



Automated equity valuation and investment opportunity alerts

Full Stack Minimum Viable Product application development

Antal Tettinger

Master's thesis

April 2024

Information And Communication Technology Master's Degree

Programme in Information Technology Full Stack Software Development

Tettinger, Antal

Automated equity valuation and investment opportunity alerts

Individual Full Stack Minimum Viable Product application development

Jyväskylä: Jamk University of Applied Sciences, September 2024, 68 pages.

Degree Programme in Full Stack Development. Master's thesis.

Permission for open access publication: Yes/No

Language of publication: English

Abstract

This thesis explores the iterative planning and execution of a time-constrained project aimed at developing a full-stack MVP application. The research investigates methodologies and tools for iterative software development, focusing on professional software development skills enhancement. A detailed diary reporting structure outlines the project's progression over five weeks, highlighting key activities, such as project setup, financial valuation method determination, hardware and tech stack setup, data querying, and app deployment. The study concludes with a discussion of learnings from the project, implications of findings, and recommendations for further improvements and research. The author reflects on the decision-making process and proposes new research paths and questions based on the project's development and documentation. Additionally, the suitability of the chosen technological stack from a User Experience perspective is evaluated. This research contributes to the understanding of iterative software development processes and provides insights for future projects in the field.

Keywords/tags (subjects)

Minimum Viable Product, diary method, full stack software engineering, financial analysis

Miscellaneous (Confidential information)

Contents

1 Introduction.....	4
1.1 Background and motivation	4
1.2 Research goal and questions	7
1.3 Research method.....	9
2 Diary Reporting.....	10
2.1 Week 1.....	10
2.1.1 Monday 22 January 2024: Project setup.	10
2.1.2 Friday 27th January 2024: Financial valuation method for the application	13
2.1.2 Weekly reflection.....	18
2.2 Week 2.....	20
2.2.1 Tuesday 30 January 2024: Hardware setup.....	20
2.2.2 Wednesday 31 January 2024: Decisions about the tech stack for the application	24
2.2.3 Thursday 1 February 2024: Data querying	26
2.2.4 Sunday 4 February 2024: Versioning and weekly reflection	28
2.3 Week 3.....	30
2.3.1 Tuesday 6 February 2024: Querying the remaining financial data.....	30
2.3.2 Saturday 10 January 2024: Calculating the first part of the Discounted Cash Flow ...	32
2.3.3 Sunday 11 January 2024: Weekly reflection.....	33
2.4 Week 4.....	34
2.4.1 Tuesday 12 February 2024: Calculating the WACC and DCF.	34
2.4.2 Wednesday 13 February 2024: Bug fixing.	36
2.4.3 Saturday 17 February 2024: Database, deployment, notifications, static website	37
2.4.4 Sunday 18 February 2024: Weekly reflection.....	41
2.5 Week 5.....	43

2.5.1 Tuesday 20 February 2024: Settings page development.....	43
2.5.2 Thursday 22 February 2024: Navigation, scheduler and automatic change commits	45
2.5.3 Friday 23 February 2024: Routing issue resolutions for final deployment.....	46
2.5.4 Sunday 25 February 2024: Weekly reflection.....	47
2.6.1 Monday 26 February 2024: Signal page styling, web crawler for dynamic stocks	49
2.6.2 Tuesday 28 February 2024: Deployment, listing page style and pagination.	52
2.6.3 Wednesday 29 February 2024: User feedback.....	53
2.6.4 Friday 29 February 2024: Rate limiting and updating notification messages.....	54
2.6.5 Sunday 31 February 2024: Weekly reflection.....	56
3 Conclusions and discussion	57
3.1 Summary of diary reporting period	57
3.2 Answers to research questions.....	57
3.2.1 What are the learnings from the iterative planning and execution of a time- constrained project that has a goal of developing a full-stack MVP application?	57
3.2.2 What methodologies and tools allow for efficient iterative software development?	58
3.2.3 What are the learnings from this project that can enhance professional software development skills?	59
3.2.4 What continuous improvements could be implemented in the initially deployed MVP?	61
3.2.5 What are the learnings and conclusions that can be drawn based on the continuous self-reflective decision-making process?.....	61
3.2.6 What kind of new research paths and questions can we derive at the end of the development and reflective documentation of this thesis project?	62
3.2.7 How suitable is the chosen technological stack from a User Experience perspective?	62
3.3 Implications of findings	63

3.4 Recommendations for further improvements and research based on results.	64
3.5 Conclusions and thoughts of the author	65
References.....	67

Figures

Figure 1. Kanban board on GitHub	12
Figure 2. Project Kanban board at the end of week 1	20
Figure 3. Accessing the desktop of the Raspberry Pi from my iPhone SE mobile device	23
Figure 4. JSON data dump display for Cash Flow data aggregate for the META stock in for the past four years.....	28
Figure 5. Kanban board as of the end of Week 2 of thesis project.	29
Figure 6. HTML display of data retrieved from the Alpha Vantage API for debugging purposes shown here as a work-in-progress stage without styling.	31
Figure 7. Kanban board status at the end of week three.	34
Figure 8. The status of the Kanban board at the end of the 4th week.....	43
Figure 9. The final settings page UI - accessible from the server only.	44
Figure 10. The status of the Kanban board at the end of week 5.	48
Figure 11. Example of a generated signal overview.	51
Figure 12. Updated explanation text of the signal page.....	54
Figure 13. Message on Slack alerting the user of potential investment candidate stock tickers based on the DCF analysis.	56

1 Introduction

1.1 Background and motivation

The aim of this applied thesis research project is two-fold. From one aspect to discover the full stack development methods of an application that requires frameworks from the full stack spectrum and a practical implementation of modern software development life cycle that results in a working application, which is a Most Viable Product (MVP). The term MVP was popularized by Eric Ries as a “version of the product that enables a full turn of the Build-Measure-Learn loop with a minimum amount of effort and the least amount of development time” (Ries, 2019, p. 82). There are several tech companies that were founded on MVP principles such as Airbnb, which started as a minimalist website listing space in the founder’s apartment. Later, the founders were accepted into a San Francisco based start-up accelerator program called Y Combinator which advocates MVP style product development by getting into the loop of delivering the product quickly to receive feedback, and to keep improving the product based on a continuous improvement-feedback loop (Seibel, n.d.). In essence the MVP is a way to start a conversation with the users and potential customers.

From the other aspect, the motivation for this project stems from the fact that the nature of modern software engineering is increasingly interdisciplinary. Drawing from the author's background in finance prior to transitioning to a full-time software engineering role, the study aims to explore novel research questions and discover learnings emerging from the development of a full-stack software application within the realm of financial analysis.

To achieve these dual objectives, the subject of this thesis project is focusing on the design, development and deployment of a full stack software application that provides timely, automated recommendations of investment opportunities in the American NYSE and NASDAQ stock markets based on a set criteria using publicly available data queried from API’s and the Discounted Cash Flow (DCF) method, which is an asset valuation method based on the Net Present Value (NPV) of the estimated future cash flows that a given company is expected to generate.

The objective of this endeavor is not solely to develop a new product, but rather to embark on an open-ended project while meticulously documenting each step of the process. Through this approach, we aim to generate valuable insights, novel inquiries, and comprehensive reflections for individuals initiating full-stack engineering projects. This documentation serves as academic material, offering a rich resource that delves into the intricacies of such endeavors, providing invaluable learnings and facilitating a deeper understanding of the development process.

The objective of this endeavor is not in itself the creation of a novel product, but instead creating an open-ended project while meticulously documenting the process. This in turn can lead to novel questions, conclusions and learnings, offering a rich resource for someone who delves into a full stack engineering project and is searching for academic material that documents and reflects on such a development process.

In the context of launching a full-stack software engineering project as an individual developer, the iterative development process bears the potential for uncovering innovative insights in financial analysis and software engineering methodologies. This study focuses on shedding light on how such a project could yield fresh perspectives through iterative refinement and reflective practices inherent in the chosen diary research method.

The proponent of the Zettelkasten method, Ahrens (2022), emphasized that it is preferable not to start a project with exact research questions. Instead, Ahrens suggests that one should reach conclusions through continuous questioning and iterative work. Ideas naturally emerge through the iterative learning process, allowing writing and research to develop organically. "Ideas will come by themselves, and your writing will develop from there" (pg. 16). This method, also known as the slip-box method, comes from Nikolas Luhman, the German sociologist who was one of the most prolific academic writers of the twentieth century. Therefore, it is argued that following this philosophy and keeping an open mind about the iterative development process and learnings can further enhance the chosen diary research method in generating valuable insights.

The author's finance background originates from the bachelor's studies of Business Administration in Ecuador at the Universidad San Francisco de Quito, which included courses on statistics and financial market analysis. Another source of certified knowledge about this field is based on a successful Stock Market Qualification Exam in Hungary in the field of stock brokerage. This exam is a prerequisite to work as a stockbroker in Hungary to trade financial instruments. The fundamentals coupled with the personal interest in making sound financial decisions is the motivation for the author to choose this field as the subject of this iterative full stack development project.

The author's expertise in finance is rooted in his undergraduate studies in Business Administration at the Universidad San Francisco de Quito in Ecuador, where he completed coursework covering statistics, financial market analysis, accounting among other related courses. Additionally, he has obtained certification in stock brokerage through a successful completion of the Stock Market Qualification Exam in Hungary, a common requirement for individuals seeking to work as stockbrokers and trade financial instruments in the country at the time. These foundational qualifications, combined with a genuine interest in sound financial decision-making, serve as the primary motivation for the author's selection of this field as the subject of the iterative full-stack development project. From a technical, software engineering standpoint the author has been working as a full stack and as a frontend software engineer for more than seven years at the time of the undertaking of the thesis project in several companies in the fields of e-commerce, affiliate marketing, Enterprise Resource Management (ERP) software and educational software development. This work experience was focused on web-based technologies, and the workflows followed in different teams included various methodologies including Agile principles-based project work and Scrum framework.

Currently, there is an increasing public interest in the future of human-machine interaction and some studies show that the most significant performance improvements can be achieved by organizations that employ AI-based technologies and humans and AI-models are working together in the decision-making process (Haesevoets et al., 2021 as cited in Wilson and Daugherty, 2018). In the light of this observation, this project also falls into the category of

machine assisted human decision making and one possible iteration of improvement that could be foreseen at this point is, AI-assisted analysis of the related financial reports, usage of Large Language Models to find anomalies, opportunities and to assist in the development process itself.

1.2 Research goal and questions

The goal of the author is to create a full stack software application, with an iterative agile development methodology and continuously evaluate the findings and consequent result during the research period. The project is intended to be flexible therefore it has a limited set of requirements in the beginning of the project. The following are the initial criteria for the software:

- Self-hosted application, instead of being deployed on a cloud service provider.
- The software can periodically query APIs for financial data.
- The software algorithm can analyze the data and handle errors.
- If specific criteria are fulfilled, such as the presence of a buy signal, the user is promptly notified via a notification method.
- The user can access the analysis and recommendations based on the financial analysis remotely.

The limited software engineering criteria is suitable for a Minimum Viable Product (MVP) production approach, allowing the product to be delivered for potential users with the minimum requirements as soon as possible. In consequence, iterative changes can be made earlier, which in turn could result in delivering increasing added value based on the users' feedback. The value in this process derives from the conclusions about the development process itself and decisions made regarding full stack technologies, development methodologies that can be incorporated in future software projects.

The decision to self-hosting the software, is partly due to the enhance the full-stack aspect of the project, partly because of other reasons such as:

- Cost: Self-hosting a server can be cheaper

Given a scenario, where a Raspberry 4b is used as a server, which consumes approximately 5W energy and is turned on 24 hours a day as server the cost calculation is as follows:

$$\text{Energy Consumption} = (5 \text{ W} \times 24 \text{ hours}) \times 30 \text{ days}$$

$$\text{Energy Consumption} = (5 \text{ W} \times 24 \text{ hours}) \times 30 \text{ days}$$

The above equation equals 3.6kWh energy consumption per month, while the price of electricity for 1kWh in Germany at the time of writing (19.01.2024) is 0.278 EUR.

This results in a cost of maintenance (given an already existing internet connection) is merely 1 EUR/month, compared to one of the cheapest cloud providers, Digital Ocean which costs 4 EUR/month at the time of writing.

- Data-privacy: All required data can be stored on a personal device with full access control.
- Control: There are no legal or policy considerations using a third-party service and any software deemed necessary can be installed and managed as necessary for the project

From the application output performance perspective, there are limited expectations. This study is not an exercise in creating a highly sophisticated and accurate financial advisory software. Instead, it is the delivery of a functioning software application that is working based on the criteria and can be iterated upon. In the same manner as software architecture, financial analysis also encompasses many competing views on what constitutes best practices and methodologies for investment. The results for financial investment can be measured based on an arbitrarily chosen holding period of assets. This results in the fact that the results can't be viewed objectively, but only in relation to a chosen time frame. Therefore, the goal of the thesis project doesn't involve returns on investments and a complete system that involves sell

signals or investment management advice beyond the buy signal on a set of criteria. The expectation is to receive correct signals based on the given market data and on the given criteria.

The research questions are:

- What are the learnings from the iterative planning and execution of a time-constrained project that has a goal of developing of a full-stack MVP application?
- What methodologies and tools allow for efficient iterative software development?
- What are the learnings from this project that can enhance professional software development skills?
- What continuous improvements could be implemented in the initially deployed MVP?
- What are the learning and conclusions that can be drawn based on the continuous self-reflective decision-making process?
- What kind of new research paths and questions can we derive at the end of the development and reflective documentation of this thesis project?
- How suitable is the chosen technological stack from a User Experience perspective?

1.3 Research method

The chosen research method for this thesis project is the diary-based method. This is deemed as an ideal research method by the author for a practical, iterative and time-bound research project that is based on the development of a software application. The “diary method is an adaptable research method for recording time-sensitive and context-specific details of a phenomenon” (Unterhitzberger & Lawrence, 2022).

Each choice of technology stack, implementation detail and the financial valuation method will be part of the body of the study as the decision-making process, its quality and outcomes are part of the study itself. The chosen diary-method emphasizes self-reflection, problem solving and learning during a research period. One of the fundamental benefits of this method is the “dramatic reduction in the likelihood of retrospection, achieved by minimizing the amount of

time elapsed between an experience and the account of this experience” (Bolger et al., 2003, p. 580). This means that the less time between the experience and taking notes results in less bias due to retrospection. Therefore, all the progress in the thesis project will be documented on the same day. The diary-method design in the current case is event-based, the triggering event being active work on the topics related to the project. This is planned, however not guaranteed daily, due to potentially unforeseen blocking events and time availability due to work and other commitments. The research will be done in the fixed period from 22nd of January 2024 to 1st of March 2024, in which the author expects to be able to answer the initial research questions. There will be a retrospective summary of the week’s events on every Sunday of the period. The entries will be written from a first-person perspective.

2 Diary Reporting

This chapter contains the diary entries for the research period, organized into weekly sessions and the corresponding weekly retrospective summaries.

2.1 Week 1

2.1.1 Monday 22 January 2024: Project setup.

The first decision to make is regarding the software and hardware tools that I need to organize, develop and document the process of full stack application development, which in the end of the research period is expected to automatically query an API, analyze the data and to send the user a buy signal notification if certain conditions are met. The ultimate purpose of this is providing value to the user by delivering timely financial notifications that can be further investigated by the user and therefore financial commitments could be made based on these signals.

The list of tools that I would require based on my experience are the following:

- A software repository.

- Project management tool.
- Note taking and brainstorming tools.
- A server device to host the developed application.
- A code editor.

