

Homework 8: Responsive Web Design

Stock Search using Alpha Vantage & Facebook Mashup
(Bootstrap/JQuery/AJAX/JSON/AngularJS/Cloud Exercise)

1. Objectives

- Become familiar with the AJAX and JSON technologies
- Use a combination of HTML5, Bootstrap, JQuery, Angular, and PHP/Node.js
- Get hands-on experience in Google Cloud App Engine and Amazon Web Services
- Get hands-on experience on how to use Bootstrap to enhance the user experience
- Provide an interface to perform stock search using Alpha Vantage and post details to Facebook

2. Background

2.1 AJAX and JSON

AJAX (Asynchronous JavaScript + XML) incorporates several technologies:

- Standards-based presentation using XHTML and CSS;
- Result display and interaction using the Document Object Model (DOM);
- Data interchange and manipulation using XML and JSON;
- Asynchronous data retrieval using XMLHttpRequest;
- JavaScript binding everything together.

See the class slides at <http://cs-server.usc.edu:45678/slides/ajax.pdf>

JSON, short for JavaScript Object Notation, is a lightweight data interchange format. Its main application is in AJAX web application programming, where it serves as an alternative to the use of the XML format for data exchange between client and server. See the class slides at:

<http://cs-server.usc.edu:45678/slides/JSON1.pdf>

2.2 Bootstrap

Bootstrap is a free collection of tools for creating responsive websites and web applications. It contains HTML and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. To learn more details about Bootstrap please refer to the lecture material on Responsive Web Design (RWD). See the class slides at:

<http://cs-server.usc.edu:45678/slides/Responsive.pdf>

2.3 Facebook

Facebook provides developers with an API called the Facebook Platform. Facebook Connect is the next iteration of Platform, which provides a set of APIs that enable Facebook members to log onto third-party websites, applications and mobile devices with their Facebook identity. While logged in, users can connect with friends via these media and post information and updates to their Facebook profile.

Below are a few links for Facebook Connect:

<https://developers.facebook.com/>

<https://developers.facebook.com/docs/javascript>

2.4 Amazon Web Services (AWS)

AWS is Amazon's implementation of cloud computing. Included in AWS is Amazon Elastic Compute Cloud (EC2), which delivers scalable, pay-as-you-go compute capacity in the cloud, and AWS Elastic Beanstalk, an even easier way to quickly deploy and manage applications in the AWS cloud. You simply upload your application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring. Elastic Beanstalk is built using familiar software stacks such as the Apache HTTP Server, PHP, and Python, Passenger for Ruby, IIS for .NET, and Apache Tomcat for Java.

The Amazon Web Services homepage is available at: <http://aws.amazon.com/>

2.5 Google App Engine (GAE)

Google App Engine applications are easy to create, easy to maintain, and easy to scale as your traffic and data storage needs change. With App Engine, there are no servers to maintain. You simply upload your application and it's ready to go. App Engine applications automatically scale based on incoming traffic. Load balancing, micro services, authorization, SQL and noSQL databases, memcache, traffic splitting, logging, search, versioning, roll out and roll backs, and security scanning are all supported natively and are highly customizable.

To learn more about GAE support for PHP visit this page: <https://cloud.google.com/appengine/docs/php/>

To learn more about GAE support for Node.js visit this page: <https://cloud.google.com/appengine/docs/flexible/Node.js/>

2.6 Angular

Angular is a toolset for building the framework most suited to your application development. It is fully extensible and works well with other libraries. Every feature can be modified or replaced to suit your unique development workflow and feature needs.

For this homework, either AngularJS, Angular 2 or Angular 4 can be used.

To learn more about AngularJS visit this page: <https://angularjs.org/>

To learn more about Angular 2 and Angular 4 visit this page: <https://angular.io/>

2.7 Node.js

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js package ecosystem, npm, is the largest ecosystem of open source libraries in the world.

To learn more about Node.js visit: <https://Node.js.org/en/>

Also, Express.js is strongly recommended. Express.js is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. It is in fact the standard server framework for Node.js.

To learn more about Express.js, visit <http://expressjs.com/>

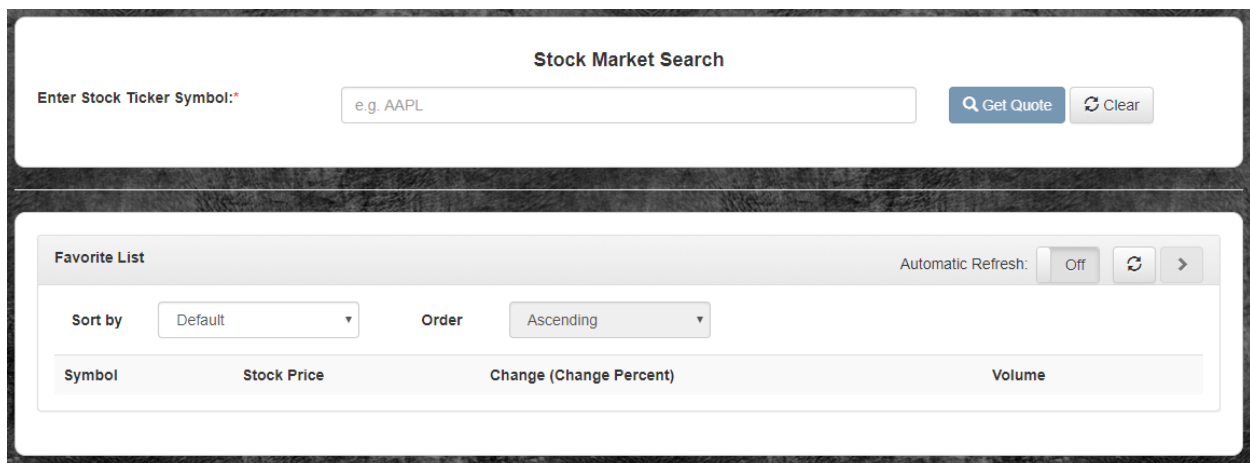
In this document when you see “PHP/Node.js” it means that you can either use a PHP script or a Node.js script; when you see “jQuery/Angular” it means that you can either use a jQuery or Angular function; and when you see GAE/AWS it means that you can either use Google App Engine or Amazon Web Services.

3. High Level Description

Like HW#6, in this exercise you will create a webpage that allows users to search for stock information using the Alpha Vantage API and display the results on the same page below the form.

The difference being, in this homework you will create a PHP/Node.js script to return JSON formatted data to the front-end. The client will parse the JSON data and render it in a nicer-looking responsive UI, using the Bootstrap toolkit.

A user will first open a page as shown below in Figure 1, where he/she can enter the stock ticker symbol, and select from a list of matching stock symbols using “autocomplete.” A quote on a matched stock symbol can be performed. The description of the form is given in section 4.1. Instructions on how to use the API are given in section 5.



The screenshot shows a web application titled "Stock Market Search". At the top, there is a text input field labeled "Enter Stock Ticker Symbol:*" with the placeholder text "e.g. AAPL". To the right of the input field are two buttons: "Get Quote" (with a magnifying glass icon) and "Clear" (with a circular arrow icon). Below the input field is a section titled "Favorite List". This section contains a table with the following headers: "Symbol", "Stock Price", "Change (Change Percent)", and "Volume". Above the table, there are two dropdown menus: "Sort by" (set to "Default") and "Order" (set to "Ascending"). To the right of these dropdowns is a section for "Automatic Refresh" with a toggle switch set to "Off" and a refresh icon.

Figure 1

Once the user has entered some characters in the edit box and selected a matching result from the autocomplete list, he would click on Get Quote, at which point validation must be done to check that the entered data is valid.

