Table of Contents

[**Requirements Mapping** 2](#_Toc465090833)

[**Design** 3](#_Toc465090834)

[**Stand Alone Java Program Object Oriented Design** 3](#_Toc465090835)

[**Web Application with Rest API Object Oriented Design** 3](#_Toc465090836)

[**Implementation Details** 4](#_Toc465090837)

[**Explanation:** 4](#_Toc465090838)

[**Version Control (GIT)** 4](#_Toc465090839)

[**Continuous Integration (Travis CI)** 5](#_Toc465090840)

[**Logging (Log4j)** 6](#_Toc465090841)

[**Unit Testing (JUnit)** 7](#_Toc465090842)

[**Testing Restful API (Restassured Test)** 7](#_Toc465090843)

[**Implemented System User Guide** 8](#_Toc465090844)

[**Get Company List (GET)** 8](#_Toc465090845)

[**Add Company (POST)** 9](#_Toc465090846)

[**Delete Company (DELETE)** 10](#_Toc465090847)

[**Get Company History (GET)** 11](#_Toc465090848)

[**Front End Visualization (d3.js, JavaScript, jQuery, HTML)** 12](#_Toc465090849)

## **Requirements Mapping**

From the given documents software requirements are constructed for the required Standalone Application and Web Application with Restful API.

**Supp1:** Find an API which helps to get stock value of a company

* Supp1.1: API should return current stock value when company code is given
* Supp1.2: API should return full company name when company code is given

**Supp2:** Create a Database where companies and stock history of company can be stored

* Supp2.1: When a company is deleted all its corresponding stock values in data base should be deleted
* Supp2.2: There should be some primary key for each table created in database
* Supp2.3: Primary Key, Foreign key constrains should be properly given when creating database tables

**Supp3:** Create job which executes after every five minutes and adds current stock values to data base for all the registered companies.

**Supp4:** Create Restful API to get the stock history of a company

* Supp4.1: When company is not found in data base http response status should be 404 (Not found)
* Supp4.2: When company is found in data base http response status should be 200(OK) and should send JSON which contains name of the company and all the stock history.

**Supp5:** Create Restful API to get the list of companies in database

* Supp5.1: When data base is empty. Response status should be 204 (No Content)
* Supp5.2: When request is successful and data base has data. Then Response status should be OK (200) and should have list of companies in response body.

**Supp6:** Create a Restful API which can add company name to database

* Supp6.1: When requested company is valid and not present in data base. Add this company to database. Response Status should be 201 (Created)
* Supp6.2: When requested company is already in database. Response status should be 403 (Forbidden)
* Supp6.3: When requested company\_code is not a valid one. Then Response status should be 400 (Bad Request)

**Supp7:** Create a Restful API which deletes company in database

* Supp6.1: When requested company is not present in data base. Response Status should be 404 (Not Found). No action should be performed in database.
* Supp6.2: When requested company is in database. This company should be deleted in data base and Response status should be 200 (OK)

