CptS 479 Final Project: Interval Runner Use & Demo Guide

Reid Reininger

Summary

The app will help runners plan, complete, and record workouts. The primary focus is on interval training workouts, where users must complete an activity in a specified interval. I have yet to find an app that offers the customization I want for these intervals. For example, I may want an interval to expire after a period of time, and the next interval to expire after walking or running a certain distance or reaching a location.

Proposal Requirement Deviations

See appendix for full project proposal.

Requirement 8 states “Interval type is selected by a segment bar.” However, the number of interval types and their label sizes made this impractical. The UISegmentBar is replaced with a UIPickerView.

Requirement 13 states “Tapping an interval cell segues to a view to edit the interval, similar to the view for interval creation.” A more practical interface evolved throughout the development process where multiple cells are edited and added to the workout at the same time. Interval cells cannot be tapped in the WorkoutViewController to edit them. Instead, they should be managed through the AddIntervalViewController. The skill of segueing to a view from tapping a UITableViewCell and passing information is still demonstrated through requirement 17.

Demo Guide

The following is a suggested series of steps to demonstrate the main app functionality using the “City Run” location for the simulator. The intervals were planned to expire in the shortest possible time given the running simulation. See screenshots below.

1. Create a new workout
    a. In the WorkoutTableView tap the “Workout” button in the navigation bar at the bottom of the view. See the leftmost screenshot below.
    b. From the WorkoutTableView tap the “+” in the navigation bar.
    c. In the text field directly below the navigation bar enter “Run/Walk”.
2. Add intervals to the workout
    a. Tap the “+” in the navigation bar.
    b. Add a timer interval.
        i. In the “Label” text field type “Run”.
        ii. In the “Interval Type” picker view select “Timer”.
        iii. In the other picker view select 0 hours and 1 minute.
        iv. Tap the blue “Add Interval” button.
    c. Add a location interval.
        i. Type “Reach” in the “Label” text field.

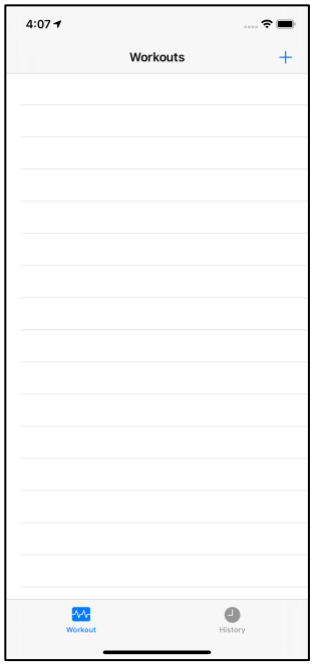
ii. In the “Interval Type” picker view select “Location”.
iii. In the search bar search “Middle school”.
iv. Select the row that says “Sam H. Lawson Middle School”
v. Tap the blue “Add Interval button.

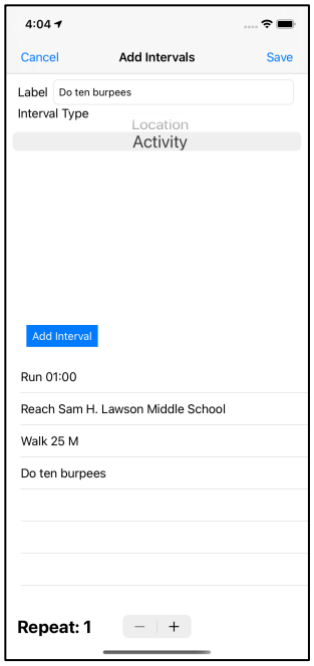
d. Add a distance interval.
i. Type “Walk” in the “Label” text field.
ii. In the “Interval Type” picker view select “Distance”.
iii. In the “Distance” text field type “25”
iv. Tap the blue “Add Interval” button.

