

Mohamad Maarouf
CSC400 – My Fitness Diary
Final Report: Documentation

<https://github.com/MohamadMaarouf/myfitnessdiary>

5/17/208

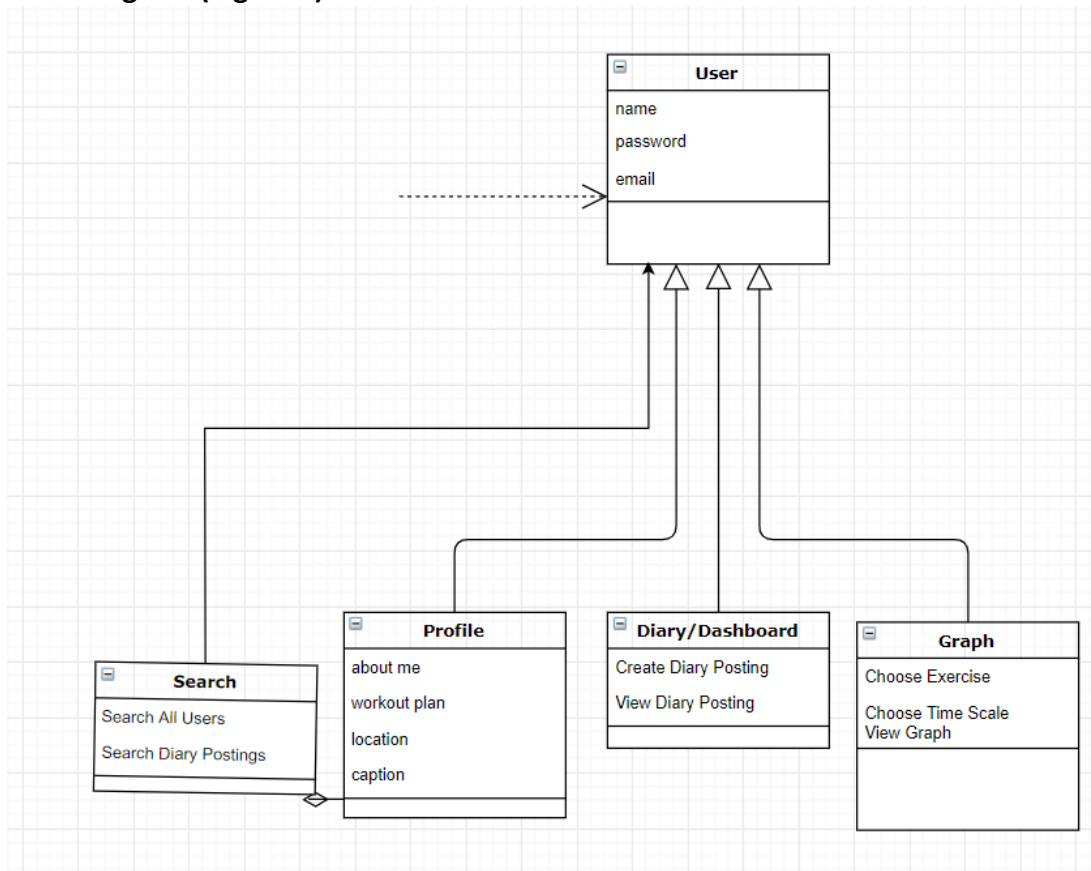
Introduction

The new system that I am creating is a Fitness Diary. The fitness diary will have a few key features including the ability to post a diary of your workout, the power move you are working on, the date you lifted, and how much you lifted for that day. There will also be a graph button which when clicked, will generate a graph of how your weight lifts are increasing week by week. You will also have a profile page where you can write about yourself, what gym you go to, set a workout goal from a choice of three power moves (bench press, squat, and deadlift) and how much you want to improve on those lifts. You will also have the ability to view other people's profiles. There will be a dashboard to help you maneuver through the website. Finally, there will be a search feature where you can search for others members by first or last name, and you can search through your own diary postings by exercise or date posted. Many apps are used to be able to track your progress in a certain lift such as the bench press. In these apps, you can choose the workout and how much you lifted for a specific day. These apps, however, lack the ability for you to have a section where you can talk about the workout. Basically, the fitness diary allows you to not only come up with a goal and track your progress for a specific power move, but to also write about your day and in this way you can keep track of what works and what doesn't work. Many apps do not have a graphing section where you can see your progress in a physical graph. Also, you get to have a profile page and there is a social aspect where you can view other people's profiles. The potential users of this app will be power lifters or anyone who wants to work on the three power moves (squat, deadlift, and bench press) and to set a goal, track their strength, track their mood, write down their thoughts about the workout, and view other people's profile and learn about how they are doing their workouts.

Architecture

Fitness Diary will be a web application created using flask and python. It will be written on vscode and will run on the google cloud. Everything will be in the form of a website, from the landing page, login page registration page, profile page, diary page, dashboard, and the articles page. It will be styled using html, and will have python and flask running on the backend. I will use SQL to hold all the information on login users, diary postings, and profile pages.

