CS496 Final Exam

Eric Rouse

An Analysis of the Rothbowl Application

An exploration of the 6 quality attributes as they relate to the Rothbowl cloud backed mobile application.

The Rothbowl application is a combination of mobile front-end and cloud back-end. When coupled it that can add and retrieve local notes, view current position and save user-specific remote data, as well as add, update, delete and view the players and bowls via the rothbowl API. It also implements a log in/log out interface.

Table of Contents

Usability:

Users can accomplish intended use cases quickly and correctly.

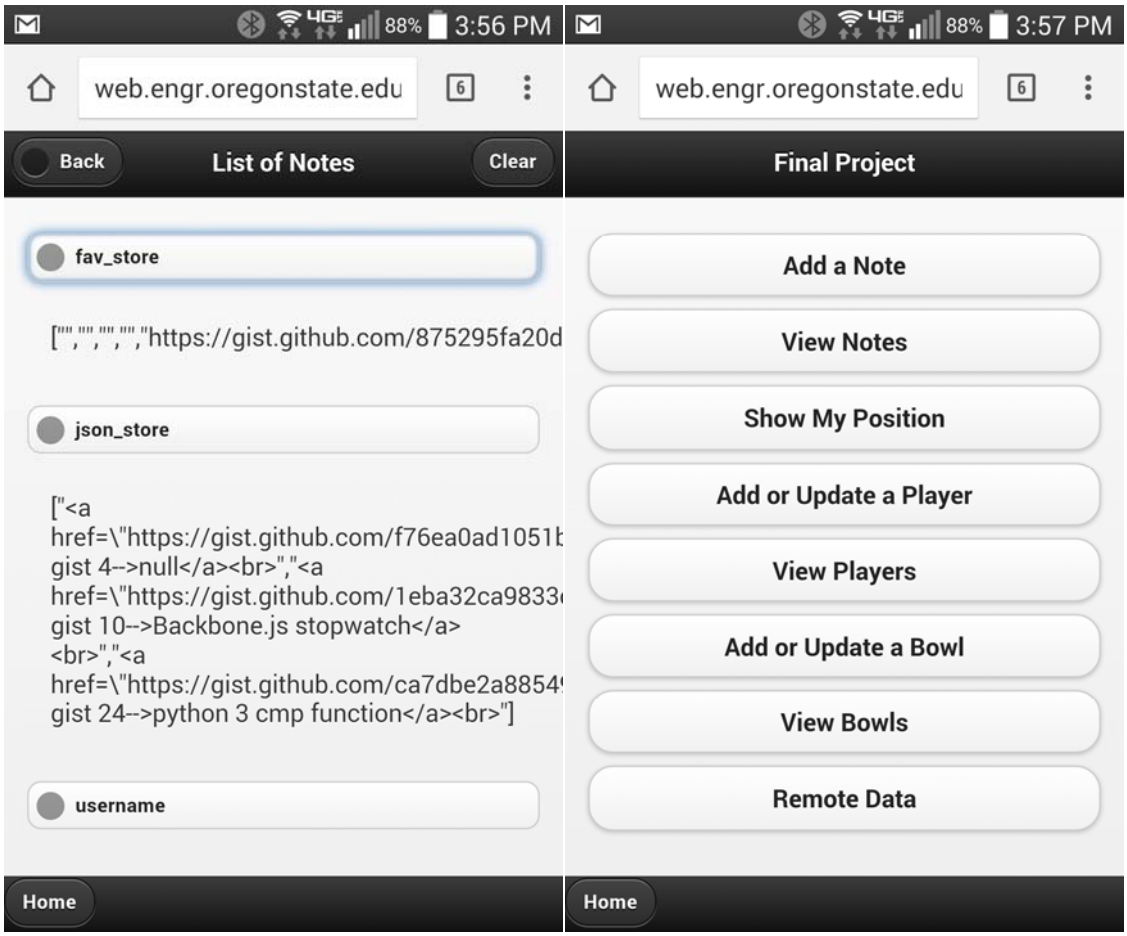
The Rothbowl application is very simple and incredibly responsive. It only does a couple of things so it is very focused. And the UI is consistent with adequate aesthetics so as to not hurt usability. As this is a very basic app, no design guidelines were followed, although the app has a consistent look and feel throughout.

