Software Engineering Final Project Fall 2019

CPS 5301-02/4301-01

Technical Design Document

TABLE OF CONTENTS

1 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.

1.1 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.

1.2 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.

1.3 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 and provide insight into how the entire project was managed.

2 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.

2.1 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.

2.2 SYSTEM ARCHITECTURE

Below is the diagram that describes system architecture in general.

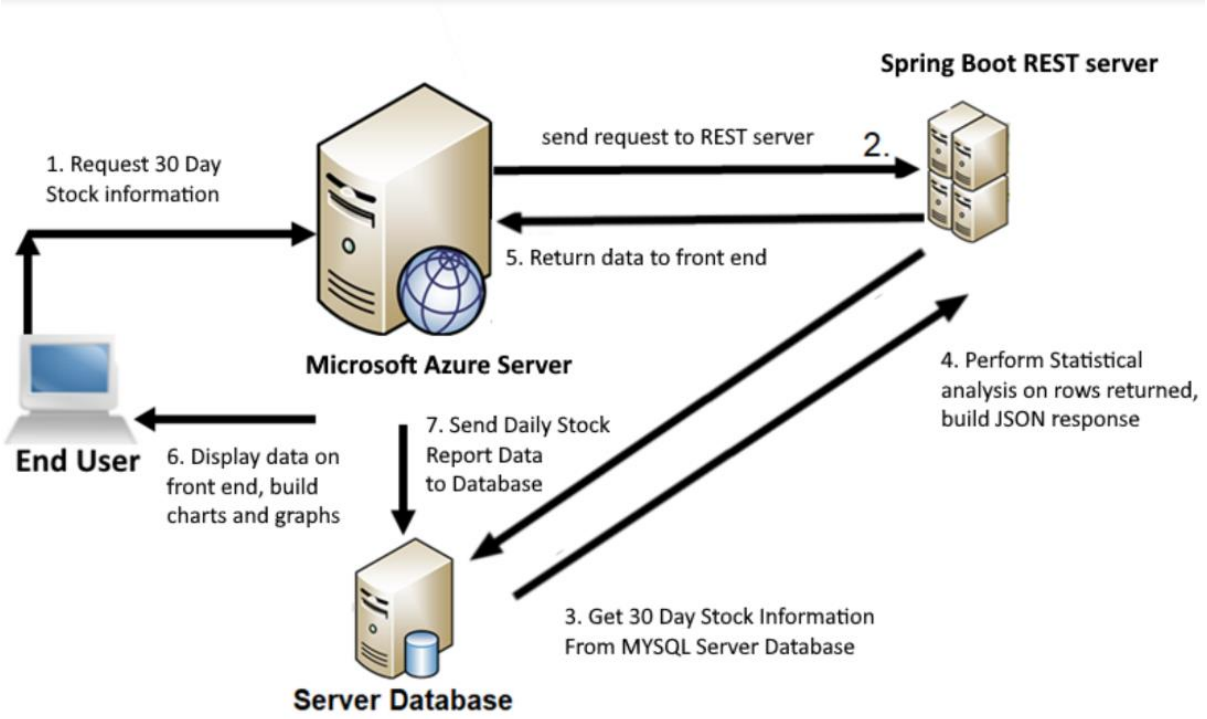

Diagram 1. – Architecture of the system

Spring Boot REST Server is 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, Scraper and front 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 Jave 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.

2.3 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:

```
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 incase of inclusion of SSL)
```

Other ports on the server are closed.

3 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)

3.1 BASE API LINK

`https://cloud.iexapis.com/`

3.2 SANDBOX TESTING LINK

`https://sandbox.iexapis.com/`

3.3 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

3.4 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"
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

Req	Key	Value
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

4 DATA DESIGN

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

4.1 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:

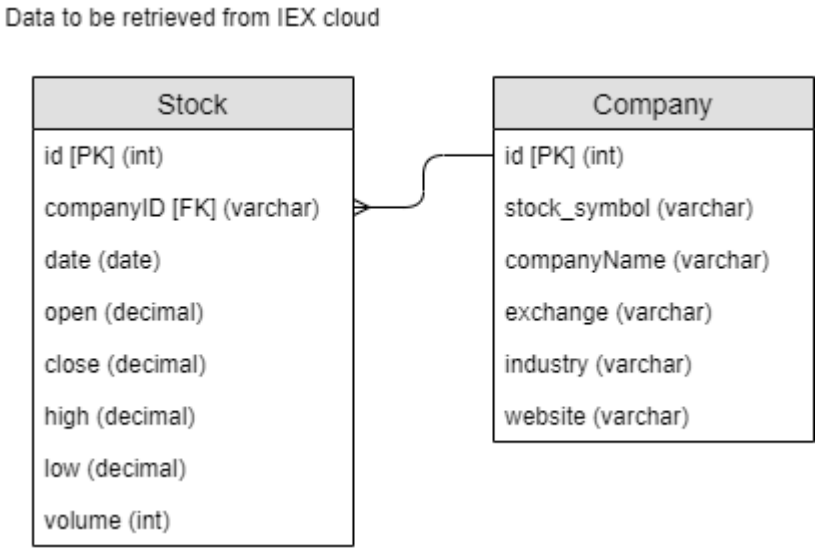

Diagram 2. – ER Diagram of DB

4.2 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

5 COMPONENT DESCRIPTION

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

5.1 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.

Related diagram

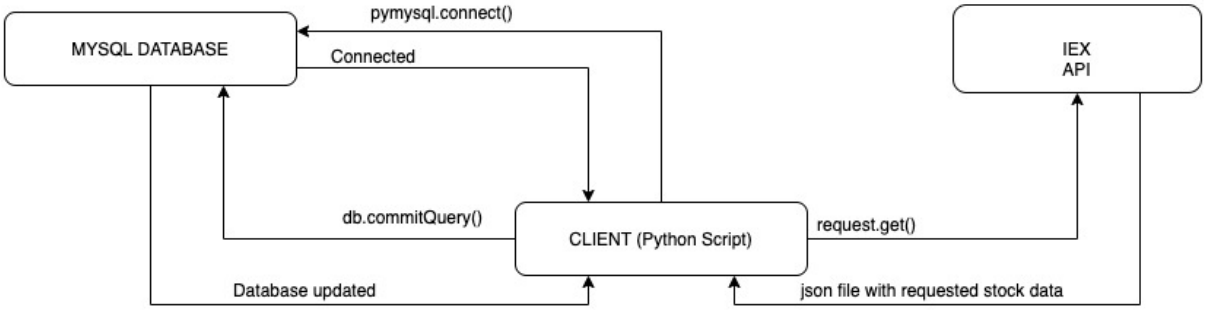

Diagram 3. – Representation of process of initializing DB data

5.2 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.

5.3 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

1. Program is created in a python script.
2. This script will be programed to run every 24 hours
3. Using pymysql to establish a connection to the database.
4. Using requests to establish a ‘GET’ request from IEX to provide stock information for a specific stock on the day request is sent.
5. SQL statement is initialized in a variable to be inserted into the database.
6. Date is initialized to specify the day for IEX.
7. 3 requests are sent to IEX for each stock.
8. Data from IEX is parsed to prepare commit to SQL database.
9. SQL statement is committed and data is inserted into the Database.
10. 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.

