# Applications of Web Development in Portfolio Research

A Capstone Report
presented to the faculty of the
School of Engineering and Applied Science
University of Virginia

by

Anish Mandalika

May 12, 2023

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

*Anish Mandalika*

*Capstone advisor*: Briana Morrison, Department of Computer Science

# Applications of Web Development in Portfolio Research

CS4991 Capstone Report, 2023

Anish Mandalika
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
am8wk@virginia.edu

**ABSTRACT**
A New York City-based investment management firm needed a new system to aid investment professionals in performing portfolio research that took advantage of advances in web technologies and data-driven methods. I worked with a team that is developing a suite of tools to address this, including tools to aid in data visualization and tools that can be interacted with programmatically. The various tools involved creating large-scale data processing routines in R, compiling and exposing relevant data endpoints using Java and Spring Boot, as well as providing a GUI front-end for the endpoints. The suite covers a range of areas—from tools to better visualize the existing portfolio, to tools to keep track of stocks worth watching—all designed to neatly integrate with the firm's existing systems. The suite of tools has already gained users from within the firm, and new features are being added regularly based on user feedback. While the suite of tools is quite robust, there are several additional tools and features now in development that have the potential to further aid the process of portfolio research.

## 1. INTRODUCTION

The rise of Big Data has led to a revolution in the investment sector. Through the power of computer-aided analysis techniques, investment professionals have been able to greatly augment the information based on which to make their decisions. Because of this interest in data-aware investing, companies in the investment space have been developing various data analysis systems for their traders. I worked for a New York City-based investment management firm that wanted to provide these analytics to its traders. The firm had an existing suite of tools that performed various analyses on trading data, but needed to optimize existing processes and add new functionality.

## 2. RELATED WORKS

Kamdar (2022) details the implementation of a Serverless Stock Market Web Application that runs in the cloud. The application ingests raw data, processes it, performs analyses and presents the results in a user-friendly format. While the specific technologies used in this application (Python and ReactJS) differ from those used in my application (Java, R and Spring), both have similar aims and use similar techniques, including the use of Machine Learning (ML) algorithms to perform basic predictions. This application also leverages cloud computing techniques while my application was designed to run on a physical server.

Lin (2018) talks about the design, deployment and performance of a Stock Forecasting Application that was built in two weeks. Lin provides deeper insights into the ML and Artificial Intelligence (AI) algorithms that analyze the data, and he explains the importance of a responsive User Design. This application relies heavily on ML and AI to perform analysis and make predictions. In contrast, the application I worked on is aimed at Investment Professionals and therefore only provides basic forecasts.

## 3. SYSTEM DESIGN

The application I worked on has been in use for a few years, and so is quite complex. Most of the application is written in Java using Spring Boot, with some data processing in R and front-end UI written in various JavaScript-based frameworks.

### 3.1 Overview of System Architecture

The Portfolio Research Application provides a number of Portfolio Analysis and Research tools. The data for each tool is processed as a daily batch on a company-owned server. The rough ordering of the data processing pipeline for each of these tools is:

- Raw data is pulled from various sources, cleaned and stored in a database. These data include various stock performance metrics, performance data for indices such as the Standard & Poor (S&P) 500, currency exchange rates and other relevant market data such as mergers and acquisitions.
- The cleaned data is used to perform analysis, including time-series analysis of various metrics as well as basic predictions. The exact nature of these analyses is dependent on the needs of the specific end-user and can be tailored to include the performance of their current portfolios as well as any what-if scenarios of their choosing.
- The results of the analyses are made available through API endpoints. Some end users use these results as inputs into their own prediction algorithms, while others have written their own User Interfaces that present results in the way they choose. These API endpoints allow those users to consume results directly.
- Many users choose to use the User Interface included with the tool. The UI consumes results from the API endpoints and displays graphs and tables of the user's choosing.

### 3.2 Requirements

I was tasked with creating two separate functionalities during my time at the company:

- A cache file to store daily index performance data using R; and
- APIs using Spring Boot and Java for Stock Ticker Overrides and Internal Quality Assurance Data.

### 3.3 Implementation

All my tasks were slight variations of existing functionality. Because of this, I studied the existing systems thoroughly and used parts of the existing codebase wherever possible.

