I. Cover Page:

Project Name: The Conscious Investor

Team Number: 10

Team Members:

Kachik Ashkaryan, kachik@seas.upenn.edu, CIS

Adya Aggarwal, adya@wharton.upenn.edu, SSE

Jason Tran, jastran@seas.upenn.edu, CIS

Anirudh Agrawal, aanirudh@seas.upenn.edu, NETS

Hagar Elhanbly, elhanbly@seas.upenn.edu, SSE

Advisors:

Professor Swapneel Sheth: swapneel@cis.upenn.edu

Date: May 5th, 2022

Table of Contents

II. Executive Summary:

We aim to empower young retail investors today who do not have a way of evaluating their portfolios and getting recommendations on which stocks to invest in based on Environmental, Social, and Governance (ESG) metrics they care about. Our solution helps bring transparency to investing in companies and increases access to information catered to our users that would make a stronger investor.

Our tool helps customers consider ESG metrics in their investment strategy that often go unmeasured. Our solution's bread and butter comes from a personalization page, where users are given 18 categories with weights they can adjust based on how important that category is to their personal beliefs. These weights are carried throughout our web application to give our users quick information about their personalized ESG scores in terms of their portfolio and companies they are interested in investing in. Furthermore, we try to display financial information as well as an attempt to blend financial and social information.

A recommendation page is at the heart of this blend, where we use a linear regression we made that combines 6 financial metrics and personalized ESG to give viable stocks users should investigate that perform decently well in the long term and align with their personal values.

III. Overview of Project:

In recent years, there has been a shift towards socially responsible investing and a good way to do that is assessing a company's ESG score. According to a recent survey by WSJ, 85% of individual investors surveyed are interested in sustainable investing. 84% of individuals surveyed believe companies following ESG practices are better long-term investments.

But first, what is ESG? ESG stands for Environmental, Social, and Governance. It is a tool for measuring a company's level of sustainability. Environmental factors take into account how a company affects its environment; for example, its carbon footprint, energy consumption, and waste disposal. Social factors encompass a company's relationship with its employees; for example, its approach to human rights and issues like diversity and inclusion. Governance factors involve a company's treatment of minority shareholders, board diversity, and political contributions. The ESG score ranges from 0-100; generally, any score below 50 is considered low, while any score above 70 is a good score.

Companies with better ESG scores have a stronger financial performance based on operational metrics, such as corporate financial performance (CFP), return on equity (ROE), and return on assets (ROA). Furthermore, there is evidence across numerous time periods and regions that Integrating ESG into the investment process benefits portfolio risk and return as it yields lower volatility, lower risk, and therefore higher risk-adjusted returns.

IV. Business Model:

Value Proposition

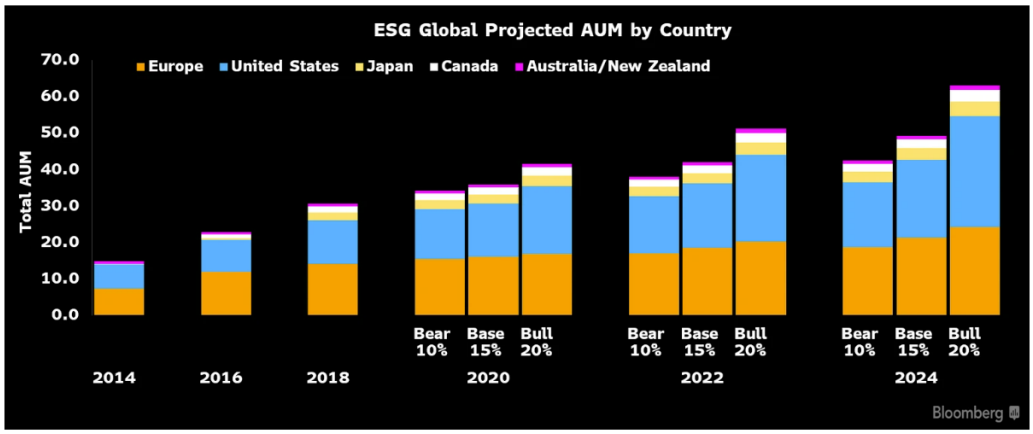
When coming up with this project, we considered the project's impact, and we were very content because this project does a two-fold job. One, it helps investors be conscious in their investment decision-making by considering an overlooked factor that is the ESG, which has proven to be a key indicator of a company's long-term performance. Second, it pushes companies to integrate ESG policies to better adapt to today's business climate. The Caveat here is that this only holds true if there are a sufficient number of investors who subscribe to conscious investing. Thereby, our project provides twofold value and increases the standard of living. The project's main objectives are to bring transparency and access, help investors take into account the 'unmeasured' or 'unrepresented', and mitigate risks, provide higher returns, and cater to investor belief.