Related diagrams

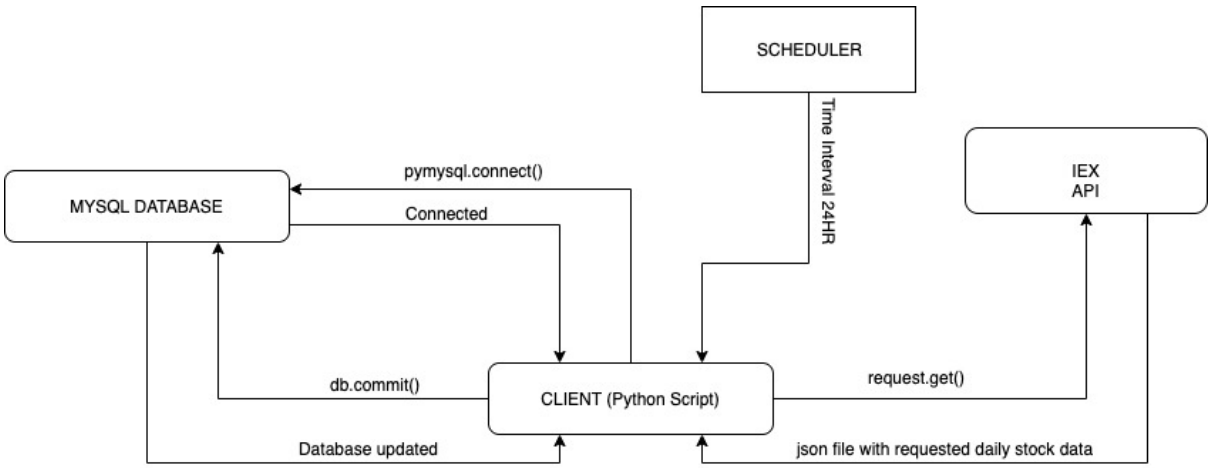

Diagram 4. – Representation of the daily scrapping (update) data process.

5.4 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.

This API endpoint will support report a 30 trading days price list (descending order from the running day) of the records with the fields “date”, “open”, close”, “high”, “low”, “volume”, “change”, “changePercent”, “changeOverTime”, “volumeChange”, “volumeChangePercent” and “volumeChangeOverTime”.

Dynamic fields such as change, changePercent etc should be calculated.

“/daterange”

Returns stocks in data range specified by request.

Returns response 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.

Information that is returned for the given period: max price, min price, average price, mean price.

“/companyInfo”

Returns information about company

Returns response 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.

Company Info that will be provided from the local DB includes Company Name, Website, Exchange, Industry.

Related Diagram

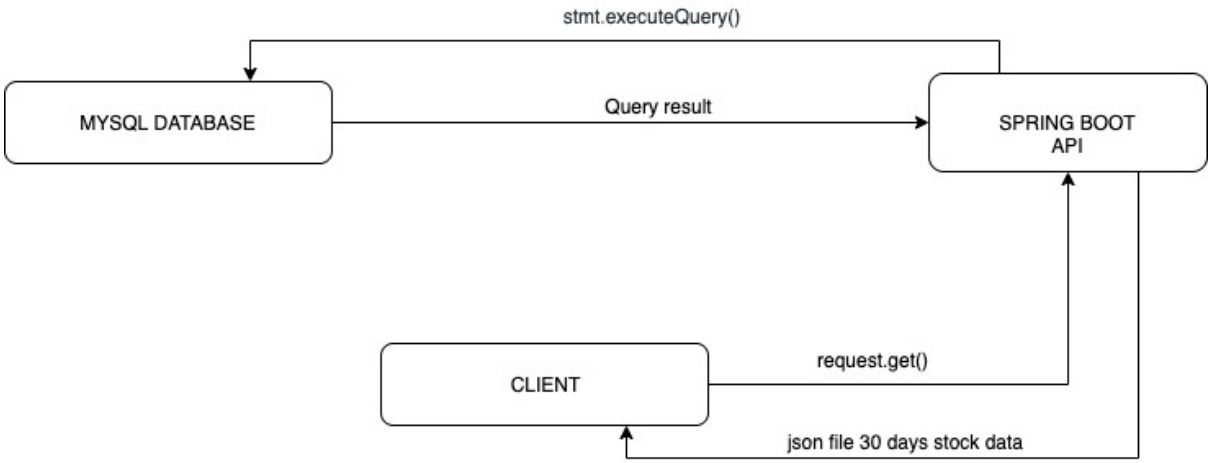

Diagram 5. – Representation of process of consuming Spring Boot API endpoints by Client

5.5 ANGULAR UI DISPLAY

Brief Description:

- Develop complete UI solution that will display responses from Spring API Endpoint requests.

Goal:

- Display JSON response from Spring endpoint.

Precondition:

- Have complete Database developed and prepopulated with crucial data.
- Have Spring API endpoints developed and tested.
- Have possibility to communicate between API level and UI
- Have Angular framework installed and setup

Trigger:

- Request to API endpoint made

Detailed Description

Angular Javascript Framework will be used for the development of the API.

UI should have fields for all 3 companies.

Information about the companies should be retrieved by contacting companyInfo endpoint in Spring API.

Besides that, clicking on each of the company would trigger the other 2 developed endpoints.

One would show the last 30 days info. - /IEX30 endpoint

The other would be working with the date filters to show all the necessary data defined by it (mean, max, avg etc) - /daterange endpoint

Information received from API in JSON format would be parsed and displayed as a text within UI.