Class Diagram (Figure 1)



Within the system, there will be four major attributes. Everyone will be considered a user, as there will be no need for an admin or admin rights, since everyone will have the same capabilities. Of course, admin profiles and users can be created in the future if needed, but all administrative work currently goes on in the background. All users will have a name, password, and email attached to their profile. From the users, each person will have their own profile and diary. They will also have the graphing capability and the search capability. Every user with a profile page will have 4 attributes which are an about me section, a workout plan section, a location section, and a caption section. For the diary, the user will have the capability to either create a diary posting or to view previous diary postings. Also, the user will have a graphing capability. Here, they can choose the workout they want to graph, as well as the timescale they'd like to see (year, month, and day). Finally, users will all have searching capabilities where they can search all users in the system by first name or last name, and they can search all their diary postings by exercise or date.

Database (SQL) (Figure 2)

```
CREATE TABLE users
(
    user_id INT
    AUTO INCREMENT,
    email VARCHAR (255),
    password VARCHAR (255),
    role VARCHAR (25),
    name VARCHAR (255),
    last_login DATETIME,
    PRIMARY KEY (user_id)
);

CREATE TABLE member
(
    user_id INT NOT NULL,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    title VARCHAR(255),
    location VARCHAR(255),
    about VARCHAR(1024),
    url VARCHAR(255),
    goalWeight VARCHAR(255),
    mainExercise VARCHAR(255),
    workoutOne VARCHAR(255),
    workoutTwo VARCHAR(255),
    workoutThree VARCHAR(255),
    verified BOOL,
    private BOOL,
    PRIMARY KEY(user_id),
    FOREIGN KEY(user_id) REFERENCES users(user_id)
);

CREATE TABLE diaryPosting
(
    diary_id INT AUTO INCREMENT,
    member_id INT NOT NULL,
    image BLOB,
    dateOfPost DATE,
    exercise VARCHAR (255),
    overview VARCHAR (1024),
    current VARCHAR(255),
```

The database will hold important information about a user such as their name, password and email. Users will have the ability to register an account which then gets added to the database. From the users, each person will have a profile which creates the member table. Also, each user will have a diary which means there is a diary table where diaries are added to the database every time a new posting is created.