Portability:

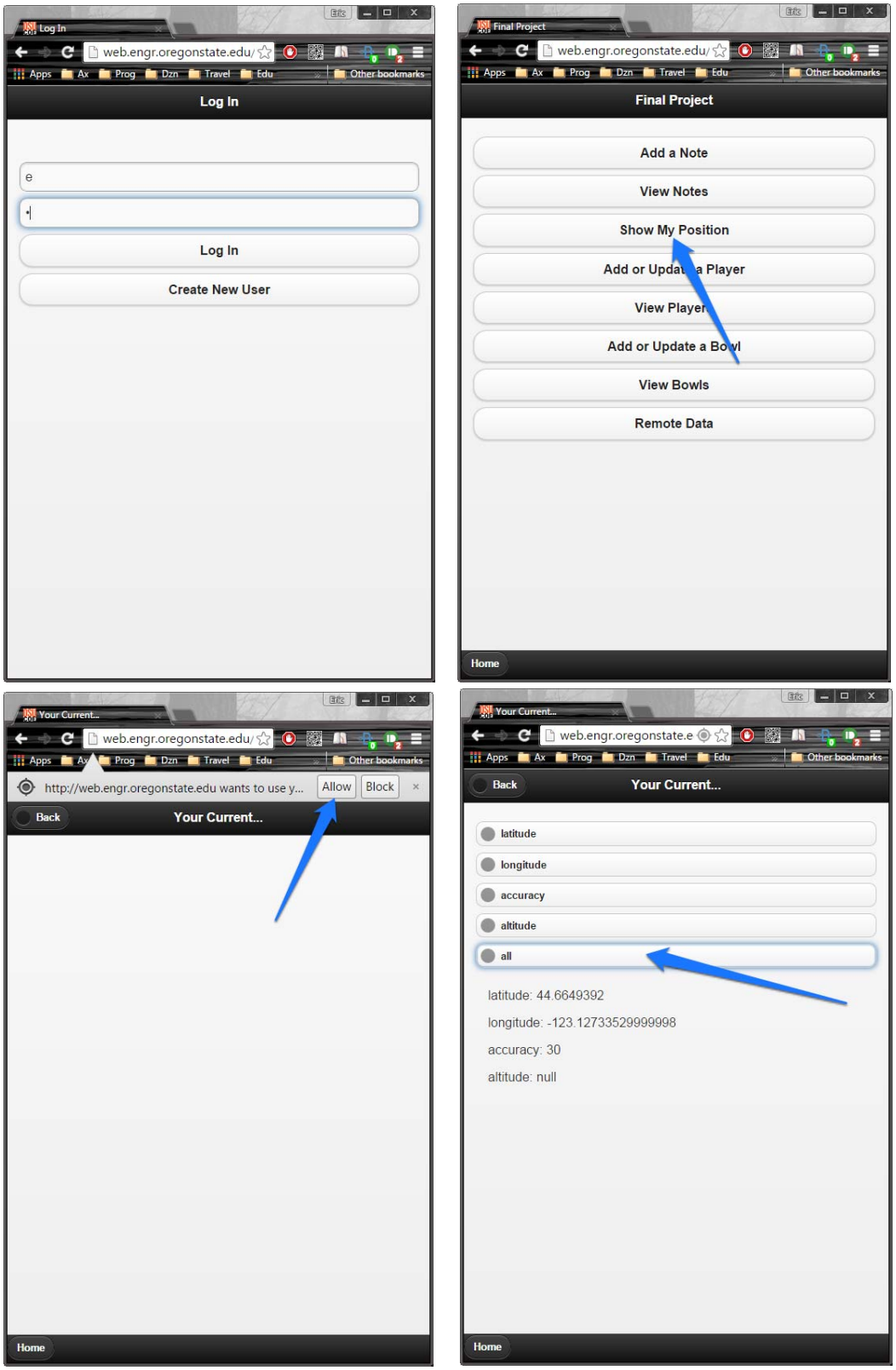
The same program runs properly on multiple platforms without changes to code.

Rothbowl is made using standard web technologies like CSS, HTML and JavaScript. As such it can run on pretty much any web server. This makes it very portable. Literally any device can run the web version (http://web.engr.oregonstate.edu/~rousee/CS496/rb/) in a browser. (Try it out with username “e” password “e”).

Also since RothBowl uses jquery for all the user interface construction the application front end is very dynamic and resizes nicely for any screen size.



Examples of the application working on a web browser on an android device.



Four images showing the show position function of the application using a web browser on a windows machine.

Reliability:

