Project 1 - Stock Analyst

*Populating a database and using Web APIs*

(teams of 2)

**Introduction**

[Personal finance](https://en.wikipedia.org/wiki/Personal_finance), and equity ownership in the [stock market](https://en.wikipedia.org/wiki/Stock_market) are important concepts dealing with savings and money. [Stocks](https://en.wikipedia.org/wiki/Stock) represent partial ownership in a publicly traded corporation: as a stock price rises, the valuation of the company increases. Many individuals and corporations invest in the stock market, hoping for gains over time that are greater than interest you can get from a bank savings account.

Please read this entire document before starting!

This project has two parts: in the first part, we will write a PHP console application which reads the provided companylist.csv file, which contains all the stocks traded on the New York Stock Exchange ([source](https://www.nasdaq.com/screening/companies-by-industry.aspx?exchange=NYSE)), to populate a database. Stocks are typically identified by a unique identifier called a “ticker symbol”. Save the csv file and your PHP console application file(s) within PHPCode/ass1/console so that you can read and execute it in the Vagrant VM.

In the second part, we write a web application that asks the user to choose a stock ticker symbol or company name, and then returns the last trade [close](https://www.alphavantage.co/documentation/) value (meaning the last trade price that was agreed upon by a buyer and a seller) as well as the latest [headlines](https://developer.yahoo.com/finance/company.html) related to the stock. Write your web application in PHPCode/ass1/public. It is important that your console application is not in your web application’s root folder to prevent [unauthorized](https://www.phptherightway.com/#command_line_interface) running of the console program. The Homestead.yaml file associates the URL https://ass1.test with PHPCode/ass1/public, so the public folder is the web application’s root folder.

Use best coding practices: modularize in functions or classes, use the data access object design pattern for separation of concerns, validate user input.

**Submission**

You will submit your code to the two empty GitLab repositories created for your team. For the console project: git init in the PHPCode/ass1/console folder, and set your git remote to the GitLab url for the console project, and code it there. Code your web application in PHPCode/ass1/public, setting your remote “origin” to the GitLab url for the web project.

You are in charge of the master branch, but please continue to use the protected branch [workflow](https://gitlab.com/dawsoncollege/CSharp411/w2018/git-tutorials/blob/master/protected_branch_workflow.md) that you have used the past two semesters; make sure you set up a process in your partner such that the other member must approve any merge request to master. [Tag](https://git-scm.com/book/en/v2/Git-Basics-Tagging) your final submissions as “final” in both [repositories](https://docs.gitlab.com/ee/university/training/topics/tags.html).

git checkout master

git tag final

git push origin --tags

Make sure that you clearly indicate your **Heroku url** in your GitLab README.md file (see below).

**Console Application**

I recommend that you write your console script to fill the database on Vagrant. Initially, fill the Homestead PostGres table, and once everything is tested, change the datasource to point to the heroku PostGres database and run the script from Vagrant. Do NOT deploy your console application to Heroku. (see below)

**Web Application and Heroku**

You will deploy your web application project to Heroku, a cloud Platform-as-a-Service (PaaS), so that anyone can use your application! Heroku uses a git workflow to deploy; you will only deploy the web application’s repository to Heroku!!

You are not required to use a credit card in this course, so you will get an unverified account with Heroku. This will [limit](https://devcenter.heroku.com/articles/limits#other) you to at most 5 apps for this course.

**Follow these steps in the very start of the project, even if you use Vagrant as a development environment.**

1. **Only one team member:** Open a free Heroku account (one partner only, you will be sharing the account):

Fill in the form at <https://signup.heroku.com/www-pricing-top>

Choose PHP as the primary development language

Wait for the confirmation e-mail (check your junk e-mail)

Sign in with a password

1. On your development system: vagrant up
2. vagrant ssh
3. Download and install the heroku command-line interface. Unfortunately in the labs, you need to do this everytime

# Run this from your terminal.

wget -qO- https://cli-assets.heroku.com/install-ubuntu.sh | sh

Verify the install:

heroku --version

1. heroku login
   * use the email and password you used earlier
2. In Vagrant shell, go to code/ass1/**public** (create the folders if necessary)

* This folder holds your web application, which is the only thing we want to deploy on Heroku for “production”

1. **Only one team member:** initialize a new git repo in the public folder, and set the origin remote.

git init .

git remote add origin *gitlabUrlFromTeacher*

1. **Only one team member:** create a Procfile. A Procfile indicates what processes your application will require. Your Procfile will have a single line to indicate that it is a web process. A shortcut to create the file:

echo web: vendor/bin/heroku-php-nginx > Procfile

1. **Only one team member:** Create the Heroku application. When you create an app, a git remote named **heroku** is associated with your local git repo. Heroku can create a random name for the app, but I recommend that you can pass a parameter to specify your own name

heroku create *mybeststockanalyst*

Note the URL of your application, as well as the git URL. You can always see these at http://dashboard.heroku.com if you log in to the web console.

Remember, a git repository can be associated with more than 1 remote. For this web application, you will have one remote (named origin) which refers to the GitLab repository, and one named heroku which refers to the deployment repository.

