Radish: A Cross Platform Meal Prepping App For Beginner Weightlifters

A Senior Project

presented to

the Faculty of the Computer Science Department

California Polytechnic State University – San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science in Computer Science/Software Engineering

By

Tanay Gottigundala, Cory Baxes, Spoorthy Vemula

June 13th, 2019

Introduction

With the increasing ease of access and decreasing price of most food, obesity rates in the developing world have risen dramatically in recent years. As of March 23rd, 2019, obesity rates had reached 39.6%, a 6% increase in just 8 years. Research has shown that people with obesity have a significantly increased risk of heart disease, stroke, type 2 diabetes, and certain cancers, among other life-threatening diseases. In addition, 42% of people who begin weightlifting quit because it's too difficult to follow a diet or workout regimen.

We created Radish in an attempt to tackle these problems. Radish makes it easier for people to achieve fitness goals without having to do a large amount of diet and fitness research that generally overwhelms beginner weightlifters. Our contributions in this field are unique because we make decisions for the user so they have fewer disinhibitions from starting and continuing on their fitness journey. Our target demographic for this app are people with limited fitness experience who want to attempt to improve their health and aesthetics. We believe we've successfully created a strong proof of concept in the scope of this senior project. We will be continuing our work with this app after the completion of this quarter and we hope to release the app on the App Store and Google Play by the end of the year.

Background

Technologies

Our biggest focus when picking technologies to work with was the importance of the application being both scalable and reliable. We sought to make this app as production ready as possible and that we kept that big picture in mind when we chose our technologies and throughout our development process.

Figma

Figma is an interface design application that enabled us to quickly prototype our ideas. Each of our sprints started out with a clear UI prototype of the screens we were going to develop during that sprint. We used these screens to guide our decision making for the both the backend and the app development.

Flutter

Flutter is an open-source mobile application development framework that is created and maintained by Google. Our decision to use Flutter rather than the alternatives are due to a few reasons. First, Flutter enables us to do cross-platform development which allowed us to make an Android app and it's iOS equivalent without any extra effort. Second, since our application has no highly hardware-specific features, we did not feel the need to make a native application. As a side note, we did initially begin by using Android Studio when we were working on a different idea and eventually moved over to Flutter when we switched ideas because we didn't have such strict hardware needs. Finally, Flutter has very low startup time due to both the programming language it utilizes(Dart) as well as its use of widgets to increase code reusability.

Node.JS/Express

Express is an open-source Node.JS framework that is widely considered the de facto standard server framework for Node.JS. It allowed us to manage everything including but not limited to routes, requests, and tokens. In addition, Node.JS is very lightweight, and allows for the building of fast and scalable applications, which we believe will be key to our success in the future as our user base grows bigger.

Google Cloud Platform

We hosted our server on Google Cloud Platform's Compute Engine. Our decision was essentially between hosting on a local server or using a cloud platform and we chose the latter due to the scalability and reliability that we desired from this project. We

ran our servers with PM2, a process manager for Node.JS that allowed us to maintain 100% server uptime.

Google Cloud Datastore

Our decision to use Google Cloud Datastore instead of the traditional relational database alternative was due to the type of data processing we anticipated on doing. Using NoSQL allows to run optimized queries for large amounts of data which is something we have to deal with since our server responses can be very large.

API Hurdles

Picking an API proved to be a very challenging problem given our constraints. While there are a lot of recipe APIs in the market, the recipes were user provided which has its disadvantages because it has the ability to make for poor recommendations. In addition, a lot of APIs cost a lot of money and we weren't ready to spend thousands of dollars each month on a commercial API. This meant that we were forced to deal with undetailed nutrition data and recipe information.

We solved this problem by curating the list of recipes that we allowed to be used in our database. This gave us more control of what we showed to users and ultimately felt like a much more personal experience. The tradeoff with this approach is that we were forced to put in a lot of manual work selecting recipes that we believed fit our desired characteristics for the meal plan. Ultimately, this meant that we did not have the resources to get a large amount of recipe data into our database. Currently, our datastore only has about 50 total recipes. On the flip side, this is something we plan to market heavily when we release the app since we are putting in a lot of effort into simply determining what belongs on our meal plan.

Finally, we solved our API problem in a somewhat unique and "hacky" manner by simply web scraping a popular recipe website called FatSecret.com. While we are allowed to use this for development purposes. In the long run, we will either need to hire a team to create our own recipes or find an API that we are satisfied with and buy a commercial license. I believe we will take the former approach in the long run so that we

have full control of the process rather than relying on external sources but the initial release of the app is likely to use a commercial recipe API.

Fitness and Diet

The fitness realm is one that involves a lot of contradictory research and an oversaturation of information. Going through this research ourselves and understanding it was a large part of our preliminary work. It helped that we have a lot of experience in this field but creating a plan specifically for this niche proved to be challenging. All the research pointed to a few key aspects necessary in order to achieve these goals. In order to lose weight, an individual had to be in a caloric deficit in order for the body to be in a catabolic stage to burn fat. In order to gain weight, an individual had to be in a caloric surplus in order to be in an anabolic state to gain muscle. Using the Mifflin St. Jeor equation, we were able to calculate the Basal Metabolic Rate (BMR) for an individual after being given their age, sex, height, and weight. In conjunction with the individual's activity level, the BMR can help estimate the individual's total daily energy expenditure (TDEE). Additionally, the body requires at least 0.8g/lb of body mass for optimal muscle gain. From these values, we were able to determine the macronutrient needs for each user.

