

SE31520
Developing Internet-Based Applications

Enhancing the CS-Alumni Application

Submission Due 4pm Monday, Decemeber 8, 2014

Thomas Mark Rosier (THR2)
110113188

Contents

Document Summary	3
Application Design	3
CS-Alumni Chrome Browser Extension	3
Rational	3
Module Diagrams	3
Program Operation	4
CS-Alumni Rails Application	9
New Additions	9
Modifications	9
Communication Between Applications	9
RESTful Web Interfaces	9
Data Flow Diagrams	9
Application Testing	10
QUnit Tests	10
CS-Alumni Chrome Browser Extension	13
Javascript	13
Chrome Extensions API	13
Bootstrap	13
jQuery	13
Moment.js	13
Alertify.js	13
Qunit	13
CS-Alumni Rails Application	13
Ruby	13
Ruby on rails	13
Qunit	13
Communication Between Applications	13
RESTful Interfaces	13
Cors	13
Basic Authentication	13
Evaluation	14
Something Extra	15

Document Summary

Application Design

CS-Alumni Chrome Browser Extension

Rational

My rational behind deciding to developing a browser extension for this project, was that they have been a technology that I have been following for a while and I have not seen them used that regularly for small little applications that interface with a bigger application.

Another reason for my choice to develop a browser extension was that I wanted to learn about implementing some of the more advanced features within the Chrome SDK for example the use of 'Background Pages' which are used to keep a task running in the background while the extension is closed.

One of the newer features within the Chrome SDK had also caught my attention in the weeks before starting this assignment was the use of Chrome desktop notifications where you can create a notification within your application that will be displayed to the end user via the Chrome UI's notification draw.

I wanted to see if it was possible to encapsulate some of the key management features of the CS-Alumni application into an attractive and well presented browser extension that also notified the user of any new broadcasts that had been made.

Module Diagrams

Program Operation

When the user opens the browser extension they will be presented with the home screen, this shows a small blurb about the application along with a disclaimer.

Due to this being an assignment this is essentially a development version of the browser extension thus I have included a link to the Unit tests on the home page to make it easily accessible during the development and testing of the application, if this extension was intended to be released as production code this link would be removed or turned off within the configuration.

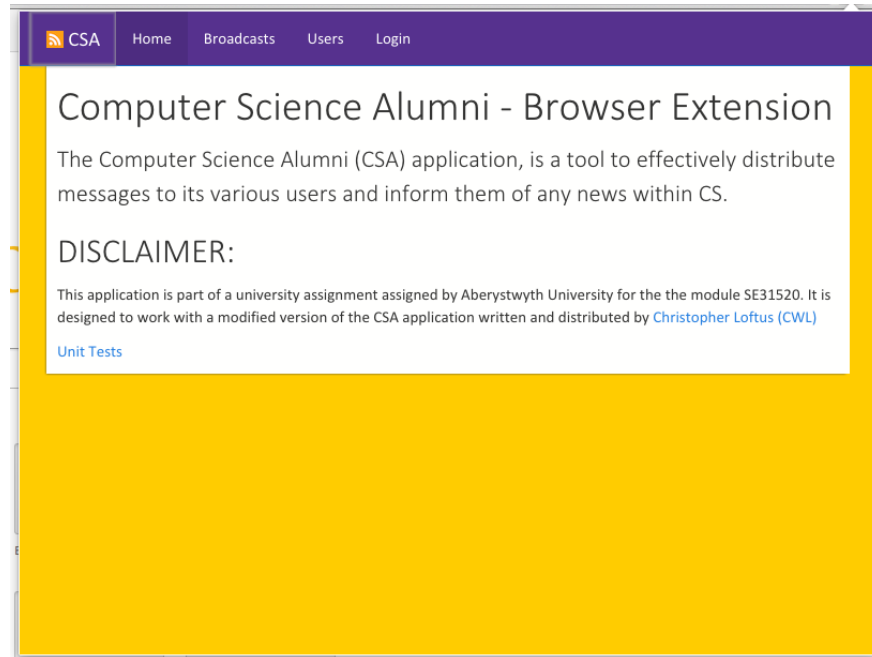


Figure 1: The screenshot above is the landing screen for the browser extension.

From the home page the user can navigate to the 'Broadcasts' tab, in the figure captioned there is no broadcasts available to the user so they are show a message that says that there is no broadcasts.