Once the validation is successful, the **JQuery/Angular function ajax()** is executed to start an asynchronous transaction with a PHP/Node.js script running on your GAE/AWS, and passing the search form data as parameters of the transaction.

The PHP script you request is based on your HW#6 PHP script. Note that in this exercise you also have the option of using Node.js to rebuild the backend, instead of using the HW#6 PHP script. We will provide **2 extra credits** for using Node.js.

The difference between HW#6 and HW#8 is that this time the PHP script will not return HTML using “echo”, but instead will return JSON data from the API to your search webpage. The webpage must then use JavaScript to extract the needed data from the JSON object and display the results on the same webpage. Description of how to display the results is given in following section.

4. Implementation

4.1 Search Form

4.1.1 Design

You must replicate the form displayed in Figure 1 using a **Bootstrap form**. The form fields are the same as in your HW#6.

The top-level interface consists of the following:

- A form which has an input to enter the stock ticker symbol;
- A result area that displays the results of a quote request or a list of favorite stocks;
- Both sections should be separated graphically as shown in Figure 1;
- The result area should start with an empty favorite list.

The search form has two buttons:

1. **Get Quote** button: On the button being clicked, validations are performed (Refer to 4.1.3). If validations are successful, then an AJAX request is made to your web server (PHP/Node.js on GAE/ AWS) providing it with form data that was entered. If validations fail, appropriate messages must be displayed under the appropriate text box, and an AJAX request should **NOT** be made with invalid data. Note that the “Get Quote” button should be disabled when the input box is empty or contains only spaces.
2. **Clear** button: This button must clear the text field, reset the result area to the favorite list and clear all validation errors if present. The clear operation is implemented as a JavaScript function.

4.1.2 Autocomplete

- A form allows a user to enter a keyword (stock ticker symbol) to retrieve information (quote and indicators information from Alpha Vantage, and news from Alpha News feed). Based on the user input the text box should display a list of all the matching company’s ticker symbols (see Figure 2). The autocomplete JSON data is retrieved from the Markit on Demand API. For an example of calling this API see:
<http://dev.markitondemand.com/MODApis/Api/v2/Lookup/json?input=Apple>
- The autocomplete function should be implemented using AngularJS Material or Angular Material. Refer to Section 7 for more details.

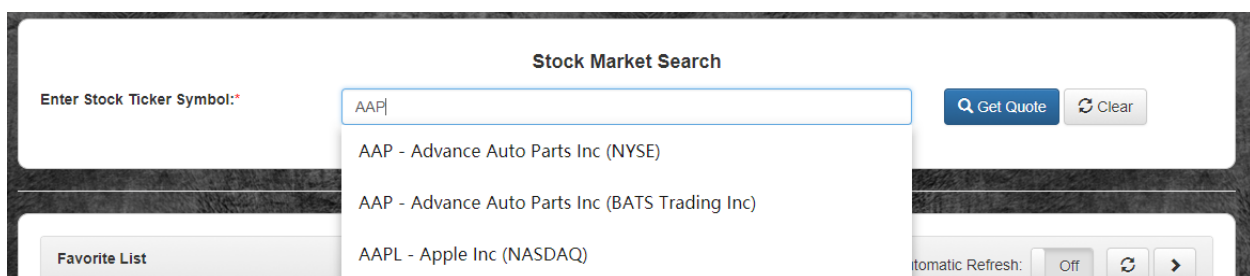


Figure 2

4.1.3 Validation

- The validations that are needed to be implemented in the input query string (stock ticker symbol) are:
 - **Empty Entry** – the border turns red when the input is empty or contains only spaces (see Figure 3)
 - Input error message: show the error message when the input is empty or contains only spaces (see the video)

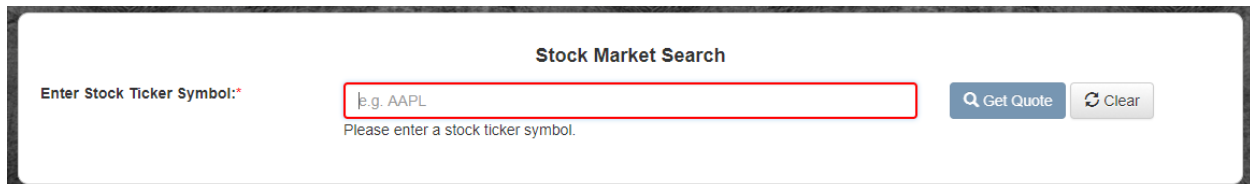
A screenshot of a web form titled "Stock Market Search". On the left, it says "Enter Stock Ticker Symbol:*". In the center is a text input field containing "e.g. AAPL". The input field has a red border, indicating a validation error. Below the input field, there is a small error message: "Please enter a stock ticker symbol." To the right of the input field are two buttons: a blue button with a magnifying glass icon and the text "Get Quote", and a grey button with a circular arrow icon and the text "Clear".

Figure 3

4.1.4 Get Quote Execution

- Once the validation is successful, you should execute an **AJAX transaction** with the **PHP/Node.js script** hosted on GAE/AWS.
- The PHP/Node.js script on GAE/AWS is used to retrieve data from Alpha Vantage. You should pass the **symbol as a parameter** of the transaction when calling the PHP/Node.js script.

For example, if your GAE/AWS service is located at example.appspot.com and the user enters 'AAPL' as the company symbol, then a query of the following type needs to be generated: <http://example.appspot.com/?symbol=AAPL>

- The PHP/Node.js script running on GAE/AWS would perform an API request to Alpha Vantage, **extract the stock details (including quote/volume and all indicators)** of the company symbol, and returns the data in JSON format to your JavaScript program.
- After obtaining the query results from the callback of the AJAX request, the JavaScript program displays the results in an appropriate table in the "result" area of the web page. Also, the successive queries will clear the data from the result area and overwrite it with new data.

4.2 Result Tabs

The result area will include a **sliding mechanism which is implemented with Angular**. Refer to Section 7.6.

- There should be **two sections**, which can be "toggled" using a **sliding mechanism**.
 - The first section should be the Favorite List.
 - The second section should be the Stock Details and charts.

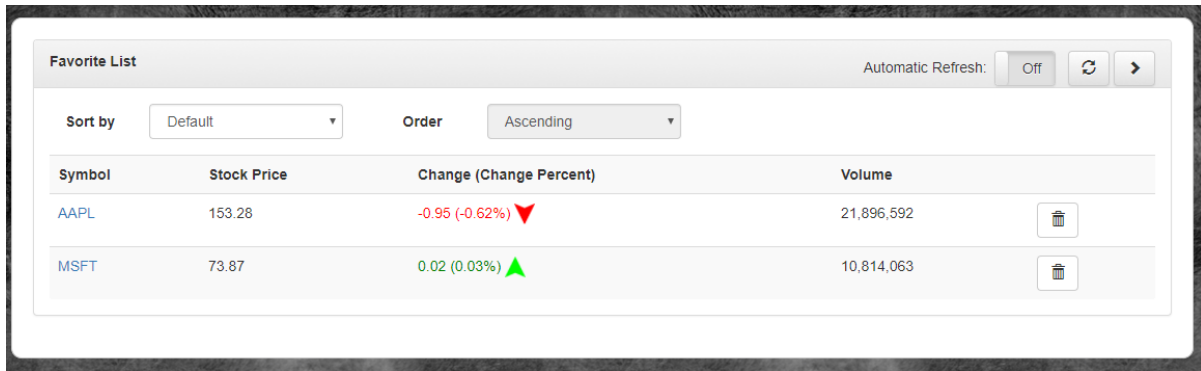


Figure 4

4.2.1 Favorite List Section

The Favorite Section should be designed as per Figure 4.

