Software Title:

MyFit App

Team Members:

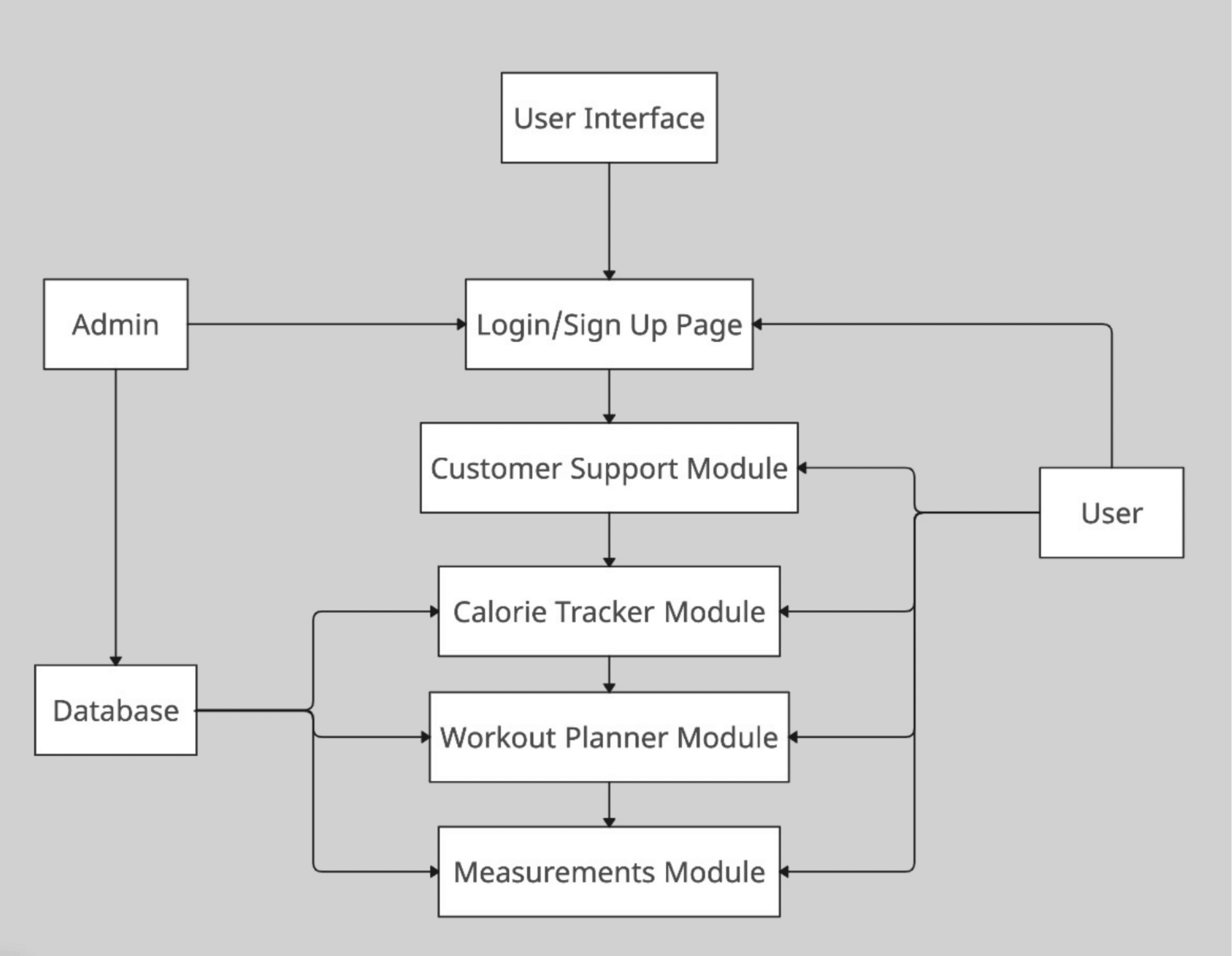
Teddy Barker
Sama Seif
Wissam Almasri
Umar Umar

System Overview:

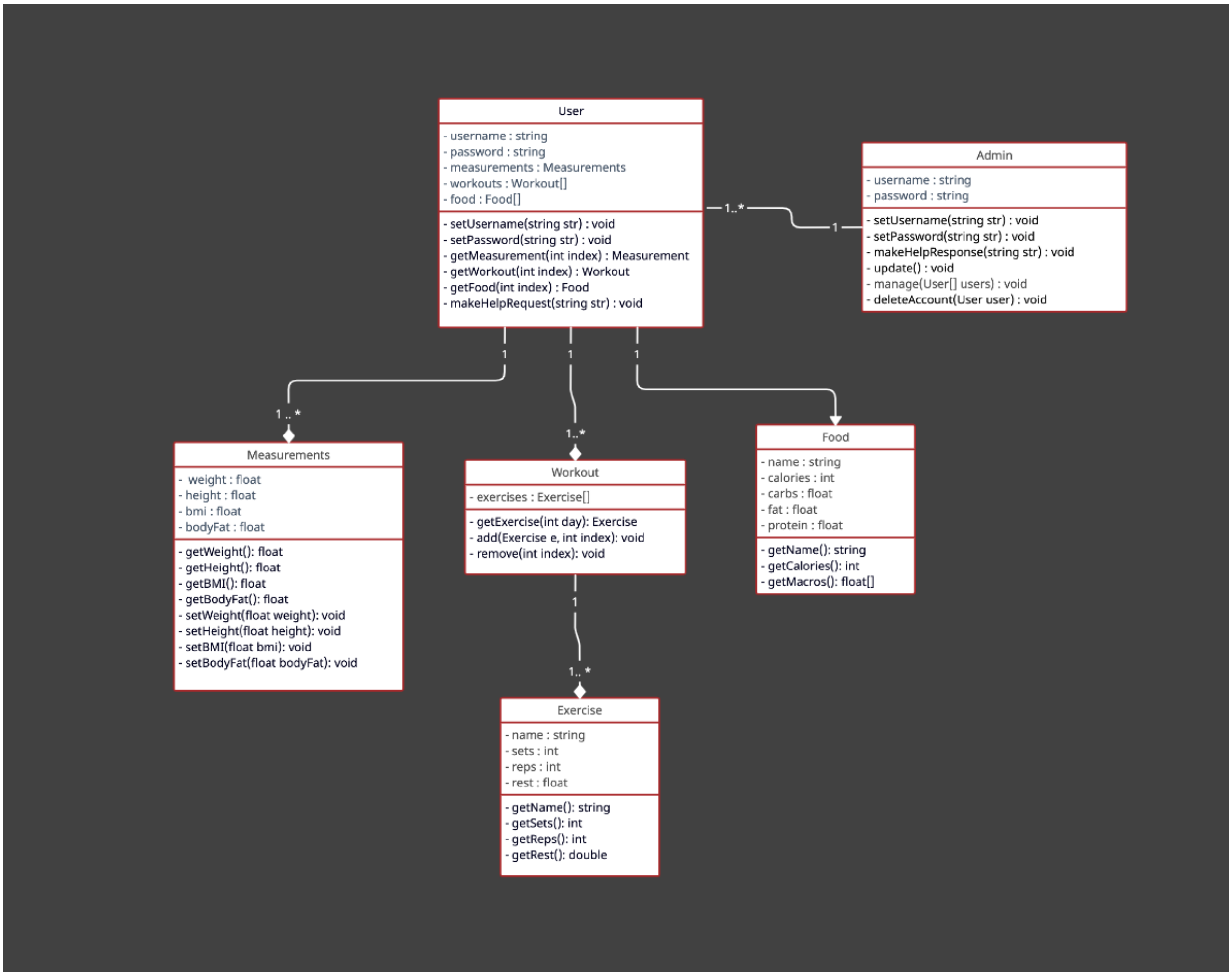
MyFit App serves the purpose of giving users the ability to track, monitor, and program their fitness goals by providing a dynamic user experience that solves multiple fitness goals including meal tracking, calorie counting, workout programming, weight monitoring, and goal setting. By utilizing this app and its features, users with the general goal of increasing their physical health and fitness can achieve their goal no matter what the specificity of their individualized requirements are. Overall, MyFit App aims to see the world become a healthier, happier place.

