Holly Harmon
CS 496
Wolford
OSUID: 932456676

# Final Project Documentation

**Project Demo:**
https://onedrive.live.com/redir?resid=47A7187A7DC0EB00!107&authkey=!AFNDIY8oSUDT6aE&ithint=video%2cmp4

(Backup demo link: https://drive.google.com/file/d/0B_K4kIFR2GzIVzktdDU2c0NsR2c/view?usp=sharing)

## API

My API is a continuation of the API I made for Assignment 3. It utilizes my Google Cloud Datastore to store data.

**Base URL:** http://api-holly.appspot.com/

**URI Structure:**

**/user:** Post allows for creation of new User entities. Requires a unique "name" value and a "password". A get request will return a list of all of the User entity keys.

**/user/<id:[0-9]+><:/?>:** user/ followed by a user's key value (or "id") with a Delete request will delete the indicated User entity along with all of its associated BikeEntry entities. A Put request will update the user's information.

**/bike:** Post request allows for BikeEntry entity creation. Must include a user key/id and a date. Get will display the keys for all BikeEntry entities.

**/bike/<id:[0-9]+><:/?>:** Get request will return the information for the BikeEntry entity indicated by the key/id.

**/run:** Post request allows for RunEntry entity creation. Must include a user key/id and a date. Get will display the keys for all RunEntry entities.

**/run/<id:[0-9]+><:/?>:** Get request will return the information for the RunEntry entity indicated by the key/id.

**/user/<uid:[0-9]+><:/?>/bike/<bid:[0-9]+><:/?>:** A Put request will update the indicated bike entry with the provided data and a Delete request will delete the BikeEntry entity and remove reference to it from the User's bikes[] list.

**/user/<uid:[0-9]+><:/?>/run/<bid:[0-9]+><:/?>:** A Put request will update the indicated run entry with the provided data and a Delete request will delete the RunEntry entity and remove reference to it from the User's runs[] list.

**/auth:** A post request with passed "name" and "pass" values will compare the name and password with the user's in the database. If the user does not exist or the password does not match the user's password in the database an error message is returned, otherwise the user is considered valid and their key/id is returned.

## User Accounts

Since I was already struggling a bit with Windows Phone development, I decided against going with OAuth for my user authentication and instead implemented a simple username and password system through my API. This app does not contain any sensitive information, so security was not a high priority for me. I used the same API I turned in for Assignment 3, but made some changes so it could function better for what I needed. I added a "password" property to the User entities and created a new auth URI to cover user authentication.

The app can make a POST request to http://api-holly.appspot.com/auth with the user's "name" and "pass." The API checks the name against the database and, if the username exists, will check the password associated with that user. If the username and password can be authenticated it will return the user's key/id which can be used to get more of the user's information, a list of their bike entries, and allow for update/deletion of entries. /auth returns status errors/messages if the username does not exist, the password is incorrect, or the app did not send a name or password variable with the call.

The security of this system is very low as the passwords are stored as string properties in the Google Datastore and the user's key/id is stored in the phone's temporary data while the user is logged in to the app. Any tech savvy person would probably find it pretty easy to obtain someone's user id and then they can access a user's information whether they have the password or not.