- The data should be saved and loaded from the **local storage** of the browser. The local storage should contain the list of the favorite stocks.
- A table containing the following information:

| Table Field | Description |
|--------------------------------|--|
| Symbol | Displays the Symbol of the Company. If symbol is clicked the table should switch to stock details section in sliding mechanism |
| Stock Price | Displays the current stock price of the data |
| Change (Change Percent) | Displays the change and the change percent of the current stock. The format should be change (change %) indicator . It should be rounded to 2 decimal places and an increase or decrease image indicator and green if increasing or red if decreasing in color. E.g. 1.52 (1.50%) |
| Volume | The Volume of last day session |
| Trash Can | Should delete the corresponding row from the table as well as from local storage. |

- Additionally, there needs to be a few important features:
 - **Automatic Refresh** – A bootstrap toggle switch: when it is on it should refresh the **price, change and volume fields** every five seconds.
 - **Refresh button** – Should refresh the **price, change and volume fields**.
 - **Go to stock information** – A button (navigation arrow) which should navigate to the Stock Details section.
 - “Go to stock information” button should be enabled **only when** the stock information is populated in the stock details section, which means that either a stock ticker was searched in the search form or a favorite stock symbol was clicked.

- “Go to stock information” button should be disabled on **clear** and if no stock information is available and the **disabled icon** should be shown on hover.
- **Initially** “Go to stock information” button is **disabled**.
- **Sorting function** (Sort by) - “Default” is the order in which companies are manually added to the list.
- When default “**Sort by**” is selected, “**Order**” selection is **disabled**.
- When sorting is not default, “**Order**” selection can be either “**Descending**” or “**Ascending**” according to the sorting option.

4.2.2 Stock Details Section

The Stock Details section should be designed as per Figure 5.

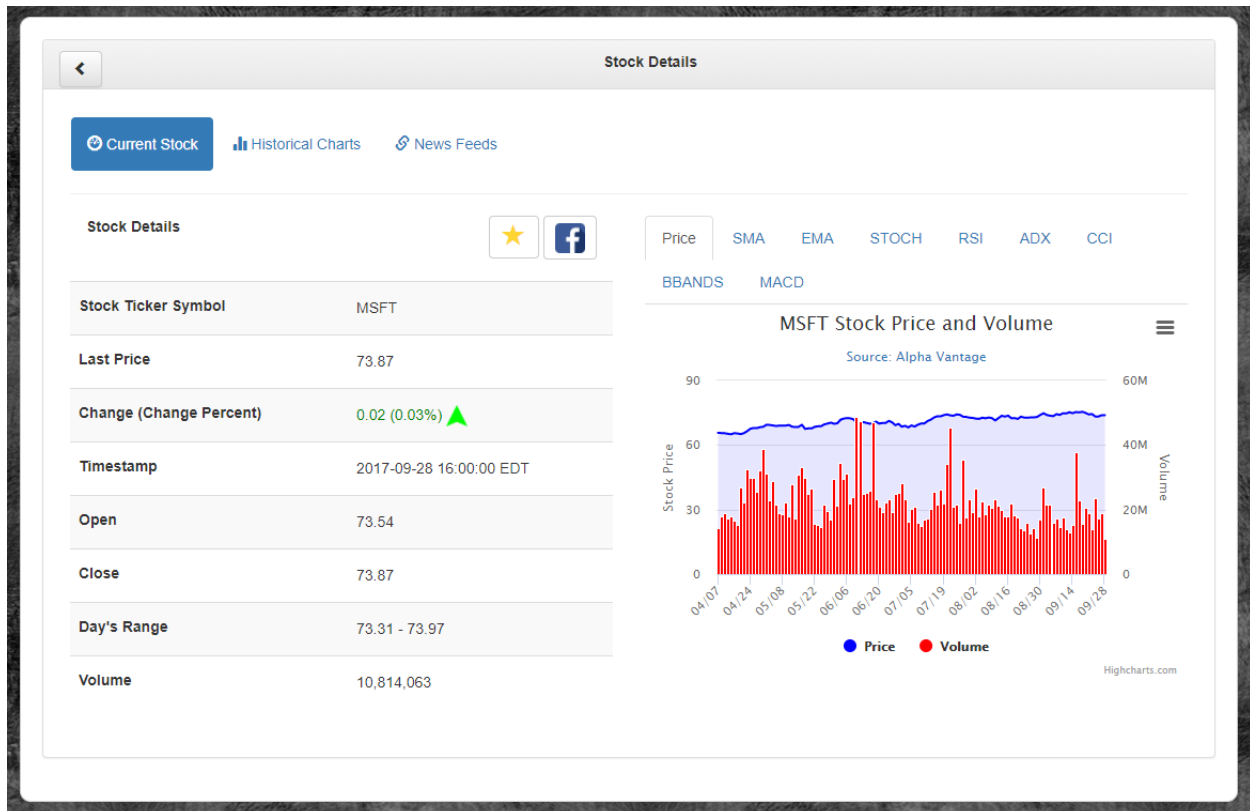


Figure 5

- The stock detail section should have 3 tabs
 - **Current Stock**
 - **Historical Charts**
 - **News Feeds**
- The **back** button (navigation arrow) in the header should navigate back to the favorite list using the **sliding** mechanism.

4.2.2.1 Tab – Current Stock

The current stock tab should be divided into two columns:

- A **table of stock details** information
- An **image of the current daily chart** of the stock of the company retrieved via **Alpha Vantage**. Refer to Section 5.3 for the API details.

4.2.2.1.1 Table of Stock Details

| | |
|-------------------------|-------------------------|
| Stock Ticker Symbol | AAPL |
| Last Price | 153.28 |
| Change (Change Percent) | -0.95 (-0.62%) ▼ |
| Timestamp | 2017-09-28 16:00:00 EDT |
| Open | 153.89 |
| Close | 153.28 |
| Day's Range | 152.70 - 154.28 |
| Volume | 21,896,592 |

Figure 7

The entries in the table shown on Figure 7 are as follows:

| Table Fields | Description |
|-------------------------|---|
| Stock Ticker Symbol | The symbol of the company |
| Last Price | The Last Price in which the company was traded at the market. It should be rounded to 2 decimals. |
| Change (Change Percent) | The change and the change percent of the current stock. The format should be change (change %) . It should be rounded to 2 decimal places with an increase or decrease image indicator, green if increasing or red if. E.g. 1.52 (1.50%) |
| TimeStamp | The time corresponding to the last price. During trading hours, it should be the current date and time. After trading hours, it should be 16:00:00 with the appropriate date. Display east coast time: EDT (when observing daylight saving time) or EST (when observing standard time). Hint: use Moment Timezone https://momentjs.com/timezone/ |

| | |
|--------------------|---|
| Open | The Open Price of last day session. |
| Close | The Close is the “Previous Close”. i.e., the close price of last business day during trading hours on business days (9:30am – 4:00pm EST). Otherwise it displays the “Close” after the close of the stock market session. |
| Day’s Range | The Low and High of last day session. |
| Volume | The Volume of last day session. |

4.2.2.1.2 Image of the Current Daily Chart

An image of the current daily stock/indicator chart of the stock of the company retrieved via Alpha Vantage API as per Figure 8-10. Note that price/volume and indicator charts should support zoom (see video).