Our workout routines are from a popular Reddit user named nSuns. His program has proved to be very effective for building muscle through linear progression for beginners. Linear progression means that the amount of weight the individual lifts each week, which is something that only happens in newer lifters. In addition, this workout routine has been widely documented to make one stronger while still achieving hypertrophy to make the individual look bigger.

Description

We began the project by clearly defining technical requirements of the app, defining how users would be able to use the app, and picking our technology stack. Shortly before beginning the development process we defined each team member's role and planned out each development sprint.

Requirements

Based on our goals for the app and our target demographic, we developed the following nonfunctional and functional requirements:

Nonfunctional Requirements

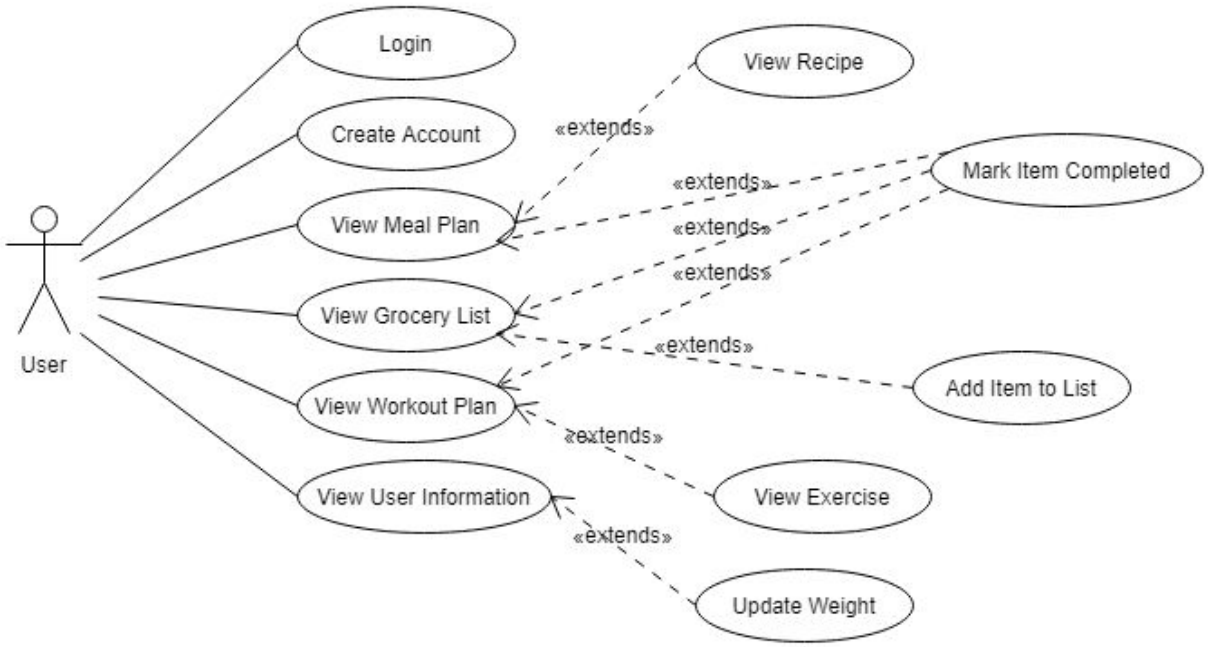
Designation	**Requirement**
NFR-1	The app must look and function the same on both iOS and Android devices.
NFR-2	The app must store user data securely in a way that only the app can access the data.
NFR-3	The app development must be completed in 10 weeks.

Functional Requirements

Designation	**Requirement**
FR-1	The app must develop one-week meal and workout plans based on each user's personal fitness goals and personal preferences.
FR-2	The app must allow users to create a new account by providing information about their name, email, age, sex, height, weight, fitness goals, and dietary preferences.
FR-3	The app must generate a new meal and workout plan when an existing plan's timeline is completed.
FR-4	The app must allow users to check off meals, workouts, and grocery items as they are completed or purchased.
FR-5	The app must provide recipe information, including macronutrients, ingredients, and cooking instructions, for each meal in a plan.
FR-6	The app must provide instructions and a video demonstration for each exercise in a workout.
FR-7	The app must generate a grocery list that includes all ingredients required to prepare the meals included in a given plan.
FR-8	The app must allow users to add additional grocery items to the automatically-generated grocery list.

