Health and Fitness Smart Device

Group 6

Prepared By: Amber Haynes, Tiyon King, Jenna Krause, Mya Odrick, Devvrat Patel, Andrew Rezk, Maria Rios, Shivani Sunil, Hedaya Walter

https://sites.google.com/scarletmail.rutgers.edu/betterfitnesscom/home?authuser=1

February 23, 2020 Software Engineering

Table Of Contents

Project Ownership

Calendar Sync & Workout Reminders

- Tiyon King & Jenna Krause

Calorie Tracker

- Mya Odrick, Maria Rios, & Hedaya Walter

Music App Integration

- Amber Haynes & Shivani Sunil

Safety & SOS Feature

- Devvrat Patel & Andrew Rezk

Individual Breakdown

			Amber Haynes	Tiyon King	Jenna Krause	Mya Odrick	Devvrat Patel	Andrew Rezk	Maria Rios	Shivani Sunil	Hedaya Walter
Project management		10									
Customer Problem Statement	Prob. statm.	5		20%	20%			20%	20%	20%	
	Glos sary	4				100%					
System Requirements	Funct. req'ts	2		25%				25%		25%	25%
	Nonfct. req'ts FURPS+	2		25%				25%		25%	25%
	UI req'ts	2		25%				25%		25%	25%
Functional Requirements Specification	Stkhld. Actors Goals	2		14%	14%	14%		14%	14%	14%	14%
	Use c. casual descr.	8	11%	11%	11%	11%	11%	11%	11%	11%	11%
	Use case diag	5			100%						

	Use c full descr.	10	25%		25%	25%	25%				
	Sys. seq. diag	5	25%		25%			25%	25%		
User Interface Specs	Prelm design	4				25%	25%		25%	25%	
	Effort estim	11				50%			50%		
Domain Analysis	Conc epts	10	25%		25%	25%	25%				
	Assoc	5	25%		25%	25%	25%				
	Attrib	5	33%		33%	33%					
	Cont racts	5	50%						50%		
Project Estimation			33%	33%					33%		
Plan of Work		5		50%		50%					
Ref’s		5	33%	33%					33%		

Customer Problem Statement

Glossary of Terms

Alert - An automated message containing the user’s location and heartbeat. An alert can be sent to the user’s most important contacts in the event that the user feels unsafe.

Calorie Deficit - A calorie deficit occurs when the amount of calories consumed is less than the amount needed to maintain current body weight. If the user’s goal is to lose weight, we will recommend a caloric deficit diet and workout. We will suggest low-calorie meals and calorie-burning exercises.

Calorie Surplus - A calorie surplus occurs when the amount of calories consumed is greater than the amount needed to maintain current body weight. If the user’s goal is to gain weight or muscle, we will recommend a caloric surplus diet and workout. This means we will recommend healthy high-calorie food items and muscle gaining exercises.

Favorites List - The user can add specific meals and songs to their Favorites List, which allows them to access those items quicker. Upon clicking on one of their favorite items, they will receive information like the number of calories in the meal, or the best exercises to perform while listening to their songs.

Freetime - Freetime is defined as a block of time where the user doesn’t have anything planned. The user can manually enter their free time and we will suggest the user to exercise during that time.

Reminder - The user will receive notifications as reminders to suggest them to stick to their scheduled workout, input their daily calorie intake, and incorporate certain food or exercises to meet their goals. However, these reminders can be manually turned off.

Tempo - The tempo of a song refers to the speed or pace of the music. We will use the user’s desired workout and recommend music based on the tempo, as well as the user’s preferred genre of music.

Proposal

We will be basing the general platform from that of group number five in the 2014 Fall Semester. The previous group focused on using the Fitbit to educate the users, and help them maintain a healthy lifestyle; however, we want to motivate users to continue to use their devices when they lose their drive to workout or eat healthily.

The user will be able to sync their Google calendar to the watch. This would allow the user to identify specific workout times. This feature, though optional, is best used for those who input their daily schedules into their calendar. The purpose of the feature is to better motivate the user to implement workouts on a greater basis. Using this feature, the device will seek out the free time, the time when no event is planned, during the day that could be used for exercising. The user will be able to select a range of time in which the feature may look for free time, such as 8 AM to 10 PM. We will be implementing a reminder system to motivate the person wearing the device to stick to their already scheduled workouts, as well as use their free time to exercise on the days where no workouts have been already scheduled. These reminders are simply to workout, whether that be at home or at a nearby fitness center; the feature does not specify. When these reminders occur, the user will have an option to snooze it - like an alarm - for a later reminder, check-in and say that they are working out, indicate that they are at an event that was not inputted, or indicate that they do not wish to workout that day.

Another added feature will be an improvement of the calorie tracker. To motivate the user to continue their healthy lifestyle, the new calorie tracker will send the user food and workout recommendations based on their calorie intake and outtake ratio. Depending on their weight goals, they will be proposed ways to either burn or consume more calories. To motivate the user to use this function, we will send out a reminder for them to input their daily calorie intake and send notifications regarding if they met their goal for the day.

Another added feature will be a virtual workout music playlist curator for users. This feature will take information from users involving their preferred types of workouts and times which they like to workout, and curate playlists for them for different workout days based on the beats per minute (bpm) of the songs. In contrast to other features currently on the market, this part of the site will closely relate the type of exercise (and intended time spent doing the exercise) to decide which and how many songs to add to the playlist. For example, if a user says that they want to jog for 15 minutes then walk for 20, a playlist will be created with 15 minutes of fast-paced music (from their preferred genres) and 20 minutes of slower paced music that they can play when they workout. This also will keep the user on track when they are working out. This also opens the opportunity for our feature which is connected to this, a feature that rewards users with online subscriptions and other prizes based on if they complete their workout.

Problem Statement

As a consumer, it becomes very difficult to find time throughout the day to work out. Whether working or in school, incorporating workouts into an already busy schedule becomes stressful and discouraging. In addition to finding time, it also becomes hard to stay motivated to stick to exercising during the time I've devoted to it. Using the calendar and reminder features on the device, the overwhelming and discouraging aspect of finding the times to dedicate to exercise are taken out of the equation. This relief combined with the additional feature of motivational reminders for the time I had already dedicated to working out take away major factors that previously prevented me from sticking to the plan.

For someone who uses their Google calendar to keep track of events and their schedule, being able to easily incorporate my already created calendar into a new device makes the experience seamless and much less stressful. The only factors that I will have to worry about are selecting the range of time in which the reminder feature will look for free time day-to-day and the amount of time needed in order to exercise, making the set-up experience quick and painless. Picking the amount of free time needed in order to exercise will give me the flexibility for whether I work out at home or if I go to the gym to work out. Once the range of times and amount of time needed to exercise are set, I will receive motivational reminders or notifications informing me of when I have enough free time to incorporate a workout into my schedule.

These features will allow me to be better motivated in order to implement workouts into my schedule on a greater basis. As most know, constant notifications can become quite annoying, usually prompting me to turn the reminders off completely. When the reminders occur, I will have the option to snooze it - like an alarm - for a later reminder, check-in and say that I am working out, indicate that I am at an event that was not inputted, or indicate that I do not wish to workout that day. The ability to snooze or dismiss the reminders and notifications with a wide range of options will lower the chance that I will disable the feature completely.

As a user, I need a way to keep track of my intake of calories and keep track of how many calories I've burned. I want to be able to see my progress and check how much I should be eating based on the exercises I complete. I can input my details into the app at the beginning of my fitness journey such as my current weight, height, age. I can let the app know my ultimate goal and the app can suggest different foods I can eat based on other users that have similar details and have had success. The calorie tracker feature will incorporate a search bar to make it easier for me to search up the foods that I am eating and will give me the number of calories each food contains. The calorie tracker feature can then use this information to calculate how many total calories I've consumed.

In addition, it will be difficult for me to know which exercises are better suited for me and how effective it will be. I will rely on the app to give me different exercise suggestions based on those that have worked for other users with similar profiles. I can choose which ones to follow and put that information into the app. For example, if I want to gain muscle, I will receive food recommendations that will help me achieve this. Likewise, if I want to lose fat, I will be given the appropriate recommendations. These recommendations will save me time and make it easier for me to make the right choices.

Because I have a busy schedule and can be forgetful at times, I will depend on the app to send me a reminder at the start of the day to let me know what type of exercises I should complete. I can adjust the frequency of the reminders so I could choose to have the app remind me at the meal times that I input into the system or just have it remind me at the start of the day. At the end of each day, the app will let me know the progress I've made that day. I will receive a notification of what I've burned off and what I have consumed. Seeing the progress I've made will motivate me to keep working hard towards my goals

We will work on a feature that is necessary for those meaning to exercise with the watch on. For those schedules that permit running late at night or early morning, it's important they feel safe. So this pair will send a reminder to the user that this is an option and make sure that when it is implemented that it triggers a set of commands in the following order: Obtain all possible information such as time, place, etc. Then notify the select list the user creates. This will be a safety feature that will give the user an easy option to get help when needed during an emergency.

This feature helps inform the user's closest friends and family about any alarming concerns if the user deems it necessary. This is ultimately to give the user a sense of safety. Some of the emergency features will include emergency calling (if the watch supports it) and notifying emergency list if the user is in danger but, unable to call themselves. This feature is really important as we want to make sure that everyone who uses our app can feel safer. A lot of elderly people would also find this particularly helpful as there are many cases where they'd be in trouble and found it really hard to reach someone. This feature can also be used in a dangerous situation when people don't have access to their phones and clicking a button on their Fitbit is much easier to access. This feature will not only share that the user is in danger but also share their live location to the select list of people. One other feature we would have is that alerting the close friends list even when the user cannot. For example, the app will know if the user is having a health issue based on their breathing pattern. This way if the user is having an emergency then the watch will automatically alarm their close friends and also call for help if that is selected. The app will also have a record of the surroundings and the user's running location so if there is anything wrong going outside.

User Stories

Table 1-1: Calorie Tracker Requirements

ID	Priority Weight	Requirement
REQ-1	3	As an authorized user, I will be able to take a picture of my food to input calorie intake.
REQ-2	2	As an authorized user, I will be able to enter my fitness goals.
REQ-3	5	The systems will suggest meals and exercises based on the goals of the user.
REQ-4	7	The system will use the user’s heart rate to calculate the number of calories burned during a workout.
REQ-5	3	The system will display the number of calories burned and the number of calories consumed.
REQ-6	7	As an authorized user, I will be able to search for my meals through a search bar to input my calorie intake.
REQ-7	3	As an authorized user, I will be able to save frequent or favorite meals/snacks for a faster and easier experience.

Table 1-2: Calendar Requirements

ID	Priority Weight	Requirement
REQ-8	6	As an authorized user, I will be able to sync my Google Calendar with the application.
REQ-9	3	As an authorized user, I will be able to snooze workout reminders for a later time.
REQ-10	5	As an authorized user, I will be able to toggle workout reminders on or off.
REQ-11	3	As an authorized user, I will be able to ‘check-in’ with the app to inform that I am sticking to my scheduled workouts.
REQ-12	9	The application will suggest times the user should workout, based on user’s free time.

Table 1-3: Music Curator Requirements

ID	Priority Weight	Requirement
REQ-13	9	As an authorized user, I will be able to pick a type of workout I would like to do and the app should provide me with playlists with BPMs suited for that workout.
REQ-14	8	My playlist options should also incorporate the artists I prefer
REQ-15	9	I should be able to set a workout goal in terms of calories or steps and upon achievement of those goals, I should be able to activate a free membership reward.
REQ-16	3	I should be able to save the playlists I prefer and shuffle the same set next time.
REQ-17	4	I should be able to rely on the app to generate new playlists for me, as well
REQ-18	2	The music tempo should be correlated to my heartbeat while working out

Table 1-4: Safety & SOS Requirements

ID	Priority Weight	Requirement
REQ-19	8	As an authorized user, my watch will alert friends and family upon a press of a button
REQ-20	3	The watch should detect whether the user is running or user sets up exercise thus activating a safe run(feature with high sensitivity to any movements that alerts people)
REQ-21	5	I should be able to go out on a run with my watch knowing that especially if it's at night
REQ-22	7	The watch can detect weather and precipitation and alarm me if it's not safe or if there’s a state of emergency
REQ-23	6	The watch should be able to detect user’s health issues and warn their family and friends
REQ-24	5	The watch should not have a false alarm and create a sense of urgency.

Functional Requirements Specification

Stakeholders

Identify anyone and everyone who has an interest in this system (users, managers, sponsors, etc.). Stakeholders should be humans or human organizations.

- Users
- Nutritionists
- Music Services
- Google Calendar
- Emergency Responder

Actors and Goals

Identify the *roles* of people or devices that will directly interact with the system, their *types* (initiating vs. participating) and the *goals* of the initiating actors.

