

Comp 2005
Software Engineering
Activity Tracker Application

Project Documentation

Project by - Group 1

Table of Contents

Introduction	3
Goals	3
Design Approach	3
Design Patterns	3
Design Principles	4
System Behavior	4
Use Case Description	5
Create User Profile	5
Import Activity Data	7
Display Activity Statistics	8
Additional Use Cases	8
Logical View	9
High-Level Design (Architecture)	9
Mid-Level Design	10
Use Case View	11
Detailed Class Design	13
Testing	17
Creating Profile	17
Logging in	18
Import Data Activity	18
Display Statistics	19

1.Introduction

This document describes the architecture and design for an Activity Tracker application developed for the course COMP 2005, during the Fall of 2018. This application's purpose is to keep and manipulate data imported from devices regarding running activities.

There is not a single diagram or model that can easily represent a system's architecture and design. Because of that, software architecture and design is often expressed in terms of multiple perspectives. Here the Activity Tracker application is described from x different perspectives:

1.Logical View - Encompasses major components, their attributes and operations. This view also includes relationships between components and their interactions. When doing OO design, class diagram and sequence diagrams are often used to express the logical view.

2. Use Case view - Used to both motivate and validate design activity. Includes the functional objectives of the design and details on the interaction of the high-level components with the necessary behavior to execute a use case.

2.Goals

This application was developed with focus on both efficiency and maintainability, as instructed by the client's (professor) requests. The design priorities for the product are:

- The design should minimize complexity and development effort.
- The design should provide a flexible code which minimizes the efforts for further modifications.

3.Design Approach

Design Patterns

We have applied two design patterns to our software.

- Proxy Pattern: The Controller class acts as a proxy by forwarding the requests and results between the Running, User and GUI classes. This is used so that only one class interacts with the GUI. If any changes are to be made to the GUI or the way we need to display in the future, we only need to change the Controller class, making our system more extensible. The Database Proxy class allows the system to connect to different databases in the future with minimal change.

- Model-View-Controller Pattern: The GUI class acts as our view, Controller class acts as our controller while the rest of the classes act as our Model, segmenting the flow of work.

Design Principles

We have applied several design principles while iterating through our design.

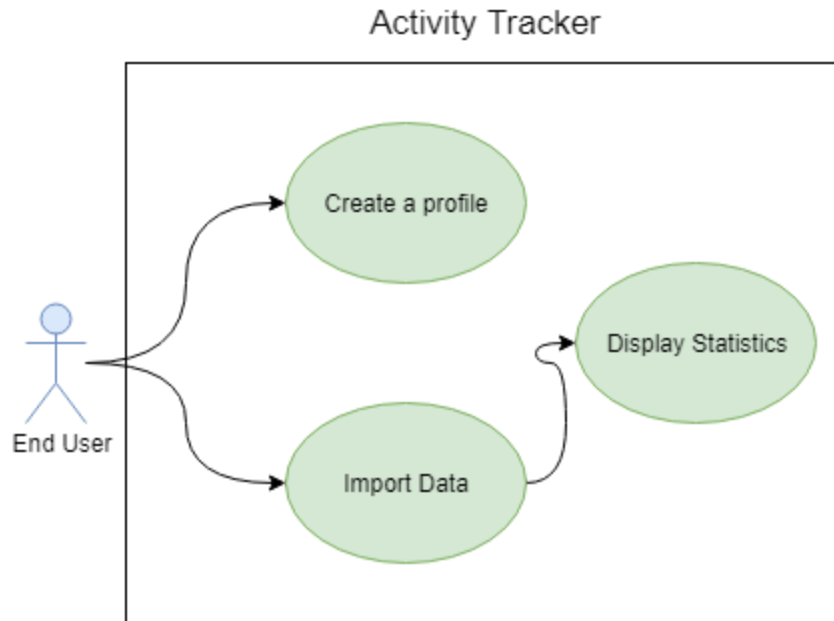
- Delegation: The controller class passes messages back and forth between the Model and the GUI.
- Don't Repeat Yourself: Running class inherits from the Activity class, preventing redundancy.
- Liskov's Substitution Principle: Even though running is an activity, we have made the Activity class abstract because Running does not have the behaviour of every activity.
- Single Responsibility Principle: The Model-View-Controller pattern allows for this principle to be applied. In particular, the model is a composite (in the conceptual sense) object that encompasses all objects used for representation purposes.

Our system is designed to accommodate changes to the type of database used, and allows to add new activities as well as modify display easily.

Risk is minimized through implementation of known attributes, and not implementing ideas that have not been fully articulated. The design is flexible enough to accommodate change.

4. System Behavior

The application reacts to the user's actions and its behavior is better represented through a use case diagram containing the actors and implemented behaviors. The figure below illustrates all the different ways in which a user can act upon the system.



5. Use Case Description

5.1. Create User Profile

Name: Create User Profile

Description: The user would like to create a profile with the intent of registering the user's basic information and set up a username(e-mail) and password. The goal of this process is to authenticate the user and gain access to the application.

Basic Flow:

1. The User provides information like: e-mail, full name, password, date of birth and height.
2. The User confirms the provided information.
3. Profile is created successfully and data is stored in a persistence entity.
4. User concludes profile creation and moves to main screen of the application.

Alternate Flow:

1a. Information provided is not accepted by the System: e-mail format is invalid or existent, date of birth is invalid etc...

1. System blocks the "Register" option and asks The User to correct the information provided.
2. Asks User to login, in the case his/her e-mail is existent.

1b. User cancels the profile creation.

1. System returns to the initial login screen/closes application.

Actors:

Non-Registered User – User does not have a pre-existing profile.