**Supp8:** Implement a logger which logs error messages or any required messages.

**Supp9:** Create a front end interface which shows stats of company stocks

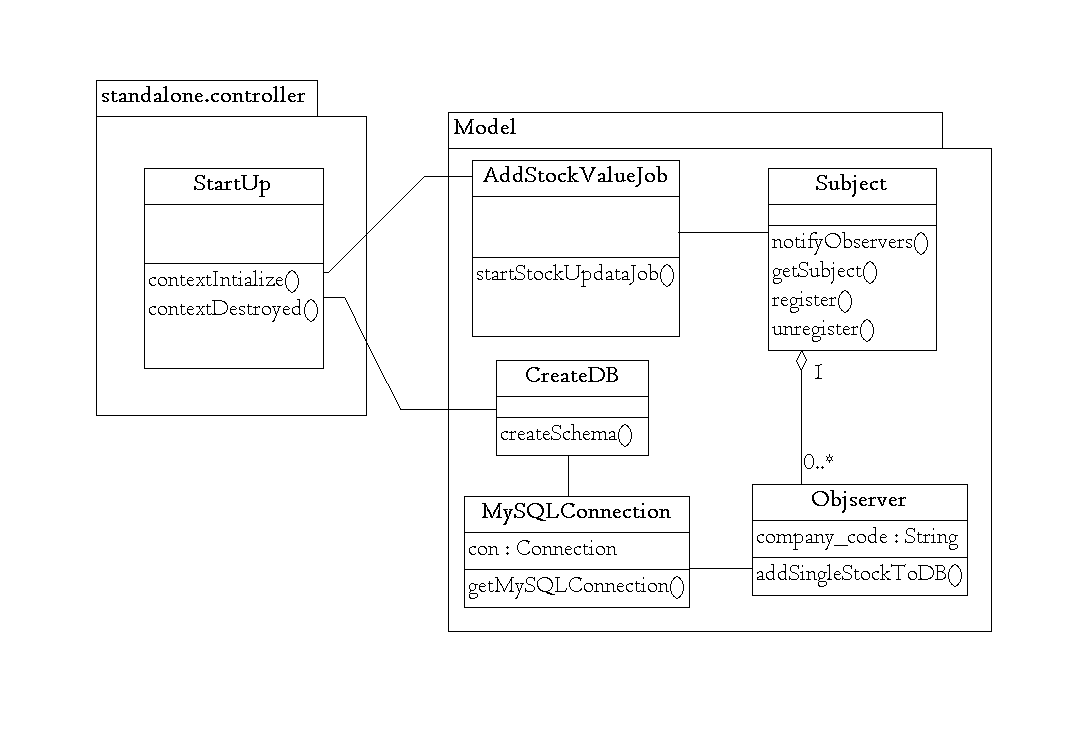
* Supp9.1: Create a time scaled area chart. X-axis represents time and Y-axis represents stock value in dollars.
* Supp9.1: On mouse over on the graph chart. Corresponding stock value and time should be shown. On mouse out this section should be hidden.

## **Design**

### **Stand Alone Java Program Object Oriented Design**

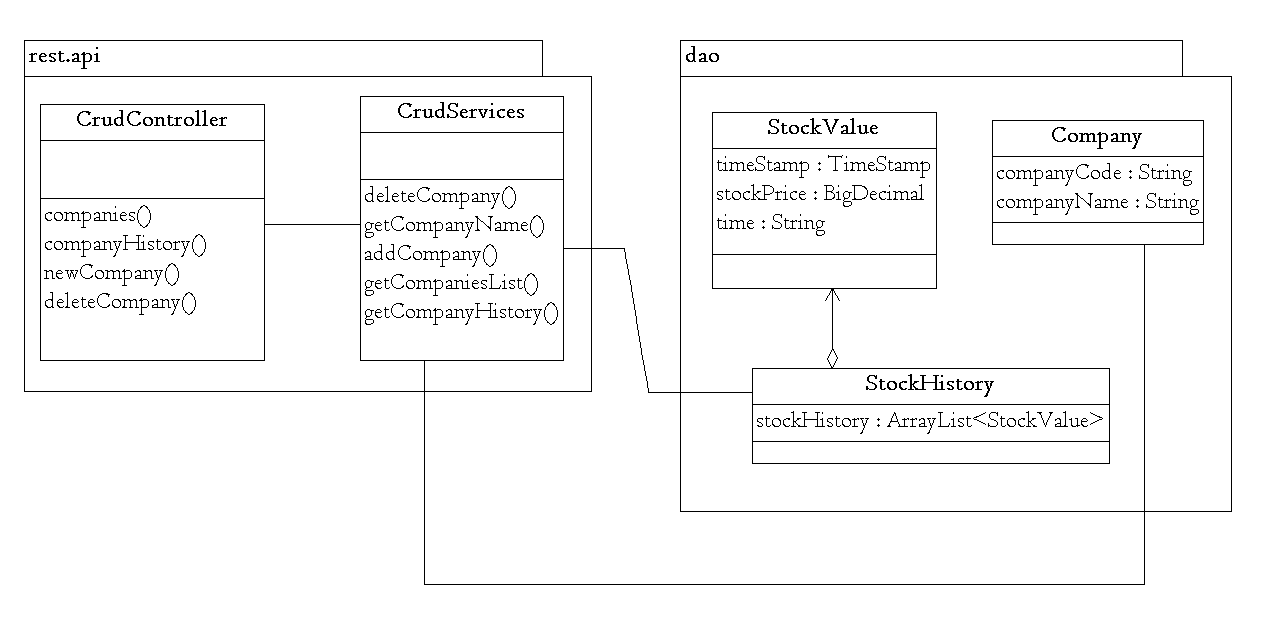
This application should get the stock values for all the companies in data base every 5 minutes and should update database.

