Jack Stephenson

SI 206

Final Project

Stockbot Report – Alt Project

1. Goals:

My goals for this project were simple: create a groupme chatbot to report

cryptocurrency information from within a specific investment groupme.

Not wanting to sit around all day running my script in my computers console, I decided to have the script always running in the cloud via Heroku and Flask.

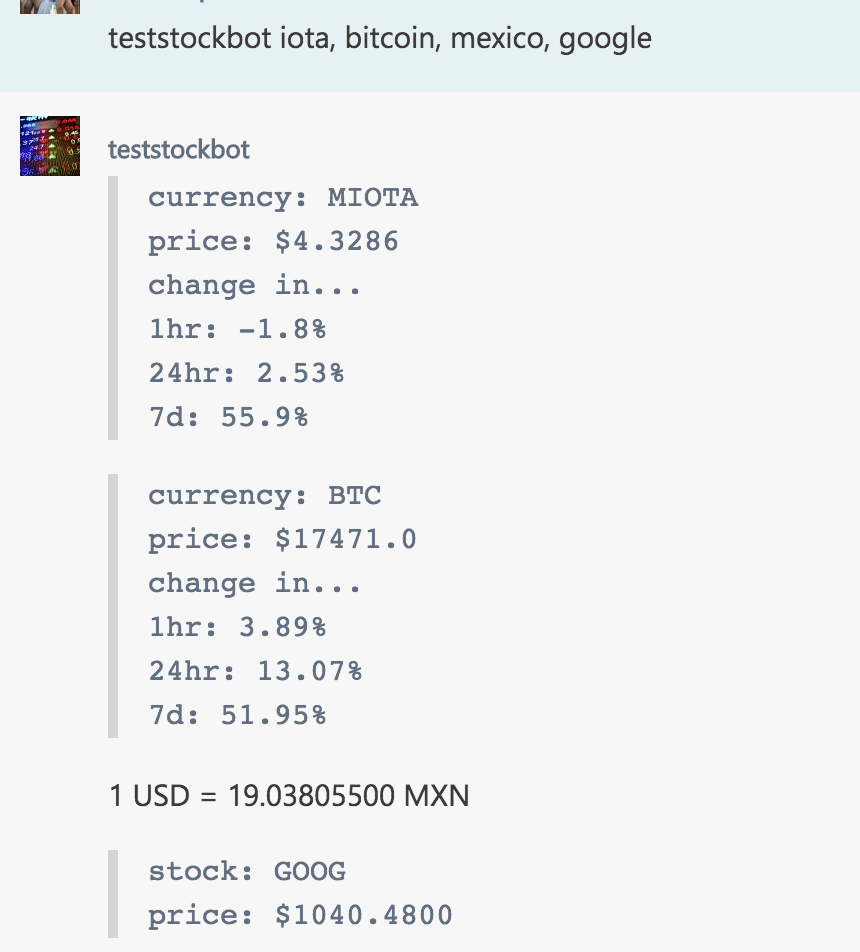
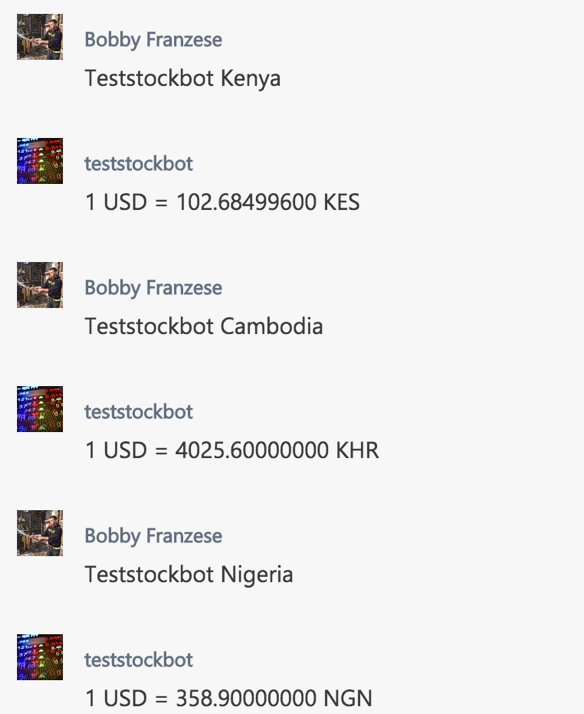
1. Achieved:

Luckily, I achieved my goals and more. In addition to cryptocurrency information, I also could add stock and country currency functionality. I got my heroku server to work, and the chatbot works perfectly. Anybody in the groupme can call the bot at any time, without the need for the script to be running on any physical machine.

1. Problems:

I initially had a caching function for one of my API calls, but learned that heroku doesn’t allow python/flask applications to create files on the dyno that the app runs on—they are instantly purged. Luckily, the fact that my script is always running means that all of the information I need is stored in the data structures in my script. The script only makes new API calls when new information is requested. Aside from this, the only *problems* I encountered were my initial cluelessness about how Heroku worked. I don’t see this as a problem as much as a learning opportunity.

1. Report: I did an alternate project, here are some screenshots of my bot in action



1. Running my code:

My code is always running in the cloud, so to *use my application* all that needs to be done is to join the Testing123 Groupme and start querying the bot. Simply type “teststockbot” (case doesn’t matter) and then the name of the company, ticker symbol, country, or cryptocurrency you want information on. Since my version of heroku isn’t paid, I only have one dyno to work with, meaning the first call after a “sleep period” will take roughly 30 seconds. If you want to make multiple queries, you can add commas and stockbot will parse them and respond in order.

If you want to run my code locally, just run the testrun.py file and enter your query in the console, it will simulate what goes on in the groupme. Note that the apikey used in the foo.py is stored as a heroku config variable (so is the GROUPME\_BOT\_ID). However, the alphavantage api is free and it is very simple to retrieve an API key [here](https://www.alphavantage.co/support/#api-key). If one were to make changes to the actual groupme stockbot, the following steps must be followed:

* Create a virtual environment on your machine and install the dependencies in requirements.txt
* Download the heroku CLI
* Sign in to heroku account, a url will be provided in console.
* Git add, commit, and push app.py, foo.py (can be named anything), requirements.txt, runtime.txt, and Procfile (these tell heroku how to compile the application slug) to the url.
* Viola! The bot is up and running

1. Documentation (see code) –MAYBE
2. Resources:

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Issue Description** | **Location of Resource** | **Result**  **(did it solve the issue?** |
| 11/27 | Basic overview of how to create the bot | http://www.apnorton.com/blog/2017/02/28/How-I-wrote-a-Groupme-Chatbot-in-24-hours/ | No, I had to mess around with mine a ton to get it to work. Some instructions were outdated. |
| 11/27 | Heroku intro | https://devcenter.heroku.com/articles/getting-started-with-python | I started with 0 knowledge of heroku, so yes this worked well |
| 11/30 | Flask intro | http://flask.pocoo.org/docs/0.12/quickstart/ | Didn’t know how to use flask, turns out all I needed was the basic starter code as Heroku did most of the heavy lifting |
| 12/2 | Gunicorn issues | https://devcenter.heroku.com/articles/python-gunicorn | I had some problems with one of the packages necessary for the flask-heroku work, I used this page for troubleshooting |
| 12/5 | Further flask/gunicorn help | https://progblog.io/How-to-deploy-a-Flask-App-to-Heroku/ | Helped me initiate the flask server for local testing/heroku deployment |
| 11/27 | Urllib2 | https://docs.python.org/3/library/urllib.html | Encoding and requests |