Market Opportunity and Customer Segments

In 2021 $120 billion poured into ESG assets and some analysts predict the ESG market size to surpass $50 trillion. 72% of US adults expressed interest in ESG investing. The amount invested in ESG increased tenfold from 2018 to 2020 and 25 fold from 1995 to 2020. We expect a similar increase if not more in years to come as ESG investing gains more momentum.

Below we see a graph of global projected AUM for ESG assets by country

ESG projected global AUM



Source: GSIA, Bloomberg Intelligence

According to ESG Analytics.Ai there are 4 types of customer segments deeply interested in ESG investing:

1. Retired people who want to be able to get steady returns 20-40 years from now and want to protect their retirement savings
2. Valuation treaders who hope to make long term sustainable financial decisions and is considering ESG criteria because they think it is a newly emerging risk
3. People guided by personal values who want to emphasis social inclusion and investments are guided by the addition of ESG criteria
4. People who want to create positive community impact and thinks that ESG criteria would help guide their investment to their life's mission

Competitors

The competitors for our product are WealthFront and Mergent Online. We bridge the gap between access to information and providing recommendations as our competitors either charge a lot of money for investing on the investors behalf or don't disclose enough information about how they invest. We expect to have a freemium business model where we charge customers $15 per month for providing recommendations on which stocks to invest in (right now the recommendation is for research). We expect to grow the number of our customers at a rate of 5% month over month (due to marketing and word of mouth efforts) and eventually integrate with a platform like robinhood where our product can be used as a plugin for ESG investing.

Cost and Revenue Model

The costs for out product are as follows:

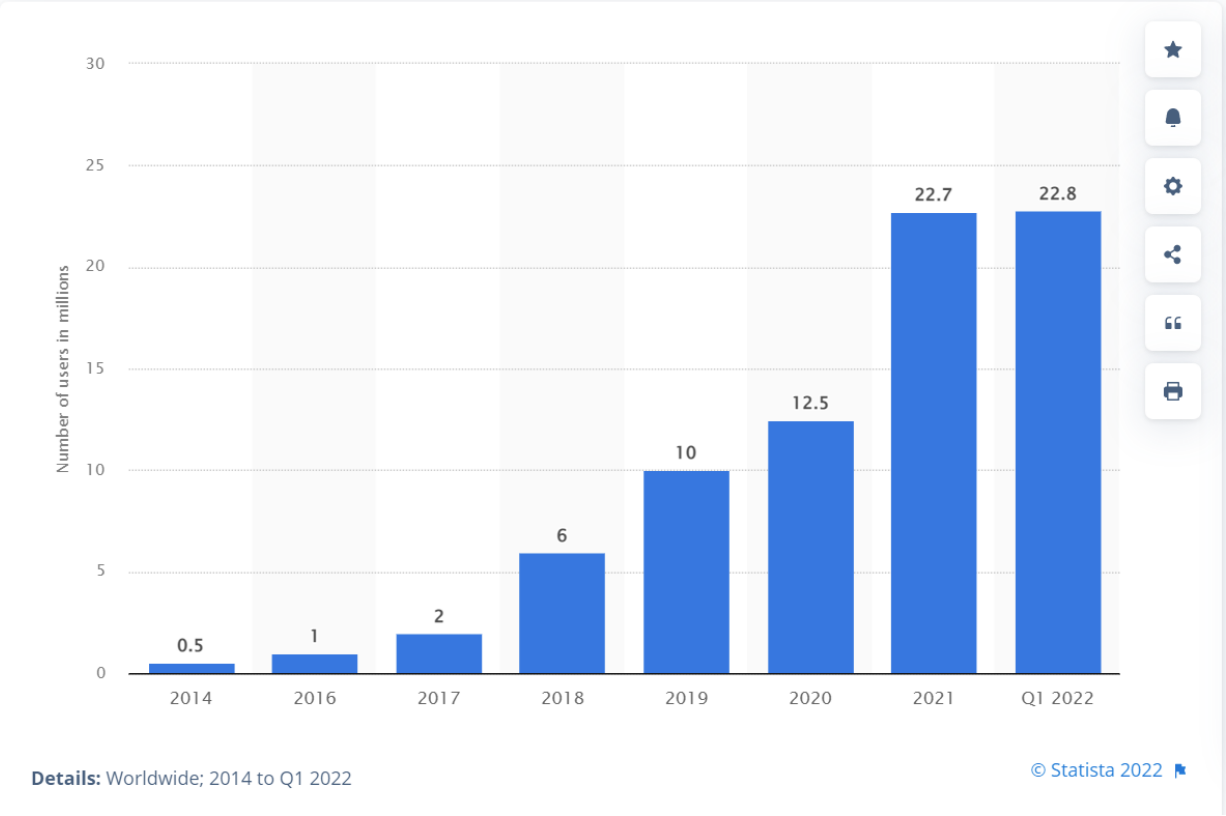
1. Database hosting for cloud service (MongoDB Dedicated Plan): It costs $57 per month or $0.30 per month. If we scale the business and get a million reads or transactions this cost will be reduced to $0
2. Website Hosting Cloud Service: This service costs around $25 per month
3. External API: This service costs $14 per month

As these costs show, we do not incur any additional costs for every transaction or read which means that these are fixed costs and we do not incur any foreseeable variable costs. As our business scales, we will be able to decrease the cost per person incurred.

Revenue Model

We wont be able to immediately commercialise our product because the recommendation portion of our product is for research purposes only right now as recommendations for financial securities need additional approvals to be available for commercial use. The way we aim to make money through our project is through a freemium model where the recommendations on companies will be paid but the ESG health of the portfolio will be free. Once we are able to charge this fee for utilisation, we aim to charge about $10/month for the recommendations portion. This fee will be able to get the user unlimited recommendations on an unlimited selection of criteria on our webpage.

As we have illustrated above, this is a growing market space. Most researchportals use a 15% growth rate for AUM under ESG, which is half the pace of the last five years (to be conservative in the estimate) and even with that growth rate ESG AUM could top at $140.5 trillion grobally by 2025. We’re looking at the number of users on Robinhood to see the number of users that could use our app. We ar going with a conservative estimate of 25% of Robinhood’s users every year.



From the graph above, we can see that in the first year we would have 125000 users, which would put us at a first year revenue of $12.5mm.

V. Technical Description:

Working in the investment space, it was imperative to us that we followed legal requirements when building our web application. Data privacy for our users is vital, thus we use OAuth to hold username and passwords. We also decided not to use the Robinhood API since this would require collecting our user's private credentials. Other regulation constraints we considered was bias in the data we were collecting and displaying for our users.

ESG scores we looked at were calculated by Refinitiv in a manner which relied on environmental, social, and governmental factors that were collected without fiscal information as the priority (examples include gender ratio of employee workforce, environmental pollution, etc.). Refinitiv is the leader in the field, and they have published many works on how they reconcile sources of bias in their calculations.

Originally, we considered creating a Chrome extension where people can quickly access information about stocks they are considering investing in. But, we realized that this would prevent us from having a personalized effect on our users, as well as not being able to implement a recommendation system for our users based on their preferences. Thus we settled on building a web application instead.

