Victoria Do
Eric Fonseca
Andrew Wilson
Nathaniel Bradbury
Danny Hsiung

CSE 134B Ask-Penny Continuity Report

State of the application

We have come to the conclusion of our work on the Ask-Penny Numismatist web application. The application is in a functional state with a variety of features that are sure to attract coin collectors. As a team we have decided that although our app provides sufficient functionality we would not advise releasing it to the public quite yet. There are some features we would like to be implemented that would ensure a better reception from the market. The new features would serve to increase the usability and security of our application. We are confident that once our application is equipped with the extra features it will be a success.

Current Features

- Up to date spot price of gold, silver, and platinum
- Graphical representation of change in user's value over past month
- Separate screens to evaluate gold, silver, and platinum independently
- User accounts to track inventory across sessions and devices
- Can easily modify their inventory with responsive CRUD support

Future Objectives

- **More ways to handle users**: Currently, the only way to use our app is to login through Facebook, which inherently limits our user base to only people with Facebook accounts. Adding new ways to sign up with our app, such as Google account login or implementing our own in-house login system, would not hinder potential users from signing up and using our app.
- **Photo upload**: Currently, our app is able to show a photo for each coin, but there is no way to upload a photo, so all coins' photos are blank. Adding a photo upload feature so that users can see their coins would improve the user experience.
- **Easier coin management**: Currently, the only way to delete coins is to click on each individual coin, go to its edit/delete page, and then click on the delete button. If a user has hundreds of coins that they want to delete at once, the user will not be able to delete them in an efficient manner. Adding the capability to click a checkbox on each (or all) coins from the My Stack page and mass deleting all checked coins would greatly increase the usability of the app.

Specific Technical Challenges

Users

We used the Facebook SDK as our login method of choice. Facebook assigned each user with a unique identifying string which we stored into the Parse database when associating them with their items.

Instead of using cookies, we decided to use the global sessionStorage variable to track whether a user is logged in. This means that every time the user closed the tab or window, they were "logged out". While not practical for a real life implementation, it was useful in debugging our application.

Code Design and Architecture

We wanted to organize our code in a relatively maintainable way. We chose BackboneJS as our underlying MVC framework and RequireJS to separate each distinct module. Backbone was a major challenge because it was unfamiliar to most of us.

Drawing Graphs

If we were to do this assignment again, we might choose D3.js for its flexibility and its ability to rapidly render changes in data. However, we chose CanvasJS for this assignment. Processing the large amounts of data into an acceptable format was a challenge in itself. We found there was no way around redrawing most of the graph.

Database queries between modules

One major problem was our need to fetch collections of items from different areas of the application. Because we used RequireJS, things that we fetched before might not be readily available once we change context. One (ugly) solution might be to make everything global, but we wanted to try alternatives before we resort to that. Another solution is just to query every time we needed something, which would cause slowdown but would not affect the functionality.

In the end, we found that some things could safely be cached. Things that could be cached only needed to be queried once, and then saved to storage. We also had a few application level variables for things that could not be safely cached this way (user collections, namely). We usually only need a new query for returning from the dashboard page, so we could safely use old data in areas like rendering the total price on mobile.

