that food). Based on these intents, the chatbot will take the appropriate action for that intent (i.e. saving a meal log to the AWS database; or a fallback mechanism if it doesn't recognize an intent; or a "goodbye" if the user says "bye"; etc.). The action that the chatbot takes is also saved into the Rasa Tracker Store before it takes the action and the user will receive a message from the chatbot through the iOS side in a conversational way within seconds.

6.2.2.4 The data analytics will be integrated into the system on the Dashboard page of the iOS application. When a user navigates to the Dashboard page, specific endpoints will be called on the Flask Application Server, which is hosted using an EC2 instance. Each endpoint will query data from the PostgreSQL RDS that is stored on AWS.

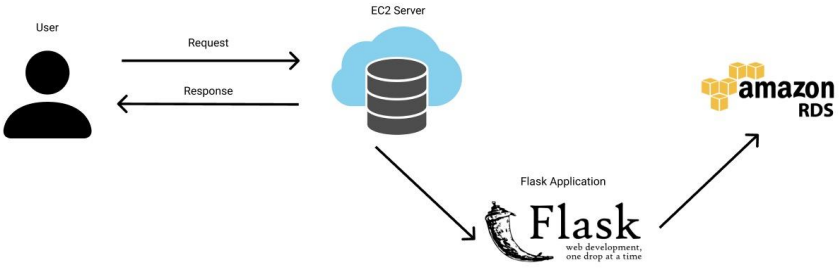
6.2.2.5 The iOS application will interact with the recipe recommendations. There will be APIs created in the AWS Flask service that will act as a link between the iOS application and the Spoonacular API. The APIs will get quiz and history information from the database to form an URL query to pass to the Spoonacular API. Once the JSON data is given back, the APIs will clean up the data, remove existing recipes, and pass an array of data back to the iOS application to process and display.

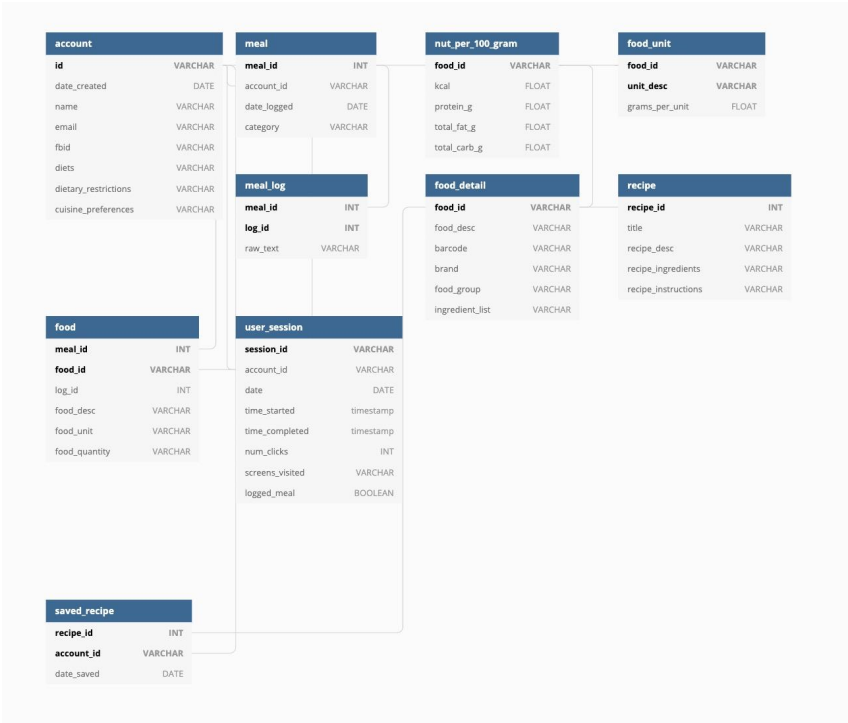
6.2.3 Architectural Design Diagrams

6.2.3.1 Chatbot Diagram

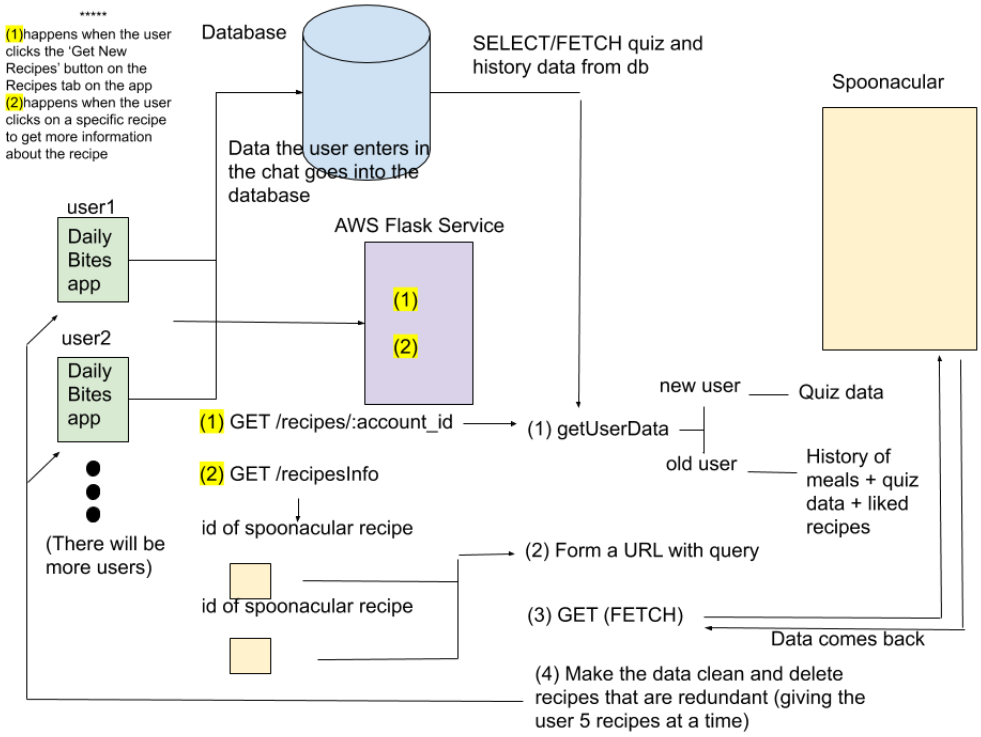


6.2.3.2 Data Analytics / Database Diagram & Schema





6.2.3.3 Recipe Recommendations Diagram



6.2.3.4 AWS Architecture Diagram