# WorkoutApp: Wearables Final Project

## Introduction:

### Background:

Fitness trackers are increasingly popular, using wearable devices to track workouts. Many people such as myself find these very valuable. iOS Devices such as the iPhone are essentially a wearable device, as they are carried everywhere with a person, and contain GPS location, gyroscope, & accelerometer sensors.

The fitness app market is valued at $4.4 Billion, and is expected to expand at a compound annual growth rate (CAGR) of 21.6% from 2021 to 2028. (Source: https://www.grandviewresearch.com/industry-analysis/fitness-app-market)

🚴🏃 📱 ⌚

My background is more in UI development and app development, so I decided to focus on this area of wearables, rather than machine learning or big data. Using iOS API's is a counterpoint to sensor tile development in this course, the former is very robust and easy to work with as a starting point.

### Goal:

Create a iOS fitness app to give user performance feedback and workout tracking, using iPhone motion sensors.

The app will read perfjoamcne data from an iPhone, and display it back to the user. Keep the app simple, focused on the goals of distance and speed.

References: Strava, AllTrails, Nike Run app, Peloton, Apple Fitness+.

## Team:

Christian Hilton
CTO, Product Manager, Engineer

## Milestones

### 7/12 Week
- Proposal
- Create Repo & Xcode project
- Research Workout app UI and data
- Create basic UI design
- Test motion data access from iPhone device

### 7/19 Week
- Create UI
- Integrate motion data with UI
- Create working prototype

### 7/26 Week
- Add additional features
- Test & iterate
- Write report
- Write slides
- Submit report

### 8/4
- Final presentation

# Technology:

## Software and Developing tools:

Software Language: Swift

Github: https://github.com/christian-hilton/workout_app_project

IDE / Build system: Apple Xcode (Version 12.5.1)

## Hardware & Sensors:

iPhone XR
- Sensors: Gyroscope, accelerometer, GPS
- - run iOS app locally on iPhone

macbook pro
- MacOS Big Sur 11.4
- Xcode Version 12.5.1
- USB-C cable (connect to iPhone)

## Frameworks:

Core Motion: Process accelerometer, gyroscope, pedometer, and environment-related events.
- Note: pedometer data was most useful.

Core Location: Obtain the geographic location and orientation of a device.
- Note: final app did not end up using this.

## Resources:

In addition to class materials, the following resources were used to develop the app.

Apple Developer:

