**CS673 Software Engineering**

**Team 3 - StockPickers**

**Project Proposal and Planning**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Sylvia Xiang | Team leader | *Sylvie* | 05/13/22 |
| Duy Nguyen | QA Leader | *Duy Nguyen* | 5/14/22 |
| Serguei Chervachidze | Design & Implementation Leader | *Serguei Chervachidze* | 5/14/22 |
| Jing Tian | Security Leader | *Jing Tian* | 5/14/22 |
| Igor Kheyfets | Requirements Leader | *Igor Kheyfets* | 5/14/22 |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
| v1 |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

[Overview](#_heading=h.gjdgxs)

[Related Work](#_heading=h.30j0zll)

[Proposed High level Requirements](#_heading=h.1fob9te)

[Management Plan](#_heading=h.3znysh7)

[Objectives and Priorities](#_heading=h.2et92p0)

[Risk Management (need to be updated constantly)](#_heading=h.tyjcwt)

[Timeline (need to be updated at the end of each iteration)](#_heading=h.3dy6vkm)

[Configuration Management Plan](#_heading=h.1t3h5sf)

[Tools](#_heading=h.4d34og8)

[Deployment Plan if applicable](#_heading=h.2s8eyo1)

[Quality Assurance Plan](#_heading=h.17dp8vu)

[Metrics](#_heading=h.3rdcrjn)

[Code Review Process](#_heading=h.26in1rg)

[Testing](#_heading=h.lnxbz9)

[Defect Management](#_heading=h.35nkun2)

[References](#_heading=h.1ksv4uv)

[Glossary](#_heading=h.44sinio)

# Overview - Serguei

The project will be a web-based application called StockGetter. The idea behind the app is to allow an end user to obtain history of stock price and volume data (i.e. time series data for stock prices and volumes) for publicly traded stocks in the US as well as provide a forecast of the said prices and volumes using a modern machine learning algorithm.

Specifically, the app will display the stock historical data and forecast in an interactive plot and table and it will also allow the user to download said data to their computer. Stock data will be available in 3 granularities: daily, weekly and monthly. Forecasts will also be available in these frequencies, and the user will be able to specify a forecast horizon.

The main users of the software are people interested in obtaining historical stock data as well as forecasts to help them to make informed investment decisions.

The technology stack will include a HTML/JS based front end, using frameworks such as Bootstrap, AgGrid, and Plotly, interfacing via web-based API’s with the Django back-end. The forecasting function will use the Facebook Prophet algorithm. Stock data will be based on publicly available stock feed from Alpha Vantage.

# Related Work - Serguei

There are many apps and software applications for obtaining stock data, oriented towards both businesses and consumers. A small fraction of these offer analytics and forecasting functions. Our project will create an MVP of a simple, easy-to-use app to quickly get historical stock data and generate forecasts.

# Proposed High level Requirements - Serguei

* 1. Functional Requirements  
     1. Essential Features
        1. Offering historic stock price data: as a user, I want to be able to obtain historical stock price and volume data (in a table format) for any traded US stock at my chosen granularity (daily, weekly, monthly) so that I can use that information for my investment decisions
        2. Offering price and volume forecasts: as user, I want to be able to get a forecast for stock price or volume at my chosen frequency and my chosen horizon, so that I can make better investment decisions
        3. Displaying stock data in plots: as a user, I would like to be able to display price and volume data in an interactive plot so that I can better analyze the information
        4. Download capabilities: as a user, I would like to be able to download the data for one or multiple stocks to my computer
        5. Displaying data for multiple stocks: as a user, I would like to be able to display data for a number of stocks at the same time, so that I can have this all in one place
        6. Downloading data for multiple stocks: as a user, I would like to be able to download data for multiple stocks (that are already being displayed) at the same time, so that I can have everything in one place
     2. Desirable Features:
        1. None at this point
     3. Optional Features (additional cool features that you want to have if there is time):
        1. Getting data for an an unlimited multiple set of stocks at the same time: As a user, I would like to be able to select an unlimited number of multiple stocks and get data for these at the same time quickly
        2. Print functionality: as a user, I would like to be able to print information for a given stock in a compact informative form
        3. Use login & database: save users favorite stock
     4. Existing Features: the project will reuse some existing frontend functionality that has been done by Serguei for a Web dev class. Existing features concern the front end only and do not have forecasting capabilities or the related backend (these will need to be implemented as part of this project):
        1. Historic price data: the existing front end is able to get some historic data from a publicly available API, but this can be reused or extended as needed.
        2. Display/download: some existing display and download capability. This can be reused or extended as needed.
  2. Nonfunctional Requirements
     1. Security requirements:
        1. None at this point

# Management Plan - Sylvie

## Objectives and Priorities

Project objectives are adding features in the US stocks picking application for users to get stock prices forecast, as well as saving and plotting the prices. Optionally to include selected stocks as favorites, a log-in page, and deploy the application if time permits.