- Users
  - Role: Implement this technology into their lives
  - Type: Initiating
  - Goal: To maintain or achieve their fitness goals
- Google Calendar
  - Role: Provide access to previously created Google calendars
  - Type: Participating
- Nutritionists
  - Role: Gives input on what foods and exercises to recommend to users.
  - Type: Participating
- Music Services
  - Role: Provide access to music streaming services
  - Type: Participating
- Emergency Responder
  - Role: Upon request, offer assistance for users who feel unsafe
  - Type: Participating

Use Cases

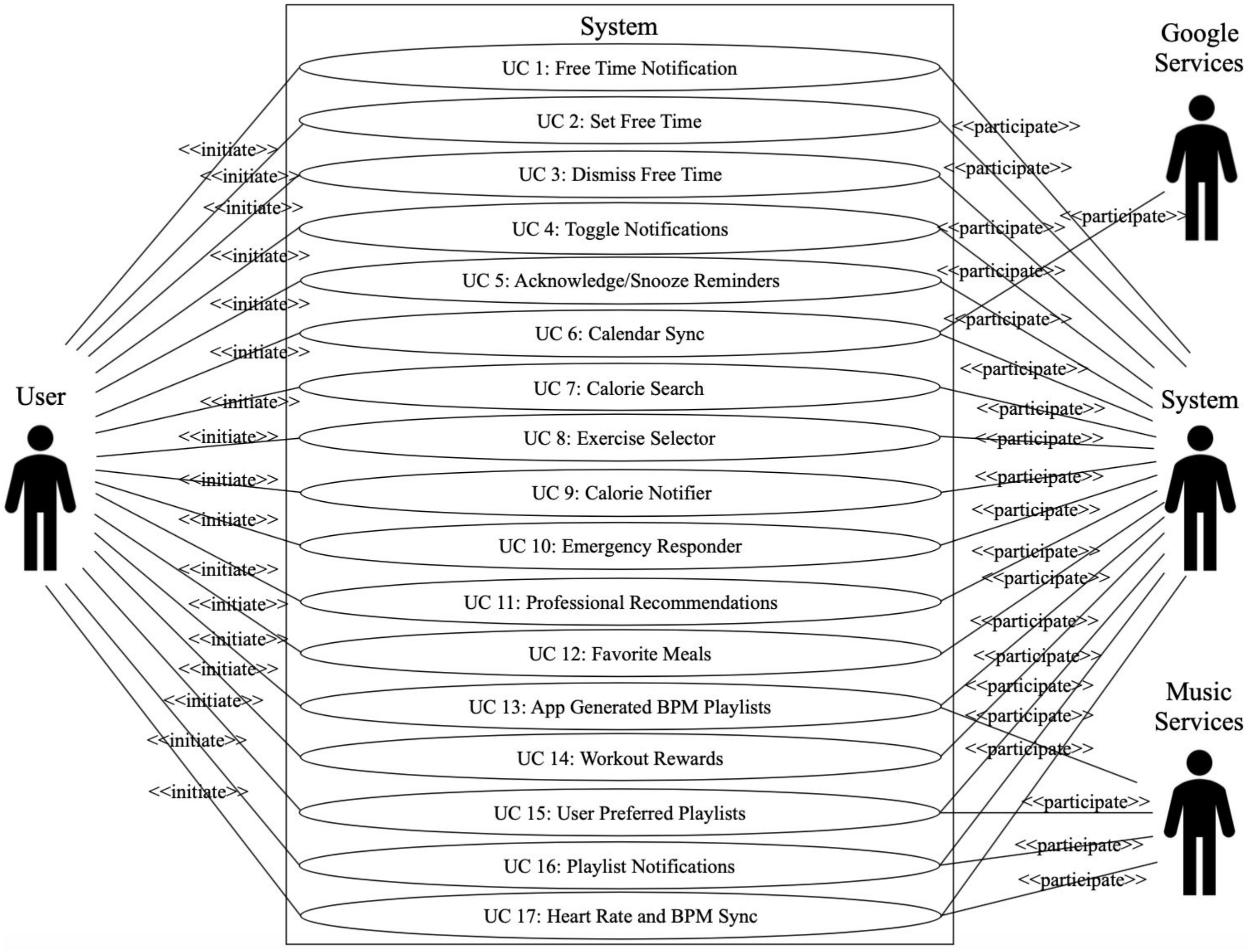
Actor	Actor's Goal	Functional Requirements	Use-Case Name
System	Remind users during the	Provides users with	Free Time

	pre-set period of the day if they have enough free time to exercise.	time throughout the day that they can implement a workout.	Notifications (UC#1)
User	Users will be able to set the minimal amount of free time needed to exercise as well as the time interval in which free time reminders are enabled.	Allows the user to customize the free time interval and amount of time required for a work out.	Set Free Time (UC#2)
System	If the user indicated that they exercised or if they dismissed the free time notification free time notifications will be disabled until the start period of the following day.	Turn off free time or work out notifications until the free time interval of the next day.	Dismiss Reminders (UC#3)
User	Users will be given the option to enable or disable free time notifications.	Toggle free time notifications ON/OFF.	Toggle Notifications (UC#4)
User	Users will be able to acknowledge, snooze, or accept the work out reminders.	Encourages the user to stick to their goal of exercising.	Acknowledge/Snooze Reminders (UC#5)
System	Remind the user to exercise, again, after a given period of time.	Waits 30 minutes before checking if the amount of free time is enough to implement a work out.	
User	Users will be able to sync their Google Calendars to the calendar on their device.	Allows users to seamlessly add their already created calendars to their device.	Calendar Sync (UC#6)
User	Users will be able to use a search bar to look up the calories in their food.	Lets users easily add the calories eaten into the system.	Calorie Search (UC #7)

User	Users will select what exercises they have completed for the day.	System will receive that information and know how many calories were burned based on pre-assigned values.	Exercise Selector (UC #8)
User	Users will be able to select a future workout based on the number of calories they will potentially burn	System will determine the number of calories burned based on the user's past exercise experiences	
System	Keeps track of the calories consumed and burned.	Notifies users of the calories lost or gained at the end of each day.	Calorie Notifier (UC #9)
User	Users will be able to notify a list of people if they are in danger.	The system will have a list of people made by the user, and the system will notify those people if the user presses a button	Emergency Responder (UC #10)
System	The system will automatically notify the close family list if the user is in a medical emergency.	The system will be able to identify user's health emergencies based on their heartbeat and notify the select list of people if need be.	
User	Users will be able to interact with recommendations suggested by nutritionists	The system will have a database of healthy recommendations suggested by professionals.	Professional Recommendations (UC #11)
User	Users will be able to save frequent/favorite meals for a quicker and easier meal-tracking experience.	App will display the list of favorite meals under the meal search bar.	Favorite Meals (UC #12)
User	Users will be able to select a certain type of workout and the app will generate	App will display the selected playlists and the user will have the	App generated BPM playlists (UC #13)

	BPM based playlists based on the intensity of the workout	option to save them for future workouts	
System and User	On the 1st of every month, the user will be prompted to enter a workout goal, either in terms of calories or steps.	The app will then reward the user, in terms of subscriptions of one month	Workout rewards (UC #14)
User	Users will be able to integrate spotify/apple music from their phone	The app will also generate playlists based on the artists the user likes/prefers and will allow the user to save them	User preferred playlists (UC #15)
System	The User will receive daily revised playlists from the app, to keep them motivated as listening to same music overtime gets monotonous	The app will generate new playlist notifications for the user, and perhaps in conjunction with the workout reminders to increase the motivation factor.	Playlist notifications (UC #16)
System	The user will experience a change in music tempo in conjunction with their heart rate.	The app will change songs/ music tempo by detecting the user's heart rate, high BPM songs will play	Heart rate and BPM sync (UC #17)

Use Case Diagram



System
Google Services
User
System
Music Services
UC 1: Free Time Notification
UC 2: Set Free Time
UC 3: Dismiss Free Time
UC 4: Toggle Notifications
UC 5: Acknowledge/Snooze Reminders
UC 6: Calendar Sync
UC 7: Calorie Search
UC 8: Exercise Selector
UC 9: Calorie Notifier
UC 10: Emergency Responder
UC 11: Professional Recommendations
UC 12: Favorite Meals
UC 13: App Generated BPM Playlists
UC 14: Workout Rewards
UC 15: User Preferred Playlists
UC 16: Playlist Notifications
UC 17: Heart Rate and BPM Sync
<<initiate>>
<<participate>>

Traceability Matrix

Req.	PW	UC 1	UC 2	UC 3	UC 4	UC 5	UC 6	UC 7	UC 8	UC 9	UC 10	UC 11	UC 12	UC 13	UC 14	UC 15	UC 16	UC 17
1	3							x						x				
2	2								x						x			
3	5							x	x			x						
4	7								x	x							x	
5	3							x		x								
6	7							x										
7	3												x					
8	6						x									x		
9	3					x												
10	5			x	x													x
11	3			x		x												
12	9	x	x															
13	9													x		x	x	
14	8													x			x	
15	8														x			
16	3															x	x	
17	4													x		x		
18	2																x	x
19	8										x							
20	3										x							
21	5					x	x				x							
22	7										x							
23	6										x							
24	5																	
Max PW		9	9	5	5	5	6	7	7	7	8	5	3	9	8	9	8	5
Total PW		9	9	8	5	11	11	13	13	10	15	5	3	15	10	15	15	7

Fully-Dressed Description

User Case UC- 5 : Acknowledge/Snooze Reminders	
Related Requirements:	REQ9 and REQ11 as stated in Table 1-2
Initiating Actor:	User
Actor's Goal:	To indicate that they have/are exercising or to snooze the notification for a later reminder
Participating Actors:	System
Preconditions:	● User has entered minimal amount of free time needed to exercise ● User has inputted the time interval in which free time reminders are enabled ● Current time is within users previously set notification interval ● Enough free time to exercise
Postconditions:	● Reminders will be stopped until the next free time interval of the following day (Acknowledge) ● Another reminder will occur after 30 minutes if enough time to exercise is still available (Snooze)
Flow of Events for Main Success Scenario:	
→	1. User has inputted the time interval in which free time reminders are enabled
→	2. User has entered the minimum amount of time needed to exercise
←	3. System scans calendar to see if there is enough free time to exercise
→	4. User snoozes/acknowledges the reminder
←	5. System disables free time notifications until the next day/looks for free time again in 30 mins

User Case UC - 7: Calorie Search	
Related Requirements:	REQ1, REQ3, REQ5, and REQ6 as stated in Table 1-1
Initiating Actor:	User
Actor’s Goal:	To use a search bar to look up the number of calories in food.
Participating Actors:	System
Preconditions:	● User hasn’t entered daily calorie intake ● User will like to know the number of calories in certain meals
Postconditions:	● The number of calories will be added to the user’s calorie tracker. ● User will be redirected back to the search bar page.
Flow of Events for Main Success Scenario:	
→	1. User enters a food item name
←	2. System (a) provide calorie count for average portion size (b) prompts user to add calories to calorie tracker
→	3. User (a) edits the portion size and/or (b) confirms calorie count
←	4. System updates the user’s daily calorie count

User Case UC- 13: App-Generated BPM Playlists

Related Requirements:	REQ-13, REQ-14, REQ-16, and REQ-18, as stated in Table 1-3
Initiating Actor:	User
Actor's Goal:	To receive personalized workout playlists based on inputted intended exercises.
Participating Actors:	System
Preconditions:	● User enters intended workout and exercises, along with duration, into the playlist section of the app. ● User enters preferred genres for listening. ● User gives the app permission to use personal music subscription account to create a playlist.
Postconditions:	● The system creates a playlist based on intrinsic data that matches songs to intended workout, workout time, and genre preference and automatically saves workout and playlist. ● The system notifies the user when the playlist is ready to use. ● The system connects to a music app to play curated songs from a playlist in correct order when prompted.

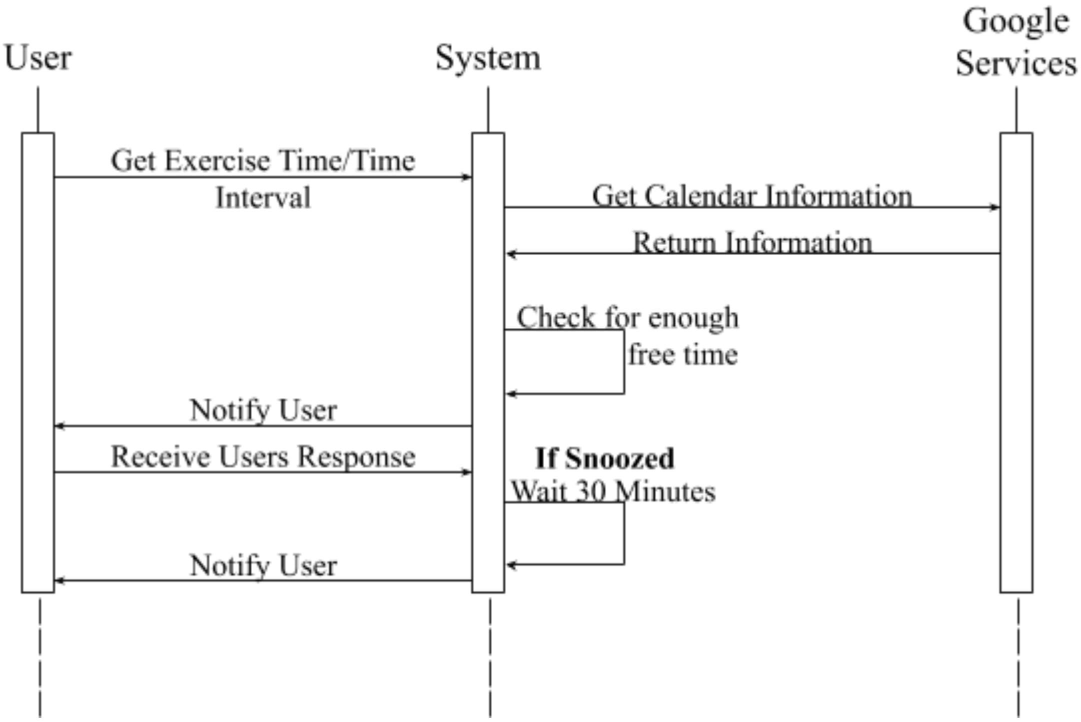
Flow of Events for Main Success Scenario:

→ 1. User enters intended exercises and preferred genres into the system.
→ 2. User gives system access to their 3rd party music account.
← 3. System creates playlist using data inputted by the user and saves the playlist and workout data for future playlists.
← 4. System notifies user of playlist creation.
→ 5. User prompts the system to play a playlist when desired.
← 6. System plays a playlist using the music app when prompted.

