

Homework 8: Responsive Web Design

Stock Search using MarkitonDemand & Facebook Mashup
(Bootstrap/JQuery/AJAX/JSON/Cloud Exercise)

1. Objectives

- Become familiar with the AJAX and JSON technologies
- Use a combination of HTML5, Bootstrap, JQuery, JQuery Plugins, CSS, and PHP
- 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 MarkitonDemand 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 XSLT;
- Asynchronous data retrieval using XMLHttpRequest;
- JavaScript binding everything together.

See the class slides at <http://www-scf.usc.edu/~csci571/Slides/ajax.ppt>

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://www-scf.usc.edu/~csci571/Slides/JSON1.ppt>

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> and [http://en.wikipedia.org/wiki/Bootstrap_\(front-end_framework\)](http://en.wikipedia.org/wiki/Bootstrap_(front-end_framework))

2.3 Markit on Demand

Markit on Demand API provides detailed description about the stock information of company as well as historical stock values. You can refer to the API description on the following link:

<http://dev.markitondemand.com/MODApis/>

2.4 Facebook

Facebook provides developers with an API called the Facebook Platform. Facebook Connect is the next iteration of Platform, which provides a set of API's 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>

2.5 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 7.5 for .NET, and Apache Tomcat for Java.

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

2.6 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 the this page:

<https://cloud.google.com/appengine/docs/php/>

3. High Level Description

Similar to homework 6, in this exercise you will create a webpage that allows users to search for stock information using the MarkitonDemand API and display the results on the same page below the form.

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

A user will first open a page as shown below in Figure 1, where he/she can enter the company name or symbol, and select from a list 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.

Figure 1

Once the user has provided data and selected a result from the autocomplete list he would click on Get Quote, when validation must be done to check that the entered data is valid.

Once the validation is successful, the JQuery function `ajax()` is executed to start an asynchronous transaction with a PHP script running on your Google App Engine/Amazon Web Services, and passing the search form data as parameters of the transaction.

The PHP script your request is based on your HW#6. The difference is that this time the file does not need to display the data as HTML but instead will return the JSON data from the API to your search webpage. The webpage must then use JavaScript to extract data from the JSON and display the results on the same webpage. Description of how to display the results is given in Section 6.

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 homework 6.

The top-level interface consists of the following:

- A form which has an input to enter the company name or 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 (**bootstrap carousel**) should start with an empty favorite list.

The search form has two buttons:

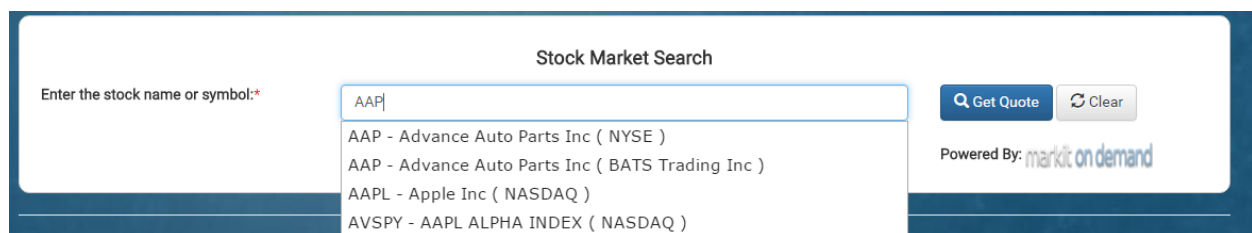
1. **GET QUOTE** button: On the button click validations are performed (Refer to 4.1.3). If validations are successful, then an AJAX request is made to your web server (PHP on Google App Engine/Amazon Web Services), providing it with form data that was entered. If validations fail, appropriate messages must be displayed under the appropriate text box, and an AJAX request should **NOT** be made with invalid data.

2. **CLEAR** button: This button must clear the text field, resets the carousel to the favorite list and clear all validation errors if present. The clear operation is a JavaScript function.

The form should include a markit on demand logo linking to <http://dev.markitondemand.com/MODApis/>. The form should end with a white horizontal line to separate the form from the search results.

4.1.2 Autocomplete

- A form allows a user to enter a keyword (company symbol or company name) to retrieve information (quote information, news and stock chart) from Markit on Demand. Based on the user input the text box should display a list of all the matching companies and symbol (see Figure 2) by making an AJAX call on every keystroke that is entered. This should be implemented using **jQuery autocomplete**. Refer to Section 7.