For our solution, we analyzed the data Refinitiv uses to evaluate a company's ESG score. We found that they break down their research into 18 subcategories. To leverage these subcategories, we first have users go through a survey to establish a priority level from 1 to 5 for each of the subcategories with 3 being the base. We leverage these priority scores to eventually return personalized ESG scores of portfolios to the user. From here, we wanted to look into a recommendation system for users.

We looked into various regression models that defined a relationship between the firms' ESG scores, returns, volatilities, and size. A published research paper from refinitiv analyzed the impact of ESG score on companies in 6 different indexes (S&P 500, STOXX600, Australia ASX 300, Japan Nikkei 500, China-A-share, South Korea Kor 200 indices) over eight years. The independent variables used were *market value of the company, cash flow-to-price ratio, book-to-price ratio, total ESG score,* and *annualized variance of the stock*. The dependent variable they used was *cumulative excess return*, which upon our research serves as an indicator for long term growth in a company.

We chose the regression model calculated in this paper because:

- The p coefficient they found in their research showed strong support for the regression.
- To handle collinearity and extraneous features in the model, we used sci-kit learn's dimension reduction library to only keep the features that explain 90 percent of the variance in the data.
- The variables they use were ones we found in our research to be indicative and critical points investors collect information about when assessing a company
- The ESG scores they utilized were consistent with what we had collected, and thus our regressions would align

Second, we moved on to our analysis of the ESG scores we had collected and the regression outcomes of the companies we were focused on. Our main goals for our end user is to provide them with a way to evaluate their current portfolio using the scores we collected, as well as recommending companies for them to research based on their societal values and our regression. For the former, we first used web scraping to collect ESG scores of 1000 companies in the Russell 1000 Index (as a starting off point) and stored these in our MongoDb database. From here, we created an API to create users, add stocks to their portfolio, and using our ESG database, return information on their current investments through the lens of ESG. In order to make the scores personalized, we normalized the ESG score by the subcategory weights we mentioned earlier.

Next, we used web scraping to get real time financial data for our regression. We filter information from the user about the companies they are interested in, for example an industry they would like to invest in, then using their ESG subcategory priorities, we found the 10 companies their values align the most with. Finally, we take our regression, and use a Python API to display information about our recommendations as well as their expected cumulative excess return to give user's an indication of how a company may perform long term.

We used Collab notebooks for their ease of use and easy collaboration. We first executed the key functions: adding and removing companies from the portfolio and performing the regression to calculate cumulative excess return. Next we used flask to turn python functions into API calls for the web application and added the remaining functions needed for our solution to work.

Third, constructing the web app for the user to interact. We decided on a web app version initially since:

- Faster time to market due to the decreased development time and the ability to reach both Android and Apple users.
- No download is required
- Less expensive and very responsive design.
- Send out updates and fixes faster
- Launch a mobile app version after seeing user traction and add requested features

For the web app, we used MongoDB for the backend. We chose it for its versatility, specifically because it supports auto-sharding. Relational databases generally scale vertically, in which a

single server hosts the complete database. This can limit the scale of operation, with the solution being to add horizontally and add servers rather than concentrating capacity on a single server. MongoDB allows for auto-sharing to automatically spread out data across various servers.

Overall we utilized the MERN stack. We used the Rest API for function calls. Express (Node.js framework) to form the back-end of the solution. We chose this as it is asynchronous, single-threaded, fast & scalable. We used React.js to build the user interface for the web app. Finally, Node.js as the JS Runtime Environment.

After implementing our web app, we used Jest for testing UI, unit testing, and end-to-end testing with cypress and supertest to make sure our frontend and backend worked separately and together. For the backend, we managed to web scrape 1000 companies' financial and ESG information we presented to our users, from 4 different sources. For our company page, we are able to scrape news articles in less than 200 ms. Finally, for our linear regression, our initial iteration took over 1 minute per stock recommendation. We used clever caching techniques recommended by our CIS 350 professor for software engineering to bring the time down to 100ms per stock. In our frontend, our goal was ease of navigation and not to overwhelm our users with the amount of information we wanted to display. We prioritized a simple UI while performing large data operations like rendering 1000 company components in less than 500ms. Searching for a specific company renders in less than 50ms, finally changing user preferences reflect in the dashboard in less than 100ms. This was vital for us since this feature is what differentiates us from other ESG investing platforms.