For development versioning a time-tested and widely used development platform is GitHub which is owned by Microsoft, allowing versioning and serves as a code repository. In my professional and personal life, I have been using GitHub for software projects. It is based on the Git versioning system, which facilitates tracking the development process and safely storing the source code. In 2020 I participated in a university level course called the Methodologies of agile software development, where we have also used the project management tool integrated into GitHub for project planning. It is a simple, user-friendly and convenient way to keep track of the progress on the same platform.

There are many competing views regarding project management methodologies, but from personal experience, I prefer the method called Kanban. It is a Japanese scheduling system that was originally developed by Taichii Ono at the Japanese car manufacturing company Toyota to provide a framework for improved manufacturing efficiency. This is achieved by a card system, where each card represents a task and the number of concurrent tasks in progress are limited. These cards are represented on a board, where the progress can be tracked by moving the cards from the leftmost to the rightmost columns. The initial stage is adding the tasks to the leftmost column, which usually represents the backlog and when the requirements are established the card can be moved to the in-progress and finally to the ready column. There are variations in the number of columns based on project requirements. The tasks that are in progress are limited to three. Traditionally all finished tasks are going to be located on the rightmost done column in the end. Fortunately, the GitHub project management tool has a Kanban board, therefore providing both a software repository and a project management tool for this endeavor.

As a next step, I have established a new private repository on the platform on the following URL: <https://github.com/tonytettinger/automated-equity-valuation-full-stack>

Following that I have set up a Kanban project management board for this project as shown on Figure 1.

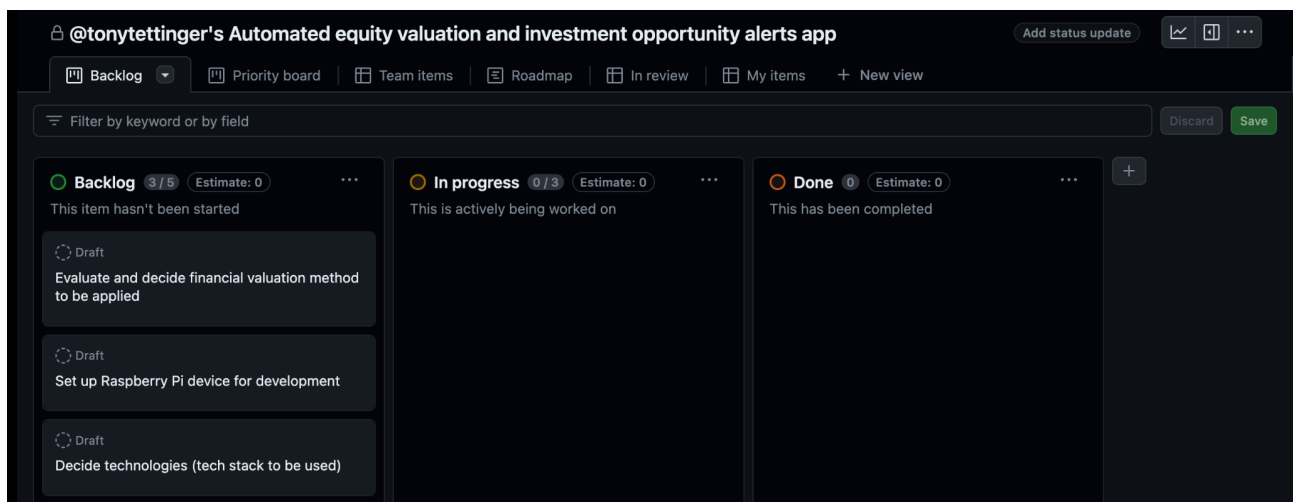


Figure 1. Kanban board on GitHub

For note taking I have set up a thesis repository on a note taking app called Notion, that I will use to collect project related information. I have also set up another tool called Obsidian on my iPad device; in case I would like to note down learnings that I want to add to my personal knowledge base. This note taking tool allows the user to store notes in an open-source format called Markdown, which is platform independent therefore the data can be exported in this platform-independent format, which is an important consideration in terms of data ownership. For this reason, I am already using this tool for permanent knowledge storage, while Notion is a great tool for dynamic note taking with an easy-to-use user interface.

Mind mapping and brainstorming are also concepts that are frequently incorporated into the project management process. For this purpose, I chose a software solution that is simple and readily available, called Freeform and is a digital whiteboard application, which is designed to encourage brainstorming. I have also acquired a Raspberry Pi 4 Modell B device with 4GB RAM.

It is ideal to use as a low-cost server and to deploy and run the developed algorithm in a cost-efficient manner. The chosen code editors for the project are WebStorm and PyCharm. I have been working with these for some time and they can handle the programming languages Python and JavaScript that I am planning to utilize in the project and speed up the development process with automated refactoring, syntax highlighting, formatting, integrated version control and efficient code navigation. I have chosen Python and JavaScript because these are the programming languages that I have study and work experience with, and both have a large community, helper libraries, packages and off the shelf solutions that could make the development process more efficient.

With the tools and related accounts set up I have added the following tasks to the Kanban board:

- Evaluate and decide the financial valuation method to be applied.
- Set up Raspberry Pi device for development.
- Decide technologies to be used for the project.

2.1.2 Friday 27th January 2024: Financial valuation method for the application

The goal of today is to choose the financial valuation method to be applied to derive a buy signal that can be presented to the user of the application. The effectiveness of exact financial decision strategies is impacted by various factors. These decisions are influenced by human psychology which is one of the principal causes of irrational investment decisions. Behavioral finance offers insights into the reasons for suboptimal or irrational financial decisions by individuals. These reasons include herd behavior, overconfidence, over and underreaction and loss aversion (Baser & Dashora, 2023, pp. 557-559). Building on these concepts, Baser and Dashora (2023) argue that one of the defining factors in achieving success in investing is to implement a robust investment strategy. Additionally, they highlight that “behavioral factors can assist investors in avoiding errors, a practice known as employing defensive behavioral finance applications in the decision-making process for investments”.

The aim of the algorithm deployed in the application to be developed is not to give a proven above the market average investment return decision. The purpose of the application is to enhance human decision making and ability to process data and give users a tool to that can be used as a basis of a robust investment strategy. This can be achieved by providing an investment buy signal based on predefined criteria which removes some of the behavioral biases such as herd behavior, loss aversion and underreaction while being able to automatically process large quantities of data. It is not the aim of this project to create an algorithm or trading system that could be proven to result in above average investment returns. Instead, to provide a tool that is an example of one of the many ways human decision making can be enhanced by automated software solutions. The user should be able to receive notifications and a summary of recommended stocks based on periodic querying and processing of market data by the software. The recommendation summary for these stocks could be an email containing plain text, a PDF or an interactive web page containing graphs and analysis.

The stock markets such as NASDAQ and NYSE in the United States of America are some of the most suitable for this exercise, because the Securities and Exchange Commission (SEC) regulations enforce a high level of transparency in terms of financial data and reporting. This data is easily accessible via financial API providers for example Yahoo Finance, which is a primary candidate for this purpose based on preliminary research and prevalence in tutorials regarding stock market data processing, therefore I have decided I will try to use it for this project.

One of the difficult decisions to make is what financial model to base applications buy signals on. I have previously been exposed to various methodologies in my university courses, my stock market qualification exam and in my preparation for the Certified Financial Analyst (CFA) courses. By the very nature of how the market works, it is not feasible to implement and apply a widely accepted consistently outperforming method, because as more investors apply the method in the market, the more results would return to the mean. There is a great range of financial valuation methods, for example based on:

- financial ratio analysis such as Free Cash Flow Yield, Return on Equity (PE), Price-to-Earnings Ratio (P/E).
- qualitative assessment of management quality and the potential total market size that could be addressed by the product - market-based approach.
- book value or liquidation value of the assets owned by the company - the asset-based approach.
- present value of future cash flows and growth potential in earnings - income based approach.
- option pricing-based approach.

With the increasing quantity and availability of information, valuations become more complex, and the investor faces a trade-off between attempting to make better forecasts by using more specific information and input or using a simpler model. Damodaran (2011) argues that more inputs create a potential for errors and results in opaque models. Instead, he recommends drawing from the scientific principle of parsimony and using the simplest model possible. Seessel (2022) in his book “Where the money is?” also states that “judgment in the investing world is often mere common sense girded with a framework to help you organize your thoughts”. Based on these arguments and drawing from my personal experience, my intention is to choose a simple model for this exercise.

My personal preference among the models is based on the present value of the income generated by a company. In my subjective opinion it minimizes the assumptions required to predict future outcomes of investment results and it is quantitative, therefore suitable for automated analysis. One of the models using this way of valuation is called the Net Present Value and it is a forward-looking analysis method that was formalized by Irving Fisher and was popularized during the Great Depression by Benjamin Graham (Bichler & Nitzan, 2010). The discounted cash-flow (DCF) valuation method is one of the Net Present Value based approaches. It is focusing on the income of the companies that must be discounted, because of the concept that the future value of money is less than the present value, due to cost of opportunity, inflation and uncertainty. I believe the concept is fundamentally robust, because it

derives the value of a company based on its capacity to generate revenue, therefore removing biases inherent in relative or qualitative valuations. Its various forms and variations are also widely used in the investment industry.

However the key elements of this estimation are the cash flows estimated in a forecast period, which is usually three to five years, and a terminal value at the end of this forecast period that represents the value of the company considering a perpetual income stream into the future. The rate with which the investor discounts the future cash flows is called the discount rate, which is called the Weighted Average Cost of Capital (WACC) in the DCF calculation. The WACC estimates the cost of both debt and equity, therefore it can be a good measure of the required return and can be used as a discount rate. If we calculate the WACC, we can estimate the terminal value of the stock by discounting the terminal value by it.

I have derived a method for the DCF calculation utilizing online resources (Learn to Invest - Investors Grow, 2019), with a few modifications and in simplifications and it is based on the following steps:

1. Calculate the Free Cash Flow to Equity (FCFE) value for the past 4 years from the Total Cash Flow from Operating Activities and Capital Expenditures. Based on that the FCFE / Net Income ratio. Take conservatively the lowest one of these ratios.
2. From the Total Revenue and Net Income, we can calculate the Net Income Margins for the past 4 years, and then we can take the most conservative estimate which is the lowest number of these.
3. With the calculated Net Income Margin, project the Total Revenue for the next 4 years, and then consecutively derive the future Free Cash Flow values by multiplying these values with the FCFE / Net Income Ratio calculated in step 1.
4. To discount the future values to their present value, the WACC must be calculated. This requires calculating the weighted average cost of debt, plus the weighted average cost of equity. To calculate this, it is a required to gather the values for the following:
 - Interest Expense

- Total Debt
 - Market Capitalization
 - The Beta coefficient which measures the asset sensitivity to market movements
 - Tax Provision
 - Risk-free rate
 - Expected return of the market
5. With the WACC and an estimated Perpetual Growth Rate value the Terminal Value can be calculated, which is the estimated value of an asset at the end of the forecasting period. We can calculate this with the Perpetuity Growth Model based on the following equation:

$$V_0 = \frac{FCFE_0 \times (1 + g)}{r - g}$$

Adding up the discounted Net Present Value of the Terminal Value and cash flow estimates for the following four years the DCF value can be calculated.

6. If the difference between the DCF value and the total market value of the company is positive, a buy signal can be generated applying a safety margin. This means for example if the set safety margin is 20%, if the DCF value is 20% higher than the market value of the stock then it would generate a buy signal.

The purpose of this thesis is not to validate or explain in a detailed manner a method widely used by the financial industry such as the DCF method. The purpose of this method is to provide a basis for an algorithm applied in a full stack software engineering project, which can be iterated upon based on the results and can serve as a starting point for further evaluations and improvements. There are several inherent estimations and guesses that can be second guessed, such as the market return rate, or the Beta coefficient, but such is the inherent nature of forecasting the future in a dynamic environment.

To summarize, I have gathered the data that I need to calculate the estimated fair value based on a DCF calculation and verified that the following data is available on Yahoo Finance:

- Total Cash Flow from Operating Activities and Capital Expenditures
- Total Revenue
- Net Revenue
- Risk free rate (10-year treasury US treasury bond)
- Beta coefficient
- Number of outstanding stocks
- Current market price
- Pretax income
- Tax Provision
- Interest Expense
- Total Debt

Furthermore, I will use an estimation for the following variables:

- Perpetual Growth Rate: 2.5%
- Expected return of the market: 7%-10%

Processing the above data, I should be able to derive the estimated value of a stock based on the DCF valuation, and applying a safety margin, I can set a buy signal. By querying data from an API, I should be able to process many stocks based on their current price and identify investment opportunities that fit the signal generation criteria.

2.1.2 Weekly reflection

During the week I have managed to set up the project framework, the Kanban board and decided about the financial valuation method to be used in the project. It is a compromise, but hopefully it is a good basis to develop an MVP. I have contemplated about the usage of artificial intelligence (AI) based tools that can possibly quantify qualitative data based on certain criteria,

such as analyzing management traits based on analysis of investor relation communiques by the management, or sentiment analysis of news or investor forum threads.

However, there is a limited time available, and the scope must be adjusted accordingly. Even with a simple model such as the DCF, there are a lot of variables, a lot of assumptions that could be argued with. For example, the method might not be suitable for a company which just took a large amount of debt in the current year or a company with negative cash flow. The expected growth rates and market returns are also based on educated guesses and on historical averages. The reality is that no matter what model one is using for prediction it will not yield a perfect result because of the nature of the financial markets and its feedback loops and self-reinforcing mechanisms. I have settled with the DCF method, because it is based on the ability of a business to generate cash, which in my opinion is a neutral way of looking at a business and relatively easy to quantify. Nevertheless, even this model resulted in many variables to consider and a feeling of needing to justify and describe decisions in detail, which took a considerable amount of time. I have also gathered which variables I will need to query and found that Yahoo Finance has the values I need to perform the calculations, that would result in an investment buy signal. I consider this a good progress.

What I've found interesting is that I am managing this project, adjusting the scope and managing the time and acceptance criteria as one person. It prompted me to reflect about similar projects, where a single developer developed a successful software project as an individual. One of the most successful examples is the Linux operating system that was developed by Finnish software engineer Linus Torvalds (Britannica, 2024). However later it became an open-source software project, developed by thousands of contributors. There are some edge cases such as the game RollerCoaster Tycoon, which was developed by a sole developer Chris Sawyer, written in 99% machine code (Sawyer, n.d.). However, in general it appears that software projects are a group endeavor or become one when they are scaling up. I believe this project has the potential to provide interesting self-reflections, insights and learnings of a software development process by an individual developer. It is interesting to reflect on the different roles that have evolved in the software development world, such as

project manager, product owner, separation of different developer roles such as frontend, backend, DevOps as an example since in an individual project all roles are combined in a single developer.

I was hoping to be able to start with the development process, but the discovery, description and justification of the theoretical model behind this application took much more time and effort than I had expected. At the end of this period my Kanban board looks as follows on Figure 2.