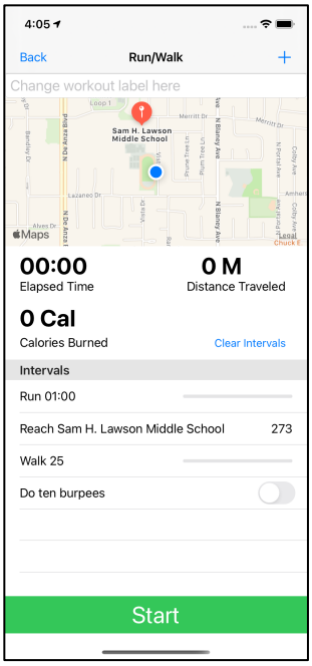
e. Add an activity interval.
i. Type “Do ten burpees” in the “Label” text field.
ii. In the “Interval Type” picker view select “Activity”.
iii. Tap the blue “Add Interval” button.

f. Ensure “Repeat” counter is set to 1. Should look like middle screenshot below.
g. Tap “Save” in the navigation bar.

3. Change the simulator location to “City Run”. Location should be the Apple Infinite Loop.
4. Start the workout. Should look like rightmost screenshot below.
   a. Tap the green “Start” button.
   b. Once the first three intervals have expired, tap the switch on the “Do ten burpees” interval to complete the interval. This will also complete the workout.
5. Save the workout.
   a. Tap “back” in the navigation bar.
   b. Tap “Save” in the alert.
6. View the workout history.
   a. Tap “History” in the navigation at the bottom of the view.
   b. Tap the “Run/Walk” row.



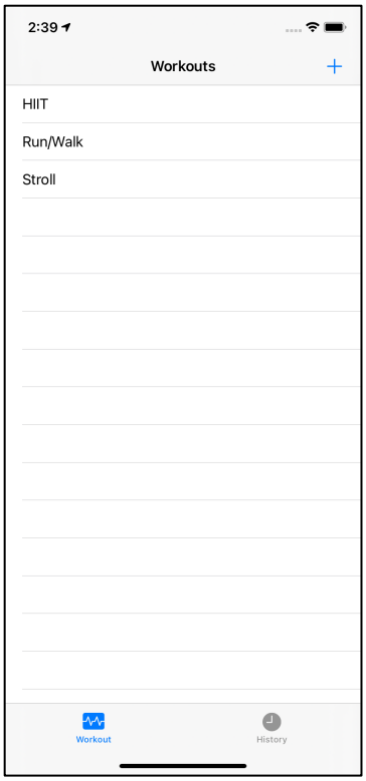


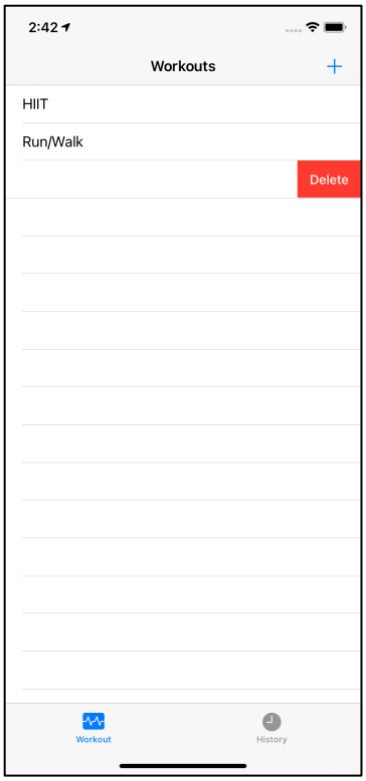


Use Documentation

WorkoutTableView

Provides a table showing all created workouts. These workouts are persisted in CoreData. Workouts can be deleted by swiping on them. New workouts are added by tapping the “+” in the navigation bar. Workouts can be edited or started by tapping on them. The TabBarController at the bottom allows for navigating between the WorkoutTableView and HistoryTableView.