We firmly believe that our product met our goals of users feeling personally catered to as well gaining access to information not readily available to them on other platforms. Taking a step back, the results show that the application we built is well equipped to be used as an app in development by multiple users since it works quick enough to not cause frustration, and the data we provide is reliable and novel. We are fitting a niche that is not being catered to to the extent that we believe it should be.

Our web application fills a gap that is desperately needed and it has the potential to have a profound impact on investors who care about what their investment money is supporting.

VI. Self-learning:

Our product is a full-stack web application for users to interact with. Last semester, we were focused on web scraping, data wrangling, and databases to gather the necessary data for our web application. These skills required us to learn Python and its frameworks to web scrape and some database fundamentals such as NoSQL to store the large amount of data on MongoDB, which our last report covered in detail.

This semester, most of us were new to full-stack web development so we had to teach ourselves Javascript and its necessary frameworks. Integral to our self-teaching, we had to teach ourselves the frontend frameworks and languages such as HTML, CSS, Javascript, React.js, and MUI (CSS styling framework). The benefit of learning these web frameworks and languages helped us develop a graphic user interface for users to interact with the data in the backend. HTML, CSS, and Javascript allows the user to interact with our product on the web. React.js helped us with simplifying HTML, CSS, and Javascript into “one language”, making common HTML elements into reusable components, and reducing the code size into smaller lines.

Second, we had to learn backend frameworks and languages such as Javascript, Node.js, Express.js, and Mongoose. We needed to develop a RESTful API to connect our database with our frontend code. The RESTful API will pass our data from our database to the frontend. The tools that we had to teach ourselves to develop the RESTful API were Node.js, Express.js and Mongoose. Mongoose is a Javascript library that allows users to write MongoDB queries to MongoDB databases.

Finally, we wanted to teach ourselves user authentication in web development, specifically in javascript full-stack web development. To accomplish this we had to learn Passport.js to develop our own user authentication for signups and logins. For the user authentication to be secure, we used Bycrpt.js to encrypt sensitive data and information that will be stored in our database. We could have used an external OAuth for Passport.js like Google, Twitter, etc., but we wanted to implement our own user oauth just for the sake of learning.

All the programs, languages, tools, and frameworks listed above were ones that we had to teach ourselves. However, some of our group members knew most of these tools already. Some of the specific classes and knowledge that are most helpful for the project are CIS 550, CIS 557, CIS 197, CIS 196, Intro CIS classes, and Javascript. Some of us have taken extensive CIS courses and came from different backgrounds. With a diverse background of CIS, we were able to apply the knowledge from pervious courses to teach ourselves new ones. By taking intro

CIS classes we were able to pick up new coding languages and frameworks easliy. Also, by taking CIS 557, CIS 550, CIS 197, and CIS 196, we were able to guide other memebers on what frameworks to learn to develop our full-stack web applications. This process made it easier for us to learn and develop our product faster.

VII. Ethical and Professional Responsibilities:

Abiding by our engineering professional duty to avoid bias, we made sure that the data we use is strictly empirical. To ensure data privacy, we will make sure that the user's password is encrypted if it needs to be stored in our database for user authentication. Otherwise, we will use an external OAuth like Google, Twitter, etc. for users to login in to our web application. This will ensure that we keep the user's data confidential and secured. In our web application, we provided the users with full access to the data and calculations we use to provide them with stock recommendations.

Furthermore, our project revolves around corporate ethics. It is about holding publicly traded companies accountable for their misgivings that range from social issues, ethical issues, environmental issues, and more. We want to be responsible for giving investors full transparency to the companies their money supports, as a means to navigate their funding towards a goal they themselves more closely align with. Given our responsibility as engineers to be objective and truthful in our reports and statements, we made sure to inform users that the stock recommendations we give are based on their personal values and not for investing.