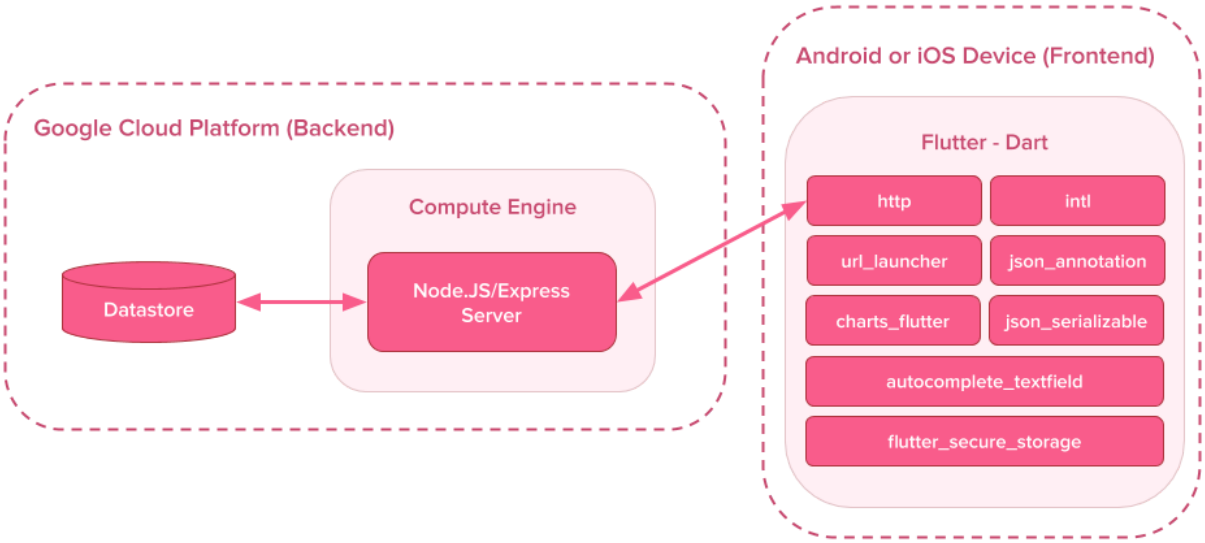
FR-9	The app must require users to login using a personally-created account where all progress and weekly plans are kept.
FR-10	The app must allow users to view their account information and fitness goals after account creation.
FR-11	The app must allow users to update their weight after account creation to track progress.

Use Cases



The diagram above shows the general use cases for users of our app. After opening the app, users will be prompted to either login or create an account. After logging in users have four tabs to choose from: Food, Groceries, Fitness, and You. The Meal, Groceries, and Fitness tabs allow some sort of item to be checked off as completed (either a meal, grocery item, or exercise respectively). Users can also view recipes by selecting a meal on the Food tab and view exercises by selection an exercise on the Fitness tab. On the Grocery tab, users can add additional items to the grocery list. On the You tab, users can update their weight to see their progress.

System Architecture



Our backend is hosted entirely in Google Cloud Platform. The API server is a Nods.JS/Express server running on a virtual machine in Google Compute Engine. The server communicates with Datastore to store and retrieve data, and it offers public-facing, secure endpoints for the frontend to access.

Users can run the app on either iOS or Android devices. The app itself is developed in Flutter using the Dart language, and the Dart code compiles down to native iOS and Android code combined with a custom rendering engine for displaying the app. The “http” Flutter package communicates with our Node.JS API server to send and retrieve data from the backend. Our Flutter app uses the following packages to provide additional features:

Package	**Reason for Use**
http	Provides easy functions for consuming HTTP resources, allowing easy access to our API server.
intl	Provides localization features used for parsing and formatting dates.
url_launcher	Easy URL launching used for launching workout videos.
json_annotation	Defines annotations for use by the json_serializable package.

charts_flutter	Cross-platform data visualization library that conforms to Material Design standards. Used for displaying user weight over time.
json_serializable	Allows generation of code for automatic serialization of an object into JSON and back from JSON into the object. Used for saving weekly plan progress locally.
autocomplete_textfield	Textfield with autocomplete as user types. Used for easy searching and selection of dietary restrictions and favorite foods.
flutter_secure_storage	Secure data storage using Keychain for iOS and AES encryption for Android. Used for storing login credentials and weekly plans locally and securely.

Development Process

Roles

We assigned project roles based on the experience and expertise of each team member as follows:

Team Member	**Role**	**Description**
Cory Baxes	Front-end Developer	Implementing UI and app functionality in Flutter
Tanay Gottigundala	Back-end Developer	Implementing API functionality in Node.JS running on Google Cloud services
Spoorthy Vemula	UI/UX Designer	Designing how the app looks and feels and implementing this in Flutter.
Dan Weeks	Advisor/Mentor	Guided us throughout the development phase of the app and will serve as our mentor after the completion of senior project

Starting Point

Since some work had been done on our original app idea during the Winter 2019 quarter, our back-end technology stack was largely in place. The functionality of the Node.JS server needed to be mostly redeveloped to support new and different API calls,

but we had a bare-bones Node.JS server running on Google Compute Engine and interacting with Firestore. The switch from Android to Flutter, however, meant that the previously-developed app technology would be scrapped and rebuilt from scratch in Flutter.

Timeline

We decided to use the Agile software development method so that we would have meaningful features completed frequently throughout the short 10-week development process. The development process was divided into five sprints that were each two weeks long. Each feature was planned into a specific sprint for completion in a way that made sense for building each feature's functionality off of others. Below is the timeline we followed, including the functional requirement associated with each task:

	SPRING 2019									
	APRIL					MAY				JUNE
	1	8	15	22	29	6	13	20	27	3
PROJECT WEEK	1	2	3	4	5	6	7	8	9	10
SPRINT 1	Onboarding Frontend (FR-2)									
Account Creation	Login Frontend (FR-9)									
	Onboarding Backend (FR-2)									
	Login Backend (FR-9)									
	TDEE Calculation Backend (FR-1)									
SPRINT 2			Recipe API Connection (FR-5)							
Meal Plan & Groceries			Meal Plan Backend (FR-1)							
			Meal Plan Frontend (FR-1)							
			Grocery List Generation Backend (FR-7)							
			Grocery List Frontend (FR-7, FR-8)							
SPRINT 3					Next Week's Plan Generation (FR-3)					
Workout Plan					Workout Plan Backend (FR-1, FR-6)					
					Workout Plan Frontend (FR-1, FR-6)					
SPRINT 4							User Page Frontend (FR-10)			
User Settings							User Account Changes Frontend (FR-11)			
							User Account Changes Backend (FR-11)			
SPRINT 5									Weekly Plan Progress Persistence (FR-4)	
Hardening									Defect Discovery & Fixes	