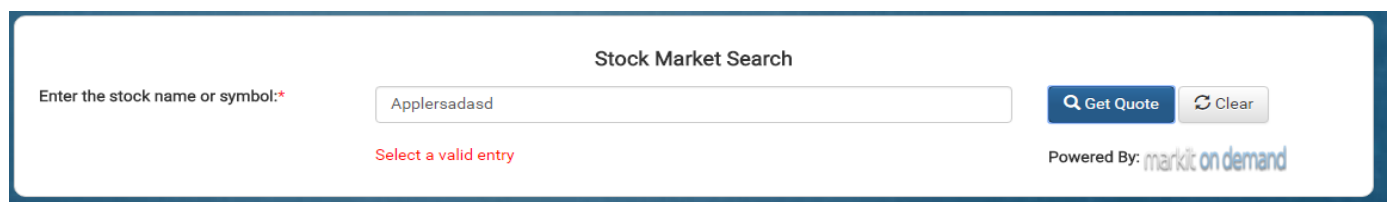


The figure shows a web form titled "Stock Market Search". On the left, there is a label "Enter the stock name or symbol: *" and a text input field containing "AAP". Below the input field, a dropdown menu displays four suggestions: "AAP - Advance Auto Parts Inc (NYSE)", "AAP - Advance Auto Parts Inc (BATS Trading Inc)", "AAPL - Apple Inc (NASDAQ)", and "AVSPY - AAPL ALPHA INDEX (NASDAQ)". 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 these buttons is the text "Powered By: markit on demand" with the Markit logo.

Figure 2

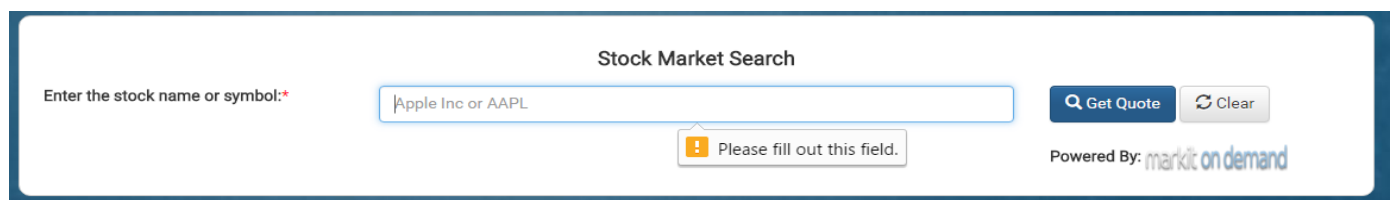
4.1.3 Validation

- The validations that are needed to be implemented in the input query string are:
 - **Invalid Entry** – Display a message which should be verbose (Figure 3)
 - **Empty Entry** – Use the HTML5 validation to display the default error (Figure 4)



The figure shows the "Stock Market Search" form with the text "Applersadasd" entered in the input field. Below the input field, a red error message reads "Select a valid entry". The "Get Quote" and "Clear" buttons are visible on the right, along with the "Powered By: markit on demand" text.

Figure 3



The figure shows the "Stock Market Search" form with "Apple Inc or AAPL" entered in the input field. Below the input field, a yellow warning box with an exclamation mark icon contains the text "Please fill out this field." The "Get Quote" and "Clear" buttons are visible on the right, along with the "Powered By: markit on demand" text.

Figure 4

4.1.4 Get Quote Execution

- Once the validation is successful, you should execute an **AJAX transaction** to the **PHP script** which is located in the **Google App Engine/Amazon Web Services**.
- The PHP script on Google App Engine/Amazon Web Services, modified after Homework 6, is used to retrieve data from Markit on Demand. You should pass the **company symbol as a parameter** of the transaction when calling the PHP script.

For example, if your Google App Engine/Amazon Web Services 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>

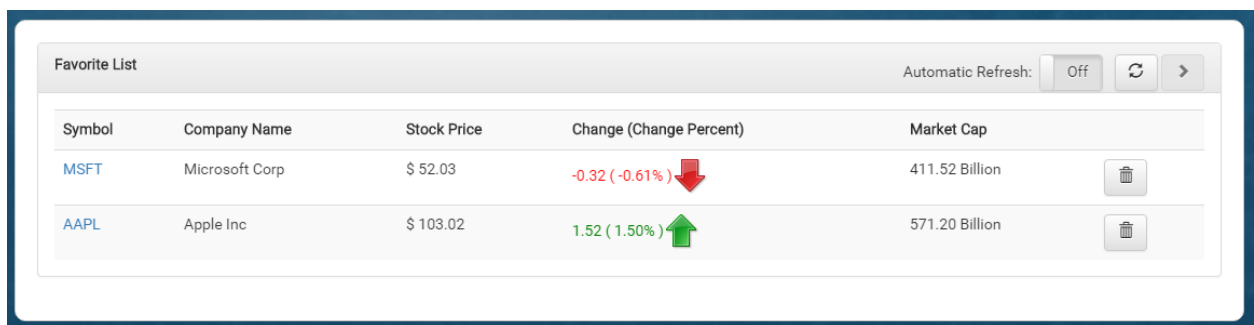
- The PHP script running on the Google App Engine/Amazon Web Services would **extract the stock details** of the company symbol, perform an API request to Markit on Demand, and returns the data in JSON data.
- Notice that in Homework 6 your PHP script produced HTML. In this exercise, the output must be changed to **JSON** and the PHP code must run on Google App Engine/Amazon Web Services.
- 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