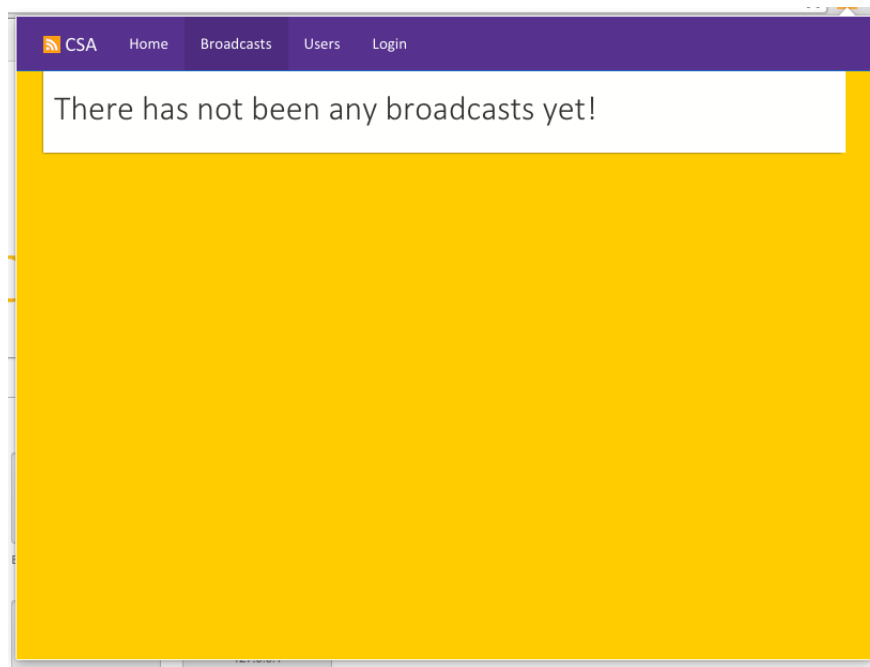


Figure 2: Above we see the broadcasts screen with no broadcasts available.

Again below we see the 'Broadcasts' tab this time we see how the page appears when there is broadcasts.

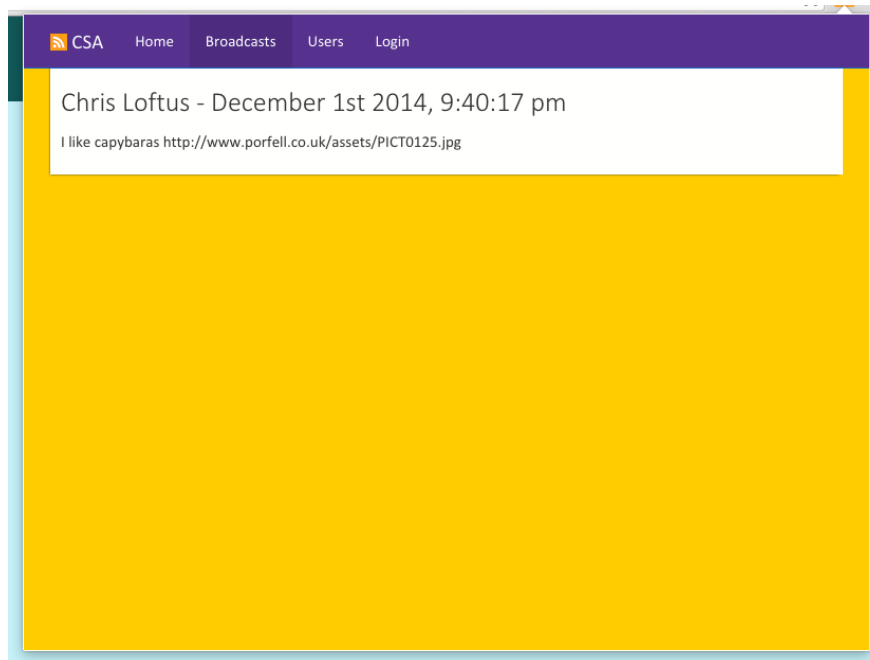


Figure 3: This is what the populated broadcast screen looks like.