Core Motion: https://developer.apple.com/documentation/coremotion

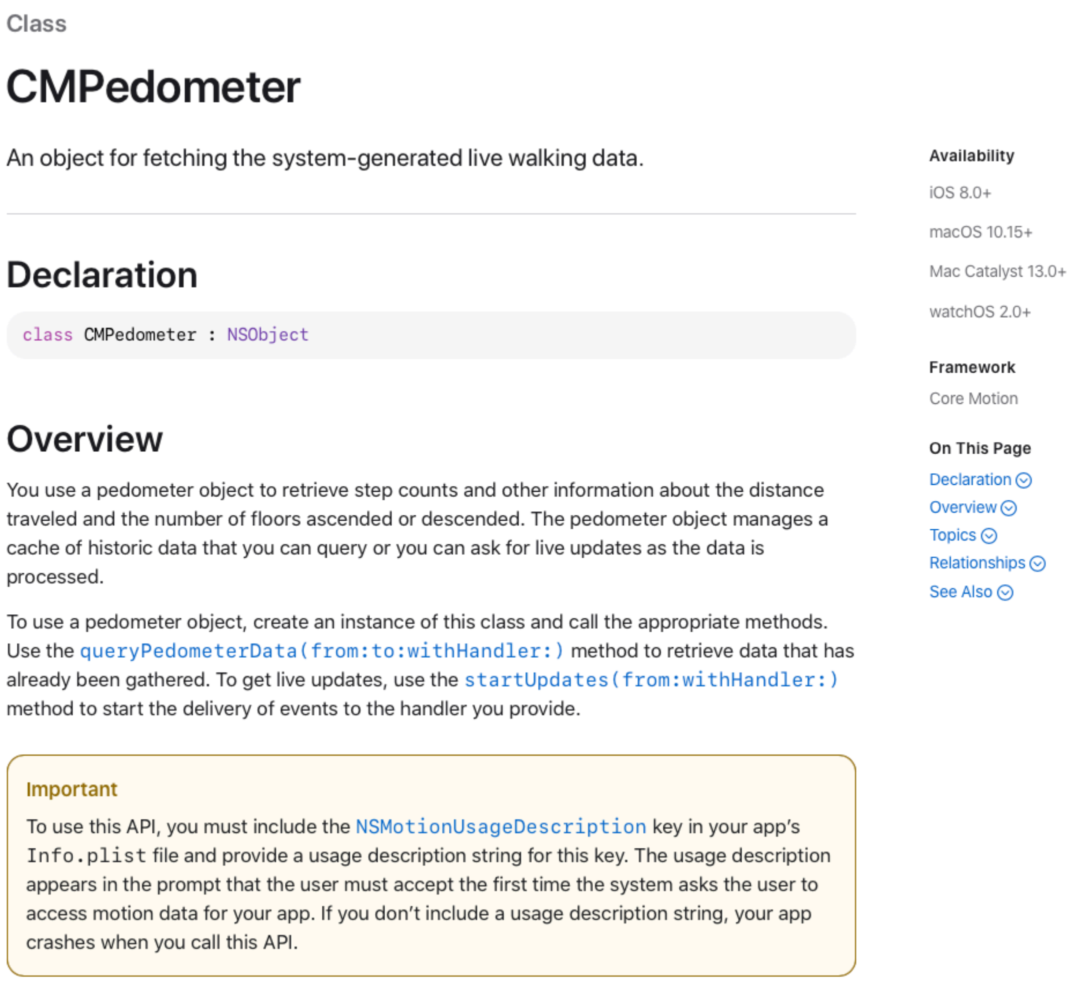
Getting Processed Device-Motion Data
https://developer.apple.com/documentation/coremotion/getting_processed_device-motion_data

CMPedometer:
https://developer.apple.com/documentation/coremotion/cmpedometer

## How does Apple's pedometer work?

"counting the number of positive peaks between comparison threshold crossings, adjusting a minimum peak-to-peak threshold for qualifying threshold crossings, and inferring a second step based on the amount of time between threshold crossings. In some implementations, the pedometer can automatically determine that the pedometer is being worn on a user's wrist."

Patent: Wrist Pedometer Step Detection
https://appft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&u=%2Fnetahtml%2FPTO%2Fsearch-adv.html&r=1&p=1&f=G&l=50&d=PG01&S1=(702%2F160.CCLS.+AND+20140313.PD.)&OS=ccl/702/160+and+pd/3/13/2014&RS=(CCL/702/160+AND+PD/20140313)



Class

CMPedometer

An object for fetching the system-generated live walking data.

Declaration

```
class CMPedometer : NSObject
```

Overview

You use a pedometer object to retrieve step counts and other information about the distance traveled and the number of floors ascended or descended. The pedometer object manages a cache of historic data that you can query or you can ask for live updates as the data is processed.

To use a pedometer object, create an instance of this class and call the appropriate methods. Use the `queryPedometerData(from:to:withHandler:)` method to retrieve data that has already been gathered. To get live updates, use the `startUpdates(from:withHandler:)` method to start the delivery of events to the handler you provide.

Important

To use this API, you must include the `NSMotionUsageDescription` key in your app's `Info.plist` file and provide a usage description string for this key. The usage description appears in the prompt that the user must accept the first time the system asks the user to access motion data for your app. If you don't include a usage description string, your app crashes when you call this API.

