Aaron Fox

CSE 570

Final Project

12/02/2020

Brief Description and Introduction of Assignment

An Android project was designed, implemented and tested using the Java programming language and Android Studio. Three different databases were used with the SQLite helpers such that the user could save and edit custom exercises, workouts, and record and edit workout histories. This application was called Work(out) Smarter, and it allowed for the user to keep up with their workout plan such that they could plan and organize different workout routines, exercises, and keep up with their past workouts that they have done. there are three man tabs to this application: a main tab which allowed for the entering and editing of new exercises and the creation and editing of workout plans and view those workouts and exercises using Spinners and Recycler Views, a workout history tab which allowed the user to keep track of past workouts and view their past workouts in a Recycler View, and a Map tab, which allows the user to view local gyms around the user. The full map API was not able to be completely implemented because the Google Places API only allowed a limited amount of calls to find nearby gyms, so the full-fleshed implementation was unable to be completed of that part of the Places API since it required payment. The user can still zoom to their current location, with permission, and look for local gyms around them, however.

Description of Logic and the Input/Output

Upon first opening the application, the user is taken to the main activity and tab. In this tab, the user can provide input by clicking on the navigation tab bar at the bottom to navigate between the main, workout history, and map tabs. In the main tab's opening Activity, the user can do three main things: view the Recycler View which contains all their current workouts, press a Delete Switch which allows for the deleting of current workouts the user can see in the Recycler View, and an Add New Workout button, which allows for the creation of a new workout routine made up of several exercises. If the user clicks on a list item in the Recycler View, they will be taken to the Workout List Activity where they can then edit and update a current workout routine, including the workout's exercises and name. If the user then provides input by clicking the Save button, then the user is taken back to the main activity and can view their updated workouts, where the workouts list the amount of exercises and the workout name in the Recycler View. In the Main Activity, the user can also toggle the Delete switch which will bring up DELETE buttons next to every item in the Recycler View. If the user provides input by clicking one of these DELETE buttons, the workout will be deleted from the Workout SQLite database.

If the user clicks the Add New Workout button, they are taken to the Workout Planner Activity which allows them to add exercises to make up their own custom workout plan. In this activity, the user can select a workout group to add from a muscle group spinner that, in turn, changes which exercises are available in the Add Exercise spinner below it. When the user has provided the input of selecting the exercise they desire from a database saved in the Exercise Database, the user can click the ADD button to add an exercise to a workout plan. If the user would like to delete an exercise, they can toggle the Delete switch to bring up DELETE buttons next to every exercise currently in the user's workout plan. Deleting an exercise from the workout only removes it from the workout, not the Exercise database. However, if the user would like to delete an exercise from the database, the user could click on an exercise in the current workout plan and then click on the “DELETE EXERCISE FROM DB” button to permanently delete an exercise from the Exercise database. Additionally, the user can edit the exercise name, calories burned, muscle group, and the number of reps in the Add/Edit Exercise activity that the user is taken to. The user can only save the exercise if all the proper forms are filled out. Otherwise, a Toast warning the user that they have to fill in all required information pops up and blocks the saving of the Exercise to the Exercise database.

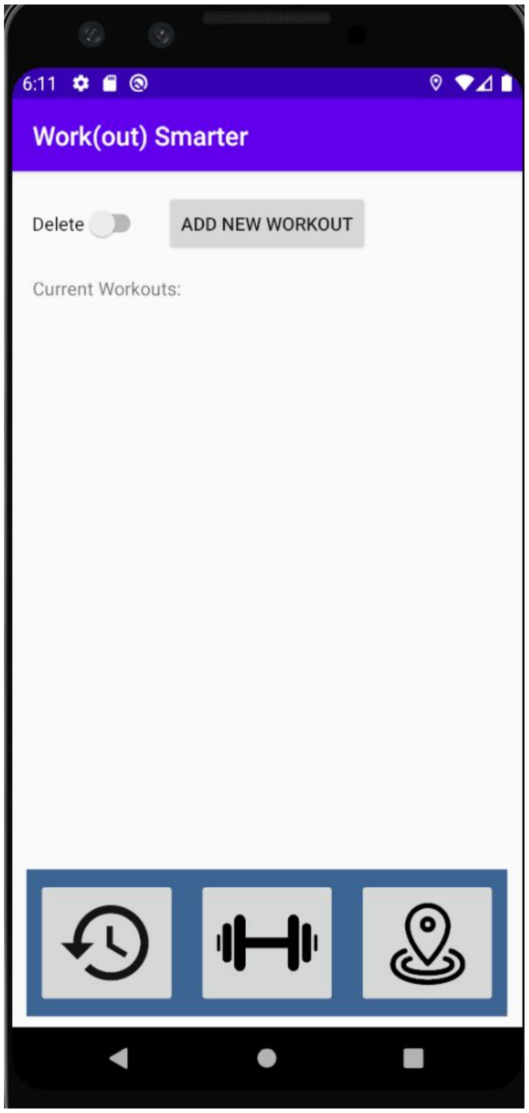
The user can also add their own custom exercise by clicking the “ADD CUSTOM EXERCISE” button which will take them to the Add Exercise Activity. This activity allows for the creation of a brand new exercise and requires an input of an exercise name, calories burned, reps required, and the user must select a muscle group from the muscle group dropdown spinner. Once the user has all their required workouts in their Workout Planner Activity and a workout name filled out, the user can Save the workout by clicking the Save button. If the user has not filled out the name of the workout and has not added at least one exercise, a Toast pops up warning the user that they must enter the info and blocks any changes from being made. The user can then see all their current workouts in the main tabs Recycler View.

In the Workout History tab, the user can select a workout from their list of saved workouts in a Workout Spinner. They can also select a date from the DatePicker. Once the user has their date selected and a workout selected, the user can click the “Record Workout” button so that they can record all their workouts to their PastWorkouts database and scroll through them in a Recycler View. The Recycler View outputs the amount of calories each workout burns, the workout name, and the date the user works out. Like the previous Recycler Views, the user can select the Delete toggle to bring up DELETE buttons that can then delete past workouts that the user has submitted.

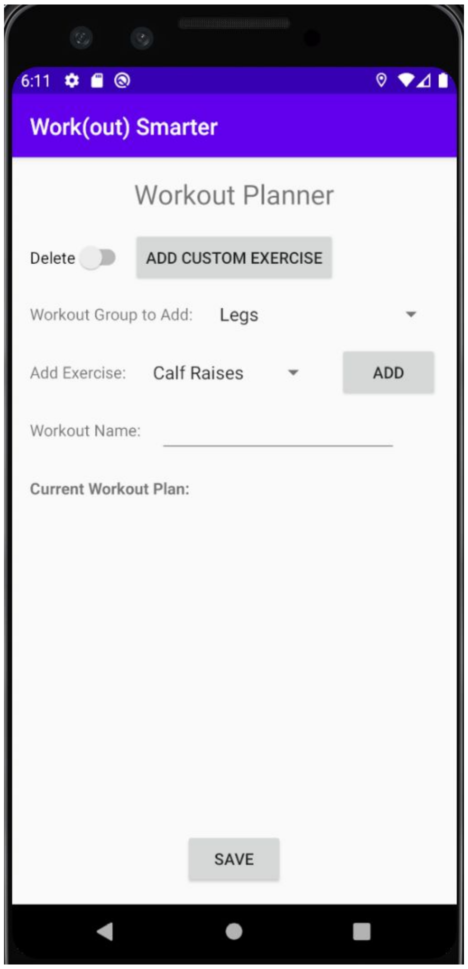
Lastly, the user can provide input by clicking the Maps tab, which will allow them to use Google Maps to scroll to their location and search around for gyms near them. Originally, the maps was able to pinpoint gyms near the user, but that was removed after the Google Places API started charging for its use after a certain amount of calls. The basics of the Google Maps API are therefore all that is used. The user can provide input by clicking the GET LOCATION button which will record their current GPS coordinates as well if the user gives the application permission to get its fine and coarse positions from the phone’s GPS.