Analytics

User Analytics

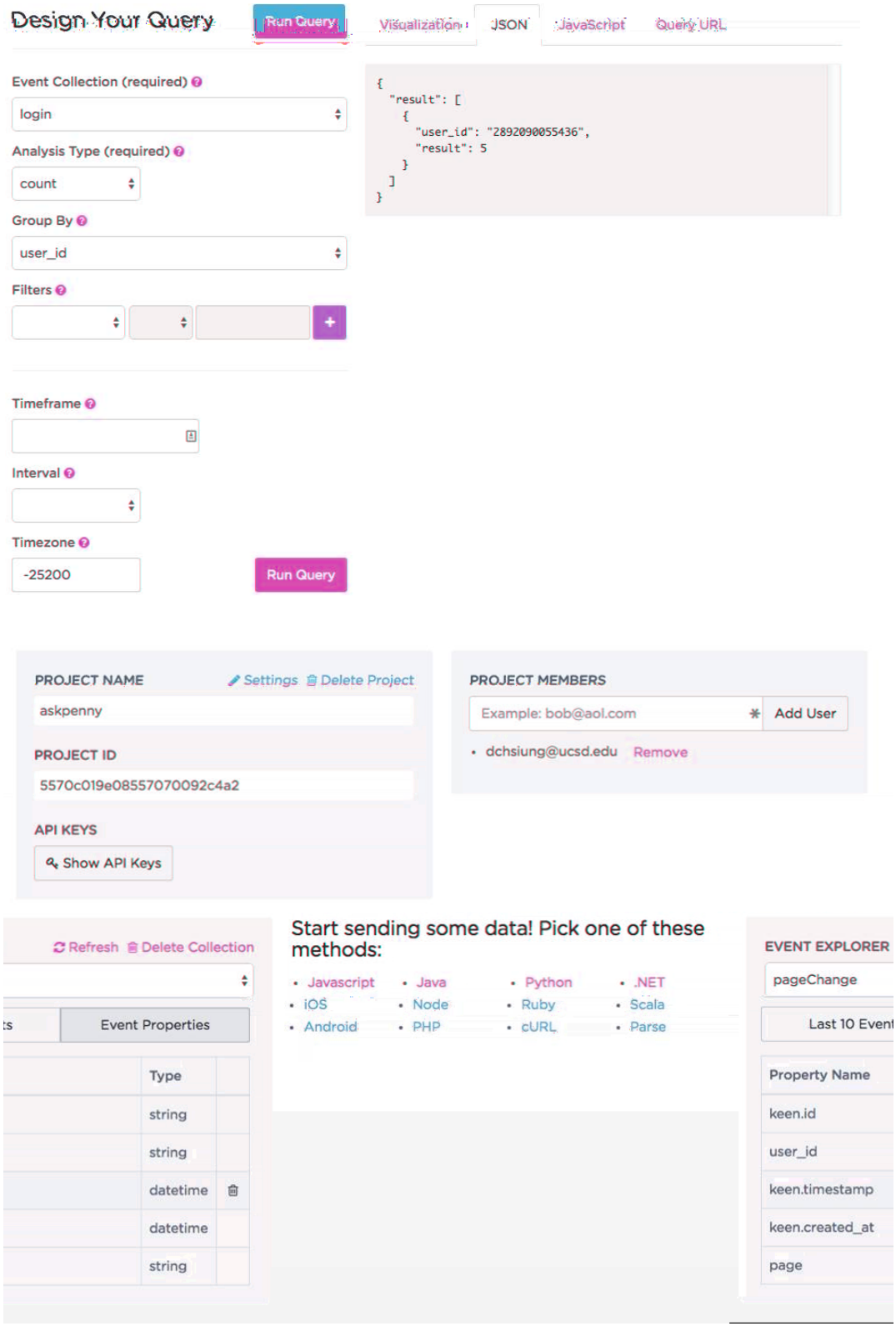
We used KeenJS to track user analytics. Since we built a single-page application, this was slightly trickier than just adding a script to the page.

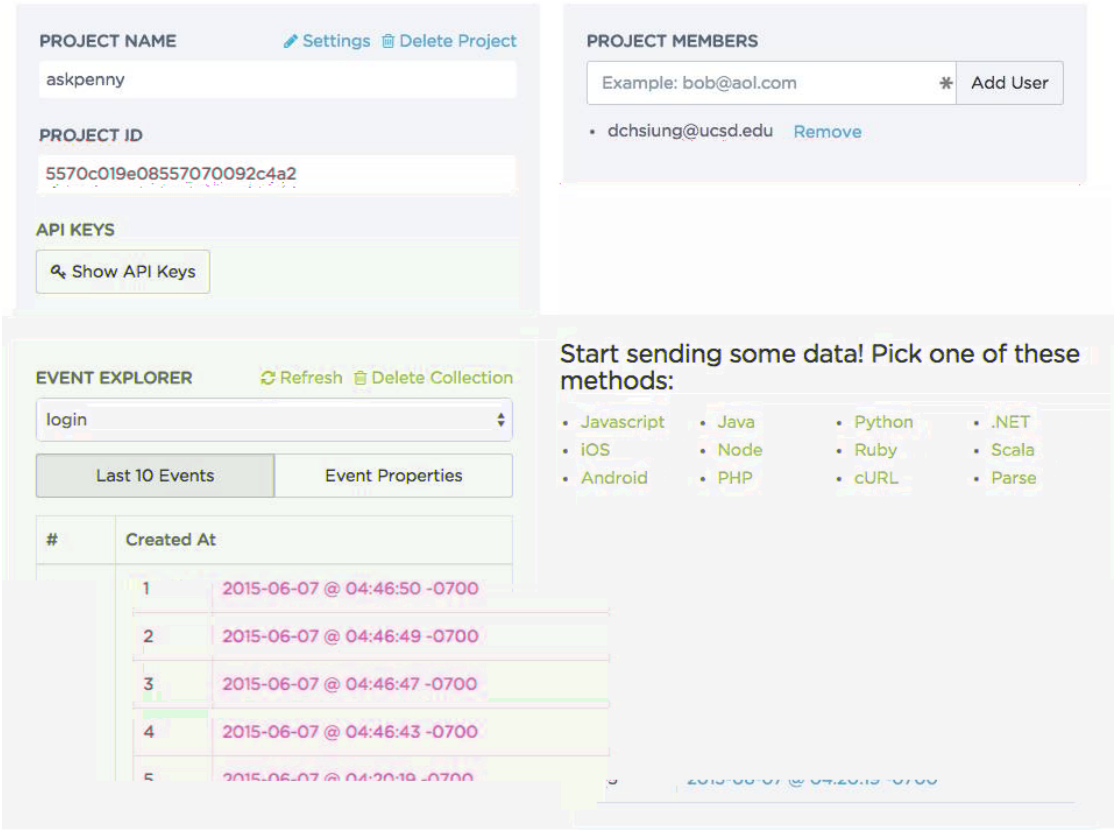
One of the major parts of our analytics includes measuring the amount of time the user spends on the app. This is done by firing a Keen event (any metric you want to measure) when the Facebook authentication processes and the user logs in. This single event is