VIII. Meetings:

As a team, we have been meeting 2-3 times a week for assignments, for office hours with TA's, and for brainstorming sessions. We exclusively meet over Zoom to collaborate at the moment because it is most flexible with our schedules. We decided to split the work by our strong suits, with tech development being the majority of work, then one student working on design and another working on testing. Working with 5 members comes with communication difficulties. However, throughout the semester we were able to better understand each other's schedules, preferred way of communication, and work around them.

IX. Proposed Schedule with Milestones:

We built a roadmap, allocated work, and visualized the basic design of the website. We have also chosen our key metrics and have defined a mathematical relationship between them. Next, we need to calculate the returns and the variance on these returns. Meanwhile, we will set up a database and create complex queries. We are using web scraping to retrieve company information and ESG values. Next, we will need to automate data retrieval and then test the database. This semester we are focussed on the backend and having a working prototype with the key features.

Key milestone targeted (Fall Semester)

	Key Tasks	Ani	Kachik	Adya	Jason	Hagar
1.	Build a roadmap	11/7		11/7		11/7
2.	Choose key metrics	11/13		11/13		11/13
3.	Define a mathematical relationship		11/13	11/13	11/13	11/13
4.	Set up the database and define relationship, create complex queries		11//18		11/18	
5.	Webscrape for company information and ESG values	11/21	11/21		11/21	
6.	Automating Data Retrieval (ETL)		11/28		11/28	
7.	Test DataBase	11/30		11/30		11/30

Milestone Targeted for Spring Semester

	Key Tasks	Ani	Kachik	Adya	Jason	Hagar
1.	Build Figma mockups for individual pages	1/15		1/15		1/15
2.	User Authorisation	1/30	1/30		1/30	
3.	API connection with Robinhood	2/10	2/10		2/10	
4.	Build the search capability/Test		2/25		2/25	
5.	Build components for Dashboard /Test	3/10	3/10	3/10	3/10	3/10
6.	Build individual company page /Test	3/10	3/10	3/10		3/10
7.	Build recommendation page/Test	3/25	3/25	3/25		3/25

X. Discussion of teamwork:

Every weekend, our group would meet together to discuss current progress, concerns, and future work that needs to be done. We figure out each other's strengths, weaknesses, and skills in certain areas and divide the work among all of us. Additionally, if one of us is struggling in a certain area, the other steps in and provides help for each other. Below is a list of contributions for every team member in this group:

- Kachik Ashkaryan Contributions:
    - Implement a fully functional and secured user oauth both frontend and backend
    - Implement encryption and security for backend
    - Work on backend code and frontend code
    - Actively web Scrape Data to put into our databases
    - Design and implement backend express app to perform api functions
    - Designed and implemented all frontend using MUI and bootstrap library
    - Implemented regression with machine learning model to justify metrics we used in recommendation algorithm
    - Frontend and backend for regression, personalization, search, routing, and authentication pages
    - Work on Senior Design assignments

- Adya Aggarwal Contributions:
    - Research and come up with the backend regression model and formula
    - Work on the design/layout of the web application pages
    - Work on Senior Design assignments

- Anirudh Agrawal Contributions:
    - Work on frontend and some backend code
    - Work on UI/UX for easier navigation across the app
    - Work on Senior Design assignments

- Jason Tran Contributions:
    - Setting up the code base both frontend and backend and connecting the frontend and backend together
    - Implement a fully functional and secured user oauth both frontend and backend
    - Implement encryption and security for backend
    - Work on frontend code and a little of backend code
    - Work on Senior Design assignments

- Hagar Elhanbly Contributions:
    - Work on Senior Design assignments
    - Work on the initial design/layout of the web application pages
    - Create the figma mockups for the website pages
    - Create visual material for the presentations

As a team we encountered a technical challenge. For the technical challenge, we were mainly concerned about our design and technology frameworks as there are multiple resources, programming languages, and frameworks out on the web. We often had discussions about design as it is important to make things simple and realistic for everyone to learn and develop this project. Additionally, we often have agreements and disagreements on certain kinds of technologies to use as there are pros and cons to using each one. Specifically, some of us came from a strong background in full-stack development in javascript while others are strong in python or even have little coding experience. What helped us resolve this technical challenge is

that we spent time discussing as a group and understanding each other's ideas. We worked as a team to discuss our opinions, trade offs, and skills with each other.

