Design Document

CMPT 276

Group 5

App: Mealify

Members: John Zheng, Justin Yu,

Juey Lew, Vincent Yu, Feng Wu

Table of Contents

Guidelines:

- Development are done using Xcode 9 on MacOS
- Github will be used for version control and group collaboration
- Jira will be used for project management
- Database and user authentication will be implemented using Firebase
- Swift 4 must be used for implementing the model and controller
- Each model needs to be separated in a different file to increase modularity
- Relational database model will be used to accommodate the data retrieved from Canadian Nutrient File (CNF)
- Variable naming must follow letter case-separation word convention
- Database containing food nutrition should be updated regularly to obtain the most up to date information
- User data such as password will be encrypted in an external database
- Developers must comment in detail what they've update for every commit on github
- Each function implemented to the system must include comments/documentation to describe its functionality
- No user data will be sold to external companies

Data Requirements:

All user interaction will be done through the mobile device. No external hardware are needed besides the mobile device that contains the application. Users can interact with the application via touch screen such as swipe, tap, hold and drag. Other internal interaction will require using the hardware within the phone. Camera is required to input and output data to the application and GPS chip on the mobile device will be used to track user location.

All backend services will be implemented using firebase. Firebase will be in real time nosql to create more flexibility. The main schema used in the database will be username, password and email. Other schemas will be later implemented as the application progressively gets more complicated. Data interaction between front end and back end will only be read and write. Deleting data will be done manually for safety reasons. All entries in user related schema will be protected using encryption framework.

Non user data will be retrieve from other system such as CNF. There will be an API to collect this data. However, these data cannot be modified due to various reasons. We will communicate with any external APIs using HTTP requests and responses in JSON string.

Feature Priority:

Version 1:

1. Defining User's Eating Habits
2. Nutrients Meal Tracking
3. View User's Analyzed Data
4. User Login

Version 2:

1. Diet Plan Recommendations
2. Food Asset Map
3. Calendar Progression

Version 3:

1. Augmented Reality View of Meals
2. Nutrients Meal Tracking in Restaurants
3. Friend System
4. Syncing External Data from Apple's Healthkit API

Version #1

Objective: The purpose of this iteration is to implement the core features to create a functional application to meet client's requirement.

Features	**Estimated Effort**[1]
Defining User's Eating Habits	30 hrs
Nutrient Meal Tracking	50 hrs
View User's Analyzed Data	50 hrs
User Login	10 hrs
Totals:	140 hrs

Version 1 of the application will contain the most important features. User login will be implemented using Firebase Authentication. Users can login using their Email upon opening the application for the first time. User will then be greeted with a form to enter their basic information such as age, gender, food preferences and special diets. This is part of the Defining user's eating habits feature. This feature will be implemented using swift and MVC paradigm.

[1]

Nutrient meal tracking will be implemented using swift and MVC paradigm as well. This will track what food the user has consumed and convert the following information into empirical data that can be analyzed using USDA API. Viewing user's analyzed data will retrieve data from both user and USDA API.

Version #2

Objective: The purpose of this iteration is to add more features and improve user experience.

Features	**Estimated Effort**[2]
Diet Plan Recommendation	30 hrs
Food Asset Map	30 hrs
Calendar Progression	30 hrs
Totals:	90 hrs

Version 2 of the application will provide extension to the core features of version 1. These features are also important and are implemented to further enhance user experience. Diet plan recommendation will be implemented using various external APIs. Food asset map will also be implemented using Apple maps, Zomato API. Calendar progression will only show user progression in the second version of this application. This feature will display user's diet plan, progress and goals. All of this will be implemented locally in the application using swift.

Version #3

Objective: The purpose of this iteration is to create a finish product

[2]

