Software Engineering Final Project Fall 2019

CPS 5301-02/4301-01

TABLE OF CONTENTS

1 Introduction 1

1.1 Purpose 1

1.2 Scope 1

1.3 Overview 1

2 System Overview 2

2.1 System Characteristics 2

2.2 System Architecture 2

2.3 Infrastructure Services 3

3 System Context 4

3.1 Base API link 4

3.2 Sandbox testing link 4

3.3 Historical prices for a stock 4

3.4 Company description 7

4 DATA DESIGN 9

4.1 DATA DESCRIPTION 9

4.2 DATA DICTIONARY 9

5 Component Description 11

5.1 Initialize database with 5 years’ worth of data 11

5.2 Initialize company data 11

5.3 Retrieve stock information daily 12

5.4 Spring boot rest api endpoints 13

6 SOFTWARE DEVELOPMENT CYCLES 14

# Introduction

This is the final project for Software Engineering Class and it will include development of the system that will use IEX API to retrieve, store data and do full reporting on it based on requirements that will be presented in this document. The system shall be able to retrieve data, make sure the consistency of data is complete, perform daily updates of certain parts of the system and report and present the scrapped data in a way that’s easily readable by humans.

This design document presents the designs used or intended to be used in implementing the project. The designs described, follow the requirements specified in the Software Requirements Specifications document prepared for the project.

## Purpose

##### The purpose of this document is to provide complete documentation regarding the process of developing the required system using agile software development methodology. It will contain both quick overview of the system as well as full and detailed descriptions of every component within it.

##### Primarily, the document is written for members of Team 1 who will use the written specification and provided design to develop required system. This will be the guideline for developers in the process of implementing the project.

## Scope

##### This document will provide detailed description of the software architecture of the final project’s system. The system itself will consist of various modules that will be developed independently but will be integrated into one big system in the end. Besides detailed description of these modules, this document will also describe data models of the system and provide a couple of class, state and system architecture diagrams within it.

## Overview

##### Chapter 1 represents the introduction of the document and provides short description of the project.

##### Chapter 2 will provide system overview with special focus on the environment where the system will be developed and run.

##### Chapter 3 provides the context in which the system runs and describes the external services that system use in communication and/or retrieval of data.

##### Chapter 4 goes into details of database design and dictionary

##### Chapter 5 describes into details components that have to be developed.

##### Chapter 6 describes the organization of work into sprints

# System Overview

##### In this chapter, quick overview of the entire system will be presented, with the focus on the way it’s organized and with provided technical details of hardware configuration.

## System Characteristics

##### Development of the system will be conducted in a way that it’s going to be easily customizable and flexible for future modification and development. System will communicate with external services (IEX API) to obtain data and update such data on a daily basis and will provide analysis and reports through RESTful API and custom made UI design based on that acquired data, that is going to be stored in a local database. Entire system will be developed as a web application with the usual MVC (model-view-controller) layer architecture, which will be explained in more details in the next chapter.

##### Since it’s expected to work as a web application, the entire system will be set up that way so it can be accessible from any point and using any type of device, including laptops, tablets and mobile phone. To accomplish this scalability, a whole set of different frameworks and programming languages will be applied.

## System Architecture

##### Below is the diagram that describes system architecture in general.

##### 

##### Database and Scraper are hosted on Microsoft Azure Server #1. Quick overview of the server:

Server is hosted on Microsoft Azure Cloud Platform.

Subscription: Azure for Students

Disk: 30GiB SSD

Operating System: Linux (ubuntu 18.04)

Size: Standard B1s (1vcpus, 1Gib memory)

Location: East US

##### Database and Scraper are hosted on Microsoft Azure Server #2. Quick overview of the server:

Server is hosted on Microsoft Azure Cloud Platform.

Subscription: Azure for Students

Disk: 30GiB SSD

Operating System: Linux (ubuntu 18.04)

Size: Standard B1s (1vcpus, 1Gib memory)

Location: East US

##### When conducting our test to prepare the front-end, we concluded that having everything on once instance was not feasible to server the information efficiently. Therefore, we spread out all of the components to 2 servers with similar specifications. If we had known of the requirements at the beginning, a more powerful instance would have been initiated at the start of the project.

##### For database server, we will be using MySQL Database (accessible through port 3306)

##### For REST Server, we will be using Java EE framework Spring Boot that will communicate with MySQL and present data to front end UI.

##### For front end UI (the actual application visible by end user) we will be using JavaScript framework Angular.

##### General workflow of data works as described on the diagram above:

##### End user (client, anyone accessing web application through any type of supported devices) will request 30-day stock information through developed UI. That request is being forwarded through Microsoft Azure Platform to REST Server that will contain the logic behind RESTful API. For development of RESTful API, we will use Spring Boot framework. Upon receiving the request, REST server will query Database Server to get 30-day stock information. After MySQL DB server provides response back to REST Server, it will perform required statistical analysis on rows returned and build JSON response that will be forwarded back to front end UI and presented in a visually appealing way to the end user who made the initial request.