#### *3.3.1 Index Performance Cache File*

Various data sources report daily performance metrics for indices such as the S&P 500. Stock traders find it useful to perform various time scale analyses on these metrics to inform their decisions. These analyses involve a large number of simultaneous calls to the database storing performance data, causing execution to slow down. Since the analyses look at week-on-week or month-on-month performance, creating a cache file containing a month's worth of data might prevent the database from being inundated with repeated requests for that month's data.

I implemented a routine that creates such a cache file. Given the system architecture, it made the most sense to generate the cache file right after the data was ingested and cleaned. Because those functionalities were implementing using R, and R includes good tools for processing large amounts of data, the cache file generation routine was implemented in R. The routine reads performance data from the database and saves it as a cache file in the server's local memory.

#### *3.3.2 Stock Ticker Override API*

Some traders may want to combine the performances of multiple companies for various reasons including shared ownership and correlated performance. These overrides were being stored in the database and needed to be made accessible to the UI frontend and/or end users' programs. I used a REST API to achieve this. I chose Spring Boot, which uses Java, to implement this API, since it was already in use across the system for implementing APIs. The

Spring Boot routine makes a call to the database querying the specified user's Overrides and returns them to the user in JSON format.

#### *3.3.3 Quality Assurance Data API*

Several Quality Assurance (QA) metrics are generated by the system during the daily data processing batch. While these metrics are generated for internal use by the development team, some end users who wrote their own programs consuming this application's data felt that QA data would help them determine how much to trust the results of their algorithms. For this reason, I implemented a REST API that aggregates various QA metrics and makes them available. Once again, I chose Spring Boot to implement this API since it was already in widespread use within the application.

## 4. RESULTS

I was able to implement all functionalities I was tasked with during my time at the company.

### 4.1 Index Performance Cache File

The cache file generation routine was put into production and led to around 180 fewer database calls per day. Since each of those database calls would have involved large amounts of data spanning multiple weeks or months, this represents a significant reduction in database traffic. While the time savings of using the cache file are difficult to quantify due to the complexities of the system, the reduction in database traffic has allowed developers to make better use of database resources.

### 4.2 Stock Ticker Override and Quality Assurance Data APIs

Both APIs were implemented and put into production. They reported over 99.9% uptime throughout my time at the company. These APIs have allowed the GUI front-end as well as end users' programs to show Ticker Overrides and QA data. Both streams of data enhance the usefulness of other metrics that the application provides by providing additional functionality and context.

## 5. CONCLUSION

Investment Management firms are constantly looking for ways to enhance data available to their traders. Web-based analysis applications can help traders derive helpful insights into their performance, thereby helping to improve it. Improvements in system performance ensure the reliability and stability of the system by reducing strain on system components during peak times, and additional data streams help traders to make better use of the data at hand.

## 6. FUTURE WORK

The entire application is being updated on an ongoing basis, with user feedback driving most of the development. The Index Performance Cache File was implemented as a proof-of-concept, and its apparent success could lead to the creation of similar cache files for data involved in similar time scale analyses. Other methods to optimize resource usage during heavy computations such as parallelization can also be explored. New APIs are constantly being implemented within the system based on user need, but implementing additional APIs to provide metadata such as QA logs and system performance may prove useful to some users depending on the application.

## 7. UVA EVALUATION

As a part of this project, I worked within a Software Development team, which used a Scrum-based methodology to measure development progress. CS 3240 (Advanced Software Development) was extremely helpful in preparing me to work in such a team. The course's discussion of collaborative development etiquette and methodologies, including technologies such as Git and techniques such as Scrum, served as good preparation for working with a team. The course's project was also a good way to get a feel for working with a team for an extended period.

## REFERENCES

Priyanka Kamdar. 2022. *Serverless Stock Market Web Application*. CS 687 Capstone Project Progress Report. City University of Seattle, Seattle, WA.

Yun Zhi Lin, 2018. *Building a Stock Forecast Application in 2 Weeks Using Serverless AI and Responsive Design*. Retrieved December 4, 2022 from https://www.contino.io/insights/building-a-stock-forecast-application-in-2-weeks-using-serverless-ai-and-responsive-design