Implemented this with **Observer Pattern**. Used observer pattern because in this application addition and deletion of companies should be performed. If observer pattern is used there is no need to change or changes in code will be very minimum when implementing restful API in web application. Components are loosely coupled so modification of code will be easy and will be specific to only one or two modules.

****

In this program MySQLConnection and Subject connection instances are only needed once. So Used Single Pattern for both these classes.

### **Web Application with Rest API Object Oriented Design**



## **Implementation Details**

### **Explanation**

One thread is executed in background and after every 5 minutes it notifies subject and subject notifies each of its observers.

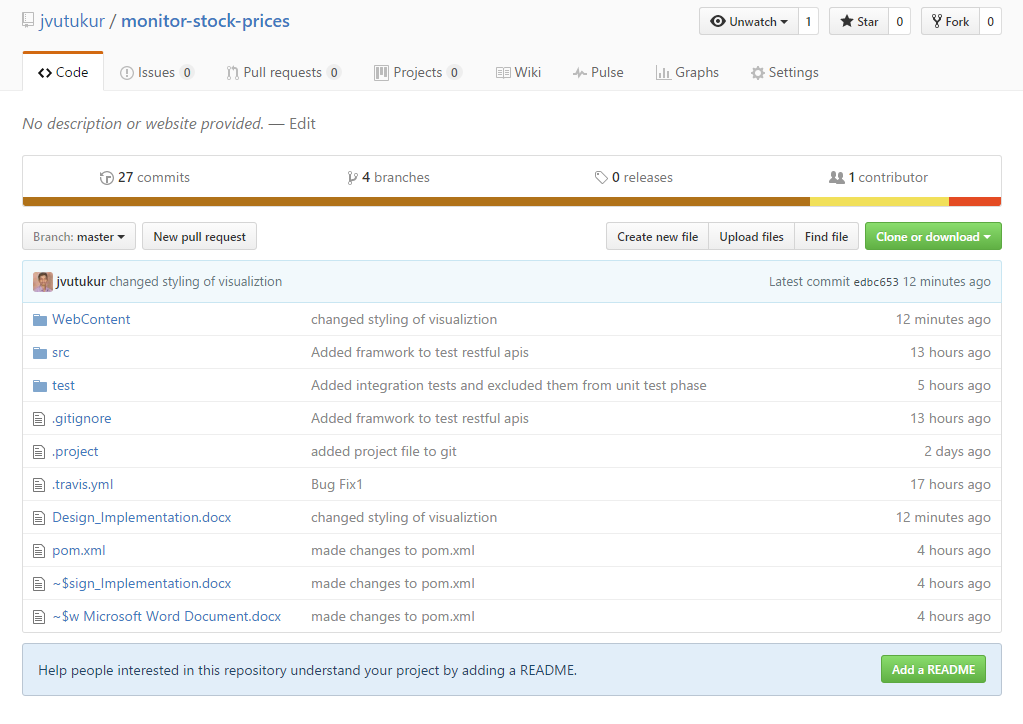
When a company is deleted from data base. Observer is unregistered from Subject.

When a new company is added to data base. A new observer is created and is registerd to Subject

### **Version Control (GIT)**

Used Git for version control. Created branches for tasks and after competing each task merged with master branch.

