A Stock Tracking Website With Social Networking Features

A Senior Project
presented to
the Faculty of the Computer Science Department
of Charleston Southern University

By
C. Kyle Flynn

Statement Of Purpose

This project has a number of purposes for myself. First and the most personal one is I wanted to prove to myself that I could development, execute a design, and implementation of a full stack website. This is not just for a grade or a requirement. This was about setting a goal and accomplishing it.

Now to the purpose of the project itself. I believe deeply that economics and understanding financial intermediaries like the stock market are so important for every individual in a healthy society. A problem can be the learning curve can be a difficult to overcome by just reading or hearing about the stock market. It is imperative to follow them and to keep track because it can show how money could potentially grow or be lost. So with this understanding, I wanted to create a website that would allow anyone to follow and watch particular stocks they are curious about and see how they are doing. I also wanted to allow people to see what stocks their friends where following as well. So that is the main two features of my website. A person can sign up with an email and password and track stocks that interest them. If a friend is also signed up they can look them up by their email or name and find out what they are tracking as well. This can give a user the possibility of discovering stocks they might not have known about and learn.

Research And Background

A lot of research and trial and errors went into this websites development. Here are some of the major aspects.

Ruby and ruby on rails:
When I started this project I was not very familiar with either ruby or rails so this project gave me the perfect opportunity to explore and learn. One of my main sources of information was the book Ruby on Rails Tutorial by Michael Hartl. It was a great way to learn the basics and some more advanced features involved with ruby on rails.

Devise: (https://github.com/heartcombo/devise#starting-with-rails)
From devise website

Database Authenticatable: hashes and stores a password in the database to validate the authenticity of a user while signing in. The authentication can be done both through POST requests or HTTP Basic Authentication.

Omniauthable: adds OmniAuth (https://github.com/omniauth/omniauth) support.

Confirmable: sends emails with confirmation instructions and verifies whether an account is already confirmed during sign in.

Recoverable: resets the user password and sends reset instructions.

Registerable: handles signing up users through a registration process, also allowing them to edit and destroy their account.

Rememberable: manages generating and clearing a token for remembering the user from a saved cookie.

Trackable: tracks sign in count, timestamps and IP address.

Timeoutable: expires sessions that have not been active in a specified period of time.

Validatable: provides validations of email and password. It's optional and can be customized, so you're able to define your own validations.

Lockable: locks an account after a specified number of failed sign-in attempts. Can unlock via email or after a specified time period.

IEX Fiance API: (https://github.com/dblock/iex-ruby-client)
A gem when configured allows for the user to pull limited stock information

Website study:
I did a really in depth look at how websites are built. It was mostly about finding things that made a website pop and be interesting

Hardware and Software Required For Project

*Any computer that could run windows 10 would be a more than acceptable computer to develop with
*Devise
*Ruby 2.6.6
*Rails 6.0.2.2
*Heroku
*Github
*Atom IDE
*Bash
*JS/JSON
*HTML
*CSS
*Bootstrap
*IEX Fiance API
*More included in the Gemfile

Test Plan And Results

In regards to testing with this project I took a very direct and frankly poor approach. As I have learned at Charleston Southern University there are not short cuts in learning software development and programming languages. I had to bury myself into the language and topic and experiment and to a degree play. What I mean by direct approach is I just started practicing the ruby language and going through Michael Hartl's book Ruby on Rails tutorial. My test plan was very simple in that each step was like a waterfall and be that learning more about user management or an API I just studied it and coded what I could. With this method I actually got pretty far into a solid prototype which I will link from my github repository. At the time I believed I was doing a solid job of test-driven development.

This approach could not even really be described a testing plan. It was a very flawed approach and it only got me so far. So I restarted the complete project and went through and wrote as much pseudo code for each method that I knew needed to be included. I planned out all of my controllers, models, routes and views. I had to take an agile approach and look over the whole project not just in parts that I learned on the way. In retrospect I should have used the test helpers that are available in rails as well but the process made more sense to myself by testing in the rails console at the time.

To be honest this process was my biggest mistake but once I got a check list of each process that needed to be passed it became a lot more manageable. Some of the major things for testing included having a user be able to sign up, log back in, check stocks, add stocks, search for friends, add friends and view friends stocks. One of the wonder things about rails is a lot of these can be tested in the console or on the fly by pulling up the server to check for errors. It is very user friendly.

Prototype https://github.com/ckyleflynn/stockmarket

Challenges Overcome

A lot of the challenges with this project have already been brought up. The first being pretty much learning ruby and rails from scratch. Obviously this was not my first language so the learning curve was not too sharp but it did slow the development process.

Another obstacle was the testing process and honestly my overall method of trying to complete the project.

Learning how to use a gem API was a little daunting at first because of all the documentation that needs to be understood can be a challenging process.
Finally time management. I think with most projects once complete you can identify areas where time was wasted but I believe that is a learning process and I will get better with time and practice.

Future Enchantments

There are quiet a few things I wish I could have been able to add in the final project. One of the first things was more information on each of the stocks. I actually started and parts of the code are still valuable to see in the project. This would have given a path to show more than just company name, ticker symbol and current price. The problem arose with the free version of the IEX Stock API. There is a lot more company information but it comes at a financial cost. To implement the more in depth information and adding graphing of the information would have just cost too much for this project. Another user page I would have liked to add was a personal notes page. Then the user could write themselves notes or reminders about specific stocks. Also it would have been fun to be able to add a messaging feature between friends. If the user wanted to give a friend a heads up on a stock they could send them a direct message. Other notification features like when a stock reached a particular level the user would be notified would have been a nice addition. Lastly removing devise and building a ground up authentication would be preferred but it offered the best opportunity to have the time to include the features that were included with this project.