System Entity – receives and verifies the provided information.

Persistence Entity – receives the profile information.

Preconditions:

The user wants to create a new profile (if one doesn't already exists).

Postconditions:

The user's account is created and valid to log in. The main screen of the application is displayed.

5.2. Import Activity Data

Name: Import Activity Data

Description: The user would like to add their activity data to the application from an external source.

Basic Flow:

1. Registered User signals intent to system to import activity data
2. System asks Source Enumerator which sources are available
3. Registered User specifies the source of the data from a list of available sources
4. System instructs Data Parser to import data from Data Source to Persistence Entity

Alternate Flow:

- 2a. There are no available supported sources.
 1. System signals to user that there are no supported sources available
- 2b. User cancels the import
 1. System returns to the previous state prior to signalling their intent to import data.
- 4a. The data source was disconnected before the user selected it
 1. System signals to the user that the selected device is no longer available.
- 4b. Data Parser is not successful
 1. Data Parser signalled that the source is no longer available. Advise user that the source must remain plugged in.
 2. Data Parser signalled that it did not understand the source. Advise user that the source is not comprehensible and to check the condition of the source.

Actors:

Registered User – has preexisting account and is logged in

Persistence Entity – receives the imported information

Source Enumerator – this entity discovers and enumerates import sources

Data Parser – this entity exchanges data from the source to the persistence entity

Data Source – this is a device that contains records of activity

Preconditions:

The user must have an account and they must be logged in.

The persistence entity must exist and be initialized.

Postconditions:

The data must have been imported from the device into the persistence entity.

The user's account must still be valid and logged in.

5.3. Display Activity Statistics

Name: Display activity statistics

Description: The user would like to display activity data as well as statistics regarding this data so they can analyze their activities and gain insight on it. They would also like to be able to sort this data and display it for convenience.

Basic Flow:

1. Registered User signals intent to view activity data
2. System pulls the relevant data from a database
3. Statistics, such as pace are calculated from the data
4. System displays the data and statistics in a neat form

Alternate Flow:

- 2a Database does not contain any records of the activity
 1. System displays a message notifying the user
- 2b System can not connect to the database
 1. System displays an error message
- 4a. User selects a different metric to sort data
 1. System sorts data with respect to that metric
 2. System displays data in the sorted order
- 4b. User sets up a different length of lap (such as 1km, 2km etc)
 1. System recalculates statistics
 2. System displays data

Actors:

Registered User – has preexisting account and is logged in

Persistence Entity – receives the intent to display activity

Data Source - this entity contains records of the activity

Data Parser – this entity exchanges data from the database to the persistence entity

Preconditions:

The user must have an account and they must be logged in.

The persistence entity must exist

Postconditions:

System must have displayed the activity information with the option to set up laps and sort the data differently

5.4. Additional Use Cases

Manage Activity Data

Name: Manage Activity Data

Description: The user would like to manually change recorded information about their activity. They will signal their intent to change recorded information, and the program will give the user a way to add, modify, or delete information manually.

Connect With Other Users

Name: Connect With Other Users

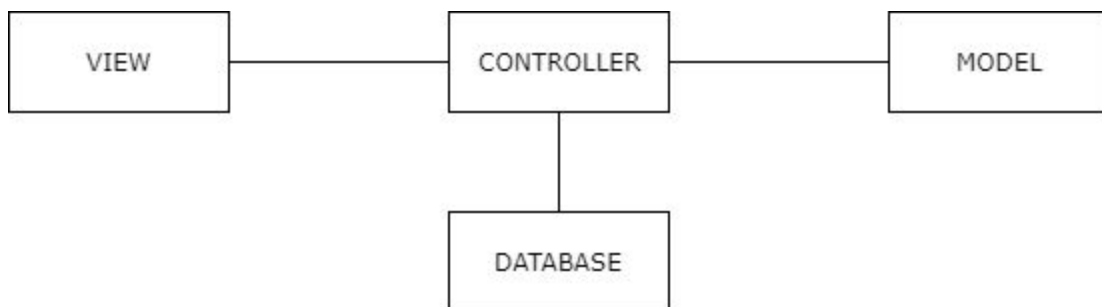
Description: The user would like to declare a relationship with another user of the program. The user will choose another user from the list of known users and formally request an association. The other user will be prompted to accept or reject the association. Once established, the association will enable either user to view the other's activity data.

6.Logical View

The main functional components of the system and their interactions are described in this part. First, the modules of the system are expressed in terms of architecture and then are progressively refined into more detailed elements (components and classes) which encompasses the system.

6.1. High-Level Design (Architecture)

This design illustrates how the main components of the system communicates with one another and how the model-view-controller pattern was a great influence to the project.