Snapshots of Inputs and Outputs

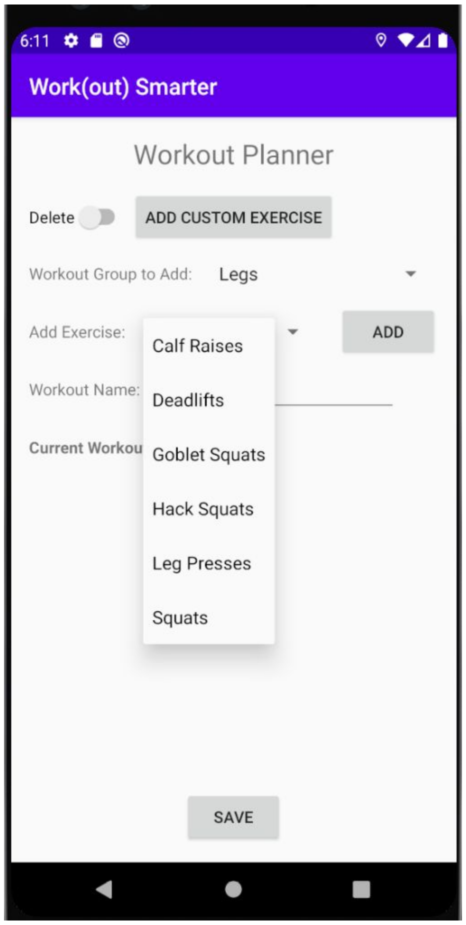
When first opening the application with no workout data saved, the application looks like



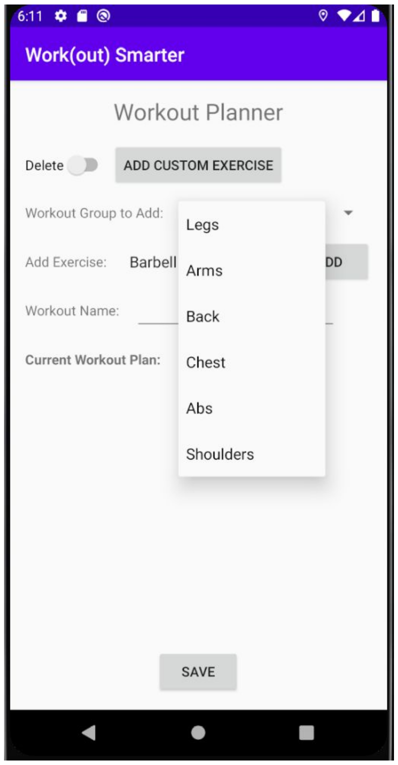
On this tab, the user can view the empty Recycler View. The Delete Switch does not yet do anything because there are no Workouts to delete yet. The user can add a new workout by clicking on the Add New Workout button, which leads to the Workout List Activity:



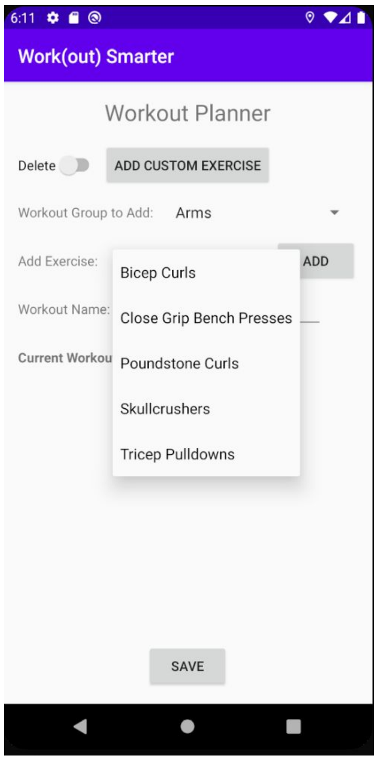
This tab allows the user to dynamically add exercises by category. For example, if they want to select a Legs type of exercise, they must only select the Legs workout group in the Workout Group to add. This changes which exercises are available to add in the Add Exercise Spinner, like so:

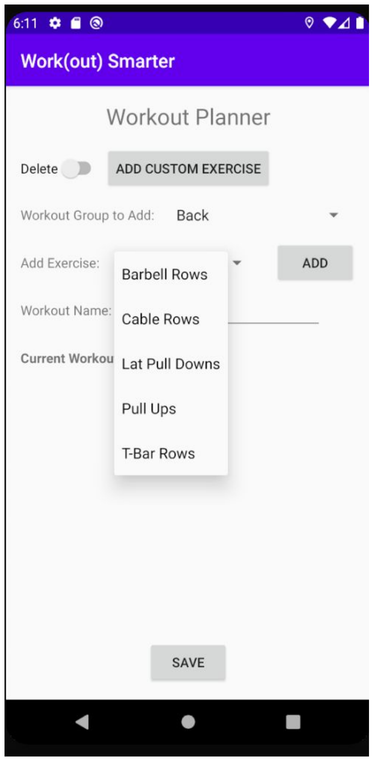


If the user changes the workout group to Arms or Chest, for example, they'd see different workouts to add. They can change the workout group to add by clicking on the Workout Group to Add Spinner:

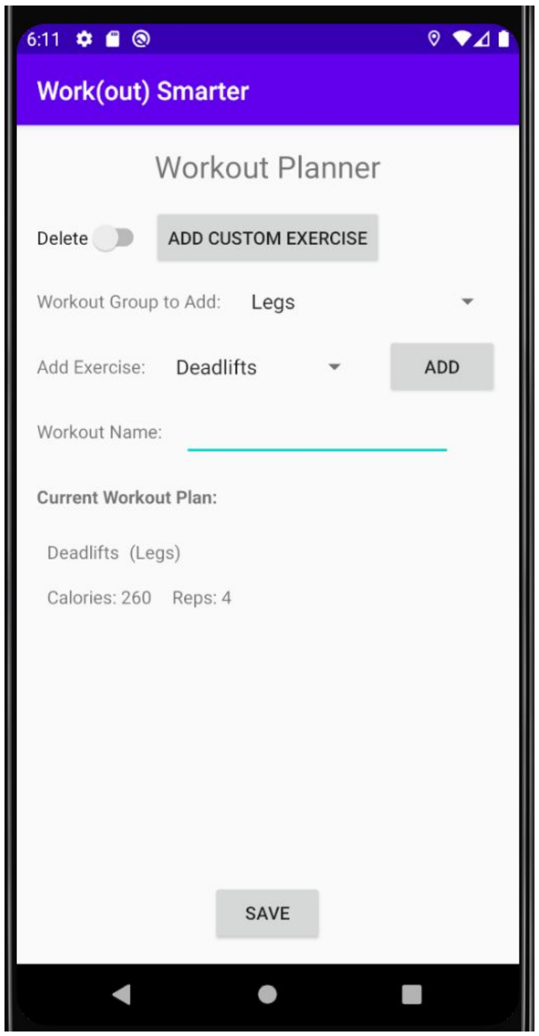


And, depending on which group they choose, they will see several different options of exercises to add, like so:

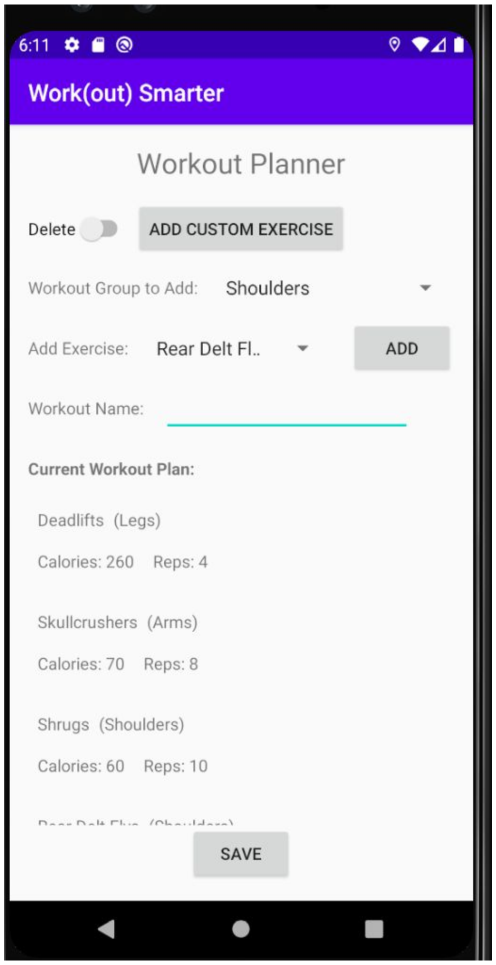




Once the user sees an exercise they like, they can add it to their current workout plan like by clicking the Add button, like so for the Deadlift exercise:



The user can add several different exercises to their liking:


6:11

Work(out) Smarter

Workout Planner

Workout Group to Add: Shoulders

Add Exercise: Rear Delt Fl..

Workout Name:

Current Workout Plan:

Deadlifts (Legs)

Calories: 260 Reps: 4

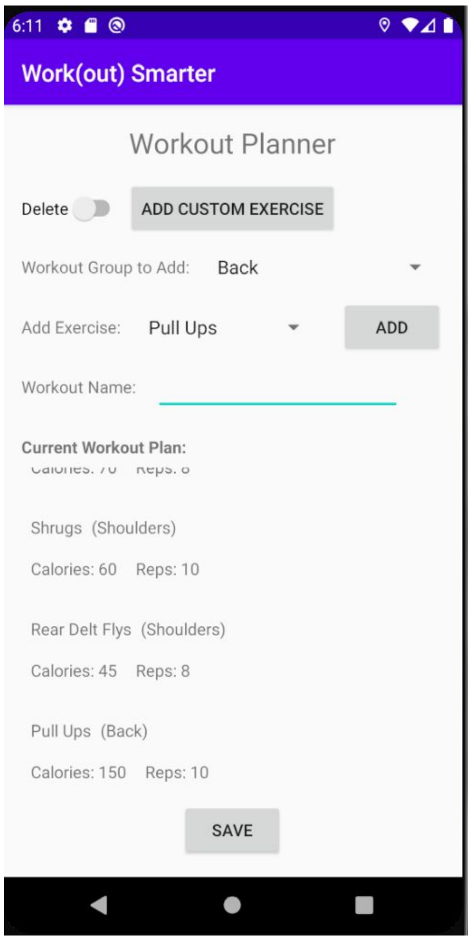
Skullcrushers (Arms)