Forms Python File (Figure 3)

```
class Login(FlaskForm):
    email = StringField('Email: ', validators=[DataRequired(), Email()])
    password = PasswordField('Password: ', validators=[
        DataRequired(), Length(min=6, max=255)])
    submit = SubmitField('Login')

class Registration(FlaskForm):
    first_name = StringField('First Name: ', validators=[DataRequired()])
    last_name = StringField('Last Name: ', validators=[DataRequired()])
    email = StringField('Email: ', validators=[DataRequired(), Email()])
    password = PasswordField('Password: ', validators=[
        DataRequired(), Length(min=6)])
    confirm = PasswordField('Confirm Password: ', validators=[
        DataRequired(), EqualTo('password')])
    submit = SubmitField('Register ', validators=[DataRequired()])

class EditProfileM(FlaskForm):
    first_name = StringField('First Name', validators=[DataRequired()])
    last_name = StringField('Last Name', validators=[DataRequired()])
    user_title = StringField('Caption', validators=[Length(min=0, max=50)])
    location = StringField('Gym Location', validators = [Length(min=0, max = 50)])
    about = TextAreaField('About', validators=[Length(min=0, max=255)])
    goalWeight = StringField('Goal Weight:', validators=[DataRequired()])
    mainExercise = SelectField('Exercise',
        validators=[DataRequired()],
        choices=[('Benchpress', 'Benchpress'), ('Deadlift', 'Deadlift'), ('Squat', 'Squat')])
    workoutOne = SelectField('Exercise One', validators=[DataRequired()], choices = [('Incline Dumbbell Press', 'Incline Dumbbell Press'), (
    workoutTwo = SelectField('Exercise Two', validators=[DataRequired()], choices = [('Incline Dumbbell Press', 'Incline Dumbbell Press'), (
    workoutThree = SelectField('Exercise Three', validators=[DataRequired()], choices = [('Incline Dumbbell Press', 'Incline Dumbbell Press')
    url = StringField('Email', validators=[Length(min=0, max=100)])
    private = BooleanField('Private')
    submit = SubmitField('Submit')
```

One important module that is used is FlaskForm and wtfforms. These modules allowed for the creation of the login form, registration form, the edit profile form, and more. These forms would be able to take in data, which can then be used to perform other functions such as graphing and displaying data on the profile page. Also, they were especially useful when it came to using the select field, which allowed the user to have a drop down menu of options to choose from. For example, when choosing an exercise as the user's main goal, they are given a drop down menu to choose from benchpress, squat, or deadlift.

Applications Python File (Figure 4)

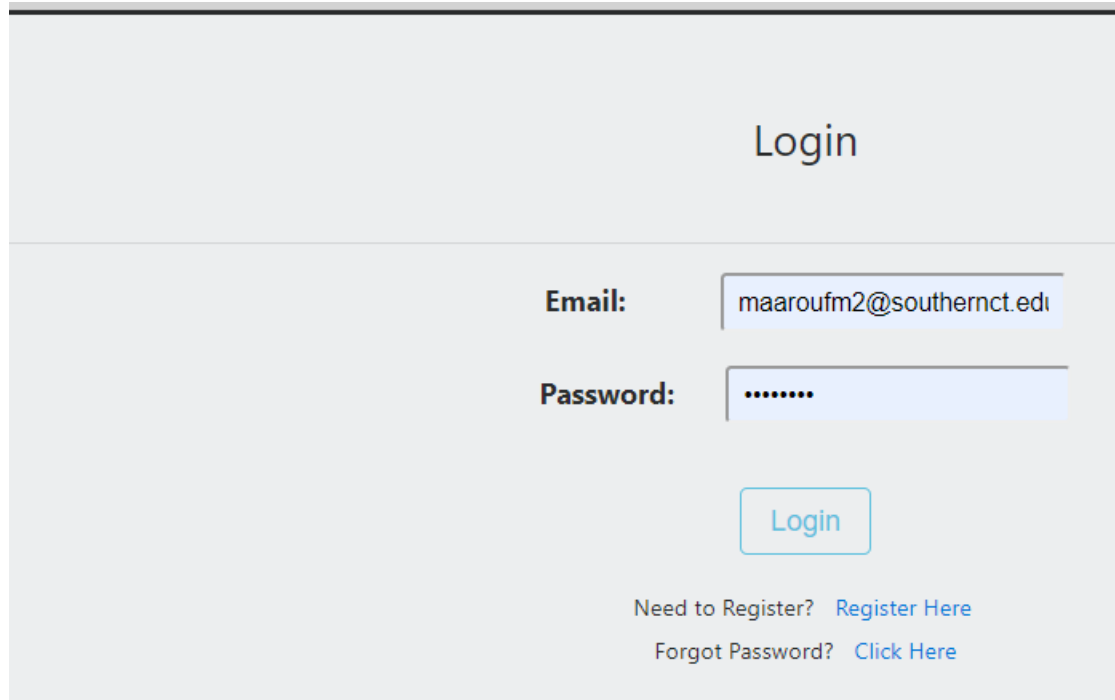
```
@application.route('/login', methods=['GET', 'POST'])
def login():
    if(current_user.is_authenticated):
        return redirect(url_for('dashboard'))
    form = forms.Login()
    if(form.validate_on_submit()):
        # Retrieve Input from Form
        email = form.email.data
        pwr = form.password.data

        if(db.credntial_check(email, pwr)):
            # get user id, email, password, role and name
            sql = 'SELECT * FROM users WHERE email="%s"' % (email)
            row = db.query('PULL', sql)[0]
            user_id = row[0]
            email = row[1]
            password = row[2]
            role = row[3]
            name = row[4]
            last_login = row[5]

            verified = db.query('PULL', "SELECT verified From {} WHERE user_id={}".format(role,user_id))
            # create user object
            if(verified[0][0] == 1): # Block all non-verified users from login in
                user = User(user_id, email, password, role, name, last_login)
                login_user(user)
```

