6.170 Project 2: Fritter

For the second project in 6.170, a Twitter clone, Fritter, was assigned. In this implementation, the website has backend persistence and frontend dynamics. The app is built with Node.js, Express and MongoDB. The implementation also uses EJS, jQuery, and Mongoose. The application is divided into separate sections of functionality. On Express initialization, a set of folders are created and these create an easy to follow division. From here, a /data folder was added to hold the Mongoose schemas and models. The /routes folder contains routing divided by models they manipulated. Views were divided by main views in the root folder and an additional sub folder of partials to be used repeatedly.

Programming

Design

The main design parts follow directly from the models. The user model maintained the criteria for each user and was used in templating the views. The Tweet model maintained the structure of each tweet. From here, each of the routes were created to allow simplicity between displaying the models and editing the database. The three main views, home.ejs, index.ejs and login.ejs, all follow from displaying logged in user experience, the landing site and the login page.

The main challenges that I experienced were how to appropriately create a database model that would allow for reuse later. I had the option of using Monk or Mongoose to implement this. I had no experience with either and was not sure what the benefits or constraints would be for using either. In one of the lectures, Prof. Jackson spoke to the usefulness of Mongoose and I decided to heed his advice and use it. As it turns out, Mongoose lent itself well to this and allowing me flexibility into being able to quickly add new schemas and models to the app.

Another choice that I had to make was the decision of using Heroku or Openshift. The class materials said that Openshift would be supported but I had experience with deploying Rail applications to Heroku. I could learn another useful cloud deployment platform or use what I had already learned. I ultimately chose Openshift to gain more exposure, but I also got the added benefit of finding an online platform that I enjoyed using. I like the use of cartridges and the ability to add any as necessary.

Highlights

Using Mongoose, the programming required use of callbacks because of the asynchronous nature of the module. This allowed me to display use of callbacks throughout the routing logic that manipulated the database.

One of the biggest highlights was the use of Mongoose and using the lecture notes from “Using a Document Database” to maintain and simplify use with MongoDB. In the /data folder, the schemas and models for tweets and users are held in separate modular folders rather than one large schema and model file. This helped make it easy relating Mongo collections and will help in Project 2.2 to help and expand the functionality from here.

A required task was the creation of user accounts. Here the user information is stored in the database simply as strings (which isn’t secure, but wasn’t a requirement of this project). The User model requires a username, name and password. Each username is unique and cannot be reused once it has been taken in the database. This prevents others from posting tweets under someone’s username.

Another interesting functionality that was required and implemented was the use of sessions. Each user is created a session when they log into Fritter and it is maintained in MongoDB using the connect-mongo module. Once the session is created, it will persist in the server until the user chooses to log out. Once the user logs out, the session is destroyed in the database and the user is required to log back in to post, edit or delete tweets. A piece of middleware was added to check the authentication of the user. This middleware can be found in the /util folder. The middleware checks if the user that is logged in can access functionality or routes. If he/she is logged in properly, the site works well. If the user is not logged in, the user is constantly redirected to the root page to either log in or create an account.