



Closing the World's Investment Gap

Intermediate Progress Report

Team Members

Ameer Shaikh
Amr Mahmud
Daniel Kecman
Stefan Momic

Contents

Project Update	3
Business Logic.....	3
Front-End.....	3
Back-End.....	4
Clarifications and Key Design Changes	4
Database Selection.....	6
Subsystem Accomplishments and Next Steps	9
Business Logic.....	9
Asset Universe Selection	9
Source and Preprocessing of Financial Data	11
Portfolio Generation Strategy and Models Used	12
Considerations for Robustness, Rebalancing & Re-Optimization	16
Considerations for Investor Preferences and Risk-Tolerance	17
Front-End.....	21
Finalized Wireframes.....	21
BPMN Diagrams and Descriptions.....	27
Usability Testing Protocol.....	29
Back-End.....	34
Description of Data Usage	34
Database Schema with Descriptions	35
Updated Project Plan and Timeline	39
Supplement A – Additional Wireframes	41
Appendix – Tearsheets	43
References	58

Project Update

The development of Alpha Factory has progressed significantly since the Initial Design Report and largely remains on schedule without any major delays. The team has strived to incorporate the feedback received from their key stakeholders, primarily Prof. Kwon and Hassan Anis.

This report aims to address and clarify the key design decisions that may have been overlooked during the Initial Design Report, address the changes made to the original plan, highlight the accomplishments for each subsystem and provide an updated view of the timeline and next steps.

Please refer to the “Clarifications and Key Design Changes” section of the report for more details on any design clarifications that were required (based on the feedback from key stakeholders) and key design changes from the Initial Design Report.

An overview of the accomplishments for each subsystem is provided below, with a more detailed examination of each subsystem presented in the “Subsystem Accomplishments and Next Steps” section.

Business Logic

Since the Initial Design Report, Alpha Factory made significant progress with their Business Logic components. In particular, the team conducted significant research on various ETFs, backtesting them over several time horizons in attempt to narrow down and finalize the selection of the asset universe. Furthermore, the team developed their selected models and backtesting capabilities using Python, more or less finishing the development of the first investment wizard function (i.e. analyze the risk and return profile of the portfolio over various horizons by backtesting over real data) as per the design requirements.

In addition, the team did extensive work on the considerations of investor preferences and risk-tolerance in their portfolio generation strategies. This includes creating a questionnaire and the corresponding logic behind determining the risk-tolerance level for each investor.

Finally, the team made significant progress on the preprocessing of data and further expanded their considerations of robustness, rebalancing and re-optimization as displayed in the corresponding sections below.

Front-End

Since the Initial Design Report, Alpha Factory made significant progress with their Front-End design. Specifically, the front-end web application is near completion in terms of UI and integration has already commenced between the three subsystems. Finalized wireframes have been completed for the portions of the web application that are still under construction. Wireframes and screenshots of the web application, along with their descriptions can be seen in the “Finalized Wireframes” subsection.

Additionally, the team has developed detailed BPMN diagrams to outline their business process flow and created a usability testing protocol to ensure that the website is functional and user-friendly by the time of delivery.

Back-End

Since the Initial Design Report, Alpha Factory made significant progress with their Back-End design. After finalizing the choice of Database Management System (DBMS) and source of financial data, the relevant data for further scoping of the asset universe was imported into our own databases. This allowed for further work on the integration between the back-end functions for data manipulation and implementation into the business logic.

In addition, the format for the rest of the data collections was changed to reflect further investigation into how they can be made to work together as efficiently as possible. These updated decision regarding the data schema can be found in the “Database Schema with Descriptions” section.

Clarifications and Key Design Changes

Alpha Factory has not made any major changes to the key design decisions identified during the Initial Design Report. The work has progressed as planned and the team has refined some of the design decisions made to be more specific. For example, during the Initial Design Report we decided to use ETFs as our investment vehicle, without specifying exactly which ETFs to use. Since the Initial Design Report, we have explored various ETFs, backtesting their performance over time to identify the specific ETFs to be used.

This trend is observed across all key design decisions. Namely, within the Business Logic subsystem we are proceeding with the decisions made regarding data source, portfolio generation strategies, parameter estimations, robustness considerations and portfolio validation techniques. As we progressed with our design, we encountered several issues forcing us to make additional considerations such as preprocessing data, rebalancing and re-optimization – please refer to the “Subsystem Accomplishments and Next Steps” section for more details. Likewise, within the Back-End subsystem, the decisions made regarding the choice of database, storage of financial data, data access layer, cleaning data and hosting have stayed the same.

Similarly, the major Front-End design decisions surrounding the user interface, gathering user inputs, display of computed data and decisions on the technologies used have not changed since the Initial Design Report. However, based on the feedback received from the Initial Design Report, we chose to dive deeper into this subsystem to clarify some of the design decisions made that were glossed over in the Initial Design Report.

Specifically, as per the project requirements, our robo-advisor is required to have the following three basic functions:

1. Given a portfolio of assets (i.e. specification of the securities and its weights), the wizard will analyze the risk and return profile over various horizons by backtesting over real data
2. Given a portfolio to be held for some time horizon, the investment wizard will seek to find a portfolio of assets that dominates it (e.g. has a better Sharpe-Ratio using a reasonably large class of assets to draw from (the user can set the number of assets k to include in the portfolio from a universe of n assets where $n \gg k$))
3. Given a return goal over some time horizon, generate a portfolio that has a projected return of at least the goal amount, but with minimum risk (e.g. portfolio variance, min VaR or CVaR over the time horizon)

The majority of the feedback received was concerning functions #1 and #2 not being satisfied, as much of the report was focusing on the generation of portfolios (function #3). These concerns were justified given that most of the Initial Design Report focused on our design decisions for function #3 as we believed that most of the differentiating aspects of our robo-advisor pertained to the third function, and hence required a more thorough and detailed analysis and justification. As such, we hope to clarify our key design decisions pertaining to the three robo-advisor functions in this section and supplement the clarifications provided here with the screenshots and wireframe sketches of our robo-advisor web application in the “Subsystem Accomplishments and Next Steps” section.

First, it is noted that our web application has two parts – the part that is available to the public (referred to as the “main site”) and the part that becomes available after logging in (referred to as the “user site”).

From the main site we have various pages including the About page where we detail our portfolio generation strategies, asset universe selection and several other key design decisions; the Contact Us page where we allow visitors to email us any questions; and Log In / Join Us pages that allow visitors to register to our platform and access the user site. It is on the main site, in the About page, that we provide visitors the ability to access function #2. Specifically, they are given the list of securities in our asset universe and the ability to enter the amount of each asset they choose to hold. After doing so and clicking “Enter”, our robo-advisor analyzes the risk and return profile of the portfolio they entered over various horizons by backtesting over real data (fulfilling function #1). Additionally, the robo-advisor finds a portfolio of assets that dominates the one they entered (fulfilling function #2). Ultimately, by doing this, we demonstrate how much better their performance would be by using our platform instead (assuming the portfolio they entered is the one they currently hold or would hold if they had to invest on their own) and encourage them to join our platform through a “Join Us” button.

Having demonstrated functions #1 and #2 from the main site, our user site focuses on providing function #3 (while also incorporating function #1). After a user joins our platform and creates an account by filling out the required sign-up fields, they are taken to a questionnaire aimed at assessing their return goals and risk-tolerance level over a specified time horizon. Using the information provided we suggest the optimal portfolio based on their inputs, fulfilling function #3. We also demonstrate function #1 during this stage as well by analyzing the risk and return profile of the portfolio we created over various horizons through backtesting on real data.

A more detailed view of our web application functionality can be seen in the “Finalized Wireframes” subsection below.

Database Selection

To store its various forms of data, Alpha Factory uses MongoDB, the popular database service that allows for the deployment, operation, and scaling of NoSQL databases on its servers. Mongo offers a free version of their services (up to 512 MB), which was estimated to be more than adequate for the purposes of Alpha Factory and made the choice to use MongoDB for our back-end database services quite easy.

The other possible DBMS solutions that could have been used in place of MongoDB ranged from a variety of products that run exclusively on a Structured Query Language (SQL) and those that don't require SQL for manipulating data. The decision to use the NoSQL MongoDB over a traditional SQL Relational DBMS (RDBMS) was discussed in the Initial Design Report, with the main decision factors listed below:

- Document oriented storage for lenience in creation of a variety of document types under one single collection of data
- Efficient querying to be able to quickly find the relevant data for any business process and effectively search at any level of data access

To further clarify this decision to use a NoSQL DBMS, and MongoDB in particular, please refer to the tables below, which were used to compare different alternatives that were found when making the initial selection.

List of possible RDBMSs

When investigating the possibility of using a SQL-oriented database, the three options that were most closely considered were Microsoft SQL Server (MSS), MySQL, and PostgreSQL. Some of the key features for each option are summarized below:

Name	Microsoft SQL Server	MySQL	PostgreSQL
Description	Microsoft's relational DBMS	Widely used open source <u>RDBMS</u>	Widely used open source <u>RDBMS</u>
Primary database model	<u>Relational DBMS</u>	<u>Relational DBMS</u>	<u>Relational DBMS</u>
Secondary database models	<u>Document store</u> <u>Graph DBMS</u> <u>Key-value store</u>	<u>Document store</u> <u>Key-value store</u>	<u>Document store</u> <u>Key-value store</u>
<u>DB-Engines Ranking Trend Chart</u>	Score 1051.55 Rank #3 <u>Overall</u> #3 <u>Relational DBMS</u>	Score 1159.89 Rank #2 <u>Overall</u> #2 <u>Relational DBMS</u>	Score 440.24 Rank #4 <u>Overall</u> #4 <u>Relational DBMS</u>
Developer	Microsoft	Oracle	PostgreSQL Global Development Group
Initial release	1989	1995	1989
Current release	SQL Server 2017, October 2017	8.0.12, July 2018	11.0, October 2018
License	commercial	Open Source	Open Source
Cloud-based only	no	no	no

DBaaS offerings (sponsored links)		<u>Google Cloud SQL</u> : A fully-managed database service for the Google Cloud Platform	<u>Google Cloud SQL</u> : A fully-managed database service for the Google Cloud Platform
Implementation language	C++	C and C++	C
Server operating systems	Linux Windows	FreeBSD Linux OS X Solaris Windows	FreeBSD HP-UX Linux NetBSD OpenBSD OS X Solaris Unix Windows
Data scheme	yes	yes	yes
Typing	yes	yes	yes
XML support	yes	yes	yes
Secondary indexes	yes	yes	yes
SQL	yes	yes	yes

After looking at the individual services, and the factors that went into the grading for each option, we agreed to trust the rankings provided by this chart, which give MySQL as the optimal choice between the three options that were considered. It is appropriate to mention that Oracle, which was the #1 ranked DBMS, was not considered for Alpha Factory because we were unsure of the range of features that was given with a free license.

List of Possible NOSQL DBMSs

The next step was to select a few contenders for NoSQL database services that would be considered given their preferable structure that doesn't rely on a query language for all functionality. The three alternatives selected were MongoDB, Amazon DynamoDB and Couchbase, which were given the top three rankings for Document-Oriented database management systems as shown in the table below:

Name	Amazon DynamoDB	Couchbase	MongoDB
Description	Hosted, scalable database service by Amazon with the data stored in Amazons cloud	JSON-based <u>document store</u> derived from <u>CouchDB</u> with a <u>Memcached</u> -compatible interface	One of the most popular document stores
Primary database model	<u>Document store</u> <u>Key-value store</u>	<u>Document store</u>	<u>Document store</u>
Secondary database models			<u>Key-value store</u>
<u>DB-Engines Ranking</u> <u>Trend Chart</u>	Score 53.81 Rank #21 <u>Overall</u>	Score 34.85 Rank #23 <u>Overall</u>	Score 369.48 Rank #5 <u>Overall</u>

	#2 <u>Document stores</u> #2 <u>Key-value stores</u>	#3 <u>Document stores</u>	#1 <u>Document stores</u>
Website	aws.amazon.com/-/dynamodb	www.couchbase.com	www.mongodb.com
Technical documentation	aws.amazon.com/-/documentation/-/dynamodb	docs.couchbase.com	docs.mongodb.com/-/manual
Developer	Amazon	Couchbase, Inc.	MongoDB, Inc
Initial release	2012	2011	2009
Current release		5.5.0, July 2018	4.0.3, October 2018
License	commercial	Open Source	Open Source
Cloud-based only	yes	no	no
Implementation language		C, C++, Go and Erlang	C++
Server operating systems	hosted	Linux OS X Windows	Linux OS X Solaris Windows
Data scheme	schema-free	schema-free	schema-free
Typing	yes	yes	yes
Secondary indexes	yes	yes	yes
SQL	no	SQL-like query language (N1QL)	Read-only SQL queries via the MongoDB Connector for BI
APIs and other access methods	RESTful HTTP API	Memcached protocol RESTful HTTP API	proprietary protocol using JSON

We see that in general, the ‘Document Store’ database systems are newer and therefore less established and ranked lower than their RDBMS equivalents. However, they are still preferred because of the reasons listed above. When we see the performance of these options in the grand scheme of database management systems, we see that MongoDB outperforms its counterparts by a significant margin.