4.2.1 Design

The result area will be a **carousel feature** of the bootstrap library. Refer to Section 7.2.

- There should be **two sections** in the carousel
 - The first section should be the Favorite List.
 - The second section should be the Stock Details.



Symbol	Company Name	Stock Price	Change (Change Percent)	Market Cap
MSFT	Microsoft Corp	\$ 52.03	-0.32 (-0.61%)	411.52 Billion
AAPL	Apple Inc	\$ 103.02	1.52 (1.50%)	571.20 Billion

Figure 5

4.2.1.1 Favorite Section

The Favorite Section should be designed as per Figure 5.

- The data should be loaded from the **local storage** of the browser. The local storage should contain the list of the favorite stocks. For more about local storage refer to the appendix.
- A table containing the following information:

Table Field	Description
Symbol	Displays the Symbol of the Company
Company Name	Displays the Name of the Company
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%)
Market Cap	Displays the market cap of the current stock. Possible suffixes are {Billions, Million, None}. Each value should be calculated and appended with appropriate suffix and rounded to 2 decimals. E.g. 5.71 Billion or 5.71 Million or 571000
Trash Can	Should delete the corresponding row from the table as well local storage.

- Additionally, there needs to be a few important features:
 - **Automatic Refresh** – A bootstrap toggle switch (Refer to Section 7): when it is on it should refresh only the price and change column every 5 seconds.
 - **Refresh button** – Should refresh only the price and change column fields and not the rest of the table.
 - **Go to stock information** – A button 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.
 - 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.
 - All the 3 buttons should have **tooltips** which describes the functionality.

4.2.1.2 Stock Details Section

The Stock Details section should be designed as per Figure 6.

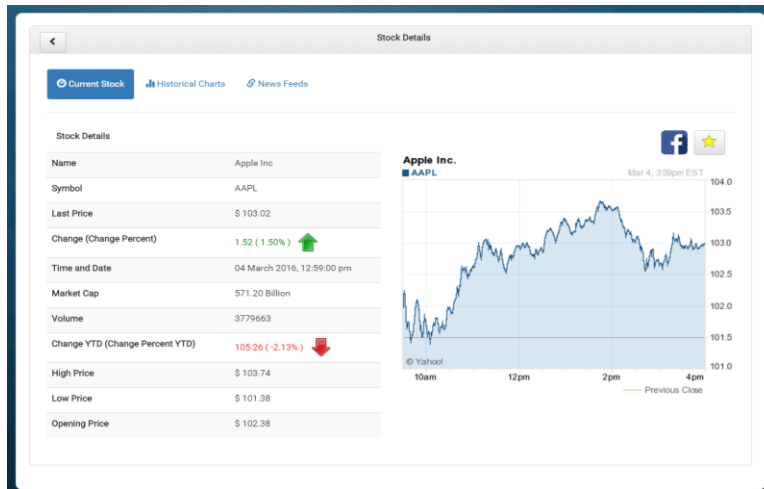


Figure 6

- The stock detail section should have 3 tabs
 - **Current Stock**
 - **Historical Charts**
 - **News Feeds**
- The **back** button in the header should navigate back to the favorite list.

4.2.1.3 Stock Details Section

The current stock tab should be divided into two columns:

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

4.2.1.4 Current Stock Table

Stock Details	
Name	Apple Inc
Symbol	AAPL
Last Price	\$ 103.02
Change (Change Percent)	1.52 (1.50%) ↑
Time and Date	04 March 2016, 12:59:00 pm
Market Cap	571.20 Billion
Volume	3779663
Change YTD (Change Percent YTD)	105.26 (-2.13%) ↓
High Price	\$ 103.74
Low Price	\$ 101.38
Opening Price	\$ 102.38

