Software Requirements Specification

for

Financial Sentiment Based on News App

Version 1.0.2 approved

Prepared by Andriy Lunin, Nicholas Raffone, Mohammad Aziz

NYU Dev Team

October 2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Latest SRS	Dec 8	Updated the format and requirements	1.0.2

1. Introduction

1.1 Purpose

Product name: Financial Sentiment based on latest News. Release version: 1.0.2. Application consists of frontend (React) and backend (Flask, API). This application is used to convert news into a non binary scale of market sentiment for better analysis of the market trends.

1.2 Document Conventions

Assume that italicized font dominates the regular font, while the bold fond has higher priority over italics. The font importance is associative. Additionally, higher level requieremts apply to **all** dependancies. Entertain **high** priority for all of the documentation, tests, and components of the Application.

1.3 Intended Audience and Reading Suggestions

The Applicaton is intended for **intermediate to advanced** non-quantitative market traders. It is deployed under the MIT license and serves on as is basis. The development team does not bare any responsibility for the potential losses, and does not claim shares for market gains.

1.4 Product Scope

We provide a trading-analytical tool for all intended users. The market scope is tailored, but not limited to, the New York Stock Exhange. The relevant currency is USD. Application server requires to be hosted in non-sanctioned US allies for retrieval of sensitive market data.

1.5 References

News retrieved with the Google News RSS for the US region and English Languge:
https://news.google.com/topstories?hl=en-US&gl=US&ceid=US:en

Flask, YFinance, and Python Native Multithreading used for backend:
https://flask.palletsprojects.com/en/2.0.x/
https://pypi.org/project/yfinance/

React and Tailwind are the main resources for the frontend:
https://reactjs.org/
https://tailwindcss.com/

Sentiment Analysis is done with BERT and NLTK classfiers:
https://arxiv.org/abs/1810.04805
https://www.nltk.org/

2. Overall Description

2.1 Product Perspective

This is a new, state-of-art project that does not build on any previous ideas, nor modifies older implementation of the same idea. The system is self contained and complete. It acts **independently** of other systems and serves as an APP by itself.

2.2 Product Functions

The main function of the Applcication is *gathering* and *assessing* the relevant news articles based on a query that is the company name. Users that would like to quickly assess the market sentiment towards some company may use the application to gauge news articles with their sentiments illustrated by the side. Users have an option of doing their research further by taking the Application response and following on the article references.

2.3 User Classes and Characteristics

The Application is designed for regular PC users, with basic expericence, and intermediate market knowledge.
Once the application is deployed, anyone with access to the www web may access the Application and enjoy the perks. We strongly reccomend to use the Application for the baseline information purposes, and follow the API response with personal research.

2.4 Operating Environment

The Application is built to have its own virtual environment, which allows anyone to run it. It was rigorously tested on *Macintosh* computers. The app needs *Python3* and *NPM* to function properly.

2.5 Design and Implementation Constraints

There are no possible contrains for the users. However, due to the multi-thredding aspect of the Application, the more cores a computer has, the faster the web scraping will be. The expected timing for each query is 8~9 seconds, which is an improvement on our first build where the timing was 10+ minutes per response.
The application was designed to support every machine.

2.6 User Documentation

The Application is served under **MIT license**, in the as-is state. The development team will **not respond** to special requests.
There is a compherensive documentation set that comes with the Application in the main Application release folder: *https://github.com/moon1ock/FinancialSentiment*

2.7 Assumptions and Dependencies

The application uses 70 external libraries. These dependencies are frozen and our development team recommends to use the same library versions for Application deployment as used for the development. Main dependencies include, but not limited to: Flask, React, Tailwind, BeautifulSoup, Numpy, Python, Multitasking, etc.
The comprehensive list of all dependencies can be found in the official doccumentation here: https://github.com/moon1ock/FinancialSentiment/blob/main/project/finsenti_backend/requirements.txt

3. External Interface Requirements

3.1 User Interfaces

User interface is aimed to be clear and intuitive. Upon loading, the user is taken to the landing page where they are able to type in any query be in company name, company name with a typo, or a ticker, and retrieve the comapny articles.
A response shows the official listed company name, recent stock data, and the retrieved news articles. Arrows near the articles indicate whether the sentiement is positive, negative, or neutral.