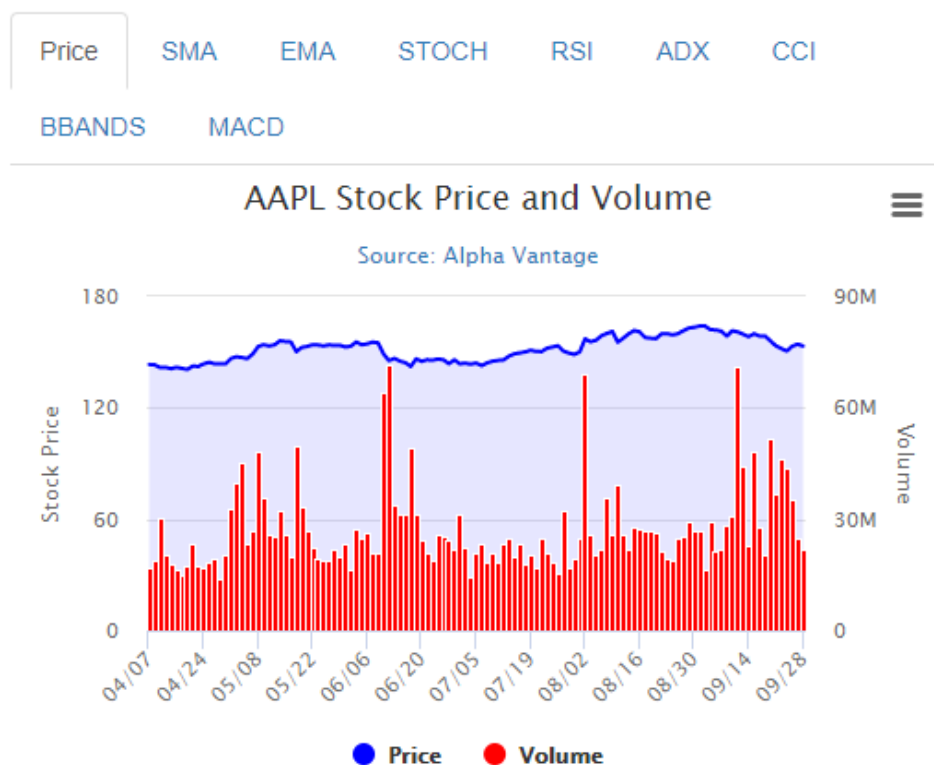


Figure 8

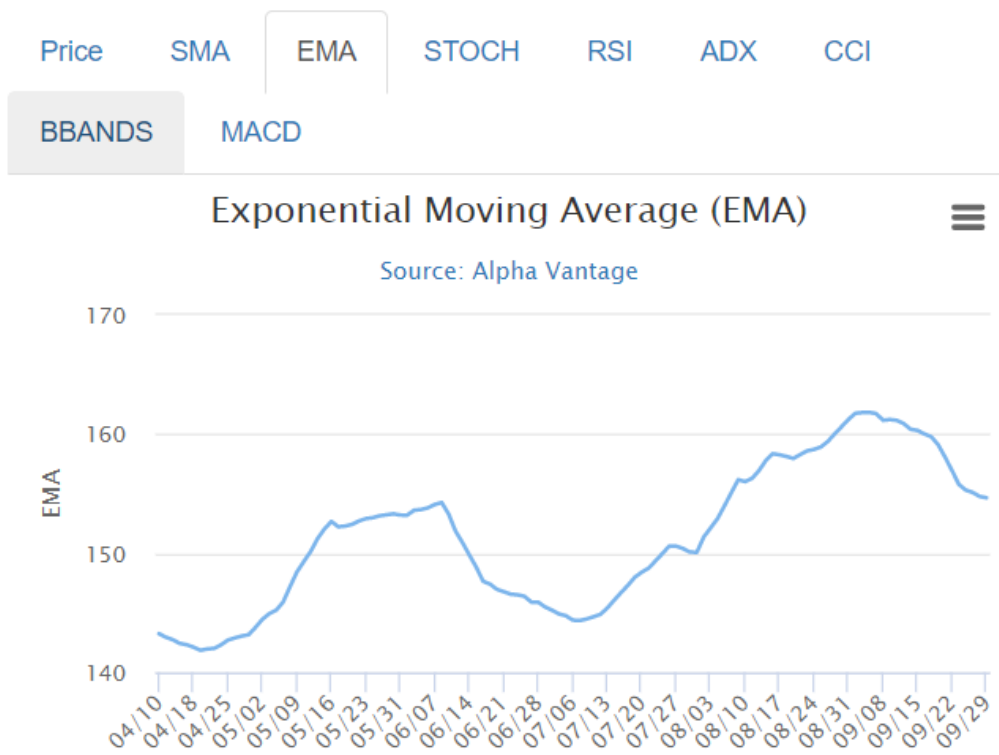


Figure 9

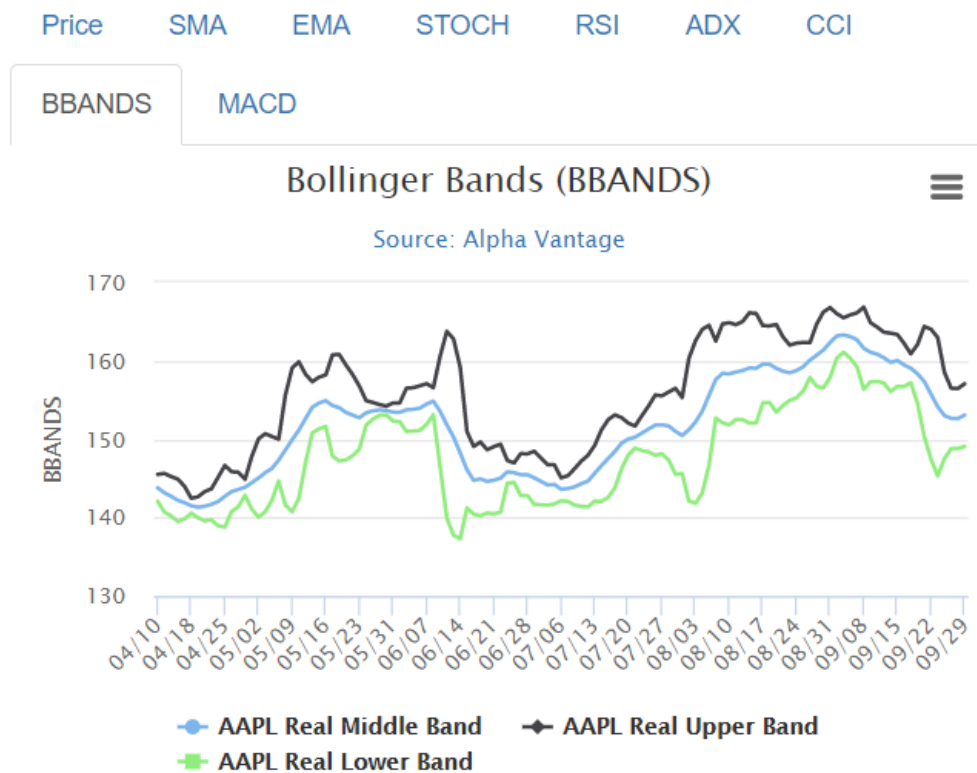


Figure 10

IMPORTANT: Unlike with HW#6, you must call the indicator's APIs **from PHP/Node.js** scripts. Directly calling the Alpha Vantage APIs from JavaScript will result in a **5 points penalty**.

4.2.2.1.3 Add to Favorite and Share on Facebook

Add to favorite:

- The favorite button (star) should allow the user to add the stock to the favorite list and store it in the browsers local storage.
- If the stock is added to the favorite list, the button should have a yellow star, otherwise a white star should be displayed (see Figure 11).
- The favorite button is disabled while we are fetching the data for the stock details table and enabled when the table is ready.



Figure 11

Share on Facebook:

- The button is disabled while we are fetching the data for the chart in the currently active tab and enabled when the chart is ready.
- When the button is clicked, the Facebook **feed dialog** should contain the chart that is in the currently active tab (see Figure 12 and demo video).
- The chart shared in the Facebook dialog is generated with HighCharts' exporting function.

