GET SWOLE - Final Report
Dartmouth College, COSC 65
Professor Andrew Campbell

Designed by Cameron Price, Andrew Pillsbury & Patricia Neckowicz

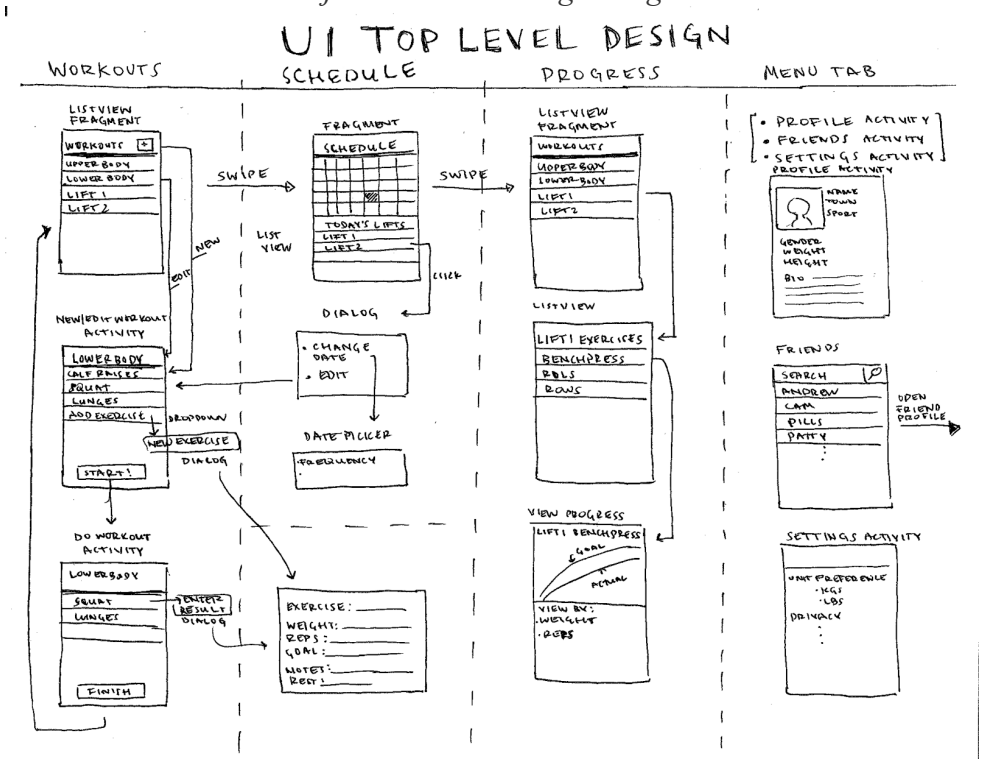
Website: http://getswoleapp.weebly.com/
Google Play:
https://play.google.com/store/apps/details?id=cs65s14.dartmouth.get_swole

1. Introduction - The inspiration for GET SWOLE

At a base level, GET SWOLE is meant to replace using a paper and pencil at the gym. We recognize the hassle in writing down a workout every time it is done, so with our app, we provide the user with a clean interface to customize and keep track of workouts. However, GET SWOLE is not just a record keeper. The app keeps track of your progress - showing you how you are doing in each exercise. It also keeps you accountable by keeping track of your workout schedule and sending you reminders on the days you have workouts. Finally, GET SWOLE keeps you connected with other athletes, allowing you to download their workouts (schedules included). Overall, GET SWOLE is meant to be your very own workout assistant.

2. Architectural Design

Object Oriented Design Diagram



Class Descriptions

GetSwoleClass.java
A parent class for Workout, WorkoutInstance, Frequency, and Exercise.
Instance variables:

- name: The name of the instance (not used for Frequency)
- id: The id of the instance in the database

Methods:

- Getter and setter methods for both variables.

Workout.java
A class for workout skeletons, containing information about exercises and scheduling.
Instance variables:

- exerciseList: A list of Exercise objects associated with the workout
- startDate: The date that the workout was created
- scheduledDates: A list of Calendar objects corresponding to individual days that this workout is scheduled.
- frequencyList: A list of Frequency objects associated with this workout
- notes: Notes or comments about this workout.
- regId: The phone's registration ID for the appengine.
- owner: The name of the creator of the workout.

