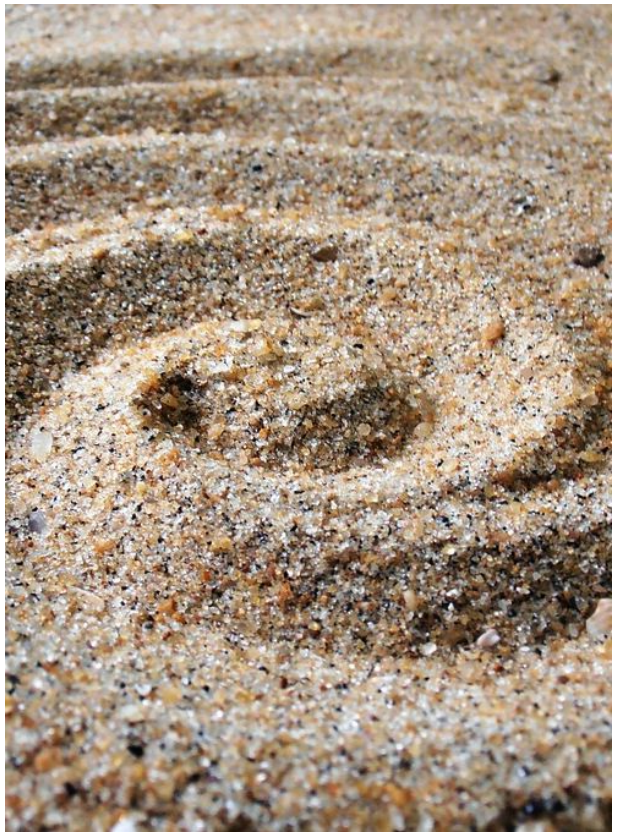
It's not just an app, It's a Lifestyle

Group 1

- Team Lead: Jonathan Sullivan
- Design Lead: Bob Allan
- Test Lead: Sam Bauter

Design

- For our mock-up I used Adobe XD
- There are many resources available to help create realistic Android designs

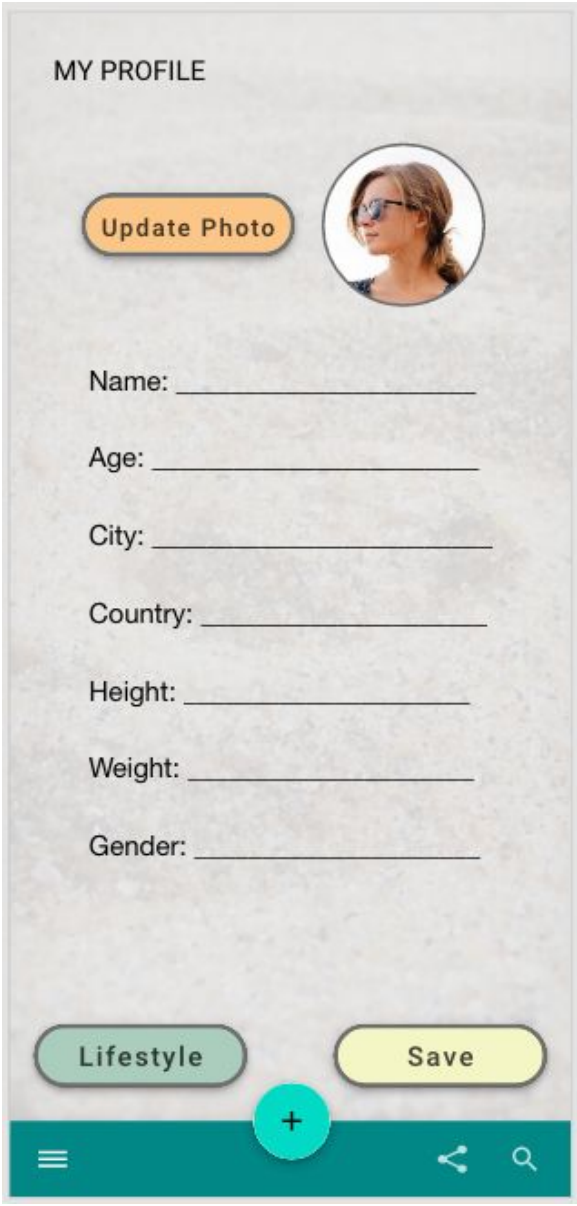


Design

- We want our app to be clean, colorful, and easy to use.