Evaluation

Requirements

Our app development was largely successful. We were able to fully implement 10 of 11 functional requirements, partially implement the one remaining functional requirement, and simultaneously ensure all nonfunctional requirements were met. To see images of all app screens and cuntaionlity, refer to Appendix A. Below is the status of each requirement:

Nonfunctional Requirements

Designation	Requirement	Achieved?
NFR-1	The app must look and function the same on both iOS and Android devices.	**Yes**
NFR-2	The app must store user data securely in a way that only the app can access the data.	**Yes**
NFR-3	The app development must be completed in 10 weeks.	**Yes**

Functional Requirements

Designation	Requirement	Completed?
FR-1	The app must develop one-week meal and workout plans based on each user's personal fitness goals and personal preferences.	**Partially**
FR-2	The app must allow users to create a new account by providing information about their name, email, age, sex, height, weight, fitness goals, and dietary preferences.	**Yes**
FR-3	The app must generate a new meal and workout plan when an existing plan's timeline is completed.	**Yes**
FR-4	The app must allow users to check off meals, workouts, and grocery items as they are completed or purchased.	**Yes**
FR-5	The app must provide recipe information, including macronutrients, ingredients, and cooking instructions, for each meal in a plan.	**Yes**
FR-6	The app must provide instructions and a video demonstration for each exercise in a workout.	**Yes**
FR-7	The app must generate a grocery list that includes all ingredients required to prepare the meals included in a given plan.	**Yes**
FR-8	The app must allow users to add additional grocery items to the automatically-generated grocery list.	**Yes**
FR-9	The app must require users to login using a personally-created account where all progress and weekly plans are kept.	**Yes**

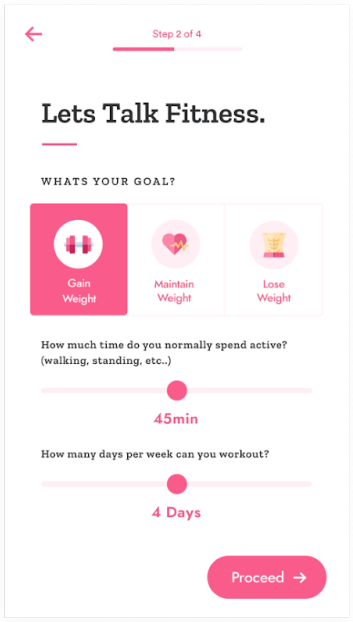
FR-10	The app must allow users to view their account information and fitness goals after account creation.	**Yes**
FR-11	The app must allow users to update their weight after account creation to track progress.	**Yes**

Cross-Platform Consistency

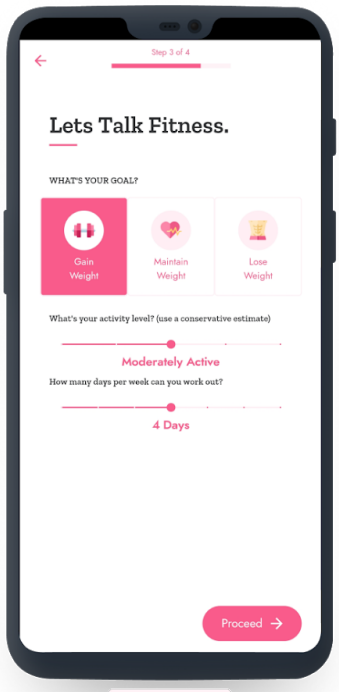
One of the key nonfunctional requirements was for the app to correctly represent the Figma UI prototypes created in both iOS and Android. To better understand if this requirement was met, we directly compared three very different screens across Figma, iOS, and Android.

We wanted to make sure our implementations on both Android and iOS matched the Figma designs as close as possible. Flutter allowed us to create custom UI components that worked on both platforms without too much hassle. There are only a few minor UI differences between the two. The back arrow style in the top right on the onboarding pages are slightly different on iOS and Android, the slider style we used on onboarding is a bit different since we decided to go with 5 discrete values instead of a continuous slider, and the checkbox style on the food page is a bit different because we wanted to reuse that component from another part of the app. Other than that they are all identical, so we met the cross-platform consistency requirement very well.