Any error messages received as a response from API should be handled properly.

Continuous connection to DB is important for the functionality to be successful.

Mock up design

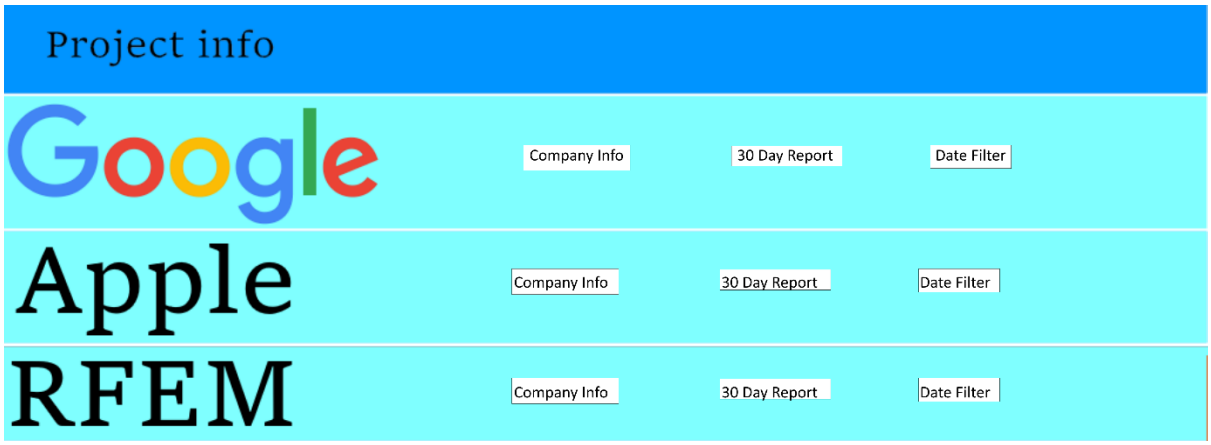

Image 1 – Front page design

Front page design displayed on Image 1 is a mock up version. Final version may defer from the original idea.

We need to have presence of company info, 30 day report and Date Filter buttons that would initiate calls to Spring Boot API.

Image 2 – Design for display of data

On image 2, mock up design of area where information received as a response from appropriate endpoints will be displayed.