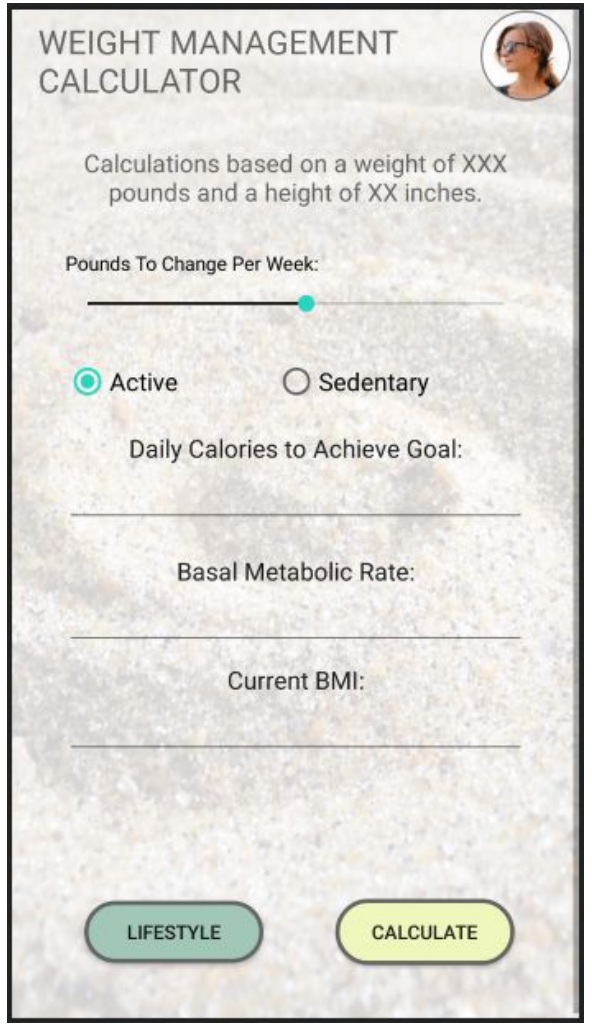
Design

- Vertical-only orientation
- There are many popular fitness apps, like Strava and MyFitnessPal that are vertical-only