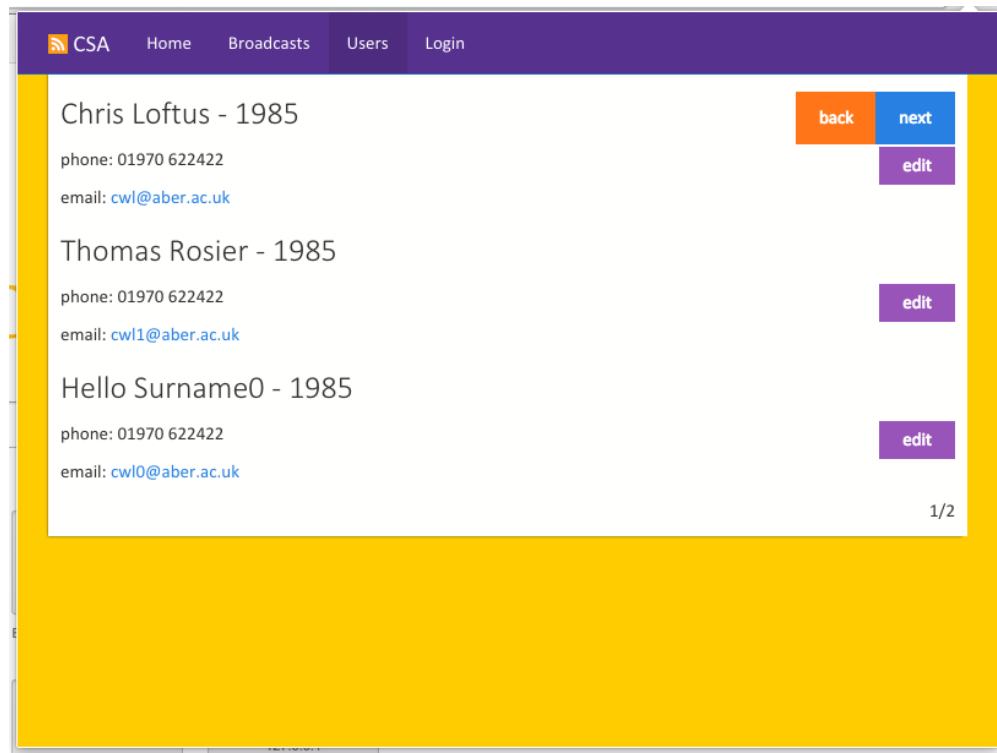


Figure 4: Here we have a screenshot of the user's screen.