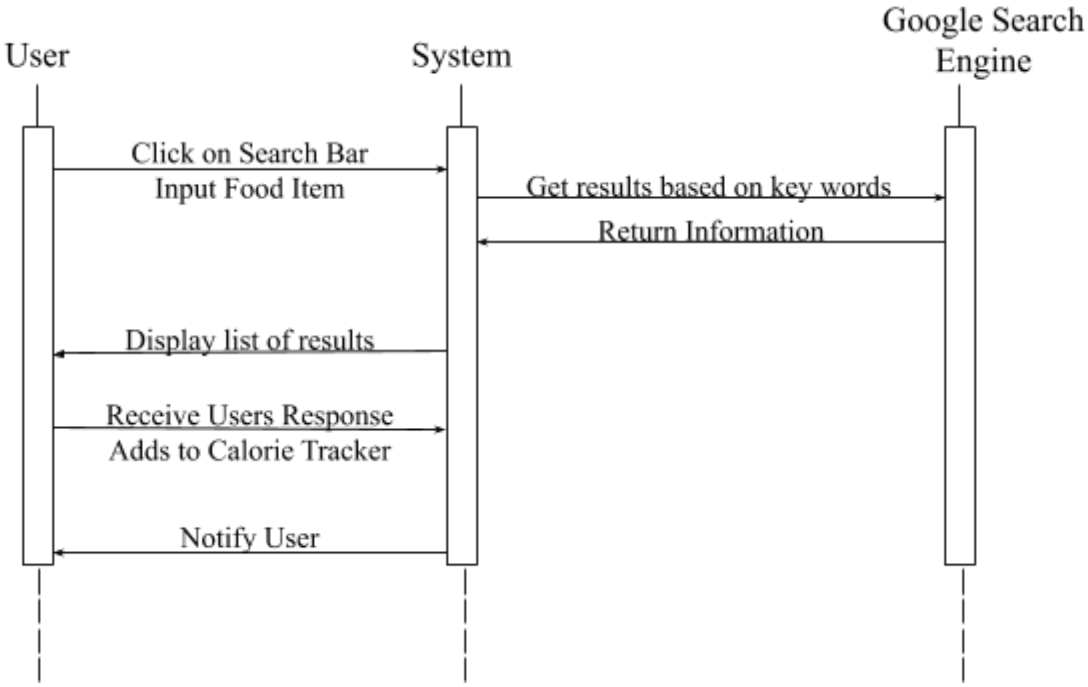
User Case UC- 10 : Emergency Responder	
Related Requirements:	REQ19, REQ20, REQ22 and REQ23 as stated in Table 1-2
Initiating Actor:	User
Actor’s Goal:	Users will be able to notify a list of people if they are in danger.
Participating Actors:	System
Preconditions:	● User must have a premade emergency contact list that they have set up ● User needs the device with them ● User would need to be in running state or user manually triggers the command via a button ● Location must be turned on, and data must be stored ● There must be a signal for the phone to transmit information
Postconditions:	● Users device information is sent to the people on the list ● Users heartbeat is then monitored and giving a refreshed update
Flow of Events for Main Success Scenario:	
→	1. User feels unsafe and triggers the button
←	2. System collects location information, heartbeat, and all other necessary and vital data
←	3. System then packages this data and send them to the provided list of contacts that user had setup
←	4. System then sends data, and has an auto update depending on the preference of the user.

System Sequence Diagrams

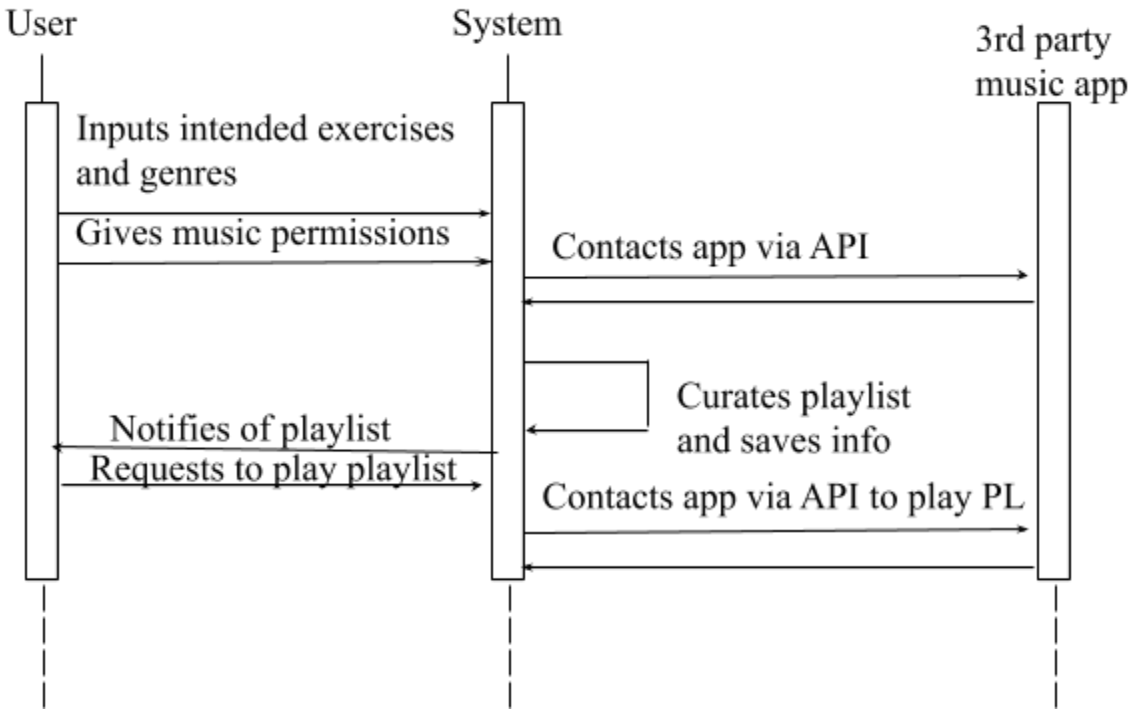
System Sequence Diagram for Use Case #5



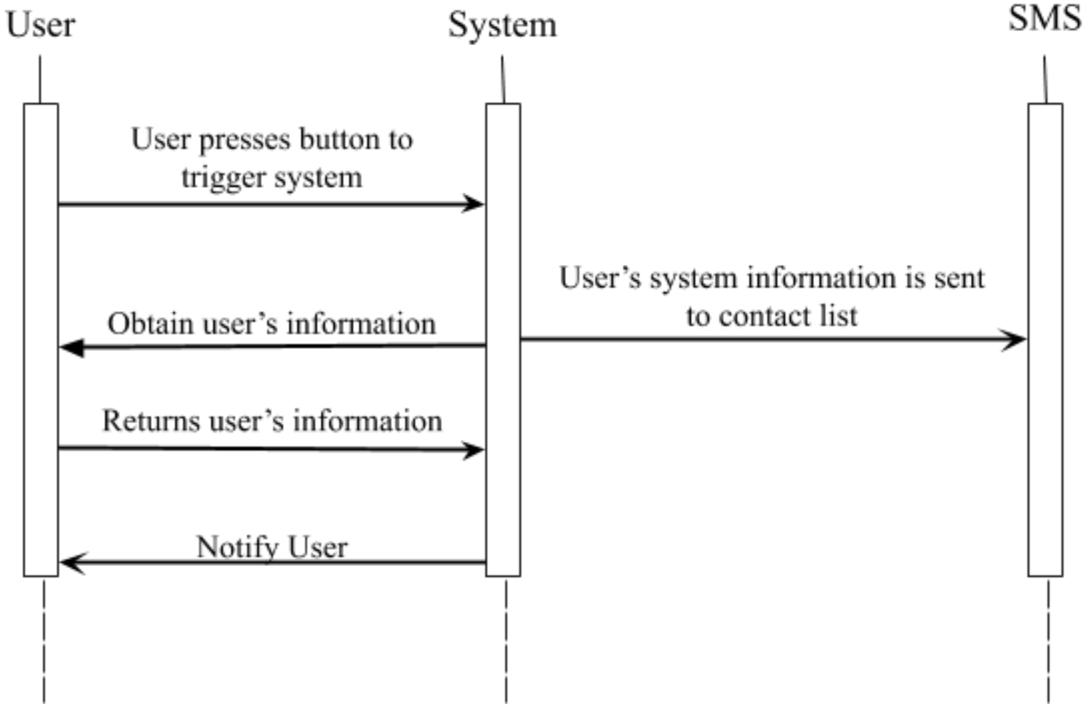
System Sequence Diagram for Use Case #7



System Sequence Diagram for Use Case #13

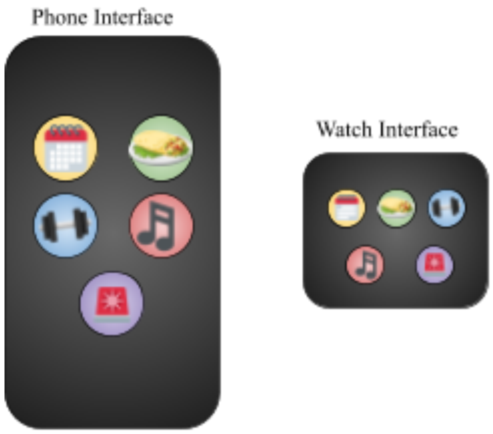

User
System
3rd party music app
Inputs intended exercises and genres
Gives music permissions
Contacts app via API
Curates playlist and saves info
Notifies of playlist
Requests to play playlist
Contacts app via API to play PL

System Sequence Diagram for Use Case #10


User
System
SMS
User presses button to trigger system
User's system information is sent to contact list
Obtain user's information
Returns user's information
Notify User

User Interface Specification

Preliminary Design & User Effort Estimation



The home screen contains 5 modules for the users to interact with:

1.) Calendar
2.) Calorie Tracker
3.) Exercise Selector
4.) Music Service
5.) Emergency

Figure 1.1 - Homescreen

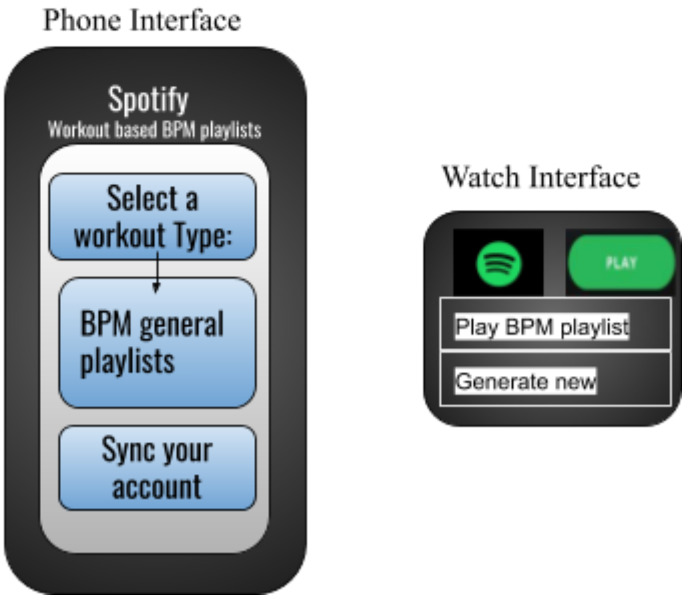
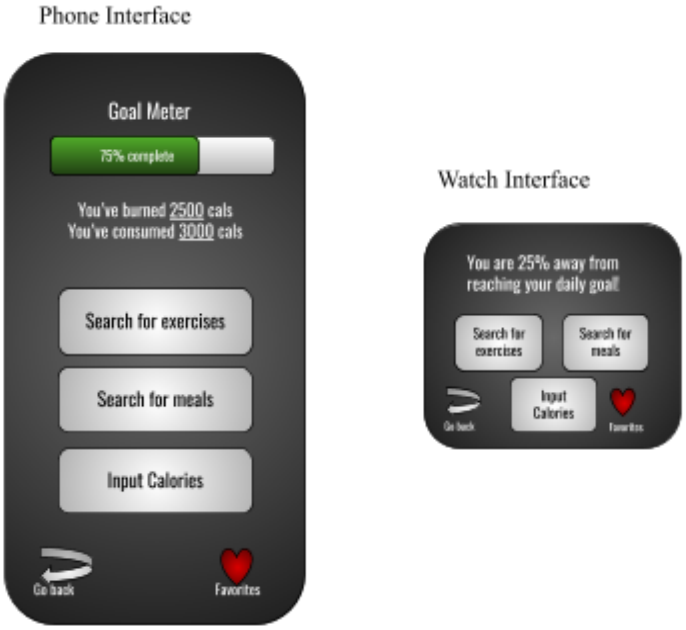


Figure 1.2 - Music Curator

<u>User Effort Estimation</u>:
For the interface of the calorie tracker, the user will see a goal meter, three options to enter their calorie count, a favorite button and a back button. The goal meter will be calculated with the user's information about their calorie intake/outtake ratio. The 'Search for exercises' button will allow the user to browse through the recommended exercises, as well as select their current workouts. The 'Search for meals' button will allow the user to search for meals and receive the calorie count for them. The 'Input Calories' button will allow the user to manually input their calories. The 'favorites' button will allow the user to access the user's favorite exercises and meals. The 'Go back' button will bring the user back to the homescreen.



You've burned 2500 cals
You've consumed 3000 cals

You are 25% away from reaching your daily goal!