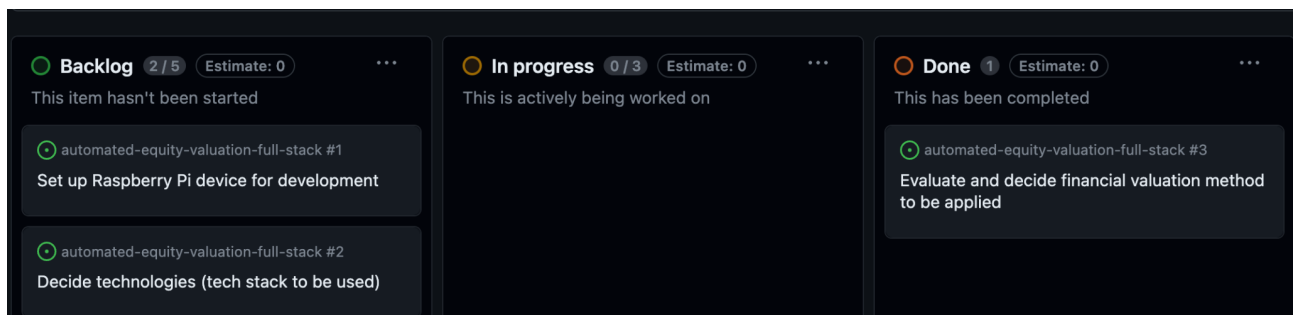


Figure 2. Project Kanban board at the end of week 1

2.2 Week 2

2.2.1 Tuesday 30 January 2024: Hardware setup

The goal today is to set up the Raspberry Pi 4b for the project. The first choice to make is regarding the operating system (OS). The Raspberry Pi Imager software makes it very easy to install a wide range of operating systems to the 32GB micro SDHC memory card that I have acquired together with the device. I am using my regular MacOS operating system for this operation as I intend to use the more powerful MacBook Pro laptop for the more compute intensive tasks, such as code editing, browsing and researching material. Hence limiting the use of the Raspberry device for what its original purpose is: running code continuously in a cost-efficient manner and serving as a server for the project. I have chosen the recommended Raspberry Pi OS (64 bit), which is a Unix-like operating system that is based on the Debian GNU/Linux distribution with a graphical interface included. Since it belongs to the Linux OS

family, I can take advantage of the fact that I have some experience setting up servers on cloud-based servers which were also Linux-based, and I am familiar with Linux related command line operations and shell scripting.

I have managed to successfully boot the operating system, but I have encountered an issue immediately. I didn't have a wired computer mouse peripheral to connect to the device, only a wireless Bluetooth enabled one. I have managed to resolve it, by navigating to the command line with a wired keyboard and researched how to complete the Bluetooth pairing only utilizing the command line, without the usual visual interface which I couldn't access without a mouse. After this obstacle was successfully resolved I have immediately updated the software on the device and made sure that the Python programming language is available. At this point I had the device up and running, but the next step was finding out how to interface with this device.

In the future I will need to be able to access the device via the local network or preferably via the internet to be able run commands and operate the device remotely. The first option that came into my mind was enabling Secure Shell (SSH) connection which is a network communication protocol that enables computers to communicate and share data. This is a common way to manage servers remotely via the command line. This would limit my interaction to command line commands, which wouldn't allow me to open applications with graphical interfaces such as a browser. Consequently, after some research I have found a way to connect to the device via Virtual Network Computing (VNC), which is a graphical desktop-sharing system developed in Cambridge, UK in the late 1990s (RealVNC, n.d.). Having a VNC server enabled on this device would allow me to access the device directly with any device that has a VNC viewer installed. Connected with a VNC Viewer software a user can see the same thing as a person sitting in front of the remote computer. It uses a protocol called Remote Framebuffer (RFB) to pass data between the client and the server. In theory the Raspberry Pi OS supports VNC connection, but I have found out that the setup didn't work as described. After some research I have found out that the newest version of the Raspberry Pi OS uses a different display server called Wayland than the previous one called X11, and it is not supporting VNC. The display server is responsible for handling the display of graphical elements

on the screen. With some research I have found that with a configuration command I can open a graphical configuration interface, where I can set the previous X11 display server to be used. Following this change I could now enable and start the VNC Server on my Raspberry Pi device. I set up a password, verified the device IP address and then installed the VNC Viewer on my MacBook Pro and successfully connected to the device with this interface and following this I could operate the device via the MacBook peripherals.

The next step was to enable remote device access via the internet. The Wi-Fi router defaults to blocking all outside connection requests, which is beneficial for security, therefore I had to open the port 9000 to be able to access the VNC from outside and forward the requests to my Raspberry Pi. I have accomplished this by accessing the router settings in my local network. Following this I have managed to connect to my device even while walking outside on the street from my mobile phone using the mobile network as shown on Figure 3.

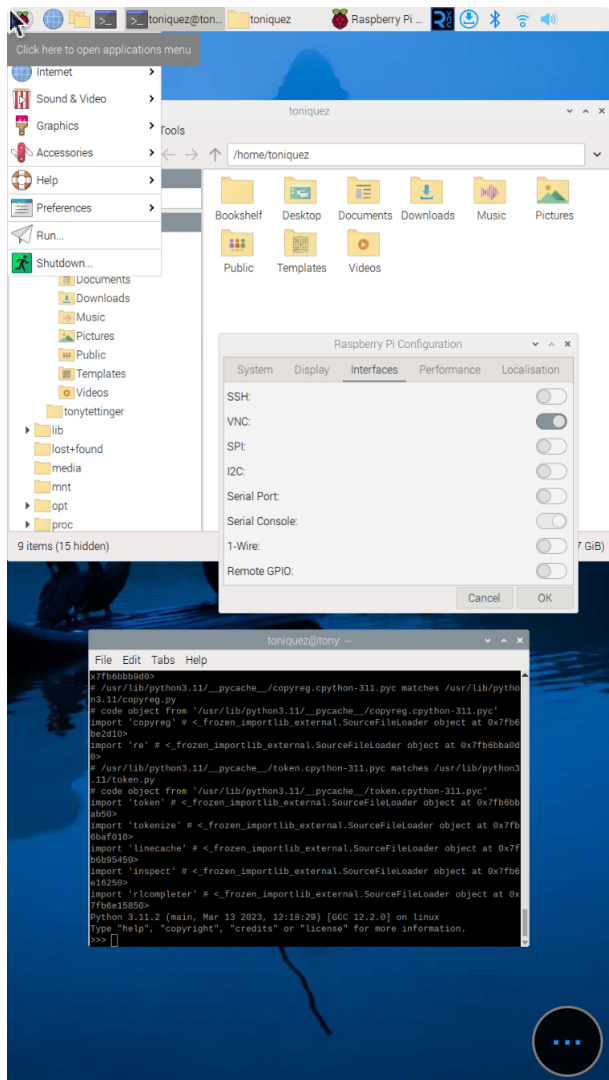


Figure 3. Accessing the desktop of the Raspberry Pi from my iPhone SE mobile device

The only thing left was to make sure that I don't have to rely on the IP address of the device, which can dynamically change, and it would mean I lose access to my device, because I couldn't know the actual IP address. After doing brief research I have found the solution is Dynamic DNS (DDNS), which is a public hostname for dynamic IP addresses. I have found a free service to create such a hostname, called dynv6 on the web site <https://dynv6.com> and created a domain. Using the built in DDNS service included in my router I linked this domain to my router. This works by automatically and periodically updating the DNS's IP records when the IP address changes. These address changes are out of my control and are made by the internet provider, therefore this solution is necessary to have a stable connection to my device from anywhere

with an internet connection. At this point I had my Raspberry Pi connected to the internet; I could access it via a public hostname from any internet connected device that has a VNC viewer application installed.

2.2.2 Wednesday 31 January 2024: Decisions about the tech stack for the application

The next item on my backlog was about deciding the tech stack to be used. There are two programming languages that I feel comfortable programming with, Python and JavaScript. They both have good frontend and backend solutions. If I would want a more performant server language, I would choose a compiled language as opposed to a slower interpreted language like JavaScript and Python, but since performance is not a primary driver in the case of this application, instead it is the speed of development. In this regard both Python and JavaScript have a lot of helper libraries, which helps to speed up the development process. One of the central questions at this point was, how to query, process, send and consequently view the data.

The signal generation process by the application is based on the following idea: Every day at a certain time, for example shortly after the market closing time, the application queries the required financial data. This data is then processed by an algorithm, based on the DCF model and if there is a buy signal the user would be notified and would be able to access details in some form of a view. The following initial questions arise based on this flow:

- What is the best way to query the data periodically? How to implement it from a technical perspective and with which data provider? Is Yahoo Finance suitable from every aspect to provide the data with its API?
- In what form to notify the user if a signal is generated? Email, mobile application notification, some form of webhook that could integrate with a messaging application such as Slack or Discord messaging services?

- How to represent the results of the buy signal? What should be included? Should it be only text, or should it feature some form of a UI, graphs or tables? Should it be in an HTML format so it could be accessed on a web browser?
- How to store results? My first idea is SQLite, which writes a database in a file, but using MongoDB or other databases are also an option. Which data needs to be stored?

At this moment I believe that the processing algorithm shouldn't be a problem, because I have already determined the methodology and it can be implemented with different programming languages. The main obstacle is obtaining and processing the required data, which I should tackle next. Because of this reason I am adding a new task to my Kanban board to decide which API to use and test the API by querying data with it, which I intend to follow-up in my next session. One of the most popular free ways to query financial data is from the free API of Yahoo Finance. I have done some research, and it might be rate limited, therefore it might not be suitable for querying large numbers of stocks reliably, this must be investigated. One advantage of it is that there is a Python package that allows easy access to this API, which is called `yfinance` and one called `yahoo-finance`. I have also read about how to automate emails with Python and in general Python has many libraries and community packages, for analyzing data (`pandas`), artificial intelligence (`PyTorch`), plotting graphs (`Matplotlib`). Another tool that can be useful is Jupyter Notebooks. It provides an interactive and flexible environment for data science education and could be used for documenting or reporting, since these notebooks can be exported to various formats such as HTML or PDF for example. Because of these reasons and its ease of use and my personal familiarity Python will be part of the tech stack used. It might be necessary to scrape some web pages for data that might not be available immediately, for that I have the `Playwright` library in mind, which is a Microsoft owned end-to-end (E2E) testing tool that could be also used for that purpose, and it is very easy to create web interactions. This tool and many other web related tools are JavaScript based; therefore, JavaScript would also be part of the stack, especially if later I would decide to create a web page most probably I would use a JavaScript based frontend web framework or library. Therefore, to summarize, the expected tech stack is Python, JavaScript and SQLite or MongoDB. The exact helper libraries must be discovered, likely Jupyter Notebooks, `matplotlib` and `Playwright`.

2.2.3 Thursday 1 February 2024: Data querying

I have set up a Python project on my MacBook Pro with the PyCharm editor. I have installed the *yfinance* package. But as soon as I have tried my first query with it based on the documentation, I have encountered an error message:

```
NotOpenSSLWarning: urllib3 v2 only supports OpenSSL 1.1.1+, currently the  
'ssl' module is compiled with 'LibreSSL 2.8.3'.
```

What I understand from this is that the currently used library inside the package called *urllib3* v2 is using an SSL library that is currently not supported by my operating system. I don't want to go deep into the reasoning as I believe some part of being a software engineer is choosing which details to ignore. I have found on the website Stack Overflow that MacBook users have successfully resolved similar issues by downgrading the *urllib3* library to a version below 2.0. I have re-run the Python code and it successfully retrieved generic information for the AAPL ticker, which belongs to the Apple Inc. tech company. However, I also wanted to see if it works with some of the other variables that I need based on my preliminary research and I have tried to query the Beta value of the AAPL stock which returned undefined. I have checked on the Yahoo Finance homepage and there it is listed as 1.29. I have tried with two different stocks. With the Meta Platforms, Inc. (META) stock I also got undefined, but on the summary chart of Yahoo Finance it is listed as 1.22 on the web page. After this I have also queried the 3D printing company 3D Systems Corporation DDD and I got back its beta correctly as 1.851, this was verified against the value on Yahoo Finance web page. These results make me think that there might be some better, more reliable alternatives, because the Yahoo Finance API doesn't seem to return data that is seemingly available on the web page itself.

After I did some more research, I found an alternative provider called Alpha Vantage. Following the creation of a free API token I have tried to query the Beta value for the same stocks, and it was successful for both AAPL and META, the value was cross matched with the values from the website of Yahoo Finance. I have decided to continue using this API for now, because it seems

to be more reliable, and it has a paid upgraded version that has even more data and querying possibilities. At this point I wanted to go through my list of data requirements and test the query for each item, but as I have queried the Cash Flow data, I have realized I need to display the data properly to verify the returned values, because the console view is not adequate for this. For this reason, I have decided I need a Python framework that facilitates creating an application that can display the data. After some investigation I have found some candidates, such as Django, Flask and Pyramid, all of which are popular frameworks. At one point in my career, I have used Django and Flask, and my company now is using Pyramid in the backend. I have cross-referenced the basic use-cases, pros and cons and I have found that for my purpose for fast development Flask is ideal, for the following reasons:

- Less boilerplate than Django and Pyramid.
- Provides built-in HTML templating language and methods to convert to PDF.

I have installed the Django environment with the PyCharm editor to evaluate it also, but it had a lot of boilerplates that I didn't want to spend the time studying to be able to work with. I have also tried to install Pyramid, but the installation failed, with a similar error that I had when I tried to use the *yfinance* package the first time, therefore I didn't continue debugging it and installed and started using Flask. Following this I have returned to try querying data and a few queries later I have started to receive key errors from the returned JSON data file. With some investigation I have discovered that the free tier of the service provider (Alpha Vantage) only allows a very limited number of queries, and the cheapest monthly plan is 24.99 USD. I have done a half an hour research for competitors, but either their price was higher, or they didn't offer the dataset I have set out to include in my calculations in the beginning of my research, therefore I have subscribed to this service, because it provides the best quality data, with clear documentation and a logical data structure. The data is also mapped to the US GAAP and IFRS taxonomies which are well documented industry standards for accounting. Following this I have spent a few hours setting up a reusable class, with methods that would allow me to iterate over queried data and collect this data into a Python dictionary.

For now, I have successfully queried the data that I need from the Cash Flow statement, which is the Operating Cash Flow and Capital Expenditures. I have aggregated these into a Python dictionary, including data for the past four years in an array as shown on the HTML based representation of the dictionary from a Flask template file on Figure 4.

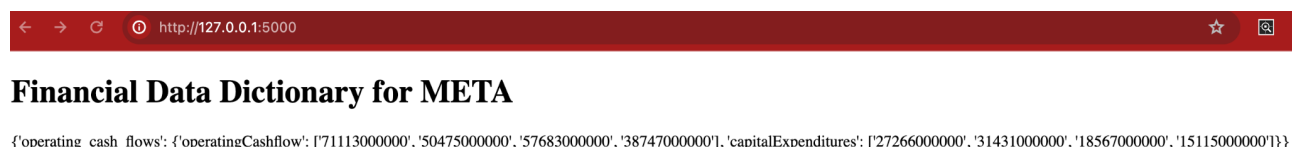


Figure 4. JSON data dump display for Cash Flow data aggregate for the META stock in for the past four years

So far, the application totals 70 lines of code, plus the HTML file to visualize the data. I expect that the methods that I have created will speed up the development of the rest of the data that I need for the calculations, which will be included in the following steps in the development process. Furthermore, this code should be committed to the GitHub repository, but I don't have time for setting and uploading the GitHub repository today. I have saved the work and made a copy of the body of code on Notion.so, which is not ideal, but better than not having a safe copy of the work.

2.2.4 Sunday 4 February 2024: Versioning and weekly reflection

I have initiated a git repository in the directory I have been working on and set the remote to my GitHub repository for the thesis project in the following repository:

<https://github.com/tonytettinger/automated-equity-valuation-full-stack>

I have also added several tasks to my project management Kanban board in the Backlog section, querying all the data required and putting it into a data structure into the "In Progress" column. At this point the Kanban board looks as on Figure 5.