Our original tech-stack was python web-scraping and a full-stack javascript web application. Now, our tech-stack is python web scraping and backend while a javascript/react frontend. In the end, we decided to make some changes in our technology stack as it was the right direction for everyone and it provided the most benefits.

XI. Budget and justification:

For our original budget we only needed Linkedin premium, Udemy Tutorials, Database Cloud service, and Website Hosting service described in detail below:

Resource	Cost	Number of months	Total Cost
Linkedin	$29.99/ month	~8	$240
Tutorials https://www.udemy.com/course/react-the-complete-guide-incl-redux/ https://www.udemy.com/course/api-and-web-service-introduction/ https://www.udemy.com/course/web-scraping-in-python-using-scrapy-and-splash/	~$100	N/A	~$100
Database Hosting Cloud Service (MongoDB Dedicated	$57 / month or $0.30 / million reads or **$0 if less than million**	~8	$456 (MAX)

Plan)	**reads/transactions**		
Website Hosting Cloud Service	~ $25 / month	~8	$200 (MAX)
		TOTAL:	$996 (MAX)

Figure 1: This table above is our original budget

There were many changes to our budget as we adjusted the tech stack to meet the needs of the project. We decided to remove LinkedIn Premium as we have an external API that has most of the data and functionality that we needed such as regression model, cash flow, book price, stock volatility, and much more. Since our web scraping code is mainly written in python we decided to change our Backend from Express.js and Node.js to Flask which required us to purchase Udemy courses to learn more about python and Flask. The changes are shown with our new budget table:

Resource	Cost	Number of months	Total Cost	
Tutorials https://www.udemy.com/course/react-the-complete-guide-incl-redux/ https://www.udemy.com/course/api-and-web-service-introduction/ https://www.udemy.com/course/	~$100	N/A	~$100	

web-scraping-in-python-using-scrapy-and-splash/				
Database Hosting Cloud Service (MongoDB Dedicated Plan)	$57 / month or $0.30 / million reads or **$0 if less than million reads/transactions**	~8	$456 (MAX)	
Website Hosting Cloud Service	~ $25 / month	~8	$200 (MAX)	
External API	~ $14 / month	~8	$112 (MAX)	
			TOTAL:	$868 (MAX)

Figure 2: This table above is our updated budget

XII. Standards and compliance:

IEEE P7002 - Data Privacy Process

- User's password is encrypted if it needs to be stored in our database for user authentication. Otherwise, we will use an external OAuth like Google, Twitter, etc. for users to login in to our web application. This will ensure that we keep the user's data confidential and secured

IEEE P7003 - Algorithmic Bias Considerations

- The data we use is strictly empirical for our linear regression. The ESG data we gather is from a reputable source (Refinitiv) which dozens of banks depend on for valid ESG information
- We are very clear about the fact that the recommendations we give is a recommendation on stocks to research that are close to the user's personal values, not for investing

IEEE STD 23026 - Engineering and Management of Websites for Information

- We attempt to put the power in the hands of users by making all the information we use for our recommendation and calculations readily available for our users in company pages. Thus access to the information we provide is not limited

EU General Data Protection Regulation (GDPR) - Defines regulations for collecting and handling user data

- Personal Data (email) is encrypted and stored by Google (third party).
- Keep a minimal amount of user data we need to give information and recommendations

XIII. Work done since last semester:

Last semester we only had our web scraping code and database which contains large amounts of data. However, our data is not intractable with the users. This semester, our goal was to develop a full-stack web application that will provide a graphical user interface that allows users to interact with our data in the form of a web application. To accomplish this, we learned and developed all the necessary programs, languages, frameworks, and tools. We achieved our goal by first teaching ourselves the tools listed in “Self Learning Section". Then, we developed the backend code to connect our database. Finally, we developed the frontend code to provide the graphical user interface and fetching data from our RESTful API. With these three general milestones, we were able to achieve and accomplish our goal of developing a full-stack web application.