Figure 1.3 - Calorie Tracker

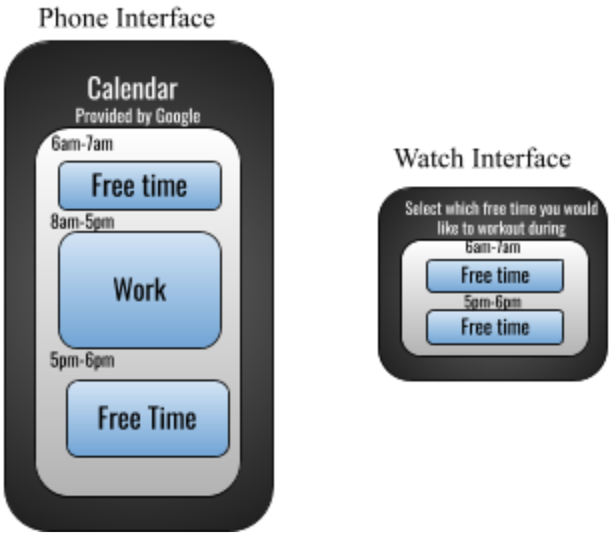


Figure 1.4 - Calendar Interface

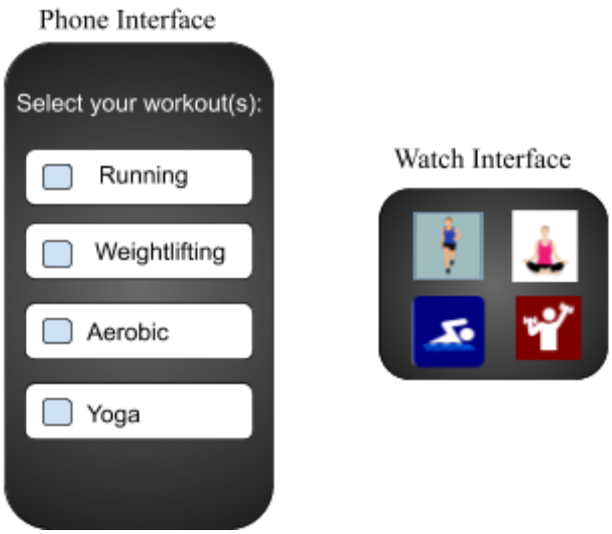


Figure 1.5 - Exercise Selector

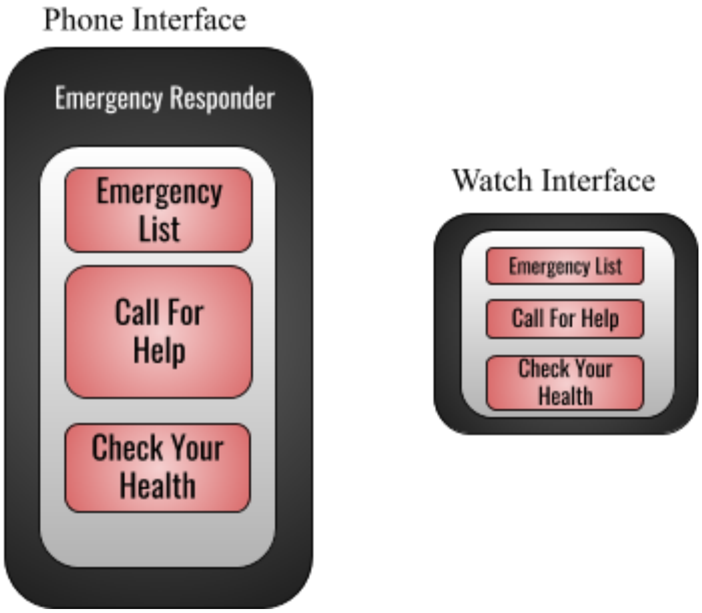


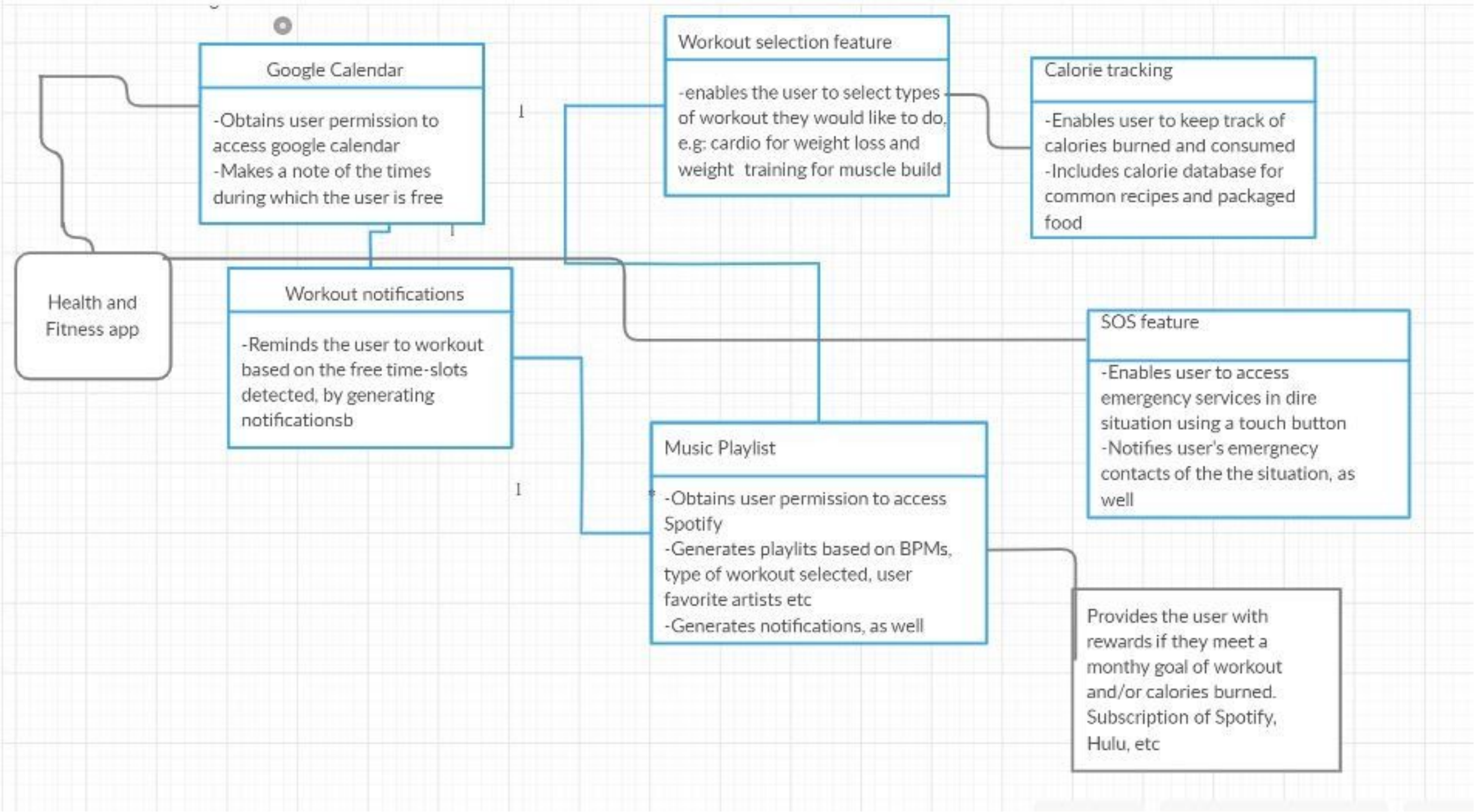
Figure 1.6 - Emergency Responder

User Effort Estimation:

The interface for our application is simple with a few buttons and clear titles per page. The user will be able to access any page by clicking the icon on the home page and can return by clicking on a back button. Each page follows a similar setup to simplify the experience for the user. Overall, the user will not require much effort to use our application.

Domain Analysis

Domain Model



-Obtains user permission to access google calendar
-Makes a note of the times during which the user is free

-Reminds the user to workout based on the free time-slots detected, by generating notificationsb

-enables the user to select types of workout they would like to do, e.g: cardio for weight loss and weight training for muscle build

-Enables user to keep track of calories burned and consumed
-Includes calorie database for common recipes and packaged food

-Enables user to access emergency services in dire situation using a touch button
-Notifies user's emergnecy contacts of the the situation, as well

-Obtains user permission to access Spotify
-Generates playlits based on BPMs, type of workout selected, user favorite artists etc
-Generates notifications, as well

Provides the user with rewards if they meet a monthy goal of workout and/or calories burned. Subscription of Spotify, Hulu, etc

Concept Definitions

Responsibility	Concept
R1: Set the users free time interval and workout time	Calendar
R2: Calculate and keep track of users free time	Calendar
R3: Send notifications to work out during enough free time	Calendar
R4: Snooze/Dismiss/Acknowledge free time notifications	Calendar
R5: Enable/Disable free time reminders	Calendar
R6: Gather data from Google Calendar	Calendar
R7: Calculate user’s calorie intake/outtake ratio	Calorie Tracker
R8: Display the amount of calories consumed/burned	Calorie Tracker
R9: Recommend exercises/ meals to meet goal	Calorie Tracker
R10: Compile list of most frequent exercises/meals	Calorie Tracker
R11: Display list of common exercises for selection and additional options for each exercise post-selection	Workout Selector
R12: Compiles and saves workouts under user-chosen name	Workout Selector
R13: Prompts user for 3rd-party music subscription login and permissions	Music Playlist
R14: Accesses user-intended workout when user prompts creation of playlist	Music Playlist
R15: Matches songs to each exercise intensity/BPM and length and compiles into playlist	Music Playlist
R16: Saves playlist under unique name for later use	Music Playlist
R17: Allows the user to create a list of close contacts for emergency	SOS Feature
R18: Allows the user to notify the close contacts list in case of emergency	SOS Feature
R19: Checks the status of the user’s health	SOS Feature

Association Definitions

Concept Pair	Association Description	Association Name
Calendar ↔ SyncCalendar	Calendar calls SyncCalendar to sync the users existing Google Calendar with the calendar on the device.	Get Request Send Response
Calendar ↔ SetFreeTime	Calendar calls SetFreeTime when the user is initially setting the free time interval and workout time needed and when the user is updating those set times.	Set Request
Calendar ↔ CalcFreeTime	Calendar calls CalcFreeTime to check if there is enough free time, within the free time interval, to exercise.	Get Request Send Response
Calendar ↔ ToggleFreeTime	Calendar calls ToggleFreeTime to enable and disable free time notifications.	Set Request
Calendar ↔ FreeTimeResp	Calendar calls FreeTimeResp to store the users response to a free time notification.	Get Request Send Response
Calendar ↔ NotifyFreeTime	Calendar sends a notification to the user when there is enough free time, within the free time interval, to exercise.	Send Response
Calorie Tracker ↔ CalRatio	Calorie Tracker invokes CalRatio to determine the ratio of calorie intake to outtake	Send Result
Calorie Tracker ↔ FoodRec	FoodRec will use the Calorie Tracker, along with user preferences, to recommend meals that will help the user meet their goals	Get Request Send Response
Calorie Tracker ↔ ExerciseRec	ExerciseRec will use the Calorie Tracker to recommend exercises that will help the user burn enough calories to meet their goals	Get Request Send Response
Calorie Tracker ↔ FreqList	FreqList will compile a list of the user's most frequent exercises and meals	Get Request Send Response

CalDisp ↔ CalRatio	CalDisp will use the information from CalRatio to display the User’s calorie consumption and calories burned	Send Result
Workout Selector ↔ExerSelect	ExerSelect will display a list of exercises for user selection, and expand to provide options for time or amount when exercise is selected.	Get Request Send Response
Workout Selector ↔ExerSave	ExerSave will take user-inputted name and save the workout under that name in the user’s account.	Get Request Send Response
Music Playlist ↔MusicSync	MusicSync will prompt the user to input their 3rd party music subscription info and give app permissions.	Get Request Set Request Send Result
Music Playlist ↔WorkChoose ↔ExerSave	WorkChoose will display a list of saved workouts and common genres for user to choose for playlist curation, and save the chosen workout and genre in temporary secondary memory.	Get Request Set Request
Music Playlist ↔MusicCurate	MusicCurate will match songs based on the selected workout chosen and the user’s genre preferences.	Get Request Send Result
Music Playlist ↔MusicSend	MusicStore will store the playlist and notify the user of playlist creation.	Send Result
Emergency ↔CreateList	Emergency service will help user create a list of emergency contacts	Get Request Set Request Send Result
Emergency ↔Report	Receive a signal from the user and notify the emergency contacts.	Get Request Send Result
Emergency↔CheckStatus	A report the user’s health status will be analysed and potentially sent to their emergency contacts	Send Response
Emergency↔LocationStatus	User’s location will be pulled from the phone's location/server and then saved and added to the alert message being sent.	Get Request Send Result

Attribute Definitions

Responsibility	Attribute	Concept
R1: Set the users free time interval and workout time	SetFreeTime	Calendar
R2: Calculate and keep track of users free time	CalcFreeTime	Calendar
R3: Send notifications to work out during enough free time	NotifyFreeTime	Calendar
R4: Snooze/Dismiss/Acknowledge free time notifications	FreeTimeResp	Calendar
R5: Enable/Disable free time reminders	ToggleFreeTime	Calendar
R6: Gather data from Google Calendar	SyncCalendar	Calendar
R7: Calculate user's calorie intake/outtake ratio	CalRatio	Calorie Tracker
R9: Recommend exercises/ meals to meet goal	CalDisp	Calorie Tracker
R9: Recommend exercises/ meals to meet goal	FoodRec & ExerciseRec	Calorie Tracker
R10: Compile most frequent exercise/meal list	FreqList	Calorie Tracker
R11: Display list of common exercises for selection and additional options for each exercise post-selection	ExerSelect	Workout Selector
R12: Compiles and saves workouts under user-chosen name	ExerSave	Workout Selector
R13: Prompts user for 3rd-party music subscription login and permissions	MusicSync	Music Playlist
R14: Accesses user-intended workout when user prompts creation of playlist	WorkChoose	Music Playlist

R15: Matches songs to each exercise intensity/BPM and length and compiles into playlist	MusicCurate	Music Playlist
R16: Saves playlist under unique name for later use	MusicSend	Music Playlist
R17: Allows the user to create a list of close contacts for emergency	CreateList	SOS
R18: Allows the user to notify the close contacts list in case of emergency	Report	SOS
R19: Checks the status of the user’s health	CheckStatus	SOS

Traceability Matrix

Domain Model	UC 1	UC 2	UC 3	UC 4	UC 5	UC 6	UC 7	UC 8	UC 9	UC 10	UC 11	UC 12	UC 13	UC 14	UC 15	UC 16	UC 17
Calendar	x	x	x	x	x	x											
Calorie Tracker							x		x		x	x					
Workout Notifications					x												
Workout Selector								x									
Music Playlist								x					x	x	x	x	x
SOS Feature										x							

System Operation Contracts

Contract CO1:

Operation:	Exercise Reminder/Indicator
Cross Reference:	User Case UC- 5 : Acknowledge/Snooze Reminders
PreConditions:	User has inputted the time intervals in which they have free time.
PostConditions:	Reminders are stopped until the next free time interval (Acknowledged) or another reminder is sent after 30 minutes if there is enough time available for exercise (Snooze).