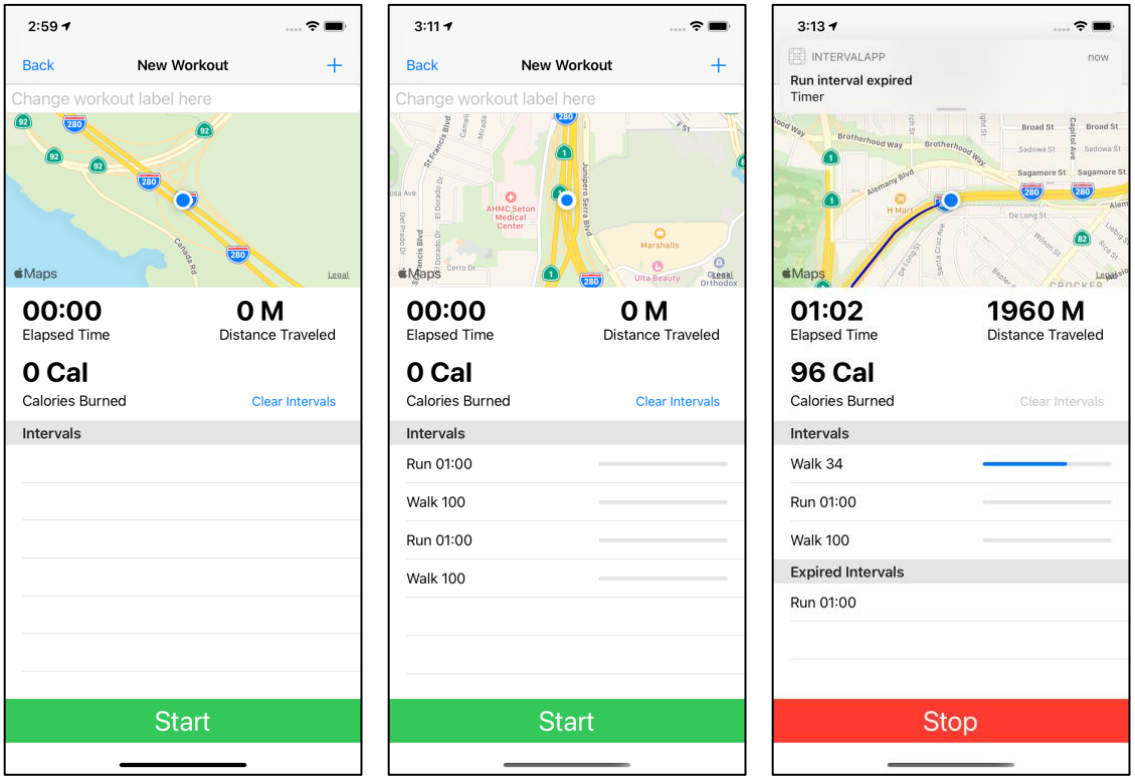
WorkoutView

The WorkoutView allows users to edit and start workouts. It displays workout information such as the workout label in the navigation bar, the user's location, a polyline of the user's location while the workout is active, the elapsed time, distance travelled, calories burned, and workout intervals.

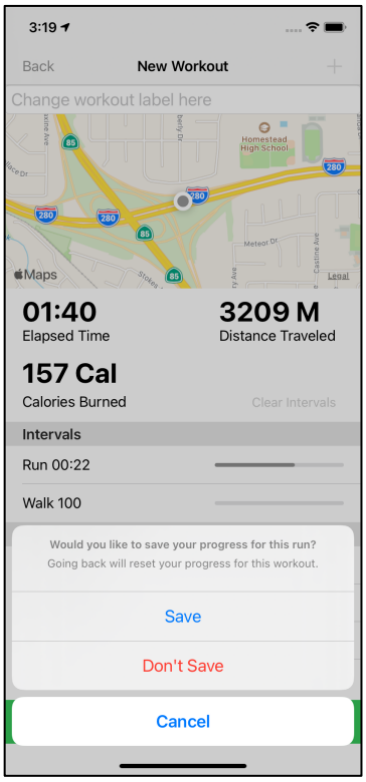
The workout label can be changed by entering a new label in the UITextField labeled "Change workout label here". This change is immediately reflected in the navigation bar. A user can add intervals to the workout by tapping "+" in the navigation bar. A user can start the workout by tapping the green start button at the bottom of the view. This turns to a red stop button when the workout is in progress. A notification appears when an interval or the entire workout expires. Intervals cannot be edited or added once a workout has begun. The workout must end before any changed to workout can be made, other than the label.