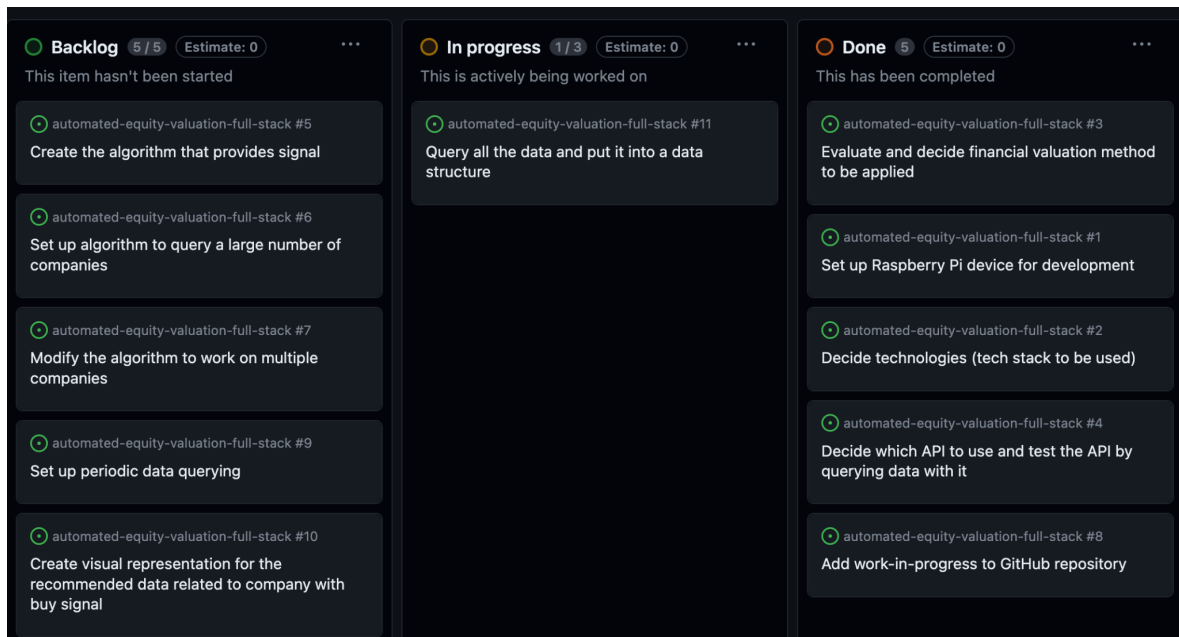


Figure 5. Kanban board as of the end of Week 2 of thesis project.

This week I have set up the hardware, encountered the different options on how to interface with it and came up with the solution of using VNC and in the process learned about the benefits of DDNS. In the end at this point I have a way to access all the functionalities of the device remotely. As a next step I have tackled the data querying from Yahoo Finance and have encountered inconsistency in the data and reliability. Researching possible alternative solutions I have found that the best cost versus benefit provider to be Alpha Vantage. The cost is still significant, and the data querying could be done in the future by applying crawling techniques and free solutions, but at this point moving fast to get to an MVP is the primary concern, therefore I have subscribed to the service. Following this I reviewed the technologies that could be used in this project development and identified a Python library that is suitable for rapid development with minimal overhead which matches my iterative MVP approach. I have used this library to create a simple HTML representation of the data and queried the Cash Flow related data from the API and some class-based helper functions to speed up the next development steps. At the end of the week, I have also added the developed project into the GitHub repository created for the thesis project. The following step is to query all the data required.

2.3 Week 3

2.3.1 Tuesday 6 February 2024: Querying the remaining financial data.

The following data is still required, to create the algorithm to analyze the data:

- Total Revenue
- Net Revenue
- Risk free rate (10-year treasury US treasury bond)
- Beta coefficient
- Number of outstanding stocks
- Current market price
- Pretax income
- Tax Provision
- Interest Expense

My initial observation discerned that my code needs to be more generic. I have refactored the code in a way that all the data querying and processing would be in one class, and then I intended to use the return value in a rendering template. This first resulted in error, since I have introduced an asynchronous (async) process into the class. Furthermore, the refactoring was done in a way to be able to loop through a few financial data types, for an array of companies. This in turn causes the code to loop through multiple asynchronous requests, that in the future I expect can be a large number. To resolve this, I had to research best practices of handling asynchronous code in Python, in particular awaiting for multiple async processes at the same time. Although Python is a single threaded language, I have found with some research a package called Async IO, which allows for cooperative multitasking. Cooperative multitasking in essence is a process where the Async IO sets up an Event Loop and then coordinates the tasks to take turns for each to take an optimal amount of time (Solomon, n.d.)

With this I have managed to successfully query data from multiple companies, and I have rewritten the HTML template in a way to display this data by companies as shown on Figure 6.

Financial Data Collection

• AAPL

```
{'operating_cash_flows': {'operatingCashflow': ['110543000000', '122151000000', '104038000000', '80674000000'], 'capitalExpenditures': ['10959000000', '10708000000', '11085000000', '7309000000']}}
```

• META

```
{'operating_cash_flows': {'operatingCashflow': ['71113000000', '50475000000', '57683000000', '38747000000'], 'capitalExpenditures': ['27266000000', '31431000000', '18567000000', '15115000000']}}
```

• IBM

```
{'operating_cash_flows': {'operatingCashflow': ['10435000000', '12796000000', '18197000000', '14770000000'], 'capitalExpenditures': ['1346000000', '2062000000', '2618000000', '2286000000']}}
```

Figure 6. HTML display of data retrieved from the Alpha Vantage API for debugging purposes shown here as a work-in-progress stage without styling.

As a next step I have extracted into a reusable function the code that is responsible for getting the sub-category data inside the retrieved main categories, such as the Net Income inside the Income Statement. I had to do some re-mapping of the original data, since for example in GAAP terminology Net Revenue is called Net Income. This way I have added the income related data to my data object successfully. I went ahead and created more methods in the class to query the remaining data points: Beta, 10 Year Treasury Rate, Interest Expense, Outstanding Stocks, latest daily price, Pretax Income, Interest Expenses. I had to do some name mapping, because income and revenue are used interchangeably, and some other terms have multiple conventions that differ from my originally defined terms. This highlights the importance of naming variables well and I have tried to maintain this principle while naming my methods in the code also.

At this point I had all the data from my original plan; therefore, I am moving the task called “Query all the data and put it into a data structure” from the “In progress” to the “Done” column on my Kanban board.

2.3.2 Saturday 10 January 2024: Calculating the first part of the Discounted Cash Flow

The current item that I am working on today from the Kanban board is about creating a method to calculate the buy signals. To start out I have created a new class called `CalculateSignal`. To implement this class the first step is to calculate the estimated Free Cash Flow values. For this, I needed three calculations:

1. The earnings growth to project the expected revenue for the following four years.
2. The net income margins in order determine the projected net incomes.
3. The Free Cash Flow to Equity / Net Income ratio to determine the Free Cash Flow value projections.

With the earnings growth, I was immediately faced with an issue. The initial idea was to take the lowest value, which was in the case of the AAPL stock was below 1, which means contraction. This would eliminate companies with only one recent year of decreasing revenue. Accordingly, I have chosen to switch to an average value instead. If these conditions would result in a less than 1 value, the company will not be suitable for a buy signal. The second calculation was straightforward with the previously queried data for Net Income and Total Revenue. As a developer who has been working with JavaScript based technologies recently it provided a good learning opportunity to develop these calculations, because I have learned how to use map methods and list comprehension in Python. The third ratio that I had to calculate was the Free Cash flow to Net Income ratio, which calculated by the formula for each period (last four years):

$$\frac{\text{Free Cash Flow} - \text{Capital Expenditures}}{\text{Total Net Income}}$$

Here I have also modified the originally planned idea and took the average value of the four results. Here I have also utilized a Python method I was not familiar with earlier called zip, that allows me to dynamically create tuples from two lists in Python. The last step was to calculate the estimated future Revenue, Net Income, and Free Cash Flow values. In total this new class with the helper methods consists of 60 lines of code and with this I am ready to start the next step, which is the WACC value, which is in effect the required return on investment. In hindsight I should have separated this algorithm into three tasks, one for the Free Cash Flow calculation, one for the WACC and then finally to calculate the Terminal Value and the Net Present Value of all the calculated future values. Despite this I am just going to continue to work on this task, since the steps are clear. I am committing each new feature to the version control system with a commit message that makes the purpose of the changes atomic, clear to follow and reversible. I have also moved the previous class and its helper functions into its own module, to create an organized and clean code structure.

2.3.3 Sunday 11 January 2024: Weekly reflection

This week I have developed the code that is responsible for querying and inserting all the required data for the calculations into a data structure. I have finished approximately half of the algorithm that would provide the analysis and the buy signal. The ease with which I have managed to query the data from the API seems to validate the idea of pivoting from using Yahoo Finance. I have also learned and re-learned some Python specific programming language skills and I am getting more comfortable with class-based programming at this point. The HTML display of the results was helpful in the process of debugging errors and to verify the outcomes. I had three days when my internet service was not working due to an outage, and I also didn't have any data balance to work with. These external factors and unforeseen events are also part of the time bound MVP development experience. They slowed down the process but didn't create any major bottlenecks or obstacles and I am still on track. As I was adding the calculations in the process, I started to contemplate about the number of assumptions that has been made and the effects of these. For example, the growth rates can significantly impact the calculated DCF values. Nevertheless, having it coded in a clean and modular way would allow me to change the variables and test if the model is robust or if certain assumptions could

significantly alter the outcomes. I have also added a task to the board about back testing, which is a method to evaluate the trading strategy model based on historical data. At the end of the third week the Kanban board state is as shown on Figure 7.

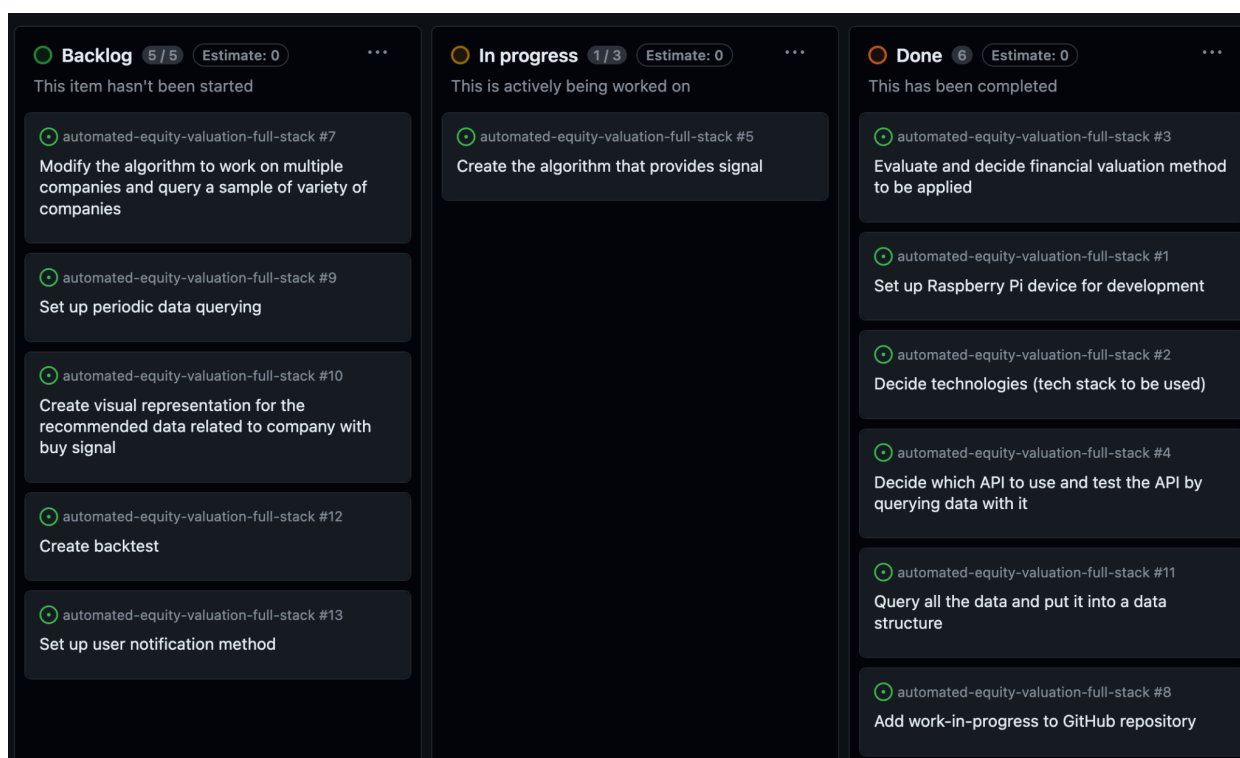


Figure 7. Kanban board status at the end of week three.

2.4 Week 4

2.4.1 Tuesday 12 February 2024: Calculating the WACC and DCF.

Today I have developed the function to calculate the WACC, which is in effect the required rate of return, which is based on two parts: the cost of equity and the cost of debt. These values are respectively multiplied by the proportional weight of total value equity and total amount of debt to obtain the total required rate of return. For the cost of equity, I have calculated the expected return from the Capital Asset Pricing Model (CAPM). CAPM is a widely used framework in finance for determining the expected return based on the risk level. Based on it the cost of equity can be calculated as follows:

$$R = R_f + \beta * (R_m - R_f)$$

In the equation R is the cost of equity, R_f is the risk-free rate, R_m is the market rate and β is the beta value queried from the API, which represents the volatility of the investment relative to the overall market.

To determine the cost of debt, I have calculated the proportion of interest expense compared to the total amount of debt based on queried values from the API. There were two values that I have defined as a constant. One is the expected market return, the value of which I have based on the historical average between 1957 and 2023, which is 10.26% per year (Webster, n.d.). The other value that I have defined as a constant is the perpetual growth rate, which is a value that could be estimated as a percentage between the inflation and the GDP growth averages. The Federal Reserve of the United States has a 2 percent inflation rate goal (Board of Governors Federal Reserve System, n.d.), therefore I estimated this value to be 2.5%. Having calculated both the equity and debt costs and their proportions I have derived the WACC.

As a next step I have calculated the Terminal Value, which is an estimated Net Present Value of the discounted cash flows in perpetuity for the future starting at year five. Having all the cash flow estimates from year one to four and the Terminal Value estimation for year 5 and the expected return, I could now calculate the present value of these numbers, which results in the DCF value of the company, with which in effect accomplished my goal to evaluate a given company.

I have run a few tests for the valuation for several companies and found that for the IBM stock the market capitalization at the time of the calculator was 170.015 billion USD, and the DCF calculation valued the company at 172.619 billion USD, which looked promising. I would have interpreted it as a cause for concern if the numbers would have been orders of magnitude off compared to the valuation, which was not the case. The next step will be to create a summary with relevant information for each company where the market value is below the calculated

DCF value minus a safety margin percentage. The list of summaries will be available for the user after being notified via one of the notification methods.

2.4.2 Wednesday 13 February 2024: Bug fixing.

Today I have re-run the algorithm with different companies drawn from a larger sample size, and I have found that the results were unrealistic in some cases, resulting in orders of magnitude higher valuations than the market price. I have spent a considerable amount of time reviewing every calculation and related methods in the code, printing all the partial calculations in the console for debugging purposes and I have found a couple of issues:

- There are unexpected values in the API responses, such as string values of 'None'. This causes errors and unexpected behavior. To address this, I have implemented a conversion of these values to the number zero and applied this conversion before inserting these values into the data structure.
- While calculating the Earnings Growth I have accidentally used Net Income instead of Total Revenue. I have addressed this by correcting it and replacing it with the correct value.
- When calculating the discount percentages, the array started already with the n squared value which should have been the second element in the array. I have addressed this by ensuring that the array starts with the correct value.
- The value of Net Income Margin was calculated incorrectly from the projected revenues, the calculation has been corrected and verified.
- In the case of negative growth rates, I have accidentally assigned the growth rate to 100% in the calculations. I have corrected this by setting the value to 0%.