The applications python file is where most of the work is done. It contains all the classes, functions, and routes that get the page running. Seen in the code above, is the route '/login'. This is the route that is called once a user logs in. Also, seen above, is the 'login' function. This controls the ability for users to login with their credentials. It checks to see if the credentials are correct, then pulls the information from the SQL database, and allows the user to login. The application file contains many more functions and routes that are the heart of the web applications functionality.

Login/Register (Figure 5)



The login/register form is centered on a light gray background. At the top, the word "Login" is displayed in a large, dark font. Below it, there are two input fields. The first is labeled "Email:" and contains the text "maaroufm2@southernct.edu". The second is labeled "Password:" and contains a series of dots. Below these fields is a blue "Login" button. At the bottom, there are two links: "Need to Register? Register Here" and "Forgot Password? Click Here".

Login

Email: maaroufm2@southernct.edu

Password:

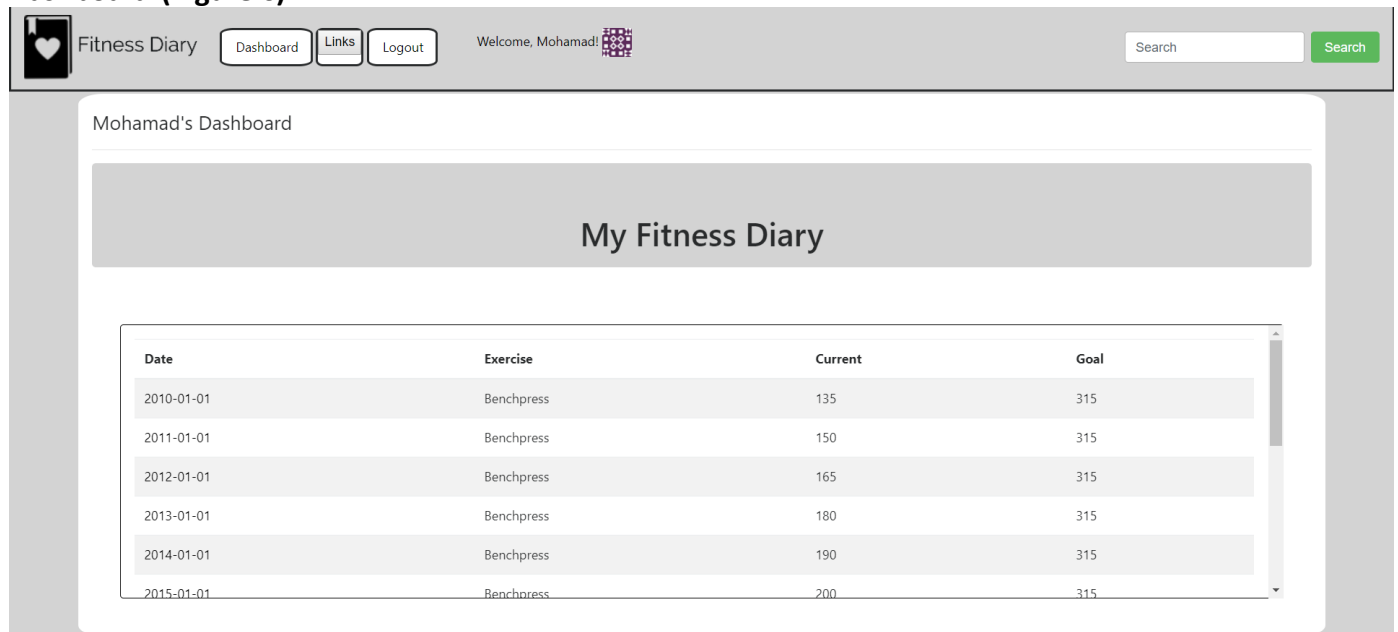
Login

Need to Register? [Register Here](#)

Forgot Password? [Click Here](#)

The user can click 'Register Here' to register a profile, and they can login on this login screen with the credentials they registered.

Dashboard (Figure 6)



The dashboard layout features a top navigation bar with a "Fitness Diary" logo, a "Dashboard" button, a "Links" button, a "Logout" button, and a "Welcome, Mohamad!" message. A search bar is located on the right. Below the navigation bar, the main content area is titled "Mohamad's Dashboard" and "My Fitness Diary". It contains a table with fitness data.