Calories: 70 Reps: 8

Shrugs (Shoulders)

Calories: 60 Reps: 10

The user can scroll up and down on their list of exercises:


Work(out) Smarter

Workout Planner

Delete

Workout Group to Add: Back

Add Exercise: Pull Ups

Workout Name:

Current Workout Plan:

Calories: 70 Reps: 8

Shrugs (Shoulders)

Calories: 60 Reps: 10

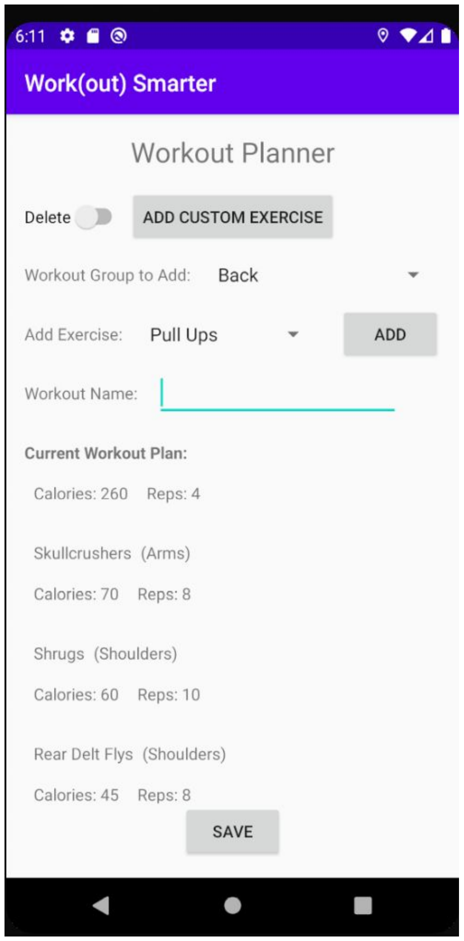
Rear Delt Flys (Shoulders)

Calories: 45 Reps: 8

Pull Ups (Back)

Calories: 150 Reps: 10

=>


Work(out) Smarter

Workout Planner

Delete

Workout Group to Add: Back

Add Exercise: Pull Ups

Workout Name:

Current Workout Plan:

Calories: 260 Reps: 4

Skullcrushers (Arms)

Calories: 70 Reps: 8

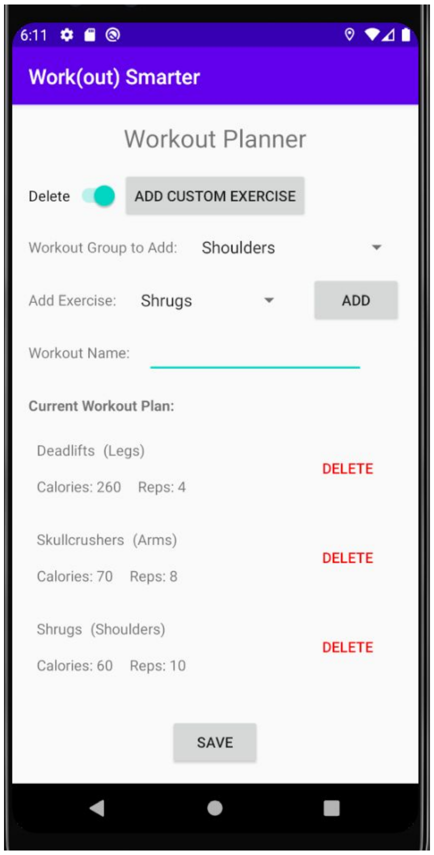
Shrugs (Shoulders)

Calories: 60 Reps: 10

Rear Delt Flys (Shoulders)

Calories: 45 Reps: 8

If the user would like to remove an exercise from their workout plan, they can achieve that by selecting the Delete switch:



Workout Planner

Workout Group to Add: Shoulders

Add Exercise: Shrugs

Workout Name:

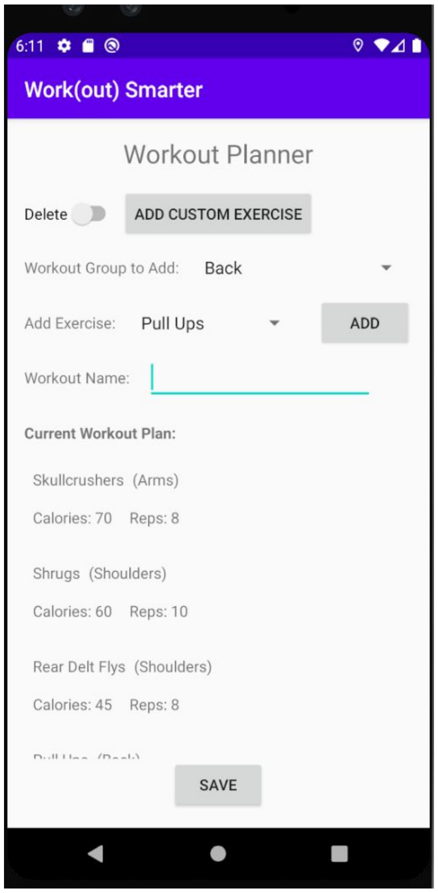
Current Workout Plan:

Deadlifts (Legs)
Calories: 260 Reps: 4

Skullcrushers (Arms)
Calories: 70 Reps: 8

Shrugs (Shoulders)
Calories: 60 Reps: 10

Clicking the Delete Button will then remove an exercise. After deleting the first exercise, Deadlifts by clicking the DELETE button, the Deadlifts exercise is then removed from the current workout plan:


6:11

Work(out) Smarter

Workout Planner

Delete ADD CUSTOM EXERCISE

Workout Group to Add: Back

Add Exercise: Pull Ups ADD

Workout Name:

Current Workout Plan:

Skullcrushers (Arms)

Calories: 70 Reps: 8

Shrugs (Shoulders)

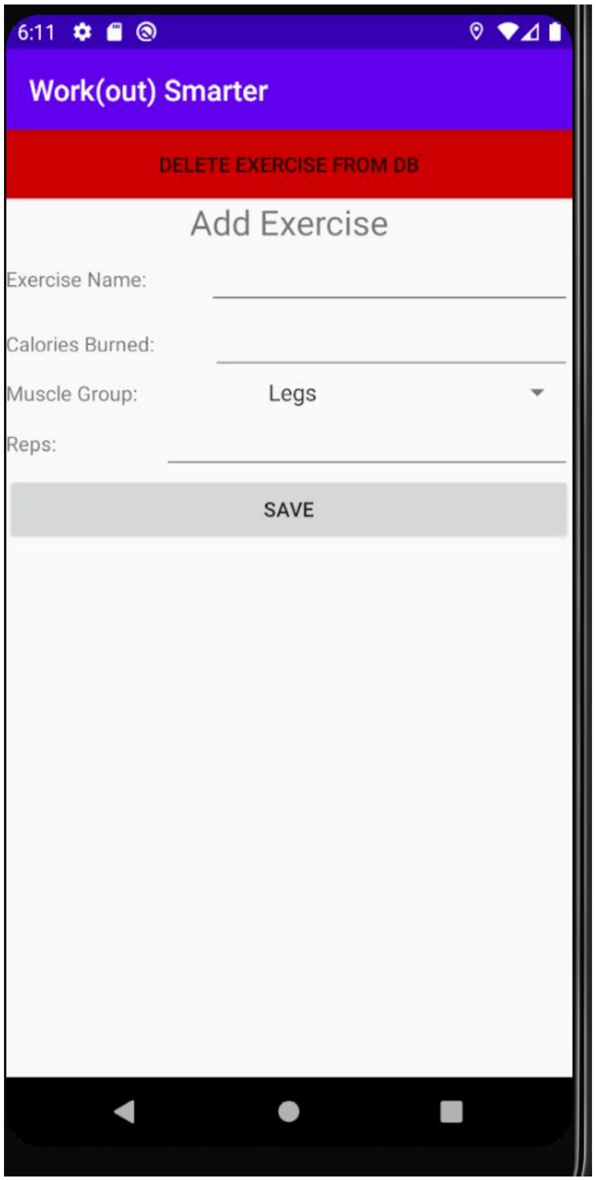
Calories: 60 Reps: 10

Rear Delt Flys (Shoulders)