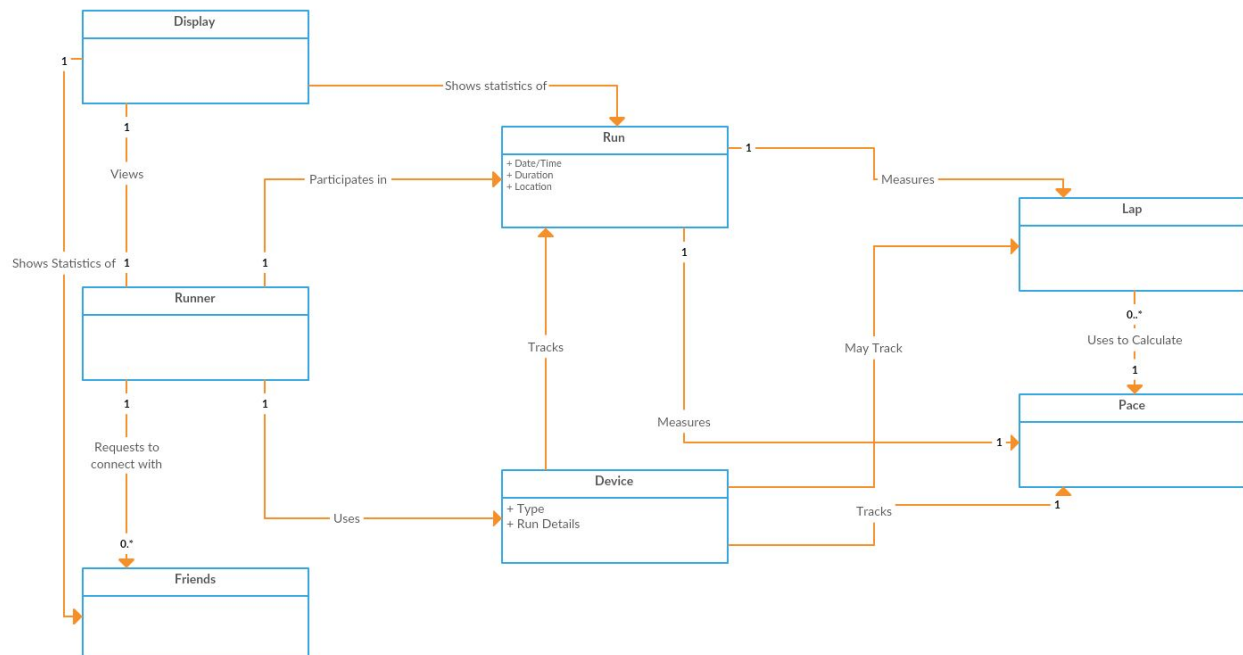
- The **View** module provides the interface that the user interacts with. The changes done here are signalized to the controller so that changes (insertion, removal, queries) can be done to the database or model.

- The **Controller** module responds to the user input and acts upon the data model objects. The controller receives the input, optionally validates (when logging in, for example) and then passes the input to the model.
- The **Database** is the repository of all the data we use in the application. Information regarding the runs and user profile are kept here.
- The **Model** represents an instance of the application data. This is the part where the user's associate run and profile are kept while the application runs.

6.2. Mid-Level Design

The Application Domain Model shown below was designed as part of our early conceptual view of the system that incorporates behavior and data. This abstraction allowed us to reflected over the selected aspects that most mattered to the application and helped us to see and solve some of the initial problems we had with our class design first attempts.

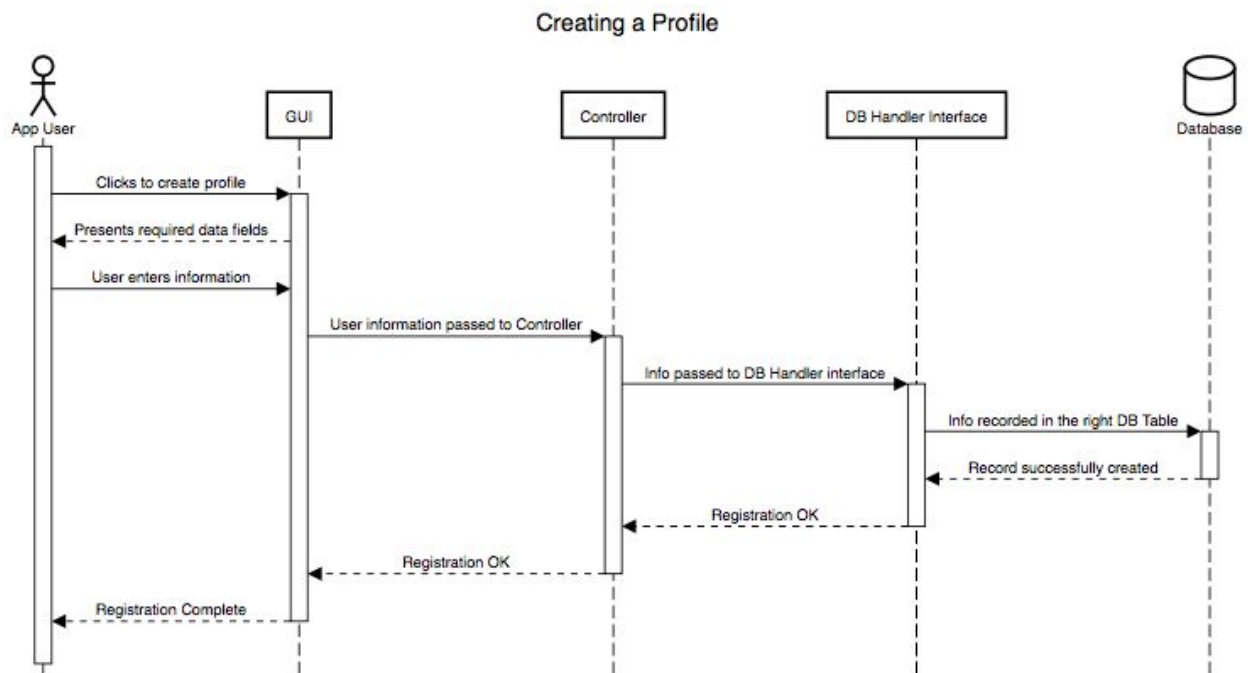
Activity Tracker
Application Domain Model