Fitness Diary Dashboard Links Logout Welcome, Mohamad! Search Search

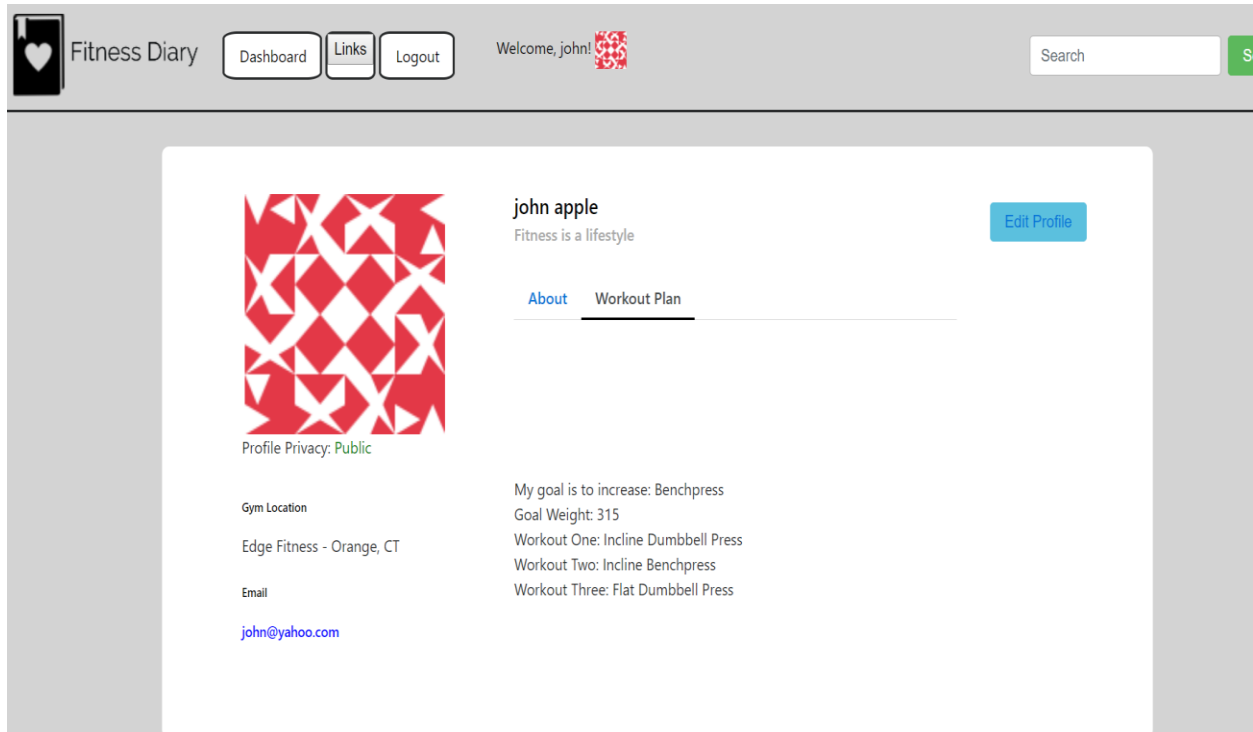
Mohamad's Dashboard

My Fitness Diary

Date	Exercise	Current	Goal
2010-01-01	Benchpress	135	315
2011-01-01	Benchpress	150	315
2012-01-01	Benchpress	165	315
2013-01-01	Benchpress	180	315
2014-01-01	Benchpress	190	315
2015-01-01	Benchpress	200	315

This is the dashboard page or the main page. Here, the user can see all their diary postings. In one example, you can see the date the post was made, the exercise that was trained that day, the current weight they are lifting, and the goal they have set. They also have the option to click on one of the postings to view the full posting, with the description they have written. Located at the top is the navigation bar where they can click to other pages on the web application, logout, or use the search function.

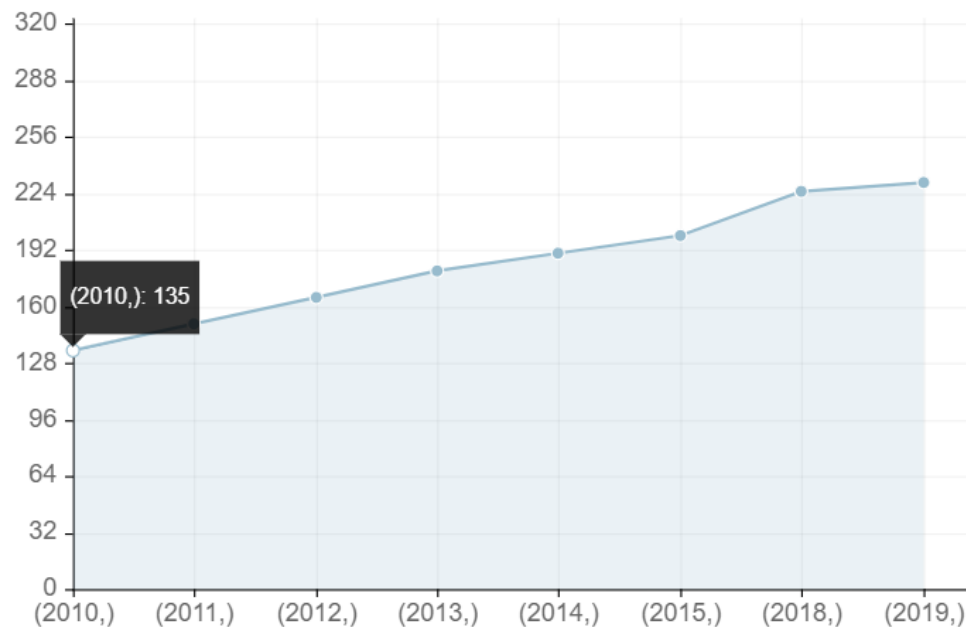
Profile Page (Figure 7)



This is the profile page. Here, the user can see their full name, a caption they have written, the location of their gym, their email, an about me section where they can write whatever they please about themselves, and finally, a workout plan section. In the workout plan section, they can choose which exercise they want to work on, the goal weight they would like to reach, as well as which three exercises they will be using. There is also an edit profile button where they can go to edit all this information.

Graph (Figure 8)

Benchpress Max Growth Over Time



This is the graphing function. Here, the user can choose between the three power workouts (bench-press, deadlift, and squat). They can then choose their timescale (year, month, and day). After they set these options, a graph will be displayed as shown above. This graph will grab data from diary postings, and display max weight growth over time, for that exercise, depending on the timescale they've chosen. Shown above is bench-press growth over the years.