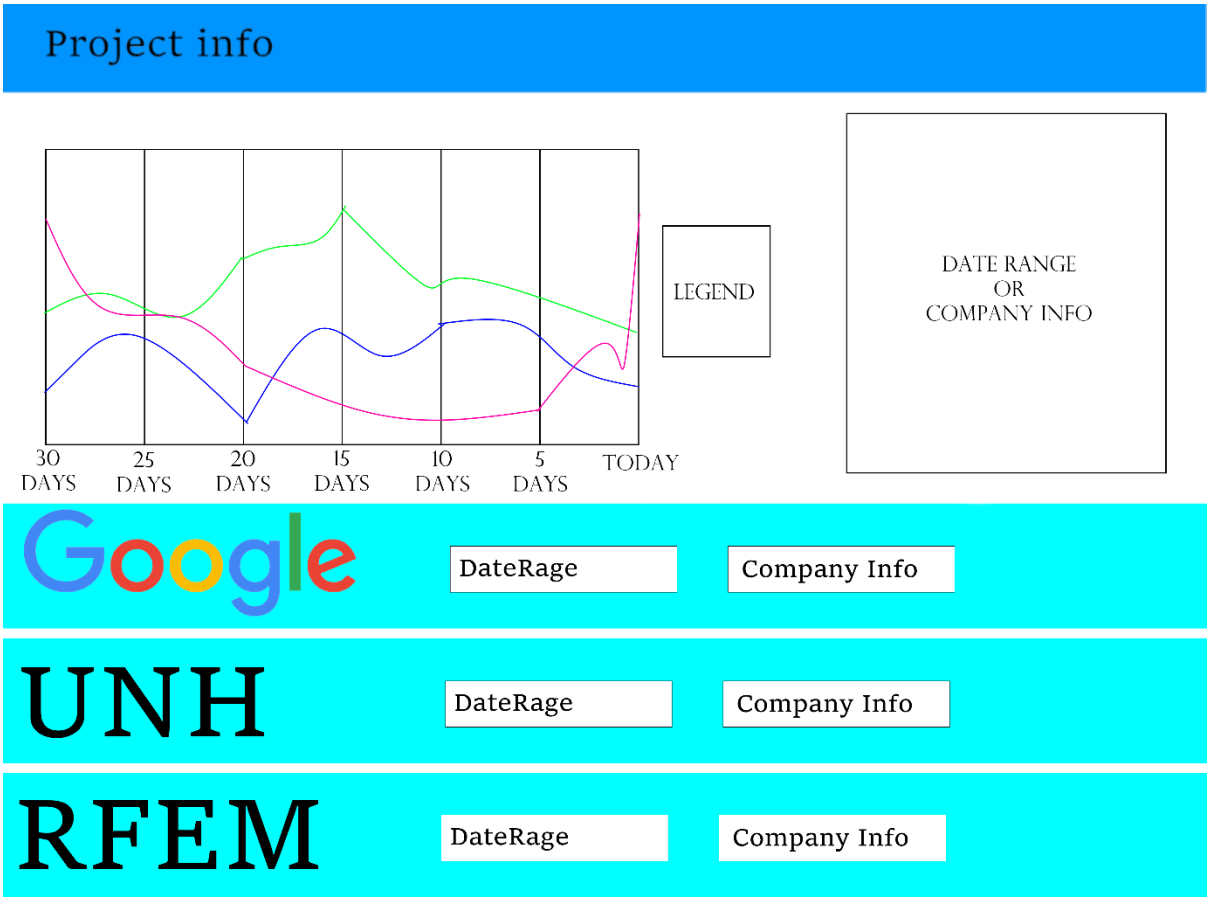

Image 3 – Design with graphs

Mock up on image 3 represents the design idea for when we include Google graphs as a way of representing the data into the application as well.

5.6 REPORTING CHARTS

Brief Description:

- Develop UI solution using Google charts to represent data retrieved from endpoints.

Goal:

- Display JSON response from Spring endpoint on a google chart

Precondition:

- Have complete Database developed and prepopulated with crucial data.
- Have Spring API endpoints developed and tested.
- Have possibility to communicate between API level and UI

- Have Angular framework installed and setup
- Have Google API for chart display connected to the app

Trigger:

- Request to API endpoint made

Detailed Description

Loaded data will be displayed on Google Charts using Google API.

We will be using Google Line and Google Datatable.

Google Datatable will be displaying loaded data received from Spring endpoint.

Google Line Chart will display comparison of close values for 3 companies that we have chosen to report on in the project.