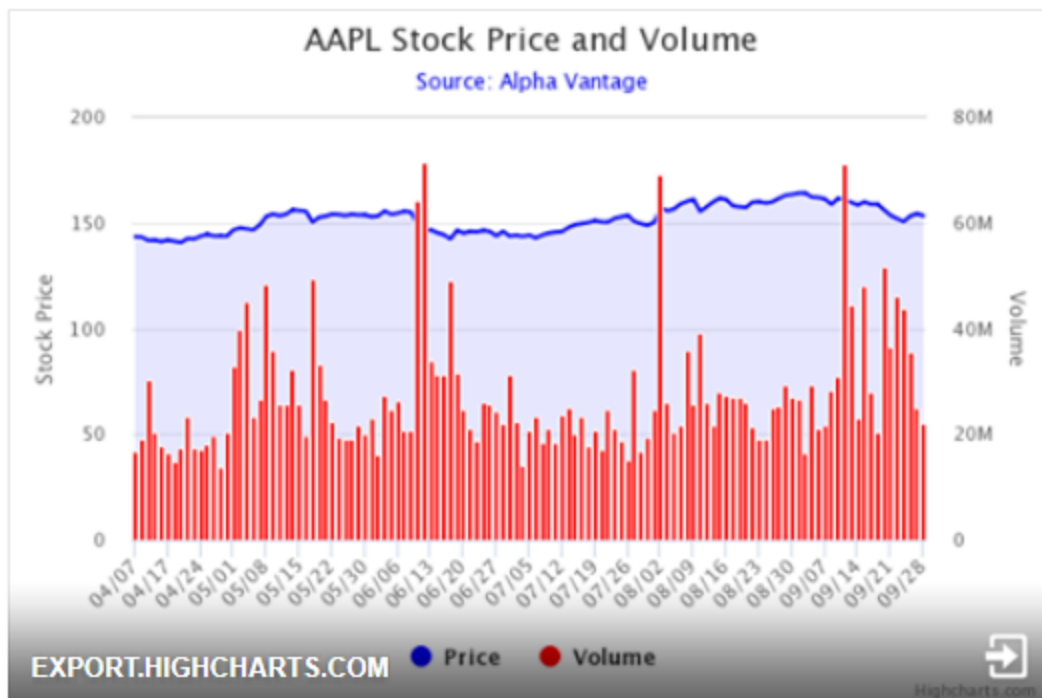


Figure 12

4.2.2.2 Tab - Historical Charts

This tab should be implemented using **HighStock** (www.highcharts.com/stock/demo) as per Figure 13 and 14 and uses the data which you have retrieved from Alpha Vantage.

- It should have these **zoom levels**: 1 week, 1 month, 3 months, 6 months, 1 year, YTD and All.
- When All level is selected, there should be 1000 data shown.
- It should **default to showing 1 week** worth of stock data.
- The **title** of the chart should be *company symbol* stock value.
- The **subtitle** should hyper link to Alpha Vantage (<https://www.alphavantage.co/>)
- The **X Axis** should be date time.
- The **Y Axis** should be Stock Value.



Figure 13



Figure 14

4.2.2.3 Tab - News Feed Section

Like HW#6, you will be using the Seeking Alpha News feed API. Each row of the stock news will have the following information (see Figure 15):

- The **title** of the news which is a hyperlink to the URL of the news source which when clicked should open the link in the new tab.
- The **Author** of the piece of news.
- The **date** when the news was published.

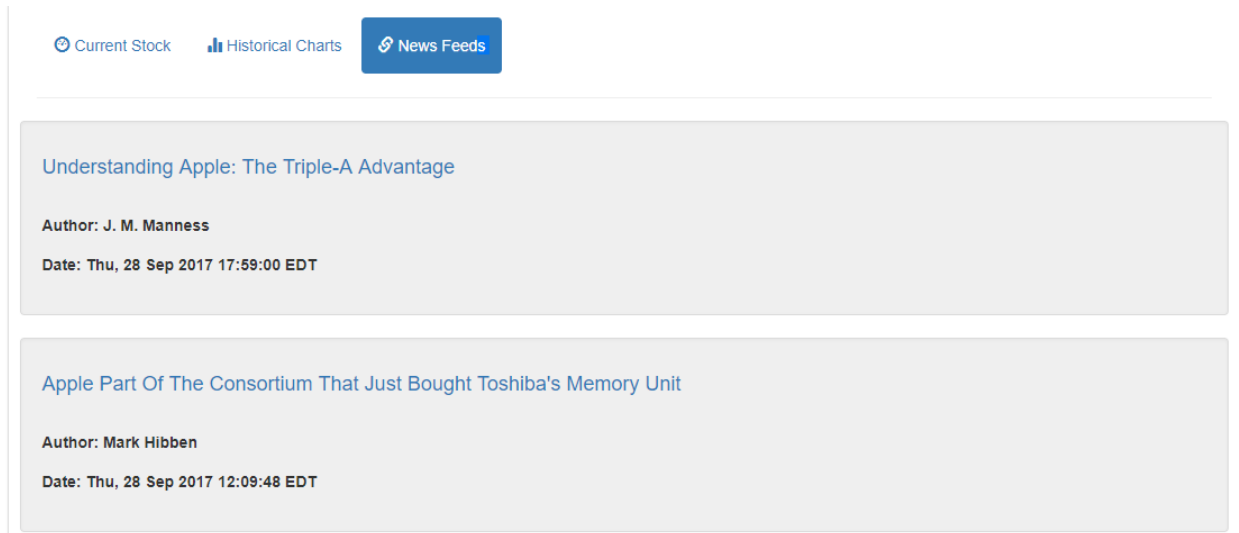


Figure 15

4.2.3 Display Results for Error

If for any reason (non-existing stock ticker symbols, API failure, etc.) an error occurs, an appropriate error message should be displayed for each of the three sections as per Figures 16-18:

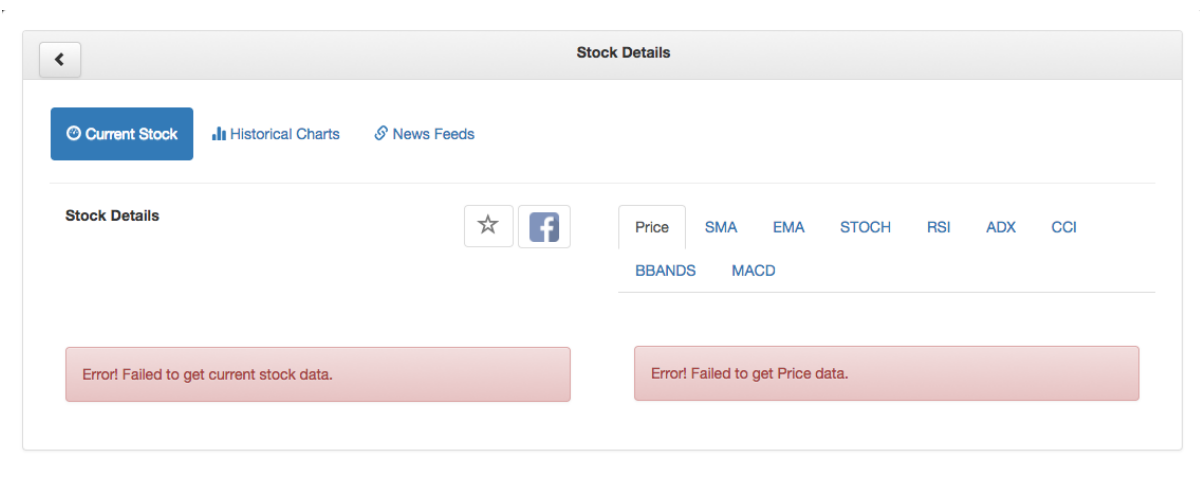


Figure 16

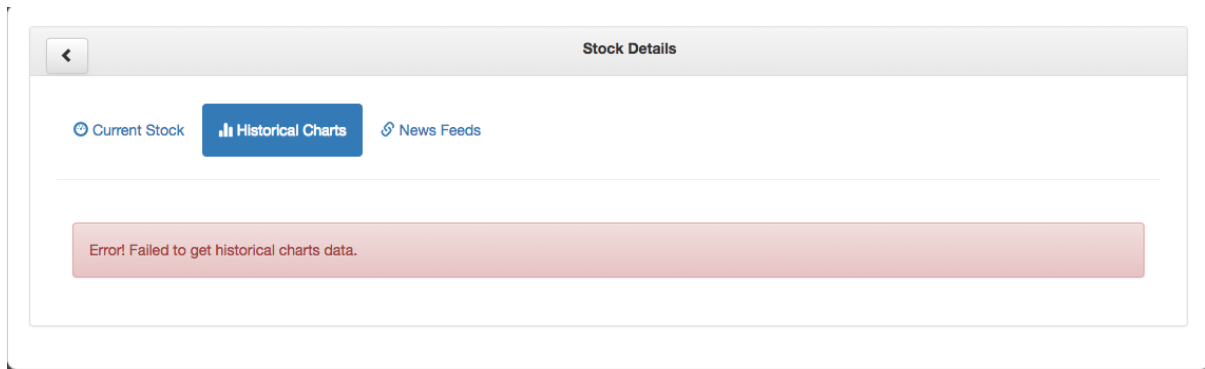


Figure 17

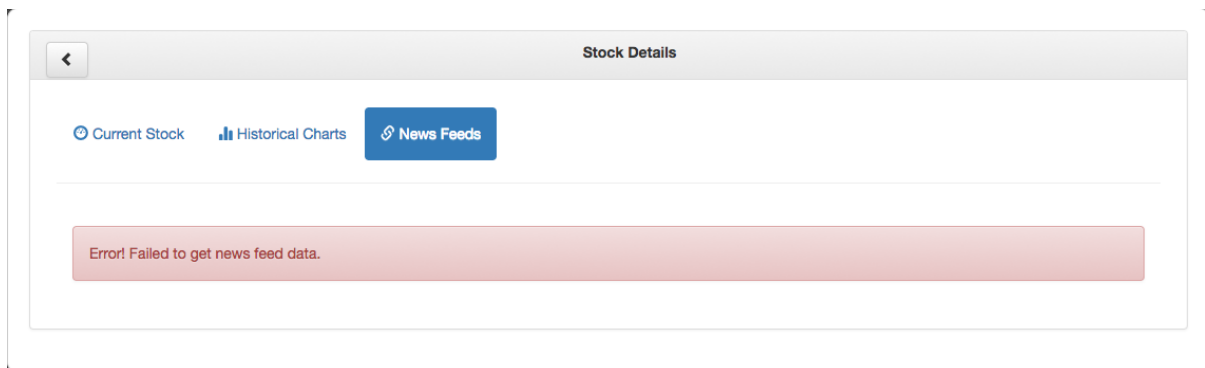


Figure 18

4.2.4 Display Progress Bar

Whenever data is being fetched, a dynamic progress bar must be displayed as shown in Figure 19 and 20. Please check the video for this functionality.

You can use the progress bar component of **Bootstrap** to implement the feature. You can find hints about the bootstrap components in the Hints section.