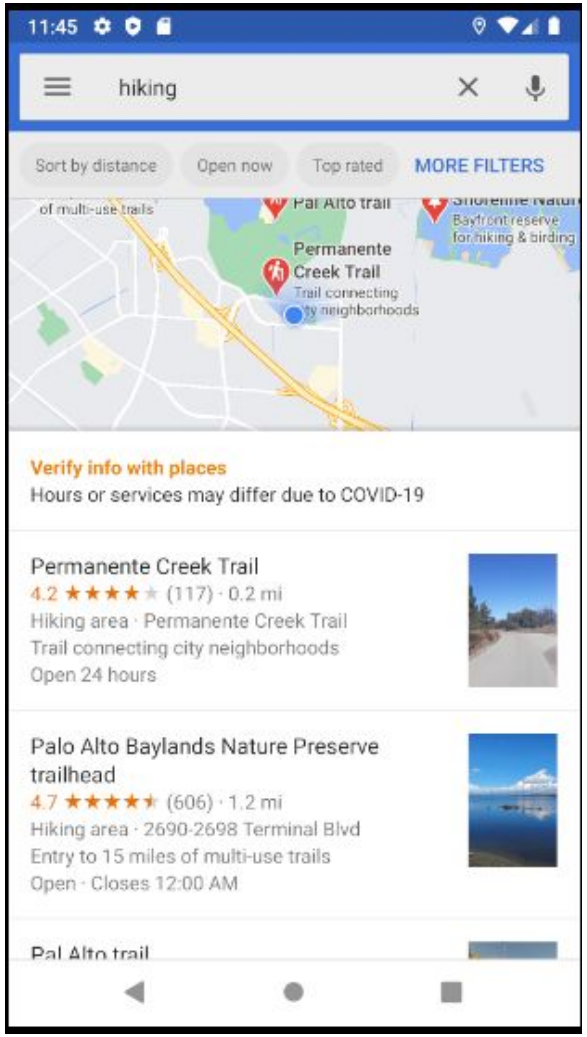
Design

- BMI button is unnecessary, the calculation can be done in with the profile data.
- Our design has evolved to meet client requests by adding sliders and removing text fields.