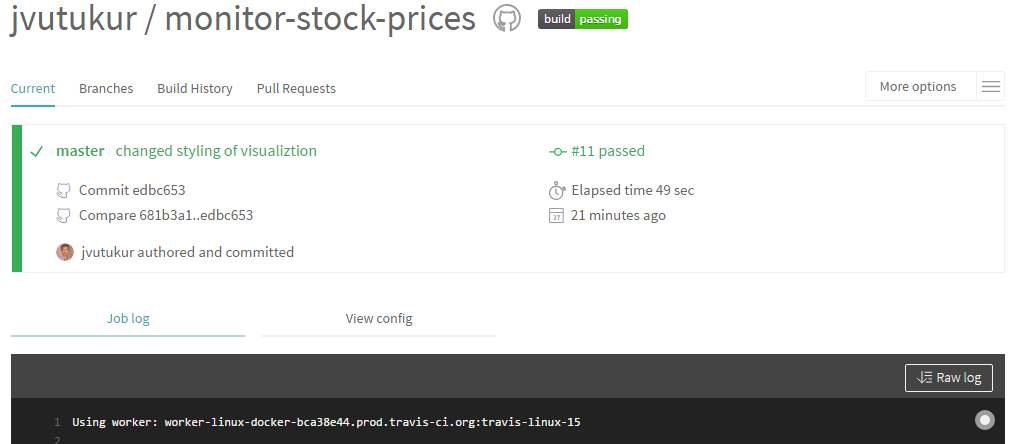
Github Repo URL: <https://github.com/jvutukur/monitor-stock-prices/>



When working on branches. Staged files which are only necessary and pushed only valid changes to maintain a clean repository. Which will not fail build.

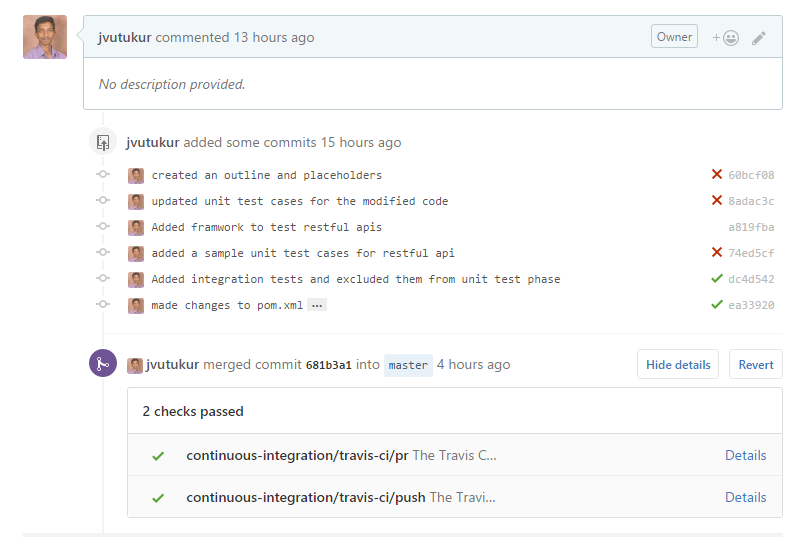
### **Continuous Integration (Travis CI)**

Used **Travis CI** for continuous integration.