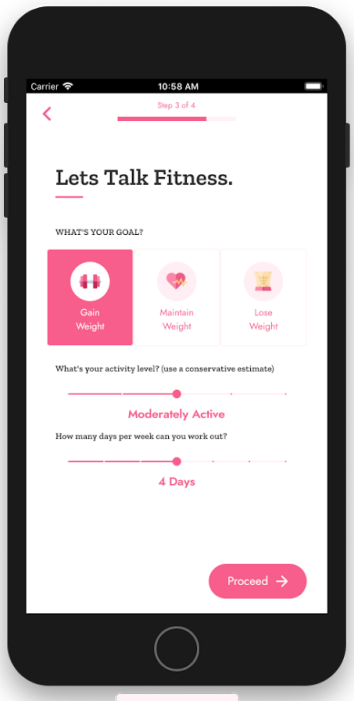
Onboarding Goals Screen Comparison



Figma

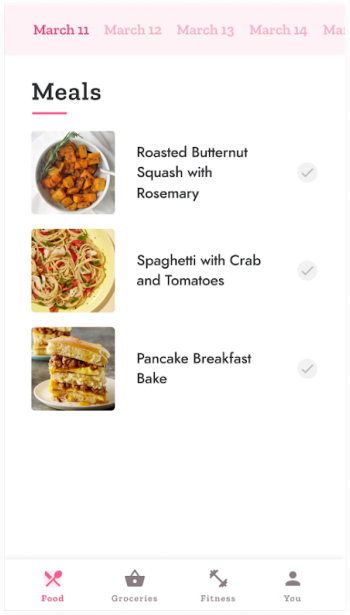


Android

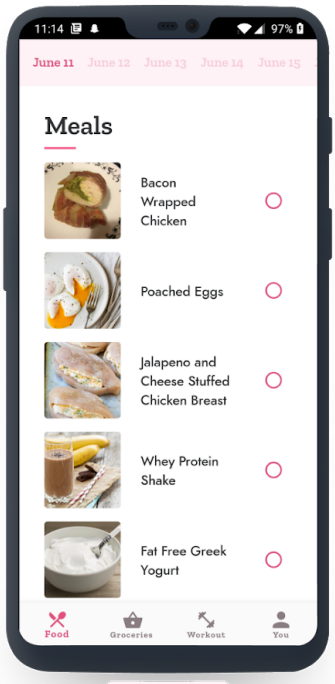


Iphone

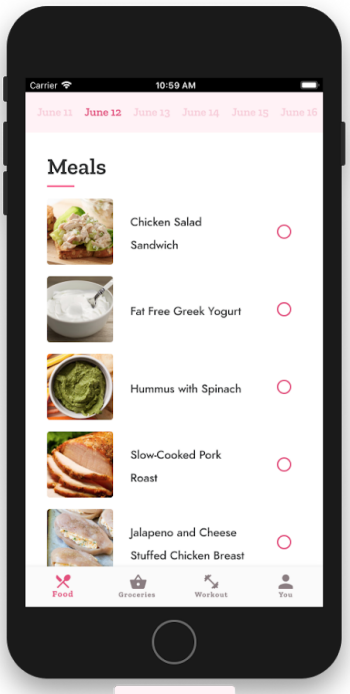
Home Screen Comparison



Figma

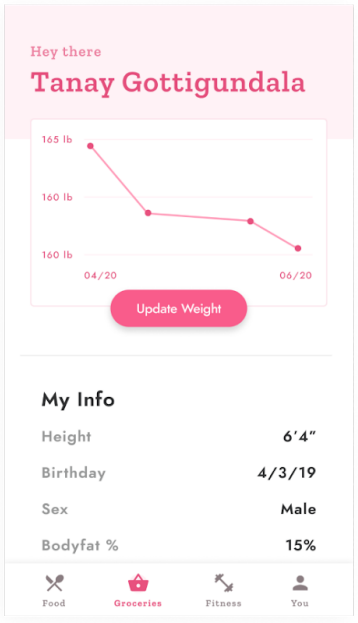


Android

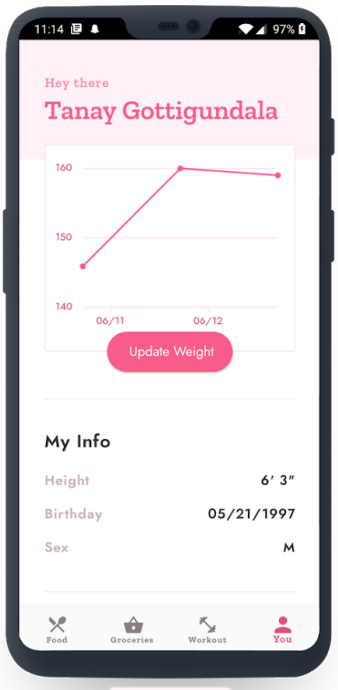


Iphone

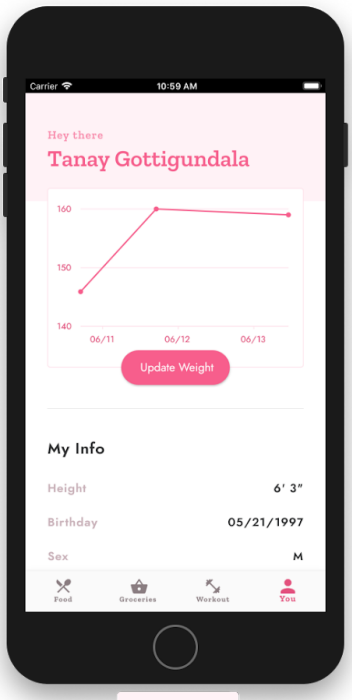
You Screen Comparison



Figma



Android



Iphone

Shortcomings

Even though we met nearly all requirements, our evaluation uncovered a few shortcomings we hope to improve in the future.

Weekly Plan Algorithm

The only functional requirement we failed to implement completely was customizing the meal and workout plans to the user's preferences, goals, and personal information (FR-1). Currently, our weekly plan generation algorithm generates generic high-protein diets and workout plans that would work decently for anyone wanting to be more fit. Our app has all the necessary information to produce more personal weekly plans, but we prioritized having every page of the app functional over a more intelligent weekly plan algorithm under the time constraints of the project. Improving this algorithm will only require work on the backend and means that frontend work in this area is completed since no more data collection than already implemented is needed.