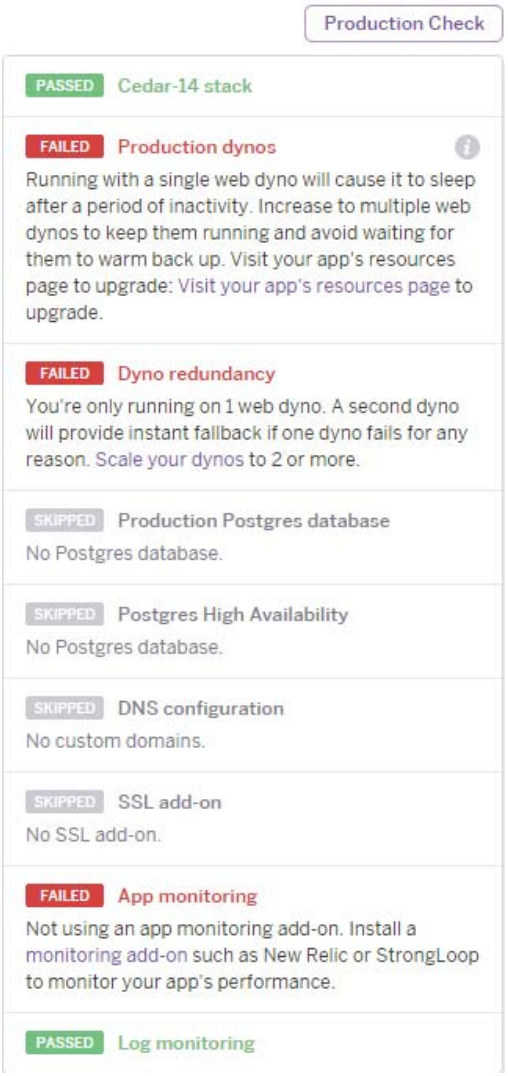
Software works correctly all the time.

The rothbowl back-end is hosted by heroku. Heroku uptime has been in the “four 9’s” range for the last year. So, the platform is pretty reliable.

However, the app itself uses the most basic (free) account so it only has 1 “dyno”. In Heroku-speak, a dyno is a virtual machine that runs the application code (python on django in our case). Since we only have a single dyno we are not redundant, so if a dyno fails we have no fallback.

Rothbowl is also unmonitored, so if it crashes there is no notification given. It falls silently, like a ninja.

Heroku US applications uptime; https://status.heroku.com/uptime



Performance:

Software uses few resources for the work being done.

The rothbowl application hasn't really been optimized for performance. For starters, it doesn't try to parallelize anything, or use separate indexes for query entities. In fact, query indexes are built on-the-fly or parsed on the mobile device end. Also, there is no local caching on the device side.

That said there are a few things that it does reasonably well performance-wise.

1. It does minimize traffic by having very few messages.
2. The messages are simple strings, no blob files or images, it's just a simple ask and answer system that returns a simple string.
3. It doesn't allow multiple accessing of resources, so there are no locks used.
4. The thing that really saves us is that the application actually doesn't do that much. On the cloud back-end it monitors an API and accesses a database in response to return a string. On the mobile front-end it makes API requests and renders the returned data.
5. And the user base is not very large, just 1 person. So, the backend doesn't get much traffic! ☺

Security:

Software keeps information confidential.

Security is about

- Confidentiality – Keep secrets a secret
- Integrity – Keep secrets from being modified by the unauthorized
- Availability – Keep access to secrets available to the authorized

As such, the security of the Rothbowl application is pretty abysmal. The application was developed first and then security was “put on top” afterward.

On the front-end, security involves a simple log in interface. A username and password are sent to a PHP login page and the application waits for a response. A response of “0” is all it takes to “authorize” the account. So, it is very easy to “crack” the log in process by hijacking the outbound request and sending back a message of “0”.

Also, the log in data are sent via a POST message. The password is not hashed or modified in any way. Hence, a man-in-the-middle attack wouldn’t just grab your password to the application data, but, any application where you use the same username and password.

Once logged in, the application assumes the user is who they say they are, no further checks are ever made.

Obviously, since the username and passwords are so terribly handled, there isn’t any point in offering two-factor authentication. And it doesn’t.

There is no encryption, either on the server or on the device, so all stored data would be plainly available if successfully harvested by an attacker.

On the plus side, it doesn’t remember your password and requires a login every time.

Interoperability:

Software can exchange data with other systems.

Rothbowl doesn't interact with anybody. This is good and bad. Because the back-end and front-end are only designed to operate with each other we aren't reliant on any third party APIs. So, we are unaffected by third party applications that suddenly stop working.

Because interoperability means integration through platform specific APIs. This means the application is subject to the whims of unknown API developers. They can (and sometimes do) change the API format of their platform at will, without warning.

But, our users do lose the ability to connect to useful things. It would be cool to post your Rothbowl results to Facebook for instance. Or view team or bowl data in the app itself, rather than look it up in a web browser.

Improvements:

If I were doing this again I'd make sure all returned objects were JSON by default, rather than plain text. I'll chalk that up to a rookie mistake, JSON objects are so much easier to parse – making the code more portable!

Also, I'd store certain indexes to make database access more efficient. I didn't fully understand how to use NoSQL when I started and kind of painted myself into a corner here. My database scheme was far too simplistic to be realisitic.

Security also should be drastically improved; adding some basic password hashing or Oauth support would be my next major revision, were I continuing to build the app. Or reworking the application with security being built first.

To Sum Up:

The Rothbowl front and back ends are pretty simple. It is reasonably portable and usable. But marginally reliable and although it should perform agreeably for a single user. The security is just awful. And it is not at all interopable.

Fin

So long and thanks for all the fish!