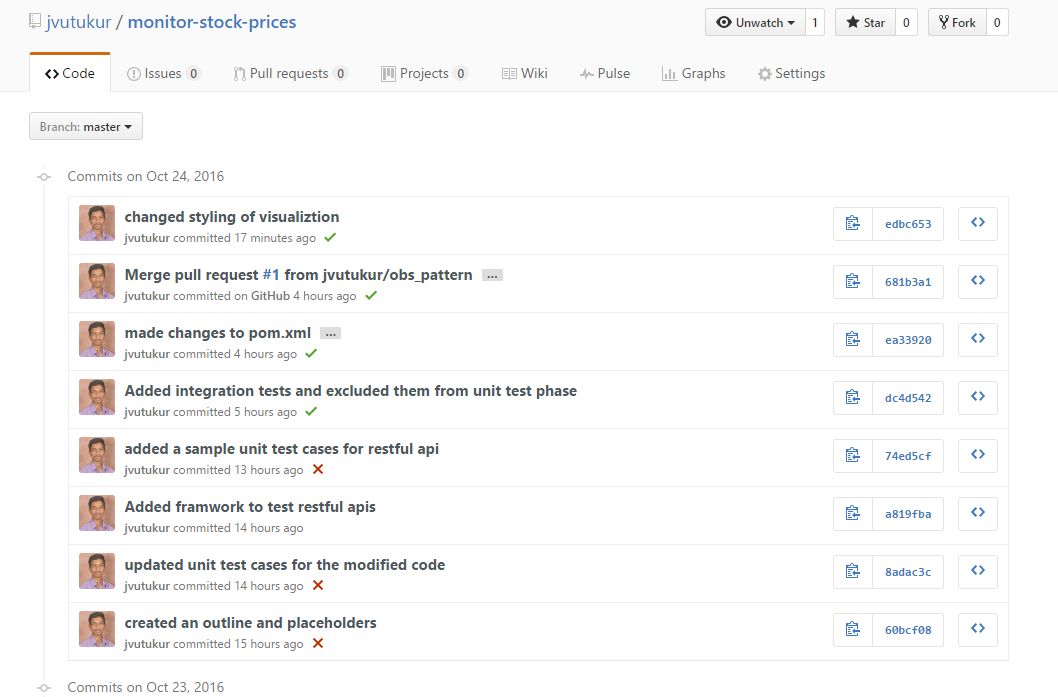


Travis CI Repo Url: <https://travis-ci.org/jvutukur/monitor-stock-prices>

• After every pull request from branches. Travis CI runs all the unit tests and will tell with this become a proper merge.



• After push, merge Travis CI runs all the unit test cases and if all unit test cases are passed then a build is generated.



### **Logging (Log4j)**

Used Log4j library for logging errors, and info. Used Current location of project execution as log location.

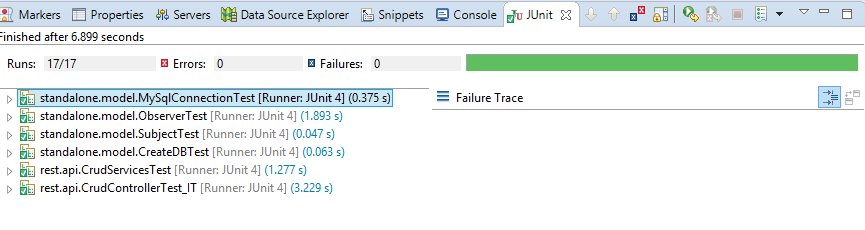
System.*setProperty*("rootPath", context.getRealPath("/"));

Logs are created in eclipse workspace \MonitorStockPrice\WEB-INF\logs.

For each request to Restful API action log is inserted which helps to track issues in application during run time.

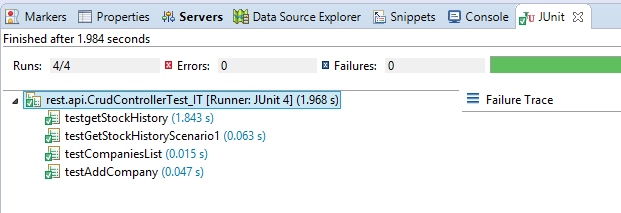
### **Unit Testing (JUnit)**

Added **Junit** test cases for all service layer modules. For each module possible success scenarios and failure scenarios are tested



### **Testing Restful API (Restassured Test)**

Added **Restassured** test cases to test the restful API. Restassured test cases are extension of Junit test cases.