Mockup design

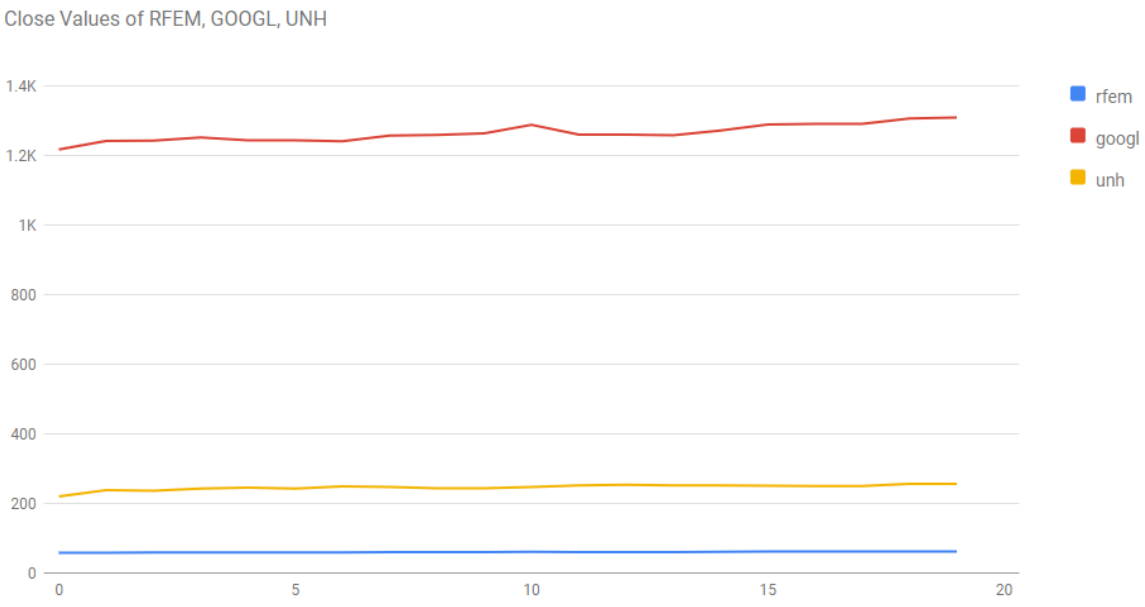

Image 4 – Design of Google Line Chart displaying data

6 SOFTWARE DEVELOPMENT CYCLES AND MANAGEMENT

6.1 SPRINTS

Following the Agile methodology of software development, we have implemented development of the entire software through sprint cycles. We needed 2 sprint cycles for this:

6.1.1 Sprint 1:

Task	Estimation Days	Person Responsible
Setting up development environment (Choosing IDE, integrating with git, helping out other team members so everyone have access to it and can have it ready for development).	2	Philip
Setting up AWS instances	3	Fahad
Creating necessary IEX accounts for development and providing team members with the credentials for it	1	Marko (initial setup, but everyone needs to have account created)
Analysis of the IEX API – list of endpoints relevant for the project, documenting the most important stuff	2	Sharig
DB Design – creating necessary documentation, diagrams and development of DDL statements for creating a DB.	5	Fahad, Hani
Point 6 from the project requirements: development of functions that will use IEX API to retrieve necessary data and store it in our DB for further use.	5	Philip, Fahad
Point 7: task (A)	5	Philip
Point 7: task (B)	5	Sharig,Hani
Point 7: task (C)	3	Fahad
Design UI for display of data obtained through RESTful APIs (general design)	5	Hani, Sharig

For the purpose of Sprint 1, we have also used Trello – Project Management tool that enables tracking developers and other employees input and work/progress made on project.

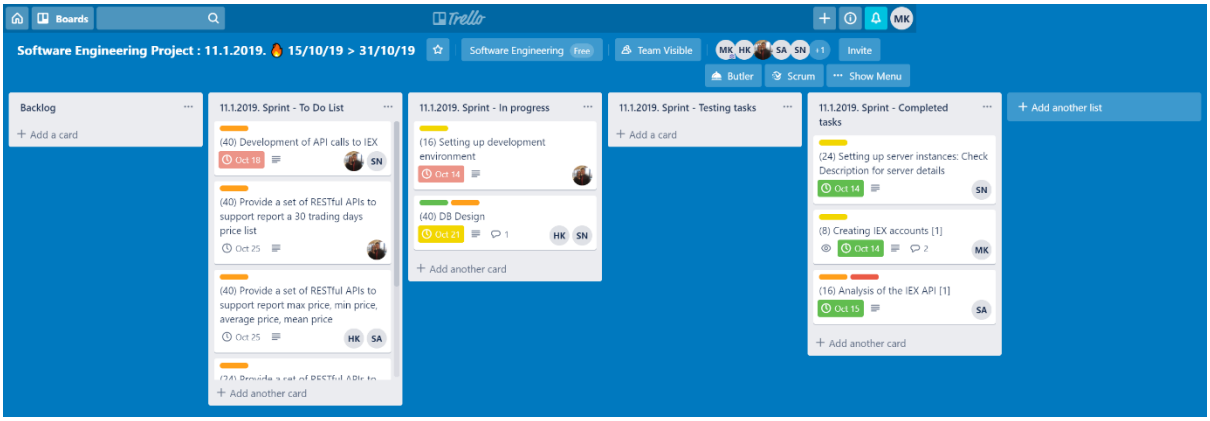

Image 5 – Trello board for Sprint 1

Image 5 represents state of Trello board for Sprint 1. Each user story/development functionality is put in a different column. User stories/functionalities are displayed as "cards" within "columns". We have following columns:

Backlog – Column that displays all the inactive user stories – once that are not being developed at the moment but are part of the scope of the project.

To do List – Column that displays all the active stories within a current sprint. These are the stories that should be developed within a certain sprint but the actual work on them haven't started yet

In progress – Column that displays those stories that are being developed/worked on in the moment

Testing tasks – Column that displays those stories which development has been completed and are now being tested by QA/Testers.

Completed tasks – Column that contains stories/functionalities that have passed the testing phase and are considered successfully completed.