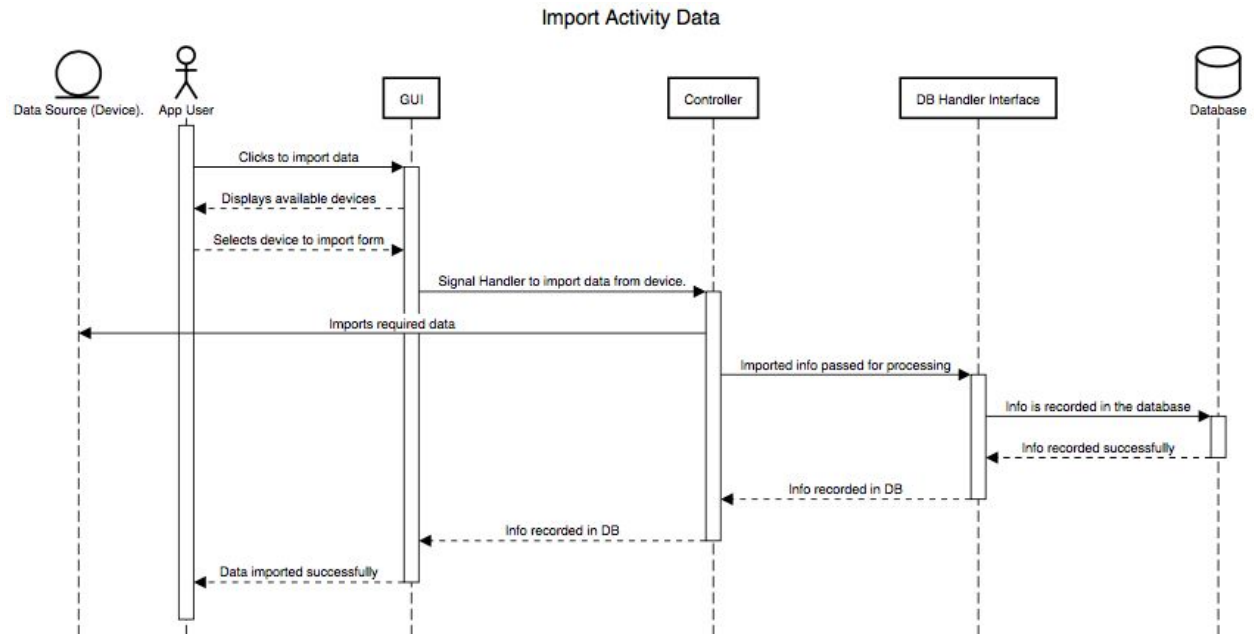
6.2.1. Use Case View

In this stage, the sequence diagrams exemplifies better how the different parts of the application interact and are arranged in a time sequence. The Diagrams below represent the three main use cases that were selected for implementation and described earlier in this document.

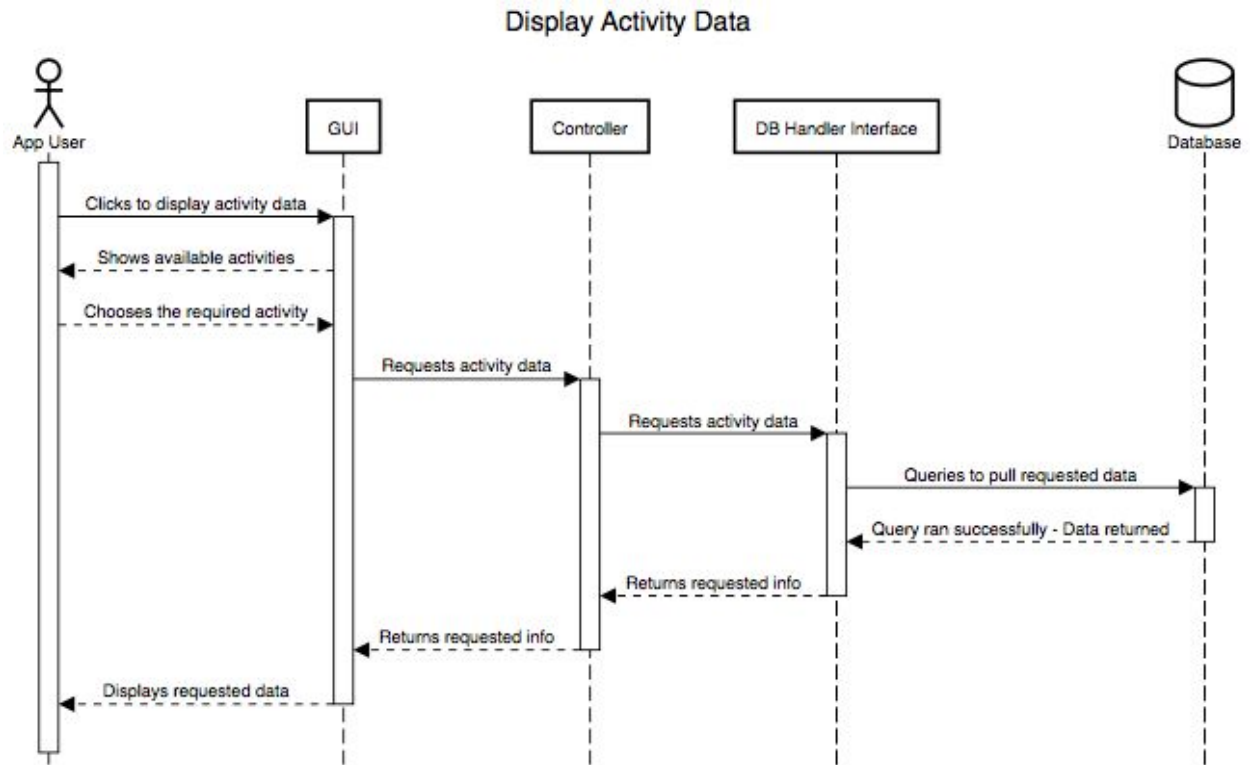
The user sequence diagram below represents the flow of events that take place when creating a profile.