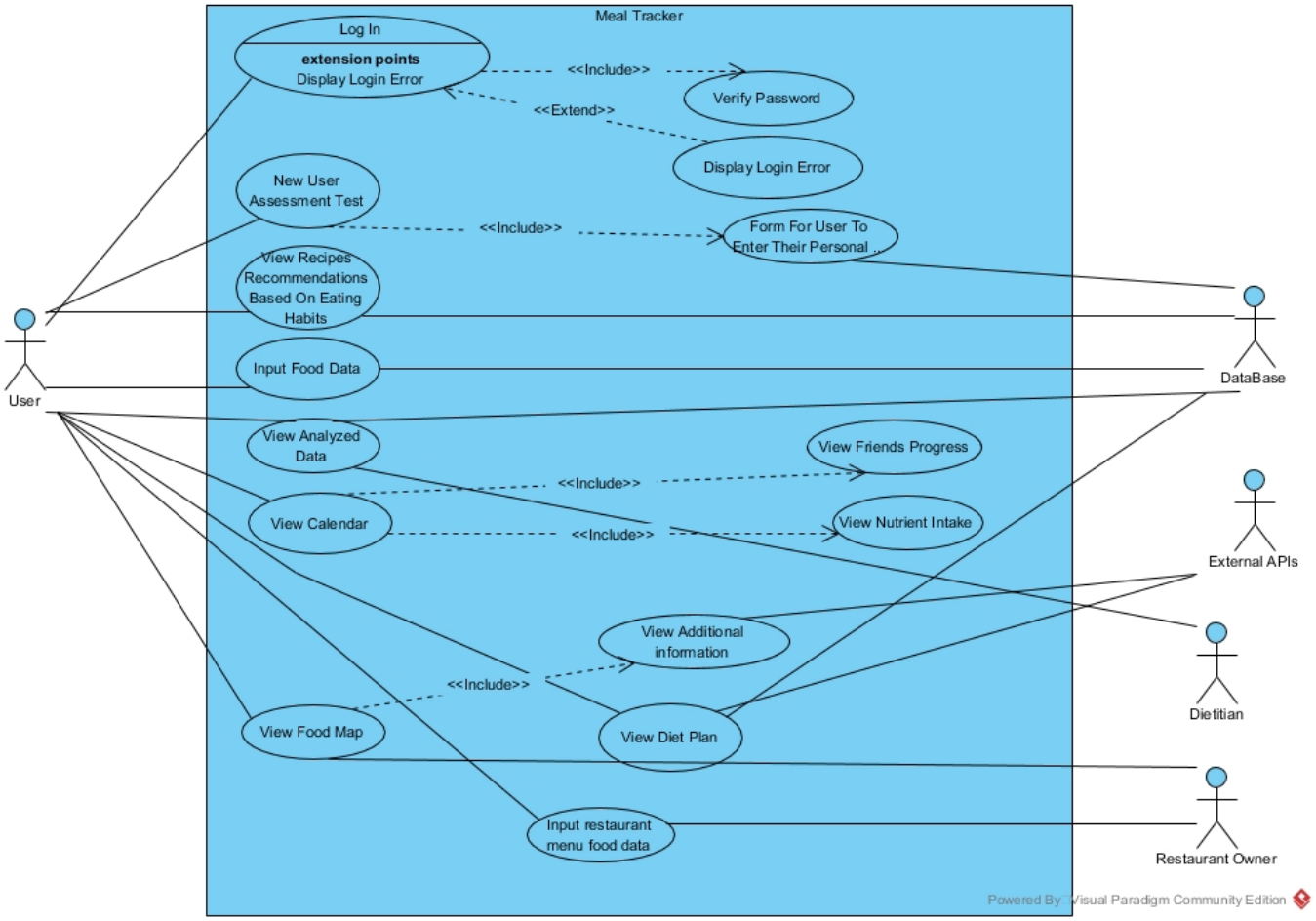
Features	**Estimated Effort**[3]
Nutrient Meal tracking in Restaurants	50 hrs
Friend System	30 hrs
Syncing External Data from Apple's Healthkit API	30 hrs
Totals:	110 hrs

Version 3 of the application is to introduce features and proof of concepts that may be further developed in the future. These features are not necessary for the required for the application to work properly but provides functionality that may be useful for small portion of the user base. Nutrient meal tracking in Restaurants will provide users the option to input data into the application automatically. Friend system is an extension of the calendar progression feature. It allows other users to view each other's progress and goals to increase friendly competitiveness. Syncing external data such as heartbeat, steps taken and sleep routine will all be retrieve from Apple's Healthkit API. This data can be converted to improve user's analyzed data feature from version 2 of the application.

System Diagrams:

Use case diagram

3



Actors

- Database
- External APIs
- Dietitian
- User

Description of the Use Cases

Log In

Log in will appear if user has not logged in previous in the application. User will be prompt to enter their username or email address. Alternative methods are available. Credentials will be verified using the database. If credentials are invalid, user are redirected back to the log in page. They will have access to the application once they pass this use case.

View New User Assessment Test

Appears once for new users. This use case allows users to enter their basic information to provide accurate feedback and prediction when using the application. When user is finish with this use case, the data gets sent and processed at the database using Firebase.

View Recipes Recommendations Based On Eating Habits

Recipes recommendation will be based on user's data that has been collected and stored in the database. System will come up with recommendation using various algorithms and external APIs to give accurate results.

Input Food Data

User enters the type of food that they consumed in the application. The application will then convert the food into raw data that can be analyzed and will be stored in the database.