Contract CO2:

Operation:	Search Calories Consumed
Cross Reference:	User Case UC - 7: Calorie Search
PreConditions:	User finished a meal and wants to know how many calories to input.
PostConditions:	Calorie tracker records the number of calories taken in.

Contract CO3:

Operation:	Create Personalized Music Playlist
Cross Reference:	User Case UC- 13: App-Generated BPM Playlists
PreConditions:	User requests personal playlist creation and selects a saved workout and genre.
PostConditions:	Music Playlist feature curates playlist, saves it, and notifies the user.

Contract CO4:

Operation:	Danger Notifier
Cross Reference:	User Case UC- 10 : Emergency Responder

PreConditions:	User has premade emergency contact list set up and location is turned on.
PostConditions:	User’s device information sent to people on the list.

Project Size Estimation (Use Case Points)

Unadjusted Actor Weight

Actor	**Relevant Information**	**Complexity**	**Weight**
User	User is interacting with the system	Complex	3
Database	Database stores data in the system that is either manually entered or fetched internally.	Average	2
System	System displays the different features the Users can choose from, display results, and send reminders.	Average	2

UAW = (# of Simple Actors x 1) + (# of Average Actors x 2) + (# of Complex Actors x 3) = 2*2 + 1*3 = **7**

Unadjusted Use Case Weight

Use-Case Name	**Relevant Information**	**Category**	**Weight**
Free Time Notifications (UC#1)	System has to provide users with a notification pop-up. It only uses one actor.	Simple	5
Set Free Time (UC#2)	User will be able to set their minimal amount of free time needed. This requires two actors.	Complex	15
Dismiss Reminders (UC#3)		Average	10
Toggle Notifications (UC#4)	Toggle free time notifications	Average	10

	ON/OFF.		
Acknowledge/Snooze Reminders (UC#5)	Encourages the user to stick to their goal of exercising.	Simple	5
	Waits 30 minutes before checking if the amount of free time is enough to implement a work out.		
Calendar Sync (UC#6)	Allows users to seamlessly add their already created calendars to their device.	Average	10
Calorie Search (UC #7)	Lets users easily add the calories eaten into the system.	Complex	15
Exercise Selector (UC #8)	System will receive that information and know how many calories were burned based on pre-assigned values.	Complex	15
	System will determine the number of calories burned based on the user's past exercise experiences		
Calorie Notifier (UC #9)	Notifies users of the calories lost or gained at the end of each day.	Average	10
Emergency Responder (UC #10)	The system will have a list of people made by the user, and the system will notify those people if the	Average	10

	user presses a button		
	The system will be able to identify user’s health emergencies based on their heartbeat and notify the select list of people if need be.		
Professional Recommendations (UC #11)	The system will have a database of healthy recommendations suggested by professionals.	Complex	15
Favorite Meals (UC #12)	App will display the list of favorite meals under the meal search bar.	Simple	5
App generated BPM playlists (UC #13)	App will display the selected playlists and the user will have the option to save them for future workouts	Complex	15
Workout rewards (UC #14)	The app will then reward the user, in terms of subscriptions of one month	Average	10
User preferred playlists (UC #15)	The app will also generate playlists based on the artists the user likes/prefers and will allow the user to save them	Complex	15
Playlist notifications (UC #16)	The app will generate new playlist notifications for the user, and perhaps in conjunction with the workout reminders to increase the	Simple	5

	motivation factor.		
Heart rate and BPM sync (UC #17)	The app will change songs/ music tempo by detecting the user’s heart rate, high BPM songs will play	Complex	15

UUCW = (Total # of Simple Use Cases x 5) + (Total # of Average Use Cases x 10) + (Total # of Complex Use Cases x 15) = 4*5 + 6*10 + 7*15 = **185**

UUCP = UUCW + UAW = 185 + 7 = **192**

Technical Complexity Factors (TCF):

Technical Factor	**Description**	**Weight**	**Perceived Complexity**	**Calculated Factor**
T1	Distributed System (app version and watch version)	2.0	3	6
T2	Users expect a good response time	1.0	3	3
T3	End-user expects efficiency but nothing exceptional	1.0	4	4
T4	Internal processing is complex	1.0	4	4
T5	Code Reusability	1.0	1	1
T6	Ease of install is important	0.5	3	1.5
T7	Ease of use is very important	0.5	4	2
T8	Portability to other platforms	2.0	2	4
T9	Somewhat difficult to modify or add new features	1.0	3	3
T10	Concurrent use (by multiple users)	1.0	3	3

T11	Special security features	1.0	2	2
T12	No direct access for third parties	1.0	0	0
T13	No special user training facilities required	1.0	0	0

Technical Factor Total: 33.5
TCF = C1 + C2*Technical Factor Total = 0.6+ (0.01)*33.5 = **0.935**

Environmental Complexity Factors (ECF):

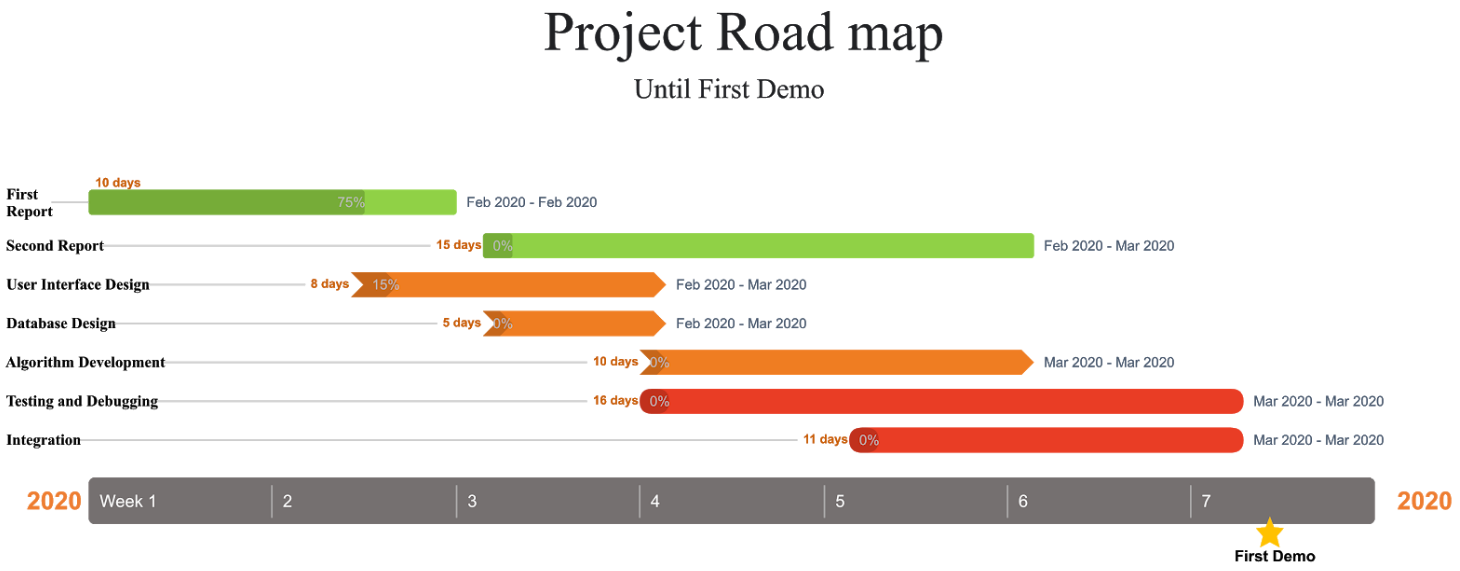
Environmental Factor	**Description**	**Weight**	**Perceived Impact**	**Calculated Factor**
E1	Some familiarity with the UML-based development	1.5	1	1.5
E2	Beginner familiarity with application problem	0.5	2	1
E3	Some knowledge of object-oriented approach	1	3	3
E4	Beginner lead analyst capability	0.5	2	1
E5	Highly motivated	1	4	4
E6	Stable requirements expected	2	3	6
E7	No part-time staff involved	-1	0	0
E8	Using Java, some familiar with it and others willing to learn	-1	3	-3

Environmental Factor Total: 13.5
ECF = C1 + C2*Environmental Factor Total = 1.4 + (-0.03)*13.5 = **0.995**

Overall Project Size Estimation:

UCP = UUCP x TCF X ECF = 192* 0.935*0.995 = **178.62**

Project Management



References

1. “FitBit Health Monitoring Analytics” https://www.ece.rutgers.edu/~marsic/books/SE/projects/HealthMonitor/2014-g5-report3.pdf
2. "Software Engineering book", Ivan Marsic http://www.ece.rutgers.edu/~marsic/books/SE/book-SE_marsic.pdf
3. “Learn about the four different types of exercise” https://www.captel.com/2013/10/learn-four-different-types-exercise/
4. “Creating a watchOS App” https://developer.apple.com/tutorials/swiftui/creating-a-watchos-app
5. “3 Music Apps that will create a BPM-based playlist for your workout” https://www.mensjournal.com/health-fitness/3-music-apps-will-create-bpm-based-playlist-your-workout/
6. “Use Case Points” https://en.wikipedia.org/wiki/Use_Case_Points

Fitness Friend

Group 6

Prepared By: Amber Haynes, Tiyon King, Jenna Krause, Mya Odrick,
Devvrat Patel, Andrew Rezk, Maria Rios, Shivani Sunil, Hedaya Walter

March 1, 2020 Software Engineering

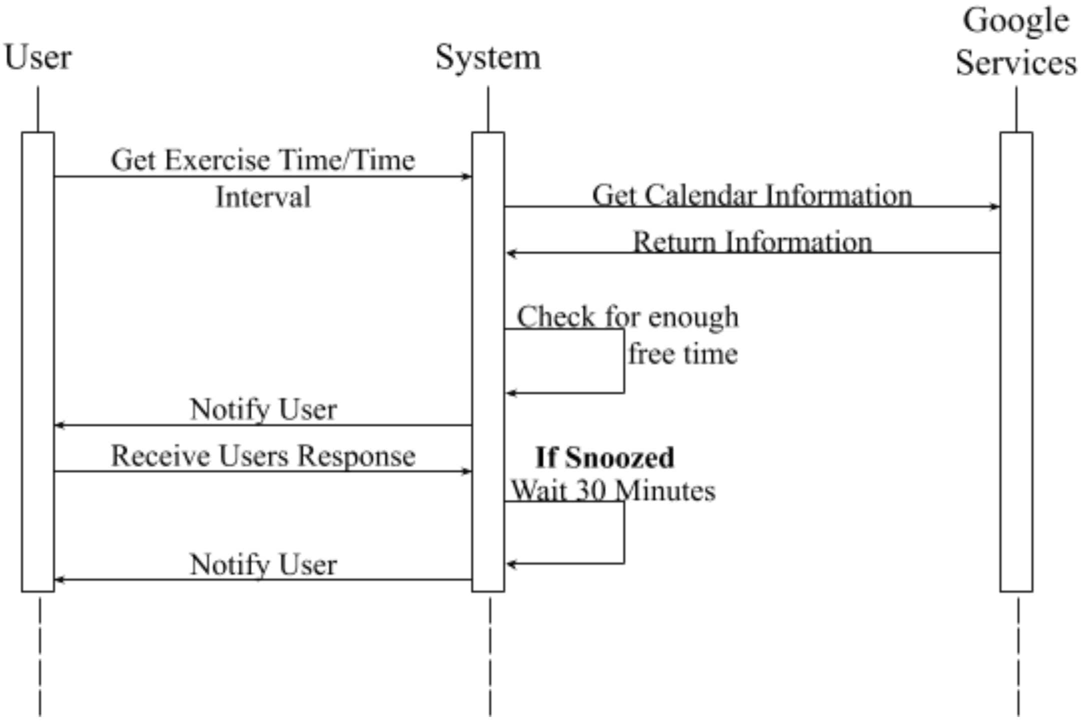
Table of Contents

Individual Breakdown

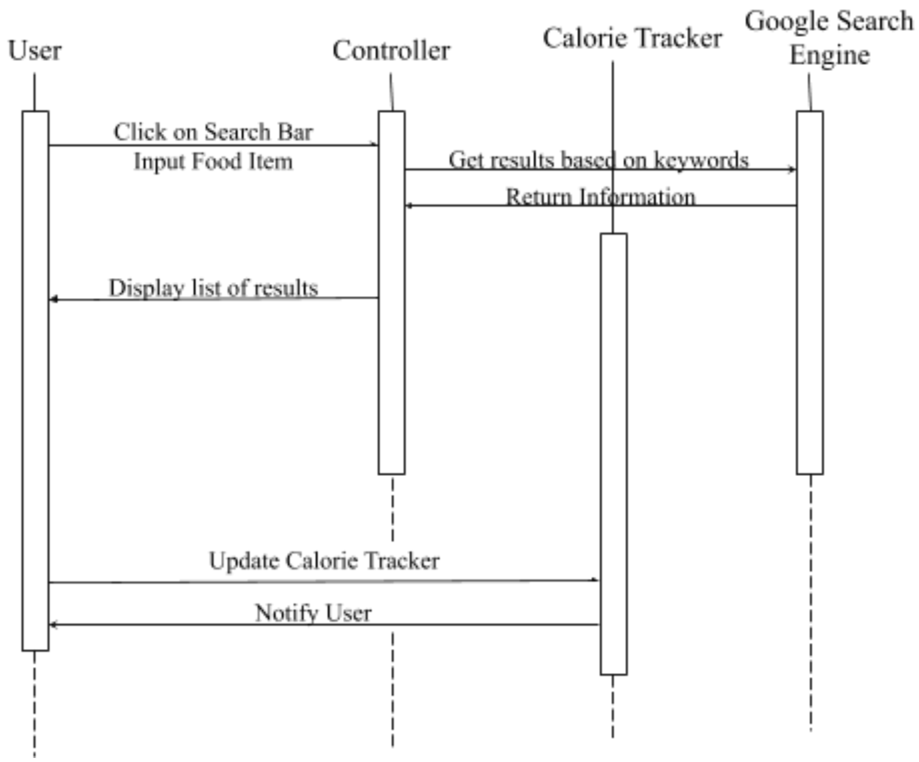
		Amber Haynes	Tiyon King	Jenna Krause	Mya Odrick	Devvrat Patel	Andrew Rezk	Maria Rios	Shivani Sunil	Hedaya Walter
Interaction Diagrams				33%	33%			33%		
Classes & Specs	Class Diag & Descr.			100%						
	Signatures			33%	33%				33%	
Sys Arch & Design	Architectural Styles	50%	50%							
	Identifying Subsystems	50%			50%					
	Mapping Hardware		100%							
	Persistent Data Storage		50%		50%					
	Network Protocol		100%							
	Global Control Flow		100%							
	Hardware Requirements		33%				33%		33%	
Algo. & Data Struct.		20%	20%	20%	20%				20%	
User Interface Design & Implementation		20%		20%		20%		20%	20%	
Design of Tests		14.2%	14.2%	14.2%		14.2%	14.2%	14.2%	14.2%	
Project Management &	Meeting Attendance	16.6%	16.6%	16.6%	16.6%		16.6%	16.6%		
	Merging the Contributions from		50%	50%						