The final decision that was made was to compare the potential benefits and drawbacks of implementing MySQL as opposed to MongoDB. Here, we looked at the overall design objective of making the implementation of the database as seamless as possible with the rest of the project. The main factor in selecting the preferred database system was through the cohesion of each solution’s API, a resource of paramount importance for reference purposes in order to ensure the proper use of the Python connector to the database. Here, MongoDB stood out with a very organized API that tackles each level of data access with a corresponding level of API documentation to give users a thorough tool for ensuring they can properly implement MongoDB’s Python connector PyMongo properly.

Subsystem Accomplishments and Next Steps

Since the Initial Design Report, Alpha Factory team members have made tremendous strides towards the creation of the final prototype. With no major delays and/or issues to report, the project remains on schedule to be completed by the December 3rd deadline.

A detailed look at the key accomplishments and what remains to be completed for each subsystem is provided below.

Business Logic

The team made significant progress in selecting the asset universe, developing their portfolio generation strategy and models, preprocessing and selecting financial data sources and various considerations for robustness, rebalancing and re-optimization.

In addition, the team made significant progress on considering the investor preferences and risk tolerance of their clients by creating an extensive framework as discussed below.

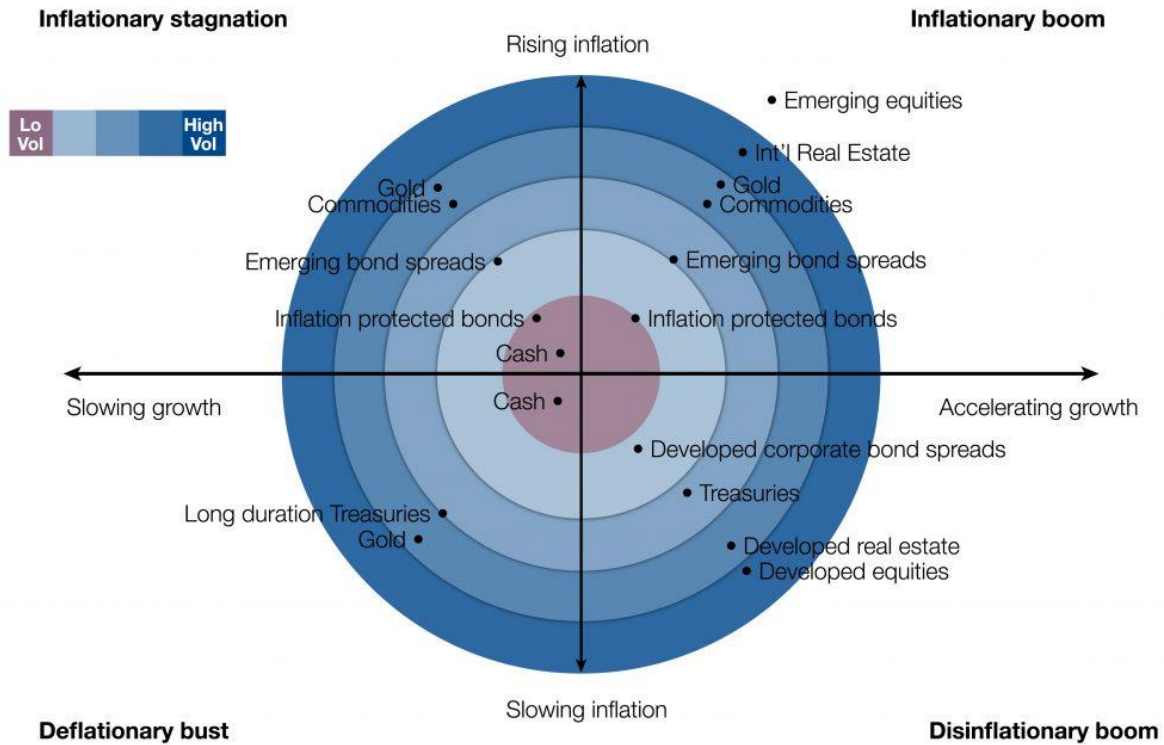
Asset Universe Selection

As noted in our Initial Design Report, Alpha Factory uses a top-down approach for selecting the asset universe, ultimately concluding to select ETFs as their investment vehicle of choice.

After considering the pros and cons of both individual securities and aggregate asset classes, it became apparent that aggregate asset classes were the right choice (refer to the Initial Design Report for more details). This view was further compounded when our key design features were considered. Namely, Alpha Factory strives to make investing accessible for everyone – as such we wanted to ensure that our customers can get highly diversified portfolios with low fees and low initial investment, while still being able to understand the assets in which they are invested. Using aggregate asset classes was the obvious choice as the fewer assets required to ensure diversified portfolio wouldn't be overwhelming to follow/monitor for our customers (in contrast to following an asset universe of hundreds of different securities) and the logistical benefits of having less rebalancing, smaller databases and guaranteed diversification made it easier to lower fees and the required initial investment balance.

Having selected to use aggregate asset classes, we turned our attention to selecting the specific asset classes and decided to base our asset selection methodology on Bridgewater's all-weather framework as an attempt to create portfolios that perform well in all financial environments. [3]

Asset Class Behaviours in Different Inflation and Growth Environments



Source: ReSolve Asset Management

Figure 1: Asset class behaviours in various inflation and growth environments

To gain exposure to the asset classes listed in Figure 1 above, ETFs were selected as the investment vehicle of choice as they cover the entire investment universe and are the easiest to invest in logistically. For example, in order to gain commodity exposure, we would require the use of futures and the constant rolling of contracts making the investment process significantly more complex. In contrast, we can easily gain exposure to commodities by investing in commodity ETFs. Other aggregate asset class investment vehicles such as mutual funds were not selected due to their higher costs and lack of readily available financial data making parameter estimation and investment decision making difficult.

Since the Initial Design Report, Alpha Factory has completed significant work on selecting the specific ETFs to be used by conducting several iterations and significant backtesting. Additionally, Alpha Factory decided on which ETFs to select based on the following criteria:

1. Similarity to index proxies (measured by tracking error against indices) with closer similarity being preferred
2. Low fees (measured by the Management Expense Ratio (MER)) with lower fees being preferred
3. Liquidity (measured by spread and Average Daily Volume (ADV)) with a minimum threshold being required (i.e. as long as we're trading something without a massive spread, we should be fine – want to avoid trading extremely rare and illiquid ETFs)
4. Data availability (measured by the length of historical data available) with longer time horizons being preferred

Using our defined criteria, the following ETFs (see Table A) were selected to gain exposure to the select asset classes. Specific asset classes were selected to provide diversity across both geographic region and asset type enabling an “all-weather” portfolio construction. All ETFs that were selected are USD denominated funds. This was a conscious decision as consistency was important to avoid currency exchange concerns and Canadian funds did not have sufficient diversity in their offerings.

Asset Class	ETF(s)
Developed Market Equities	SPY - SPDR S&P 500 ETF EFA - iShares MSCI EAFE ETF
Emerging Market Equities	EEM - iShares MSCI Emerging Markets ETF
Developed Real Estate	VNQ - Vanguard Real Estate ETF
Long Duration US Bonds	TLT - iShares 20+ Year Treasury Bond ETF
US Intermediate Bonds	AGG - iShares Core U.S. Aggregate Bond ETF
International Developed Market Bonds	BWX - SPDR Bloomberg Barclays International Treasury Bond ETF
Emerging Market Bonds	EMB - iShares J.P. Morgan USD Emerging Markets Bond ETF
Inflation-Protected Bonds	TIP - iShares TIPS Bond ETF
Municipal Bonds	MUB - iShares National Muni Bond ETF
Money Market (Short Term Bonds)	SHV - iShares Short Treasury Bond ETF
Commodities	DBC - Invesco DB Commodity Tracking Fund
Gold	GLD - SPDR Gold Shares ETF

Table A: Current Investable Universe (subject to future additions)

As evident in the table, most assets fall under fixed income however, we have yet to explicitly incorporate factor tilts into our asset universe. As we do so, we expect to see an increase in the number of equity focused ETFs with investment products for factor tilted equities more accessible than other asset classes. Incorporation will be done by replacing asset class ETFs with factor tilted counterparts where available. For example, in the place of US equities we will include US value and momentum ETFs. Selection of these factor ETFs will follow the same criteria as our original ETF selection (see above). Ultimately portfolio performance will be compared with basic asset exposure against factor tilted exposure to determine if inclusion is warranted. Current expectations are to include at the very least, value and momentum factor ETFs for all equity exposures (US, developed and emerging markets). Other factors to be considered where available include size, liquidity and quality.

Current back-testing with ETFs has been done purely with live data equating to approximately ten years worth. Incorporation of proxies in the form of indices with much longer histories will be done to improve the reliability of results. At this point we are confident that a 10-year period of data is sufficient for making most decisions as this period includes multiple economic regimes and is long enough to accurately capture differences in relative performance. Ultimately, we would expect that any portfolios we generate to perform relatively consistent across any 10-year time period.

Source and Preprocessing of Financial Data

As discussed in our Initial Design Report, Alpha Factory determined that getting financial data from Bloomberg Terminal was the best option (refer to the Initial Design Report for more details). With

Bloomberg Terminal lacking real time data fetching abilities and other APIs, such as Yahoo Finance, lacking access to index prices and Quandl lacking free historical ETF prices it became evident that no source of financial data is truly ideal for our usage. However, Alpha Factory opted to use Bloomberg Terminal at the expense of manual data fetching because of their index pricing. Since most ETFs have very limited historical data available, having access to the index proxies was very important for longer back testing capabilities. For this reason, ensuring the selected ETFs were similar to the index proxies was a key criterion in the asset selection above – specifically, knowing that a very limited amount of historical data was available for each particular ETF, we wanted to ensure that we selected ETFs that accurately tracked the indices so those indices can be used to fill missing data.

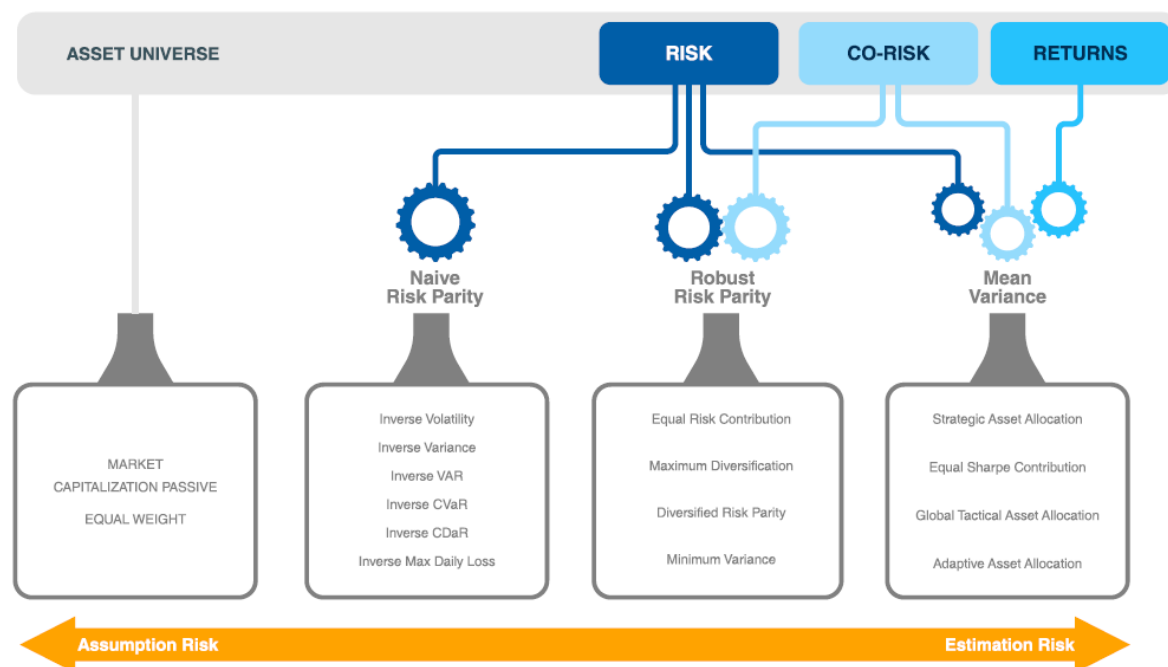
Using our selected sources, we have acquired the price time series for relevant ETFs and ensured the capability of acquiring index time series by testing for a select few.