Availability

iOS 8.0+

macOS 10.15+

Mac Catalyst 13.0+

watchOS 2.0+

Framework

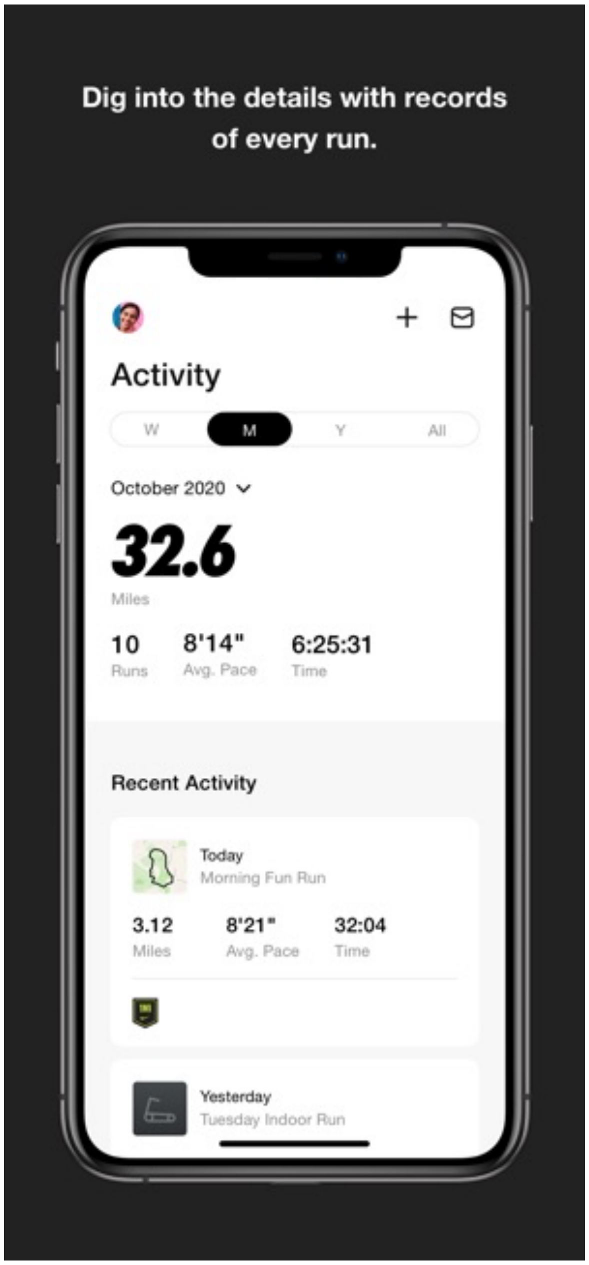
Core Motion

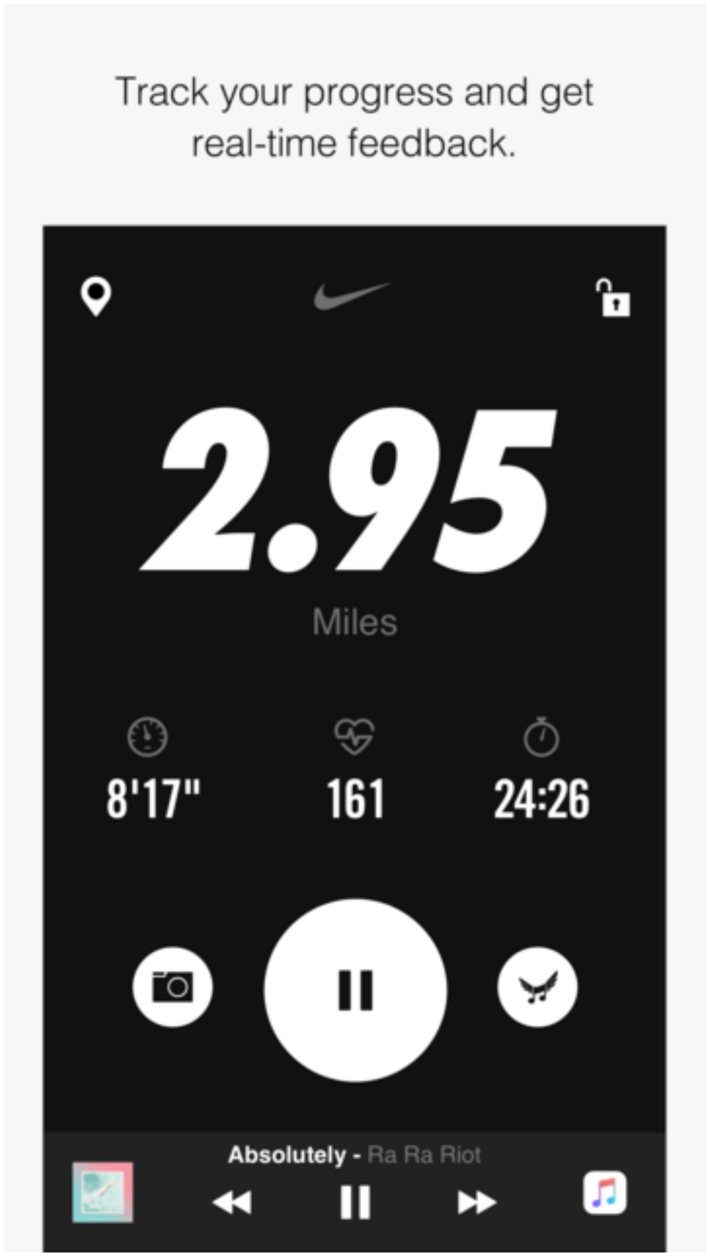
## Product Research:

I researched existing iOS Fitness tracker products, including Strava, Nike Run Club, Peloton, & Apple Fitness+.

Nike Run Club was my favorite design, it was both elegant and simple.

Nike Run Club screenshots:





# Functional Requirements & Features

## Features:

1. Select Workout
- Users should be able to select between running or biking

2. Start Workout
- User should be able to start a new workout

2. Pause Workout
- user should be able to pause the workout and restart it

4. End Workout
- User should be able to end a workout, adding it to the list of completed workouts.  after this they can start another new one.

View Speed
- While workout out, user should be able to view their speed (m / hr).

View Acceleration
- while workout out, user should be able to view their acceleration (m / hr / s).

View timer
- While workout out, Users should be able to view a timer of the duration.  The timer pauses when they hit pause, and stops when they hit stop.

View overview of past workouts
- Users can view summary of recent past workouts, if there are any.

Feedback string regarding workout performance
- Users receive a message regarding their performance.
Example: “0 Workouts so far this week, let’s get started”
“Awesome streak with 5 workouts this week, keep it up!
“New personal record: best speed: 12 miles / hr!”

## Challenges:

Several challenges were encountered during the development process.

1. Errors:

Example: Authorizing data from user and providing explanation is a requirement.

```
2021-07-31 17:12:38.196706-0700 ExerciseApp[6028:1457708]
[access] This app has crashed because it attempted to
access privacy-sensitive data without a usage description.
The app's Info.plist must contain an
NSMotionUsageDescription key with a string value explaining
to the user how the app uses this data.
```

2. Imprecision of GPS Data

Calculating distance from GPS location was unsuccessful. it was too unreliable. Instead I pivoted to using the pedometer data.

3. Too many ideas.

I had a lot of dreams for a workout out that were not accomplishable during this short timeframe. See list of future developments.

4. Too little time

Refining UI can be very time intensive. I had to minimize amount of time spent on this.

## Data:

There were several different data objects read from the iPhone.
Below examples of data that was processed.

### CLLocation:

The latitude, longitude, and course information reported by the system. (https://developer.apple.com/documentation/corelocation/cllocation)

```
<+37.76843873,-122.44815369> +/- 14.20m (speed 0.42 mps /
course 148.29) @ 7/31/21, 3:45:25 PM Pacific Daylight Time

<+37.76844741,-122.44815209> +/- 13.95m (speed 0.42 mps /
course 106.12) @ 7/31/21, 3:45:26 PM Pacific Daylight Time
```

### CMDeviceMotion:

Encapsulated measurements of the attitude, rotation rate, and acceleration of a device.

```
Optional(x -0.144440 y 0.224274 z 0.360016 @ 177439.619961)

Optional(x -0.062149 y 0.396927 z 0.273666 @ 177440.625387)
```

### CMPedometerData

Information about the distance traveled by a user on foot.

```
CMPedometerData,<startDate 2021-08-01 00:49:02 +0000
endDate 2021-08-01 00:49:29 +0000 steps 18 distance
16.49390449002385 floorsAscended 0 floorsDescended 0
currentPace (null) currentCadence (null) averageActivePace
0.6389671109332803>

CMPedometerData,<startDate 2021-08-01 00:49:02 +0000
endDate 2021-08-01 00:49:31 +0000 steps 18 distance
16.49390449002385 floorsAscended 0 floorsDescended 0
currentPace (null) currentCadence (null) averageActivePace
0.6389671109332803>
```

## Methodology:

### Structure:

The following is the structure of the WorkoutApp codebase:

MainViewController: Main landing page, contains all UI.
Main.Storyboard: layout for main landing page

UITableViewController: Used to display list of workout metrics in table format.