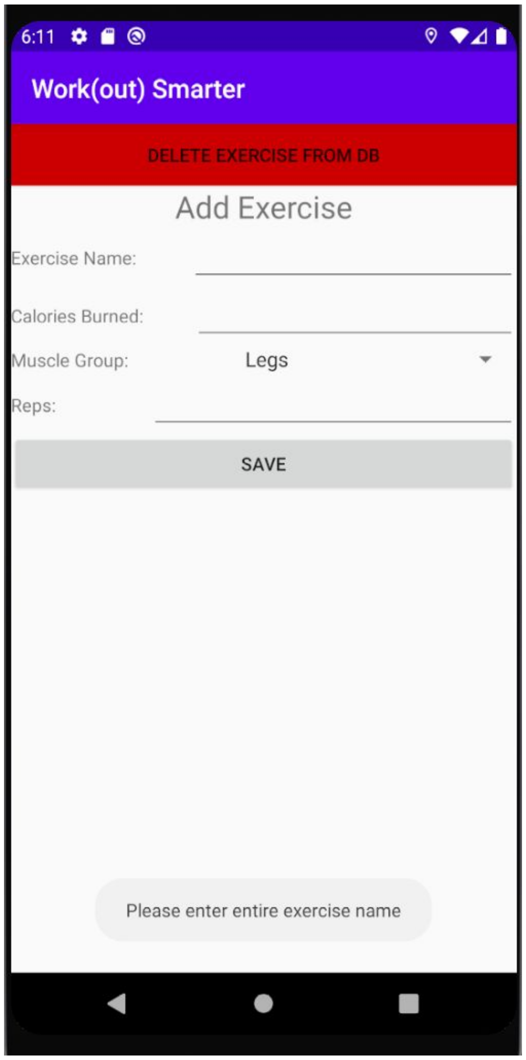
Calories: 45 Reps: 8

SAVE

If the user has a custom exercise they would like to add, the user can click the Add Custom Exercise button so that they can add their own exercise, leading to the Add Exercise Activity:



The user can select a muscle group from the spinner, add a name, calories burned, and number of reps, and then save it. If they don't add in the proper data, a Toast warning shows up to the user, like so:



Exercise Name:

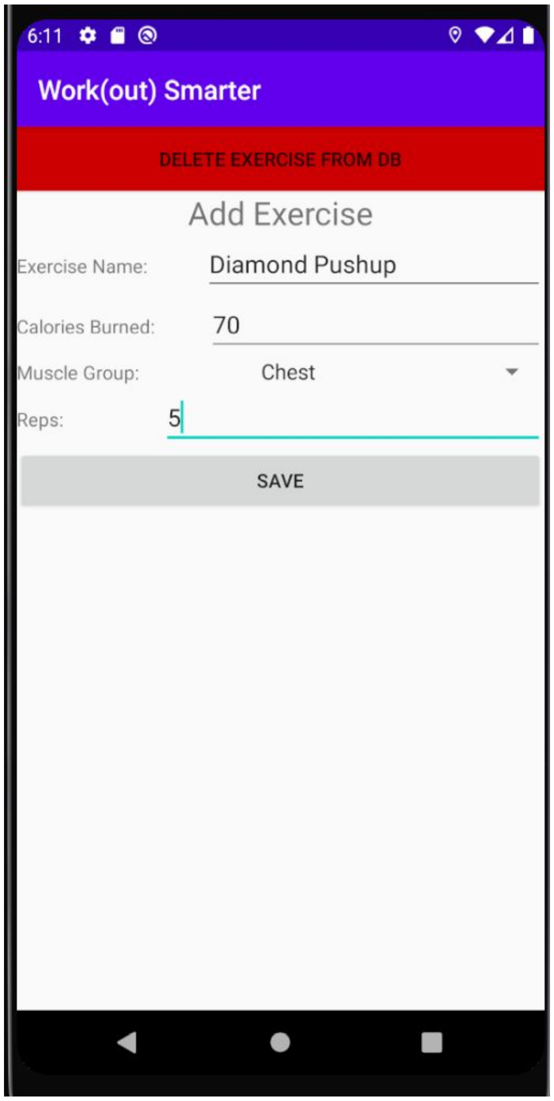
Calories Burned:

Muscle Group: Legs

Reps:

Please enter entire exercise name

Let’s say a user entires the following information an adds it to the Chest muscle group of exercises, like so:



Add Exercise

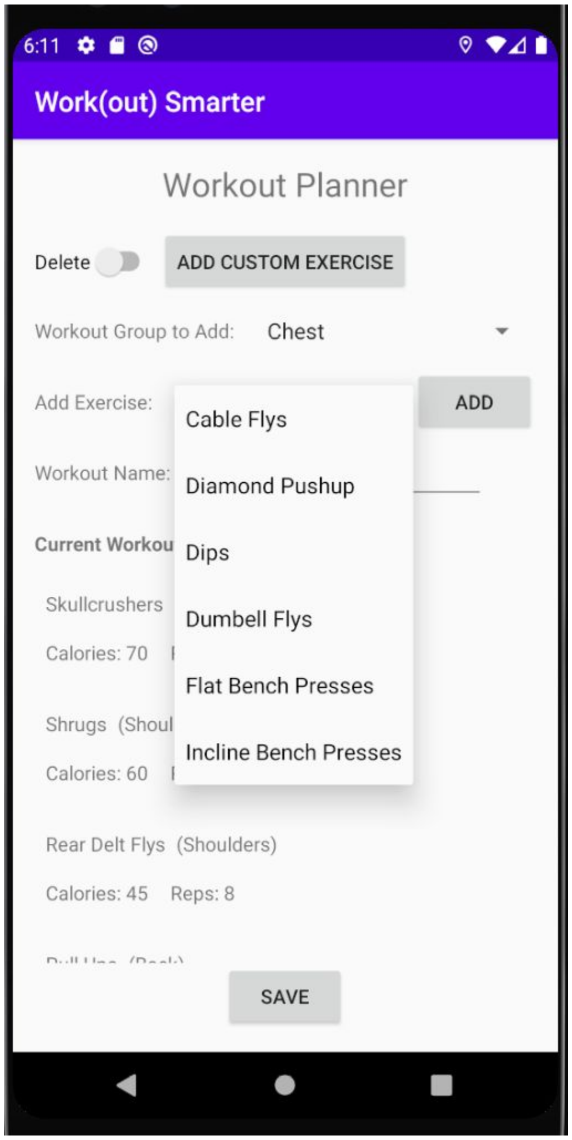
Exercise Name: Diamond Pushup

Calories Burned: 70

Muscle Group: Chest

Reps: 5

Once the user presses the save button, they can then add it to their workout:


6:11

Work(out) Smarter

Workout Planner

Delete ADD CUSTOM EXERCISE

Workout Group to Add: Chest

Add Exercise: ADD

Cable Flys

Diamond Pushup

Dips

Dumbell Flys

Flat Bench Presses

Incline Bench Presses

Workout Name:

Current Workou

Skullcrushers

Calories: 70

Shrugs (Shoul

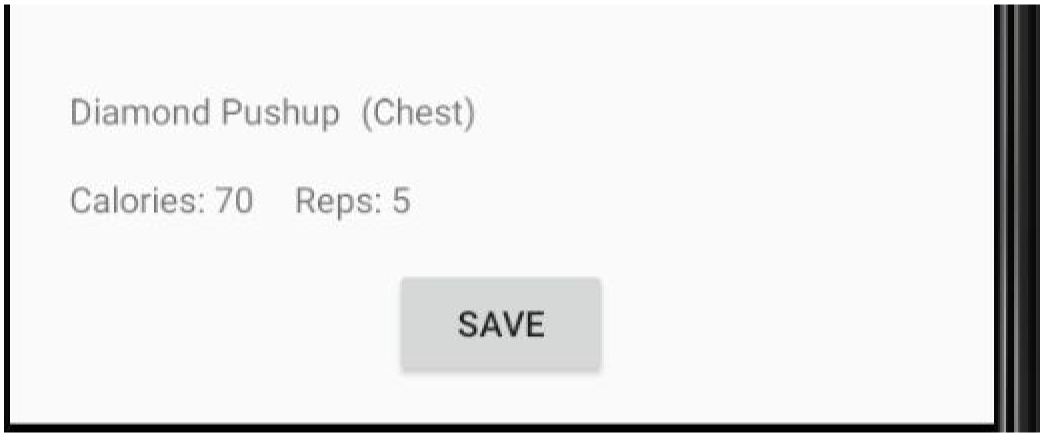
Calories: 60

Rear Delt Flys (Shoulders)

Calories: 45 Reps: 8

SAVE

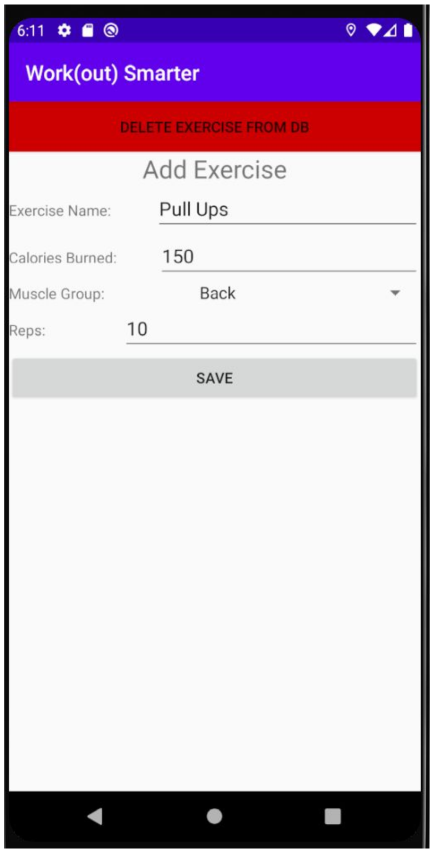
The user, when adding the diamond pushup to their workout, can then view its relevant data:


Diamond Pushup (Chest)

Calories: 70 Reps: 5

SAVE

If the user would like to edit an exercise, say Pull ups , they can click on the Pull Ups list item in the Recycler View to edit it:


6:11

Work(out) Smarter

DELETE EXERCISE FROM DB

Add Exercise

Exercise Name: Pull Ups

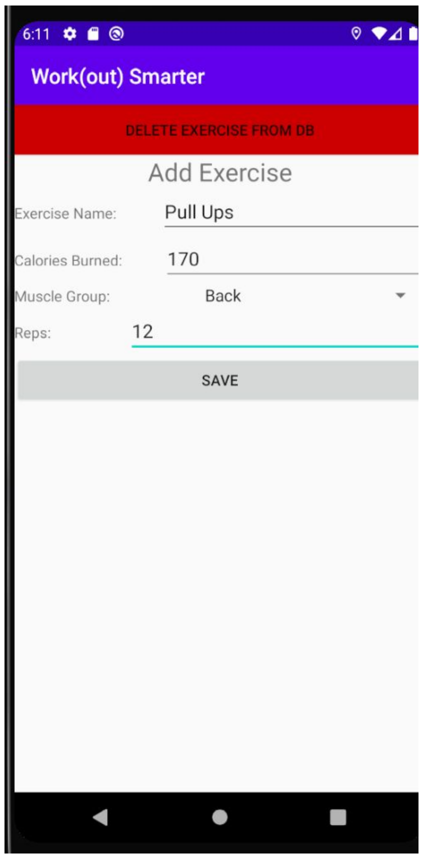
Calories Burned: 150

Muscle Group: Back

Reps: 10

SAVE

=>


6:11

Work(out) Smarter

DELETE EXERCISE FROM DB

Add Exercise

Exercise Name: Pull Ups

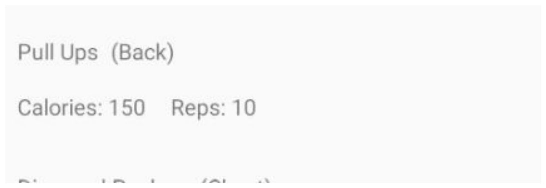
Calories Burned: 170

Muscle Group: Back

Reps: 12

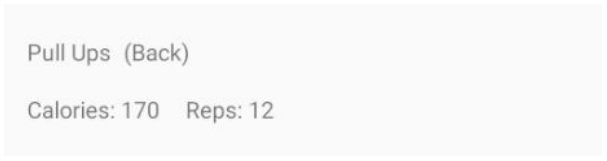
SAVE

Editing this data changes it in the exercises database and is reflected in the user's recyclerview:


Pull Ups (Back)

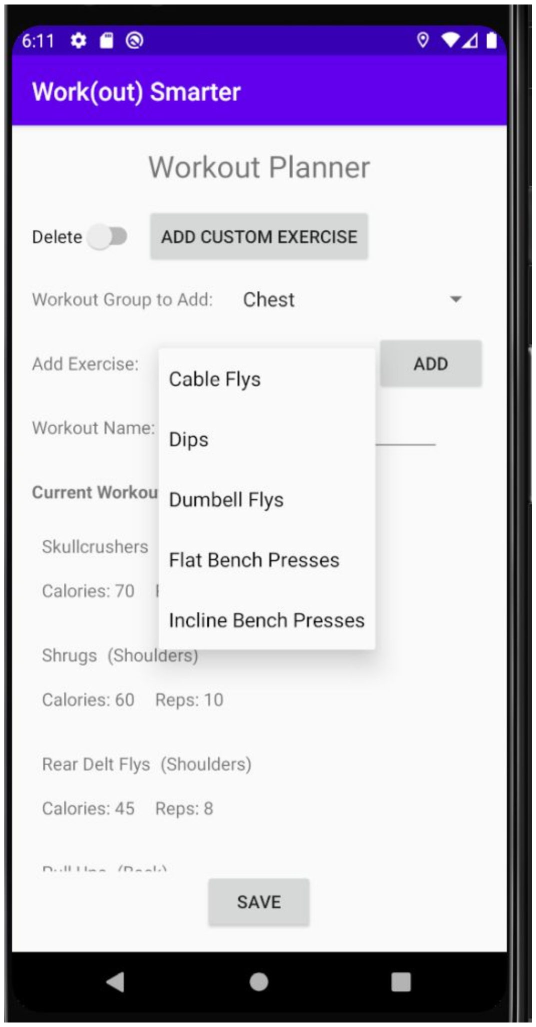
Calories: 150 Reps: 10

=>

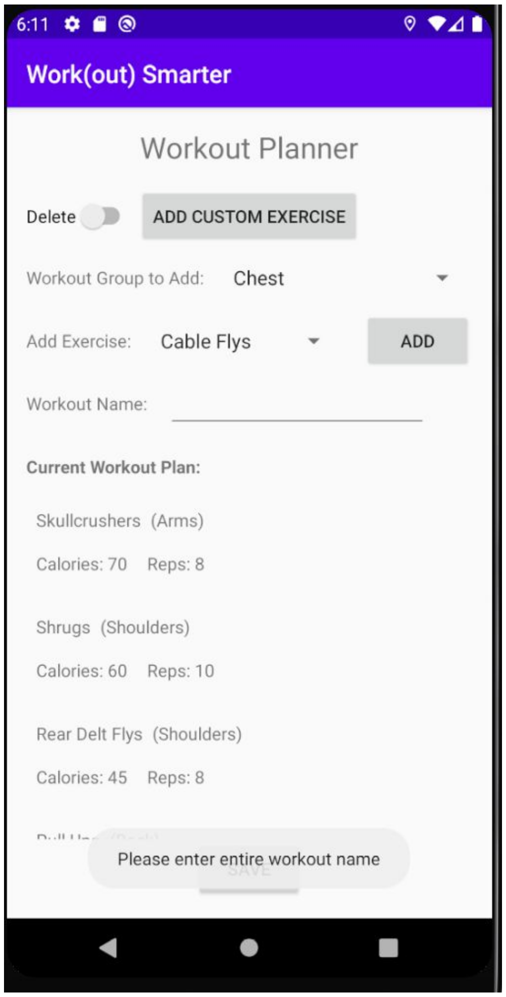

Pull Ups (Back)

Calories: 170 Reps: 12

If the user would like to permanently delete an item from the Exercise database, the user can select an item, say the Diamond Pushup they just created. Clicking the DELETE EXERCISE FROM DB button then permanently deletes it from the database and the user cannot select it as an exercise anymore (Note how Diamond Pushup is now missing from the Chest muscle group exercises because the user just clicked the DELETE EXERCISE FROM DB button):



When the user is ready to add a workout to the workout database, they can click save. However, if they don't add at least one exercise or provide a sufficient workout name, a Toast like the following pops up and prevents the save from occurring:



6:11

Work(out) Smarter

Workout Planner

Delete ADD CUSTOM EXERCISE

Workout Group to Add: Chest

Add Exercise: Cable Flys ADD

Workout Name:

Current Workout Plan:

Skullcrushers (Arms)

Calories: 70 Reps: 8

Shrugs (Shoulders)

Calories: 60 Reps: 10

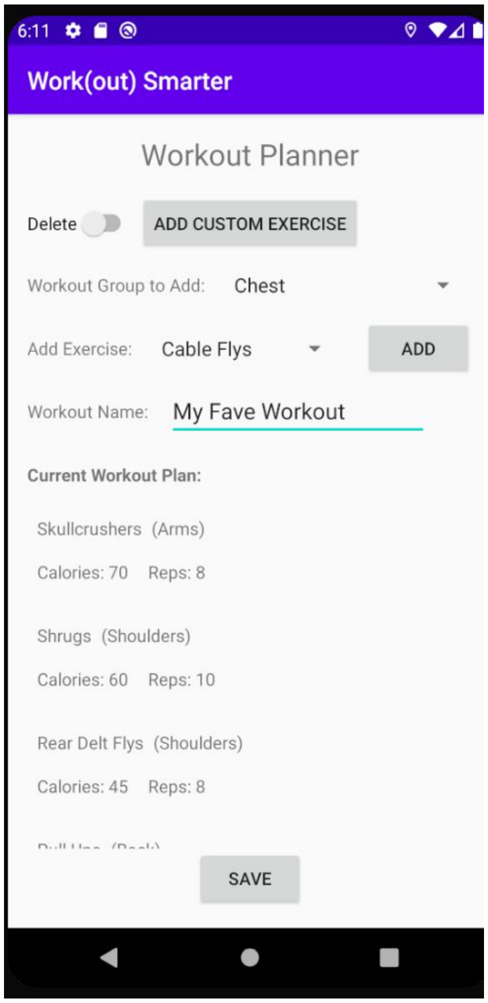
Rear Delt Flys (Shoulders)

Calories: 45 Reps: 8

Please enter entire workout name

SAVE

Let’s say the user enters the Workout Name “My Fave Workout” and presses save. The user can then view the workout in the main activity view:


6:11

Work(out) Smarter

Workout Planner

Delete ADD CUSTOM EXERCISE

Workout Group to Add: Chest

Add Exercise: Cable Flys ADD

Workout Name: My Fave Workout

Current Workout Plan:

Skullcrushers (Arms)

Calories: 70 Reps: 8

Shrugs (Shoulders)