Software Architecture Overview

Architectural Diagram of all Major Components



UML Class Diagram



Class Descriptions:

- ❖ User
    - ○ The user class controls a client that is using the app. It maintains the attributes of username, password, measurements, workouts, and food. User name is composed of 3 other defined classes and has an association with the Admin class.
    - ○ Attributes
        - ▪ username: string that controls unique user identification and login details

        - password: encrypted string that controls user login details
        - measurements: instance of class measurements that controls the users personal information like height, weight, bmi, and body fat percentage.
        - workouts: list of workouts that compose the user's desired workout program
        - food: list of food items that the user has eaten throughout the days
    - Operations:
        - setUsername(): void function that allows user to change their username if they want to
        - setPassword(): void function that allows user to change their password if they want to
        - getMeasurement(): returns the measurements associated with the user
        - getWorkout(int index): returns the specified workout associated with the user
        - getFood(int index): returns the specified food item logged by the user
        - makeHelpRequest(string str): void function that controls a user customer service request
- Admin
    - The admin class controls an administrator of the company that has more selective control over aspects of the application such as maintenance and responding to customer service requests.
    - Attributes
        - username: string that controls unique admin identification and login details
        - password: encrypted string that controls admin login details

- Operations:
    - setUsername(): void function that allows admin to change their username if they want to
    - setPassword(): void function that allows admin to change their password if they want to
    - makeHelpResponse(string str): void function that allows admin to respond to a request
    - update(): void function that allows the admin to make any necessary updates to the app
    - manageUser(User[] users): void function that allows the admin to manage a list of users
    - deleteAccount(User user): void function that allows the admin to delete a user's account if they want

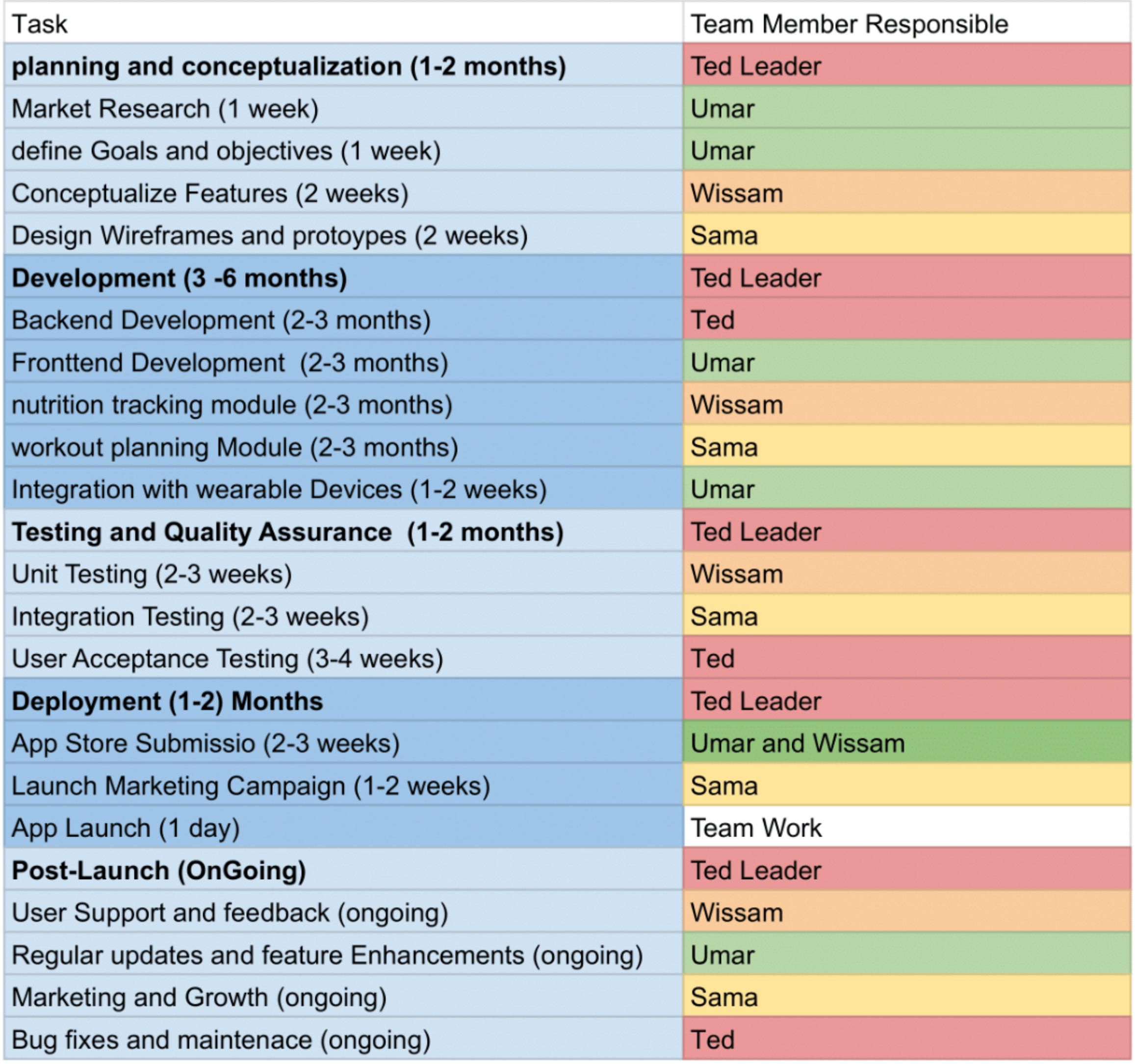
- Measurements
  - This class resembles a collection of user measurements and manages attributes weight, height, body mass index (bmi), and body fat percentage (bodyFat). This allows users to see their progress change overtime so they can achieve their goals.
  - Attributes
    - weight: float that represents the user's weight in pounds
    - height: float that represents the user's height in inches
    - bmi: float that represents the user's body mass index
    - bodyFat: float that represents the user's body fat percentage
  - Operations
    - getWeight(): returns weight
    - getHeight(): returns height
    - getBMI(): returns bmi
    - getBodyFat(): returns bodyFat
    - setWeight(float weight): void function that changes weight