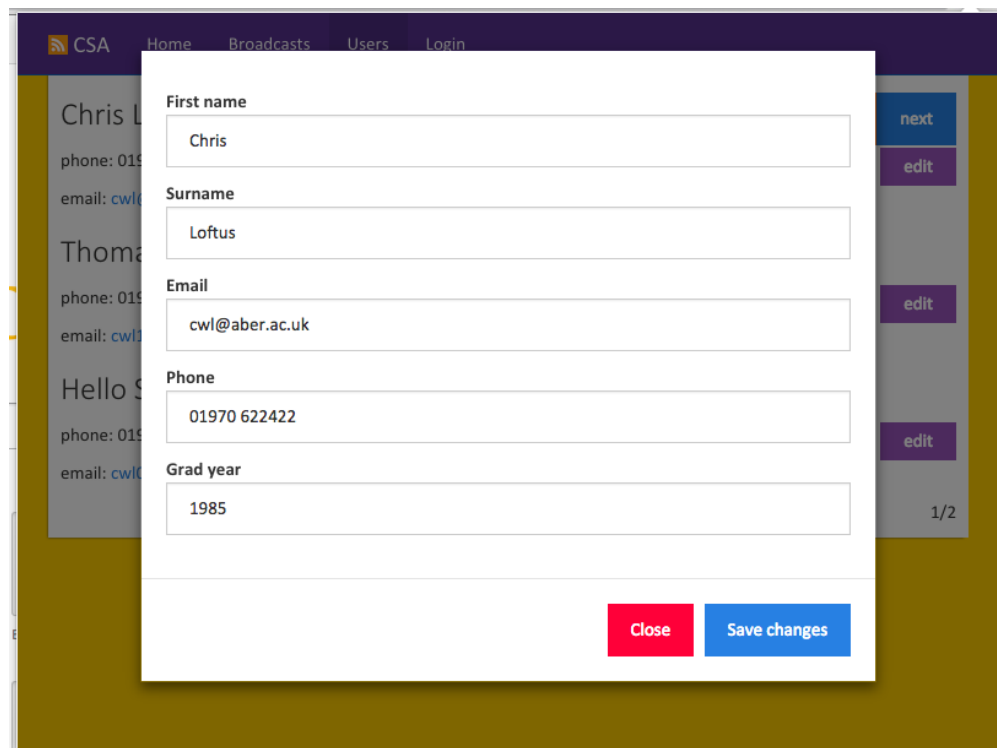
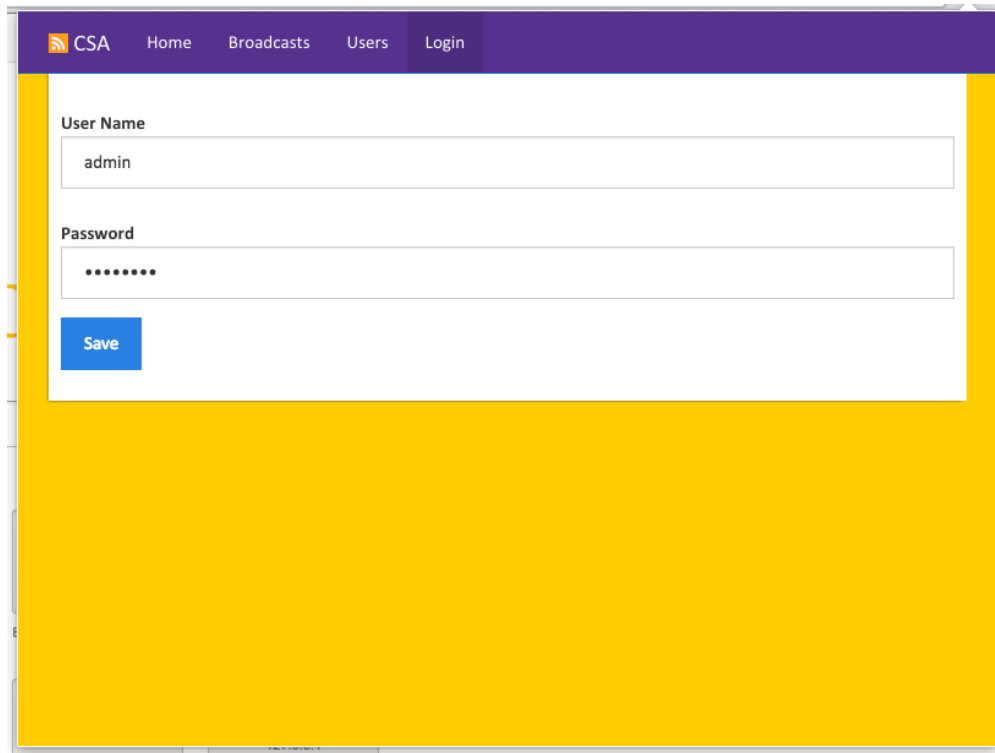
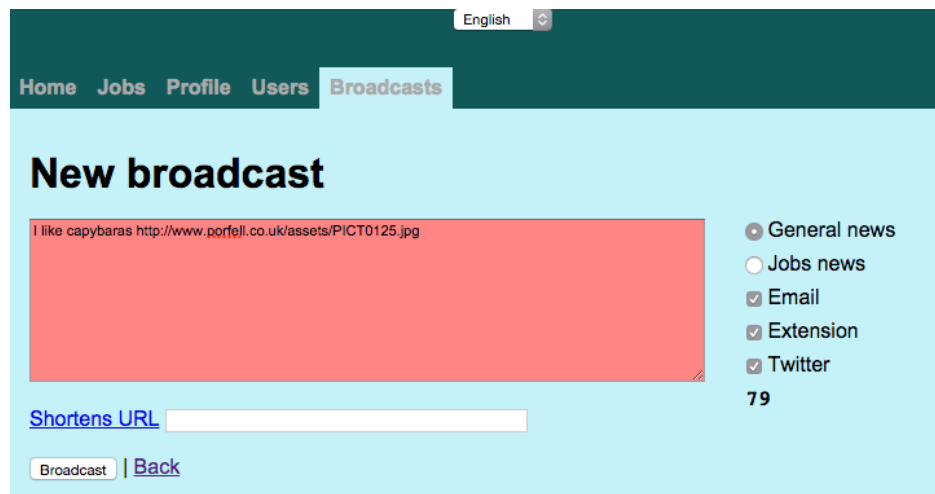


Figure 5: Above we have a screenshot showing the appearance of the modal window we used to edit the user information.