Methods:

- Getter and setter methods for every field.
- Byte array conversion methods for frequencyList and exerciseList.
- JSON conversion methods for the workout, frequencyList, and exerciseList.
- String conversion methods for scheduledDates.
- Methods for adding and removing dates, frequencies, and exercises.

WorkoutInstance.java
A specific completed workout, with the exercise information completely filled out (including weights and reps).
Instance variables:

- workout: A reference to the workout containing the skeleton for this instance.
- time: The time this instance was completed.
- exerciseList: A list of Exercise objects corresponding to this completed workout with weight and reps completely filled out.

Methods:

- Getter and setter methods for every field.
- Byte array conversion methods for exerciseList.

Exercise.java
An exercise with some or all of the information filled out.
Instance variables:

- setList: A list of Set objects associated with this exercise
- repsGoal: The user's goal for the maximum number of reps to do on this exercise

- weightGoal: The user's goal for the maximum amount of weight to do on this exercise
- rest: The number of seconds the user should rest in between sets.
- exerciseInstance: A flag that is true if this is an instance of an exercise filled out for a WorkoutInstance object, and false if it is an exercise skeleton.
- oldId: For exercise instances associated with WorkoutInstance, oldId holds the id of the exercise skeleton. This is used for progress.
- notes: Notes or comments about this exercise.

Methods:

- Getter and setter methods for every field.
- JSON conversion methods.
- String conversion methods for setList.
- Methods for adding and removing sets and getting the maximum weight or maximum reps done in the sets.

Frequency.java

A class for holding the frequency with which a workout should be scheduled.

Instance variables:

- day: The day that the frequency repeats on
- startDate: The earliest day that the frequency could be.
- endDate: The latest day that the frequency could be.

Methods:

- Getter and setter methods for every field.
- Methods for converting to and from a JSON object.
- String conversion methods for scheduledDates.
- Methods for adding and removing dates, frequencies, and exercises.

Set.java

A class holding a weight and a number of reps meant to represent a set of an exercise.

Instance variables:

- reps: the number of reps to do in this set
- weigth: the weight at which to do those reps

Methods:

- Getter and setter methods for every field.

ProfileObject.java

A class meant to store profile information that is sent to the Google App Engine and maintain a social network.

Instance Variables:

- regId: String to save the registration id of the user's device.
- pictureString1: First half of string converted from profile picture byte array (split for memory size issues that cause corruption when importing inside a Text object in the cloud).
- pictureString2: Second half of string converted from profile picture byte array
- firstName: String to save first name.
- lastName: String to save last name.
- hometown: String to save the name of the hometown.

- sport: String to save the name of the sport.
- gender: Int that saves the gender selected by the user (0 = Male, 1 = Female, -1 = Not entered).
- height: Double to save height in inches.
- weight: Double to save height in pounds.
- bio: String to save custom bio information entered.
- email: String to save the email.
- phone: String to save the phone number.

Methods:

- Getter and setter methods for every field
- Methods for converting to and from a JSON object.

3. Team Roles in Developing App

Andrew -- database, classes, calendar
Andrew created the entire backend with the database and all the classes except for the ProfileObject class. He also made the schedule fragment, and dealt with scheduling for the workouts. He used code from https://github.com/mukesh4u/Android-Calendar-Sync to create a basic calendar.

Patty -- UI, workouts, progress, dialogs for the app
Patty spent a lot of her time making the user interface for GET SWOLE as user friendly and intuitive as possible. In particular, the idea behind the workouts design is that the user should have complete flexibility when creating a workout. On the other hand, the user should not have to do much work to enter workout stats while doing a workout. Patty also implemented the workouts and progress portions and wrote the dialogs used all over the app. She used the GraphView library (at https://github.com/jjoe64/GraphView) to display progress.