Figure 7

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

Table Fields	Description
Name	Displays the name of the company
Symbol	Displays the symbol of the company
Last Price	Display the Last Price in which the company was traded at the market. It should be preceded with a \$ and should be rounded to 2 decimals
Change (Change Percent)	Display 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%)
Time and Date	The time and date of the request. Should be the timestamp field of the JSON
Market Cap	Displays the market cap of the current stock. Possible suffixes are {Billions, Million, None}. Each value should be calculated and appended with appropriate suffix and rounded to 2 decimals. E.g. 5.71 Billion or 5.71 Million or 571000.
Volume	Displays the volume of the stock in the market
Change YTD (Change Percent YTD)	Displays the change YTD and the change percent YTD 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%)
High Price	Displays the High Price in which the company was traded at the market. It should be preceded with a \$ and should be rounded to 2 decimals
Low Price	Displays the Low Price in which the company was traded at the market. It should be preceded with a \$ and should be rounded to 2 decimals
Opening Price	Displays the Opening Price in which the company was traded at the market. It should be preceded with a \$ and should be rounded to 2 decimals

4.2.1.5 Current Day Stock Chart

An image of the current daily chart of the stock of the company retrieved via Yahoo charts API as per Figure 8. Refer to appendix.



Figure 8

4.2.1.6 Facebook

A Facebook button which should generate a **feed dialog** with the following information (see Figure 9):

- The text with “Current Stock Price of *company name* is *amount*”.
- The text with “Stock Information of *company name (symbol)*”.
- The text with “Last Traded Price: *amount*, Change: *change (change %)*”.
- The *amount* should always be the value with 2 decimals preceded by \$.
- The image of the current day stock value.



Figure 9

- The favorite button should allow the user to add the stock to the favorite list and store it in the browsers local storage.

- If the stock is in the favorite list, the button should have a yellow star, otherwise a white star (see Figure 10)



Figure 10

4.2.1.7 Historical Charts Section

This should be implemented using **HighCharts' Highstock** (www.highcharts.com/stock/demo) as per **Figure 11** and the data which you have retrieved from your AJAX call.

- It should have the following **zoom levels**: 1 week, 1 month, 3 months, 6 months, 1 year, YTD and All.
- It should **default to showing 1 week** worth of stock data.
- The **title** of the chart should be *company symbol* stock value.
- The **X Axis** should be date time.
- The **Y Axis** should be Stock Value.

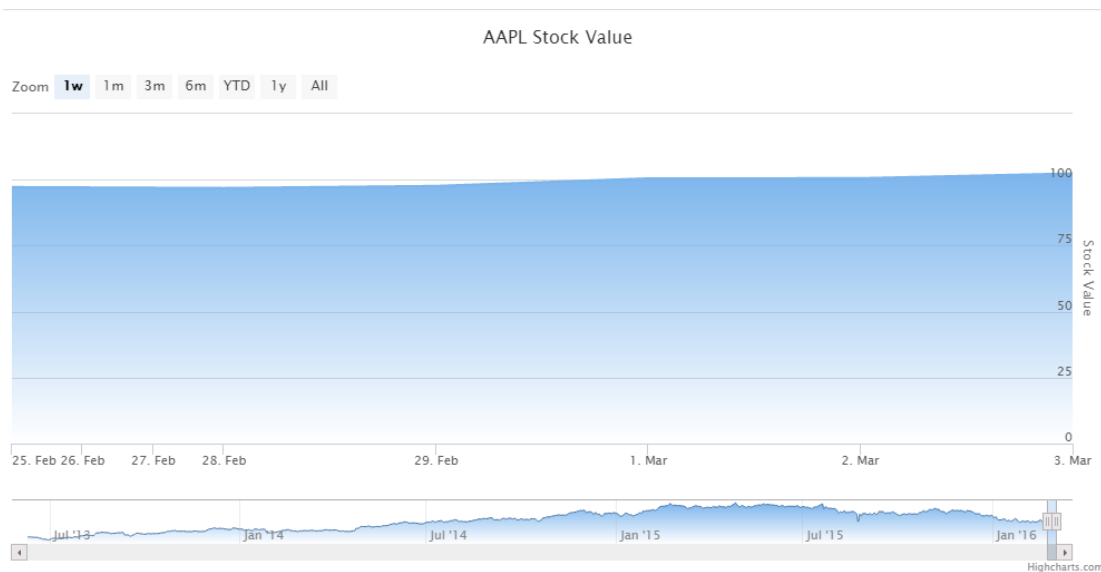


Figure 11

4.2.1.8 News Feed Section

You will be using the Google News Feed API or Bing Search API. Refer to the appendix on how to use the Google News service or Bing Search API.

- Each row will have the following information (see Figure 12):
 - 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 **content** of the news.
 - The **publisher** details.
 - The **date** of the news.