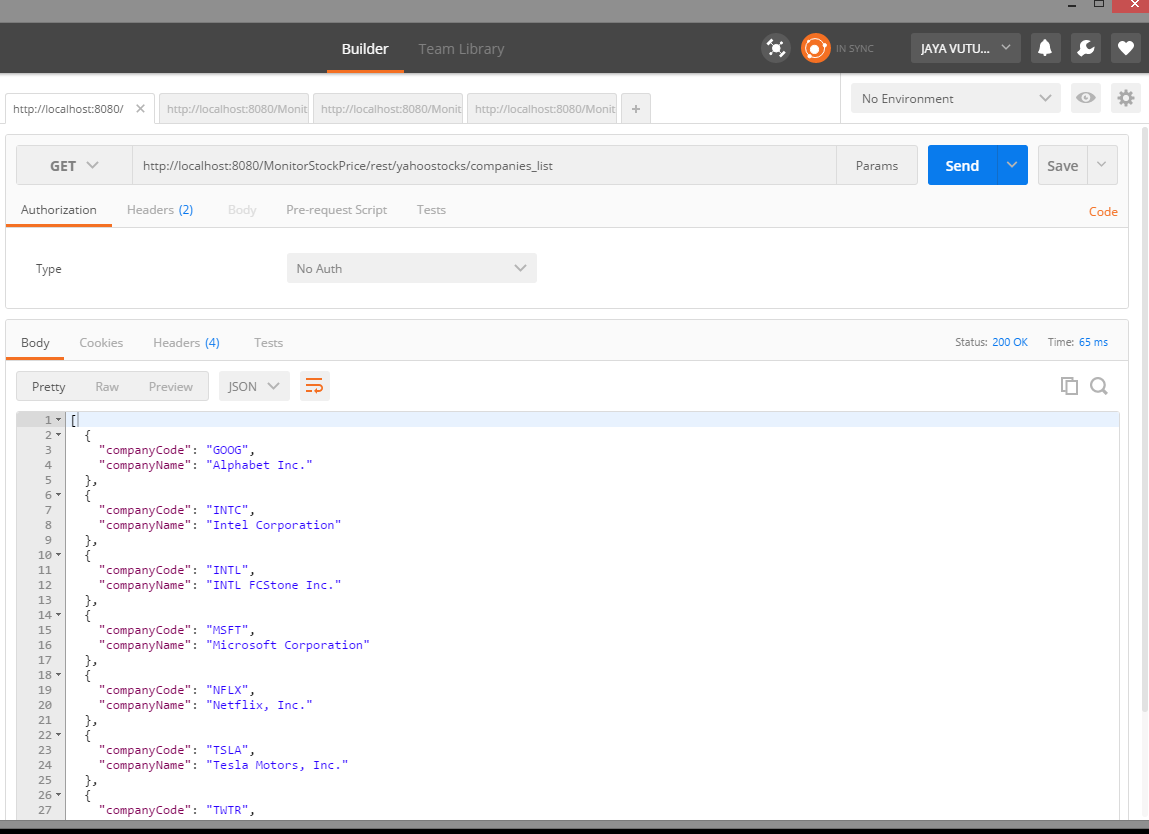


## **Implemented System User Guide**

### **Get Company List (GET)**

Url: <http://localhost:8080/MonitorStockPrice/rest/yahoostocks/companies_list>

Request Type: GET



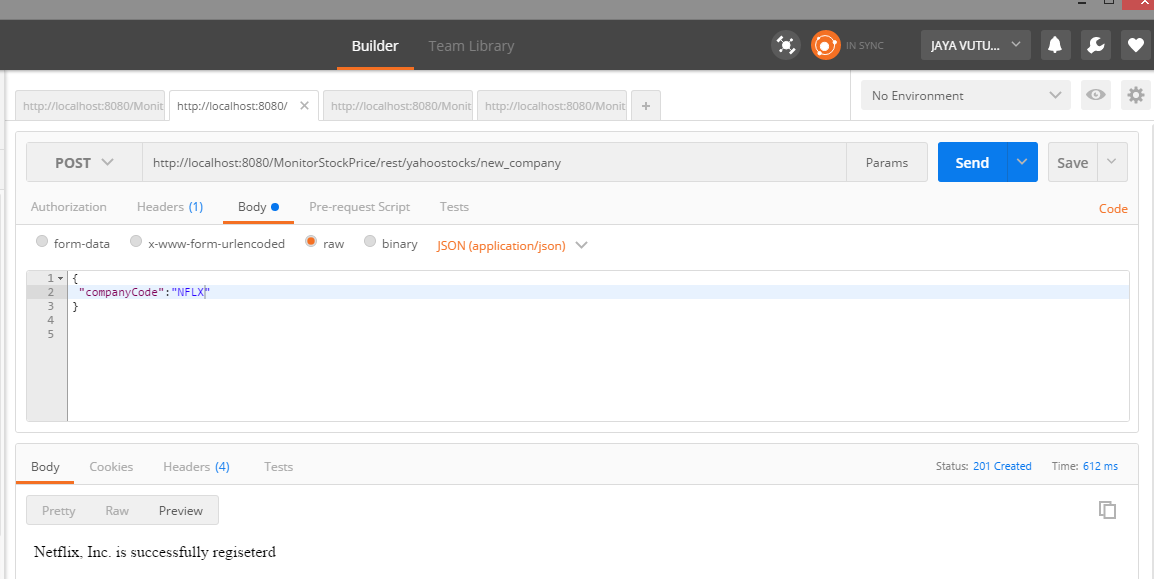
### **Add Company (POST)**

Url: <http://localhost:8080/MonitorStockPrice/rest/yahoostocks/new_company>

Request Type: POST

Input Type: application/json

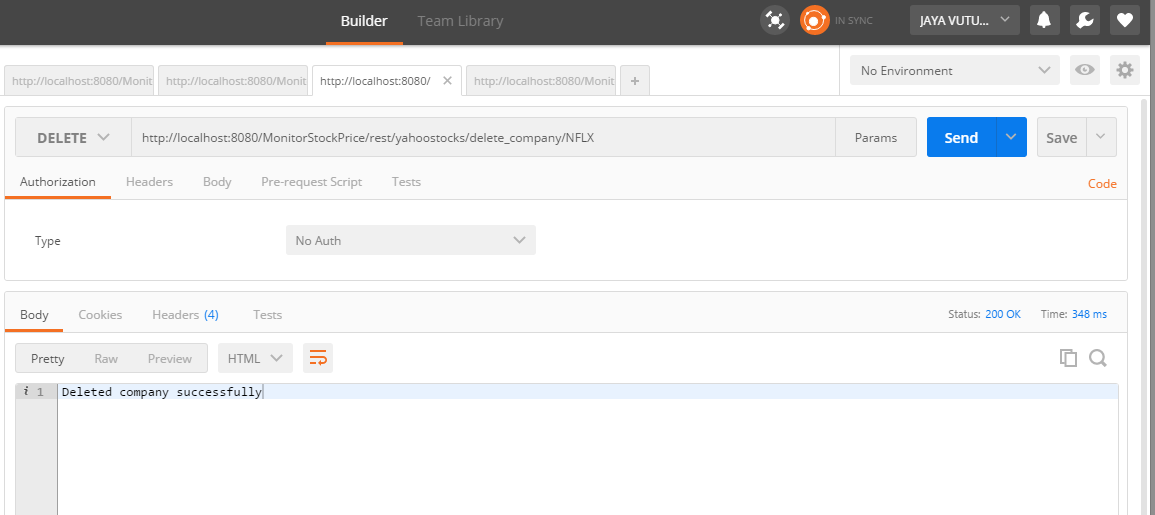
JSON body sample input {“companyCode":"NFLX”}



### **Delete Company (DELETE)**

Url: <http://localhost:8080/MonitorStockPrice/rest/yahoostocks/delete_company/NFLX>