Expired intervals are shown under the "Expired Intervals" header in the TableView.

All intervals can be cleared by tapping the "Clear Intervals" button. This will present an alert to confirm the action.

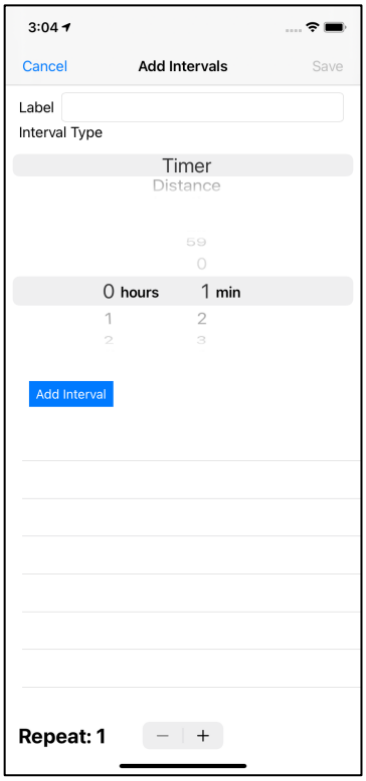


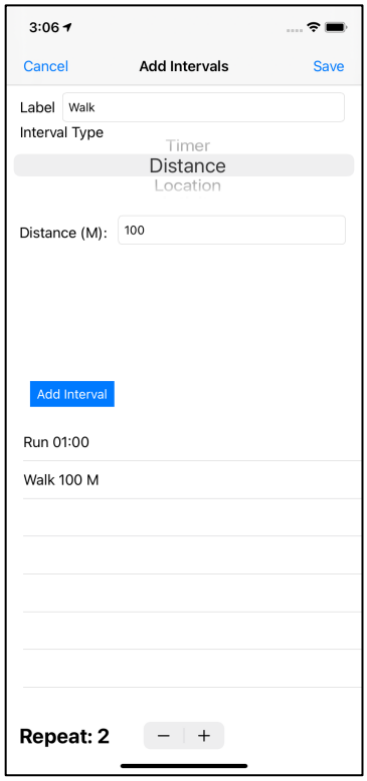
At any time the workout is paused (after having been started), the user can tap the back button in the navigation bar. This presents an alert asking to save or don't save the workout in the workout history before navigating back to the WorkoutTableView.



AddIntervalViewController

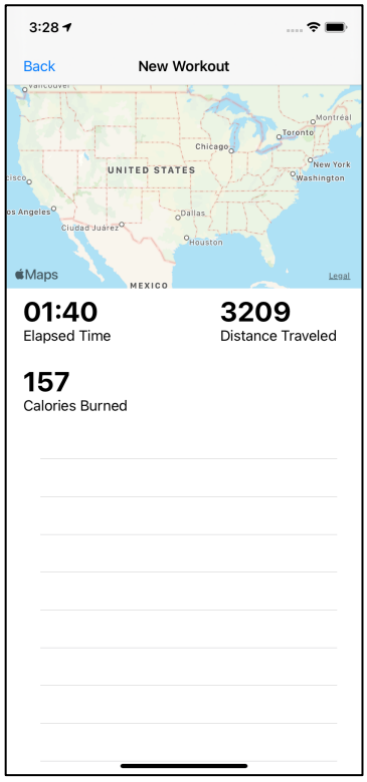
This view allows the user to add intervals to the workout. A label is given to the interval in the "Label" UITextField. The interval type is selected using the UIPickerView labeled "Interval Type". The UI changes to display to appropriate data entry elements for the selected interval type. Once the data has been entered for the interval, users tap the blue "Add Interval" button to append the interval to the TableView below. Intervals can be removed from the TableView by swiping. The intervals in the TableView will be appended to the TableView in the WorkoutView after tapping "Save" in the navigation bar. Alternatively, press "Cancel" in the navigation bar to not add any intervals. To append the intervals in the TableView multiple times, increment the repeat counter at the bottom of the View.