##### Besides this workflow, we will implement daily script in Python that will query IEX API and scrape the data to store it in our local database and perform daily updates of that data in our database.

##### All the necessary documentation regarding server, including server deployment and credentials is uploaded can be found on GitHub.

## Infrastructure Services

##### As described in chapter before, system is running on Microsoft Azure Cloud Platform, on Ubuntu OS.

##### To ensure security of the entire system, proper procedures for login and overall security of the projects have been enforced. Some of those include (but not limited to):

##### Only certain set of standard ports being opened between the 2 instances:

1. MySQL: 3306 (Opened for testing for each developer)

2. SSH: 22 (Used for connecting via SSH)

3. HTTP: 80 (Used for webserver)

4. HTTPS: 443 (Forwarded in case of inclusion of SSL)

5. Spring Boot: 8080

6. Angular: 4200

##### Other ports on the server are closed.

# System Context

##### One of the main characteristics of the system is that initializes the initial dataset and updates daily data by consuming IEX API. IEX API is external system that offers a whole variety of the endpoints. In this chapter, we will present and document only those endpoints that are going to be used for the scope of the project (based on the list of functionalities that we have)

## Base API link

**https://cloud.iexapis.com/**

## Sandbox testing link

**https://sandbox.iexapis.com/**

## Historical prices for a stock

Do NOT use unadjusted data prefixed with u | Project #5

**GET stock/{symbol}/chart/{range}/{date}**

| **{range}** | **Description** | **Project Relevancy** |
| --- | --- | --- |
| 5y | 5 years with daily reports | #6 |
| 1m | 1 month with daily reports |  |
| 1mm | 1 month with reports for every 30 mins |  |
| 5d | 5 day with daily reports | #9 |
| 5dm | 5 day with reports every 30 mins |  |
| 1d | 1 day with per minute reports |  |

Example for **5y** and **5d**

| **Req** | **Key** | **Value** |
| --- | --- | --- |
| Y | "date" | "2019-10-15" |
| Y | "open" | 1221.5 |
| Y | "close" | 1242.24 |
| Y | "high" | 1247.12 |
| Y | "low" | 1220.92 |
| Y | "volume" | 1527216 |
| N | "uOpen" | 1221.5 |
| N | "uClose" | 1242.24 |
| N | "uHigh" | 1247.12 |
| N | "uLow" | 1220.92 |
| N | "uVolume" | 1527216 |
| Y | "change" | 24.47 |
| Y | "changePercent" | 2.0094 |
| N | "label" | "Oct 15" |
| Y | "changeOverTime" | 0.020094 |

Example for **5dm** and **1d**

| **Req** | **Key** | **Value** |
| --- | --- | --- |
| Y | "date" | "2019-10-18" |
| TBD | "minute" | "09:30" |
| TBD | "label" | "09:30 AM" |
| Y | "high" | 1254.79 |
| Y | "low" | 1254.36 |
| Y | "open" | 1254.775 |
| Y | "close" | 1254.36 |
| Y | "average" | 1254.78 |
| Y | "volume" | 127 |
| TBD | "notional" | 159357.095 |
| TBD | "numberOfTrades" | 4 |

**GET stock/{symbol}/chart/date/{date}**

* {date} format is YYYYMMDD
* returns per minute reports
* append with flag chartByDay=true for historical OHLCV data

## Company description

**GET /stock/{symbol}/company**

* returns company name, exchange, industry, description, CEO, sector, employees, address, phone number

| **Req** | **Key** | **Value** |
| --- | --- | --- |
| Y | "symbol" | "GOOGL" |
| Y | "companyName" | "Alphabet, Inc." |
| Y | "exchange" | "NASDAQ" |
| Y | "industry" | "Internet Software/Services" |
| Y | "website" | "[http://abc.xyz](http://abc.xyz/)" |
| Y | "description" | "Alphabet, Inc. is a holding company..." |
| Y | "CEO" | "Lawrence E. Page" |
| TBD | "securityName" | "Alphabet Inc. Class A" |
| TBD | "issueType" | "cs" |
| TBD | "sector" | "Technology Services" |
| TBD | "primarySicCode" | 7375 |
| Y | "employees" | 98771 |
| Y | "tags" | ["Technology Services","Internet Software/Services"] |
| Y | "address" | "1600 Amphitheatre Parkway" |
| Y | "address2" | null |
| Y | "state" | "CA" |
| Y | "city" | "Mountain View" |
| Y | "zip" | "94043" |
| Y | "country" | "US" |
| Y | "phone" | "1.650.253.0000" |