The user sequence diagram below represents the flow of events that take place when importing activity data.

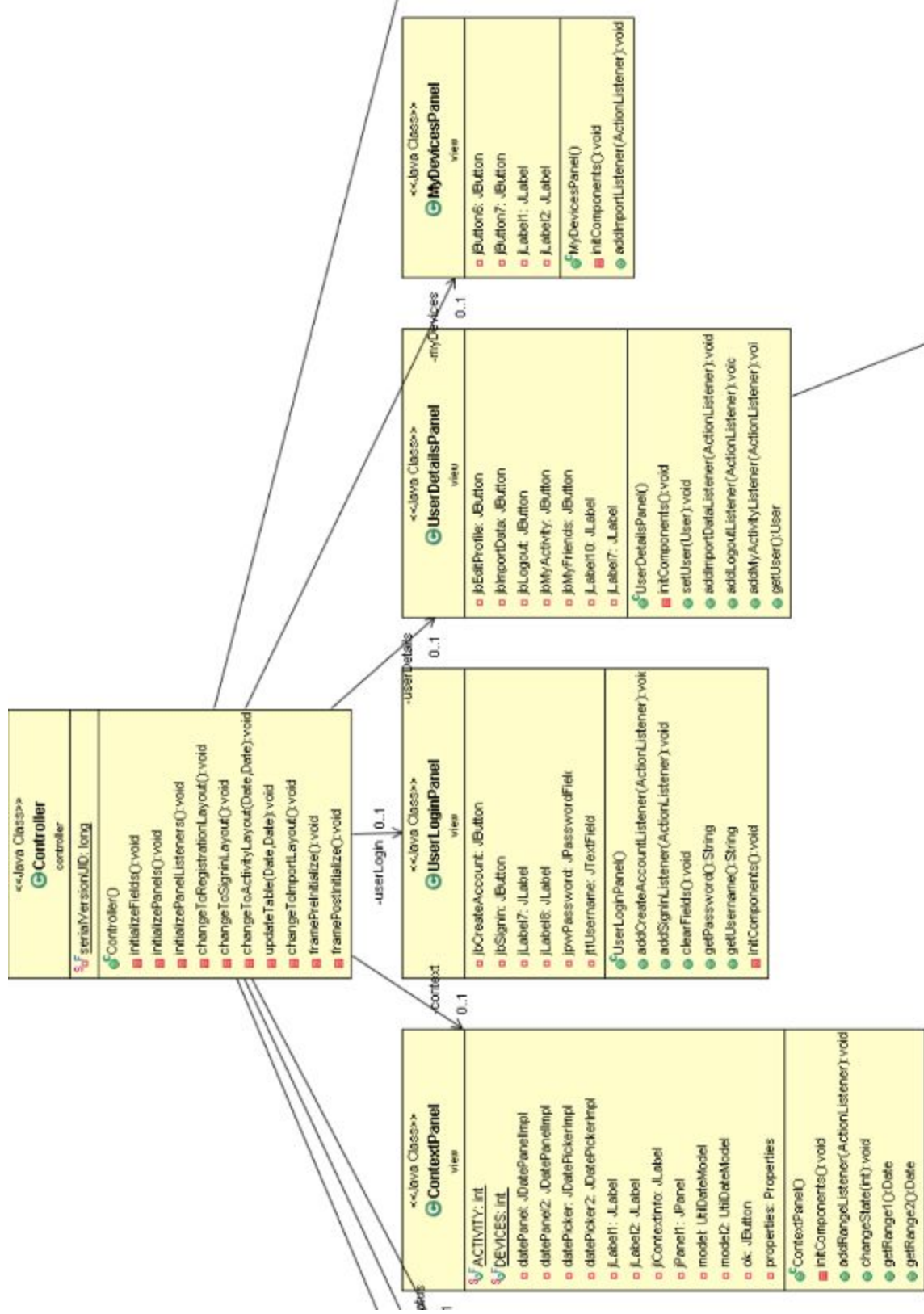


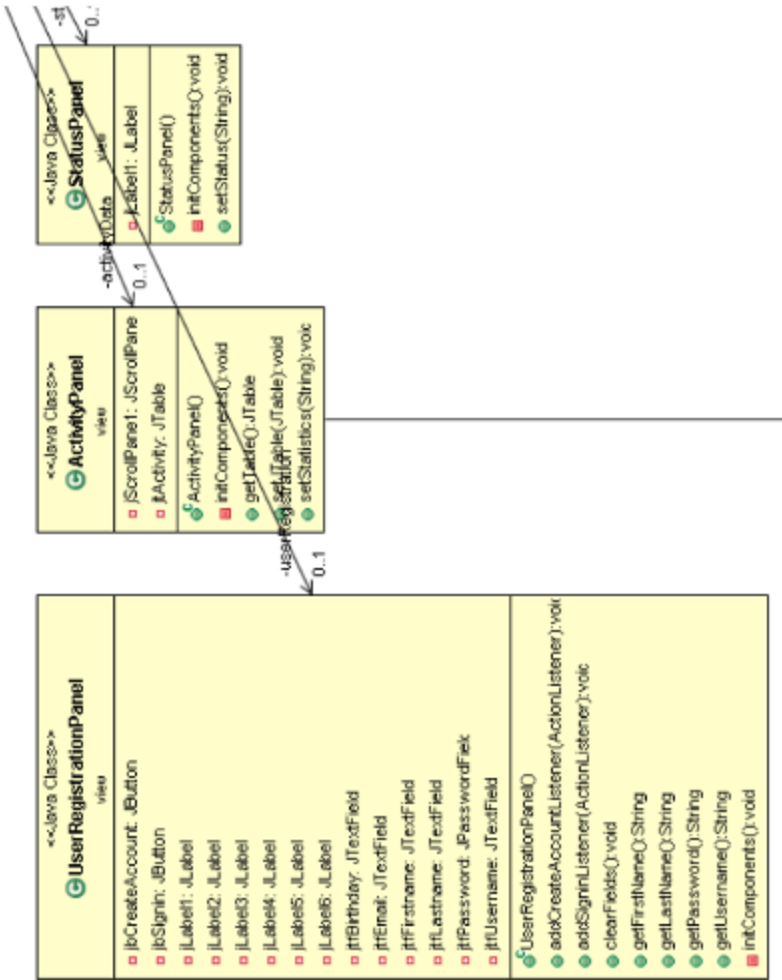
The user sequence diagram below represents the flow of events that take place when displaying activity data.