Plan of Work	Individual Team Members									
	Project Coordination & Progress Report	16.6%	16.6%	16.6%			16.6%	16.6%	16.6%	
	Plan of Work		25%	25%		25%			25%	
	Breakdown of Responsibilities		16.6%	16.6%	16.6%	16.6%	16.6%		16.6%	

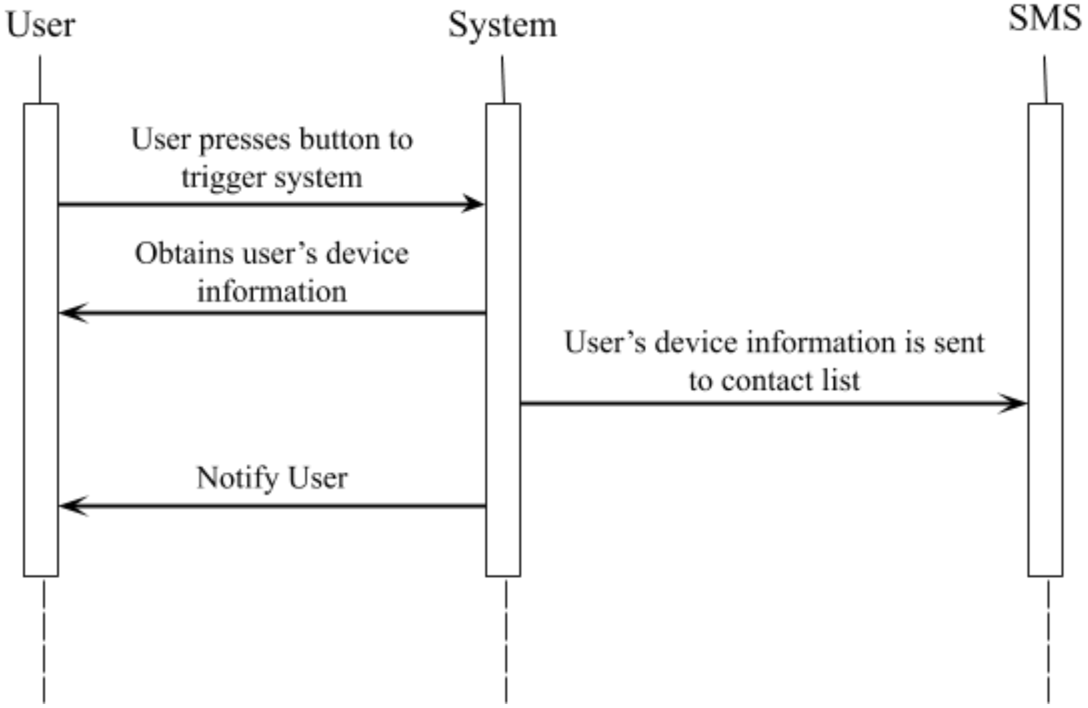
Interaction Diagrams



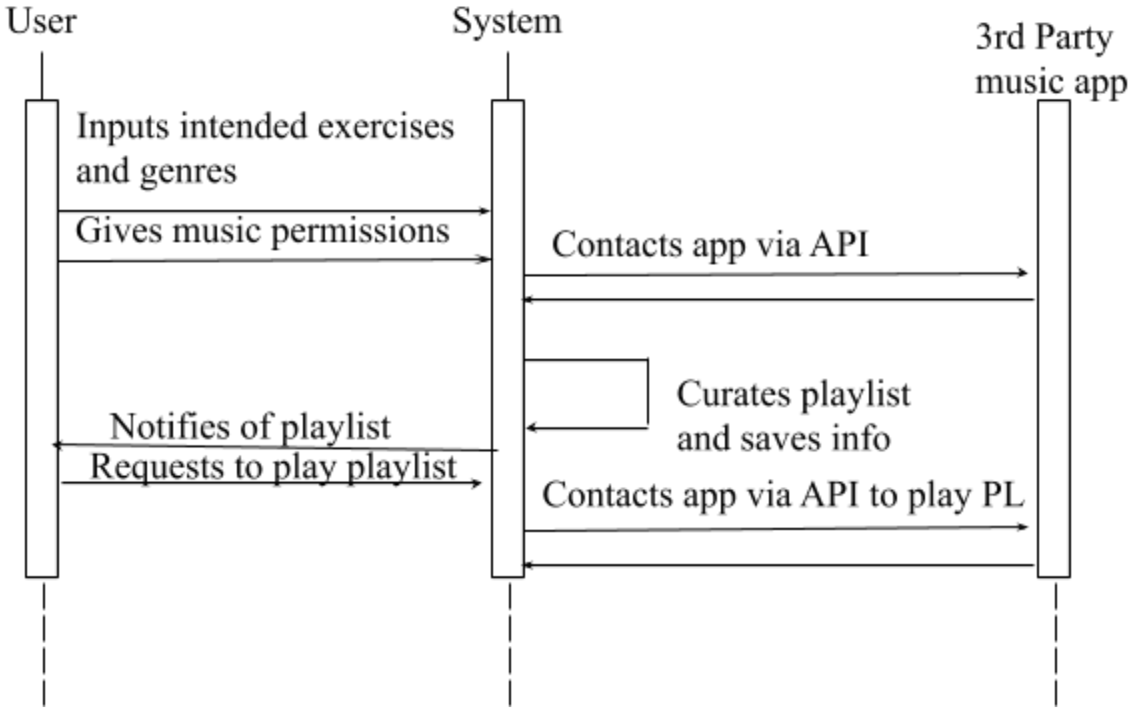
Description of Use Case #5:
This sequence is used to notify the “User” of free time that is available to be used for exercise. Once the “User” inputs the required amount of free time necessary to work out and the interval of time in which free time notifications will be enabled, the “System” will send a request to “Google Services” for the user’s calendar information. Then, the “System” will check for free time which meets the two inputted requirements. After that, if there is enough free time within the free time notification interval, the device will send a notification to the “User” informing them of this availability. The “System” will then wait for a response from the “User” and then act accordingly. In this case, the Interface Segregation Principle is used, being able to understand each class in isolation.



Description of Use Case #7:
This use case shows the sequence of the search function within the calorie tracker. In this case, we use the Single Responsibility Principle (SRP). The search engine is responsible for relaying information about the requested food, the calorie tracker is used to keep an updated calorie counter, and the controller serves as the main unit for the user's interaction. We also use the Interface Segregation Principle (ISP) where each class can be understood in isolation. The calorie tracker requests the number of calories consumed and calculates the caloric ratio with the number of calories burned. The search engine class on its own performs the task of generating calorie information upon receiving serving size information and food names as keywords. These independent functions are combined to create success for this use case.


User
System
SMS
User presses button to trigger system
Obtains user's device information
User's device information is sent to contact list
Notify User

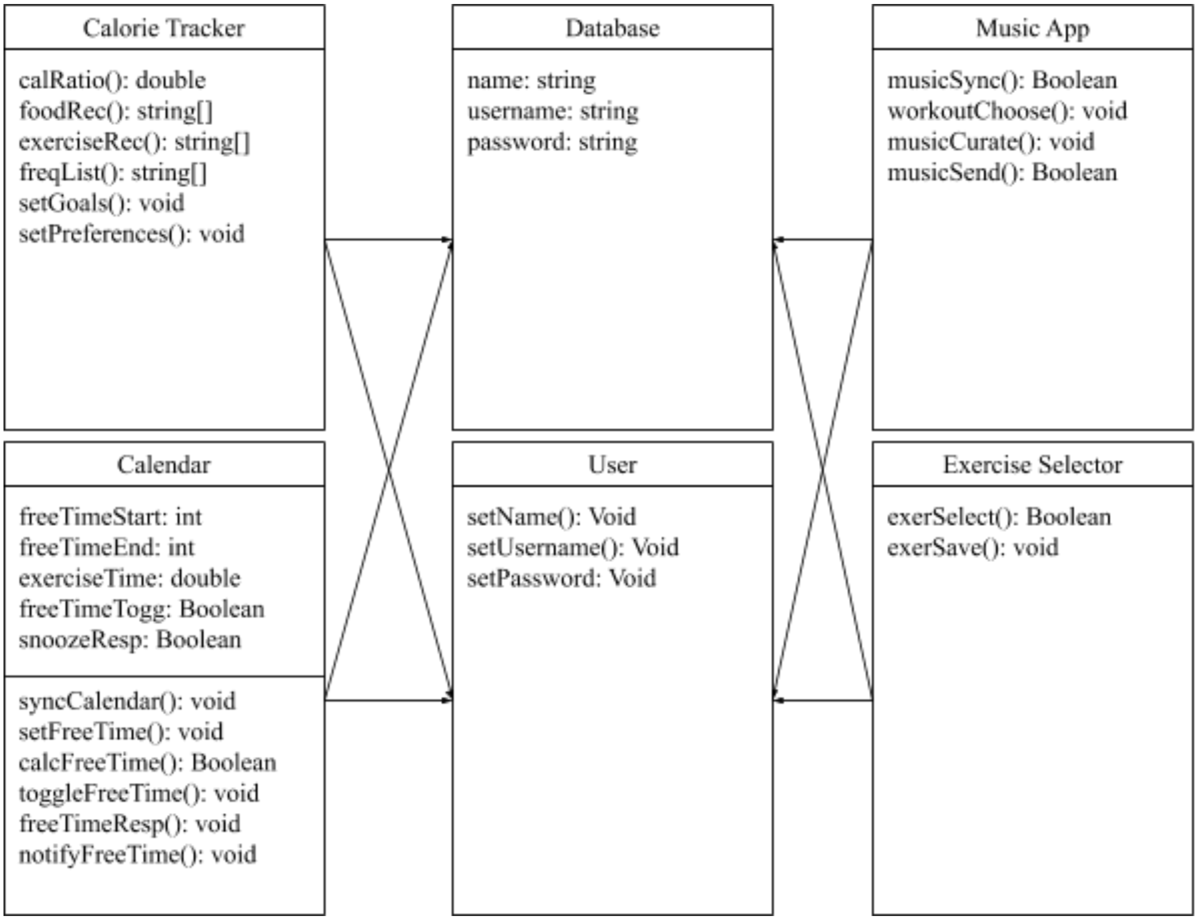
Description of Use Case #10:
The process starts with the “User” pressing a button to alert the “System” that the “User” is in some type of danger. The “System” grabs the user’s device information, the most important being the current location, and sends this information to the contacts listed on the “User’s” premade emergency contact list. The “System” notifies the “User” that this information has successfully sent. Ideally, the “User” will have their heartbeat monitored during this time. Here, we can use Liskov substitution principle (LSP) to create an algorithm that would work on any future interface that we wish to implement. This would make it easier to modify the feature depending on the needs of the User and make any necessary improvements.


User
System
3rd Party music app
Inputs intended exercises and genres
Gives music permissions
Contacts app via API
Curates playlist and saves info
Notifies of playlist
Requests to play playlist
Contacts app via API to play PL

Description of Use Case #13:
This use case shows the process of the creation of the curated music playlists. Once the "User" inputs the exercises they will be performing, music genres they wish to listen to, and allows the app to have access to its music permissions, the "System" is able to contact the "3rd Party Music App". After the "3rd Party Music App" responds with the needed information, the "System" is able to create a curated playlist that meets the "Users" inputted needs. Once the music playlist has been created, the "System" will notify the "User" indicating that their custom playlist is complete and is ready to be played. If the "User" indicated that they wish to play the songs, the "System" will contact the "3rd Party Music App" to retrieve the music requested in order to play the song. In this case, the Interface Segregation Principle is used, being able to understand each class in isolation.

Class Diagram and Interface Specification

Class Diagram



Data Types and Operation Signatures

Database:

name: string

String variable that stores the user’s name for the given account.

username: string

String variable that stores the user’s username for the given account.

password: string

String variable that stores the user’s password for the given account.

User:

setName(): Void

Sets the name for the given account.

setUsername(): Void

Sets the username for the given account.

setPassword: Void

Sets the password for the given account.