**GET /stock/{symbol}/intraday-prices**

* This returns 1 minute bar data where open, high, low, and close are per minute.
* Must complete a vendor agreement with UTP to use this API, see IEX API docs

# DATA DESIGN

##### In this chapter we will present design of database needed for the project.

## DATA DESCRIPTION

##### For data storage, as it was mentioned in chapters before, we are going to be using MySQL database. This database has been installed on Microsoft Azure Server. Since the whole system will be developed by different programming languages, we will need to use various drivers to establish successful connection to DB (for example JDBC for Java is not the same driver as it is for Python)

##### Based on the requirements for the system, we need DB with 2 tables. ER diagram of DB looks like this:

##### 

## DATA DICTIONARY

##### **Table Stock**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Null** |
| Id (primary key) | Int | No |
| Company\_id (foreign key) | Int | No |
| Date | Date | No |
| Open | Decimal | No |
| Close | Decimal | No |
| High | Decimal | No |
| Low | Decimal | No |
| Volume | Int | No |

##### **Table Company**

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Null** |
| Id (primary key) | Int | No |
| Stock\_symbol | Varchar | No |
| Company\_name | Varchar | No |
| Exchange | Varchar | No |
| Industry | Varchar | No |
| Website | Varchar | No |

# Component Description

##### In this chapter we will describe components that are to be developed as single functionalities within the system.

## Initialize database with 5 years’ worth of data

##### **Brief description:**

##### Create a script that will call proper IEX API endpoint to retrieve 5 years’ worth of data to initialize the DB. This will serve as initial state of DB.

##### **Goal:**

##### Successfully initialized DB.

##### **Precondition:**

##### Created IEX account to use API end points.

##### Structure of Database have already been created and is ready for prepopulating.

##### **Trigger:**

##### Initial state of the system: it gets triggered when we initialize the system.

##### **Detailed description (with technical details)**

##### 1. Using python for writing a necessary script.

##### 2. Use necessary driver to establish connection with MySQL database where retrieved data will be saved

##### 3. Create SQL query for insert data into DB.

##### 4. Make request to appropriate IEX API to retrieve 5 years’ worth of data.

##### 5. Loop through retrieved information and make insert into database with each retrieved raw of information.

## Initialize company data

##### **Brief Description:**

##### - Create a script that will call IEX to retrieve information regarding the companies that have a stock.

##### **Goal:**

##### - Initialize the ‘company’ table in the database.

##### **Precondition:**

##### - Have empty database table created with correct columns.

##### **Trigger:**

##### - Initial state of system.

##### **Detailed Description**

##### 1. Program is created in a python script.

##### 2. Using pymysql to establish a connection to the database.

##### 3. Using requests to establish a ‘GET’ request from IEX for the company data.

##### 4. SQL statement is created to insert data into the database.

##### 5. Request is made to IEX to retrieve company information.

##### 6. Company information is parsed and inserted into the database.

## Retrieve stock information daily

##### **Brief Description:**

##### Create a script that will retrieve information from IEX daily. Data relating to the specific stock will be stored into the database to be used by the rest api.

##### **Goal:**

##### Insert information into database daily.

##### **Precondition:**

##### Have ‘stock’ database created

##### **Trigger:**

##### Scheduler in the system

##### **Detailed Description**

##### Program is created in a python script.

##### This script will be programed to run every 24 hours

##### Using pymysql to establish a connection to the database.

##### Using requests to establish a ‘GET’ request from IEX to provide stock information for a specific stock on the day request is sent.

##### SQL statement is initialized in a variable to be inserted into the database.

##### Date is initialized to specify the day for IEX.

##### 3 requests are sent to IEX for each stock.

##### Data from IEX is parsed to prepare commit to SQL database.

##### SQL statement is committed and data is inserted into the Database.

##### If there is no value in the ‘OPEN’ column from IEX, we assume that there is no data for that stock for that day and nothing is inserted.

## Spring boot rest api endpoints

##### **Brief Description:**

##### Create a Rest API Endpoint to serve get requests.

##### **Goal:**

##### Serve a GET request to deliver information from the mySQL database.

##### **Precondition:**

##### Have Stock Table Populated

##### Have Company Table Populated

##### Have request to IEX be stored in a database daily

##### **Trigger:**

##### Web Server sends request to REST Api

##### **Detailed Description**

##### **“/IEX30”**

##### Return the stock for each company from the last 30 days in a JSON format to be parsed by the front end.

##### If there is an error in the sql, an error message will be printed on the backend.

##### **“/daterange”**

##### Returns stocks in data range specified by request.

##### Returns request in JSON format.

##### If there is an SQL error, an error message will be returned

##### If there is an exception in the parsing of the request, an error message will be returned.

##### If there is a general exception, an error message will be returned.

# SOFTWARE DEVELOPMENT CYCLES