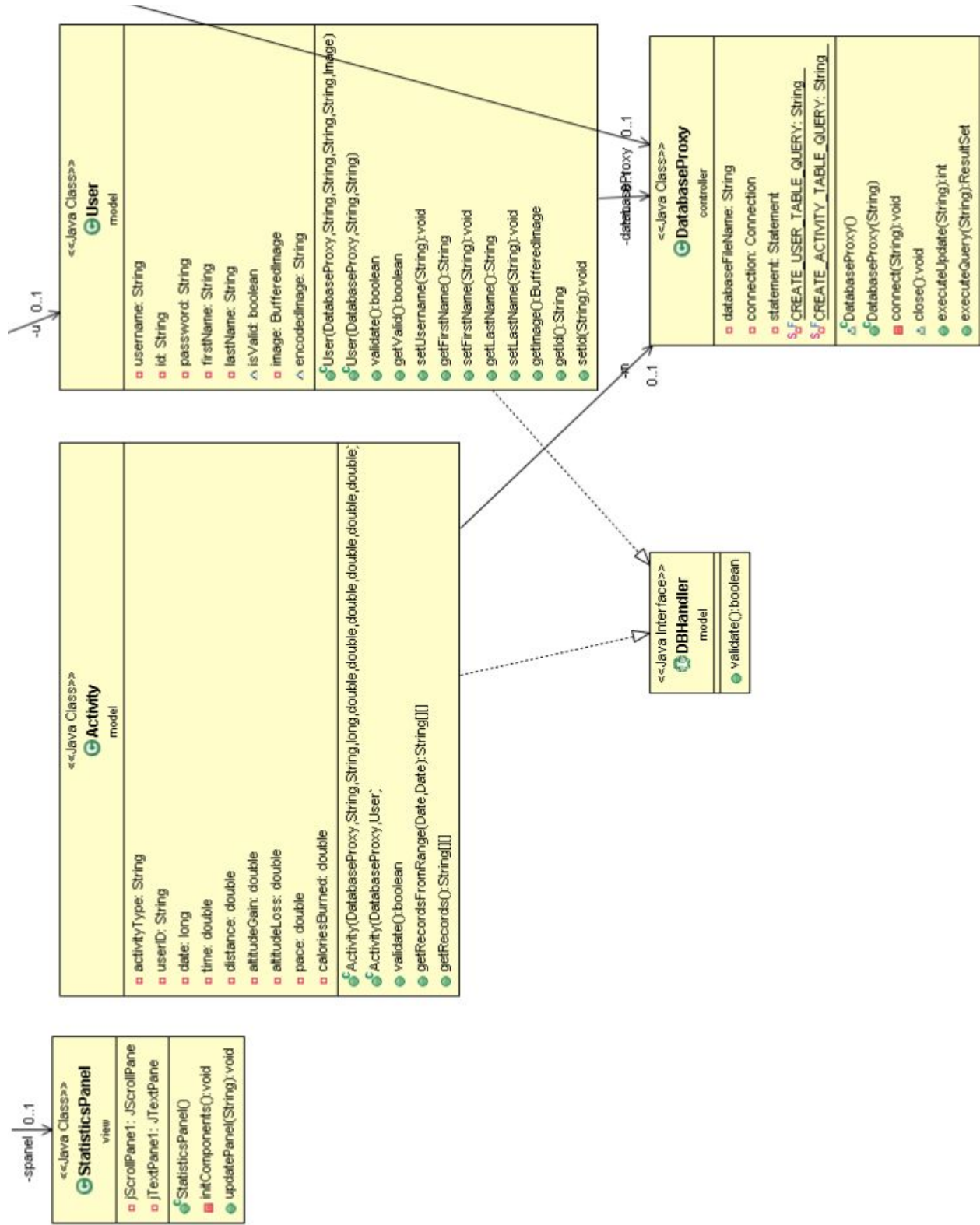


6.3. Detailed Class Design

The class diagram below describes the various entities that compose the functioning system. All the components are properly labeled as either being a part of **view**, **controller** or **model** modules.








7. Testing

1) Creating Profile

The user attempts to create a profile, by clicking on the create profile button. That switches to a different window where user has to insert essential information, like username/password, and confirms the information inserted. After doing so, the user is logged in and the screen switches to the main app page where it shows the currently connected devices which the user can import activity data from.

Catch-up! Standard Edition

User Details

 not logged in
edit profile logout My Friends Import Data My Activity

User Login

Username

Password

Sign In Create Account

Login successful

The screenshot shows a window titled "Catch-up! Standard Edition". Inside, there are two main sections: "User Details" and "User Registration".