The screenshot shows a web browser window with a purple header bar containing the CSA logo and navigation links: Home, Broadcasts, Users, and Login. The main content area has a yellow background. A white login form is centered, featuring a 'User Name' field with the text 'admin', a 'Password' field with masked characters, and a blue 'Save' button.

Figure 6: This is a screenshot of the login screen.



The screenshot shows a web browser window with a dark teal header bar containing a language dropdown set to 'English' and navigation links: Home, Jobs, Profile, Users, and Broadcasts. The main content area has a light blue background. A 'New broadcast' section includes a red rectangular area for a broadcast image, a 'Shortens URL' field, and a 'Broadcast' button. To the right, a list of destinations is shown: General news (selected), Jobs news, Email (checked), Extension (checked), and Twitter (checked). A count of '79' is displayed below the list.

Figure 7: Creating a broadcast within the CSA application. Note: The destination list differs from the standard CSA application.

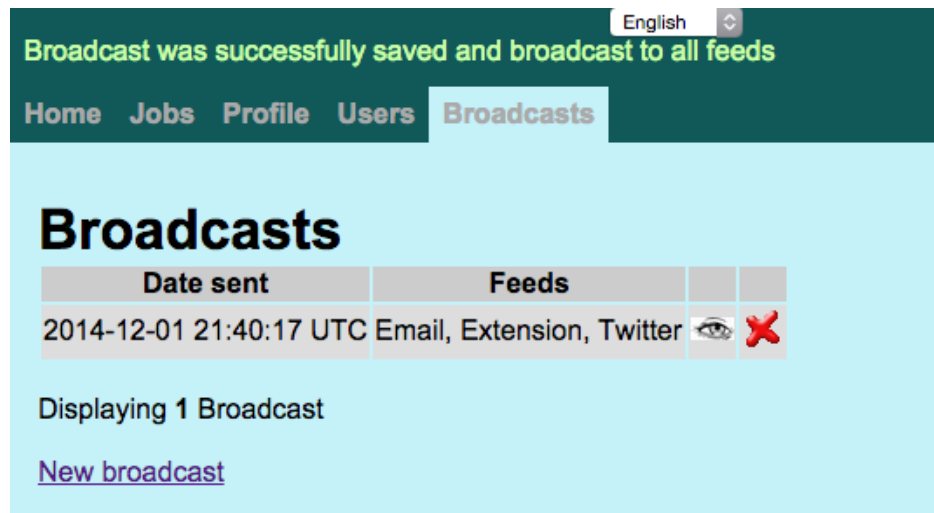


Figure 8: Here is the confirmation that a new broadcast has been created.

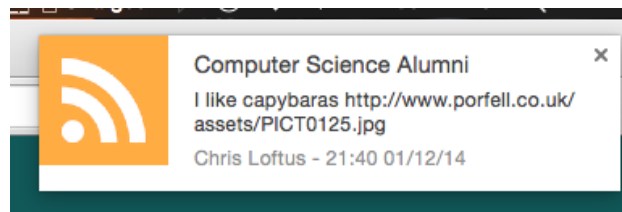


Figure 9: Here is a Chrome notification showing the latest broadcast.

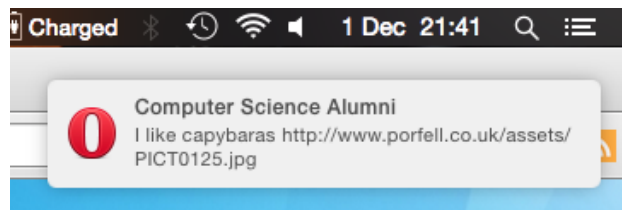


Figure 10: Same again but for Opera this time just to show multi platform support.



Figure 11: Proof that the broadcasts still get sent to Twitter.

CS-Alumni Rails Application

New Additions

New Database View

New RESTful interface

Modifications

Changes to Seeds

Changes to Broadcasts

Changes to Security

Changes to REST implementation

Changes to Users

Communication Between Applications

RESTful Web Interfaces

Data Flow Diagrams

Application Testing

JUnit Tests

Background.js – Unit Tests.