enough for a variety of purposes. First and foremost, it will tell us how long the user spends on the app. Secondly, since the timestamp is associated with the sessionStorage.uid, a simple Keen query that counts how many distinct user ids there are will allow us to get the bounce rate of users. What time of the day they generally use the app and how many times a day they use their app can also be gleaned from a simple Keen query which can tell us about the user's behavior.

Another portion of the analytics deals with seeing how the users migrate from each of the precious metals dashboards. This tell us how they move from each page to page, which tells us a lot about how a user prioritizes their session. Linked with session time, it can also tell us about the users that bounce and which pages caused them to bounce. Over time, trends may be noticed which will allow us to optimize our application. Long-term users can have tailored advertisements or information based on their usage patterns regarding certain precious metals.

In the screenshots provided, I demonstrate a simple count query that counts the number of times a user as logged in. Since I am on my local machine, and have logged in 5 times, that is the result I got. I also show how we can use a simple query to query the page changes as well as the associated time stamp given within the page change.

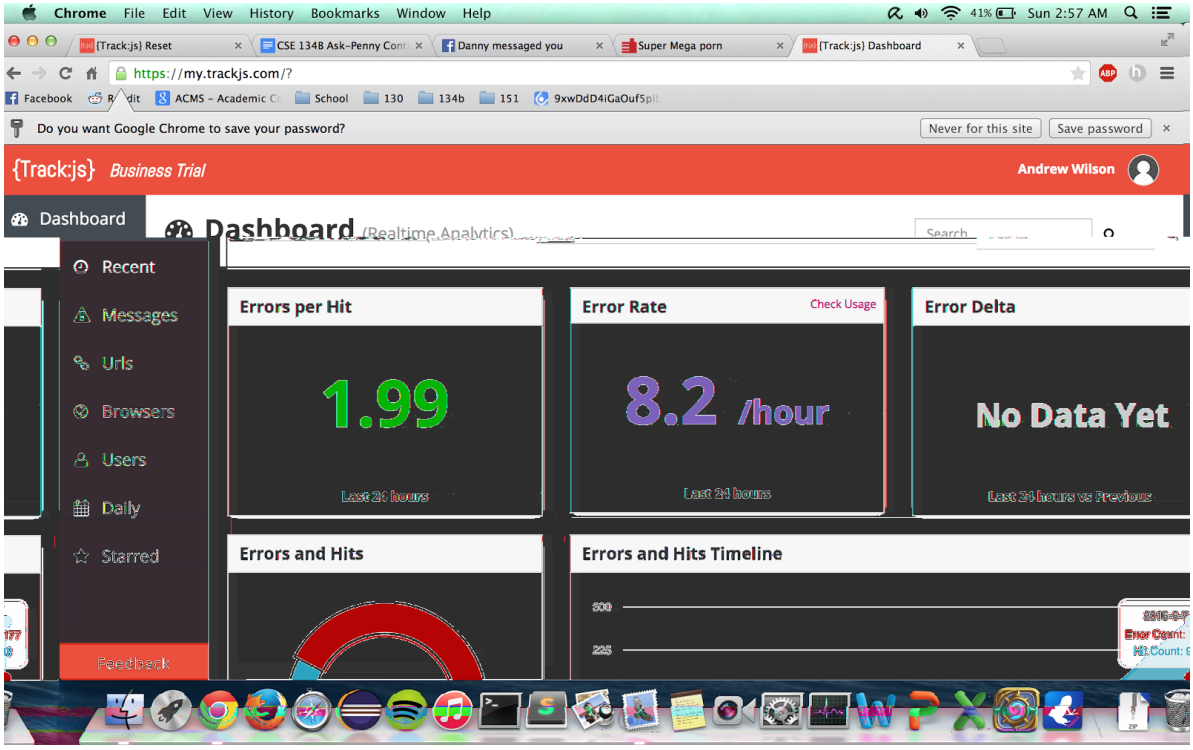

Design Your Query
Run Query
Visualization
JSON
JavaScript
Query URL
Event Collection (required)
login
Analysis Type (required)
count
Group By
user_id
Filters
Timeframe
Interval
Timezone
-25200
Run Query
{
"result": [
{
"user_id": "2892090055436",
"result": 5
}
]
}
PROJECT NAME
Settings
Delete Project
askpenny
PROJECT ID
5570c019e08557070092c4a2
API KEYS
Show API Keys
PROJECT MEMBERS
Example: bob@aol.com
Add User
dchsiung@ucsd.edu
Remove
Refresh
Delete Collection
Event Properties
Type
string
string
datetime
datetime
string
Start sending some data! Pick one of these methods:
Javascript
Java
Python
.NET
iOS
Node
Ruby
Scala
Android
PHP
cURL
Parse
EVENT EXPLORER
pageChange
Last 10 Event
Property Name
keen.id
user_id
keen.timestamp
keen.created_at
page