User Details: This section contains a profile picture placeholder, the text "not logged in", and buttons for "edit profile", "logout", "My Friends", "Import Data", and "My Activity".

User Registration: This section contains a form with the following fields:

- First Name: John
- Last Name: Doe
- Email: john@gmail.com
- Username: john
- Password: (masked with dots)

Below the form are two buttons: "Create Account" and "Sign In".

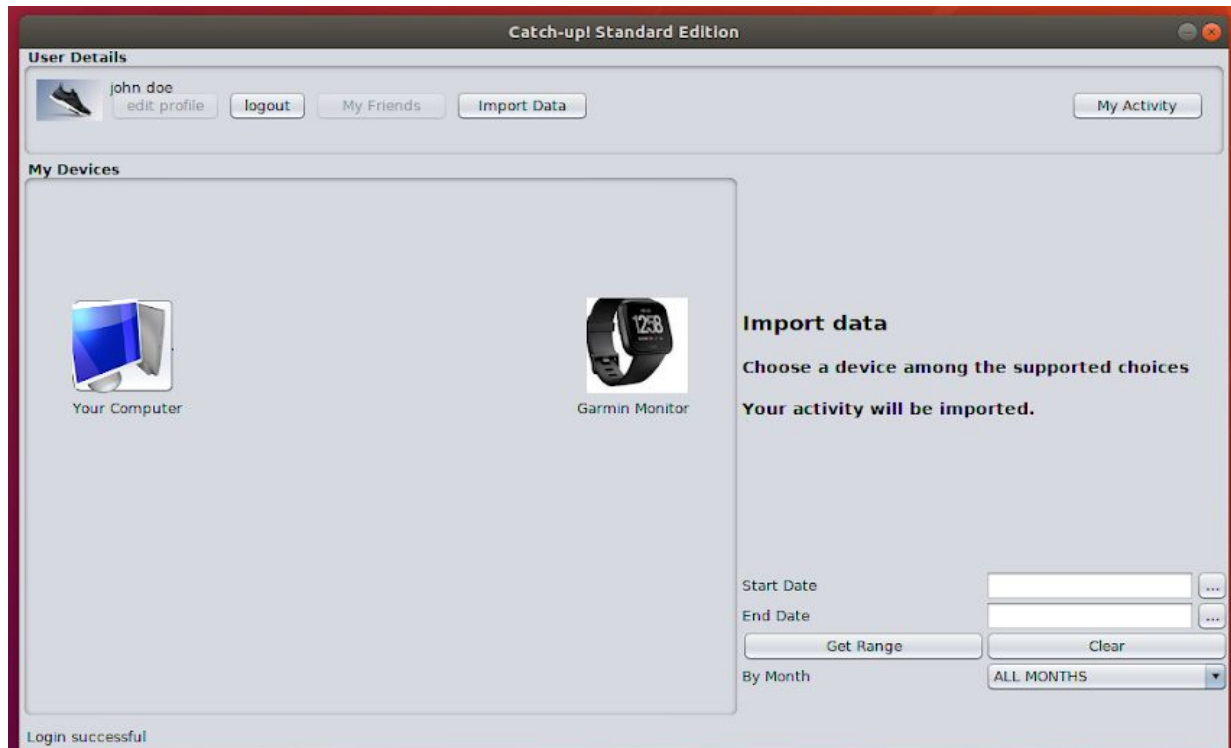
At the bottom of the window, a message reads "Login successful".

2) Logging in

After creating a new account, the user is promptly logged in and brought to the starting page of the application. Alternatively, the user can also type his username and valid password to log-in the application. In the case the credentials are invalid, the user is asked to retry his credentials

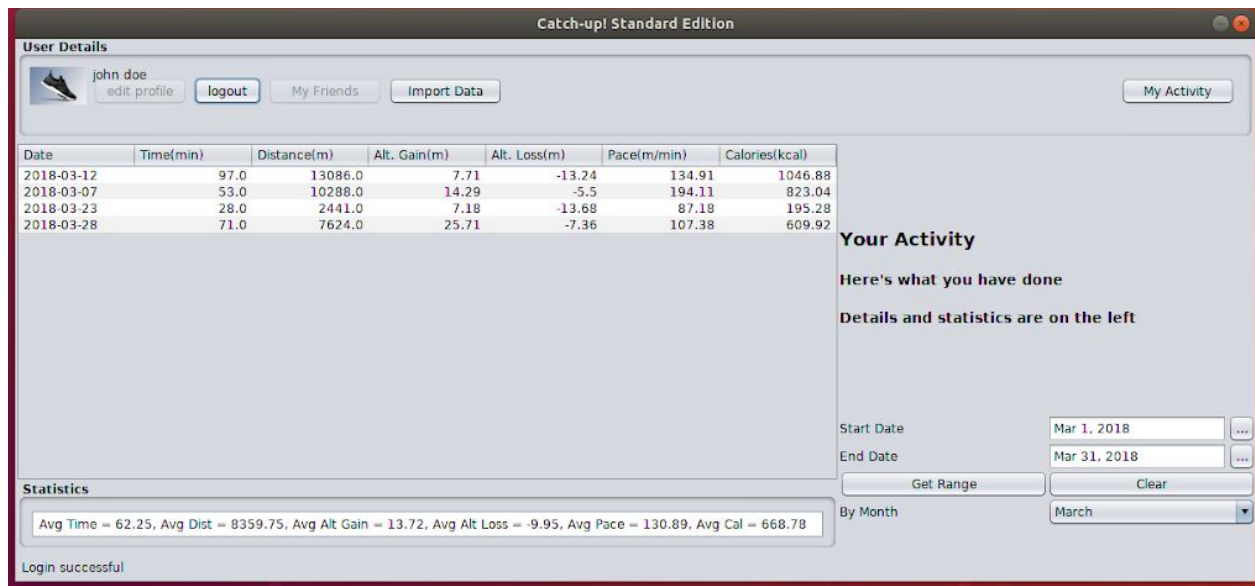
3) Import Data Activity

In this screen, the user selects the source where data will imported from. In this version, the only enabled option is to choose a file from the computer, however, the application is prepared to incorporate future changes/extensions.