The screenshot displays the 'CSA Extension - Unit Testing' interface. At the top, there are three checkboxes: 'Hide passed tests', 'Check for Globals', and 'No try-catch'. Below this, a blue bar indicates the browser environment: 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.71 Safari/537.36'. The main area shows the test results: 'Tests completed in 41 milliseconds. 3 assertions of 3 passed, 0 failed.' Below this, a table lists three tests:

1. Background pulls from REST correctly and stores it. (1)	Rerun	19 ms
2. Background Initiates Correctly. (1)	Rerun	2 ms
3. REST generates auth token correctly. (1)	Rerun	1 ms

At the bottom, there is a link labeled 'Back to Testing'.

Figure 12: This is the unit tests for the background.js module for the browser extension.

Browser Info – Unit Tests.

The screenshot displays the 'CSA Extension - Unit Testing' interface. At the top, there are three checkboxes: 'Hide passed tests', 'Check for Globals', and 'No try-catch'. Below this, a blue bar indicates the browser environment: 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.71 Safari/537.36'. The main area shows the test results: 'Tests completed in 30 milliseconds. 3 assertions of 3 passed, 0 failed.' Below this, a table lists three tests:

1. Check Browser Name (1)	Rerun	2 ms
2. Check Browser Minor Version (1)	Rerun	0 ms
3. Check Browser Major Version (1)	Rerun	0 ms

At the bottom, there is a link labeled 'Back to Testing'.

Figure 13: This images shows the browserinfo.js module completing its tests successfully.

Local Storage – Unit Tests.

CSA Extension - Unit Testing

☐ Hide passed tests ☐ Check for Globals ☐ No try-catch

Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.71 Safari/537.36

Tests completed in 27 milliseconds.
3 assertions of 3 passed, 0 failed.

1. Load Nothing (1)	Rerun	3 ms
2. Local Storage Save. (1)	Rerun	2 ms
3. Local Storage Reload. (1)	Rerun	0 ms

[Back to Testing](#)

Figure 14: This shows the LocalStorage.js module passing all its unit tests.

Rest Tool Kit– Unit Tests.

CSA Extension - Unit Testing

☐ Hide passed tests ☐ Check for Globals ☐ No try-catch

Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.71 Safari/537.36

Tests completed in 70 milliseconds.
7 assertions of 7 passed, 0 failed.

1. Auth Token is built correctly. (1)	Rerun	2 ms
2. Rest Get Object (1)	Rerun	20 ms
3. Authentication succeeded. (1)	Rerun	12 ms
4. Build Auth Token (1)	Rerun	1 ms
5. Authentication clear succeeded. (2)	Rerun	19 ms
6. Update user record via POST (1)	Rerun	2 ms

[Back to Testing](#)

Figure 15: Above we see the RestToolKit.js module successfully completing all its unit tests.

Utils.js – Unit Tests.

CSA Extension - Unit Testing

☐ Hide passed tests ☐ Check for Globals ☐ No try-catch

Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.71 Safari/537.36

Tests completed in 19 milliseconds.
2 assertions of 2 passed, 0 failed.

1. **Diffed Array (1)** [Rerun](#)2 ms

2. **Reverse Diffed Array (1)** [Rerun](#)1 ms

[Back to Testing](#)

Figure 16: Included above we see the Utils.js module efficiently complete all of its unit tests.

Technologies

CS-Alumni Chrome Browser Extension

Javascript

Chrome Extensions API

Bootstrap

jQuery

Moment.js

Alertify.js

Qunit

CS-Alumni Rails Application

Ruby

Ruby on rails

Qunit

Communication Between Applications

RESTful Interfaces

Cors

Basic Authentication

Evaluation

Something Extra



Figure 17: Above is a image of the Pebble smartwatch running the CSA application.



Figure 18: This is a screenshot taken from the Pebble smartwatch.

Attributions

References

- [1] Mark McDonnell. *SQLite and ActiveRecord*. Oct. 2013. URL: <http://www.integralist.co.uk/posts/sqlite-and-activerecord/>.
- [2] Daniel Morrison. *Building Awesome Rails APIs: Part 1*. June 2013. URL: <http://collectiveidea.com/blog/archives/2013/06/13/building-awesome-rails-apis-part-1/>.