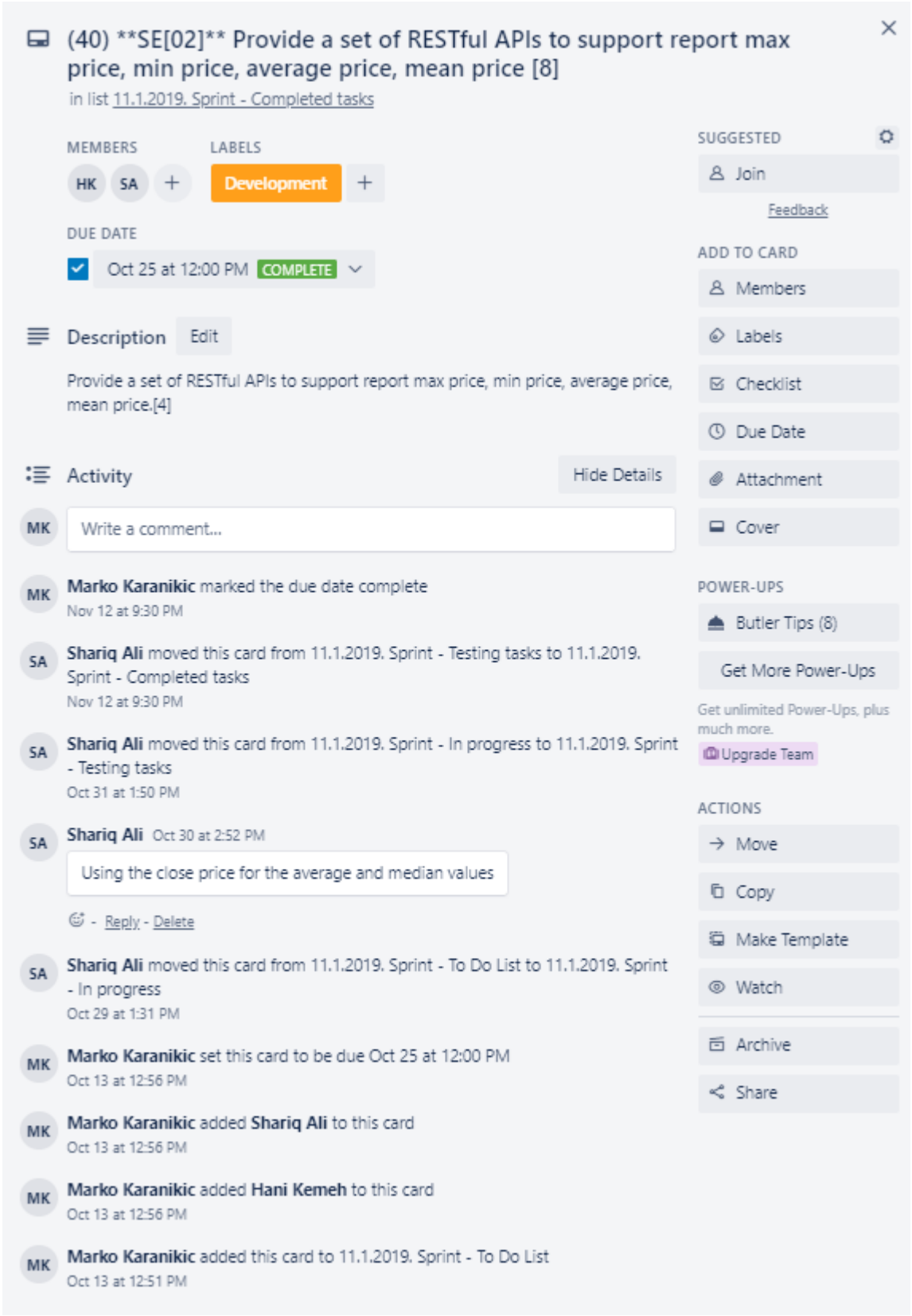

Image 6 – Card Information

Each card, represented by image 6, contains a whole chronological information of a certain functionality, including the activity of people that have been assigned the actual story for development or work on. Some functionalities such as time tracking can be implemented by purchasing add-ons which hasn't been included in this project.

In the end, some of the reporting is made possible by Trello itself, such as burndown chart. Example of burndown chart for Spring 1 of our project is shown on image 7.