Calorie Tracker:

calRatio(): double

Returns the ratio between the number of calories consumed and burned

foodRec(): string[ ]

Returns a string array of food recommendations based on user goals and preferences

exerciseRec(): string[ ]

Returns a string array of recommended exercises that would help the user meet daily set goal

freqList(): string[ ]

Returns user's favorite items, based on a manual selection or based on most frequent exercises and meals

setGoals(): void

Sets the user's workout and calorie goals

setPreferences(): void

Sets the user's diet preferences

Calendar:

freeTimeStart: int

Integer corresponding to the time, during each day, in which free time notifications will be enabled.

freeTimeEnd: int

Integer corresponding to the time, during each day, in which free time notifications will be disabled.

exerciseTime: double

Double variable corresponding to the amount of free time needed for the user to implement a work out.

freeTimeTogg: Boolean

Boolean variable corresponding to the status of free time notifications (ON/OFF).

snoozeResp: Boolean

Boolean variable that is toggled "True" if the user snoozed the response and will be toggled "False" if a snooze was not pressed.

syncCalendar(): void

Sync the events from the user’s existing Google calendar with the calendar on the device.

setFreeTime(): void

Sets the amount of time needed for the user to implement a work out, and the time interval in which free time notifications will be enabled each day.

calcFreeTime(): Boolean

Calculate if there is enough free time within the user specified time interval for the user to use to exercise.

toggleFreeTime(): void

Enable or disable free time reminders.

freeTimeResp(): void

Stores the user's response to a free time notification.

notifyFreeTime(): void

Sends a notification to the user, during the specified free time notification interval, that there is enough free time to work out.

Music App:

musicSync(): Boolean

Gives the user an option to either sync their personal account to spotify or create a new one, and use a playlist generated by default.

workoutChoose(): Void

Allows the user to select which workout the user wants to complete

musicCurate(): Void

Syncs the workoutChoose() function with the playlist generate

musicSend(): Boolean

Sends the selected playlist to the user

Exercise Selector:

exerSelect(): boolean

Returns the amount of calories from selected exercise and adds exercise to user’s information

exerSave(): void

Adds the selected exercise to the user’s frequent exercises list

Traceability Matrix

Domain	Calendar	Calorie Tracker	Music App	Exercise Selector

Concepts				
Calendar	This domain concept mapped directly to its class.			
Calorie Tracker		This domain concept mapped directly to its class.		
Workout Notifications				These domain concepts were reduced to one class.
Workout Selector				
Music Playlist			This domain concept mapped directly to its class.	

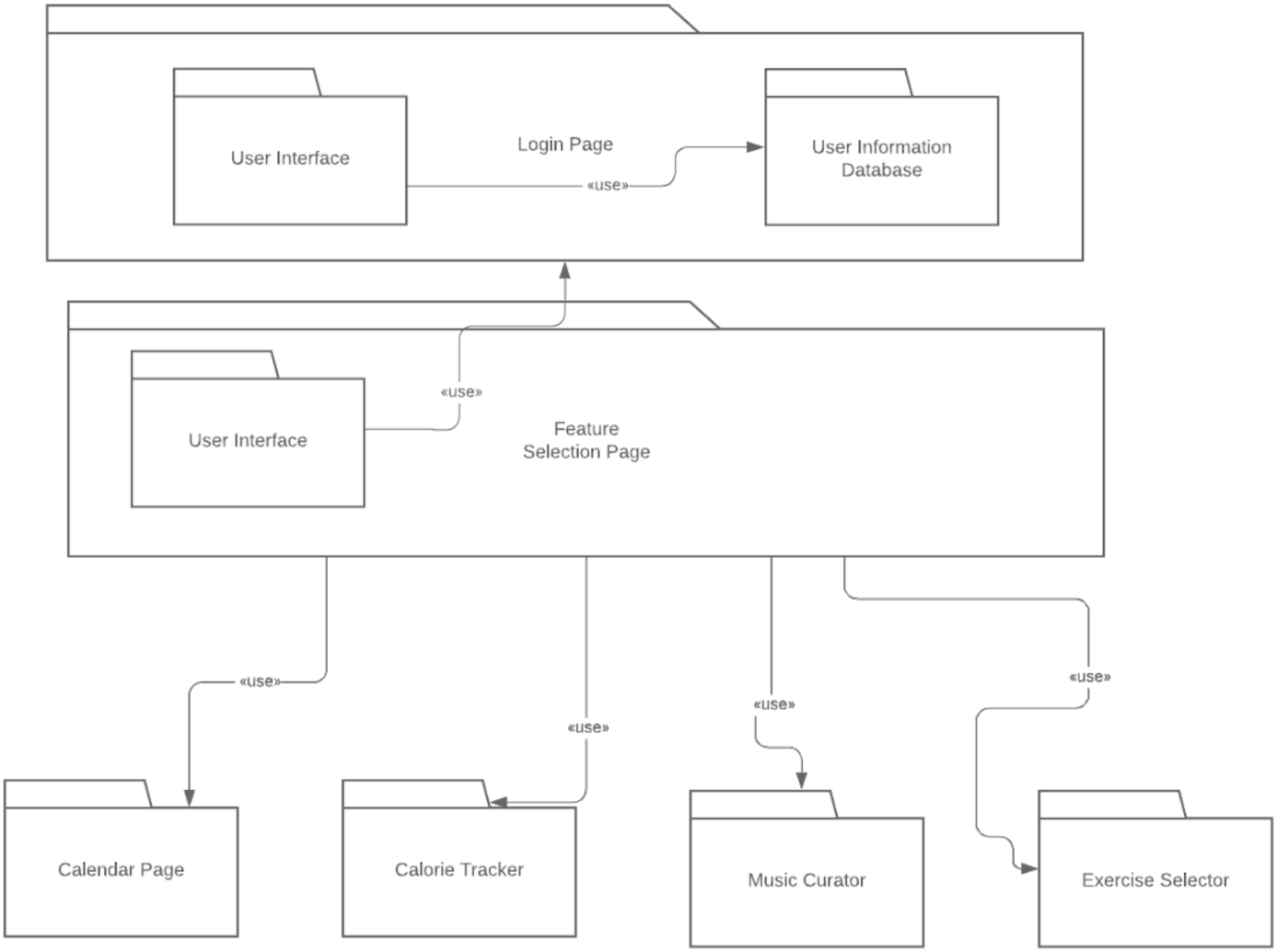
System Architecture and System Design

Architecture Styles

The architecture style of our system is largely component-based because the app is broken down into several logical subsections with mostly distinct functions. These subsections are our user options--i.e. the Calorie Tracker, Music Curator, etc. Instead of being dependent on one another, each subgroup has operations and attributes unique from other subgroups. This decreases the likelihood that bugs in one area will affect the entire code. In addition, since this particular type of app offers many services, separating the services streamlines the user experience. In effect, a user can select an option and only provide information needed for that subsection.

In addition, our system architecture is event-driven. Many sections of the app rely on user input and data. For example, the workout notification part relies on data from the user's Google Calendar, such that a change to the calendar changes the notifications. Consider also that the Calorie Tracker and Music Curator parts of the app rely on user inputted workout data and the user's heart rate. In this way, the app is very sensitive to user action.

Identifying Subsystems



Mapping Subsystems to Hardware

Our system does not need to run on multiple computers.

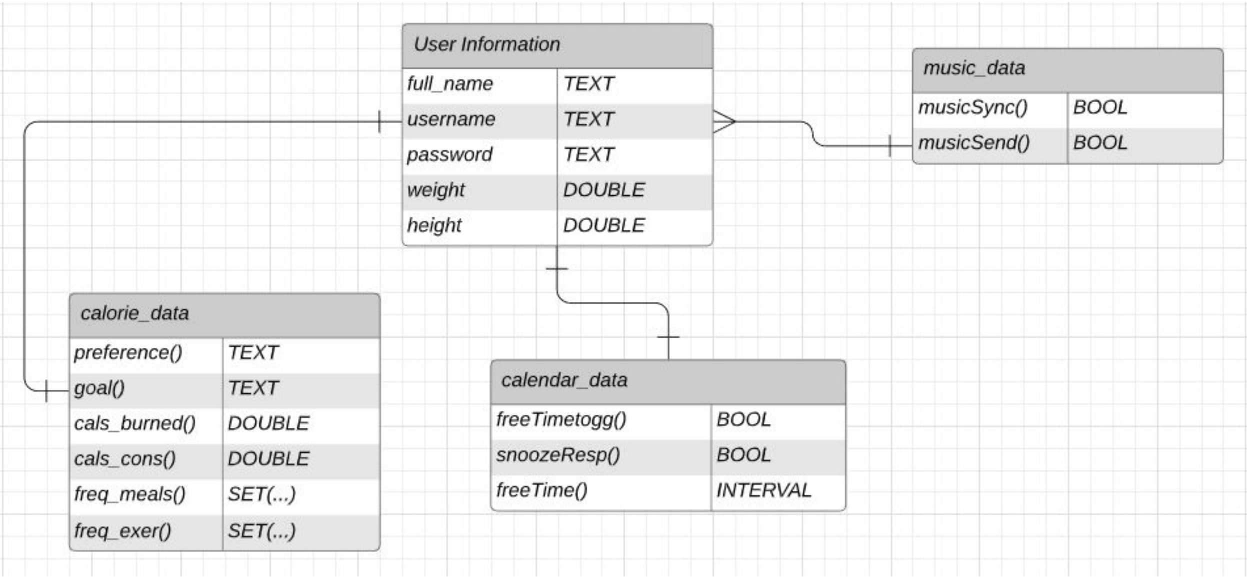
Persistent Data Storage

Our system relies on data that will outlive a single execution of the system, so we will use a relational database to store all of the users data.

Persistent Objects	Storage Management Strategy
User information (username and password)	Relational Database
User diet preferences	Relational Database
User music preferences	Relational Database
User health goals	Relational Database

User total calorie count	Relational Database
User designated Free Time	Relational Database
Workout Reminders	Relational Database
Saved Workouts	Relational Database
Saved Meals	Relational Database

Database Schema



User Information	
full_name	TEXT
username	TEXT
password	TEXT
weight	DOUBLE
height	DOUBLE

music_data	
musicSync()	BOOL
musicSend()	BOOL

calorie_data	
preference()	TEXT
goal()	TEXT
cals_burned()	DOUBLE
cals_cons()	DOUBLE
freq_meals()	SET(...)
freq_exer()	SET(...)

calendar_data	
freeTimetogg()	BOOL
snoozeResp()	BOOL
freeTime()	INTERVAL

Network Protocol

This does not apply to our group.

Global Control Flow

Execution Orderness:

Due to the different features of our project, our system is both procedure-driven and event-driven. The user must follow the same steps for the music and calorie features of the project. However, for the calendar, it is dependent on the previous events that are logged into the user's calendar already.

Time Dependency:

There are timers in our system. For the 'snooze' option, there is a timer in place to wait for 30 minutes before sending another notification to the user.

Hardware Requirements:

Feature	Requirements
Calendar Sync	Watch screen display: - size 34 mm or higher Android Display Server communication: Android→ watch Server communication access → Google calendar Network connection with minimum network bandwidth
Calorie Tracker	Watch screen display - size 34 mm or higher Android Display Calorie SQL database Network connection with minimum network bandwidth
Music Curator	Watch screen display - size 34 mm or higher Android Display Server communication→ Spotify Network connection with minimum network bandwidth (if premium option not selected)
Exercise Selector	Watch screen display - size 34 mm or higher Android Display

Algorithms and Data Structures

Algorithms

The calendar feature does use an algorithm. We use the Google Calendar API in order to be able to pull data from the user's pre-existing calendar information. In addition, the app is able to create new events in the calendar. This could allow for the user to keep a record of the days and time that they do work out. Utilizing the

API rather than creating a separate calendar within the app allows for us to integrate our app with what the user already has on their phone. Furthermore, it allows for our app to be created without “reinventing the wheel.” It also allows for the user to be provided with a seamless integration between the app and their calendar, which they previously spent time creating.

The calorie tracker feature does not use any complex algorithms.
The music curator feature utilizes the developer API plug-in provided by Spotify, which will be integrated within our app/OS as a shell. The component of the music curator which matches intended workouts to songplay implements an algorithm that plays songs with BPMs matching the pace of each exercise. This algorithm can be thought of as a song locator, and it requires information about preferred genres, communication with the spotify API, as well as some randomization so the same workout can result in different mixes. Another algorithm builds the playlist as each workout component is matched to a song.

Data Structures

The calendar feature does not use any complex data structures.
The music curator utilizes temporary memory via a linked list to build the playlist before it is saved. The music curator also uses an SQL database which is a part of the developer API provided by Spotify.
The calorie tracker feature stores the users most frequent exercises and meals using linked lists. Since the user can add data dynamically, we will use a linked list because the size of the list is unknown at compile time. However, the performance of linked lists is a drawback, but the flexibility is more important for our feature.

User Interface Design and Implementation

Calendar Sync

Only one modification has been made to the initial screen mock-up for the calendar page. Instead of providing the user with a view of their calendar, one that they would already have access to in their phone’s build in calendar app, our calendar page will consist only of the user’s preferences for free time notifications. This was implemented by using the same format and layout as the settings page and login page except with the substitution of dropdown menus for the input boxes.

Calorie Tracker;

For the Calorie Tracker section, we decided to create separate pages based on the different actions the user could take. This makes it easier for the user to navigate the app and make it clear what actions are associated with the tracker. There will be a Calorie Tracker home screen that will list the actions available and will have the overall intake of calories at the top of the screen. From there, the user could go to either the exercise selector screen or the adding calories screen. In the exercise screen, the user can choose what exercises they've done and that will be calculated. Then, the calories burned can be taken out of the Calorie Tracker. In the adding calories screen, the user can search the specific food items they've consumed and the corresponding calories will be added to the overall tracker.

Music Curator:

The modification that we decided to make was to include Spotify as a shell within our app and not make use of the Spotify app on the user's phone. This way the connection can be established through our application and the curator can work as it was intended to. The Spotify app integrated with our application will have features that the default app does not have. These features will be added by means of the android SDK which is a package included in the developer API provided by Spotify. For example, Spotify typically does not send push notifications to users with regards to the playlist, but the Spotify within our application, fitness friend, will notify the user each time a new playlist is generated, the reason being to increase the overall motivation aspect leading to a healthier lifestyle.