Table Mapping between Google News API and Bing Search API

Field	Google News API	Bing Search API
URL	<i>unescapeUrl</i> field in the result array of the response	Url field in the result array of the response
Title	<i>titleNoFormatting</i> field in the result array of the response	<i>Title</i> field in the result array of the response
Content	<i>content</i> field in the result array of the response	<i>Description</i> field in the result array of the response
Publisher	<i>publisher</i> field in the result array of the response	<i>Source</i> field in the result array of the response
Date	<i>publishedDate</i> field in the result array of the response	<i>Date</i> field in the result array of the response

[Apple \(AAPL\) Stock Gains on Potential Dividend, Buyback Hikes](#)

TheStreet TV anchor Rhonda Schaffler details Apple's possible increases to its dividend and share repurchase program in the above video. NEW YORK (TheStreet) -- Apple (AAPL - Get Report) stock is climbing by 2.06% to \$103.59 in afternoon trading on ...

Publisher: TheStreet.com

Date: 04 Mar 2016 09:25:07

[Apple \(AAPL\) Creates Customer Support Page on Twitter](#)

NEW YORK (TheStreet) -- Apple (AAPL - Get Report) created a new Twitter (TWTR) account for customer support, which expands the iPhone maker's social media presence. The account spurred instant questions and complaints from users about their ...