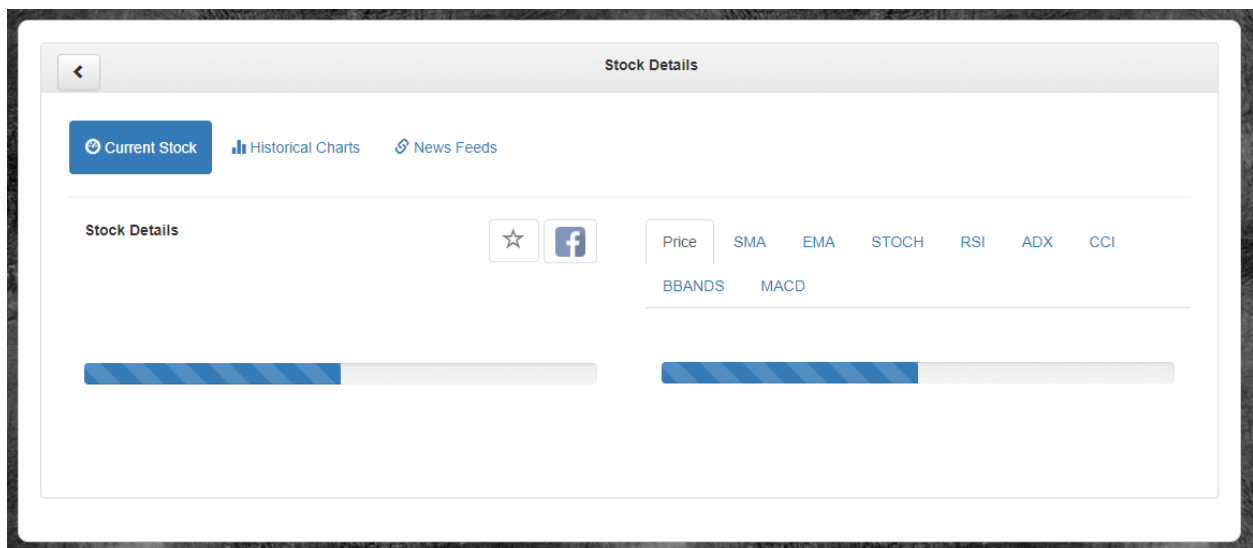


Figure 19

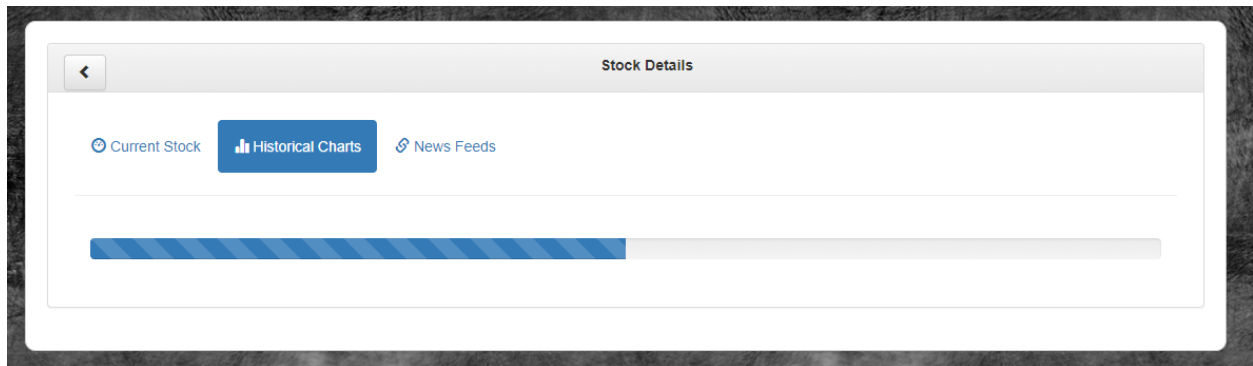
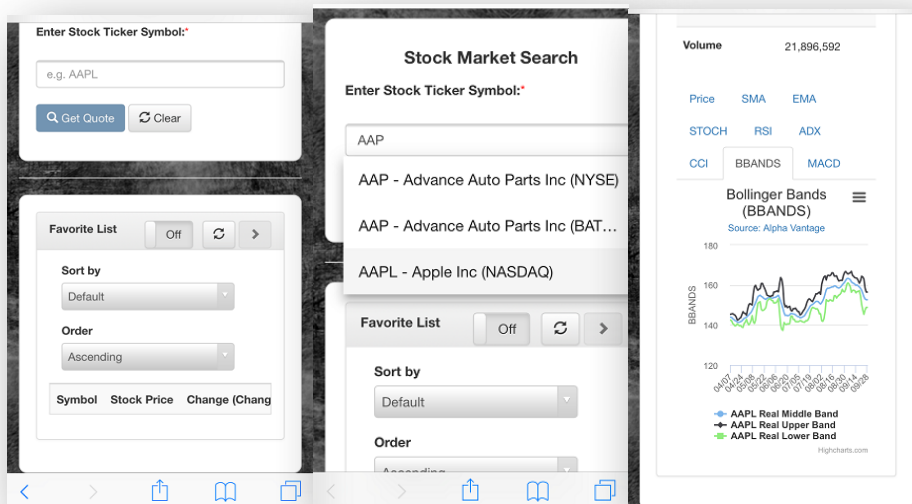
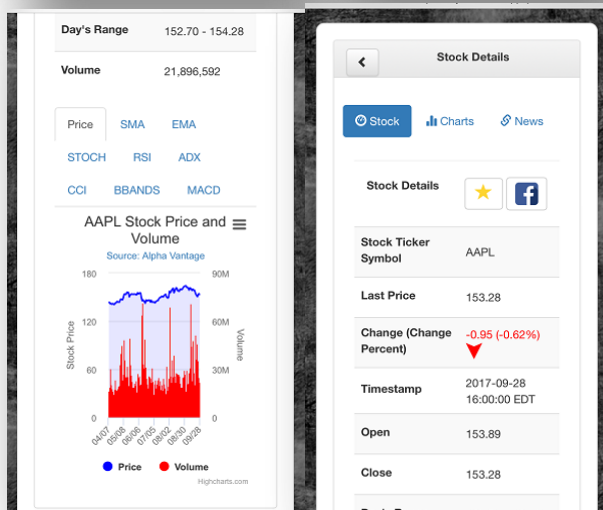
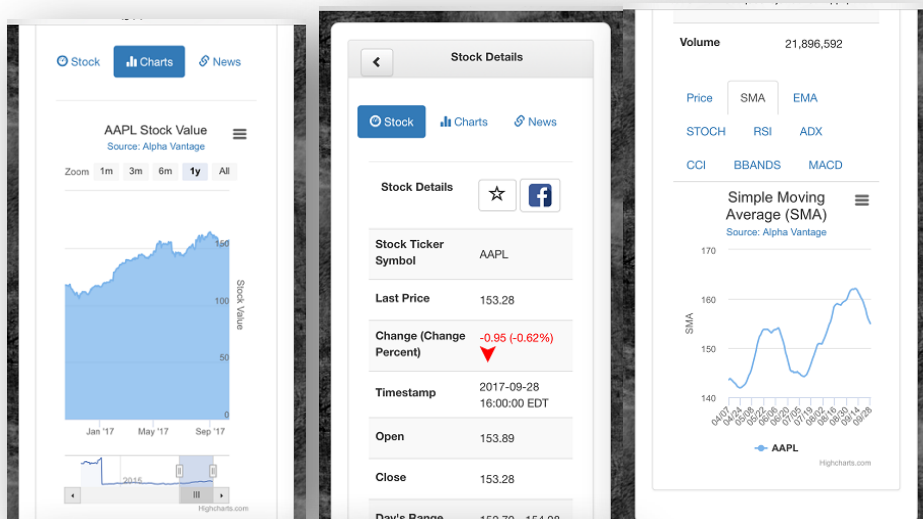
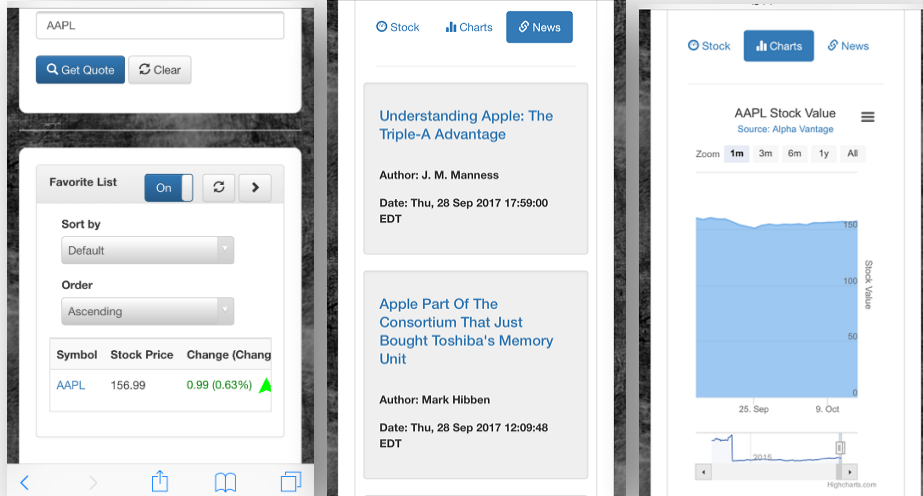


Figure 20

4.3 Responsive Design

The following are snapshots of the major screens taken on Safari of iPhone 6s. See the video for mode “dynamic” details.





Some requirements in the mobile view are listed here:

- The search form should display each component in a vertical way on smaller screens.
- Favorite table content for all favored stock can be scrolled horizontally.
- The stock details table and chart image should be aligned vertically on smaller screens.

5. APIs Documentation

The same APIs that were used for HW#6 can be used for HW#8 as well. Additionally, you should also use the Market on Demand API to implement the autocomplete function in HW#8.

5.1 JSON data from Market on Demand

The following is a sample JSON data that explains the section of the data to be used for autocompleting:

Request URL: <http://dev.markitondemand.com/MODApis/Api/v2/Lookup/json?input=Apple>

```
[{
  "Symbol": "AAPL",
  "Name": "Apple Inc",
  "Exchange": "NASDAQ"
}, {
  "Symbol": "APLE",
  "Name": "",
  "Exchange": "NYSE"
}, {
  "Symbol": "APLE",
  "Name": "Apple Hospitality REIT Inc",
  "Exchange": "BATS Trading Inc"
}, {
  "Symbol": "VXAPL",
  "Name": "CBOE Apple VIX Index",
  "Exchange": "Market Data Express"
}]
```

5.2 JSON data from Alpha Vantage

The following is a sample JSON data that explains the section of the data to be used for displaying stock quote/indicators information:

You can learn more about the API by visiting <https://www.alphavantage.co/documentation/>

5.2.1 Stock Quote API

The API to retrieve a Stock Quote should be:

https://www.alphavantage.co/query?function=TIME_SERIES_DAILY&symbol=MSFT&apikey=demo

The MSFT keyword should be replaced by the symbol from your search. The JSON returned is as follows:

```
{
  "Meta Data": {
    "1. Information": "Daily Prices (open, high, low, close) and Volumes",
    "2. Symbol": "MSFT",
    "3. Last Refreshed": "2017-09-15",
    "4. Output Size": "Compact",
    "5. Time Zone": "US/Eastern"
  },
  "Time Series (Daily)": {
    "2017-09-15": {
      "1. open": "74.8300",
      "2. high": "75.3900",
      "3. low": "74.0700",
      "4. close": "75.3100",
      "5. volume": "37901927"
    },
    "2017-09-14": {
      "1. open": "75.0000",
      "2. high": "75.4900",
      "3. low": "74.5200",
      "4. close": "74.7700",
      "5. volume": "15373384"
    },
    "2017-09-13": {
      "1. open": "75.0000",
      "2. high": "75.4900",
      "3. low": "74.5200",
      "4. close": "74.7700",
      "5. volume": "15373384"
    }
  }
}
```