View Analyzed Data

Data will be visualized and retrieve from the database. This will show the user the analyzed data that has been collected. External APIs are used to provide accurate results.

View Calendar

User will see their daily goals, achievements and progress. User also have an option to enter competitive mode to see their friends' progress.

View Food In Augmented Reality

3D model will appear in the application when viewing in augmented reality point of view. User can visualize and interpret food without going to the restaurant.

View Food Map

Map that shows location serving specific food. Filter option will be provided to distinguish different categories of food.

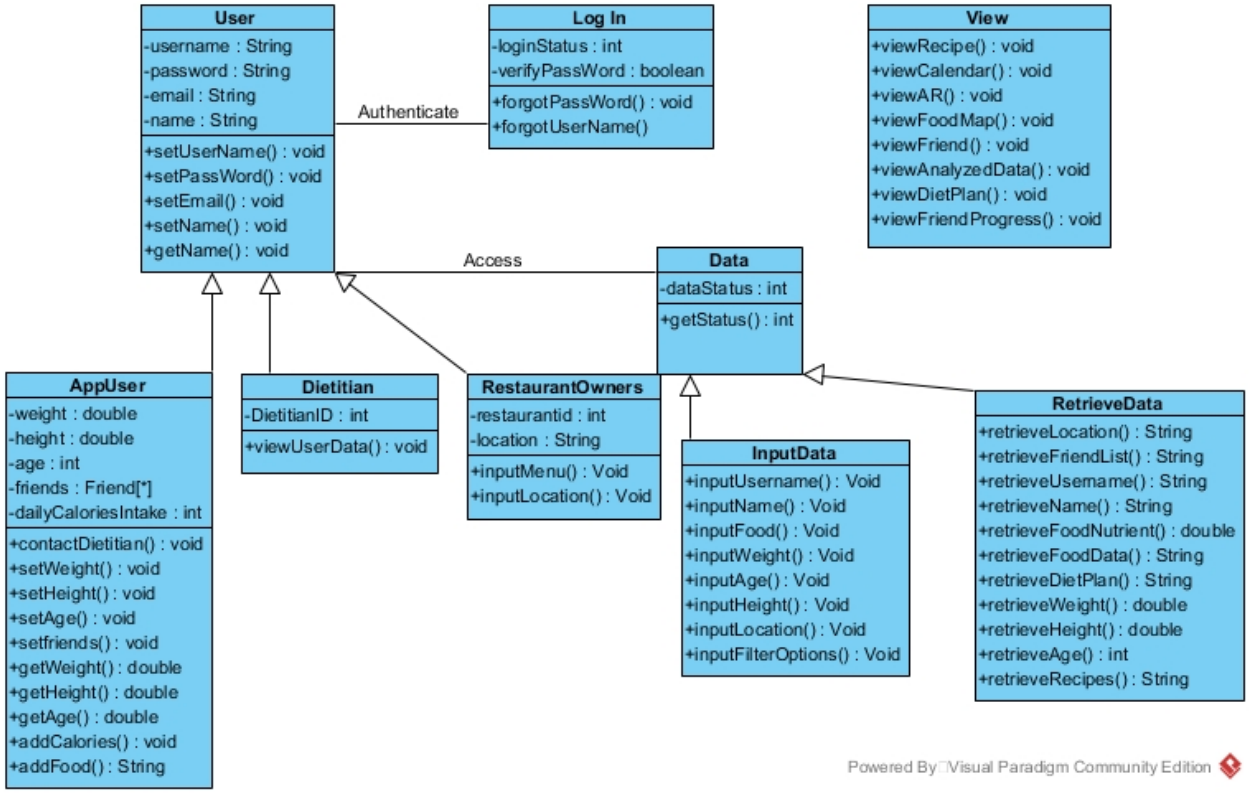
View Diet Plan

Uses data collected to generate a diet plan. Diet plans will be presented and user have a choice to choose the plan or any of the alternatives that the application has come up with.

Input Restaurant Menu Food Data

User can input data into the database with the information that restaurant owners has already pre defined.

Class Model Diagram



Description of the classes

User

Class that contains generic set of variables and operators. This class contains basic information about the user.

AppUser

Class that has attributes inherited from User class. AppUser class is mainly used for managing application user information

Log In

Class that manages the login system.

View

Functions that are used in the controller to display certain sections of the application

Data

Generic class that displays the status of the data transferred.

InputData

Functions that stores data in the database.

RetrieveData

Functions that stores data in the database.

Dietitian

Class with super class of user that can view certain user data when permitted

RestaurantOwner

Class with super class of user that have access to input location and food menus in the database