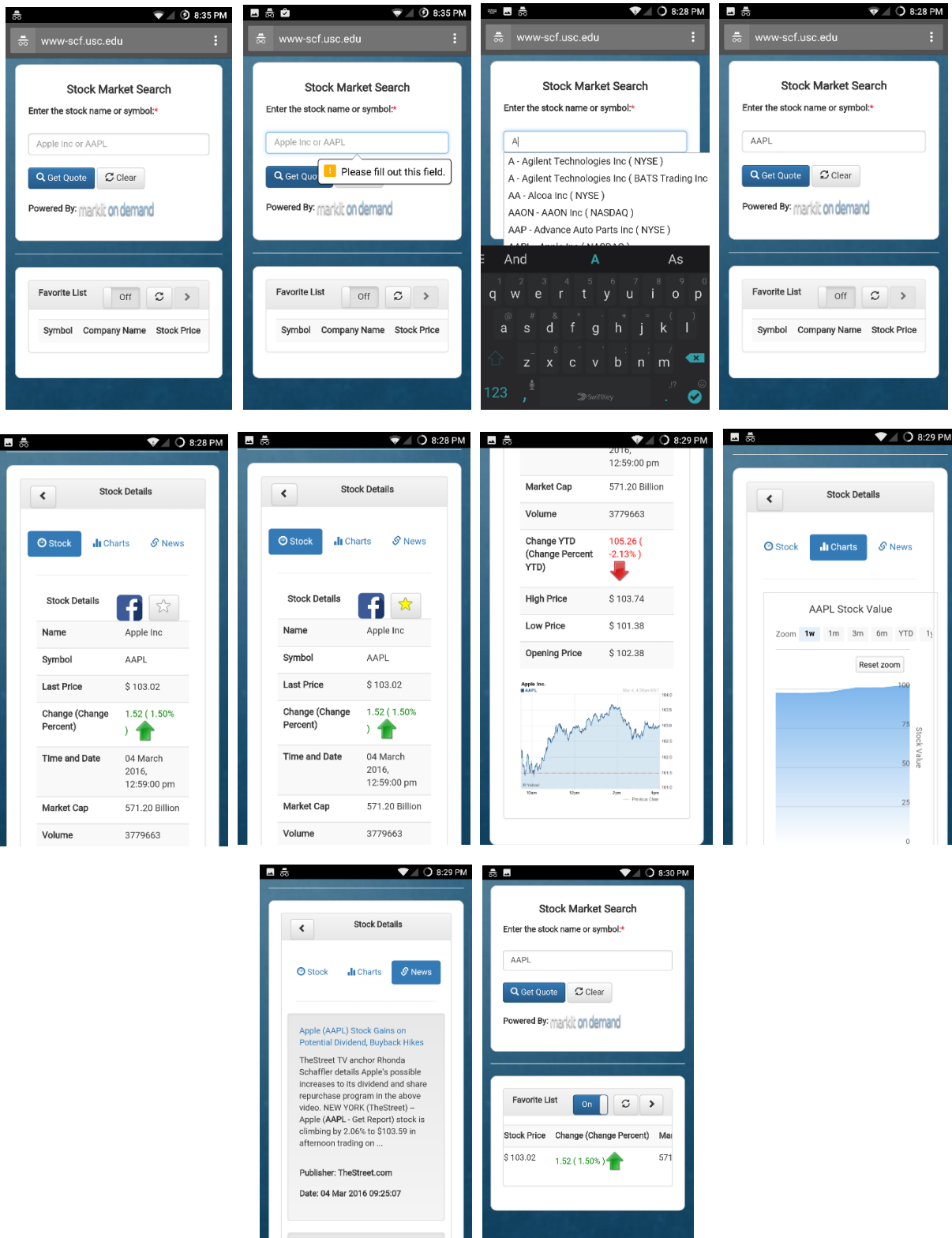
Publisher: TheStreet.com

Date: 03 Mar 2016 12:48:45

Figure 12

4.2.1.9 Responsive Web Results

The following are snapshots of the major screens taken on an Android phone.



5. APIs Documentation

The same APIs that were used for homework 6 can be used for homework 8 as well.

5.1 JSON data from Markit on Demand

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

You can learn more about the API by visiting <http://dev.markitondemand.com/MODApis/#doc>

5.1.1 Lookup API

The API to lookup the symbol or name of the company

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

The **Apple** keyword should be replaced by the value of the textbox input. A sample JSON result is shown in Figure 13.

```
[{
  "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"
}]
```

Figure 13

5.1.2 Stock Quote API

The API to retrieve Stock Quote should be:

<http://dev.markitondemand.com/MODApis/Api/v2/Quote/json?symbol=AAPL>

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

```
{
  "Status": "SUCCESS",
  "Name": "Apple Inc",
  "Symbol": "AAPL",
  "LastPrice": 103.02,
  "Change": 1.52,
  "ChangePercent": 1.4975369458128,
  "Timestamp": "Fri Mar 4 15:59:00 UTC-05:00 2016",
  "MSDate": 42433.6659722222,
  "MarketCap": 571202940660,
  "Volume": 3779663,
  "ChangeYTD": 105.26,
  "ChangePercentYTD": -2.12806384191527,
  "High": 103.74,
  "Low": 101.38,
  "Open": 102.38
}
```

Figure 14

5.1.3 Interactive Chart API

The API to retrieve the data for a range of data is by passing values to a parameter in JSON format [http://dev.markitondemand.com/MODApis/Api/v2/InteractiveChart/json?parameters={\"Normalized\":false,\"NumberOfDays\":1095,\"DataPeriod\":\"Day\",\"Elements\":\[{\"Symbol\":\"AAPL\",\"Type\":\"price\",\"Params\":\[\"ohlcv\"\]}\]](http://dev.markitondemand.com/MODApis/Api/v2/InteractiveChart/json?parameters={\)

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

```
{
  "Normalized": false,
  "NumberOfDays": 1095,
  "DataPeriod": "Day",
  "Elements": [{
    "Symbol": "AAPL",
    "Type": "price",
    "Params": ["ohlcv"]
  }]
}
```

Figure 15

A sample data that is retrieved is as follows:

```

{
  "Labels": null,
  "Positions": [0, 0.004, 0.008, 0.012, 0.016, 0.02, 0.024, 0.028, 0.031, 0.035, 0.039, 0.043, 0.047, 0
  "Dates": ["2015-03-06T00:00:00", "2015-03-09T00:00:00", "2015-03-10T00:00:00", "2015-03-11T00:00:00",
  "Elements": [{
    "Currency": "USD",
    "TimeStamp": null,
    "Symbol": "AAPL",
    "Type": "price",
    "DataSeries": {
      "close": {
        "min": 93.42,
        "max": 132.65,
        "maxDate": "2015-04-27T00:00:00",
        "minDate": "2016-01-27T00:00:00",
        "values": [126.6, 127.14, 124.51, 122.24, 124.45, 123.59, 124.95, 127.04, 128.47, 127.495
      }
    }
  }]
}

```

Figure 16

The values of the parameter should be:

- Number of Days should be 3 years
- Data Period is Day
- Elements Symbol should be the symbol which was searched
- Elements Type is price
- Elements Params is "ohlc" – open high low and close

5.2 HighStocks API

The HighStocks API renders the data retrieved via the Interactive Stock API. A sample implementation of the high charts API is in the links below.

The chart type that you would be using is the area chart, the documented here:

<http://www.highcharts.com/stock/demo/area>

The HighStocks documentation of the Markit on Demand can be found here:

<http://markitondemand.github.io/DataApis/InteractiveChartSample/>

Here is an example of the Time Series Implementation of the above displayed interactive chart on MarkitonDemand:

<https://github.com/markitondemand/DataApis/blob/master/MarkitTimeseriesServiceSample.js>

There is no AJAX call specifically for HighStocks API, the data is retrieved from the Interactive Stock API and rendered using HighStocks scripts.