Incorporating Indices requires preprocessing of the data in the form of evaluating and appropriately modifying the index return time series before combining them with their respective ETF return time series. Evidently, simply using the returns on the Index as a perfect proxy for the returns of an ETF before going live has multiple issues. Given that any ETF has some tracking error with respect to its index, it is important to use the available information we have to modify index time series so that they are better representative of the ETF in question. Our next steps involve using overlapping return series of each index and ETF to determine the tracking error and modify our index returns accordingly. This should incorporate explicit differences like management fees and implicit differences like management style.

Portfolio Generation Strategy and Models Used

During the Initial Design Report, Alpha Factory did extensive research and surveyed a vast landscape of portfolio generation techniques to determine and select their models. Noticing that portfolio generation techniques can be broadly classified based on the assumptions and estimates made on asset characteristics, we first narrowed down our selection into groups based on which parameters we believed we could accurately estimate. In selecting the portfolio optimization technique, we first classified them into one of the following groups, with Figure 2 illustrating how various portfolio optimization techniques can be classified based on estimation of asset parameters. [4]

- 1) Naive Risk Parity:
 - Requires estimates of a risk parameter for each asset while assuming no accurate estimates can be made on asset co-movement or returns.
- 2) Robust Risk Parity
 - Estimates both risk and co-movement parameters but assumes that returns cannot be accurately estimated.
- 3) Mean Variance:
 - Estimates all parameters (risk, co-movement and returns) and therefore is very susceptible to estimation risk.



Source: ReSolve Asset Management. For illustrative purposes only.

Figure 2: The Portfolio Optimization Machine

The so-called Mean Variance category was ruled out based on well-researched and documented evidence that estimating returns is an extremely difficult task (refer to the Initial Design Report for more details). [5, 6, 7, 8] The Black-Litterman model was also ruled out as we believe that we cannot rationally claim to have a better than average knowledge on future asset returns nor do we target investors with extensive investment knowledge – as such, in the absence of views, the Black-Litterman model is simply the market-capitalization weights and provides no added benefit for us as these are already captured in our selected ETFs. Finally, Naïve Risk Parity approaches, although suspect to less estimation error, make excessive simplifying assumptions. These techniques assume that no accurate estimation can be made with regards to how assets move together. It is evident though history that assets perform differently in certain economic conditions and this fact is vital to the foundation of the all-weather portfolio. As we believe that this relationship can be estimated with better than random precision, this leads us to abandon Naïve Risk Parity approaches in favour of Robust Risk Parity techniques.

Since the Initial Design Report, Alpha Factory has made significant progress by developing a Python based back-testing framework to validate research conclusions and empirically inform any other portfolio generation decisions. The back-testing framework enables the historical simulation of a given portfolio's returns producing a portfolio tearsheet that captures its performance and risk characteristics and can be used to evaluate and compare various portfolio generation techniques. See ([Appendix: Tearsheet 1](#)) for an example tearsheet generated of an equal weight portfolio rebalanced monthly and refer to Table B for details on the various components.

Having settled on a robust risk parity approach we confirmed our selection through our own empirical validation. To do so we first back-tested an equal weight portfolio of all our assets in the current investable

universe. The results for this portfolio with a monthly rebalancing can be seen in [Appendix: Tearsheet 1](#). After doing so, we performed a similar back-test with a static risk parity weighting scheme in the form of equal risk contribution with results shown in [Appendix: Tearsheet 2](#). As is evident and expected, the risk parity portfolio is superior to the equal weight portfolio on a risk-adjusted return basis (measured by Sharpe and Sortino ratios). Important to note is that this risk parity approach is a naive implementation as it uses all available data to generate volatility and covariance parameters that input into the generation of weights. This subjects the portfolio to a look ahead bias and therefore is not an accurate representation of real-time performance. To address this concern, a dynamic risk-parity portfolio was also back-tested. This dynamic portfolio using a rolling 200 day look back period to calculate parameters and update weights on an on-going basis. As with other portfolios, all rebalancing is done monthly. Although the results [Appendix: Tearsheet 3](#) maintain an improvement of risk-adjusted returns against the equal weight portfolio, the dynamic risk parity approach does not improve on the static risk parity. Interestingly, this result is contrary to conclusions from research claiming that dynamic approaches outperform static ones. This concept will be further explored with various combinations of parameters and assets. However, if our findings continue to confirm that a static approach is optimal we will use a long lookback period that would approximate a static approach and produce similar results.

An important aspect that we have yet to investigate is leverage; a key component of any risk parity portfolio. Leverage is required for adjusting a portfolio to various risk targets and is usually used to scale up the returns of a risk parity approach to desired levels. Leverage in terms of cost and specific implementation are to be explored and incorporated into our back-tests.

Additionally, our risk parity portfolios are currently a very crude implementation. Our methodology assigns equal risk to every asset in our universe. We do, however, have the capability of assigning desired risk targets for each asset. This enables us to be more selective in how we create our “all-weather” portfolio. Without doing so, and by blindly assigning equal risk to each asset, we are severely exposed to asset selection bias (results dependent on universe selection). Consideration and experimentation will be done on setting asset-specific risk targets.

As specified in our Initial Design Report, we also verified our choice to include a trend following overlay to all our portfolios as an added measure of risk protection. This concept was tested on both equal weight and risk parity portfolios. The specific trend-following approach employed is as follows: if an asset is above its 200-day simple moving average then it is deemed to be favourably trending and the target allocation is made, if not, the allocation is made to cash instead. As evident in the results ([Appendix: Tearsheet 4](#) for equal weight and [Appendix: Tearsheet 5](#) for risk parity and [Appendix: Tearsheet 6](#) for dynamic risk parity) the trend following overlays help reduce overall risk in terms of volatility, improve risk-adjusted returns and reduce drawdowns, all with respect to their non-trend following counterparts. Results generated are also conservative as when not allocating to an asset deemed to be trending unfavourably cash is held and assumed to yield 0%. In a more realistic implementation, allocations will instead rotate into something like a money market instrument with positive yield. This will result in greater returns at the expense of almost no additional risk, resulting in better overall risk-adjusted returns. Further testing will be done with rotation into various cash alternatives as well reallocation to favourably trending assets.

Other considerations yet to be included in our back tests are the effects of transaction costs and taxes which are affected by the portfolio’s methodology. We expect that the effects will be minimal, but for the

sake of completeness and accurate representation of performance we plan to have them as a factor in some way. In our current tests, the effects will, for the most part, net out given the similar turnover rates.

Component	Description/Details
Cumulative Return Chart	Displays the cumulative historical return over back-tested period.
Underwater/Drawdown Chart	Displays cumulative drawdowns experienced by the portfolio.
Positioning Chart	Displays allocation percentage to each asset and how it changes over the duration of portfolio. Weights are affected by the daily returns of each asset and are reset to their target based on the rebalancing frequency.
Overview	Details main characteristics of the portfolio: <ul style="list-style-type: none"> • <i>Start Date</i>: first day of returns • <i>End Date</i>: last day of returns • <i>Rebalanced</i>: how often weights are rebalanced to their target • <i>Trend Following</i>: if a trend following overlay is used will display the trend indicator otherwise will show N/A • <i>Weighting</i>: indicates portfolio weighting methodology
Statistics #1	Performance and risk metrics of the portfolio: <ul style="list-style-type: none"> • <i>Total Return</i>: cumulative return of portfolio from start to end date • <i>CAGR</i>: Compound Annual Growth Rate • <i>Annual Volatility</i>: annualized value based on daily standard deviation of returns • <i>Sharpe</i>: CAGR/Annual Volatility • <i>Max Drawdown</i>: Largest peak to trough loss • <i>Sortino</i>: CAGR/annualized volatility of negative returns
Statistics #2	Additional performance and risk metrics: <ul style="list-style-type: none"> • <i>VaR</i>: 99th percentile 1-day value at risk using historical approach • <i>CVaR</i>: 99th percentile 1-day conditional value at risk using historical approach
Performance Attribution	Displays the contribution of each asset to the overall portfolio's return in each year.
Yearly Returns Chart	Shows the total return of the portfolio for each year.

Table B: Components of a Portfolio Tearsheet

Considerations for Robustness, Rebalancing & Re-Optimization

As noted in the Initial Design Report, the relevant parameters requiring estimation for our selected risk-parity portfolio optimization approach are risk and co-movement of assets. Risk is measured by volatility of returns as it is the simplest to estimate and representative of alternative measures. Other types of risk measures like VaR, CVaR and maximum drawdown may be more representative of an investors preference to preserve capital but require additional assumptions with regards to return distributions introducing unnecessary complexity. The following statement adeptly shows how volatility can be an all-encompassing risk measure: “consider that, when a systematic strategy is regularly rebalanced, a large expected mean relative to volatility strongly implies a smaller risk of permanent loss, a smaller expected maximum drawdown, and a smaller expected shortfall.” [10] As long as our asset universe does not contain assets with extremely asymmetric return distributions (i.e. options, high yield credit, etc.) volatility should be an adequate risk measure. [11] Co-movement is measured by the covariance of returns as is common practice. Both parameters are estimated using a historical lookback period, therefore avoiding any look-ahead bias and resulting in more indicative backtesting results. Additionally, parameters are estimated using a rolling lookback period thus creating dynamic allocations in accordance with shifting covariances and volatilities.

To address any uncertainty in our parameter estimations, multiple computations and comparisons were made. By analyzing how varying parameter estimation factors affect results we can identify whether results are simply driven by data mining/overfitting of the parameters.

The first parameter in question was the rebalancing frequency; how often weights would be readjusted to their targets. Our decision to use a monthly rebalancing was based on both quantitative evidence and logistical feasibility. Daily ([Appendix: Tearsheet 7](#)), Weekly ([Appendix: Tearsheet 8](#)), Monthly ([Appendix: Tearsheet 1](#)) and Yearly ([Appendix: Tearsheet 9](#)) rebalancing frequencies were tested on an equal weight portfolio. Although there was a clear trend of decreasing performance as the rebalancing frequency was reduced, the differences were minimal. Daily and weekly rebalancing were ruled out due to significant turnover that would lead to higher transaction costs and yearly demonstrated noticeable deviations from target weights. As such Monthly rebalancing was selected providing a compromise between deviation and turnover.

The second parameter examined for robustness was the rolling lookback period; the amount of data that would input into the generation of dynamic risk parity weights. Using a 200-day lookback period ([Appendix: Tearsheet 3](#)) as a starting point, comparisons were made with both 150-day ([Appendix: Tearsheet 10](#)) and 250-day ([Appendix: Tearsheet 11](#)) lookback periods. Although there was stability above the 200-day look back period, using a shorter lookback period decreased performance noticeably. This concept will be explored further to determine optimal lookback period. As mentioned earlier, increasing the look back period drives the weights to a static risk-parity approach.

The third parameter analyzed was the trend following indicator. We decided to use the 200-day simple moving average in line with *A Quantitative Approach to Tactical Asset Allocation* [12] which also verifies the robustness of the selection by comparing it to other look-back periods. Our own testing validates their findings with 100-day SMA ([Appendix: Tearsheet 12](#)), 150-day SMA ([Appendix: Tearsheet 13](#)) and 200-day SMA ([Appendix: Tearsheet 4](#)) all having similar results. As we increased the look back period performance

started to suffer (300-day SMA [Appendix: Tearsheet 14](#)) but this is as expected given the nature of a trend following system. With too long of a look back, the system will react too slowly to changes in an assets performance and will most likely be late for both entry and exit.