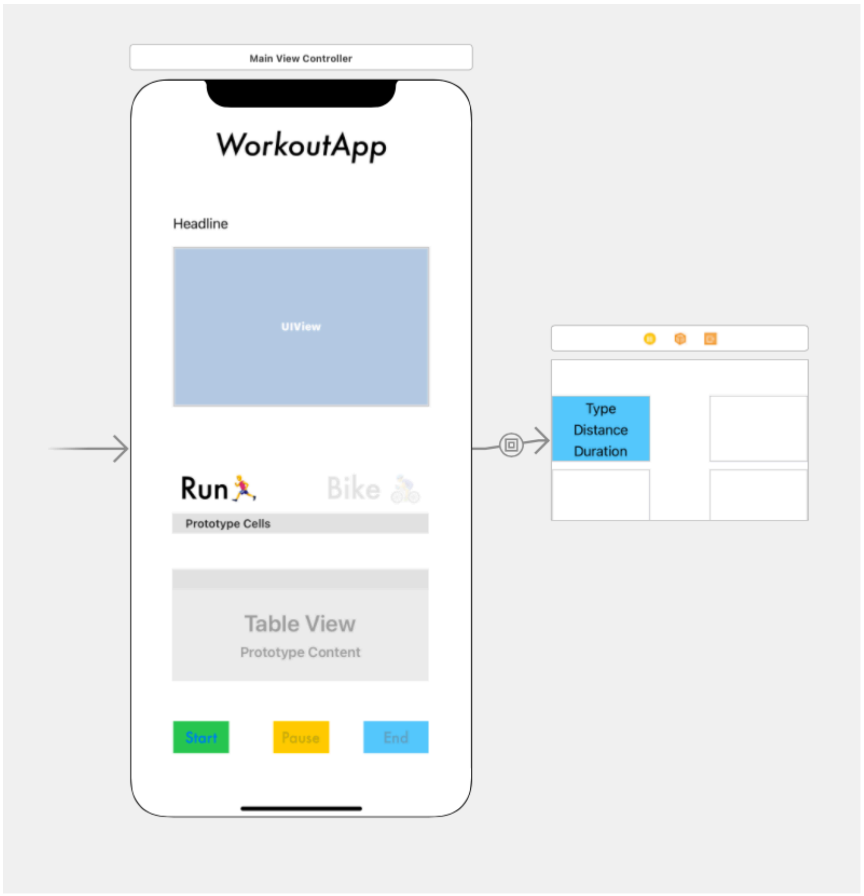
UICollectionviewController: Used to display list of previous workouts in grid format. Similar to TableViewController, but more flexible.