After these corrections I have also implemented a sorting algorithm, to sort the results by Market Capitalization. I ran tests for a range of technological companies printing out results in an HTML form to get an overview of the results. In this overview I have listed the DCF, the Market Capitalization and the difference between the two, in billion dollars for readability and ease of understanding. In my tests I have added a few big tech firms, and some smaller ones.

One observation was that the smaller relatively new, emerging companies such as the 3D printing companies have negative cash flows, therefore their valuation is negative. These won't have the possibility to be included into the recommended companies list, because of the nature of early-stage companies having negative cash flow. This is however not a problem from this project's perspective, because it is not possible to cover with one valuation method the companies in different life cycle stages. Therefore, the calculated signal is applicable to established, positive cash flow generating companies. There were two positive results, where the estimated value is higher than the current market price. In the case of HP Inc (HPQ), the valuation was 53% higher and in the case of Cisco Systems (CSCO), there was a 16.92% difference. If I would set the buy signal to be tied to a 20% discount between the two values, the remaining candidate would be HP inc.

2.4.3 Saturday 17 February 2024: Database, deployment, notifications, static website

At this point having a working valuation model. Upon reflection, I've identified a critical oversight in the backlog planning regarding the timely delivery of a functional solution. Making sure that there is a working application can be deployed to the server is one of the key steps. Consequently, I've created and elevated this task to the top of the list. I had to make sure that the full stack application has a working model deployed before moving on to the other tasks.

I had to transfer the developed application to the server and make sure it can be run successfully. For this I had to download the contents of the GitHub repository to the device. I have set up the GitHub CLI via command line, added my credentials and pulled the repository to the Raspberry Pi while logged in via the VNC Viewer. After setting up the virtual environment and installing the required packages, I have successfully run the application. Following this I have added a method to generate a static HTML including results from the latest run of the algorithm. This worked well on my development device, but when deployed on the server no file was generated. I have discovered with some debugging statements, that it was due to file permission issues. After resolving this, every time upon refreshing the page I had a static HTML file generated based on the algorithm outputs and the device was querying the API and performing the same calculations as on the development device.

After verifying that the application is successfully running on the server, I have moved the deployment task to the Done column of the Kanban board and pulled three tasks into the In Progress column:

- Set up periodic data querying.
- Set up a user notification method.
- Allow the user to change values used in calculations on the user interface.

To be able to easily test different scenarios and to understand how changing some constants would affect the outcome of the calculations I wanted to enable the use to change the constant values. These values include the market return rate, the perpetual growth rate, the safety margin and ideally it could even include the companies to be queried. To achieve this goal and keep the code organized I have installed the `sqlite3` package as a project dependency and created a new folder titled SQL. In this folder, I have created a schema file and an `init_db.py` file where I have declared the initial values as 0.025 (2.5%) for the perpetual growth rate and 0.1 (10%) for the expected market return rate, based on the estimated value. Following this I have created a `DatabaseAccess` class, to contain methods to get the data from the database. At this point it became apparent that typing manually long strings such as `perpetual_growth_rate` is error prone, repetitive and not scalable. After brief research, I have imported the Enum library and declared these strings in an Enum class called `FinancialVars`. This way I won't have to rely on typing these manually without making mistakes. Instead, I can just choose from the available values from this class, where the autocomplete of the PyCharm IDE also helps saving time by not having to type, instead I can choose the value from a dropdown list. Following this I have executed the SQL database initialization file. I have also refactored the class responsible to calculate the buy signal into a separate module to make the project more organized and maintainable, calling it the `CalculateSignal` class. I have imported the getter methods from the `DatabaseAccess` class into it and replaced the hardcoded values from values from the database. I have executed the main script, and it successfully calculated the same results, without errors as before having made the changes.

To finalize this process the next step was to create a settings page where I could update the values for the constants, which I have added to my project backlog. This in turn led me to consider the optimal way to display the data generated by the application algorithm to the user who receives the notification. One option was to use an HTML page, which would be convenient, since I have already built the application with the Flask framework which provides an easy-to-use templating engine, and the setting page could be an HTML page as well. Other options would be to generate a PDF or send the raw data in a JSON or a CSV format. The most cost and time-efficient method seemed to be using the templating system to generate HTML files. This raised three questions:

1. How should the user be notified?
2. What is the best way to distribute the generated template files?
3. Which pages can the user access and how to set up the navigation between those pages?

I thought about many different options about notifying the users. The most obvious one that I had originally planned was via email. I have done some research and found that sending emails from the Raspberry Pi would be cumbersome and hard to set up for the following reasons:

- It needs a lot of steps to set up, including security updates, DNS configuration and monitoring.
- Residential IP addresses could be marked as spam messages.
- Some ISP providers block outbound SMTP connections.
- It would provide another attack vector for bad actors such as hackers.

I have found a common recommendation of using Google's Gmail service to send emails, and there is a popular library called *Yagmail* (<https://pypi.org/project/yagmail/>), which is a Gmail/SMTP client to simplify sending emails via Google. I have registered a new Gmail email address, however following that I had difficulties configuring the Google Account to add an authorized application that can send emails. I have successfully set it up after a while,

nevertheless when I have tried to send multiple emails, I have been informed that my account has been suspended for not complying with the terms and conditions of usage. I have read the terms and conditions, but I couldn't find a relevant passage that I have violated. I have submitted a request to reinstate my account, but it doesn't seem like a straightforward option, and it takes days to review such a request. Therefore, I have searched for alternatives, and found a company called Mailgun (<https://www.mailgun.com/>) which offers free email sending services up to 5000 emails a month. I have registered and set up the Python code based on the documentation, but after multiple tries, I didn't succeed in sending and receiving emails.

I started contemplating alternatives for notification. One option that came to my mind is to create a Progressive Web Application (PWA), that is a web application that can be installed like a native application on a smartphone or tablet, and it can access features like alert messages for the user. In theory I could set up a PWA and register an alert tied to the generated signal. However, this seemed like a time-consuming process, albeit a very good one, since it would allow me to create an application that can be installed on my phone without having to go through the application and vetting process of a particular app store and this method would be taking advantage of native web technologies that work across multiple platforms. After careful consideration I have discarded this option, because of the time constraints, my lack of knowledge in this technology and because it would add unnecessary complexity and time to completion to the MVP. Despite this I must acknowledge and remember that this could be a possible future improvement or path to the project.

After exploring various options, I have decided against directly opening a port on my server due to security concerns. After considering multiple options I have concluded that for my case I could combine GitHub Pages for the display of a static website and the Slack messaging application as a notification service. The reasons were the following:

- I am leveraging existing tools in the project, because I am already using GitHub for project management, and versioning and I have also Slack installed on my phone and desktop.

- These tools are used by a larger number of users, are well tested, well documented and have developer APIs.
- It is easy for any non-technical user to subscribe to the notifications just simply by joining the channel where the notification is set up on Slack.

Following this deliberation I have created a workspace on Slack called Investment Opportunities and created a channel called `#alerts`. Based on the documentation I have sent a test message to this channel, which was very convenient and worked immediately, in contrast to the difficulties of setting up email notifications with Mailgun.

At this point I wanted to test the site deployment on GitHub pages as well, since I have already generated a static HTML with the application. For this to work I had to generate an *index.html* file at the root of the project and change some settings on my GitHub account. Following this I have deployed the site on the URL <https://tonyettinger.github.io/automated-equity-valuation-full-stack/>.

2.4.4 Sunday 18 February 2024: Weekly reflection

This week I have finalized the main signal producing algorithm, deployed the application to the server, set up an SQL database and a related settings page, devised a notification method and a deployment mechanism to display the generated signal results as a static HTML file. The next steps are becoming clearer, but at the same time the number of the new tasks are increasing with each new feature. The next two big topics to address would be the following:

1. The pages to display the data and interact with the database. Currently I envision having a display page, where the user can see the buy signal recommendation with the company details; a main page, where the user can navigate to the other pages; and a settings page that could be used to change variables used in the signal generating algorithm calculations.

2. The scheduling function, that would periodically query the data automatically and push the new static HTML files to GitHub, triggering a new deployment and following that notifying the user via the Slack messaging application.

For me the personal takeaway from the week was dealing with unforeseen errors, mistakes and obstacles. I have made some calculation mistakes, that is partially due to the speed of development and lack of testing, but on the other hand I also don't have reference values that I could use to test against, because the calculation algorithm is customized, therefore I cannot verify my numbers against a trusted source reference value. Among the obstacles, the difficulty of sending emails was an unexpected one and it highlighted that relying on third party providers can cause dependencies and issues. Google has recognized later that I have not violated the Terms and Conditions, but I had no access to my email for nearly two days and I couldn't successfully use the Mailgun based on their documentation either. As I have encountered these difficulties, I had to come up with different solutions, such as relying on the features of the Slack messaging app. I believe it is more user friendly and already includes the capability to access the notification system of a smartphone, its capable of displaying messages and various users can join the notification stream with ease. It is also a third-party dependency, but one that comes with a working API, good documentation and it's free to use. Taking advantage of already developed features such as Slack's notification system and API, and GitHub's integrated static site deployment functionality greatly reduces the development time compared to custom made solutions. This means I didn't need to register mail servers, set up domains and DNS for the site deployment and think about how to trigger the deployment when there is a new page generated. In the case of GitHub Pages, pushing the changed files automatically triggers a deployment. As follow up tasks I have noted to myself that I must develop a settings page, where I can modify the database values of variables used in the calculations. I plan to do this in a way that the settings page would be only accessible via the VNC connection on the server, to reduce security risks. The static page deployment setup allows me to secure my application, by separating the things that need to be exposed such as the signal page view that would be visible on the web, and the settings page that would be only

visible and usable on the server. At the end of the 4th week the Kanban board looked as on Figure 8.

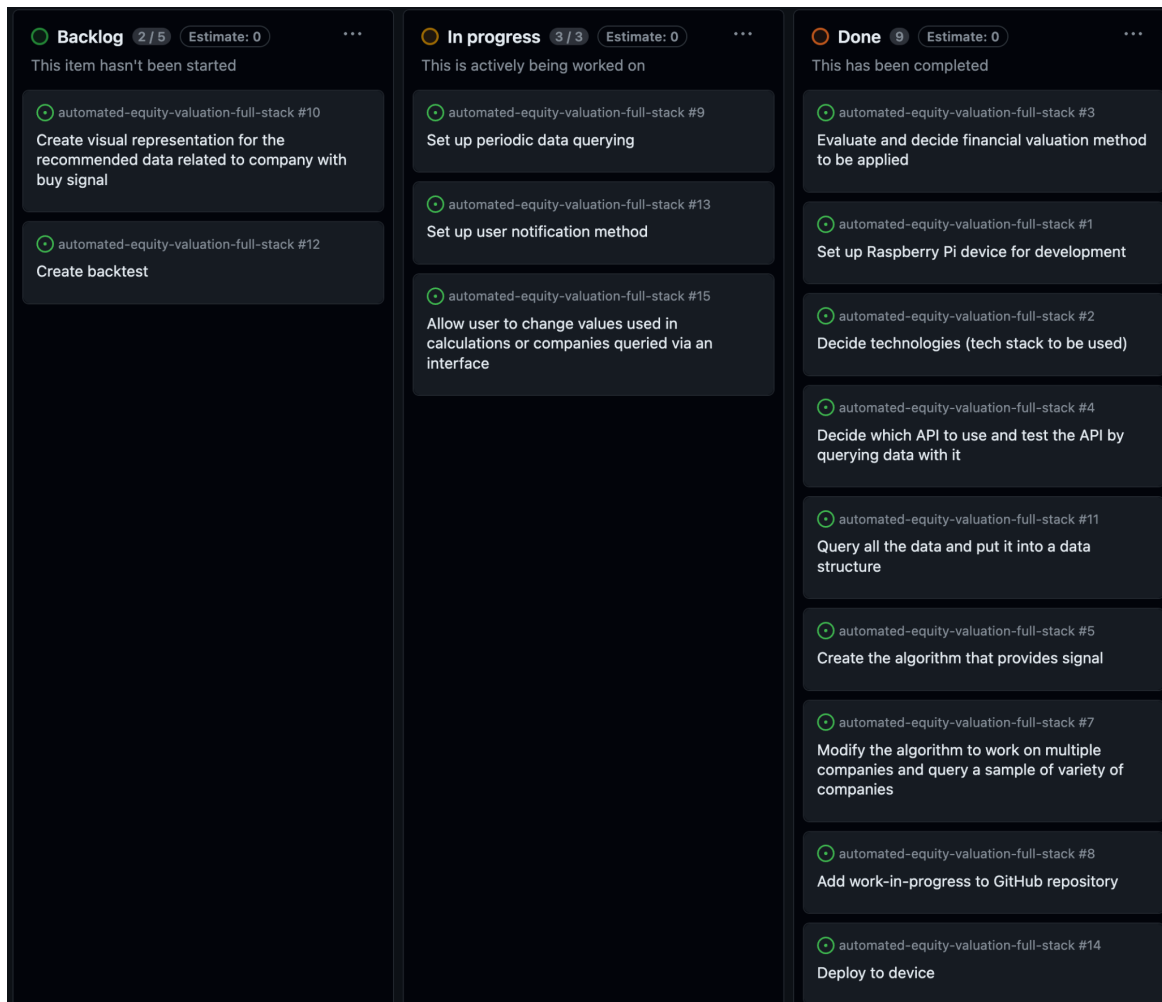


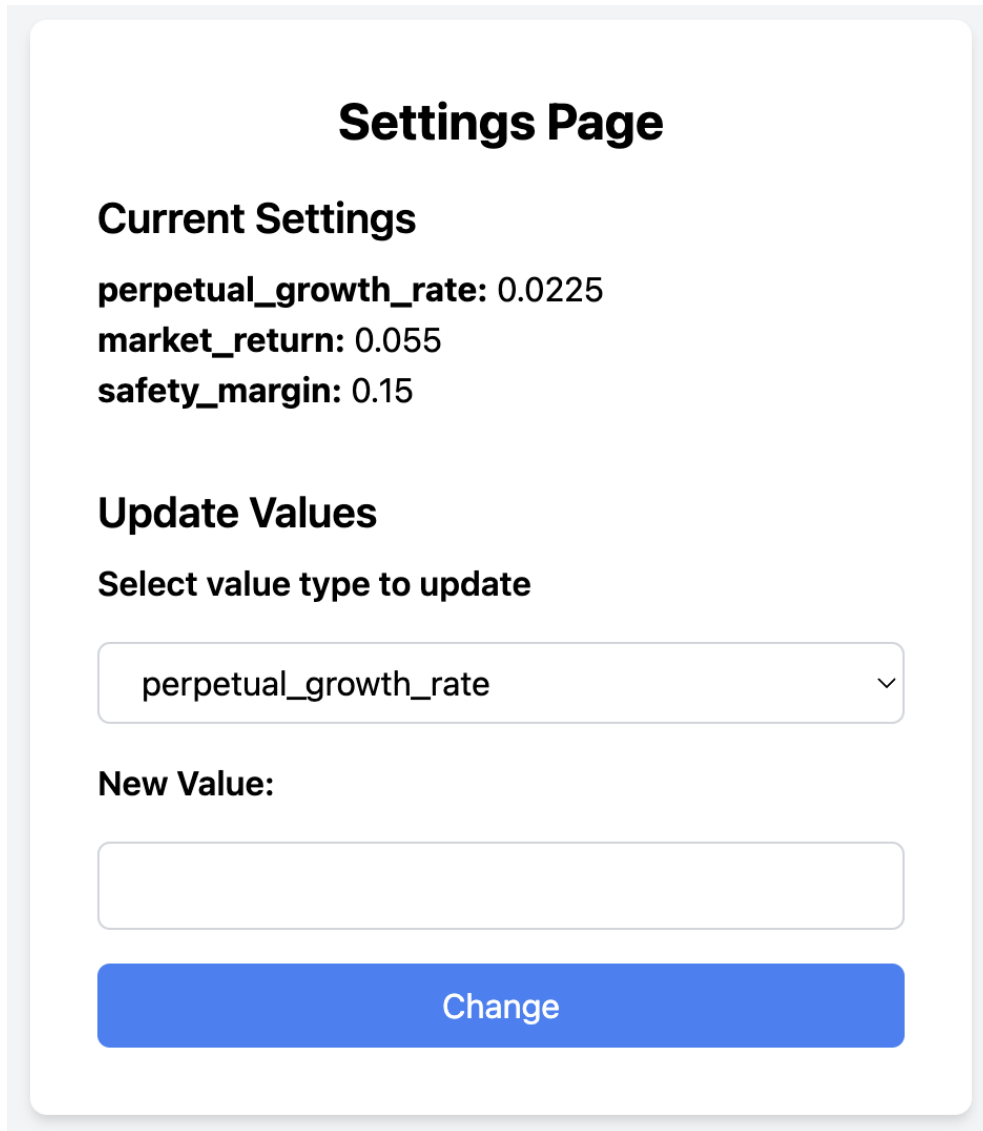
Figure 8. The status of the Kanban board at the end of the 4th week.