The factor that remains to be assessed for its robustness is the asset universe. Multiple iterations will be tested to ensure selection of our universe is not the main driver of performance. Additionally, stability of results across small parameter variations will be continued to be monitored for indication of adequate robustness as significant volatility with small variations would indicate otherwise.

Considerations for Investor Preferences and Risk-Tolerance

As noted in our Initial Design Report, an investor's attributes, particularly their investment goals and risk tolerance, will factor into the optimal portfolio allocation. However, since risk tolerance did not inform the selection of the portfolio generation strategy, merely requiring slight modifications to generate the unique portfolio, it was largely ignored during our Initial Design Report. Specifically, in selecting our portfolio generation techniques we examined them on their property's agnostic of investor preferences as any technique can be tailored to produce portfolios for various risk tolerances with tactical uses of leverage and additional constraints.

Since our Initial Design Report, we turned our attention to determining investor preferences. Specifically, we want to consider the return goals and risk-tolerance for each of our investors and we further explore each category in the subsections below.

Return Goals

Everyone is different and has different investment goals. Often times investors do not even know what their investment goals are, and if they do, it's hard to put them in numbers/words. So try to match return goals to life goals by considering what our clients are actually saving for. With life goals usually easier to define, we find this approach to be more intuitive and user-friendly for our users.

Since we are a passive investment fund, we build our portfolios for longer time horizons (5+ years). As such, our return goals are largely catered to the following broad categories of life goals:

- Saving to purchase a car/home (**short-term**: ~5 to 10-year investment horizon)
- Creating an education fund for your children (**medium term**: ~10 to 20-year investment horizon)
- Saving for retirement (**long term**: 20+ year investment horizon)

In addition, we consider age, initial investment amount and the amount of money contributed each year by requiring the user to input all of these fields when creating an account. If the user can provide a specific return goal in terms of percentages, we can also accommodate that, but we find that putting a specific number to our goals is much harder to do and as such do not require that field to be filled out during our questionnaire!

Risk-Tolerance

Finally, our unique portfolio generation techniques depend on the risk-tolerance level of our investors. Rather than overfitting with a continuous spectrum of return goals and risk constraints, Alpha Factory aims to create five distinct portfolios to accommodate the different risk-tolerance levels of our customers. This is a trend that was noticed in our survey of existing solutions with Wealthsimple having only three distinct portfolio classes (called Conservative, Balanced and Growth). We find this strategy to better fit our business mantra as it would allow us to build and manage five distinct portfolios and pool our investor's money to the management of each. Ultimately, this will allow us to keep our fees low and require a low initial investment balance. With that said, we understand that the portfolios are not truly unique for every single investor, but we believe that is a compromise they are willing to make to ensure the costs are kept low – at the end of the day, money matters and lower costs mean more money is contributed to their life goals. More than this, we believe that having five distinct portfolios will still ensure that their views are met.

We have divided our five distinct portfolios as shown in the Table C below. To determine the risk-tolerance of our investors we require them to fill out a questionnaire consisting of several questions. To keep the process as simple as possible, we decided to limit our questionnaire to 15 questions containing only two options for each question. Doing so will allow our users to breeze through the questions since they will only have to consider two options. As an added benefit, this questionnaire method will also limit the extent of outlier data and greatly simplify the logic for determining the risk tolerance of our users.

Specifically, our questionnaire will consist of a series of scenarios with one option being risk-adverse and the other being risk-seeking in a random order (i.e. the risk-adverse option will not always be on the left in an attempt to reduce bias). Depending on the number of risk-adverse selections made and time horizon of the investment goal, users will be grouped in their respective portfolio as shown in the Table C below. Note that the number of questions that are required to be classified for each risk-tolerance level will depend on the time horizon and goals of the investor (further explained below). At the end of the questionnaire the user will be assigned a risk-tolerance level and be recommended a portfolio to hold. Users will be required to consent their approval on accepting the recommended portfolio based on their risk-tolerance and will be given the option to override this recommendation by selecting their own portfolio. For example, a medium-term investor who selected 6 risk-averse options will be classified as having a Neutral risk-tolerance and be recommended the Balanced portfolio but will be given an option to select another portfolio instead.

Risk -Tolerance		Extremely Risk-Averse	Risk-Averse	Neutral	Risk-Seeking	Extremely Risk-Seeking
Portfolio		Preservation	Conservative	Balanced	Adventurous	Aggressive
# of Risk-Averse selections	Short-term	12 to 15	9 to 11	5 to 8	2 to 4	0 to 1
	Medium-term	13 to 15	10 to 12	6 to 9	3 to 5	0 to 2
	Long-term	14 to 15	11 to 13	8 to 10	4 to 7	0 to 3
Tentative Target Asset Allocation		Equities: 10% FI: 90%	Equities: 30% FI: 70%	Equities: 50% FI: 50%	Equities: 70% FI: 30%	Equities: 90% FI: 10%

Table C: Risk-Tolerance Level Determination Logic

Note that although the users will get the same 15 questions, the number of selections required to be classified as each risk-tolerance level will vary by their time horizon and investment goals. The thought process behind the number of selections required is hopefully fairly intuitive: with a shorter time horizon, individuals are usually more risk-averse – as such, you are required to be more risk-seeking to be considered “extremely risk-seeking.” On the other hand, with a longer time horizon, you can be considered “extremely risk-seeking” even if you wouldn’t be considered “extremely risk-seeking” in a short-term setting as the longer-term investment horizon affords investors the luxury and comfort of more fluctuations. The numbers in the table above were selected with the aim of recommending portfolios according to the logic displayed below:

The Preservation/Conservative portfolios are best for:

1. Users with a short-term goal (~5 years) like buying a home or saving for vacation and are only comfortable with small fluctuations
2. Retired users who are comfortable with small fluctuations
3. Anyone else who’s really risk adverse

The Balanced portfolio is best for:

1. Users with a medium-term goal (~5 to 15 years) and are comfortable with some fluctuations
2. Users with longer term goals (15+ years) and are willing to take some risks to ensure funds grow over time but aren’t comfortable with very large fluctuations
3. First time investors who haven’t had experience with investing or account fluctuations before

Finally, the Adventurous/Aggressive portfolios are best for:

1. Users with long-term goals (15+ years) who are comfortable with seeing very large fluctuations
2. Users with stable sources of income and are comfortable with larger fluctuations
3. Users investing a small portion of assets and are okay to “gamble” more
4. Users hoping to grow their money more aggressively for a medium-term goal but accepting the fact that with a market dip might require them to push back on the time horizon – goals with flexible time horizons like buying a house can fit this bill

Currently, we are still experimenting with the constraints required to create the five individual portfolios. However, at a base level, we will aim to achieve the asset allocation percentages presented in Table C above. This will either be ensured by adding the required constraints or considering risk measures such as variance and co-movement in such a way they produce those targeted asset allocations.

Tentative Questions

Finally, provided in Table D below are the questions we will ask our users. Please be reminded that both the questions and options will occur in random order in attempt to reduce survey bias.

Question		Risk-Adverse Option	Risk-Seeking Option
1	Rather:	Gain \$20 for sure	20% chance of gaining \$100, 80% chance of no gain
2	Rather:	No gain	50% chance of getting \$100, 50% chance of losing \$100







	Question	Risk-Adverse Option	Risk-Seeking Option
3	Rather	Lose \$20 for sure	20% chance of losing \$100, 80% chance of no loss
4	Which portfolio history would you prefer:	Option 1 [low vol]	Option 2 [high vol with exceeding returns]
5	Poker tournament preferred purse:	Prizes for final table	Winner takes all!
6	Bigger fear:	Portfolio performance that is consistently less than industry benchmarks	Missed investment opportunity that could have yielded higher returns
7	Bigger fear:	Loss of principle over any period of 1 year or less	Rate of inflation that exceeds rate of return long-term
8	Bigger fear:	Portfolio performance insufficient to meet financial goals	Portfolio performance that is consistently less than industry benchmarks
9	Your focus when monitoring portfolio(s):	Recent results of overall portfolio	Long-term performance of overall portfolio
10	Your focus when monitoring portfolio(s):	Investments doing poorly	Investments doing very well
11	Your focus when monitoring portfolio(s):	Investments doing very well	Recent results of overall portfolio
12	Value of portfolio drops 20%, what do you do:	Move money elsewhere immediately	Consult with advisor to ensure asset allocation/portfolio is correct
13	Range of portfolio returns:		
14	Range of portfolio returns:		
15	Range of portfolio returns:		

Table D: Tentative Risk-Tolerance Determination Questionnaire

Front-End

Finalized Wireframes

This section aims to display the finalized wireframes of the main pages required for our robo-advisory web application. Additional pages that supplement the user experience can be found in Supplement A.

As mentioned in the “Clarifications and Key Design Changes” section, our web application has two main parts – the part that is available to the public (referred to as the “main site”) and the part that becomes available after logging in (referred to as the “user site”). You can identify the main site and user site by the menu bar located on the top of the page – please refer to Figures 3 and 4 below.

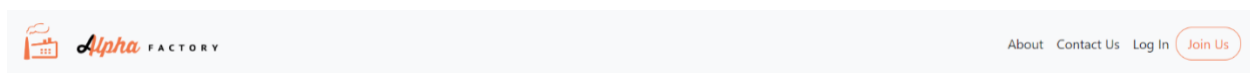


Figure 3: Main site menu bar

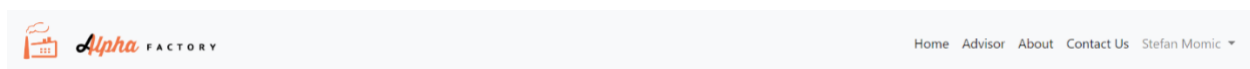


Figure 4: User site menu bar

As seen from Figure 3, visitors have access to various pages from the main site including the About page where we detail our portfolio generation strategies, asset universe selection and several other key design decisions; the Contact Us page where we allow visitors to email us any questions; and Log In / Join Us pages that allow visitors to register to our platform and access the user site.

When accessing our URL, visitors are taken to our main site landing page, as shown in Figure 5 below. This landing page is simply a part of our About page where the visitors are provided a brief overview of our objectives and goals.

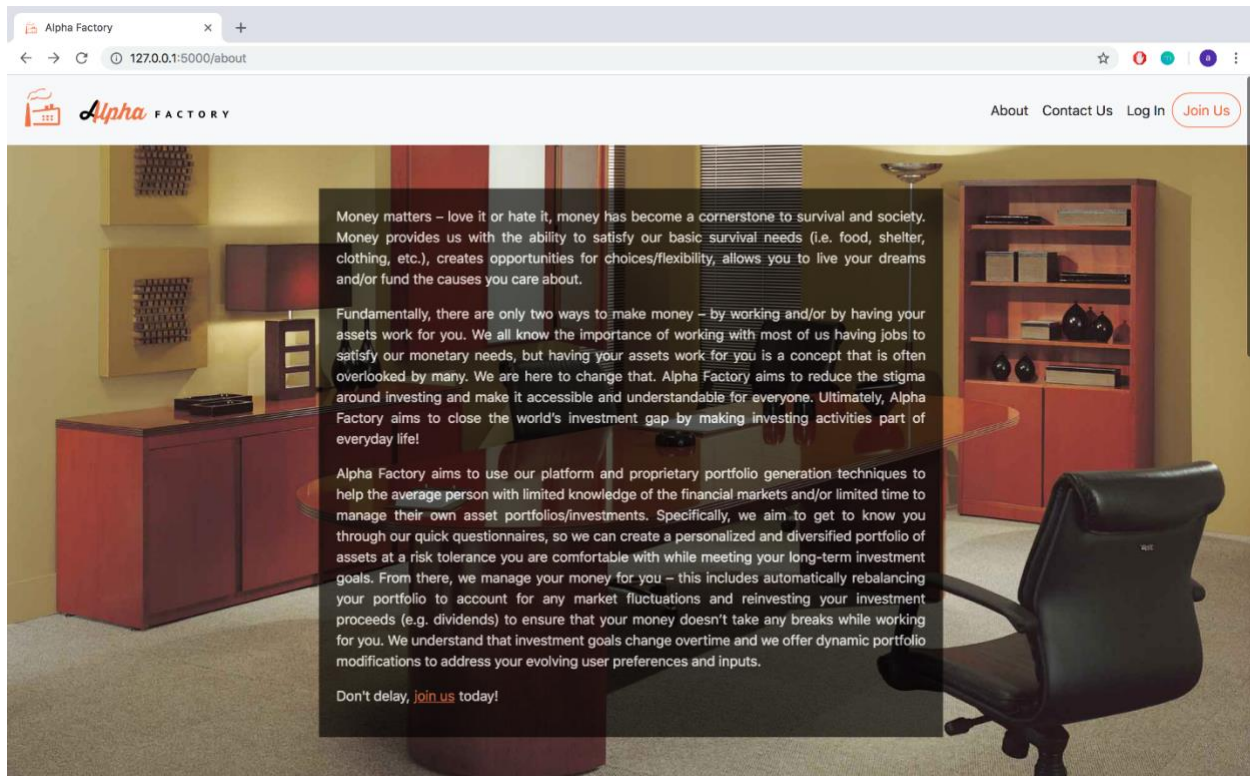


Figure 5: About Landing Page

Scrolling along this About page, we provide visitors the ability to access functions #1 and #2 (as defined in the project requirements as well as the “Clarifications and Key Design Changes” section). Specifically, they are given the list of securities in our asset universe and the ability to enter the amount of dollars they plan to invest in each asset they choose to hold, as shown in Figure 6 below.