HistoryTableView & HistoryDetailView

The HistoryTableView displays all recorded workout histories and is persisted in CoreData. Each row has the label of the workout and the time the workout was saved. Swiping a row deletes the history. Tapping a row segues to the HistoryDetailView which provides more information about the workout such as the time, distance, and calories. There is a MKMapView in the HistoryDetailView however the location of the workout is not recorded. This is saved for future work. Note saving the workout map information was not a proposal requirement. Requirement 18 states “[persisted] data includes total distance, time, and calories burned.”



Appendix

Reid Reininger
CPTS 479 Project Proposal
April 7, 2021

*Additions in second submission highlighted

Interval Runner App

Summary

The app will help runners plan, complete, and record workouts. The primary focus is on interval training workouts, where users must complete an activity in a specified interval. I have yet to find an app that offers the customization I want for these intervals. For example, I may want an interval to expire after a period of time, and the next interval to expire after walking or running a certain distance or reaching a location.

Details

1) The main view is embedded in a navigation controller.
2) The main view has labels displaying the total distance, time, and calories burned for the workout.
3) The main view has a mapKit showing the users location. The mapKit will have pins showing any locations the user has selected for an interval expiration. It will also trace the route of the user while the workout is in progress.
    a. This will be tested using the run location option in the simulator.
4) The main view has a table showing the intervals for the workout in order.
5) Intervals can be deleted by swiping.
6) Intervals can be added by tapping the + icon, which segues to a new view.
7) Intervals can be of type timer, distance, location, or activity
8) Interval type is selected by a segment bar.
9) The UI elements in the interval creation view are dependent on the selection in the segment bar.
10) A new interval is saved or canceled. Saving it adds it to the interval table in the main view. Cancel does nothing and returns to the main view. The save and cancel buttons are in the navigation bar.
11) Timer, distance, and location intervals have a progress bar showing their progress, and display relevant information in their label.
12) Activity intervals have a switch which users tap when the activity for the interval is completed.
13) Tapping an interval cell segues to a view to edit the interval, similar to the view for interval creation.
14) There is a button that says start when the workout is paused, and stop when the workout is in progress.
15) When an interval expires, the user gets a notification. There is some change in the UI to show the interval has expired.
16) The time or distance in the interval cells are updated as the user makes progress.

17) In the main view there will be a button to segue to a view with a history of workouts. They will be displayed in tabular format and cells can be tapped for more detail about the workout in a different view. Data includes total distance, time, and calories burned.
18) The workout data, both current and old, will be persisted using Core Data.

Required Skills

- Using segues
- Using dynamic tableViews with various prototype cells
- Accessing Sensor data
- Sending notifications
- Using and displaying info on a mapView
- Using a timer
- Updating UI elements
- Persisting data with Core Data
- Tracing routes and placing pins on a mapKit

Similar Apps

RunnerUp
https://github.com/jonasoreland/runnerup
Features:

- See detailed stats around your pace, distance and time
- Get stats and progress with built-in highly configurable audio cues
- Run free runs with target pace or target heart rate zone
- Easily configure and run effective interval workouts modeled after Garmin

My app will allow for different interval customizations than allowed in the RunnerUp app. It will also have a simpler interface and not offer as many features.

Intrvl
https://github.com/sirJiggles/intrvl
Features:

- Pause time
- Resting time
- Interval time
- Warm up time
- Stretch time

- Preparation time
- Interval count

My app will offer a different set of intervals than Intrvl. My app will offer intervals that are expired by a distance travelled or location reached, among others. My app also has a very different conceptual model for creating workouts. My app does not have a concept of workout, rest, and pause time like Intrvl. This simplifies things for the user by not imposing these ideas of what a workout is or forcing the user to figure out how the app handles these different concepts. For example, if a user wants a rest period after an interval of intense work, they simply add an interval labeled rest to expire after a certain amount of time. This allows the user to customize their workout using whatever model they want. My app does not provide ideas for workouts, it simply provides the blocks needed to build one, making it much simpler and customizable.