1. **Only one team member:** Add your partner as a collaborator to the Heroku application/git repository: go through heroku.com > login > Dashboard > select the app > Access > Add collaborator. Your collaborator will receive an email.
2. **Only one team member:** Create an empty (i.e., {}) composer.json file so that Heroku correctly detects this as a PHP application. A shortcut to create the file:

echo {} > composer.json

1. **Only one team member:** Commit and push your changes in the master branch (note: if needed, first set your name and email first through git config)

git add .

git commit –m “Add starting files”

git push heroku master

git push origin master

1. **The other team member:** You can clone the GitLab repo into your code/ass1/**public**
2. **Only one team member:** Add a database at the free level:

heroku addons:create heroku-postgresql:hobby-dev

Note the name of your store.

Alternatively, you can do this through the heroku.com web app:

Elements – Addons – Heroku Postgres addons – install

1. To connect to your Heroku Postgres database from Vagrant console:

heroku pg:psql

1. To connect from your web application: To find the database url:

heroku config

The DATABASE\_URL consists of:

postgres://user:password@host:port/databasename

Or from the data.heroku.com web application – settings – view credentials

Use this as the dsn (replace the database name, user and password and change the host server)

$dsn = "pgsql:dbname=databasename;host=blahblh.amazonaws.com;port=5432;sslmode=require";

$user=”user”;

$password=”pwd”;

1. Code your application. Regularly commit and push your changes to origin (GitLab) in your working branch. When you have tested on Vagrant and want to deploy to Heroku, you must merge your branch to master, then:

git push heroku master

1. to see your web application once deployed:

<https://yourapplicationname.herokuapp.com>

**Part 1 – Console application - Populate the database**

You have been provided with a csv file ([source](https://www.nasdaq.com/screening/companies-by-industry.aspx?exchange=NYSE)) containing all the stock ticker symbol information with a header row. Write a modularized PHP command-line script (run from Vagrant VM command line, NOT a web application through the browser) that:

* drops the existing table and creates a new table with appropriate columns
* opens the file
* extracts the information: Symbol, Name, Sector and Industry
* writes to the table

Note: you will run from command line by ssh’ing into Vagrant and running the command from the appropriate directory: php yourscriptname.php

Make sure that you modularize for console application with either functions or classes. Store your PDO connection variables in a separate file that you require.

This program will run from Vagrant and populate the database. Initially, populate the Postgres database on Vagrant.

Once you are satisfied, change your PDO connection variables to refer to the Heroku database and populate it. Note that this may take a little time. You will be using 3150 out of the 10000 maximum rows on Heroku.

**Part 2 – Web application**

Once the database table is populated, code a simple web application. Initially, test in Vagrant and when you are satisfied deploy to Heroku. The web application will works as follows:

The home page (**index.php**) asks the user to choose the desired company from a list of ticker symbols - company name. Once the user selects the company present the following information:

1. The last trade closing price
2. The last 5 news items on the company

The last trade closing price: Make a request to the Alpha Vantage [API](https://www.alphavantage.co/documentation/) to get the last trade closing price. You will be using the TIME\_SERIES\_INTRADAY API, with the api key **6MD59T737Q345MPQ**. If you prefer, you can get your [own API key](https://www.alphavantage.co/support) . Your URL will look something like this, with MSFT replaced with your selected ticker symbol:

[https://www.alphavantage.co/query?function=TIME\_SERIES\_INTRADAY&symbol=MSFT&interval=1min&apikey=6MD59T737Q345MPQ](https://www.alphavantage.co/query?function=TIME_SERIES_INTRADAY&symbol=MSFT&interval=1min&apikey=xxx)

You don’t have to use cURL to get the Alpha Vantage json response. Since you are retrieving a single json page with a GET request, your code can use a simple file function:

$results = file\_get\_contents($url);

$json = json\_decode($results, true);

I recommend that you set the second parameter to true so that the [json\_decode](http://php.net/manual/en/function.reset.php) function returns an associative array instead of an object. When using an array, you will be looking at the first element of "Time Series (1min)" element. The key of the first element is a timestamp, which changes and which you will not know; so use the [current](http://php.net/manual/en/function.current.php) function to get the first element of the array without knowing its key. This first element will be an associative array: you want the value associated with the key '4. close'.

The last 5 news items on the company: Yahoo provides an [RSS feed](https://developer.yahoo.com/finance/company.html) which returns an XML document. Your URL will look something like this, with YHOO replaced with your selected ticker symbol:

http://finance.yahoo.com/rss/headline?s=YHOO

Once again, you don’t have to use cURL to get the Yahoo XML response. Since you are retrieving a single XML page with a GET request, your code can use a simple file function:

$results = file\_get\_contents($url);

$xml = new \DOMDocument();//backslash to indicate global namespace

@$xml->loadXML($results);//@ to mute XML errors

Look at the XML format by viewing the source in the browser. Recall that RSS feeds are in XML format, and their basic schema is simply a list of items:

<item>

<description>Summary of the story</description>

<link>URL of the original story</link>

<pubDate>Publication date of the story</pubDate>

<title>Title of the story</title>

</item>

You will display the title, description and publication date of the first 5 items in the feed.

Please think about:

* usability of your web application: make sure the information presented is nice and pretty ☺
* good coding practices: modularize your code, validate user input