[Get Fit]

*Your proposed project must have the following overall requirements:*

* *It must be intended for mobile, wearable, or embedded-style platforms.*
* *You may be required to use data persistence pending the project type.*
* *The project must have significant depth and appropriate complexity given the size of your team and the time allotted.*
* *The responsibility for any technical training your project may require rests upon you and your teammates.*

# Project Description

This is a fitness tracker that will record calorie intake as well as calories burned. Users may input meals eaten, which will return an estimate of calories taken in. Workouts will also be recorded and an estimate of calories burned will be shown. The target audience is for those looking to lose weight, and those who are trying to have a stricter diet. This is a health tracking application for mobile devices.

# Part 1: Non-Functional Requirements

Non functional requirements describe the overall operation of a system, not specific behaviors or operations. Important information such as the runtime platform, system compatability, and overall quality requirements are listed here.

1. The system is to be an Android mobile application
2. Recommended specs:
   1. Android OS 4.4 (Kit Kat)
   2. Screen Resolution: 1080 x 1920
   3. Screen Size: 5.0in
   4. Available storage space of at least 500MB
3. The system will be thouroughly tested prior to delivery.
   1. The system must be shown to have functional errors in no more that 15% of delivered requirements.

# Part 2: Functional Requirements

Below are an example functional requirements. Your team must fully define what your system will do. Language and terminology must be appropriate for laypeople, as this document is for the client. **Note:** *The following requirements are not complete!* The requirements must be validated with your client through interviews, emails, or other communication.

## Glossary

Before formally defining the functional requirements, it is necessary to define the domain-specific terminology to avoid confusion or ambiguity. For example:

* “System” – the delivered software and any additional resources required to make it function.
* “User” – any person who wishes to user/interact with the application.

Each requirement has a priority level of [1], [2], or [3].

1. Must-have functionality critical to the problem solution.
2. Highly desirable feature that should be included.
3. Optional requirements that will be completed if time allows.

## Account creation

1. Users must create an account in order to use the application[1]
2. The following required information is stored:
   1. Username[1]
   2. Password[1]
   3. Email[1]
3. The following optional information is stored:
   1. Age
   2. Weight
   3. Height
   4. Goal Weight
      1. Calorie Quota is created if current weight and goal weight are entered.
   5. Gender

## Ingredients

1. A list of base ingredients – users cannot add to this table[1]
   1. The system stores the following required information about the ingredient:
      1. Calories[1]
      2. Category(vegetable, meat, etc)[3]
      3. Ingredient name[1]

## Meals

1. A user can add a meal to their own library[1]
   1. The system stores the following required information about a meal:
      1. Ingredients [1]
      2. Calories [1]
      3. Servings [1]
      4. Publication Date [1]
      5. Meal type (breakfast, lunch, dinner, snack) [1]
      6. Recommended for (weight loss, weight gain, etc) [2]
      7. Dietary Restrictions[2]
      8. Meal name[1]
2. A user can add their meal to the system for other users to see[2]
   1. Required stored information:
      1. Creator[2]
3. A user can rate a meal [3].
   1. Stars 1-5 on meals created by other users

## Meal Search

1. Meals will be filtered based on goal weight
   1. If calorie quota is 2000, only meals 2000 and under are shown.
   2. As meals are eaten, calories are deducted from quota, thus narrowing search results

## Workouts

1. Users can add workouts to their own library[1]
   1. The system stores the following required information about a workout:
      1. Workout type (strength training, cardio)[1]
      2. Length [1]
      3. Workout name[1]
2. Users can add workouts to the system for other users to see[2]
   1. Required stored information:
      1. Creator[2]

## Timer

1. There is a timer that users can use [3]
   1. Will make a noise every 5 minutes

# Part 3: Basic Design

Before developing any major project, you should first have a design and a plan. For this section, consider your logical architecture and produce a reasonably complete design to express it. Most commonly, a UML diagram will suffice, but supplementing that with wireframe designs of the UI, flow chart diagrams, or other design documents is acceptable and encouraged.

Your design should encapsulate the logical architecture (classes, interfaces, data members, methods, etc) and a component model as needed (these show the relationships between things like a server, mobile device, local application, etc).

# Part 4: Project Plan

You must define milestones for each phase of development. There will be 4 phases of development followed by the final product presentation at the end of the sprint. Each phase begins on the first day of the school week and ends on the last day. On the first day, each team will deliver a document containing the goals of each member of the team. On the final day of the phase, the instructor will meet with each team and ask for a demo. Each person must be able to demonstrate what he/she contributed during the week.

For this proposal, outline the major goals your team will have collectively completed by the end of each phase.

Project length: May 8th – June 5th

Phase 1: Project Proposal/Training of Xamarin

Phase 2: Set up database/begin structuring of code, allow login

Phase 3: Have a working prototype, allow creation of meals/workouts

Phase 4: Allow users to add to the database(priority level 2s)

Phase 5: Debug, add level 3 priorities