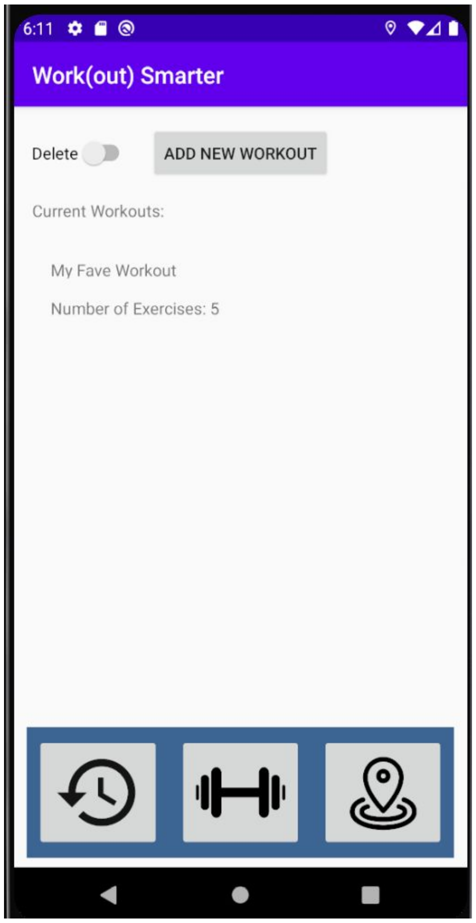
Calories: 60 Reps: 10

Rear Delt Flys (Shoulders)

Calories: 45 Reps: 8

SAVE

=>


6:11

Work(out) Smarter

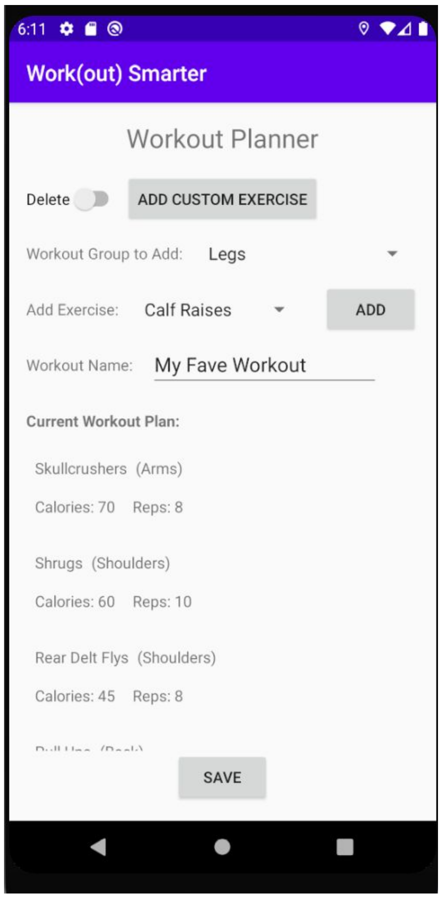
Delete ADD NEW WORKOUT

Current Workouts:

My Fave Workout

Number of Exercises: 5

If the user would like to edit a workout, they can then click on the “My Fave Workout” list item in the Recycler View to edit and update it, like so:



Workout Planner

Delete ADD CUSTOM EXERCISE

Workout Group to Add: Legs

Add Exercise: Calf Raises ADD

Workout Name: My Fave Workout

Current Workout Plan:

Skullcrushers (Arms)

Calories: 70 Reps: 8

Shrugs (Shoulders)

Calories: 60 Reps: 10

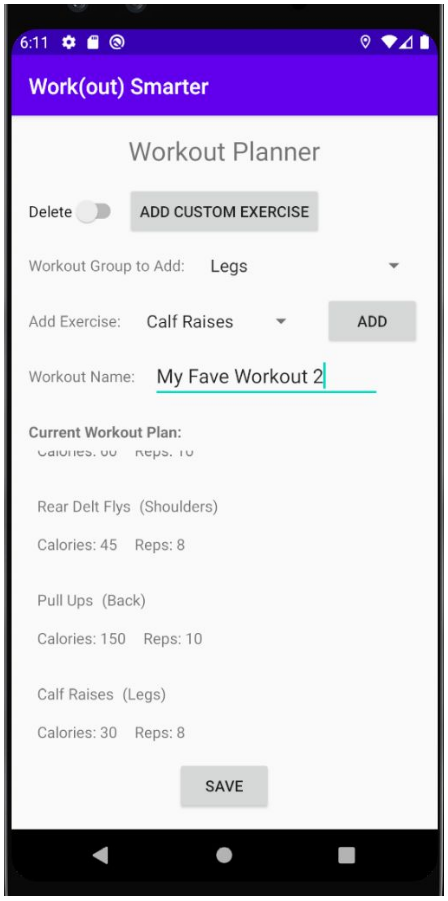
Rear Delt Flys (Shoulders)

Calories: 45 Reps: 8

Pull Ups (Back)

SAVE

=>



Workout Planner

Delete ADD CUSTOM EXERCISE

Workout Group to Add: Legs

Add Exercise: Calf Raises ADD

Workout Name: My Fave Workout 2

Current Workout Plan:

Calories: 60 Reps: 10

Rear Delt Flys (Shoulders)

Calories: 45 Reps: 8

Pull Ups (Back)

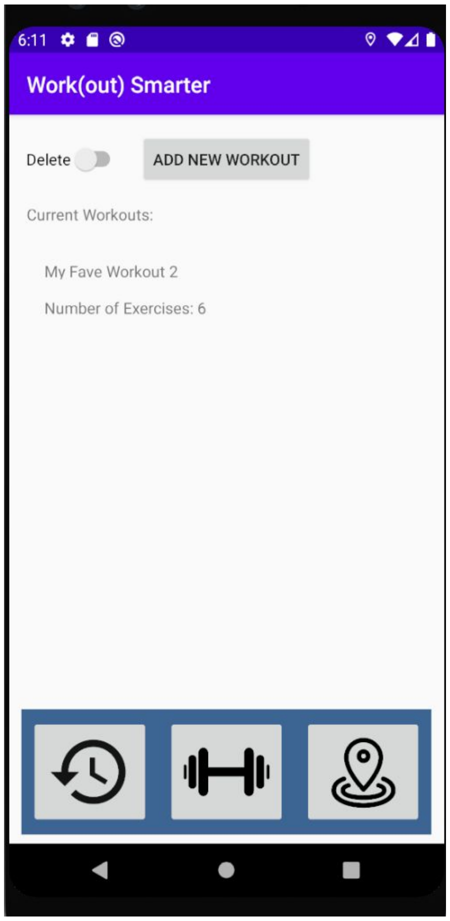
Calories: 150 Reps: 10

Calf Raises (Legs)

Calories: 30 Reps: 8

SAVE

This leads to an updated Recycler View in the Main tab:



6:11

Work(out) Smarter

Delete

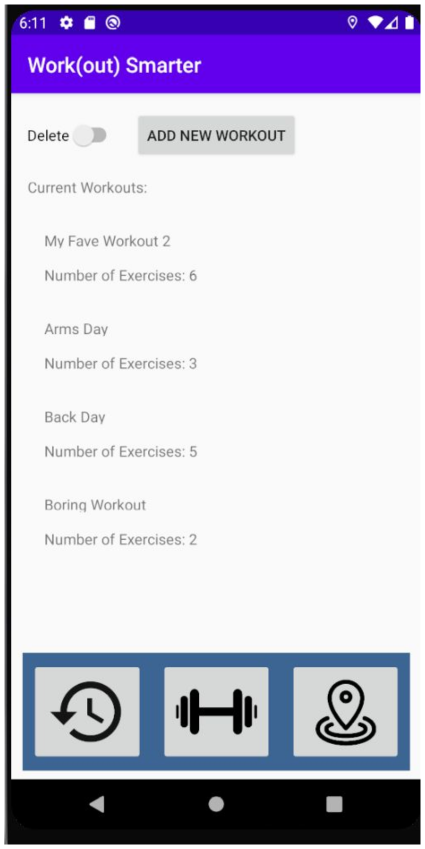
ADD NEW WORKOUT

Current Workouts:

My Fave Workout 2

Number of Exercises: 6

Let’s say the user uses this process to add three more workouts, called “Arms Day,” “Boring Workout,” and “Back Day”:



Current Workouts:

My Fave Workout 2

Number of Exercises: 6

Arms Day

Number of Exercises: 3

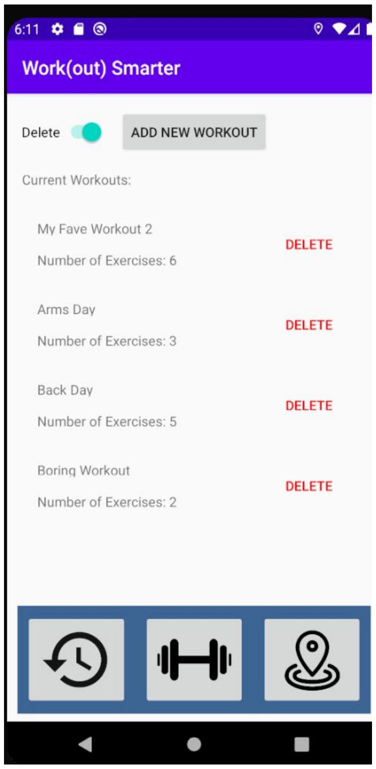
Back Day

Number of Exercises: 5

Boring Workout

Number of Exercises: 2

If the user wishes to delete “Boring Workout” from the Workout database, they can simple toggle the Delete switch and click the DELETE button of the list, like so:



6:11

Work(out) Smarter

Delete

ADD NEW WORKOUT

Current Workouts:

My Fave Workout 2

Number of Exercises: 6

DELETE

Arms Day

Number of Exercises: 3

DELETE

Back Day

Number of Exercises: 5

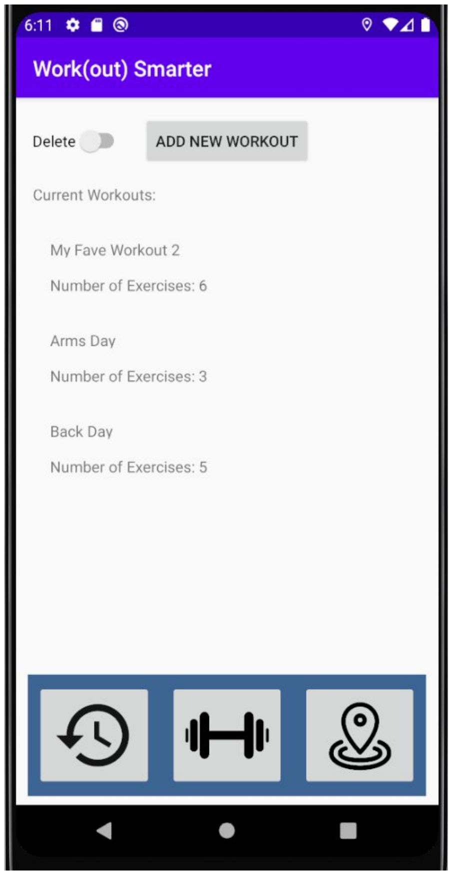
DELETE

Boring Workout

Number of Exercises: 2

DELETE

After the user deletes the workout, the main tab looks like:



6:11

Work(out) Smarter

Current Workouts:

My Fave Workout 2

Number of Exercises: 6

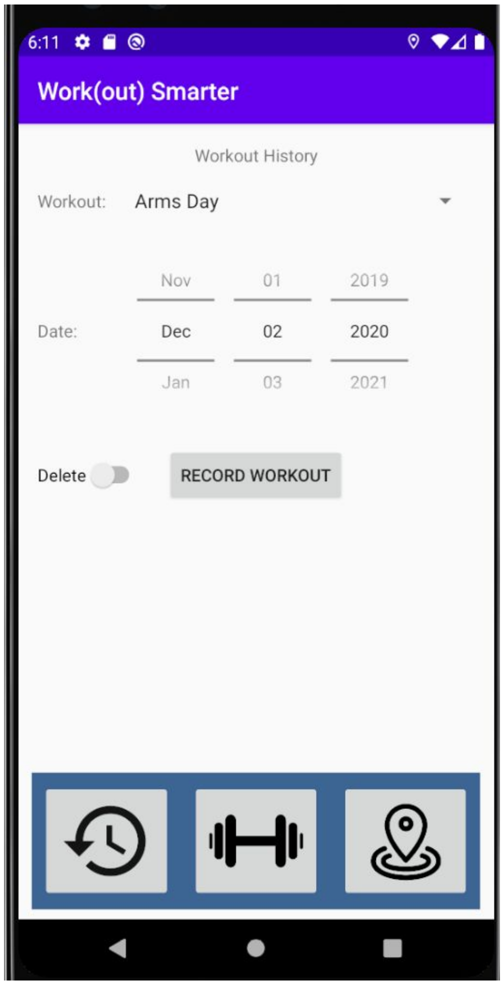
Arms Day

Number of Exercises: 3

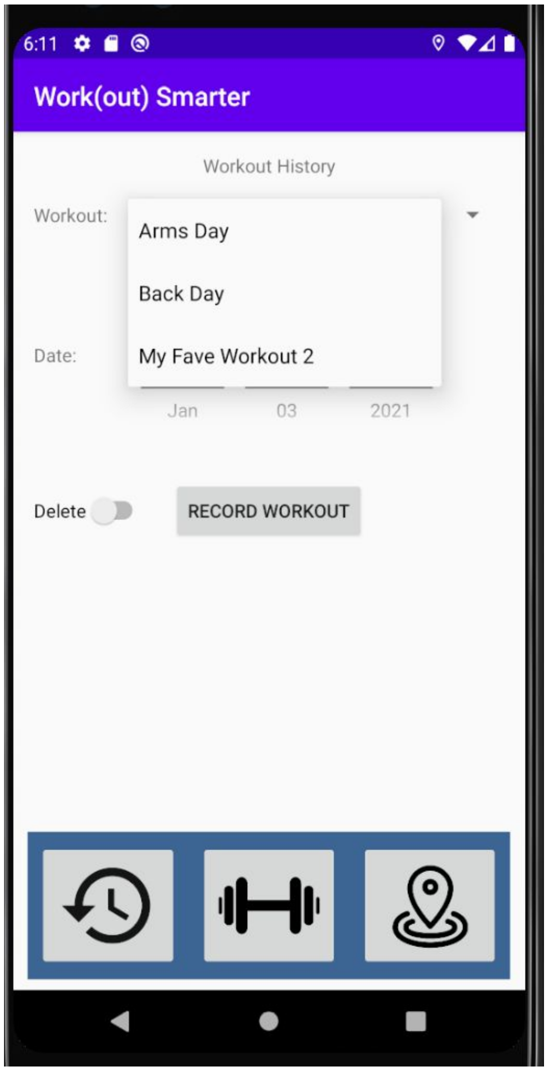
Back Day

Number of Exercises: 5

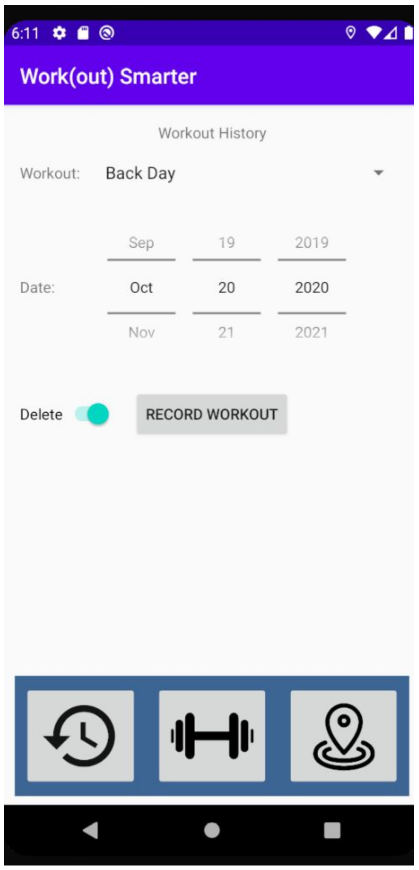
If the user wished to view their history of workouts, they can click on the history tab on the left of the navigation bar, like so:



Here, the user can record workouts they have performed in the past by selecting a workout in the workout group that shows them all their current workouts in the group, like so:



If the user selects Back Day and sets the DatePicker to October 20th 2020, they can then record the workout by pressing the Record Workout button and view it in the Recycler View:

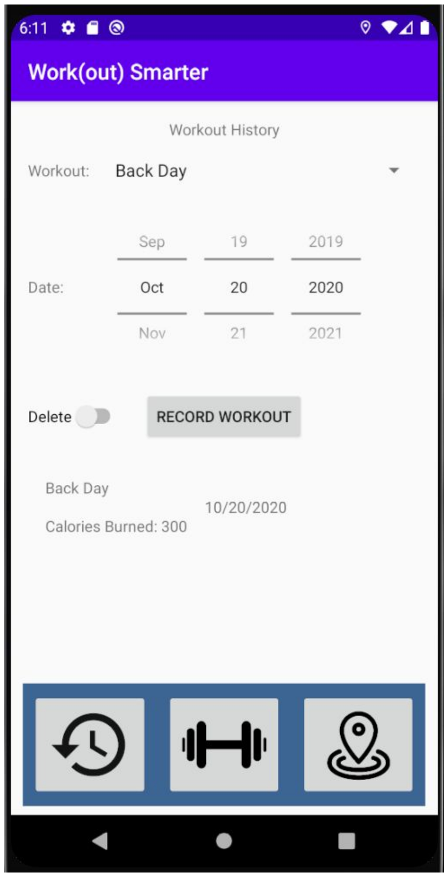


Workout: Back Day

Date: Oct 20 2020

Delete: on

=>



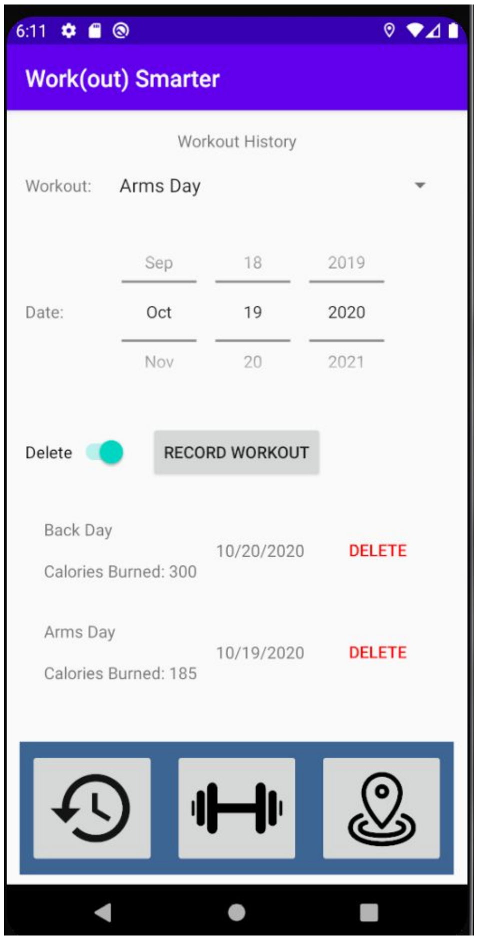
Workout: Back Day

Date: Oct 20 2020

Delete: off

Back Day
10/20/2020
Calories Burned: 300

Each workout history informs the user of how many calories they burned and on what date. If the user adds more workout dates, they can scroll through their workout history as recorded in the PastWorkout database. If the user would like to delete a database, they can toggle the Delete Switch and then click the DELETE button to remove it from the SQLite database, like so for deleting an Arms Day workout here:


6:11

Work(out) Smarter

Workout History

Workout: Arms Day

Date: Oct 19 2020

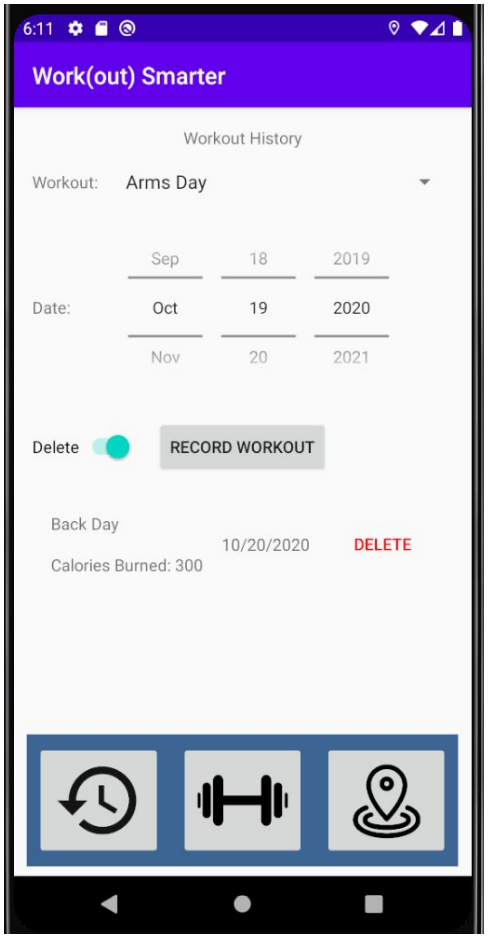
Delete

RECORD WORKOUT

Back Day
Calories Burned: 300
10/20/2020
DELETE

Arms Day
Calories Burned: 185
10/19/2020
DELETE

=>


6:11

Work(out) Smarter

Workout History

Workout: Arms Day

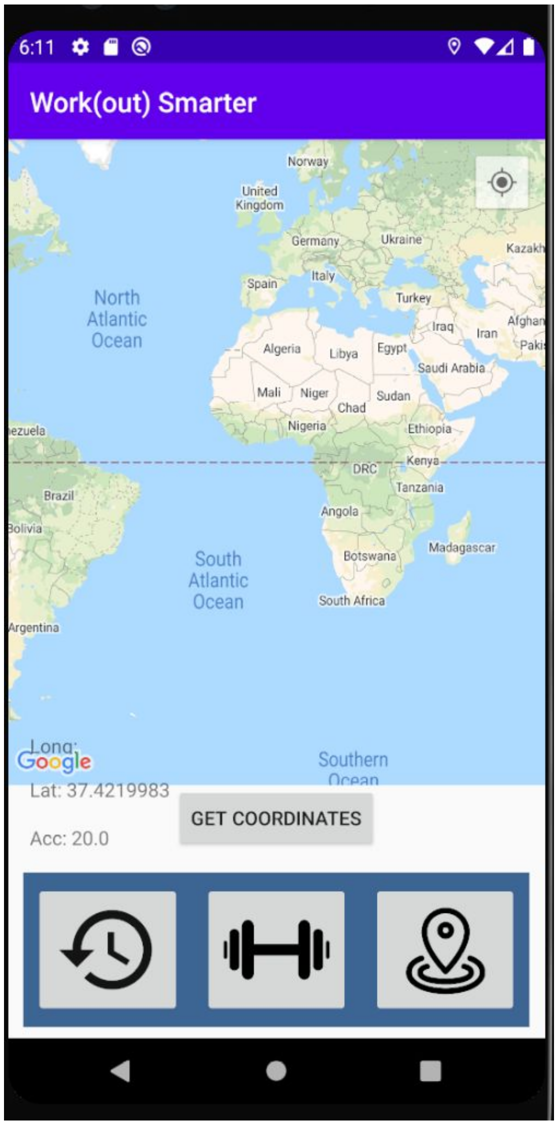
Date: Oct 19 2020

Delete

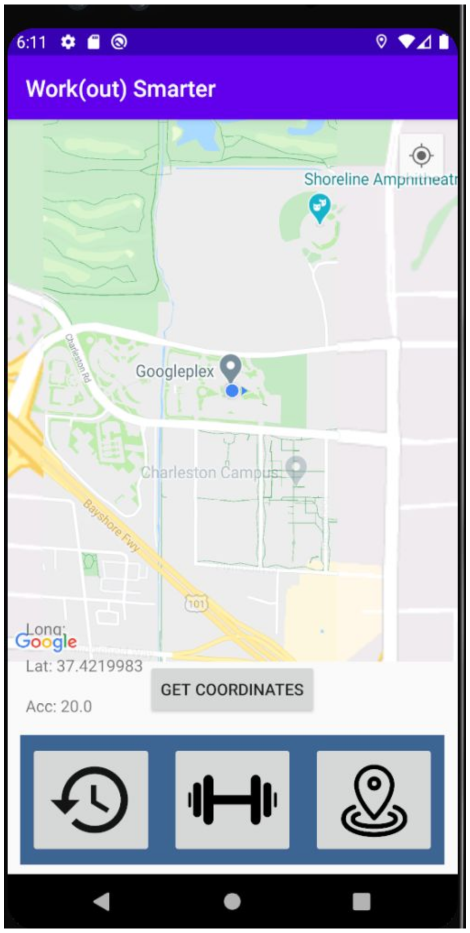
RECORD WORKOUT

Back Day
Calories Burned: 300
10/20/2020
DELETE

Lastly, the user can use the Maps tab to view gyms near them. Clicking on the Maps tab leads to the following Activity:



The user can then use the Google Maps API to zoom to their location and get their coordinates so that they can look for gyms near them. It should be



It should be noted that the original intent was to use the Google Places API to allow the user to automatically search for gyms within a nearby radius of them. The API, it turns out, begins to cost after a certain amount of calls, so I was not able to replicate the behavior in this report.

Conclusions and Analysis

The design, implementation, and testing of the many edge cases of the Work(out) Smarter application was a laborious, educational, and time-intensive process. Not only was I able to extend all of the lesson learned from class, including, but not limited to app design, Gradle, Manifest files, XML files, Android debugging, Activities, Intents, SQLite databases, Adapters, ArrayLists, Location Sensors, Maps, and using APIs, but I was also able to learn and adapt many more lessons and methods of improving applications such as using Spinners and other Android APIs to complete the needs of the Work(out) Smarter application.

This illustrates just how many mobile device programming skills I was able to learn throughout the time of this course. Not only was I able to take and extend the concepts I have already learned in class, but I was able to use the basics of the concepts to implement new features and programs that I could read and understand from Google’s Android programming documentation. Another lesson I learned was just how time-consuming mobile development can be. Sometimes, just to get a simple Spinner working with my SQLite database would take a few days alone. This time-intensive process for a seemingly simple task has certainly allowed me to gain much respect for the development process and all the knowledge, patience, and skills that developers must have in order to create more complex and more useful applications.

I am confident that the application I began working on could continue to be developed and worked on for a real company or business. I am proud to be able to say that I’ve created an application that I could genuinely use for keeping up with my personal fitness if I continued to work on the application, and I am very grateful to the CSE department at the University of Louisville for allowing me to learn so much and be able to implement in a hands-on project based approach the lessons learned in class.

All in all, I can confidently say that this final project shows how I have learned all the desired course objectives of the course per the syllabus, namely: understanding Android/iOS Navigation and Interface Design, understanding Tables/Lists in iOS and Android, understanding and using Maps and Location in iOS and Android, understanding accessing to hardware and sensors in iOS and Android, and, most definitely, implementing data persistence in iOS and Android with SQLite.

To conclude, I will continue learning mobile device programming and will extend the many concepts I learned in this class to other aspects of programming, software, and computer science in general on this life long journey of learning computer science and engineering.