Figure 21

5.2.3 Indicators API

The API to retrieve the indicators data should be: https://www.alphavantage.co/query?function=SMA&symbol=MSFT&interval=15min&time_period=10&series_type=close&apikey=demo

The parameter in the query string is a JSON value as in this example:

```
{
  "Meta Data": {
    "1: Symbol": "MSFT",
    "2: Indicator": "Simple Moving Average (SMA)",
    "3: Last Refreshed": "2017-09-15",
    "4: Interval": "weekly",
    "5: Time Period": 10,
    "6: Series Type": "open",
    "7: Time Zone": "US/Eastern"
  },
  "Technical Analysis: SMA": {
    "2017-09-15": {
      "SMA": "72.8130"
    },
    "2017-09-08": {
      "SMA": "72.3150"
    },
    "2017-09-01": {
      "SMA": "72.1210"
    }
  }
}
```

Figure 22

5.3 Seeking Alpha News feed

Like HW#6, all the stock news comes from the Seeking Alpha News RSS feed. The URL to request the news is: <https://seekingalpha.com/api/sa/combined/SYMBOL.xml>

The response is a XML-formatted object (whose content is maintained by Seeking Alpha News). The PHP/Node.js script should translate the XML to JSON and send the JSON formatted news data to the JavaScript program.

5.4 Facebook API

When the “share on Facebook” button is pressed, the web application should display a popup to authorize the user to Facebook (i.e. logs him/her in) using the application and user credentials if the user is not already logged in to Facebook. More details about Facebook feed dialog can be found here:

<https://developers.facebook.com/docs/sharing/reference/feed-dialog>

A function to call Facebook feed looks like this:

```
FB.ui({
  app_id: YOUR_APP_ID,
  method: 'feed',
  picture: URL_OR_PICTURE
}, (response) => {
  if (response && !response.error_message) {
    //succeed
  } else {
    //fail
  }
});
```

6. External Libraries

This a comprehensive list of JavaScript libraries that you will use:

- JQuery – <https://code.jquery.com/>
- Bootstrap – <http://getbootstrap.com/getting-started/>
- Bootstrap Toggle – <http://www.bootstraptoggle.com/>
- HighCharts/HighStocks – <http://code.highcharts.com/>
- Moment.js – <http://momentjs.com/> for time conversion
- Moment Timezone – <https://momentjs.com/timezone/> for time conversion
- Facebook - <https://developers.facebook.com/docs/javascript>
- AngularJS Material (for AngularJS) – <https://material.angularjs.org/latest/demo/autocomplete>
- Angular Material (for Angular 2 and Angular 4) – <https://material.angular.io/components/autocomplete/overview>

7. Implementation Hints

7.1 Images

The images needed for this homework are available here:

<http://cs-server.usc.edu:45678/hw/hw8/images/background.png>

<http://cs-server.usc.edu:45678/hw/hw8/images/facebook.png>

<http://cs-server.usc.edu:45678/hw/hw8/images/Down.png>

<http://cs-server.usc.edu:45678/hw/hw8/images/Up.png>

7.2 Get started with the Bootstrap Library

To get started with the Bootstrap toolkit, please refer to the link - <http://getbootstrap.com/getting-started/>. You need to import the necessary CSS file and JS file provided by Bootstrap.

7.3 Bootstrap UI Components

Bootstrap provides a complete mechanism to make Web pages responsive to different mobile devices. In this exercise, you will get hands-on experience with responsive design using the Bootstrap Grid System. At a minimum, you will need to use Bootstrap Form, Tab, Wells, and Glyphicons to implement the required functionality.

| | |
|-----------------------|---|
| Bootstrap Form | http://getbootstrap.com/css/#forms |
| Bootstrap Tabs | http://getbootstrap.com/javascript/#tabs |
| Bootstrap Wells | http://getbootstrap.com/components/#wells |
| Bootstrap Glyphicons | http://getbootstrap.com/components/#glyphicons |
| Bootstrap Progressbar | https://v4-alpha.getbootstrap.com/components/progress/ |
| Bootstrap Alerts | https://v4-alpha.getbootstrap.com/components/alerts/ |

7.4 Google App Engine/Amazon Web Services

You should use the domain name of the GAE/AWS service you created in HW#7 to make the request. For example, if your GAE/AWS server domain is called example.appspot.com /example.elasticbeanstalk.com, the JavaScript program will performs a GET request with parameter name="AAPL", and a query of the following type will be generated:

(GAE) - <http://example.appspot.com/?symbol=AAPL>

(AWS) - <http://example.elasticbeanstalk.com/?symbol=AAPL>

7.5 Deploy Node.js on GAE/AWS

If your backend is implemented with Node.js, when you deploy HW#8 to AWS or GAE, you should select Nginx as your proxy server, which should be the default option. If you select any proxy server other than Nginx, you will NOT get the 2 (two) extra credits.

7.6 AJAX call

You should send the request to the PHP script by passing the URL to the Ajax function (jQuery or Angular).

You must use a GET method to request the resource since you are required to provide this link to your homework list to let graders check whether the PHP script code is running in the "cloud" on Google GAE/AWS. Please refer to the grading guideline for details.

Figure 23 shows an example AJAX call:

```
$.ajax({
  url: 'URL in GAE',
  //parameter list
  data: {symbol:"AAPL"},
  type: 'GET',
  success: function(response,status,xhr){
    //parse the output
  },
  error: function(xhr, status, error){
    //parse the error
  }
});
```

Figure 23

7.7 Angular Autocomplete

The library used to implement autocomplete is AngularJS Material (for AngularJS):

<https://material.angularjs.org/latest/demo/autocomplete>

or Angular Material (for Angular 2 and Angular 4):

<https://material.angular.io/components/autocomplete/overview>

Use of Angular is **must**. The reasons we want you to learn Angular are:

- Angular extends HTML with new attributes.
- Angular is perfect for Single Page Applications (SPAs).
- Angular is easy to learn.

To get started with AngularJS, the W3SCHOOL website has easy tutorials:

<http://www.w3schools.com/angular/>

Or, you can follow the tutorials at the official website:

<https://docs.angularjs.org/tutorial>

To get started with Angular 2 or Angular 4, visit:

<https://angular.io/docs>

7.8 Use Node.js to parse XML

You can use any node.js module to parse xml.

Xml2js is recommended: <https://www.npmjs.com/package/xml2js>

7.9 HTML5 Local Storage

Local storage is more secure, and large amounts of data can be stored locally, without affecting website performance. Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server. There are two methods `getItem()` and `setItem()` that you can use. The local storage can only store strings. So, you need to convert the data to string format before storing it in the local storage. For more information see:

<https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage>
http://www.w3schools.com/html/html5_webstorage.asp

8. Files to Submit

In your course homework page, you should update the HW#8 link to refer to your new initial web page for this exercise. Additionally, you need to provide an additional link to the URL of the GAE/AWS service where the AJAX call is made with a sample parameter value (i.e. a valid stock symbol). Also, submit all your files (HTML, JS, CSS, PHP or Node.js) electronically to the csci571 account so that they can be graded and compared to all other students' code. Do not include any images that we provided or that are included in any library.

****IMPORTANT**:**

All videos are part of the homework description. All discussions and explanations in Piazza related to this homework are part of the homework description and will be accounted into grading. So please review all Piazza threads before finishing the assignment.