Figure 6: Inputs for Function #2

It is noted that we choose to allow visitors to input the dollar amount they wish to invest in each asset class as we find dollar amounts to be more intuitive for our users – we will then compute the weights invested in each asset class while doing the calculations and computations. Also note that the asset classes shown in Figure 6 are not the finalized securities we plan on holding – in our final design we will list the actual names of the ETFs we chose to include. After entering the amount of money they plan to invest in each asset class and clicking “Enter”, our robo-advisor analyzes the risk and return profile of the portfolio they entered over various time horizons by backtesting over real data (fulfilling function #1) and finds a portfolio of assets that dominates the one they just entered (fulfilling function #2). The users are then displayed the results in the same area, as shown in Figure 7 below. Ultimately, by doing this, we demonstrate how much better they could be doing by using our platform (assuming the portfolio they entered is the one they currently hold or would hold if they had to invest on their own) and encourage them to join our platform using the button prompt seen in Figure 7.

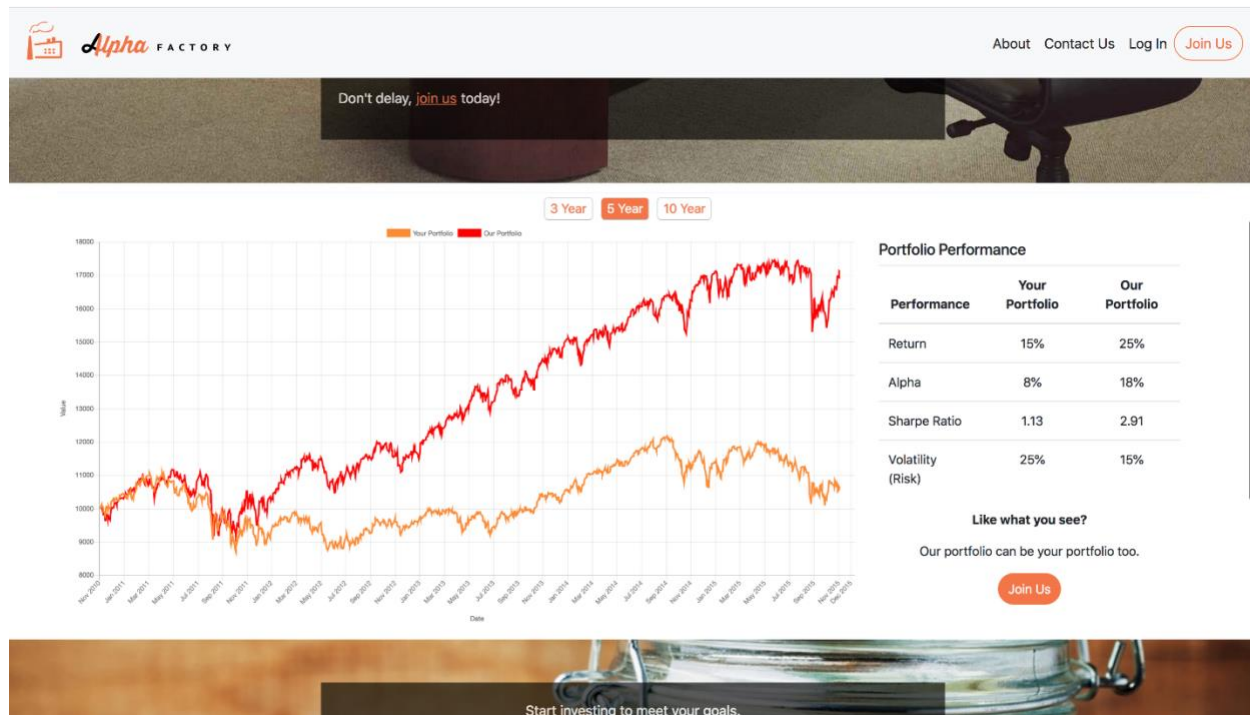


Figure 7: Outputs of Function #2 (and #1) for a selected time horizon (5 years in this case)

Having demonstrated functions #1 and #2 from the main site, our user site focuses on providing function #3 (while also incorporating function #1). After a user joins our platform and creates an account by filling out the required sign-up fields (shown in Figure 8 below), they are taken to a questionnaire aimed at assessing their return goals and risk-tolerance level over a specified time horizon, as shown in Figure 9 below.

Join Us

First Name

Last Name

Date of Birth

Email

Password


Sign up

Already a member? [Log in now.](#)


Figure 8: New user Sign Up Page

Questions

Select the option that best suits your personality.



A guaranteed \$100 return



50% chance of getting \$200

Previous

Figure 9: Risk-Tolerance Assessment Questionnaire

Portfolio Advisor

Tell us what your financial goals are and we will custom tailor a portfolio to fit your needs!

Initial investment

Financial goal

Yearly contribution

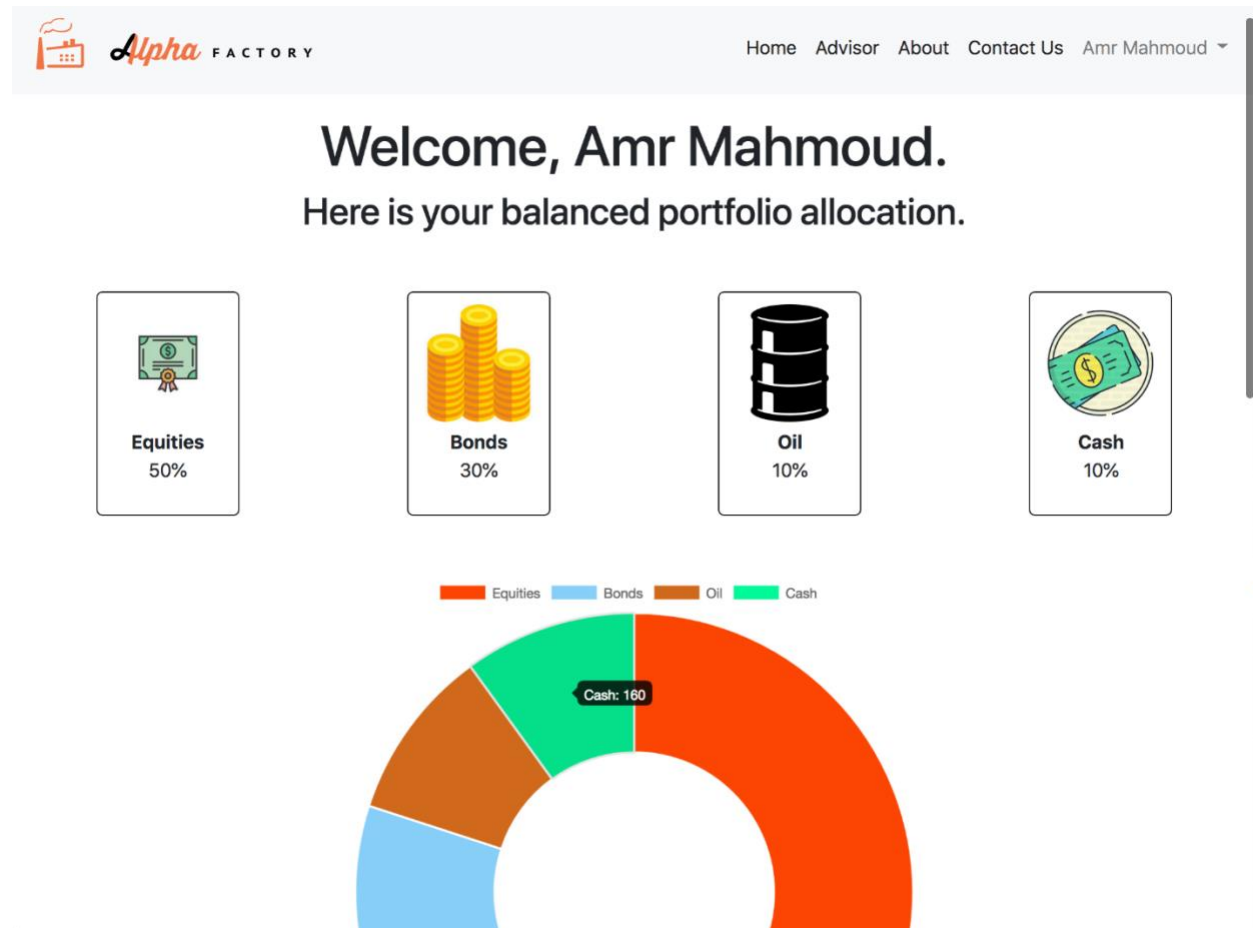
Time Horizon

16 years

Generate

Figure 10: Advisor Page with Function #3 Inputs

Using the information provided from the questions above, we suggest the optimal portfolio for the user, fulfilling function #3. We also demonstrate function #1 during this stage as well by analyzing the risk and return profile of the portfolio we created over various horizons by backtesting over real data. The computed portfolio along with its performance is displayed on the Home page of the user site, as shown in Figure 11 (split into 3 images as length of page is constraining the full view). Note again that the categories for the asset classes/ETFs are not finalized as well as the final metrics of performance seen in the image directly above the Figure 11 caption.



