Aryan Aziz
aziza@onid.oregonstate.edu

CS496 – Final Project

API

I wrote 3 main API's for my app that handle all of the creating, updating, modifications, and deletions for my app. Each of the APIs are completely self-contained in their single PHP files, and use the request type/parameters passed to determine what the API should accomplish.

- **Users**: http://52.11.75.34/users.php
  - GET:
    - Without any params: Will return all users registered in the system, with their id, username, team they are following, and link to the user.
    - With the "username" param: Will return that specific user, with their id, username, and team they are following.
      - EX: http://52.11.75.34/users.php?username=testaccount2
    - With both the "username" and "password" param: Will return that specific user with their id, username, password, and team they are following. The idea is that this would have access to more private information.
      - EX: http://52.11.75.34/users.php?username=testaccount2&password=testing
  - POST:
    - Must have both the "username" and "password" param. Will create a new user or return an error if it cannot (either missing param, or username taken, etc)
      - EX: username=newaccount&password=pass
  - PUT:
    - Must have both the "username" and "teamId" param. Will modify the user's account to either add this team as their team (if they don't have one already), or change the current team to this one.
      - EX: username=oldaccount&teamId=3
  - DELETE:
    - Must have the "username" param. Will delete the specified user's account. In hindsight, this should probably also required password; otherwise anyone could delete anyone's account.
      - EX: username=oldaccount
- **Teams**: http://52.11.75.34/teams.php
  - GET:

        - Without any params: Will return all teams that are setup in the system.
        - With the "teamId" param: Will return that single team and all available information for it.
            - EX: http://52.11.75.34/teams.php?teamId=2
        - With the "username" param AND "type" param:
            - If Type = 1. Will return all teams that the user is following.
            - If Type = 2. Will return all other teams (the ones they aren't following)
            - Please note: This was meant to be the primary way of handling it, and allowing people to follow multiple teams (hence the need for this). However I decided to not follow this route. These methods should still work, however, as long as the specific user has a "teams" array in their account.
        - With the "lat" and "long" param: Will take your lat and long, then go through every team in the system and determine which one is closest to you. It will then return that single team, similar to passing "teamId".
            - EX: http://52.11.75.34/teams.php?lat=42.3&long=-71.0
- **NBARSS**: http://52.11.75.34/NBARSS.php
    - GET
        - Must have the "team" param. What this will do is take in a "teamId", and then query the team's table for that id and grab the "nbaId" from it. It will then take that nbaId and grab the nba.com rss feed for that specific team. Once it does that, it will convert the rss xml into valid json and return that valid json.
        - EX: http://52.11.75.34/NBARSS.php?team=1

Account System

My account system was coded by scratch, and takes in a username and password from the user and ensures they are identical (both the username and password are case sensitive).

Because of this, there's a very low standard of "authentication" on my end. The only thing I am taking in is a case sensitive username and password from the user, whoever the user is, wherever the user is. I didn't worry about it too much, because this app is just one to follow your favorite team(s) and see their news, so there shouldn't be any items you can lose or get spoofed on; only the possibility of account deletion.

Lessons learned:

- I shouldn't use GET for authentication. Sending it in the clear is an easy risk of security vulnerability.

- The password should never be sent in plaintext, I should be encoding it always.
- All items should be encoded and checked for possible mysql injections
- I don't check for any custom chars, all string values and numbers are allowed.

Overall Lessons Learned

I actually enjoyed the API side of things much more than the iOS development side, which surprised me. iOS was VERY finicky, and I got caught trying to debug minor things that caused me huge issues (for example, it took me 2 days of debugging to get the tableview to generate properly. Not of coding, of debugging). Because of this, I got caught on time issues with the iOS development that ultimately led to not having multiple teams per user available on the app. The API functionality was there, but integrating it into the app was not possible.

Also, I really like having all of my API functions available through the single endpoint and letting it handle what it needs to do. Rather than having a "createUser.php" and "getUser.php" and "loginUser.php" and "editUser.php" and "deleteUser.php", it seems much cleaner and more effective to just have a "users.php" and let the php file handle what it needs to do based on the request type and variables it's being passed. This might not be the best way for large scale applications with many features, but for my smaller ones it makes things cleaner and more concise.