The Exercise Selector portion of the Music Curator app was modified to include separation of common workouts in subsections followed by division of those subsections into subsections, along with time and pace options. I.e. The subsections include cardio, muscle training, and yoga. The sections within those subsections are as follows: i.e. for cardio, when clicked the subsections include jogging, walking, elliptical, running, jumping jacks, etc, with additional options to include the number of reps or length. The end of exercise selection for one exercise gives the user the option to add another or finish.

Design of Tests

Calendar Sync

For the free time notification system, the Google Calendar API will be integrated in order to pull from a user's pre-existing calendar. To test whether or not the calculation system works, we will have to create a Google account and begin operating as a user to see if we receive the necessary notifications to properly utilize this feature. This notification will notify the user if there is enough free time throughout the day, during the free time interval, to implement a workout. We will be testing the integration of the API and free time calculation system utilizing the following steps in the app:

1. The user will set their free time notification interval and the amount of free time needed for a proper workout page on the calendar page
2. The user will sync their Google Calendar to the app
3. The app will retrieve the events from the synced calendar and calculate whether enough free time is available during the pre-set free time interval
4. If the test is successful AND there is enough free time during the day, the app would output the expected free time notification

Google Calendar Scenarios:

- There is enough free time during the free-time interval specified.
- There is no free time available during the entire day.
- There is enough free time but it is **not** during the free-time interval specified.
- There are numerous free time slots during the free-time interval specified.
- There is free time but not enough available as determined by the user.

Calorie Tracker:

For the calorie tracker, we will focus on how the calories are calculated based on the exercises selected and the foods searched. We will be testing our calculation algorithm to check the accuracy of the overall calories displayed on the screen as well as the accuracy of the calories assigned to each exercise and the food calories found based on the search. It would be difficult to get the exact numbers, but the goal is to get them as accurate as possible so the user doesn't have to do any calculations by hand. Here are some actions the user should be able to perform:

1. The user finishes a specific exercise and can go to the exercise selector screen to choose the exercise from a list. The burned calories associated with that activity can then be subtracted from the overall calories.
2. The user finishes consuming a meal and can head to the adding calories screen and search for the foods they ate. The calories corresponding to each food will be added to the overall calories.
3. The user checks their overall calories for the day to see if they have reached their goal.

Calorie Tracker Scenarios:

- The food searched is **not** found.
- The amount of calories burned exceeds the amount of calories consumed.
- The portion of food eaten does **not** correspond to the calories shown.
- The exercise completed is **not** an option on the list.

Music Curator

For the music curator, we will integrate Spotify within our app as the first demo. For this we will be using the developer API- Android SDK via the following steps:

- First an application is registered with the spotify dashboard which provides us with a client ID as well as URI. So that the app relaunches every time the user logs in.
- The Android SDK is then used to create a link between the application server and the Spotify database.
- If successful, the user should be able to log into spotify via the application we have created. In turn, giving our API access to their account.

In addition to the Spotify API connection, the matching component of the music curator which matches user-intended exercises to a song matching pace and style preference will have to be extensively tested for accuracy using test cases. An ideal scenario for the use of this algorithm is:

1. The user, John, is prompted to enter exercises. Using the exercise selector subsection of the music curator, John enters that he wants to do 15 minutes of medium paces jogging, followed by 20 pushups, 100 sit-ups, and 10 minutes of walking to cool down.
2. The curator returns a Spotify playlist with 15 minutes of music with a high BPM of around 150, then lower BPM music fit for anaerobic exercise for about 3 minutes (estimated using the algorithm), then 10 minutes of calm BPM music around 50 BPM for John's cool down.
3. If John enters this same workout again when reusing the music curator, step 2 should be carried out with a slightly different list of songs (via randomization).

Calendar Sync	This feature will be testing the function coverage: whether or not the free time notification feature is able to pull from an existing calendar using the Google calendar API and accurately determine if there is enough free time to work out during the free time interval. This is very important because the integration of the API will help with efficiency. Majority of technology users use Google Calendar as their default. So instead of allowing the user to input their schedule twice, we will use what they have already created. This will better integrate it so the user experience is better and more comfortable.
Calorie Tracker	This feature will test the accuracy of the calculation algorithm implemented in the back end. It will test the calculation of calories for the foods consumed and for the exercises selected.

	Adjustments will be made as needed based on the results of the tests.
Music Curator	For this feature, we will be testing the function coverage, whether or not the user is able to log into/create their own Spotify account through the web. We will be using an Android SDK provided as a part of the Spotify developers API to establish connection between our application and the Spotify server. For this feature we are also testing the accuracy and successful randomization of our song matching algorithm given the multiple inputs about user preference.

Feature	Integration testing
Calendar Sync	The integration feature for the calendar sync involves incorporating the google calendar API and using it in the app. The calendar integration will have the users most recent events and be able to distinguish the free time the user has. To figure that out, the app would incorporate the events/tasks the user has and find blocks of time where the user is not scheduled for anything, and then notify the user for suggested times/reminders to work out. If the user has some events and has time in between them and our app recognises that free time block. Then we know it works, because the app will create a free time slot, and send the notification to the user. If the notification is sent, then the calendar sync feature works successfully.
Calorie tracker	The integration feature for the Calorie Tracker involves incorporating a calorie database to the app. The calorie database will be made beforehand and will consist of calories assigned to common food items. The user can select any of the food items in the database and the result will be fed into the calculation. The same approach will be used to select the exercises completed.
Music Curator	The integration feature for the curator involves connecting the dummy app to integrate spotify as a shell. This is the main testing that has to be done, as everything after the spotify app integration involves the use of the android SDK API. The algorithm will be tested by launching the app, which will then enable the user to log in to Spotify via the app. If the user succeeds, our first demo testing is complete; however, if there are failed login attempts then some debugging will have to be done. The testing will be performed by the developers as well as testers.

Project Management and Plan of Work

Merging the Contributions from Individual Team Members

The largest issue that was faced was the lack of knowledge in creating an app. In addition to the general creation, many of us were not familiar with the languages, such as Javascript, needed to do so. After much research and outside advice, we decided to use snack.expo to create our app's interface. Fortunately for us, snack.expo provides a lot of well-explained documentation as well as examples that have allowed us to be successful in the creation of our app.

In addition to this issue, the group also faced a smaller issue midway through the first half of the semester. During the initial creation of our group, four subgroups were created; three groups of two and one group of three. After multiple reports and team meetings, it was determined that the best direction for our project to follow would be to split one of the subgroups containing two people into the two other subgroups also containing two people. The subgroup that was initially working on the SOS feature was the group that was split up due to their feature having already been implemented on the Apple watch.

A final issue that our group is dealing with a new change in operation. Our weekly meetings were removed due to the immediate evacuation of Rutgers. Though we didn't have a Project Manager, we did agree on a time - as a group - where there was at least one representative per subgroup. This meeting allowed for each group to get a look into what the other groups' statuses are. Currently, each subgroup has been communicating utilizing their preferred method [video conferencing, group messaging, etc.]. We plan to overcome this issue by utilizing When2Meet to find a time where the entire group can work on the project together. We have found that working on the project simultaneously allows questions to be answered regarding shared interfaces and ensures that each group member is being held responsible for providing some contribution to the overall project.

Project Coordination and Progress Report

What use cases have been implemented?

For the music curator, we are in process of finishing up linking the app with Spotify; as this is one of our integral/base features, upon which we will code/add some advanced features using SDKs. For the exercise selector, adding the massive list of possible exercises is in the process of being implemented.

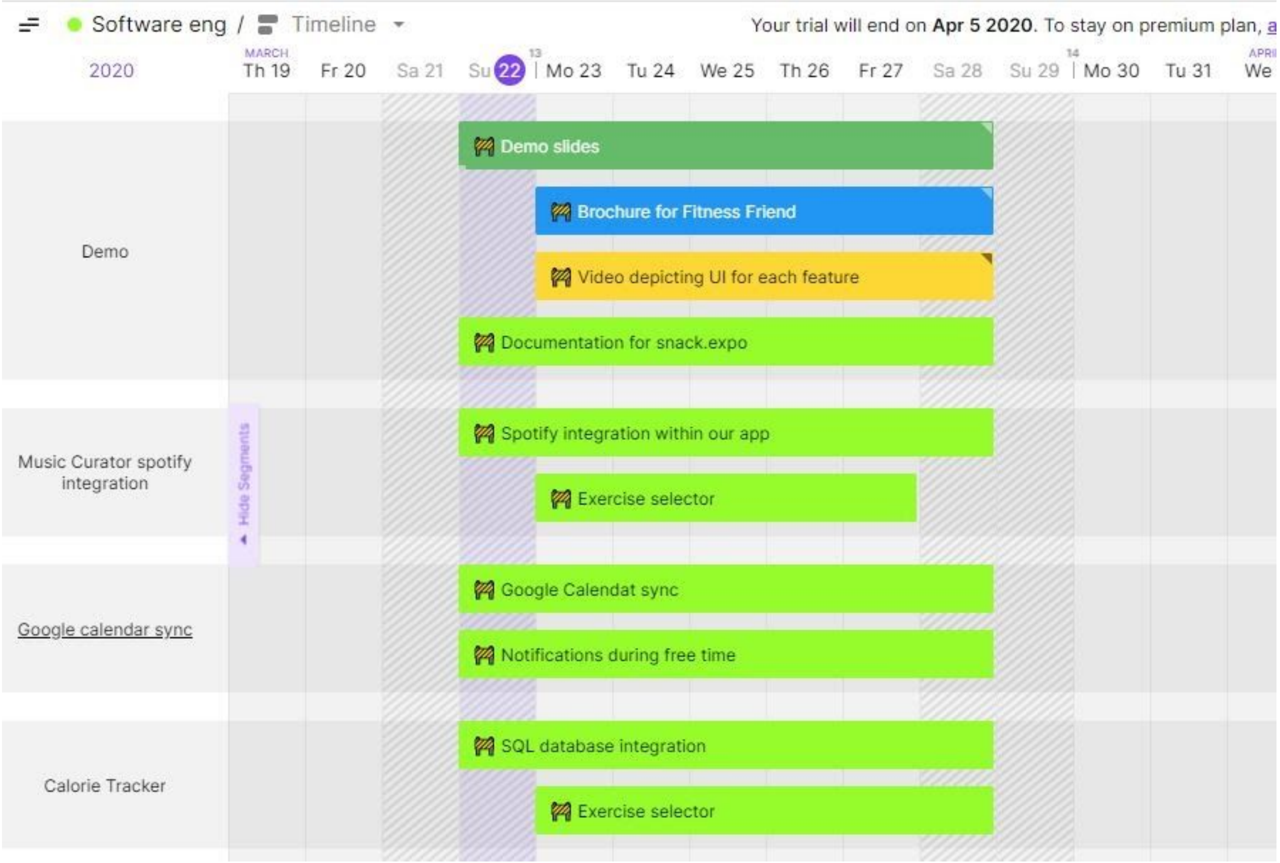
What is already functional, what is currently being tackled?

The calendar sync group has been successful at the creation of their three pages with working buttons and dropdown menu options. Upon the creation of an account database that will be able to store the user's information and preferences, the pages will be fully completed. The database that will be used to create user's accounts, as well as pull information (for preferences and settings and to log in), is still currently being tackled by the group as a whole.

For the music curator, the connection with Spotify is nearing completion, whereas the algorithm linking exercises to playlist curation, another core feature, is still being immensely researched and drawn out. This involves immense research into exercise pace and how exercise pace varies depending on age and health level. This algorithm needs to be very accurate in order to be considered successful. Else, the music curator will not be very customisable. Research is also being done on BPM in songs in relation to exercise and being able to reach a target heart rate.

The calorie tracker has made the interface of the screens uniform and are working on putting the database together. From there, we will connect the database to the backend of the app. The numbers returned from the database will be used in our calculation algorithm. The calculation algorithm will allow for the overall calories to be displayed on the screen for the user's convenience.

Plan of Work


Your trial will end on **Apr 5 2020**. To stay on premium plan, a

Breakdown of Responsibilities

Subgroup 1: Tiyon King, Jenna Krause, Andrew Rezk

Tiyon King: Google Calendar API Integration

Adding events

Snoozing exercise notifications

Storing the user’s response

Toggling free time notifications

Jenna Krause: Interface and Push Notifications

Calendar, User Settings, Login Pages

Notification integration of all app’s features together

Setting free time interval and workout time

Notifying free time

Andrew Rezk: Google Calendar API Integration

Calculating free time

Creating test Google Calendar accounts
Syncing Google Calendar
Subgroup 2: Mya Odrick, Maria Rios, Hedaya Walter
Calorie Tracker
HomeScreen Interface
Mya Odrick
Food Input Interface
Calculating amount of calories from food
Maria Rios
Calorie Tracker Main Interface
Formulating the ratio of calories in and out
Hedaya Walter
Exercise Input Interface
Calculating amount of calories from exercises

Subgroup 3: Amber Haynes, Shivani Sunil, Devvrat Patel
Music Curator
Shivani Sunil: Spotify API integration
Exercise Matching
Amber Haynes: Exercise Selector and Matching
Spotify Integration

Integration will be coordinated by the developers of the app which include all the members from each subgroup, with each subgroup representing a feature of the fitness friend. Each sub aspect of the feature will be tested by the individual it was assigned to. This will ensure that each of the subfeatures of each main feature work accordingly. In other words, unit testing as well as the documentation for it will be done individually and the pieces will then be put together to create a fully functional and integrable feature.

After the unit testing has been successfully completed, each subgroup will select one member who will coordinate the integration testing of the feature, as a whole. This will ensure clarity among each group and also contribute to the overall efficiency. The fact that we are using snack.expo will make it easier to come up with test drivers and test stubs. However, the integration testing will also be done by the testers.br

References

1. "Software Engineering book", Ivan Marsic
   http://www.ece.rutgers.edu/~marsic/books/SE/book-SE_marsic.pdf
2. “SQL Data Types for MySQL, SQL Server, and MS Access”
   https://www.w3schools.com/sql/sql_datatypes.asp
3. Spotify Android SDK
   Android SDK
4. Snack expo app development environment
   Push Notifications