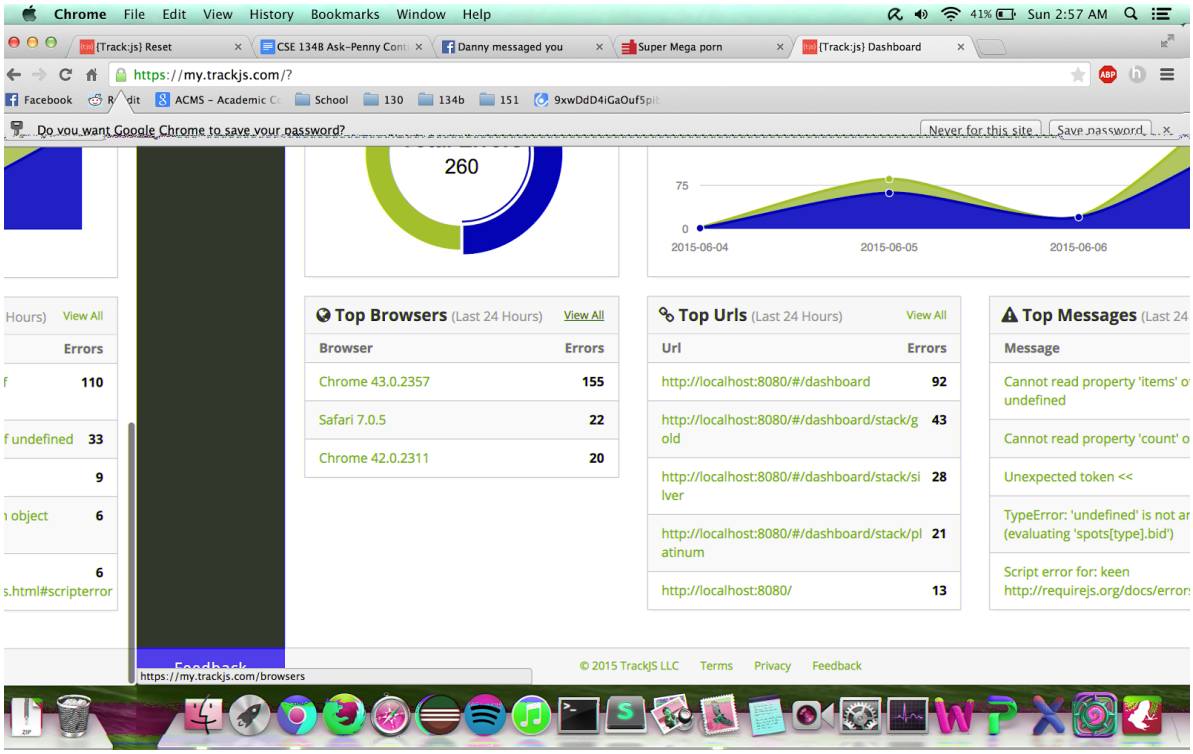

PROJECT NAME
Settings
Delete Project
askpenny
PROJECT ID
5570c019e08557070092c4a2
API KEYS
Show API Keys
PROJECT MEMBERS
Example: bob@aol.com
Add User
dchsiung@ucsd.edu
Remove
EVENT EXPLORER
Refresh
Delete Collection
login
Last 10 Events
Event Properties
#
Created At
1
2015-06-07 @ 04:46:50 -0700
2
2015-06-07 @ 04:46:49 -0700
3
2015-06-07 @ 04:46:47 -0700
4
2015-06-07 @ 04:46:43 -0700
Start sending some data! Pick one of these methods:
Javascript
Java
Python
.NET
iOS
Node
Ruby
Scala
Android
PHP
cURL
Parse

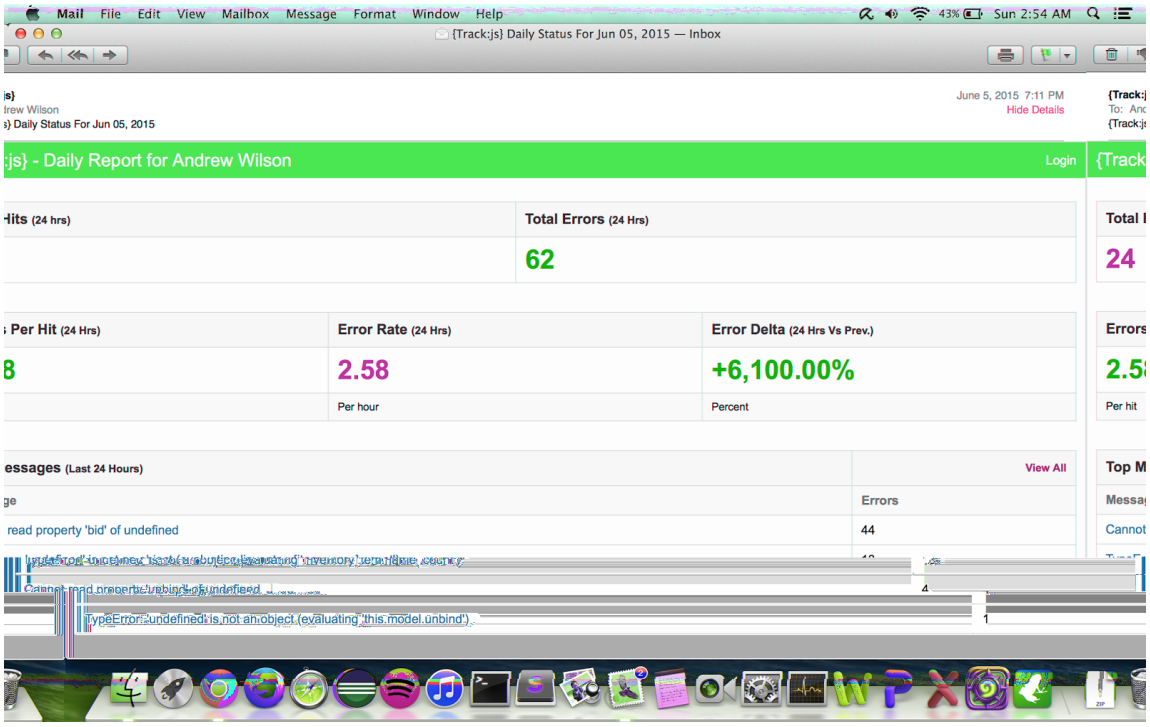
Error Tracking

We integrated TrackJS, a client-side error and event tracker, into our application. It then collects these errors and displays in them in a dashboard and sends periodic emails containing error digests.

Any exceptions or instances of “console.error” from JavaScript will be displayed on the dashboard.