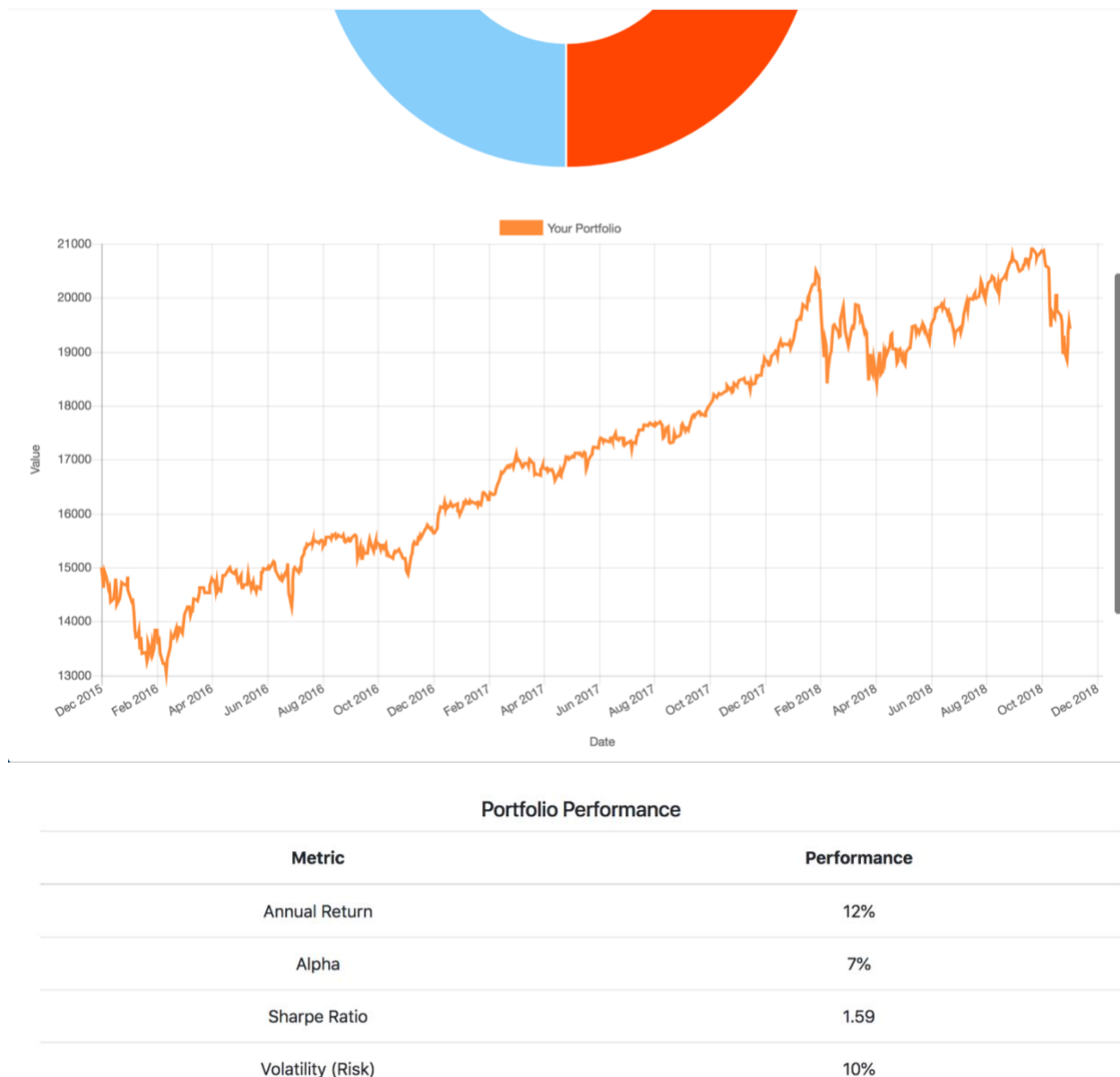


Figure 11: Home page with information and statistics on portfolio

Please refer to Supplement A for wireframes of additional pages that have not been mentioned here (e.g. Contact Us, Profile, Log In, etc.). Those pages have been omitted from the main body of the report as they are not central to the business model and flow, but have been included in the web application design as they are integral for the user experience nonetheless. Additionally, note that we focused the discussion of our wireframes to the business flow and user experience as we already considered the detailed design decisions during our Initial Design Report. For example, we simply show the risk-tolerance assessment questionnaire here as we already justified the choice of only having two options and the design features (e.g. being automatically taken to the next question after selecting the answer, inclusion of a progress bar at the top, etc.) in earlier sections of the report and/or the Initial Design Report.

BPMN Diagrams and Descriptions

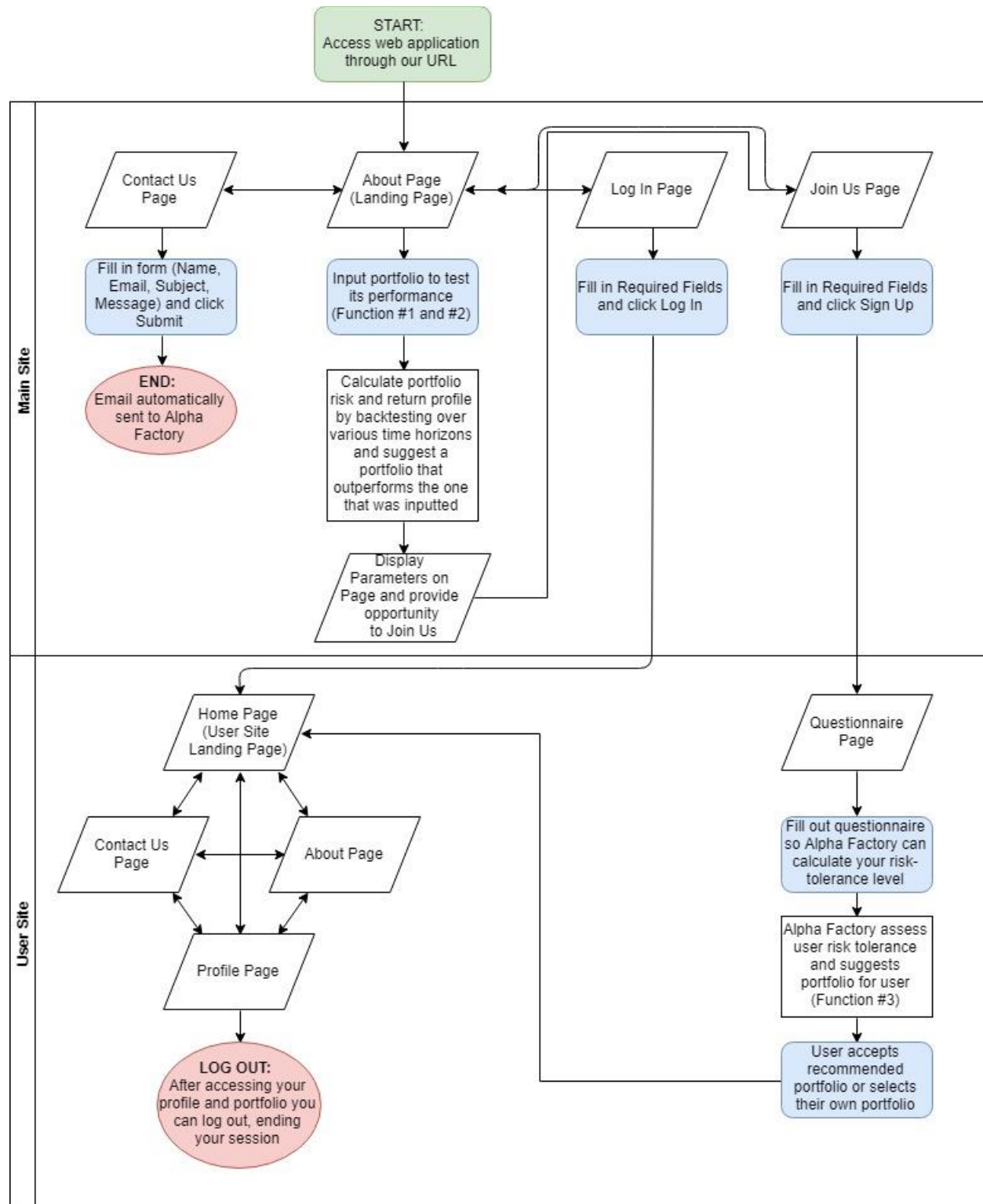


Figure 12: High-Level BPMN Diagram

A detailed original Business Process Model was already provided as part of the Initial Design. Based on the feedback received, we determined that providing a high-level diagram would be most beneficial to our stakeholders as it would enable them to obtain a better understanding of our web application flow without worrying about all the back-end nuances (please refer to the original Business Process Model provided in the Initial Design Report for more details).

Figure 12 above contains our high-level BPMN diagram. For starters, our diagram is split into two main swimlanes – the main site and the user site to distinguish between the two distinct platforms of our website.

As evident from our diagram, users can access our web application through its URL where they are taken to the About Page. From the About Page they can access the other pages using the menu bar on top of the page. These pages include Contact Us, Log In and Join Us pages. Staying on the About Page, visitors can read more about our goals, objectives, asset universe and portfolio generation strategy. They can also input their own portfolios by specifying the quantity of each asset they would like to hold. After submitting their portfolio, we calculate the risk and return profile of their portfolio and generate a portfolio that outperforms the portfolio they just submitted, fulfilling functions #1 and #2 of the project requirements. Using this as a marketing technique for our platform we ask the user if they want to join us and conveniently provide a hyperlink that redirects the visitor to our Join Us page.

The Contact Us page contains a form that the users can fill out. Once completed, the users can submit their form resulting in an e-mail being automatically sent to the Alpha Factory email account.

The Log In page asks for the user to input their e-mail and password and if it's inputted correctly grants them access to the user site. This page also contains fields such as "Sign Up" for users that have not registered yet and "Forgot Username / Password" for those users who forgot their credentials, but these nuances have been omitted from the high-level BPMN diagram for simplicity.

The Join Us page, whether accessed through the menu bar or output of function #1/2, provides the user with several fields to fill out. After filling out the initial fields, the user is taken to the user site to complete their registration. Here they are required to go through our detailed questionnaire which determines their return goals and risk tolerance level. From here, Alpha Factory recommends a portfolio for the user to hold based on their return goals and risk tolerance level, fulfilling the third, and final project requirement. Next the user either accepts the recommended portfolio or overrides it with their own choice. Having selected the portfolio, the user has completed registration and is granted access to the main landing page of the user site.

The Home Page of the user site, which is either accessed immediately after logging in or after completing the questionnaire, provides the user with information about their portfolio and its performance. From the Home Page the user can access the About Page (which contains the same information as the About Page from the main site) and the Contact Page (which only includes the Subject and Message parts of the form to fill out, as the Name and Email are already known from the user's login credentials). The user can also access their Profile Page where they can see their risk tolerance level and repeat the questionnaire / change their profile info. Having completed the session, the user can log out where they are taken back to the landing page of the main site (About Page).

Usability Testing Protocol

With project delivery being less than one month away, Alpha Factory developed an extensive usability testing protocol to ensure that their robo-advisory platform is properly tested prior to its official release. The purpose of the usability testing protocol is to document what we are going to do, how we will conduct the tests, the metrics we will capture, what scenarios will be used and the number of participants we will survey. The elements of the testing protocol include the scope, purpose, participants, sessions, scenarios and metrics, with each element being further discussed below.

Scope

The scope of our usability test includes testing both the web application for bugs and the general user experience and sentiment. As such, we will conduct extensive debugging of our web application along with a survey of participants to gauge their feel towards the web application. Specifically, we want to ensure that our web application is easy and intuitive to use, the risk questionnaire is sufficiently accurate, the user is likely to use our platform again, and that there are few, if any, web application bugs/glitches.

Purpose

The purpose of our testing protocol is to ensure that users can navigate the web application and gauge how they feel about the web application. We are ultimately striving for a very simple and user-friendly design, hence making most of our design decisions with that in mind (e.g. we wanted to ensure that our risk questionnaire was short and easy to fill out ultimately deciding to limit the number of questions to 15 and only allowing two options for each question). At the same time, we wanted to make sure that our web application produced accurate results and correctly identified users return goals and risk tolerance levels.

Participants

Alpha Factory will test its application by surveying a random sample of 20 University of Toronto Engineering Finance Association (UTEFA) members. UTEFA members were selected as the participants as they are largely composed of undergraduate engineering students who are interested in finance. As such, it is believed that their feedback will be extremely valuable for future iterations and improvements as they can comment on both the computer programming aspect, through their engineering background, and the investment aspect, through their background/interest in finance.

Alpha Factory will also attempt to survey a random sample of 20 students in Sidney Smith hoping to obtain feedback from students who have an unknown level of financial and computer literacy. With the belief that these survey participants are representative of the general population, we will mainly be focusing on whether or not these students find our platform easy and intuitive to use.

Sessions

Survey sessions will be conducted with a 10-minute time limit. This decision was made for two reasons. First, we are respectful and conscious of our participants' time and wanted to ensure our surveys intrude as little as possible. Second, the ease of use design factor was one we greatly valued and focused on while making our design decisions. As such, we believe that our users should be able to set up an account in 10 minutes or less! To ensure this is the case, we will limit survey sessions to 10 minutes and gauge the web application usability and engagement accordingly.