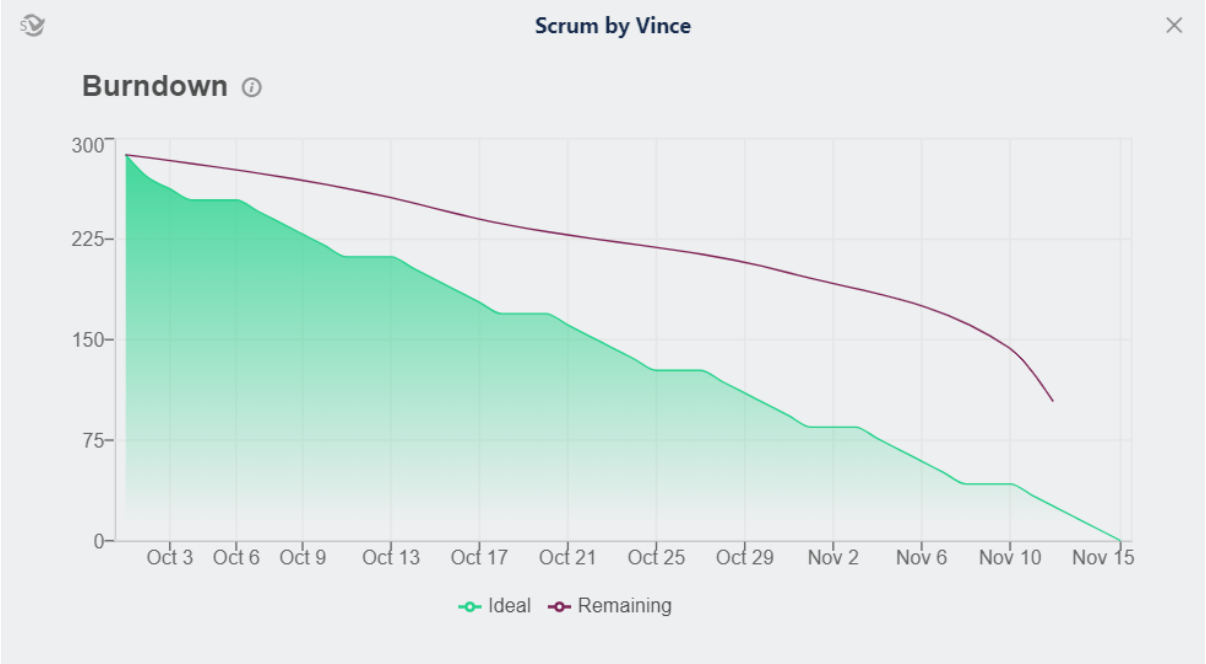

Image 7 – Burndown chart for Sprint 1

6.1.2 Sprint 2:

Task	Estimation Days	Person Responsible
Writing Test Cases	3	Hani, Fahad
Development of UI with Angular	5	Sharig
Integrating UI with API	3	Sharig, Phil
Development of Charts	5	Phil, Fahad
Integrating Charts with UI	1	Sharig
Development of necessary diagrams, completing documentation	2	Hani, Marko

Implementation of Sprint 2 hasn't been followed through Trello, just internally and through regular weekly meetings.

Sprint 2 represents continuance of Sprint 1 in terms of logic and functionalities. Test cases for UI integration have been included and the actual UI development which design has been created in Sprint 1 has taken place in this Sprint. Also, Google charts as an additional functionality have been added to the project.

Final state of Sprint 2 represents verification that the scope of the project have been fulfilled, completing any possible missing documentation and preparing the presentation of the project.

6.2 WEEKLY MEETINGS

In order to follow the progress on every activity and to provide any potential input or support/help for the team members, Skype group has been created and every week we have had weekly meeting where our overall progress have been followed. Usually, these meeting have taken place every Tuesday evening at 8 pm, although, depending on team member's availability, some of these meeting have been held on different dates.

After every meeting, a short recap of what was discussed, how much progress have been made and what would the next steps be, has been pushed to **GitHub Discussion Board.** Addition to this document – PDF document called weekly meeting recap will contain all these recaps on one place.

As an information in this document, an **overall of 8** official meeting have taken place over the period of almost 2 months. Every meeting have been attended by every team member. Last week of the project implementation, several adhoc Skype conversations have taken place, but have not been recorded within GitHub.