Design

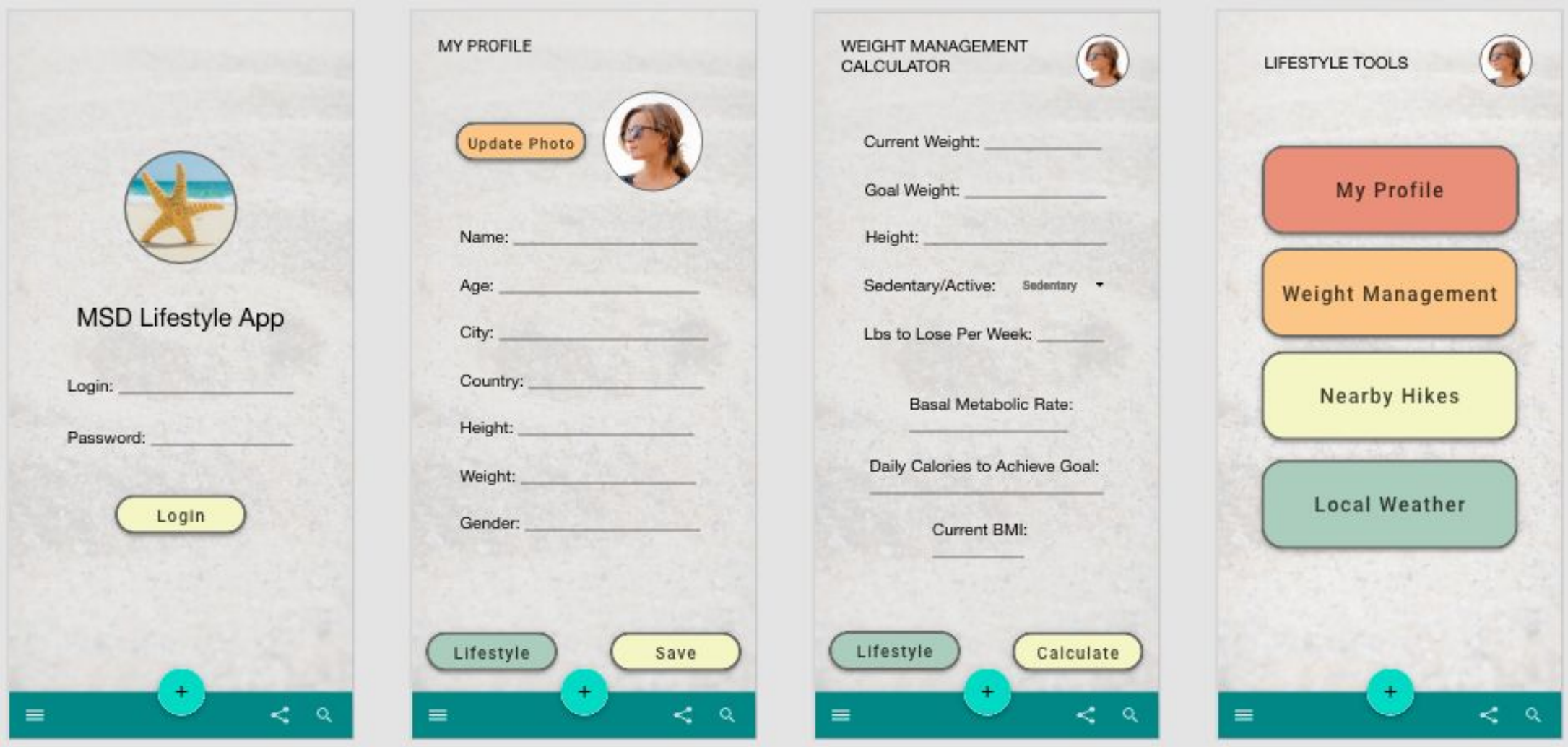
- Google Maps API is as easy as pie



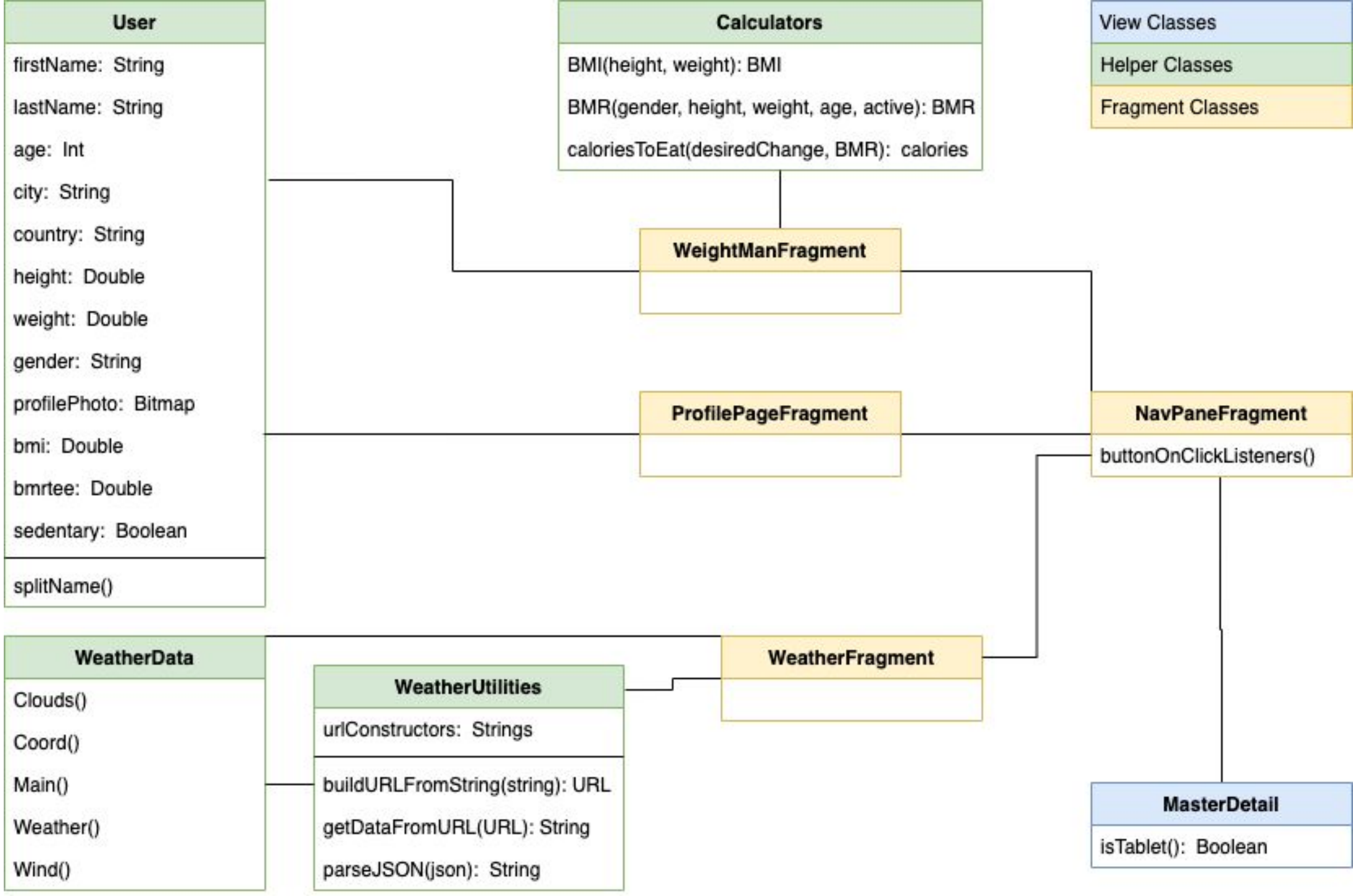
Design

- Weather API is not as easy as pie, but it will look nice once we sort it out

Design


MSD Lifestyle App
Login:
Password:
Login
MY PROFILE
Update Photo
Name:
Age:
City:
Country:
Height:
Weight:
Gender:
Lifestyle
Save
WEIGHT MANAGEMENT CALCULATOR
Current Weight:
Goal Weight:
Height:
Sedentary/Active:
Sedentary
Lbs to Lose Per Week:
Basal Metabolic Rate:
Daily Calories to Achieve Goal:
Current BMI:
Lifestyle
Calculate
LIFESTYLE TOOLS
My Profile
Weight Management
Nearby Hikes
Local Weather

Class Diagram



Class choices

- User data class
  - Allows for additional future users
  - Kotlin automatically generates getters/setters
- Weight-related calculator helper class
  - Static functions that can be called from anywhere
- Weather data class
  - Easier to parse JSON with Gson
  - Supported by weather helper class functions

Bugs and their fixes

- Activities were not scrollable on smaller screens
  - Include ScrollView tags
- Profile male/female radio buttons not radioing
  - Do not put a LinearLayout inside of a RadioGroup, instead use the orientation attribute of RadioGroup
- Camera crashing when attempting to launch from profile activity
  - Does not actually seem to be app-related. Possible issues with emulators / API versions.
- Input field text does not go away when field selected
  - Use “hints” rather than “text”
- Managing GitHub for multiple users is complicated
  - Branches are your friend