WorkoutSession: Model data for

### Patterns & Techniques:

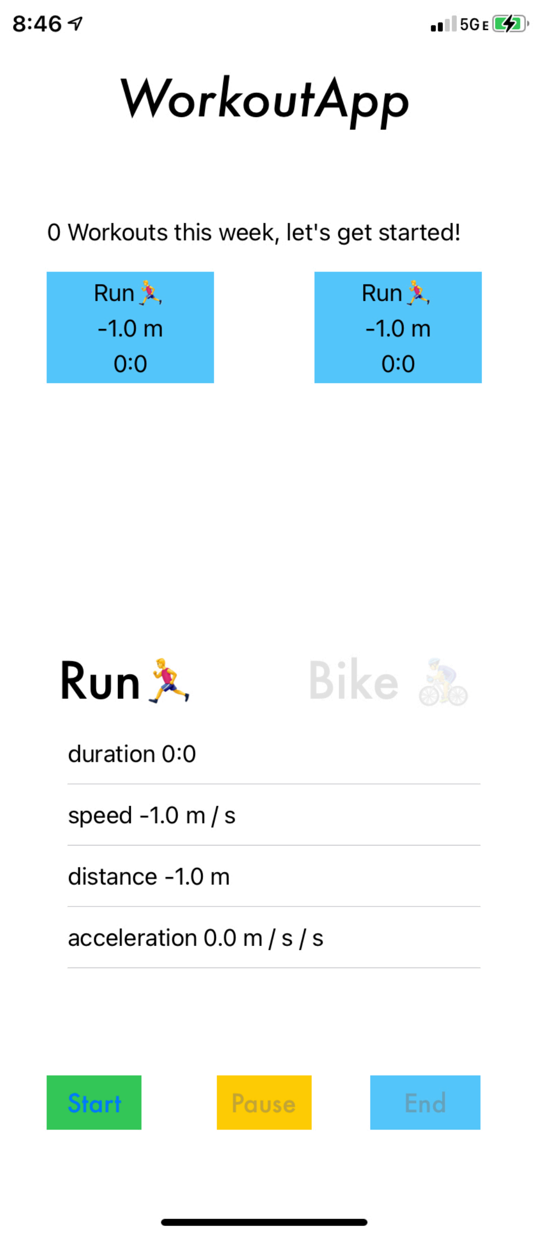
Xcode Interface Builder / Storyboards:
iOS development leverages Storyboards for layouts and connects them to code via outlets and actions.

Model - View - Controller
Separation of the Model Data, the View, and the Controller



## User Interface

User Interface for WorkoutApp



8:46

5G E

WorkoutApp

0 Workouts this week, let's get started!

Run🏃‍♀️
-1.0 m
0:0

Run🏃‍♀️
-1.0 m
0:0

Run🏃‍♀️ Bike 🚴

duration 0:0

speed -1.0 m / s

distance -1.0 m

acceleration 0.0 m / s / s

## Codebase:

See GitHub: https://github.com/christian-hilton/workout_app_project

### Instructions to run:

Required: Current Xcode macOS app.

Download code from GitHub.

Open ExerciseApp.xcodeproj file.

Connect iPhone or iOS device to Mac via USB.

Select iOS device as target for build.

in Xcode, Product > Run.

ExerciseApp opens on device.

## Conclusion:

Several conclusions are drawn from this project.

### Location Data:

GPS Location data was not reliable enough to use for distance tracking in my experience. This fluctuated too frequently and led to inaccurate readings. Instead, using the built in pedometer function of the iPhone was much more accurate.

### iOS Development:

These powerful frameworks built into iOS are a contrast to the SensorTile. It was relatively quick to receive data from

### Future development:

With additional time and resources, this project could be expanded to become more feature rich.

### Future development:

Due to time constraints, many ideas were reserved for future development.

1. Data persistence across sessions
- No permanent database, instead just a local session.

2. Multiple pages
- focus on single page to minimize complication

3. Distribution
- Currently the app is only run locally on device, rather than distributed through the app store.

Additional workout types
- Currently only running or biking are supported. Future workout types such as swimming,

Watch
- Currently the app is only run on iOS, including iPhone and iPad. In the future a watch app could be supported.

Improved accuracy
- Accuracy of the velocity and acceleration has not been fully tested and might have potential for improvement.

Machine learning
- Machine learning could be used to provide analysis of workout data

Dynamic feedback
- Currently feedback messages are quite simple and static. In the future these could be more complex and dynamic in order to provided ongoing feedback that helps improve workouts.

## Summary:

This course provided a survey of a wide range of different technologies, from digital signal processing, to motion detection, to machine learning on real world data. My interest lies more in the User Interface rather than the raw data and machine learning. But there is great opportunity to provide user interfaces for wearable devices, where the interface is especially personal.

Within the span of a 2 week timeline, I launched a function personal fitness tracking app, WorkoutApp! This was definitely a learning experience.

While sensors can read a vast amount of data from the real world, it is not equally valuable. Pedometer data proved the most useful for my goal. Additionally, data is not valuable unless you know to use it.

Wearable technology and software is an exciting field, and I would love to participate in it further.