2.5 Week 5

2.5.1 Tuesday 20 February 2024: Settings page development

To enable changing variables stored in the database, I have created a settings page HTML template and added a `/settings`` route in the main `app.py` file. Then I have developed some helper methods to create a drop-down form selector that displays all the available variable

names. I have included this in the template as part of a form element, and added an input field that can be used to modify the selected element from the dropdown as shown on Figure 9.



Settings Page

Current Settings

perpetual_growth_rate: 0.0225
market_return: 0.055
safety_margin: 0.15

Update Values

Select value type to update

perpetual_growth_rate

New Value:

Change

Figure 9. The final settings page UI - accessible from the server only.

This initial unstyled state of this settings form led me to add to the Kanban board a task to find a way to style the application web pages with a certain framework or library. Bootstrap and Tailwind CSS came to mind, but it must be investigated which one is a better choice using Python Flask.

2.5.2 Thursday 22 February 2024: Navigation, scheduler and automatic change commits

I have tested the settings page, it works well, the values are changing. I have made some changes to the code of the form so when the user changes a value they get redirected back to the same settings page, therefore the change is visible immediately. The next task was to add a periodic querying functionality that would run the signal generating algorithm. For that I have refactored the existing code, in a way that the algorithm runs on an endpoint on the relative URL route of `/check-stocks``. I have changed the root URL page to serve as a homepage and collected the date-stamped generated URLs to be displayed as a list of links on it.

At this stage I had three URLs: the root page, settings page and an endpoint to generate the signal and the static HTML files subsequently. Following this I have researched and found the easiest way to schedule running a recurring function periodically is with a package called *schedule* (<https://schedule.readthedocs.io/>). It makes it easy to set up a time zone and recurring frequency. I have added it to the project dependencies and tested it with a shorter time frame and it worked well, running a given function minute by minute. After this I was more confident to set up the trigger time to be one minute after the market closing time in the US, which is New York time 16:01.

After the algorithm processing the results from the Alpha Vantage API query and generating the files such as the updated homepage and signals page, I had to automatically commit the generated files to GitHub to trigger a deployment. For this purpose I have found the easiest way is another package called *sh* (<https://pypi.org/project/sh/>), which in effect allows to run command line commands, that in turn enable committing and pushing git changes to the GitHub repository. I have tested the implementation, which resulted in an accidental committing of the `.env` file which contains the secret key that shouldn't be part of the repository. I spent some time researching how to rewrite the git history and remove traces of the secret key from the remote repository which could compromise the secret key by exposing it publicly and have managed to remove it.

The last step after running the signal generating algorithm and committing the changes triggering a deployment is to inform the user via the notification system. I have added a notification in its module and tested it again and it worked well. At this point I had three new modules: a GitHub, a notification and a scheduler module in the codebase.

2.5.3 Friday 23 February 2024: Routing issue resolutions for final deployment

The next task I had in hand is to ensure that I can generate the static files for the deployment, but at the same time I also wanted to be able to display the HTML template on my local development environment. The issue was that for the static file deployment I had to use different routing than on the local development environment. This was because during the GitHub static site deployment the URL of the folders is automatically prefixed by the name of the project, and I didn't find a way to stop this from happening. I needed to ensure that when I visit a certain URL on my local development machine it would behave the same way when navigating to that URL on the deployed static site. The links, the CSS stylesheet files must all behave consistently in the two different environments. The situation was made even more difficult by the fact that GitHub pages doesn't offer an out of the box solution to select which folder to deploy, it takes the `index.html` file in the root folder and while the flask application has a separate folder `/static` for the files generated based on the Python template files. I needed was to deploy the static folder therefore I have created a redirect directive in the `index.html` file in the root folder to the `index.html` file in the static folder. In the Flask routing related code, I have added a new variable to the templates called `production`, in which a Boolean value can indicate if the template is used in production or in the local development environment. With this new variable I was able to display a different HTML template on the local environment and the deployed production version. In these files the style file and link paths are modified depending on which environment the user is opening them. I have carried out one more adjustment, so the signals and aggregated data generated were saved into a JSON file, and the HTML templates use the JSON file therefore I could refresh a page containing the signal template page without querying the data every time which could take minutes. It was necessary to be able to test local changes in the template quickly while developing the template styles for example. At this point I had a list of URLs: `/signals`, `/settings`, `/check-`

stocks` and a root homepage with a list of the generated signal files. The `/check-stocks` is the API endpoint that generates the new date stamped signal files that contain the stocks that reach the criteria based on the DCF calculation. The settings page URL is based on the template that can modify the database values for constants used in the calculation, namely the estimated perpetual growth rate, the market return rate and the desired safety margin that is the difference between the DCF-based valuation calculated by the algorithm and the current market value of the stock. The settings route can only be reached on the server. The signals page is a development page for the signal template file, which loads the latest queried data into the template from the JSON file and it could be utilized for developing the UX/UI of this page. The signals page is the most important part of the application, because it is intended to contain all the relevant data of the selected stocks which is the main source of value provided to the user. I have run a few rounds of signal generations, pushing the generated files on GitHub triggering deployments to test everything is working well. Following that I have tested the routes on local and on the deployed sites. The style files were loaded correctly, and the links worked on both environments, everything worked seamlessly.

2.5.4 Sunday 25 February 2024: Weekly reflection

This week I have developed the routes used in the web application, without any styling. This provides the basis of interacting with the application. The routing is working on the deployed static site as well on the local development environment and includes all the key views. Several key functionalities have been added, utilizing existing Python packages for speeding up and simplifying development. These include interacting with the command line and committing changed files to GitHub and the scheduling and the running of the signal generation algorithm periodically at a certain time of the day. These efforts were accompanied by running the signal generation repeatedly, investigating the results, fixing bugs and refactoring code continuously to keep it clean and modular. During these reruns of the signal generation with different stocks pointed out bugs that had to be investigated and corrected. These were mainly due to extreme values or unexpected values and the handling of those values, and the assumptions included in the calculations related. For example, if the estimated revenue is always negative and the WACC value is also negative, this would result in a positive valuation, because negative times

negative results in a positive value, while in fact the stock should not be on the recommended list. This has been resolved by returning early when a value is out of range and eliminating the stock in question from the list of stocks to be considered. The main and most demanding task became at this point to include all the relevant data in the signal page and present it in a way that is easy to overview and analyze for the user, providing a clean and useful user interface. This requires CSS-based styling, organizing the data in a visually appealing way and potentially creating graphs that would visually represent aspects of the time series data queried from Alpha Vantage. At the end of the 4th week the Kanban board looked as on Figure 10.

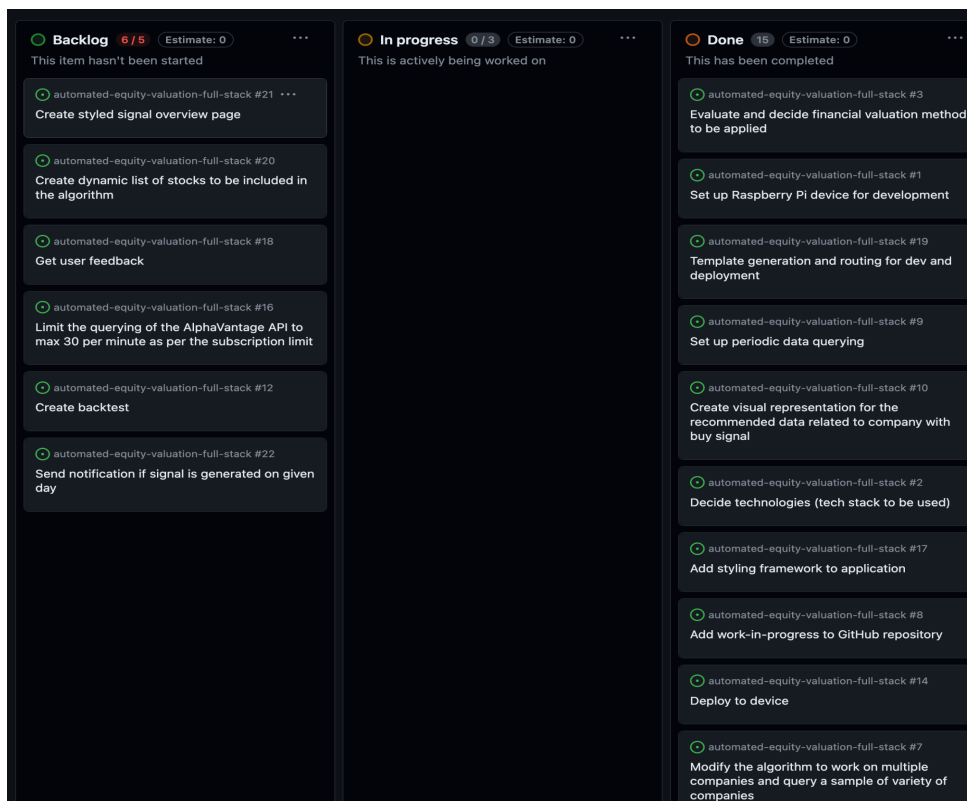


Figure 10. The status of the Kanban board at the end of week 5.

2.6 Week 6

2.6.1 Monday 26 February 2024: Signal page styling, web crawler for dynamic stocks

The most important view of the application is the one where the generated signals can be viewed in addition with any relevant data that would help the user to analyze the given stock. This data shouldn't be overwhelming, it should be compact enough to fit on a screen per company for a convenient visual overview. I have imported and set up the CSS library Tailwind to make the styling process faster and easier. It uses atomic classes for styling and has a convenient documentation. The original plan was to include two containers, one detailing the results of the signal generation which is based on the valuation of the application algorithm and a second one including facts and data queried from Alpha Vantage that could enhance the decision-making process. In addition, I wanted to include a chart about the Moving Average Convergence Divergence (MACD), which is a technical indicator used by investors to detect trends and changes in strength, direction and momentum in a stock's price, based on plotting two moving average lines of different period lengths. This in effect can visually indicate if the stock's short term average price drops below a longer-term trend, which could be a positive indicator. The last piece of information I wanted to display was a list of selected news articles, because Alpha Vantage provides a very convenient way to query news for a stock ticker including sentiment analysis of the article pieces and I consider this as a useful addition for the user who wants to further investigate by reading relevant articles.

For the chart drawing implementation, after some research I have found a free and well documented JavaScript based solution called ChartJS. The chart could be generated in the signal page template based on JSON data, utilizing this library. Since the data returned by the API includes many years of MACD chart data I had to select the last 180 days to be relevant and I had to sort the data. I also had to make sure each chart has a unique ID related to the ticker symbol so there is no collision with JavaScript variables and the ChartJS can execute properly.

Then I have started styling the organized HTML data with CSS Flexbox, and after a few attempts to make it responsive and visually appealing I have switched to CSS Grid and thought it might

be a better option. However, I was struggling to find a right balance, to keep the page simple and the elements aligned in a way that they have an easy overview on different screen sizes. To take advantage of new and emerging AI-based tools I have prompted the presently popular free-to-use AI system called ChatGPT (<https://chat.openai.com/>), to update my Tailwind CSS-based classes and the Document Object Model (DOM) structure in a way that would result in a clean design. After this I have modified the resulting output provided by ChatGPT, enhancing some features and changing some others such as border rounding values and colors. One of my main objectives was to display all signals associated with each company on a single screen. However, incorporating news articles as a list would have made each company's block excessively long. I had an idea to resolve this by adding the news articles as a dropdown menu, therefore this news related information would only take one line on the screen. Once more I have prompted ChatGPT to help me to convert the article list into a dropdown menu, where when the user clicks on an article on the list it would open in a new tab. When I got back the result it was working seamlessly, the articles were listed, and I was able to click on them and this initiated opening a new tab with the given article URL. I have adjusted the label for this dropdown and added the article's sentiment indicator next to a given article in the dropdown list. On the top of the page, I have added an explanation of these sentiment levels. The result of styling the DCF analysis and related data at this stage was as shown on Figure 11.

Summary and analysis of selected stocks based on DCF signal

[Go Back](#)

Sentiment:

$x \leq -0.35$: Bearish; $-0.35 < x \leq -0.15$: Somewhat-Bearish; $-0.15 < x < 0.15$: Neutral; $0.15 \leq x < 0.35$: Somewhat_Bullish; $x \geq 0.35$: Bullish

DCF Analysis for MTCH (Value in Billions)

DCF: 12.23
Market Capitalization: 9.09
Difference between Market Capitalization and DCF: 3.15
Percentage difference between DCF and Market Capitalization: 34.63%
Latest queried price: 33.9
DCF price: 45.64
[Link on Yahoo Finance](#)

Additional Data for MTCH

Company description: Match Group, Inc. offers dating products globally. The company is headquartered in Dallas, Texas.
52 week high: 49.24
52 week low: 27.85
Analyst target price: 44.03
PE ratio: 15
Forward PE: 12.11
Profit margin: 0.194
Price to sales ratio (TTM): 2.7
Price to book ratio: 10.97

News (overall news sentiment score: 0.07)

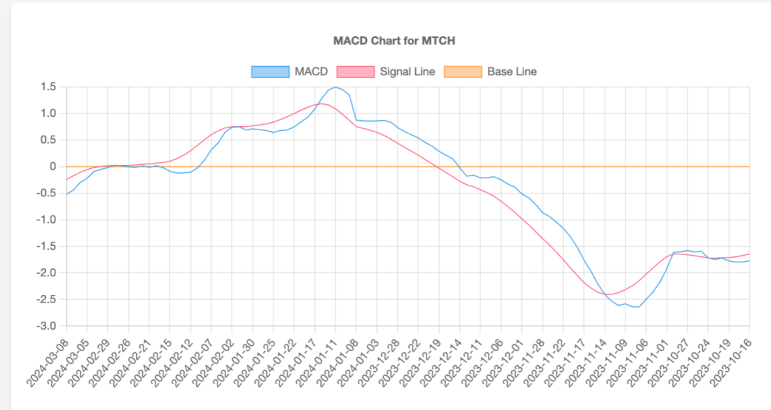


Figure 11. Example of a generated signal overview.