Scenarios

Apart from the unique scenarios our randomly selected survey participants generate, we have come up with the following full functionality-encompassing user scenarios which omit the nuances of backend/DB processes for simplification:

#	Scenario	Action	Expected Result
1	New/Existing User accesses the site to browse for information and contact us for more details	<ol style="list-style-type: none">1. Access the site through the URL2. Browse the landing/About page3. Explore all the menu links4. Access the Contact Us page through the menu link5. Fill out contact form and send email	<ol style="list-style-type: none">1. User is shown landing page/home page depending on the logged in status2. User is able to access the About page with the corresponding information displayed3. User is able to freely access all parts of the main site/user site depending on their logged in status4. User is able to access the Contact Us page and the contact Form is displayed5. User is able to fill out form and send email successfully (receipt of corresponding email and contents must be confirmed). User is shown a confirmation of receipt screen

#	Scenario	Action	Expected Result
2	New user trying out our platform through Function #2 and deciding to join us	<ol style="list-style-type: none"> 1. Access the site through the URL 2. Scroll to the Function #2 form. 3. Fill out form with corresponding information 4. Review results and select Join Us 5. Fill out sign up form 6. Answer risk assessment questionnaire 7. Input financial goals 8. Review generated portfolio 	<ol style="list-style-type: none"> 1. User is shown landing page 2. User is able to see Function #2 form 3. User is able to fill out correct information in form and click Enter 4. User is returned a comparison of results of the two portfolios over several different time horizons. User is able to click 'Join Us' button in the same section or menu link 5. User is able to fill out sign up form and click Sign Up 6. User is shown the correct questions of the risk questionnaire with a progress bar and back/'previous' question button.(all must be correct and working) 7. User is shown the advisor page with the ability to enter their financial goals into the form and generate a portfolio. 8. User is redirected to the Home page with portfolio information and statistics displayed

3	Existing user logs in to access their portfolio and change their risk tolerance	<ol style="list-style-type: none"> 1. User accesses the site through the URL 2. User chooses to Log In from the menu link 3. User fills out log in form and clicks Log In button 4. User accesses the Home page with portfolio information displayed 5. User accesses the profile page through the menu link 6. User selects edit information 7. User selects to manually change risk tolerance 8. User agrees to the prompt 9. User fills out questionnaire 10. User is able to see new risk tolerance 	<ol style="list-style-type: none"> 1. User is shown landing page 2. User is able to navigate to the log in page and see the form 3. User is able to fill the form and select log in 4. Upon verified credentials, user is directed to the Home page with the correct corresponding portfolio information and statistics displayed 5. User is able to access and view profile Page and their corresponding information 6. User is able to click edit button to change their profile information 7. User is able to select to change risk tolerance and is prompted a confirmation alert informing them they must answer a questionnaire 8. User is taken to the questionnaire page and the corresponding questions are displayed 9. User is able to easily navigate through the questionnaire 10. User is led to Home page on completion with the new risk tolerance (and portfolio) updated in the db and changes reflected in profile page
---	---	---	---

#	Scenario	Action	Expected Result
4	Existing user forgot their password and wants to reset it	<ol style="list-style-type: none"> 1. User accesses the site through the URL 2. User clicks the Log In page 3. User clicks the 'Forgot Password?' link 4. User fills the reset password email form and selects 'Reset' 5. User types in code from received email and fills out the reset password form 6. User clicks 'Confirm' 7. User logs in with the new password 	<ol style="list-style-type: none"> 1. User is shown landing page 2. User is directed to the Log In page with the log in form displayed 3. User is directed to the forgot/reset password page 4. User is able to fill out form and select 'Reset'. Email is sent to user with confirmation code included 5. User is able to fill out the reset password form and validation of variables is expected to work correctly 6. User information is updated in the db and user is directed to log in page 7. User is able to log in with the new credentials and is directed to the home page

Metrics

As mentioned previously our testing protocol aims to test three main things.

First, we hope to ensure that the web application is functional without any bugs. As such we will monitor our tests for any bugs that arise during our surveys and/or scenario test cases and work to rectify any and all issues prior to the December 3rd deadline.

Second, we want to make sure that our web application is user-friendly and intuitive/easy to use. To do this, we will look at the length of time it takes our users to set up. If a large majority of the users (more than 80%) are able to set up their account and obtain a customized portfolio in under 10 minutes, we can conclude that we met our ease of use target relatively well. We will also survey session participants asking for their thoughts on the user experience and general feel of the web application. Questions to be asked will include:

- Ranking their user experience on a scale of 1 – 10 with 10 being preferred
- Ranking the ease of understanding questions on a scale of 1 – 10 with 10 being preferred

- Ranking the intrusiveness of the questions on a scale of 1 – 10 with 1 being preferred
- Ranking the likelihood of using the service on a scale of 1 – 10 with 10 being preferred

By cumulating all of the survey results we will conclude on the general user sentiment of our web application and attempt to improve on the user feel based on the feedback received.

Finally, we hope to test the accuracy of our risk-tolerance questionnaire and portfolio recommendation strategy. To do so, we will proxy how accurate our risk-tolerance level assessment is by seeing how accurately it predicts the surveyed individuals' self-identified risk-tolerance level. To do this, we will ask participants to select a risk-tolerance level from one of the five discrete options prior to filling out the risk questionnaire. We will then gauge the accuracy of our risk questionnaire by determining whether or not our platform can accurately predict our user's self-identified risk tolerance level.

Note, that we will continuously gauge the accuracy of our risk questionnaire even after releasing the platform. To do this, we will assess the percentage of users who accepted the recommended portfolio. For example, if a user is identified to be risk-neutral they will be recommended to hold our Balanced portfolio. If this user accepts the Balanced portfolio, we will interpret this as accurately predicting their risk-tolerance level. However, if the user chooses to invest in another portfolio, we will interpret this as not being an accurate prediction of the user's risk-tolerance level and store their risk-tolerance level and portfolio of choice in our database. If we observe a trend of a select score favouring a particular portfolio, we will modify our risk-tolerance level assessment logic to start classifying users accordingly.

Back-End

Description of Data Usage

To use the financial data from the previous section in Alpha Factory's business processes and portfolio generation, we are taking that data and storing it in our own database, which was stated previously to be MongoDB. In storing the data for the different items in our asset universe, the format of each asset's dataset is expected to be similar and in addition to similar formats, with each asset being its own 'object', any interactions between the assets can be done seamlessly. (ie. finding different ways to track the relation of different assets over various different horizons for the purpose of better predicting the movement of our asset universe as a whole.)

After having located the first set of assets that were considered for the asset universe, their data was pulled from the Bloomberg terminal as CSV files. For ease of importing, these CSV files were then converted to JSON documents, a widely accepted file format for MongoDB and other Document-Stored database systems. From there, each individual file was imported into its own collection in Mongo, with each individual document representing the data for one trading day. The most important values in each document is the date and the adjusted closing price, while the opening price, closing price, and trading volume were also imported.

The inclusion of the raw data of 32 ETFs and the SPX index from Bloomberg into MongoDB amounted to a total data usage of 17 MB, with the largest number of documents belonging to the collection of the SPX

index dating back to 1950 for a total of over 17,300 data points, while some of the more recent ETFs only have information for the last decade, with approximately 1,000 data points. Using one of the collections as a sample, the number of documents and size of the collection was used to determine that the average size of one data point for one asset was approximately 0.13 KB. Thus, given the current space occupied by the data in storage, our choice of MongoDB as a database management service is more than adequate for dealing with the addition of more assets like indices that could be added to proxy the values of the ETFs for which there may be less data than we would like.

In addition to the raw data, the computed data related to each asset needs to be stored on our databases as well, such as parameters for returns, volatility and correlation, as well as the data pertaining to these assets in the various portfolios that get created through Alpha Factory's business logic and processes. Once again, the choice to use an object-oriented database, and MongoDB in particular, allows a very straightforward organization for this data. The detailed plan for how these objects are going to be set up can be seen below in the Database Schema section. We expect that the data usage for calculated values will be much lighter, as there are significantly fewer values when compared to the large number of data points that went into creating them.

For the raw data, the main objective is simply to ensure that all the data needed for estimating the parameters of each asset is readily available and can be called upon when these values need to be used or updated. Within the overlying object for each asset, the raw data will be contained in another object type that is then embedded in its asset's structure. Lastly, as the calculated values are also a part of each asset's object, they are also included, but in such a way that the data and computed values are two separate objects that have a chance to interact with one another. The next section covers the in-depth schema for the database and the organization of collections based on their intended use.

Database Schema with Descriptions

As explained above, there are two components to the data that is being stored in MongoDB. The first is the raw data, which includes the pricing data for all the assets in our asset universe for the time periods that we were able to secure them for. The data is taken for each trading day, and all relevant data is transferred into MongoDB by first downloading the data, and then importing it manually, as was outlined in the Initial Design Report as one of the conditions for using the Bloomberg terminal as our chosen source of data. Thus, the resulting data format in MongoDB is given below with:

```
SampleETF_Prices : Collection
{
  Document
  {
    ID : ObjectID    "Unique ID for this document"
    Date : Date      "Trading date for all values"
    Open : Double    "Opening price of the asset"
    High : Double    "Highest intraday value of the asset"
    Low : Double     "Lowest intraday value of the asset"
    Close : Double   "Closing price of the asset"
```

```

Adj Close : Double "Adjusted closing price of the asset"
Volume : Int      "Trading volume of the asset"
}
...
}

SampleIndex_Prices : Collection
{
...
}

```

In addition, the values that can be calculated from these different collections would then be compiled into another collection that includes the relevant values for each asset, and the asset universe as a whole:

```

Assets : Collection
{
  Document
  {
    ID : ObjectID    "Unique ID for this document"
    Number_Assets : Int    "Number of Assets in the Universe"
    Risk_Free : Double    "The 'risk-free' rate used to find risk premiums"
    Market_Vol : Double    " $\sigma_M$ , the volatility of the asset universe"
  }
}

```

```

Document
{
  ID : ObjectID    "Unique ID for this document"
  Ticker : String   "Ticker of the asset"
  Return : Double   "Calculated return of the asset"
  Volatility : Double   "Standard deviation of the asset"
  Start_Date : Date   "Date of first data point in the database"
  Sharpe : Double    "Sharpe Ratio of the Asset"
  Beta : Double      "Beta for the Asset"
  VaR : Double       "VaR of the Asset"
  CVaR : Double      "CVaR of the Asset"
}
...
}

```

The second component is the data that is created and then manipulated within Alpha Factory itself. This includes the users and generated portfolios. The data stored for the users is under 3 different

collections. The first is for all the user information, which includes the data they enter when creating their profile and the data that gets assigned to them in the application, like risk tolerance and the portfolio(s) they hold. The second collection acts as a user log and stores all changes that occur to the data in the first collection. The last is to store the questions that are given to the user in the questionnaire to determine their risk tolerance.

While the questionnaire data collection is not expected to grow (ie. there will only be a constant 15 questions, even though Alpha Factory can choose to change the contents of each whenever they wish), the Users and User_History collections are expected to continue growing. Keeping in mind that MongoDB provides the first 512 MB of their services free, the concern lies in the ability for the database to outgrow the constraints of the provided free services. It was calculated that, with 7 users in the Users collection taking up approximately 1.65KB of data, each user document takes approximately 0.25 KB of data. The user history document, on the other hand, can be found to take up around 125B per user transaction. With this information, combined with the knowledge that a user must have at least 3 historical items attached to them (account created, risk tolerance assigned, portfolio generated), we can see that the minimum size one user will take up on the database is 0.625 KB. Ignoring all other DB storages, we can find the crude upper limit on the number of users Alpha Factory can have without paying for extra features from MongoDB as $512000/0.625 = 819,200$ users max. For the purposes of this capstone project, there will never be a time where over 800 thousand users are needed but if Alpha Factory wishes to expand their services to the general public, the limited capabilities of the free services will likely be inadequate for production performance.