Cameron -- appengine, profile, friends, settings
Cameron focused on the App Engine and social network implementation, and wrote the user interfaces and methods for each of the following: ProfileActivity, FriendsActivity, FriendProfileActivity, ProfileObject, and SettingsActivity. This side of the app adds functionality beyond the app's personal uses by allowing the user to upload profile information, workouts, and schedules to the Google Cloud, and view other users' data and personal information. Cameron also worked on the settings activity, which allows the user to change units for weights/heights and manage data.

4. How to Run the Demo

How to Download Source Code (User Credentials Required)

Code: https://svn.cs.dartmouth.edu/classes/cs65-S14/GETSWOLE/Code/

In this code svn repository you will find the following:

GET_SWOLE_APP/	Directory that holds all the source code for the final version of the app.

GET_SWOLE_APPENGINE/	Directory that holds all the source code for the final version of the App Engine side of the app.
google-play-services_lib/	Directory of Google Play Services library needed to run the app.
GET_SWOLE_APP.apk	The final version of the apk for the app. Download this to directly install the app to your Android device.
GET_SWOLE_APP.zip	Zip file that contains all source code held in GET_SWOLE_APP/.
GET_SWOLE_APPENGINE.zip	Zip file that contains all source code held in GET_SWOLE_APPENGINE/.

How to Compile and Run the App

1. Download GET_SWOLE_APP.zip from the SVN repository and unzip it. Open Eclipse and go to File → Import → Android → Existing Android Code Into Workspace and choose the unzipped GET_SWOLE_APP.
2. If you don't have the google-play-services library, check that code out from the SVN (svn co https://svn.cs.dartmouth.edu/classes/cs65-S14/GETSWOLE/Code/google-play-services_lib/). In Eclipse, go to File → Import → General → Existing Projects Into Workspace and choose the google-play-services_lib folder.
3. Open the project properties for the GET_SWOLE_APP project, and choose "Android" from the sidebar. In the Library section there will be a broken reference to the google-play-services library. Remove that reference, then click "Add" and choose google-play-services_lib.
4. If you are running the App Engine code locally, make sure to change server_url in strings.xml with the appropriate IP Address of your computer (i.e. http://IP_ADDRESS:8888) and follow the steps for compiling and running the App Engine. Otherwise, leave server_url as it is because the App Engine code is already deployed at http://getswoleandroidapp.appspot.com/.
5. Run the app as an Android Application.

How to Compile and Run the App Engine (Running App Engine Locally)

1. Download GET_SWOLE_APPENGINE.zip from the SVN repository and unzip it. Open Eclipse and go to File → Import → Android → Existing Projects Into Workspace and choose the unzipped GET_SWOLE_APPENGINE.
2. Before running the App Engine be aware that datastore data is already saved in the local_db.bin file. If you want to open an empty datastore in the App Engine go to GET_SWOLE_APPENGINE → war → WEB-INF → appengine-generated and delete local_db.bin.
3. Run the project as a Web Application.

5. Lessons Learned

The development of GET SWOLE did not come without some difficulties. However, because we managed our time well throughout the process, we were able to make small but important changes to our app design during the last few days of the project. We also found it useful to outline the user interface and structure of the app (database, classes, etc.) before coding. After all of this, we recognize that there are always improvements to be made, and below we outline some changes we think could take our app to the next level.

6. Conclusion

As stated before, GET SWOLE is meant to be a user's workout assistant, one that enables the user to create custom workout logs, track exercise progress, create schedules, and share those workouts and schedules with other users on the web. As a team, we really enjoyed coding this app, and we think that it will be quite useful for users who want to get organized and do workouts with friends. To better improve this app, we think that adding privacy settings, which would force users to search for one another and make connections via friend requests, would be beneficial for creating a well-managed social network. In addition to that, the Google Cloud Storage methods for saving user profile information and workout information could be upgraded to handle a much larger audience. On the workouts side, we recognize some weaknesses in the exercise database design; when a user edits an exercise, a new one is created in its place so that the original exercise is not altered in other workouts. However, this overwrites all progress data that previously existed for this exercise. The scheduling could also be improved, with features added to remove a single day from a frequency, remove a frequency entirely, or view and edit frequencies. Overall, we are happy with final version of GET SWOLE and hope that others find it as useful as we do.