Moreover, during the completion of all objectives, the team will identify and fix any bugs to maintain high quality of code.

## Risk Management (need to be updated constantly)

Our team estimated 8 categories of risks which are the following:

“personnel” – risks of losing or adding team members, as well as if any members are lacking responsibility. Our team will be checking weekly on each other’s progress to avoid this risk.

“communication” – risks of duplicate, useless, or wrong components of work. Our team will be as detailed as possible when assigning work and when discussing tasks.

“requirements” – risks of unclear requirements and scope creep. Our team will be communicating and explaining tasks in detail, as well as eliminating unnecessary steps.

“management” – risks of improper task assignments, planning or management. Each team member will seek help or announce concerns immediately through communication tool.

“technology competence” – risks of unfamiliarity with framework, programming language, unit testing, Git, or other technologies. Our team has discussed and agreed on tools to be used, and also will learn more technologies as discussed.

“design and implementation” – risks of improper design and messy code. Our team will constantly check on code and implementation. All codes will go through a code review process ensuring code meets team’s standards.

“testing” – risk of not enough testing. Our team will learn unit testing and ensure high coverage.

“integration and deployment” – risks of not enough time for integration or deployment. Our team will be meeting and working on the project every week to ensure smooth progress. Deployment would be optional for the project.

**Risk Management Sheet Link:**

https://docs.google.com/spreadsheets/d/1jmcOHYI6mC7HBWWxX6f9xOuKKddMRtsi/edit#gid=1028329057

## Timeline (need to be updated at the end of each iteration)

Following table is an estimation and will be updated in each later iteration

| Iteration | Functional Requirements(Essential/Disable/Option) | Tasks (Cross requirements tasks) | Estimated/real person hours | Presentation Recording Link (5-10 minutes for iteration 1 and 2) |
| --- | --- | --- | --- | --- |
| 1 | May 21 | May 24 | 100 | May 30 |
| 2 | May 31 | Jun. 5 | 100 | Jun.6 |
| 3 | Jun.7 | Jun.14 | 100 | Jun.20 |

# Configuration Management Plan - Igor

## Tools

In this project we will be using Git as our main version control system. Team members will use IDEs/text editors they are most comfortable with. Project management will be handled in PivotalTracker.

## Code Commit Guideline and Git Branching Strategy

We will be using feature branching for this project, i.e. when working on a new feature a new branch is created. Team members working on this feature will commit their code to this branch. When the feature development is completed, a pull request (PR) to the main branch is created.  
We will implement a one person pull request approval, however, we also decided to have an approval window, so the other team members can go over the PR and give their input (time-permitting).

## Deployment Plan if applicable

We are not actively planning on deploying our application as we do not think that is feasible in this short timeframe, however, time-permitting, we would use Gunicorn WSGI. Whitenoise would handle serving staticfiles. Since we are not planning on using a substantial amount of storage space for our databases, we would deploy to Heroku.

# Quality Assurance Plan -Duy

## Metrics

| **Metric Name** | **Description** |
| --- | --- |
| **Cost** | # of person hours used |
| **Test case pass** | % of unit test passed/total unit test |
| **# of Feature completed** | Number of planned features that are completed |
| **Customer Rating** | MFA of completed program to be sent out to customer for rating. |

* 1. Coding Standard

PEP8 Style Guidelines will be used for this project.

## Code Review Process

Team members will review each other’s code. Whenever a new change is needed, a new branch will be created. Once changes are made within the branch, a pull request will be initiated. At least 1 other member on the team has to approve before the merge is authorized. All code review activities will be following the established Code Review Checklist. Any unapproved pull requests should have feedback including reason for rejection and suggestion for improvement.

[Team 3 Stockpickers Code Review Checklist.xlsx](https://docs.google.com/spreadsheets/d/1YKXsXZPdIoScveJWW2MEeZKNJHPTcfl0/edit?usp=sharing&ouid=105226445373325506315&rtpof=true&sd=true)

## Testing

Unit tests will be written for key functions of the program. Every team member will help with unit tests but the QA leader will be responsible for developing an adequate amount of unit tests to validate through integration tests. QA leader is also responsible for running unit tests and integration testing.

## Defect Management

We will be using GitHub Issues for the defect/bug tracking system.

# References

Rossum, G., Warsaw, B. and Coghlan, N., 2001. *PEP 8 – Style Guide for Python Code | peps.python.org*. [online] Peps.python.org. Available at: <https://peps.python.org/pep-0008/> [Accessed 14 May 2022].

REDDY GUTHA, S. (2015). *Code Review Checklist – To Perform Effective Code Reviews*. Evoke Technologies. Retrieved 14 May 2022, from https://www.evoketechnologies.com/blog/code-review-checklist-perform-effective-code-reviews/.

# Glossary - Everyone as needed

MFA - Market Feedback Analysis, a survey that tracks performance of a process based on end user(market) opinion.

PR - Pull request

IDE - Integrated Development Environment