Chrome File Edit View History Bookmarks Window Help
{Track:js} Reset
CSE 134B Ask-Penny Cont
Danny messaged you
Super Mega porn
{Track:js} Dashboard
https://my.trackjs.com/?
Do you want Google Chrome to save your password?
Never for this site
Save password
260
75
0
2015-06-04
2015-06-05
2015-06-06
Top Browsers (Last 24 Hours)
View All
Browser
Errors
Chrome 43.0.2357
155
Safari 7.0.5
22
Chrome 42.0.2311
20
Top Urls (Last 24 Hours)
View All
Url
Errors
http://localhost:8080/#/dashboard
92
http://localhost:8080/#/dashboard/stack/gold
43
http://localhost:8080/#/dashboard/stack/silver
28
http://localhost:8080/#/dashboard/stack/platinum
21
http://localhost:8080/
13
Top Messages (Last 24
Message
Cannot read property 'items' of undefined
Cannot read property 'count' o
Unexpected token <<
TypeError: 'undefined' is not ar (evaluating 'spots[type].bid')
Script error for: keen http://requirejs.org/docs/error
© 2015 TrackJS LLC
Terms
Privacy
Feedback
https://my.trackjs.com/browsers


Mail File Edit View Mailbox Message Format Window Help
{Track:js} Daily Status For Jun 05, 2015 — Inbox
June 5, 2015 7:11 PM
Hide Details
Daily Status For Jun 05, 2015
{Track:js} - Daily Report for Andrew Wilson
Login
Hits (24 hrs)
Total Errors (24 Hrs)
62
24
Per Hit (24 Hrs)
Error Rate (24 Hrs)
2.58
Per hour
Error Delta (24 Hrs Vs Prev.)
+6,100.00%
Percent
Errors
Per hit
View All
Errors
read property 'bid' of undefined
44
Cannot read property 'unbind' of undefined
TypeError: 'undefined' is not an object (evaluating 'this.model.unbind')
1

Technical Specifications

Development Environment

Requires Ruby and npm installed.

1. Clone the repository.
2. **npm install** to retrieve all the required dependencies.
3. **npm start** to start the webserver. Navigate to localhost:8080 to view the site.
4. In another Command Line tab, type **grunt** to start the task runner. Keep this running in the background as you make changes.

Frameworks and Libraries

BackboneJS
Front-end framework used for page routing in a single page app architecture and for organizing views. Backbone models are used to connect with the Parse back end and populate fields.

Underscore
Backbone dependency. Underscore templates are used for rendering partial views with Backbone's built-in router.

RequireJS
Loads JS files asynchronously**.** Useful for separating Backbone views from the actual HTML templates.

RequireJS/text.js and **tpl.js**
RequireJS plugin used to integrate Underscore template files.

JQuery
Used for the CanvasJS plugin, Bootstrap, and general DOM manipulation.

Bootstrap
Provided the responsive grid system, as well as related utility classes.

Keen

Used for Analytics.

Developmental Tools

NodeJS
Used as a platform for our other development tools, such as Grunt.

Grunt
JavaScript Task runner. We used Grunt for CSS minification, JS uglification, and compiling SASS to CSS. We used a watch task to copy and compile files from the source folder to the distribution folder.

Compass
Ruby-based command-line tool that compiles SASS to CSS.

Parse
Our database of choice. We were able to query Parse from the client-side to fetch and update items.

Git
This was our version control system of choice.

Third Party Integrations

Facebook
Facebook login is used in lieu of a traditional sign-up process. Bullion and collections are linked to user IDs.

Quandl
Used for bullion data.

Data transfer and management:

1) Quandl: We are currently using Quandl to fetch the most up to date spot prices of gold, silver, and platinum from the Wall Street Journal's database. As soon as the app is started up, it will query Quandl for the spot prices for the last 30 days. We currently have a free account which is sufficient for our testing environment. Given the how sensitive and volatile spot prices can be we could fetch our data from a premium professional grade database if we see that our users' are interested.

2) Parse: We used Parse to store the data for the various coins that users can add to their portfolio. Inside Parse, coins are able to identify themselves with users by having a userID field, which is filled in during the creation of the coin. The userID is directly linked to the Facebook account that the user signed up with. When populating a user's bullion stack (gold, silver, or platinum), the app queries Parse for all (gold, silver, or platinum) coins with the currently logged in userID. From there, users are able to click on their coins to edit its details, such as purchase date and quantity, and these changes are saved into Parse without creating a new database entry. When users delete coins from their stack, they are also deleted from the Parse database. Once we release our application we will have to buy a subscription to account for the increase in requests.