XIV. Discussion and Conclusion:

These past few months have been fun but challenging. As we aim to tackle a relatively less researched topic, we had a lot to figure out early on. What metrics to choose, how they relate to each other, the impact of these variables on the performance of a company, how to measure the performance in the long term, how to break down the data so it is easier to understand and so on. Through long discussions and after consulting our advisors, we went over multiple research papers that tried to define a mathematical relationship and we chose the relationship published in “Probability and Partners” which would form the meat of the solution. After that we got to work to gather all the data required which involved web scraping and going through the Refinitiv database to get specific values for ESG scores and their breakdown. Next we did our analysis in python which involved connecting Robinhood API to collect stock information, allowing the user to put a weightage on the ESG components that they care about and spitting key information accordingly.

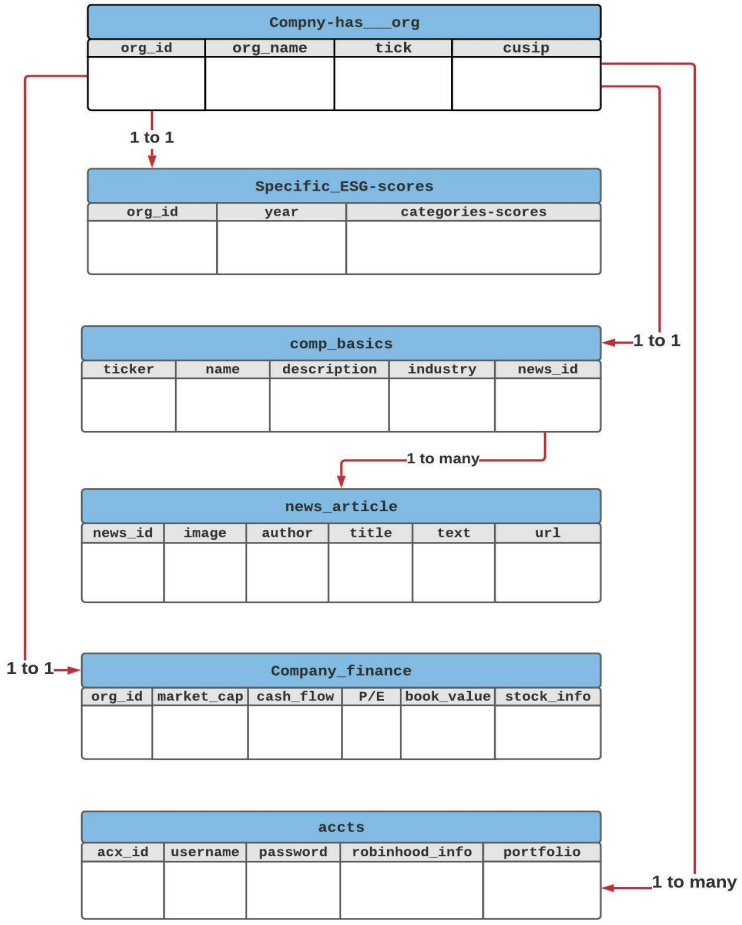
Potential challenges involved building the backend of the webapp, being able to update the results in real-time without any significant lags, handle multiple users simultaneously, and build a robust platform which will be immune to any cyber attacks and interferences.

Some of the valuable lessons we have learned is the importance of doing market research early on, the added value of giving the user a personalized experience, and that it is better to offer an innovative solution to one specific problem than to offer a non-original solution to multiple problems. Furthermore, we gained valuable experience in communicating unique value propositions and crafting structured elevator pitches.

XV. Appendices:

Appendix 1: Webapp layout

Appendix 2: Database Structure



Demo :

https://drive.google.com/file/d/1qD6d9UvLsc6xnc1rNpHcw7zc0JOCLJWf/view?usp=sharing