Rudimentary Logging

Though we met the requirement for allowing users to log meals, grocery items, and workouts as completed (FR-4), we think users could use more flexibility in how they log meals and workouts. Currently, they can only check off that a meal or exercise is completed. However, since may deviate from the recommended meal plan or exercise plan, we think users need the ability to add in different, off-plan meals they consumed or exercises they accomplished as well as adjust the amount of a meal or number of reps of exercises completed. Ideally, the app would allow for this more detailed logging and readjust the weekly plan to still work well for the user.

Lack of Customization

Even with improved weekly plan generation, users are likely going to find meals or exercises they dislike and want to remove or swap. Currently, users cannot change any

items in the weekly plan, so we want to allow users to swap or remove individual items followed by an automatic update to the weekly plan that ensures fitness goals are met.

Big Picture

Overall, we successfully created a cross-platform app that can help people achieve their fitness goals, even in the app's early stages of maturity. While the weekly plans are not catered to the individual user yet, we believe improvements to this area will be easy to implement. The app is effectively feature-complete and usable on a day-to-day basis, so users that follow the weekly plans generated for them in Radish will be one step closer to their fitness goals. The look and feel of the app is fully fleshed out, so users can simply expect a more personal experience of the same type as the app matures.

Conclusions and Future Work

With the completion of senior project, Radish is at a stage where we consider it a successful implementation as a proof of concept of what we hope to accomplish in the long run. We learned how to build scalable applications from start to finish, which is an invaluable skill in the industry. While we believe that we still have a solid amount of work to do before it is production ready, we've taken steps to ensure that the work that we have completed so far scales and translates very well into our next stage of development.

There are some things we hope to accomplish before we will release this up. This list of key tasks that must be completed before it is market ready is as follows:

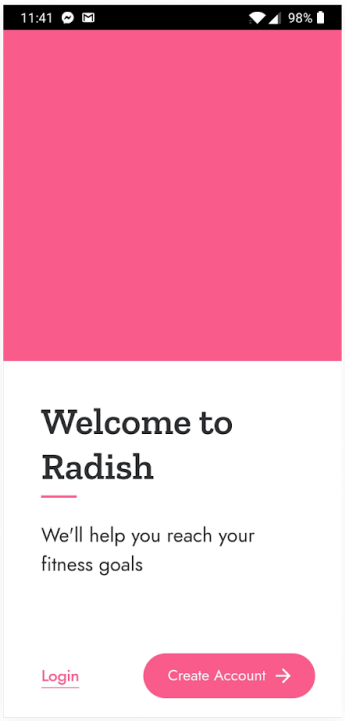
- More meal and workout customization
- Implementation of a user-based and item-based collaborative filtering algorithm to allows for better recommendations
- More curated recipes
- Ability to substitute meals from restaurants

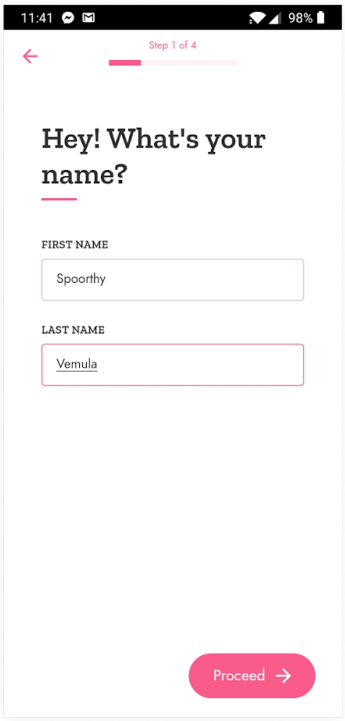
- Detailed instructions for exercises

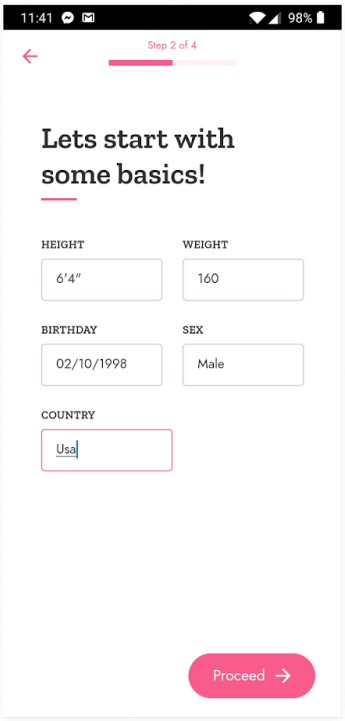
These are key features that we believe are essential to the value that our app will provide. We fully believe in Radish's ability to capitalize on the market needs of beginner weightlifters and that we will be able to turn this app into a startup once we find the right product-market fit and revenue model.