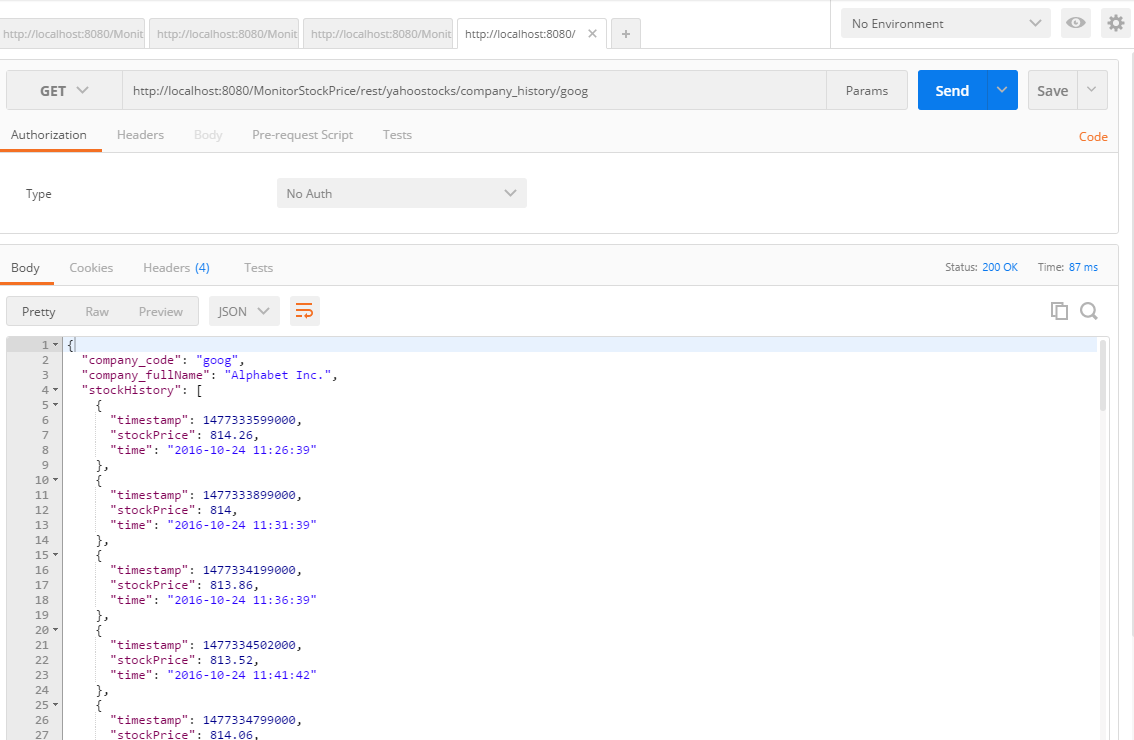
Request Type: DELETE



### **Get Company History (GET)**

Sample Url: <http://localhost:8080/MonitorStockPrice/rest/yahoostocks/company_history/goog>

Request Type: GET

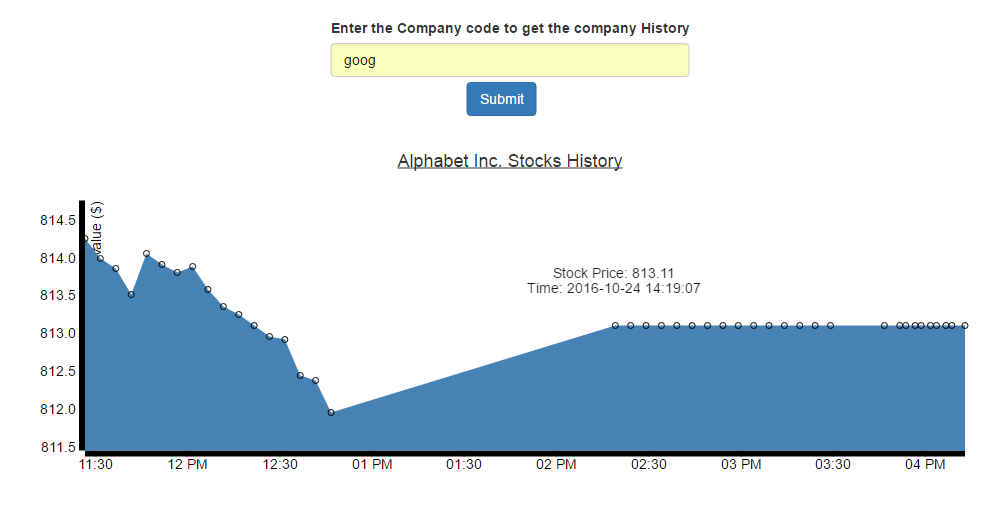


### **Front End Visualization (d3.js, JavaScript, jQuery, HTML)**

Used d3.js for plotting svg tags, JavaScript, jQuery for making ajax calls and updating components on html page. Mouse over on each small bubble indicates shows more details of the stock value.

Url: <http://localhost:8080/MonitorStockPrice/index.html>

If application is not running on localhost:8080 – url in jScript file should be changed



Not exist

http://localhost:8080/MonitorStockPrice/rest/yahoostocks/new\_company