Nanotwitter 0.3

What we have done in nanotwitter 0.3

1) We have made a good UI interface, it has a navigation bar, kind of like the real twitter.
2) We implemented the methods and web APIs to post tweets and search for tweets. After comparing with Nanotwitter 0.2 Nanotwitter0.3, I changed the way to query the user' s home page time line a little. I think it may result in a little performance improvement, theoretically. In v0.2, I used a three table join to do this, while in this version, I split this query into two steps. The first is to get all the following users' id(add the owner' s id into it) and do a conditional query based on that. In this way, I think, we can cache the following users' IDs and query the home time line with access to only one table(tweet).
3) We implemented the API to post tweets.
4) We refactored the code structure carefully to keep code decoupled and logic clear.

What are the remaining problems.

1) Web APIs always return in JSON, while we need some service to do logics like redirection and so on. Except for the response, their logic is very similar, but we can' t find a way to combine them.
2) Every time a request which needs authentication comes, the code will invoke

method: authenticate to check the authority. We think these lines of codes are duplicated and want to eliminate them by using Rack::Auth::Basic, we didn't use it before.

3) We think redirection this a waste of resources. Since it has to send another HTTP request from browser and query the home time line again(which is the bottle neck of this application). So maybe in the future, we will make the front-end to realize data-driven by using some frameworks like Backbone.js, so that it will make fully use of the web APIs and re render the page automatically when the data models get updated.