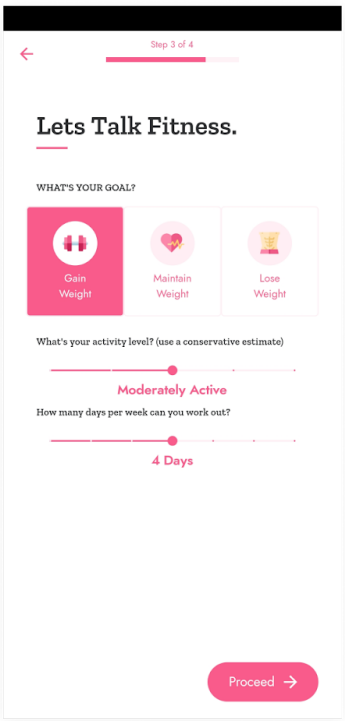
Appendix A: Completed App Images

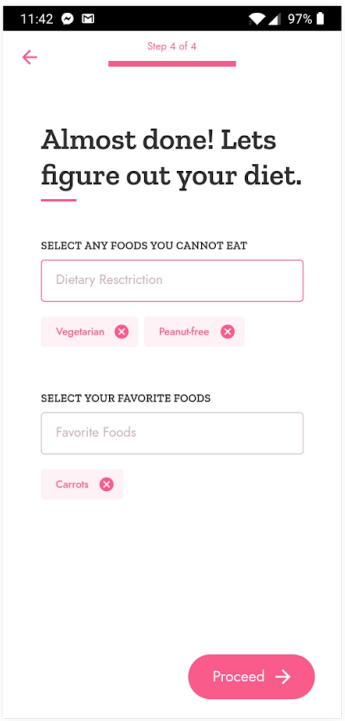
Onboarding:

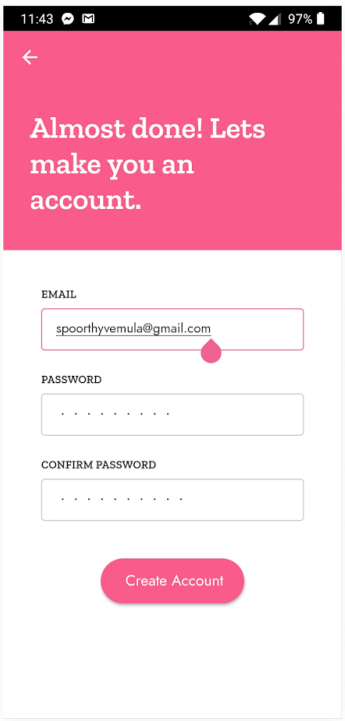




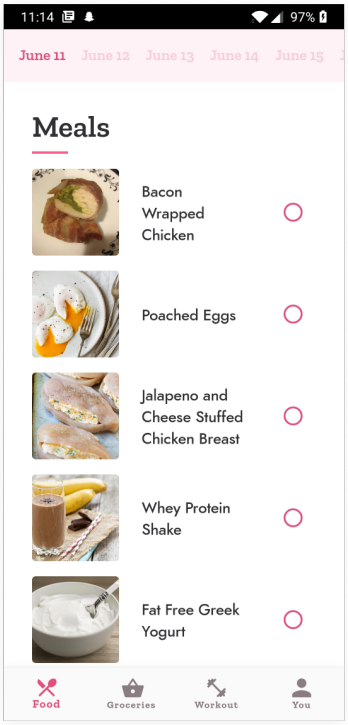


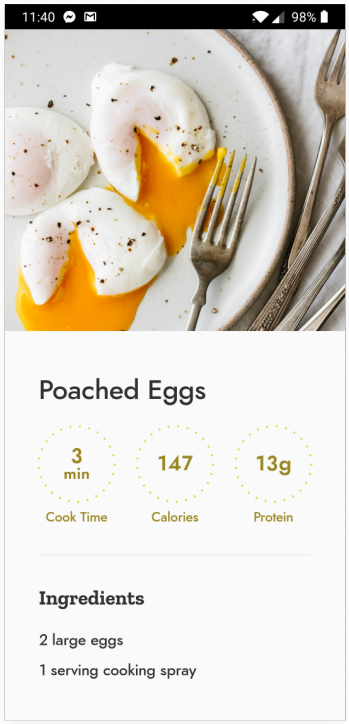

Step 3 of 4
Lets Talk Fitness.
WHAT'S YOUR GOAL?
Gain Weight
Maintain Weight
Lose Weight
What's your activity level? (use a conservative estimate)
Moderately Active
How many days per week can you work out?
4 Days
Proceed


11:42
97%
Step 4 of 4
Almost done! Lets figure out your diet.
SELECT ANY FOODS YOU CANNOT EAT
Dietary Resctriction
Vegetarian
Peanut-free
SELECT YOUR FAVORITE FOODS
Favorite Foods
Carrots
Proceed

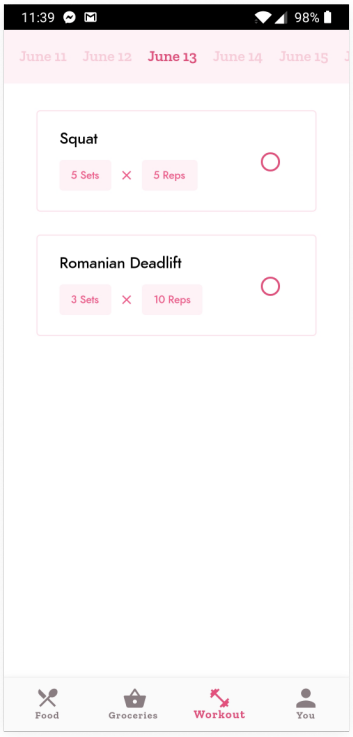

11:43
97%
Almost done! Lets make you an account.
EMAIL
spoorthyvemula@gmail.com
PASSWORD
CONFIRM PASSWORD
Create Account