Testing and evaluation/Lessons learned

In order to test my web application, I double checked for the functionality that I specifically coded for. If the functionality did its job correctly, then I knew I was successful. In one instance, I came across the fact that I could not search by the date of the diary post. At first, I thought this was some sort of glitch, but then I looked back the code. I had specifically coded that you can search by exercise and by user's first name, but nothing else. There was no glitch, but rather it was exactly the way I coded it. I went back and edited the code in order to fix this issue. Similarly, I played around with the code to make sure there were no hidden glitches. If I came across one, I'd figure out why and fix it. As mentioned previously, however, most of the testing on the code proved that I had just not accounted for that specific function, and in order to fix it, I just had to edit the code. This taught me a valuable lesson. This lesson is that you don't always account for everything the first time around. You have an idea of how you want the code to work in your head, only to realize that there were some unaccounted for

factors. This isn't necessarily a bad thing, but it proves that you might need to take more time to consider exactly what it is you want to code, how you're going to code it, and if there are any changes you can always go back and edit the code.

Version 2

The web application I have created is a great app for fitness users to keep track of their progress. On top of that, it is a great platform for connecting with others. Considering the fact that you can search for other members in the system, as well as view other profiles, then having further social capabilities would work very well. In the future, I can add the capability to view other people's diary postings. Also, I can add a diet plan section where the user can write down his diet plan and keep track of his weight/body fat, along with other metrics. I can also include a messaging capability, and the ability to follow or friend request these members. Another possibility would be the ability to comment on a person's diary posting. There are many routes you can go with an app like this, and these are just a few ideas with which to start with. In this case, however, we are considering to turn it into a social app. If we would like to keep it an app purely for the users own personal use where they can purely track progress over time, we can add as mentioned previously, the capability to keep track of one's diet and more graphing capabilities.

The Process

Creating this program took a lot of learning and a huge learning curve. There were times where I almost wanted to give up, but I didn't. Often times, I'd come across a small portion of the program, which played a major role. This little problem would sometimes take hours to complete. Other times, I'd try to fix one thing, and end up breaking everything. During these times is when I was most tested as I'd have to revert to a previous version of the program and remake all the changes that I had originally made. Writing the code took a lot of trial and error. Sometimes I'd be prepared for a working code, and it wouldn't work. Other times, I'd be prepared for an error, only to realize I had done it correctly. I also used Google plenty of times for answers. Some skills I had to pick up along the way were SQL, python, flask, html, and more. Although I did have previous knowledge with these programming languages, there would be times that I came across a specific problem I wanted to solve. Often times this included doing some research on how to code for that solution. Sometimes the answer was right at my doorstep, and other times I had to search deeply. Unfortunately, not all the answers I could find. During these times, I found other ways of figuring out the problem. For example, I'd add print statements in certain portions of the code, and this helped with troubleshooting as I can see where exactly the code would break. Other times, I'd make changes little by little. I'd write a working code, then add on to it piece by piece until it broke. Once I figured out exactly which code was breaking everything, I could then go on to figuring out the solution or how to fix it. Once in a while when I'd come across a problem I couldn't solve, which happened around twice, I'd reach out to an old teammate. This was only in times of dire necessity. Often times, having a peer look at your code, or even just to talk things out with them, helps you find a solution. One major issue I faced was when it was finally time to deploy my code. I tried downloading SDK so I can deploy my web application on the Google Cloud, but I was met with an error and could not download it. I, instead, began deploying on the AWS. After hours

figuring out how to deploy, I was met with an error that I simply did not know the solution for. I trashed the whole idea and tried Google Cloud one more time. The SDK downloaded this time and began the process to deploy on the Google Cloud. This took help from a teammate, but I was able to figure it out and deploy successfully. This project surely helped me increase my knowledge in the tools I mentioned previously, and I am now more confident in my coding capabilities as well as my problem solving capabilities.