Reviewing the generated summaries of signals, I have noticed that my stock valuations were not too far off from the analyst target prices for comparable established tech stocks such as PayPal, and Match Group, a dating application. This makes sense to me, since these companies can grow their user base and income without significantly increasing their costs, since they are digital products that don't require massive capital investment for expansion. Therefore, the amount of cash compared to the operating costs is more significant and a better indicator of a company's value in the long-term. Based-on this realization I had the idea to query a dynamic list of market moving tech stocks for the analysis. This would include stocks that have increased or decreased significantly during the previous trading day. I have found a page on Yahoo Finance <https://finance.yahoo.com/u/yahoo-finance/watchlists/tech-stocks-that-move-the-market/> with such stocks. I have investigated the HTML code and noticed that the tickers could be selected easily based on their `data-testid` attribute. I have searched for a web crawler

solution and found that the Python-based ones are more suitable for my purpose than Playwright for example, because I can easily include a Python module into my existing project instead of using a JavaScript-based crawling solution that I have originally had in mind. The library I have found most suitable for the task is called *BeautifulSoup* (<https://pypi.org/project/beautifulsoup4/>). It has a large user base, and it is well documented. I have added it to my application requirement list, installed it and created a function to get the market moving stocks from the Yahoo web page. It successfully queried the list of stocks from the web page and created an array including the stock ticker names.

At this stage I have also decided to slightly adjust some of my fixed variables such as the assumed market return rates and perpetual growth rates in the settings so I could calibrate the value estimates based on the analyst price estimates for established internet-based companies. Since my calculations have no baseline or verified standard that I could compare it to or test against, it seemed like a reasonable way to calibrate my algorithm to some extent by anchoring the valuation to analyst estimates of a particular kind of stock where the DCF calculation is a good value estimate. This means internet-based, not capital intensive and not early-stage tech companies. I attempted to develop backtests to accomplish that goal from my Kanban board. I have checked the API capabilities to see if it would be feasible to easily query past stock data, but the API interface doesn't allow for this kind of queries, therefore I have abandoned the creation of backtests in this timeframe. At least it appeared to me that my DCF-based estimates are in-line with analyst estimates for the stocks that I focus my analysis on, and this gave some level of confidence in the usefulness of my valuation method.

2.6.2 Tuesday 28 February 2024: Deployment, listing page style and pagination.

The main objective for this day was to complete the final deployment of the application to the Raspberry Pi device and to run the application in production mode together with the scheduler script. I have pushed the latest changes from the main development MacBook Pro machine and pulled these changes via GitHub on the server device. I had to update all the Python dependencies in the requirements.txt file and install them on the server. The other related task was to choose a production server solution for the Flask application, for which I have selected

Waitress, considering that it was recommended by the official documentation of Flask (<https://flask.palletsprojects.com/en/2.3.x/deploying/waitress/>). I have successfully started the server and run a few tests runs of the scheduler to test that the static files are being generated and changes are published to GitHub, which in turn triggers a deployment by committing and pushing to the GitHub repository the contents of the generated files in the static folder.

Following this I wanted to make sure that the overview page looks professional and easy to navigate even when there are many signal files for multiple days. To achieve this, I planned to order the signal pages by date and paginate them, to have 10 items displayed on a single page and to have buttons to navigate between the pages. Once more I have turned to ChatGPT to assist in creating and styling this page, because I know that pagination is a common and traditionally time-consuming task in web development, and I didn't want to re-invent it. Instead, I have took advantage of the capabilities of ChatGPT which has been trained on data that contains answers to such common tasks. The result of the prompt was a simple clean design, with working JavaScript logic included that allows the page to show 10 pages at a time. I wouldn't consider this production ready for a live application mainly because of the simplicity of the UI, but for the purpose of this MVP application I considered it adequate.

2.6.3 Wednesday 29 February 2024: User feedback

At this point the application was deployed, had an adequate user interface and was running daily periodical queries automatically deploying the generated pages on GitHub. I have asked for feedback from a friend of mine who is an active investor and has experience investing personal funds. The main feedback that I have received is that the abbreviations were unclear, and it was not obvious for someone who is not familiar with DCF valuation, what does it mean and what is the recommended action. Therefore, I have decided to expand the explanations, instead of writing "DCF valuation" I have changed the text to "Theoretical share price based on DCF valuation" and I have changed the abbreviation of "DCF" on the first row to a more descriptive "Discounted Cash Flow based valuation". These changes are shown on Figure 12.

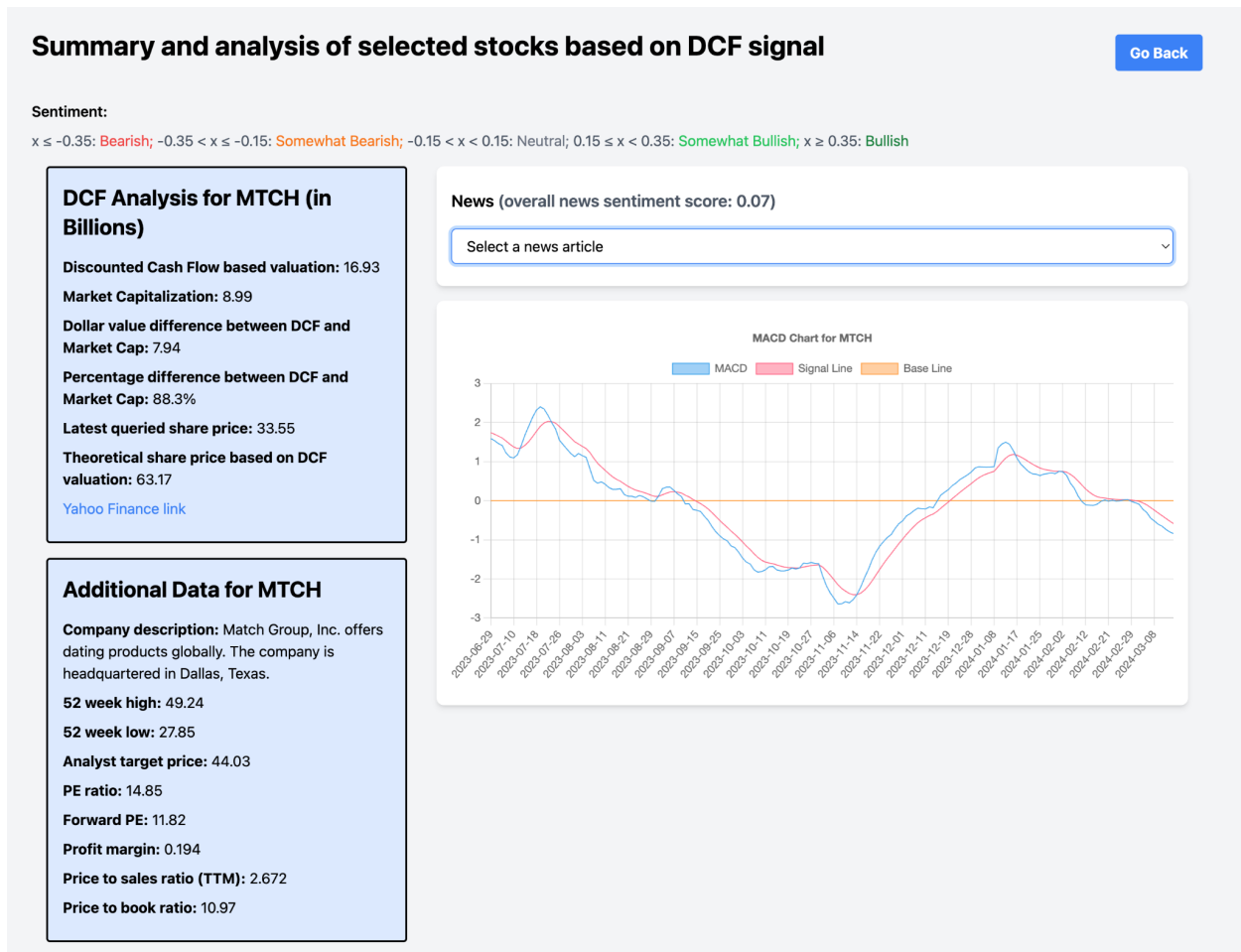


Figure 12. Updated explanation text of the signal page.

2.6.4 Friday 29 February 2024: Rate limiting and updating notification messages.

As a last step I wanted to add some additional functions to be able to select more stocks, not only the market moving tech stocks, but also the overall biggest losers in the market for example. This would provide a larger pool of stocks to analyze and a higher chance of encountering a buy signal. The problem with this was that my subscription for Alpha Vantage only allows 30 API queries per minute and adding more stocks would produce an API limit error. The higher tier subscription plan that would allow more queries is significantly more expensive. Due to this reason, I wanted to implement a function that would allow for limiting the calls of the data aggregation step of querying to maximum 30 per minute. After some research I have found a solution using a library called *ratelimit* (<https://pypi.org/project/ratelimit/>), and created a function that allows less than 30 calls in a

minute and then delays the running of the next cycle by putting the function to sleep until the next minute. I have added this function to the top of the querying functions in the data aggregation module. Subsequently, I have added the stocks from the “Day Losers” page of Yahoo Finance from the URL <https://finance.yahoo.com/losers/> with a new web crawling function that collects an array of stocks from this page to the list of stocks to be analyzed. I have run the signal generation algorithm including both the previous and new sets of stocks and it ran successfully. I have noticed some stocks had missing values and I did some refactoring of the code that would allow the removal of which are missing some critical data that is required for the DCF calculation. Following the implementation of these changes there were no failures due to exceeding query limits that I have experienced before when trying to query too many stocks.

I have also received feedback regarding the availability of the application on GitHub pages. It seems that I was logged out from GitHub automatically, and it seems to have resulted in a 404 missing page error when someone attempted to access the page. Considering this, I have decided to also deploy the static site in a different way, and I have chosen Netlify to add another automated static site deployment. With a few easy steps to set it up I have managed to also deploy the applications web pages to <https://automated-equity-valuation.netlify.app/>. This way whenever the GitHub repository has been changed the application would be re-deployed on both GitHub Pages and on Netlify, which also provides a free deployment service. I have added this page to the generated signal alert message on the Slack application. There was one more adjustment I wanted to make, which is to ensure that if there is no signal generated there is no notification for empty dictionaries. I have added this check to the code, and at the same time I have passed the stock tickers from the signal generation function to the Slack notification message, which meant that now the user would be already informed in the Slack messaging application about which stock tickers are fitting the algorithm criteria as shown on Figure 13.

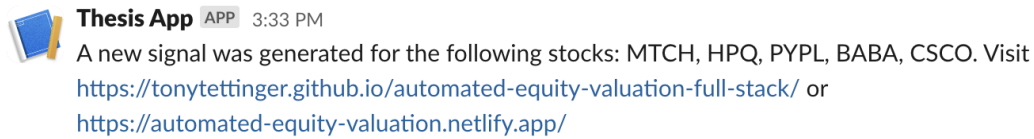


Figure 13. Message on Slack alerting the user of potential investment candidate stock tickers based on the DCF analysis.

2.6.5 Sunday 31 February 2024: Weekly reflection

This week I have deployed the version of the MVP application that fits the criteria and goals set on my Kanban board. It has the styled overview and signal pages, where each stock signal is viewable on a single page, including a DCF analysis, additional information such as analyst estimates, financial ratios among others. There is also a news dropdown selector with links included that lead to relevant articles and a MACD chart to provide some additional insight into the stock's price movement momentum. These pages are automatically generated and published by a scheduling application that is responsible for calling endpoints which result in the deployment and notification of the user via the Slack messaging application. With this I consider the MVP application development to have reached the set-out milestone, and although there are possible improvements, adjustments and features that could be added I consider this full stack application development project complete, with most of the items from the Kanban board backlog completed. A few items have been canceled due to feasibility issues such as the back testing and at this point there are no more items in progress and the backlog is empty. The MVP product is deployed on a Raspberry Pi server and is remotely accessible. This week highlighted the importance of getting early user feedback and thanks to my exposure to different available solutions in the web application development field I was able to create an alternative deployment process to be less dependent on one provider. This week was focused on details and adjustment instead of major new feature development, which was expected at this final stage of development.

3 Conclusions and discussion

3.1 Summary of diary reporting period

The time bound development of a full stack application provided many valuable insights into the possibilities and limitations of working on an MVP project with a Kanban style project management. The result of this thesis project is an MVP that includes features that in the opinion of the developer have the potential to provide value to its prospective users. At this point it is in a readiness stage where the application contains a sufficient set of value-adding features with an adequate UX and UI that it could be used to present to users and initiate a conversation, that in turn could potentially result in the feedback and improvement loop to deliver a unique and valuable software product to its future users.

3.2 Answers to research questions

3.2.1 What are the learnings from the iterative planning and execution of a time-constrained project that has a goal of developing a full-stack MVP application?

With limited time and resources, it became crucial to prioritize features based on their importance and impact. Identifying and focusing on core functionalities were crucial due to the constraints. A background knowledge of the project domain, experience with technologies from the backend and frontend development fields and the ability to actively discover and curate different solutions were equally important for the successful completion of the project. These skills and intuitions that have been acquired working in software development for several years and they were indispensable to be able to move quickly, make informed decisions and to reason about them. Moreover, pivoting away from the starting ideas and cutting down features was important to keep the project inside the time boundary. In this aspect the Kanban system was beneficial, because it limits the work-in-progress and backlog tasks which allowed it to focus on the truly necessary features and to eliminate most non-critical functionalities. Focusing the attention with self-imposed enabling constraints by the Kanban system made it possible to focus on the important parts, cutting down the time that would be required to develop additional features, ones that would have put the project timeline in danger. Choosing

a paid API over the originally planned Yahoo Finance free API was a big advantage in project delivery, because the quality of the documentation, feature richness and ease-of-use sped up the development significantly and provided a lot of options that would have been unavailable and unreliable with the original API options. The other factor that enabled project delivery on time was the strategic use of third-party solutions and libraries. Choosing technologies, with great documentation, large user base and readily available solutions that can be integrated into the project was crucial to this. This validated the point of choosing such base programming languages that are popular and well established, because building upon these fundamentals made the next steps easier. Lastly, taking advantage of tools that assist the development such as professional code editors and LLMs allowed for faster prototyping and to draw-up solutions without having to code them from scratch. In summary, reflecting on the project development: domain knowledge, experience, right choice of tools and libraries and enabling constraints were key factors in delivering the MVP.