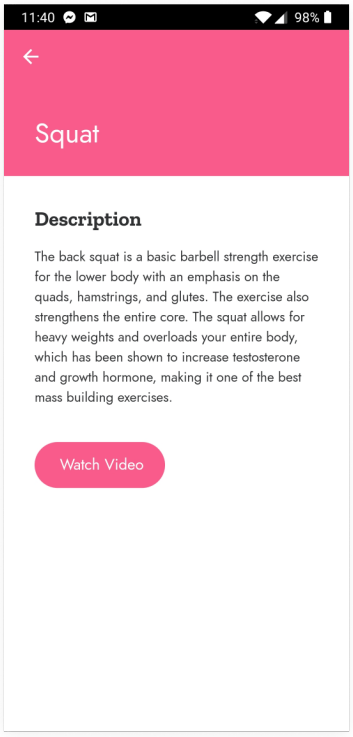
Meals:


11:14
97%
June 11
June 12
June 13
June 14
June 15
Meals
Bacon Wrapped Chicken
Poached Eggs
Jalapeno and Cheese Stuffed Chicken Breast
Whey Protein Shake
Fat Free Greek Yogurt
Food
Groceries
Workout
You

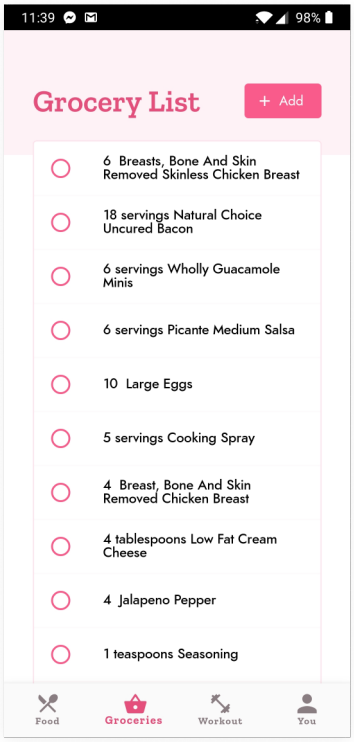

11:40
98%
Poached Eggs
3 min
Cook Time
147
Calories
13g
Protein
Ingredients
2 large eggs
1 serving cooking spray

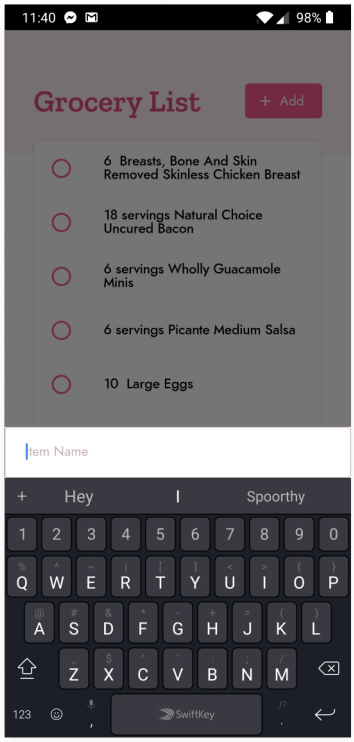
Workouts:



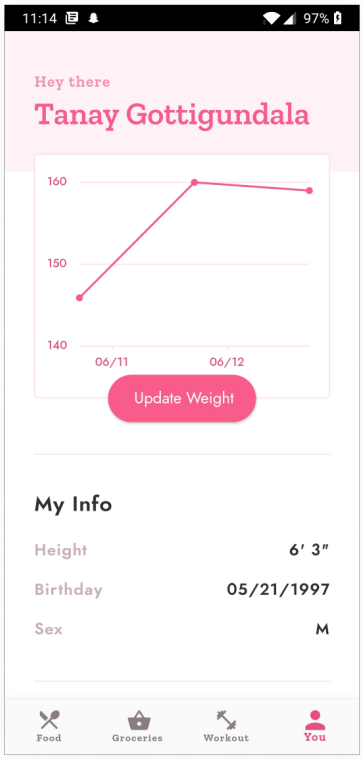


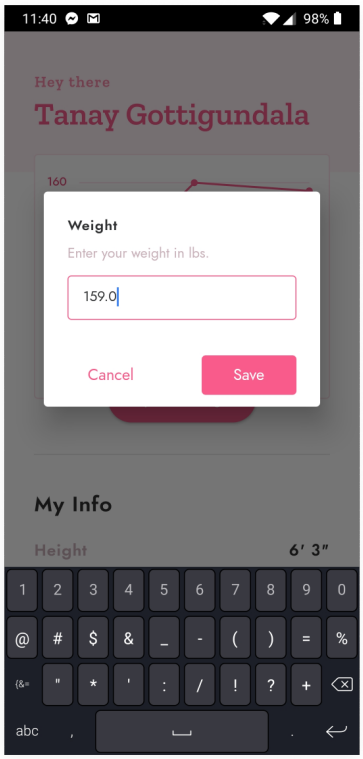
Groceries:





You Page:


11:14
97%
Hey there
Tanay Gottigundala
160
150
140
06/11
06/12
Update Weight
My Info
Height
6′ 3″
Birthday
05/21/1997
Sex
M
Food
Groceries
Workout
You


11:40
98%
Hey there
Tanay Gottigundala
160
Weight
Enter your weight in lbs.
159.0
Cancel
Save
My Info
Height
6′ 3″