5.3 Yahoo Daily Stock Chart

The Yahoo daily stock chart is a straight forward implementation. You need to construct an URL as in this example:

<http://chart.finance.yahoo.com/t?s=AAPL&lang=en-US&width=400&height=300>

The parameters of the daily stock charts are:

- s – symbol of the company
- lang – always en-US
- width – the width of the image
- height – the height of the image

You can change the value of the width and height to suit your requirement. This chart would be used in two places: the **current stock tab** and in the **facebook feed dialog**. The URL returns an image of the current day stock, so you can use it directly in your image tag no need of explicit AJAX call.

5.4 Google News Feed

For the Google news feed, the service is very simple to call. Refer to the Google developer page:

<https://developers.google.com/news-search/v1/jsondevguide>.

Even though the API is deprecated, you can still query for the results.

<https://ajax.googleapis.com/ajax/services/search/news?v=1.0&q=AAPL&userip=INSERT-USER-IP>

The parameter to be used are

- v – version which is always 1.0
- q – query for the news feed, the symbol of the company
- User IP – the IP address of your site which would your local server (your internet IP) or www-scf.usc.edu or cs-server.usc.edu. This is a mandatory field.

5.5 Bing Search API

To use the Bing Search API, you need to register for the primary account key to use the functionality. Follow the steps below to use the Bing Search API

- Go to <https://datamarket.azure.com/home>
- Click on Sign In and login with your Microsoft credentials. If not sign up for a new one.

Registration

Please enter your information to create a Microsoft Azure Marketplace account.
Your privacy is important to us! For more information, check out our [privacy statement](#).

ACCOUNT DETAILS

* First name

* Last name

Organization

* E-mail address

samualkrish@live.in

Country / Region

United States

?

Language

English

☐ I agree that Microsoft may use my email address to provide information and offers regarding Microsoft Azure Marketplace.

- Accept the Terms of use and Click on Register


[Previous](#)

☒ I accept the Terms of Use

REGISTER

- Then visit the link <https://datamarket.azure.com/dataset/bing/search>
- Click on signup on the first option 5000 transaction/month which is a free version (\$0.00)

Sign Up



BING SEARCH API
Published By Microsoft
[View Publisher Offer Terms](#) | [View Publisher Privacy Policy](#)

Your Selection:
Price:

Subscription with 5,000 Transactions
Free, starting on 3/24/2016.
You can cancel your service at any time by visiting <https://datamarket.azure.com/account/datasets>.

Offers are licensed to you by the Publisher under Publisher's Offer Terms. Microsoft provides no rights or licenses for third party Publisher Offers.


☒ I have read and agree to the above publisher's Offer Terms and Privacy Policy.

[Cancel](#)


SIGN UP

- Then go to the following link <https://datamarket.azure.com/account/datasets> and click on Use

My Data

Details	Type	Status	
 <div>Bing Search API Microsoft (Started on 3/24/2016)</div>	Subscription 5,000 Transactions/month	Active 4,994 Transactions remaining	Use Show Terms Cancel

- Copy the Primary Account Key which will contain the credential to use the news API



Bing Search API
 The Bing Search API enables developers to embed search results in applications or websites using XML or JSON. Access web, image, news and video results as well as related searches and spelling suggestions.
[4,994 Transactions remaining](#)
[Primary Account Key Show](#)

URL for current expressed query:
<https://api.datamarket.azure.com/Bing/Search/v1/News>

- A URL should be constructed as follows

`https://api.datamarket.azure.com/Bing/Search/v1/News?Query=%27<your_symbol>%27&$format=json`

- A sample URL is as below
[https://api.datamarket.azure.com/Bing/Search/v1/News?Query=%27AAPL%27&\\$format=json](https://api.datamarket.azure.com/Bing/Search/v1/News?Query=%27AAPL%27&$format=json).
 You have to provide username and password credentials to query the result. The **Primary Account Key** acts as both username and password
- It is recommended that you call the API via PHP
 - To get started with the implementation this [migration guide](#) will be to help

5.6 Facebook API

When clicking the Facebook post button, a Facebook post must be executed. In particular, when the button is pressed, the web application does the following:

- Displays 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;
- Posts an Update Status message to the feed.

The above two steps can be performed using the Facebook Connect API, using the JavaScript SDK, which provides a rich set of client-side functionality for accessing Facebook's server-side APIs. It is documented at:

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

The REST call will display the appropriate values for title, caption, description and icon image as shown in Figure 9. The hyperlink on the post should link to <http://dev.markitondemand.com/>.