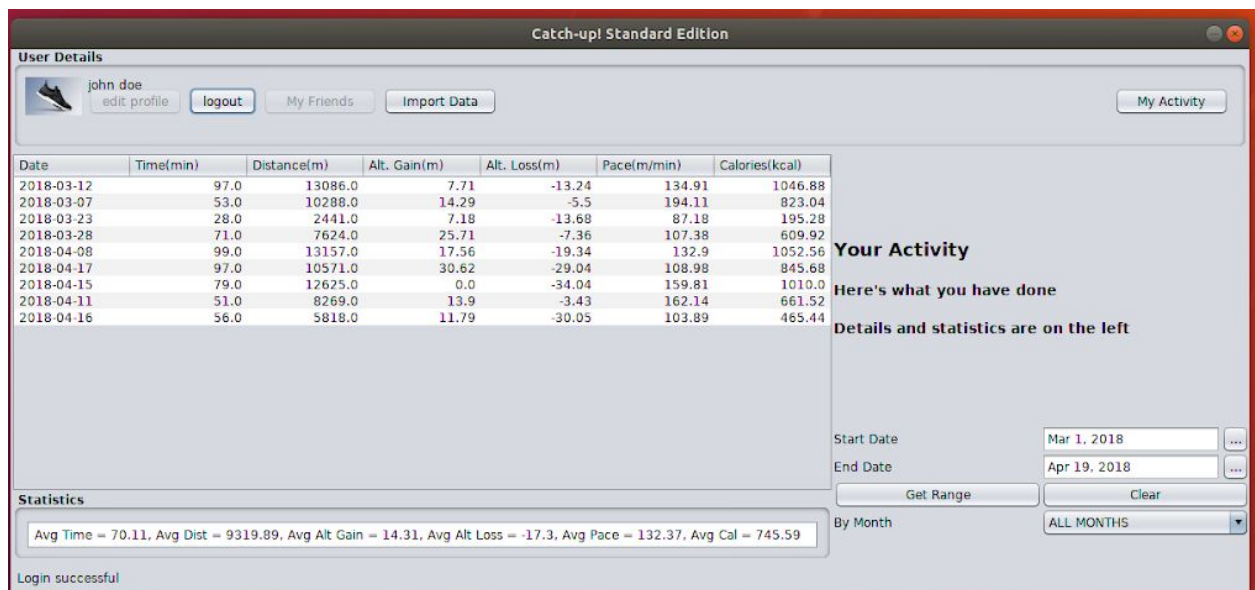


4) Display Statistics

After importing data and clicking under “My Activity”, the screen will change to the one shown below. In this screen the user is able to filter runs by a specific interval of dates or a given month. As data appears on the screen, the statistics are instantly calculated and shown at the bottom part of the screen. The figure below shows a sample filtering done with a sample set of imported data.




This figure shows the additional data shown when the time frame is increased.



Finally, this last picture shows the result of when the user clears the query by clicking on "Clear". This makes all the available data to be displayed.

Catch-up! Standard Edition

User Details

john doe

edit profile

logout

My Friends

Import Data

My Activity

Date	Time(min)	Distance(m)	Alt. Gain(m)	Alt. Loss(m)	Pace(m/min)	Calories(kcal)
2018-01-07	35.0	4526.0	22.59	-45.5	129.31	362.08
2018-01-03	56.0	11012.0	32.82	0.0	196.64	880.96
2018-01-08	78.0	11249.0	36.88	-31.52	144.22	899.92
2018-01-03	47.0	9392.0	23.71	-22.47	199.83	751.36
2018-01-02	96.0	8624.0	16.27	-18.07	89.83	689.92
2018-01-22	92.0	4903.0	0.24	-21.14	53.29	392.24
2018-01-27	89.0	10234.0	0.0	-8.38	114.99	818.72
2018-02-03	74.0	11586.0	36.72	-5.05	156.57	926.88
2018-02-04	61.0	10116.0	14.66	-11.48	165.84	809.28
2018-02-09	97.0	10409.0	0.0	-34.82	107.31	832.72
2018-02-24	90.0	7977.0	35.05	0.0	88.63	638.16
2018-02-22	87.0	11164.0	17.65	-31.66	128.32	893.12
2018-03-12	97.0	13086.0	7.71	-13.24	134.91	1046.88
2018-03-07	53.0	10288.0	14.29	-5.5	194.11	823.04
2018-03-23	28.0	2441.0	7.18	-13.68	87.18	195.28
2018-03-28	71.0	7624.0	25.71	-7.36	107.38	609.92
2018-04-08	99.0	13157.0	17.56	-19.34	132.9	1052.56
2018-04-17	97.0	10571.0	30.62	-29.04	108.98	845.68
2018-04-15	79.0	12625.0	0.0	-34.04	159.81	1010.0

Statistics

Avg Time = 70.89, Avg Dist = 9575.06, Avg Alt Gain = 16.88, Avg Alt Loss = -16.74, Avg Pace = 136.19, Avg Cal = 766.01

Your Activity

Here's what you have done

Details and statistics are on the left

Start Date

End Date

By Month

Get Range

Clear

ALL MONTHS

Login successful