Tests and testing

Two Main types of test in Android Development:

- Normal JVM Testing
  - Tests simple functionality of Java helper classes and other non-Android API tests.
  - Faster Runtimes! Only requires the JVM to run testing rather than emulating a full android environment.
- Instrumented Tests
  - Versatile testing strategy that allows you to test Android features and capabilities
  - Can test user interactions with the device.
  - Ex: ViewMatcher methods: withId, withText, isDisplayed, isChecked etc
  - Ex: ViewAction methods: click, typeText etc

A Simple Instrumented Test

```
import androidx.test.espresso.ViewInteraction;
import androidx.test.ext.junit.rules.ActivityScenarioRule;
import androidx.test.ext.junit.runners.AndroidJUnit4;
import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;
import static androidx.test.espresso.assertion.ViewAssertions.*;
import static org.hamcrest.Matchers.*;
import static androidx.test.espresso.Espresso.*;
import static androidx.test.espresso.action.ViewActions.*;
import static androidx.test.espresso.matcher.ViewMatchers.*;

@RunWith(AndroidJUnit4.class)
public class MasterDetailClickTests {
    @Rule
    public ActivityScenarioRule<MasterDetail> masterActRule = new ActivityScenarioRule<>(MasterDetail.class);

    @Test
    public void clickMyProf(){
        ViewInteraction moveToMyProfBtn = onView(withId(R.id.my_prof_btn_frag));
        moveToMyProfBtn.perform(click());
        onView(withId(R.id.title_my_prof_frag)).check(matches(withText(containsString( substring: "MY PROFILE"))));

    }



}
```