        - setHeight(float height): void function that changes height
        - setBMI(float bmi): void function that changes bmi
        - setBodyFat(float bodyFat): void function that changes bodyFat
- Workout
    - This class represents a collection of exercises that formulate a specific workout routine. It manages a list of exercises as an attribute.
    - Attributes
        - exercises: list of exercises that comprise the workout routine
    - Operations
        - getExercise(int day): returns the Exercise of the specified day in the workout
        - add(Exercise e, int index): void function that adds the specified exercise to the workout
        - remove(int index): void function that deletes an exercise from the program
- Exercise
    - This class represents all the information for one exercise and manages a string for the name, two integers: sets and reps, and a float for the time in seconds of rest required to recover from this movement.
    - Attributes:
        - name: string that describes the name of the exercise
        - sets: integer value for number of sets for desired exercise
        - reps: integer value for the number of reps per set
        - rest: float value to represent the rest time in seconds after exercise
    - Operations
        - getName(): returns the string for the name of the exercise

        - getSets(): returns sets
        - getReps(): returns reps
        - getRest(): returns rest time

- Food
    - This class represents one food item. It manages a name, calories, carbs, fat, and protein attributes.
    - Attributes
        - name: string for the food item's name
        - calories: integer value of caloric specifications
        - carbs: float value of number of grams of carbs
        - fat: float value of the number of grams of fats
        - protein: float value of the number of grams of protein
    - Operations
        - getName(): returns the name
        - getCalories: returns calories
        - getMacros: returns a float array of 3 elements of the calories, fats, and proteins

Development Plan and Timeline



Task	Team Member Responsible
planning and conceptualization (1-2 months)	Ted Leader
Market Research (1 week)	Umar
define Goals and objectives (1 week)	Umar
Conceptualize Features (2 weeks)	Wissam
Design Wireframes and protoypes (2 weeks)	Sama
Development (3 -6 months)	Ted Leader
Backend Development (2-3 months)	Ted
Fronttend Development (2-3 months)	Umar
nutrition tracking module (2-3 months)	Wissam
workout planning Module (2-3 months)	Sama
Integration with wearable Devices (1-2 weeks)	Umar
Testing and Quality Assurance (1-2 months)	Ted Leader
Unit Testing (2-3 weeks)	Wissam
Integration Testing (2-3 weeks)	Sama
User Acceptance Testing (3-4 weeks)	Ted
Deployment (1-2) Months	Ted Leader
App Store Submissio (2-3 weeks)	Umar and Wissam
Launch Marketing Campaign (1-2 weeks)	Sama
App Launch (1 day)	Team Work
Post-Launch (OnGoing)	Ted Leader
User Support and feedback (ongoing)	Wissam
Regular updates and feature Enhancements (ongoing)	Umar
Marketing and Growth (ongoing)	Sama
Bug fixes and maintenace (ongoing)	Ted