Once the post has been published, you should show an alert box informing the user of whether the post has been published successfully or not, as shown in Figure 17.

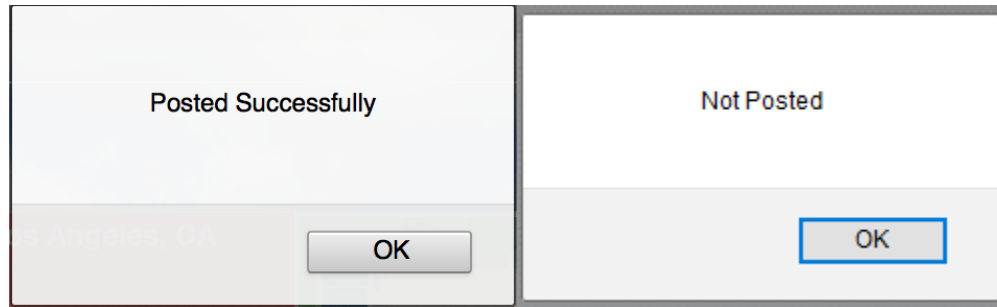


Figure 17

6. External Libraries

The list of external libraries that you use are

- JQuery – <https://code.jquery.com/>
- JQuery UI – <https://code.jquery.com/>
- Bootstrap – <http://getbootstrap.com/getting-started/>
- Bootstrap Toggle – <http://www.bootstraptoggle.com/>
- HighStocks – <http://code.highcharts.com/>
- Moment JS - <http://momentjs.com/> for time conversion
- Facebook - <https://developers.facebook.com/docs/javascript>

7. Implementation Hints

7.1 Images

The images for this homework will be present in <http://cs-server.usc.edu:45678/hw/hw8/images/>

7.2 Get started with the Bootstrap Library

To know how to get started with Bootstrap, 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. You will at a minimum need to use Bootstrap Form, Tab, Wells, Carousel 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 Carousel <http://getbootstrap.com/javascript/#carousel>
Bootstrap Glyphicons <http://getbootstrap.com/components/#glyphicons>

7.4 Google App Engine/Amazon Web Services

You should use the domain name of the Google App Engine/Amazon Web Services you created in HW#7 to make the request. For example, if you're GAE/AWS server domain is called `example.appspot.com` /`example.elasticbeanstalk.com`, and the user performs a GET request with parameter name="AAPL", then 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 AJAX call

You can send the request to the PHP file by passing the URL to `$.ajax()`. 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 code is running on Google GAE (Please refer to the grading guideline for details).

The 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 18

7.6 JQuery autocomplete

You have to implement a JQuery autocomplete feature that would load the data on each key stroke. The sample implementation of the `$.autocomplete` can be found here:

<https://jqueryui.com/autocomplete/#remote-jsonp>

7.7 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 method `getItem()` and `setItem()`. The local storage could only store strings. So need to convert the data to string before storing it in the local storage.

<https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage>

http://www.w3schools.com/html/html5_webstorage.asp

8. FAQ's

1. Which server should I use, where would server be hosted?
You can use either host it in Google App Engine or Amazon Web Services. While grading we just need to see the data
2. Can we use external libraries?
Only the libraries given in Section 6.
3. What is the threshold for Market Cap suffixes?
A value which is greater than or equal to 1 Billion the suffix will be Billion similarly a value greater than or equal to 1 Million the suffix will be Million, otherwise just the number.
4. When you trigger autocomplete functionality?
On every key that is entered in the input field.
5. Can I use any other way to store data?
You are allowed only to use browsers local storage. No other way of storing data like cookies, session storage is allowed.
6. How long should the local storage be persistent?
The local storage data should be persistent until the browser is closed. Even if the page is refreshed, the local storage data should not be deleted.
7. Table contents on Auto Refresh functionality?
You should not recreate the entire table. You just need to update the value of Price and Change column.
8. Is it required to use Bootstrap Carousel?
Yes. It is mandatory to use bootstrap carousel. The carousel should not slide (you will understand while implementing) to other section automatically.
9. Facing issues with Cross Origin Request?
*There are several ways to avoid CORS, you can route all your request through the server. If you like to use `$.ajax()` you can use **datatype: jsonp** with callback function appended with the URL.*

9. Files to Submit

In your course homework page, you should update the HW8 link to refer to your new initial web page for this exercise. Additionally, you need to provide a link to your homework page, and a link to the GAE/AWS server where the AJAX call is made with a sample parameter value. Also, submit your files electronically to the csci571 account so that they can be graded and compared to all other students' code.

****IMPORTANT**:**

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.