3.2 Hardware Interfaces

The Application is meant to run on Mac. Minial processing power and minimal RAM, as well as 1 core is required.
Improvements with RAM and more cores will result in faster and sharper performance.

3.3 Software Interfaces

As described above, the Application is self contained and tailored for a virtual environment.
A comprehensive list of libraries can be found in the product repository here:
https://github.com/moon1ock/FinancialSentiment/blob/main/project/finsenti_backend/requirements.txt

3.4 Communications Interfaces

The Application requires the latest SSL ceritificate for proper functioning.
The server is natively hosted on port 5000, and web scarping requires HTTPS, while the front end utilizes POST/GET requests for communication with backend.

4. System Features

The main system feature presented is the API for news retrieval. The API is hosted with the backed component of the Application. Upon request, it returns a JSON file with a set of latest news articles, their links, publishers, publication date, title, description, and the sentiment score, where -1 is most negative, and +1 is most positive.

4.1 System API

4.1.1 Description and Priority

At the backed, the server makes a request to Google News RSS and gets the 100 most recent articles on the topic. It scrapes and processes each article simultaneously, and returns a JSON as an output.
Priority: **high**: 8.

4.1.2 Stimulus/Response Sequences

User opens the landing page, types in a query and hits **return**.

4.1.3 Functional Requirements

REQ_1: map the query to the official company name
REQ_2: map the company name to the stock ticker and retrieve the stock data
REQ_3: retrieve recent articles on the topic
REQ_4: asynchronously scrape each article and calculate its sentiment, as well as the head image
REQ_5: reduce the threads to the parent thread without losing data and memory leaks
REQ_6: format the data into a JSON and send it to the requester.

4.2 System Frontend

4.2.1 Description and Priority

User Interface is designed to be facilitating the overall experience. It is not necessary for the proper fucntioning of the server. Priority: low.

4.2.2 Stimulus/Response Sequences:

User opens the page.

4.1.3 Functional Requirements

REQ_1: server responses are pretified and more readable.
REQ_2: the landing screen has intuitive design and animations.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

For proper performance please aim to have these hardware specifications:
processor: Inter Pentium or above
RAM: 2GB or above
cpu: 1 or above (for multithreading 2-8 processors best)
GPU: not required.

5.2 Safety Requirements

No safety concerns are expected from the Application.
Personal data is well protected and never shared with the server. The server does not track IP addresses, locations, or browser IDs of the clients.

5.3 Security Requirements

Users do not need to authenticate for using the Application.
For the server side, the latest Transport Layer Security (SSL) protocol is required to be installed.

5.4 Software Quality Attributes

The product has no learning curve and is very intuitive to use.
Availability is not critival and is estimated to be 3 nines (99.9%)
Testability is native and easy.
Portability is native. Available on every device.

5.5 Business Rules

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

THE SOFTWARE IS OPEN SOURCE AND FREE OF CHARGE. ANY ONE MAY COPY THE APPLICATION AND MODIFY IT TO SUIT THEIR NEEDS.

THE DEVELOPMENT TEAM (e.g. Andriy Lunin, Nicholas Raffone, Mohammad Aziz) DO NOT PROMISE ANY RESP

6. Requirement Gathering

The system requirements were gathered and elicited by interviewing each other while we assumed the roles of potential user and high-level project manage. By putting ourselves in our users' shoes, we were able to think of requirements that would make it easiest for the user to navigate our system. For instance, we believe that users want autonomy over how they view the information as well as clarity, so having a filtering tool and responsive design would help with this. When we looked from a high-level perspective, we saw that we wanted our frontend and backend to work independently as they accomplish two different goals: the frontend is to display data and the backend is to process data. Seeing this, we put a lot of emphasis on creating specific requirements that helped divide these processes for their respective goals. This can be seen above, as we have no overlap between the two different subsystems' requirements.