To follow up on the previous points, the first data collection is set up as seen below:

```
Users : Collection
{
  Document
  {
    ID : ObjectID    "Unique ID for this document"
    First_Name : String    "User's first name, provided on creation"
    Last_Name : String    "User's last name, provided on creation"
    Email : String    "User's email, provided on creation"
    DoB : Date    "User's date of birth, provided on creation"
    Filled_question : Boolean "Has the user filled the questionnaire?"
    Risk_Tol : Int    "The risk tolerance assigned from the questionnaire"
    Portfolio : Object "Portfolio held by the user"
      Portfolio_ID : ObjectID    "ID of the portfolio"
      Time_Horizon : Double    "Investment Horizon"
      Capital : Double    "Value invested into the portfolio"
  }
  ...
}
```

The other user collection, containing the info log for all users, is included below:

```
User_History : Collection
{
  Document
  {
    ID : ObjectID    "Unique ID for this document"
    User : ObjectID  "Unique ID of the relevant user"
    Date : Date      "The date of the action"
    Action : String   "The message for the action"
    Result : String   "The result for the action"
  }
  ...
}
```

Finally, we have the questions in the questionnaire:

```
Questions : Collection
{
  Document
  {
    ID : ObjectID    "Unique ID for this document"
    QuestionID : Int  "Chronological order of the question"
    Question : String "Wording of the question"
    Optiona : String  "First option to answer question"
    Optionb : String  "Second option to answer question "
  }
  ...
}
```

Lastly, we have the collection for the portfolios created by Alpha Factory's business logic processes, and the data for the weights and performance metrics of each portfolio is stored in a separate collection for the purpose of being accessed and displayed to appropriate users efficiently.

```
Portfolios : Collection
{
  Document
  {
    ID : ObjectID    "Unique ID for this document"
    Risk_app : String "Appropriate risk appetite for the portfolio"
    Risk_tol : Int    "Numerized risk value for the portfolio"
    Return : Double   "Calculated return of the portfolio"
    Volatility : Double "Standard deviation of the portfolio"
```

```

    Sharpe : Double    "Sharpe Ratio of the portfolio"
    Beta : Double    "Beta for the portfolio"
    VaR : Double    "VaR of the portfolio"
    CVaR : Double    "CVaR of the portfolio"
    Weights : Array    "Array of weights for the portfolio"
    Description : String    "Description of the portfolios asset class/holdings"
  }
  ...
}

```

Updated Project Plan and Timeline

As presented in the Initial Design Report, the official launch and presentation of Alpha Factory will occur on December 3rd, 2018 from 3-6pm in room 440 of the Myhal Centre for Engineering Innovation & Entrepreneurship at the University of Toronto. The Final Report and documentation of Alpha Factory will be released shortly after, but no later than 5pm on December 5th, 2018.

To ensure timely delivery, Alpha Factory has imposed an internal deadline of November 29th, 2018. Currently, they aim to have a draft of the Final Report and complete robo-advisor platform by that time. To meet these deadlines, the team has laid out the following timeline with the corresponding milestones.



Figure 13: Project Timeline

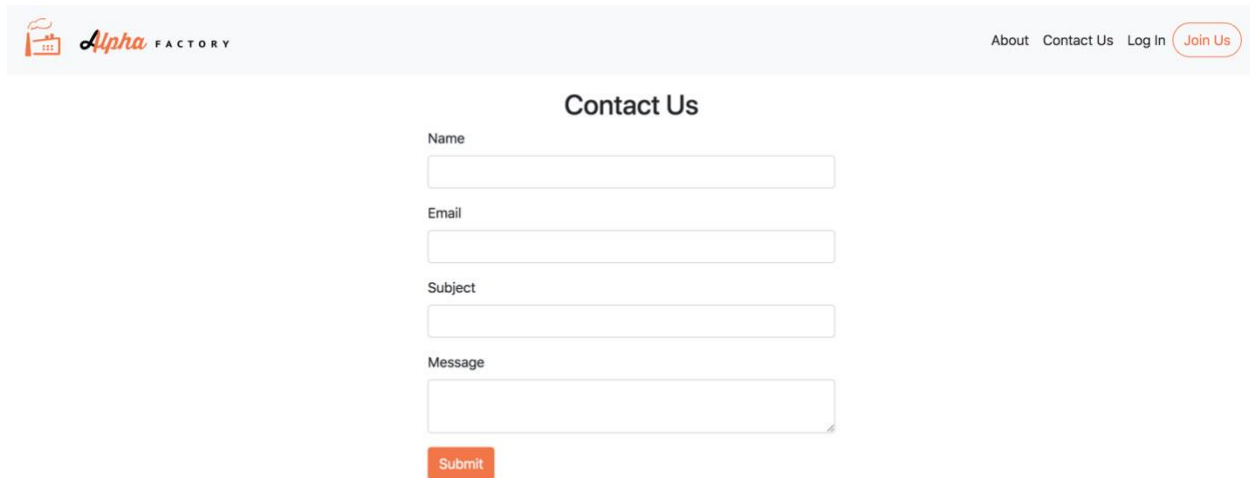
Additional notes regarding select milestones from Figure 13 above are provided in Table E below.

Item	Detailed Description	Status
Oct 15: Business Logic complete	Completion of asset selection and financial data location	COMPLETE
	Completion of portfolio generation algorithm	IN PROCESS
	Completion of parameter estimation	IN PROCESS
	Complete justification for all design decisions made including asset selection, portfolio generation and parameter estimation	COMPLETE
	Thorough understanding of all models and methodology used completed as part of Literature Review	COMPLETE
Oct 27: Front and Back-End complete	Ability to gather user inputs and feedback	COMPLETE
	Ability to hook up to Business Logic and display computed data (e.g. graphs, tables, numbers, etc.)	IN PROGRESS
	Complete user-friendly design and user interface	ALMOST COMPLETE
	Ability to clean data and/or deal with missing data	COMPLETE
	Database selection and initialization including the ability to store raw and computed data from the financial data selected and generated by Business Logic	COMPLETE
Nov 17: Integration between Business Logic and Front-End / Back-End	<ul style="list-style-type: none"> • Compute and present the final outcome (e.g. optimal portfolio given the user inputs) • Perform all the required analytics (e.g. backtesting, portfolio performance metric calculations, etc.) • Validate portfolios generated by the system 	IN PROGRESS

Table E: Detailed descriptions for select milestones

Although Alpha Factory has not fully completed any of their subsystems, there has been significant progress in each subsystem with most having completed significant portions of their requirements and initial subsystem integration underway. This should not be seen as a cause for concern as we immediately discovered the value of working together and began integration as soon as possible. Rather than working individually to complete the various subsystems and worrying about the integration at the end (which could be a very difficult process), we decided to begin integration right away, eliminating the need to do so at the very end. Table E has a view of the status for each milestone with the next steps being easily summarized as continuing to develop the items deemed as “IN PROGRESS” until they are “COMPLETE.” A more detailed view of the specific next steps for each subsystem has already been provided in the “Subsystem Accomplishments and Next Steps” section.

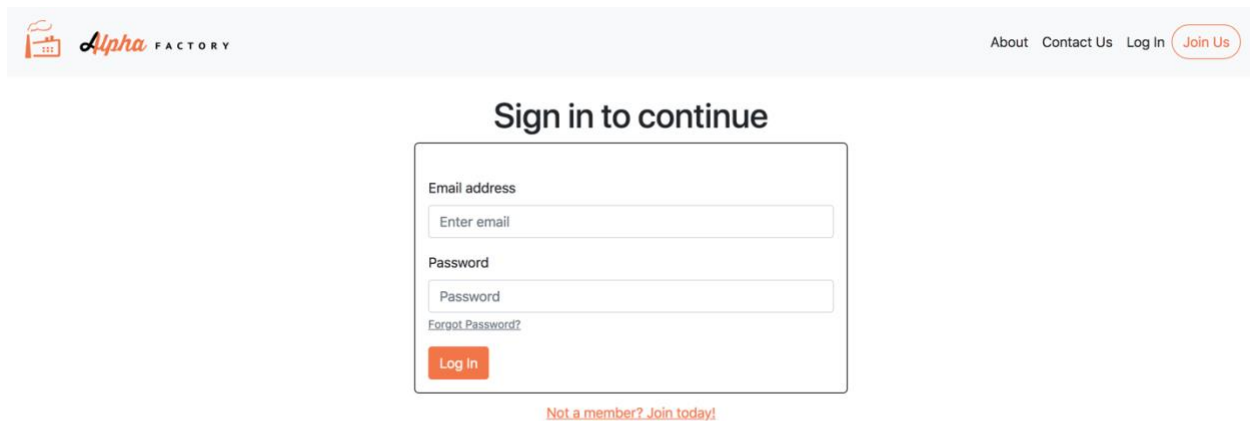
Supplement A – Additional Wireframes



The wireframe shows a 'Contact Us' page. At the top is a header with the 'Alpha FACTORY' logo on the left and navigation links 'About', 'Contact Us', 'Log In', and a 'Join Us' button on the right. The main heading 'Contact Us' is centered. Below it are four input fields labeled 'Name', 'Email', 'Subject', and 'Message'. The 'Message' field is a larger text area. At the bottom of the form is an orange 'Submit' button.

Figure A1: Contact Us Page

Accessible through the Contact Us button in the menu bar for both the main site and user site, the contact us page is available for everyone to access. By filling out the form completely, an email is automatically sent to the Alpha Factory Gmail account for our members to review. Upon completion, the user is notified that the message has been sent and that we will be in touch with them shortly.



The wireframe shows a 'Sign in to continue' page. At the top is a header with the 'Alpha FACTORY' logo on the left and navigation links 'About', 'Contact Us', 'Log In', and a 'Join Us' button on the right. The main heading 'Sign in to continue' is centered. Below it is a login form with two input fields: 'Email address' (with placeholder text 'Enter email') and 'Password'. Below the password field is a link 'Forgot Password?'. At the bottom of the form is an orange 'Log in' button. Below the form is a link 'Not a member? Join today!'.

Figure A2: Log In Page

The Log In page is available through the Log In menu link on the main site and directs the user to a login form. The user can enter their credentials and, after successful validation through the backend system, the user will be able to access the user site. If the user does not have an account, they can be directed to the Join Us page through either the 'Not a member? Join today!' link or the Join Us button in the menu bar. If the user has forgotten their password, they can select the 'Forgot Password?' link and they will be directed to the Reset Password page below.

Reset Password

Enter your email address to reset your password.

Enter email

Reset

Figure A3: Reset Password Page (forgot password)

If the user wishes to reset their password, they may enter their email and a verification code will be sent to the user's email. The user will then be prompted by the form in the page below to enter the code they received from their email. They can access/navigate away to the main site through the menu bar.

Enter your code and new password to complete the reset.

Enter the 6 digit code sent to your email.


Enter your new password

Confirm your new password

Confirm

Figure A4: Reset Password Page continued

After the user has received their verification code, they can enter the code as well as a new password into the fields above. If the verification code and the confirmed/new password match up, the user's information will be updated in the backend and they will be directed to the log in page where they can sign in with the new credentials. If the user chooses to, they may access their account information through the menu bar as shown below.



Name: Amr Mahmoud
 Email: amrmahmoud.am@hotmail.com
 Date of Birth: 2018-10-04
 Risk Tolerance: very risky

Change Password

Edit Info

[View Profile](#)
[Sign Out](#)

Figure A5: View User Profile Page

The user has the ability to access and edit the information related to their account through the profile page, which can be accessed through the menu bar under the user full name -> 'View Profile' link. On the page, the user can select to change their email,

password, and risk tolerance. In each case, the user will be prompted to confirm their decision. In the case of a user wishing to change their risk tolerance, they will be required to fill the questionnaire again. It is also possible for the user to sign out through the 'Sign Out' menu link displayed in Figure A5 above. Clicking on sign out will lead the user back to the landing page of the main site and end their session.

Appendix – Tearsheets

References

- [3] Bridgewater. “The All Wealth Story – How Bridgewater Associate created the all weather investment strategy, the foundation of the “risk parity” movement.” <https://www.bridgewater.com/resources/all-weather-story.pdf>
- [7] Fama French “The Cross Section of Stock Returns” (1992)
<https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1992.tb04398.x#>
- [4] Jagannathan, R., T. Ma. 2003. Risk reduction in large portfolios: Why imposing the wrong constraints helps. *J.F inance* 58(4) 1651–1684.
- [6, 10, 11] Portfolio optimization: A general framework for portfolio choice, Resolve Asset Management (<https://investresolve.com/file/pdf/Portfolio-Optimization-Whitepaper.pdf>)
- [12] *Quantitative Approach to Tactical Asset Allocation*
(https://papers.ssrn.com/sol3/papers.cfm?abstract_id=962461)
- [9] Risk Parity Solution Brief, Resolve Asset Management (<https://investresolve.com/risk-parity-solution-brief-2/>)
- [8] Simin, Timothy, 2008. The poor predictive performance of asset pricing models. *Journal of Financial and Quantitative Analysis* 43, 355–380.
- [2] “System Properties Comparison Amazon DynamoDB vs. CouchDB vs. MongoDB”, <https://db-engines.com/en/system/Microsoft+SQL+Server%3BMongoDB%3BMySQL%3BPostgreSQL>
- [1] “System Properties Comparison Microsoft SQL Server vs. MongoDB vs. MySQL vs. PostgreSQL”, <https://db-engines.com/en/system/Microsoft+SQL+Server%3BMongoDB%3BMySQL%3BPostgreSQL>
- [5] <https://www.aqr.com/Insights/Research/Journal-Article/Investing-With-Style>