3.2.2 What methodologies and tools allow for efficient iterative software development?

From a business perspective producing a usable and value adding product with quick iteration cycles is desirable. For this reason, in most cases the tools and methodologies are project-dependent: depending on the expertise of the team, the scale, the type of software to be delivered, functional and quality requirements and other factors. The Kanban method seemed like a good guiding methodology for setting up and starting the project, but occasionally the speed of development and cascading ideas created situations where features and changes were implemented without adding them to the board. This happened because it felt that it would have added extra friction between the idea and implementation and that seemed it would slow the process down while breaking the flow of thoughts and development itself. Therefore, I believe the process should be only there to help the developer, not to hinder the progress. The chosen programming language of Python and JavaScript provided a lot of benefits for iterative development, such as allowing to use a Python specific IDE, that helps to automatically add project dependencies, reformatting files, refactoring, auto-complete suggestions, project setup, integration with git. This was one of the driving factors that allowed the project to be completed on time. This was very apparent, when occasionally editing files on

the Raspberry Pi device and the built-in vi editor usage was error-prone and slow without additional setup and practice. The other benefit of choosing a popular high-level language was the ability to tap into open-source libraries that were available due to the large number of users and use cases. These libraries simplified and sped up development significantly and thanks to these a lot of implementation details and complexity remains hidden. This is a positive value attribute from a business perspective, but at the same time there is a sense of tradeoff using third-party solutions regarding creating unknown dependency trees and potential security gaps. In the case of my application the resulting signal page is a static page, that is detached from the server and the only way to access the database is on the device itself, therefore the security aspects were not a major concern from these packages. Therefore, my conclusion is that the best tools for fast paced iterative MVP development are ones that have an ecosystem of related documentation, libraries and frameworks and a large community of developers with a considerable knowledge base. But this involves the caveat that the tools and methodologies are project and team specific and in retrospect this research question was too broad. Just as financial analysis employs various methods beyond DCF to value financial instruments, employing Python and JavaScript with the Kanban method, while emphasizing MVP delivery, represents one approach among many for conducting a full-stack software engineering project. There are many possible combinations of tools and methodologies, in an environment where new solutions keep emerging continuously and it is dependent on the team, project criteria, cost considerations to decide which tools are ideal for a given project.

3.2.3 What are the learnings from this project that can enhance professional software development skills?

Having a wide range of knowledge of the possible solutions in the various fields such as deployment with GitHub Pages and Netlify, development frameworks and libraries such as Flask and ChartJS were essential to maintain the pace of the thesis project. It is my opinion following this project experience that being aware of the landscape of available best practices and solutions is of great importance and in my opinion might be even more important than core computer science skills to some extent. Nevertheless, from the computer science perspective the class-based encapsulation of logic, modularity, structuring the project into

logical units and having a folder structure that reflects this logic was contributing to the ability to add new features, debug existing issues. I have deliberately chosen a few frameworks and tools that I am less experienced with to be able to experience the difficulties that could result from this, such as the Raspberry Pi device, Flask and using Python as the core language since I am mainly working with JavaScript-based technologies. Because of this I had to learn language-specific technicalities, such as how to replace JavaScript array methods with list comprehension methods in Python, to understand decorator functions, lambda functions, dealing with undefined keys in Python dictionaries and how to pass data from the Flask application to the JavaScript functions in the web application. Nevertheless, these didn't prove to be significant obstacles, because of being familiar with generic programming principles such as encapsulation, object-oriented programming, software architecture considerations, which apply across different languages and projects. Another learning is that most common problems have already optimized solutions for them, either in the form of existing code, libraries or frameworks. An example for this is the collections module (<https://docs.python.org/3/library/collections.html>), which implements specialized container data types, which helped me to deal with missing value errors when creating new dictionaries instead of writing custom code. Machine learning based applications have already been trained on the existing solutions and AI tools such as the large language model (LLM) based ChatGPT are an efficient way to tap into essentially crowd-sourced knowledge. For instance, implementing pagination for the links may seem relatively complex at first, but on the other hand is a very common requirement and therefore ChatGPT offers effective solutions for solving this generic problem based on some simple prompts. After the solution is sketched out it can be updated by the discretion of the developer to include extra features or a customized styling for example.

The main takeaway in terms of learnings about development best practices is that it is important to learn about and keep implementing clean and logical software design based on software architecture and development principles centered around simplicity and modularity. This results in faster development cycles, ease of debugging and ease of adding new features. Being aware of the current technological landscape for best practices and off the shelf solutions

is another valuable skill, therefore having up-to-date knowledge can save a lot of development time and allow the developer to choose the best solutions for the given project.

3.2.4 What continuous improvements could be implemented in the initially deployed MVP?

Continuous refactoring to make sure that functions are reusable, modular, well organized, have descriptive names and ideally only perform a single task is something that I have been striving to achieve through the project. This allows the development team in the future to expand the functionalities instead of having to excessively rewrite existing modules. However, there is an important development best practice that is notably missing from this MVP project due to time-constraint, which is test-based development. Having proper unit tests would have made the code more robust and changes could have been made in the codebase with a higher level of confidence. I think it would be essential to start writing tests for the existing modules and this would be the first step that I would recommend implementing following the initial MVP development period. That would be an important update to the project that could be implemented and would pave the path for any further improvement efforts.

3.2.5 What are the learnings and conclusions that can be drawn based on the continuous self-reflective decision-making process?

Being able to pivot in terms of process, deliverables and tools is a quality that shouldn't be underestimated. As an example, if I would have encountered an issue to gather the required data for the DCF calculation via an API, probably the best course of action would have been to find a different valuation method or choose a simplified version of the valuation instead of spending a considerable amount of time trying to force to comply with the original calculation method. The same principle is true for third party services and technologies used. At any point in time the tech stack of the project or providers could be changed, based on a cost versus benefit analysis of the change. Changing from the free Yahoo Finance API to the well documented, detailed and reliable Alpha Vantage cost a certain amount of money but provided greater benefits for fast prototyping and getting data such as news, MACD data and sentiment analysis. The time saved for building an MVP was significant, and although in the long-term it might make sense to build crawlers to get the data from free sources, within the project

constraints this was a reasonable alternative that allowed for the project to be on time. Writing down what was accomplished and what are the next steps was a good way to keep track of the following tasks and a written record and documentation helped to reason better and oversee the progress being made during the development period. Often one idea leads to another, and things can be implemented, fixed or improved without going through the official process of the chosen task management. These discretionary decisions were documented in the reflection sessions, and without these reflections it could have been easy to overlook the importance and role of developing in a flow-state, focusing on delivering value for the users instead of rigidly trying to focus on an established project framework. The self-reflection was beneficial to think about the systems set up around the projects as helper tools that should serve the developer, instead of the developer being a servant of those tools for the sake of compliance.

3.2.6 What kind of new research paths and questions can we derive at the end of the development and reflective documentation of this thesis project?

An interesting question would be how this one person MVP project could be converted into an Agile methodology-based team effort? What steps need to be considered to scale up the project if there is a demand for it and what steps need to be taken to convert it into a scalable infrastructure and architecture and how to structure a team taking care of this project?

Another avenue of research could involve exploring ways to augment the existing MVP with additional features and capabilities. It is possible that the valuation method could be combined with other methods, such as a querying of the quarterly earnings reports and earnings calls transcripts and attaching an AI-based text analysis of these documents to the reports. In addition, the recommendations could be customized to the user's financial status and savings and provide summaries of investment results. It would also be an interesting research question to discover, what can we learn from the process of iterative discovery of new features based on continuous user feedback and what are the best ways to conduct such a discovery process?

3.2.7 How suitable is the chosen technological stack from a User Experience perspective?

I believe that the rigidity of the decision to not expose the server to the internet and having a static site deployment instead of a live web page server diminished the UX of the application. A

fully JavaScript framework-based solution would have allowed for better interactivity and features such as updates based on live data among other improvements. It appears that the templating system of Flask is good for the initial stage of development, but in retrospect I would have converted the Python to be fully headless backend API and the frontend to be a completely JavaScript-based web project that consumes the data provided by the backend. I believe the choice of Tailwind CSS allowed for a clean design where the report fits on one screen therefore it is easy to overview for the user, utilizing atomic classes it made the styling easy and maintainable. It can be argued that the choice for Slack to be the notification tool was validated, because it is easy to use, available on a wide range of platforms and the notification messages are informative and provide links to the deployed web pages. The time-based ordering of signal pages and pagination solution also increases the user experience value of the application. There are many features that could be improved, for example if the user could define stock tickers they are interested in and add it to the analysis algorithm would provide for a more personalized experience for the users. The tech stack chosen was suitable for the MVP and prototyping, however a separation of frontend and backend would be necessary and refactoring parts of the code to confine them to one of those fields would be beneficial. This would allow an interactive user experience by implementing user interfaces based on dynamic interactive frameworks and a faster development cycle for the user interface due to better separation of concerns.

3.3 Implications of findings

The following are my conclusions in terms of implications based on answering the research questions and conducting the project:

- Knowing the technological landscape, the currently available off-the-shelf solutions and tools reduces the unknown unknowns and enables developers to make informed robust decisions about the full stack project by enabling the assessment of the feasibility and cost versus benefit analysis of the various alternative solutions.
- Maintaining core programming principles of modularity, simplicity and clean code allows for faster development cycles of new features, easier debugging, reusability of

components and increased maintainability. These principles should be observed and considered from the beginning of the project.

- Being able to cut down the scope of the project to the most necessary features allows it to be delivered quickly and therefore the development could focus on user feedback instead of speculation about user needs.
- Pivoting is crucial in case a planned solution becomes infeasible for cost or other practical reasons. In addition, pivoting from the original goals or methods enables developers to find better solutions within the given time and resource constraints of the project.
- Project management, domain knowledge and software architecture could be as important and impactful than core programming skills, since there are many open source or paid solutions that are already optimized therefore it is not necessary to reimplement those programmatically. An example of this in the game development world is the Unreal Engine, which is a 3D video game engine that has been used in the development of numerous successful games since 1998.

3.4 Recommendations for further improvements and research based on results.

The possible improvements could be categorized either as software engineering type improvements or feature enhancements. In terms of software engineering, attaching a debugging tool and a logging service would be necessary additions for a production grade product. This would allow for efficient error tracking and debugging. The other area where there is an obvious gap in the current project is about testing. Adding unit tests would be the first step here, which would increase the confidence that the software is still working as expected when updating packages or adding features. Another type of testing that I consider important is called endpoint testing. The third-party API response types could change unexpectedly, which could result in unexpected behavior. Therefore, monitoring that the API response conforms to the expected response would be beneficial for maintaining the project in the long run. Furthermore, refactoring the Python code to serve as a headless backend service that would serve JSON responses for the frontend would be beneficial, because it would make the project flexible in terms of choosing a frontend framework that could be maintained with a

higher level of separation of concerns. It would allow for the development of a better user experience, due to the availability of many JavaScript based frontend libraries that facilitate the development of dynamic user-friendly frontend interfaces. This would also possibly lead to reimplementing the frontend with one of the presently popular JavaScript frameworks for interactive web applications such as React, Vue, Svelte or Angular among others. It is also a possible concern for maintainability that the current implementation doesn't have a static typing system. There are solutions for that in Python, while TypeScript could be used instead of JavaScript on the frontend for better scalability, maintainability and reliability due to its static type system.

From the feature enhancement perspective, there is a lot of potential in AI assisted analysis, especially of documents that would take a considerable amount of time for users to consume and analyze. Examples for these include the AI-based analysis of earnings call transcripts, but also the queried news articles could be analyzed, and the most important findings presented as a summary. There are many areas where machine learning models and methods could be applied, such as automated annotation of graphs, anomaly detection, financial fraud detection are amongst many other possible use cases. In addition, providing alternative valuation methods for different kinds of companies seems like an area for improvement. The current valuation method produced results comparable to analyst estimates in case of mature tech companies, which are not capital intensive. But this valuation method ignores asset value that is owned by the company for example. Therefore, to value a later-stage non-tech company or to value an early-stage company different methods could be offered to the user, providing other types of useful recommendations that are less narrow in scope.

3.5 Conclusions and thoughts of the author

Software engineering projects are rarely a one-person endeavor. However, in the early stage of an idea it can be valuable and cost-effective to manage a software project personally to a certain stage such as an MVP, at which point it can be handed over to users, and the user engagement could provide the feedback which could lead to the expansion of the project. Finding the product-market fit this way and iterating on the solution is a resource-efficient way

to start a product and scale up a company. I have worked in a software company called Xentral, which started this way in Augsburg, Germany by the founder who created the software as a personal project for his own needs for enterprise resource planning and kept iterating on it. When he realized that other entrepreneurs found his product useful, he let them use the software, gathered valuable feedback and kept iterating on it. At a later stage the project started to scale up, and the founder acquired significant funding that allowed hiring and building a team around the product (About Xentral, 2024). The expanded team in turn optimized, refactored and enhanced the product with new features based on feedback from a continuously increasing number of users. There are many examples and proponents of the MVP model, including the founders of Dropbox who started with a 3-minute demonstration video as an MVP to showcase the product and when it received overwhelming positive feedback it validated the necessity for this product (How Dropbox Started, 2024). I find value in exploring this early-stage process and documenting it with the diary-based method, which highlights important learnings, caveats and insights into the process as experienced by the developer. The resulting implications, findings and lessons from the experience could be used to derive a blueprint or to serve as an anchor point for developers approaching a similar time-constrained full stack software application MVP project.

References

About Xentral, (2024, March 1). Xentral. <https://xentral.com/en/we-are-xentral>

Baser, D., & Dashora, P. (2023). *A Study of Behavioral Finance in Investment Decisions*.

Bichler, S., & Nitzan, J. (2010). *Systemic Fear, Modern Finance and the Future of Capitalism*.

Board of Governors Federal Reserve System. (n.d.). Why does the Federal Reserve aim for inflation of 2 percent over the longer run?.

https://www.federalreserve.gov/faqs/economy_14400.htm

Bolger, N., Davis, A., & Rafaeli, E. (2003). Diary Methods: Capturing Life as it is Lived. *Annual Review of Psychology*, 54(1), 579–616.

<https://doi.org/10.1146/annurev.psych.54.101601.14503>

Britannica, T. Editors of Encyclopaedia (2024, January 18). Linux. Encyclopedia Britannica.

<https://www.britannica.com/technology/Linux>

Damodaran, A. (2011). *The little book of valuation: How to value a company, pick a stock, and profit*. John Wiley & Sons.

Haesevoets, T., De Cremer, D., Dierckx, K., & Van Hiel, A. (2021). Human-machine collaboration in managerial decision making. *Computers in Human Behavior*, 119, 106730.

<https://doi.org/10.1016/j.chb.2021.106730>

How Dropbox Started as A Minimal Viable Product. (2024 March 1). TechCrunch.

<https://techcrunch.com/2011/10/19/dropbox-minimal-viable-product/>

Learn to Invest - Investors Grow. (May 28, 2019). Discounted Cash Flow - How to Value a Stock Using Discounted Cash Flow (DCF) - DCF Calculation [Video]. YouTube.

https://youtu.be/fd_emLLzJnk?si=1wttfzb8v2kbN5b7

RealVNC, (n.d.). *All you need to know about VNC remote access technology*, Retrieved January 30, 2024 from <https://discover.realvnc.com/what-is-vnc-remote-access-technology>

Ries, E. (2019). *The lean startup: How constant innovation creates radically successful businesses*. Penguin Business.

Sawyer, C. (n.d.). *...about Chris Sawyer & Game Development*. Retrieved January 25, 2024 from <https://www.chrissawyer games.com/faq.htm>

Solomon, B. (n.d.). *Async IO in Python: A Complete Walkthrough*. Retrieved February 6, 2024 from <https://www.chrissawyer games.com/faq.htmhttps://realpython.com/async-io-python/>

Seessel, A. (2022). *Where the money is: Value investing in the digital age* (First Avid Press hardcover edition). Avid Reader Press, an imprint of Simon & Schuster, Inc.

Seibel, M. (n.d.). How to build an MVP. Retrieved March 1, 2024, from <https://www.ycombinator.com/library/lo-how-to-build-an-mvp>

Unterhitzenberger, C., & Lawrence, K. (2022). Diary method in project studies. *Project Leadership and Society*, 3, 100054. <https://doi.org/10.1016/j.plas.2022.100054>

Webster, I. (n.d.). Stock market returns between 1957 and 2023, Retrieved February 13, 2024 from <https://www.officialdata.org/us/stocks/s-p-500/1957?amount=100&endYear=2>