Technische Universität München

Lehrstuhl Für Finanzmanagement Und Kapitalmärkte

Portfolio Builder System: A Markowtiz Efficient Frontier Based Application

Developer:
Oliver Xavier Baptista

Supervisor:
Patrick Bielstein

February 25, 2015

Chair of Financial Management and Capital Markets

Contents

1 Introduction

Harry Markowitz coined the idea of mean-variance portfolios in 1952 [1], theorizing that an investor can either maximize the expected return for a given level of risk, or minimize the risk for a given expected return. These principles can help the investor to compute various portfolios known as the efficient frontier of portfolios.
The goal of this interdisciplinary project is to develop a web based application to calculate this efficient frontier of portfolios for various assets like stocks, bonds, commodities etc. traded on various exchanges around the world.

1.1 Markowitz Portfolio Theory

When Markowitz published his paper, he provided the foundation for modern portfolio theory as a mathematical problem [2].
The return R_t of a portfolio at time t can be defined as the ratio of the total value T_t of the portfolio to the total value at an earlier time *t - 1*, i.e.

$$R_t = \frac{T_t}{T_{t-1}} - 1$$

Markowitz portfolio theory analyzes how good a portfolio is based on only the mean and the variance of the returns of the assets in the portfolio.

1.2 Portfolio selection as an optimization problem

An investor is supposed to be risk-averse, i.e the investor wants high expected returns for a small risk (i.e. a small variance of the return) [3]. Markowitz portfolio theory states that an investor should choose a portfolio from the efficient set, depending upon his/her risk aversion. One way to handle this is to consider the optimization problem. Here we try to minimize:

$$\frac{A}{2}[[W]^T [C] [W] - [W]^T [R]]; \text{ subject to } |[w]| = 1$$

Where A is risk aversion factor, C is the matrix of co-variances, R are the expected returns and W are the weights of individual assets.

1.3 Problem Statement

Given various assets like stocks, bonds, commodities etc. traded in different currencies, the application should convert all assets to a common currency and calculate an efficient frontier of portfolios for the given assets. Once all the portfolios are generated with different risk aversion factors, the performance of these portfolios should be tested for a period of 1, 3 and 5 years from the date of portfolio creation.

1.4 Motivation

As the number of financial assets is ever increasing in today's market, investors are exposed to greater opportunities and risks. Since these assets have different risk return characteristics, there is a need for a mechanism that can select assets in appropriate proportions to create a portfolio. Since Markowitz portfolio selection model laid the foundation for modern portfolio theory and many sophisticated models for portfolio creation have been developed on the basics of Markowitz portfolio theory, it is crucial to study and understand the Markowitz model.

1.5 Objective

The goal of the project was twofold:

1. Understanting and implementing the Markowitz Efficient Frontier.
2. A user friendly web based application for building and analyzing the portfolios created by the Markowitz Efficient Frontier.

1.6 Methodology

To implement the above goals , the following methodology needs to be followed:

1. Specifying the application and various components of the architecture.
2. Identification of all components and tools required.
3. Specifying the dependencies between the resources.
4. Extracting the data required for building such a model.
5. Building a robust model satisfying the criteria.
6. Evaluation of the model.

2 Project Management

2.1 Planning

As with any software development task, planning played a major role in the development of the project. The development model chosen for this project was the waterfall model.

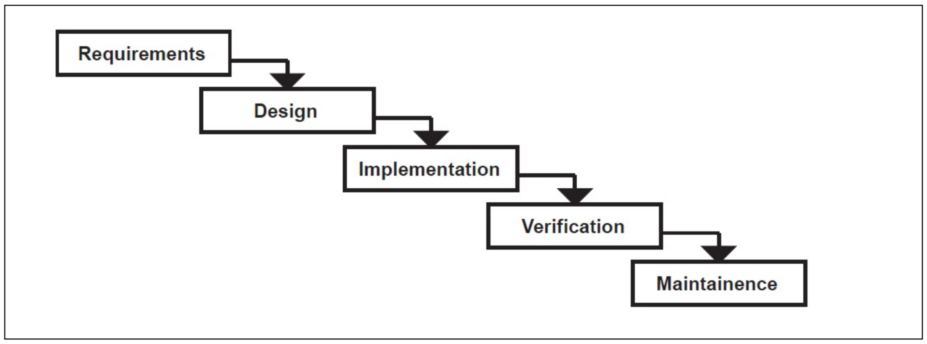


Figure 1: Waterfall development model.

2.1.1 Requirements

During this phase, reasearch was conducted on what is the purpose and the functionalities of the application. In this stage, the reqirement from the supervisor were collected and understood. Understanding of each reqirement was validated with the supervisor so as to be on the same page. As per the requirements, the appropriate software needed for the implementation of the application was analyzed. This stage is critical since we decided on the very basics of the application right from the language, patterns for designing the software to the type of database system needed.

2.1.2 Design

In this phase the design of the application and the program structure was created. This phase forms the backbone for the acutal implementation process. Proper planning related to design of user interfaces, flowcharts etc. was performed here. The wireframes shown in figure 8 of the Appendix section, were discussed with the supervisor. Feedback from the supervisor was noted and some basic modifications were carried out.
The first aspect of the design stage was to distribute the objectives into a number of tasks and subtasks. Approximate durations were assigned, taking into account their difficulty and any other concurrent commitments such as exams and coursework. A number of milestones were then identified by analysing the project deliverables.
The task list was then used to produce a Gantt chart, which can be found in the figure 6 of the Appendix section. This gave a clearer breakdown of the tasks and allowed for them to be easily scheduled. The Gantt chart played a crucial role in time keeping of the project.

2.1.3 Implementation

Based on the feedback, the design was improved and the actual coding was carried out at this stage All the functions were developed independently and later integrated.

2.1.4 Verification

The application designed, was subject to various tests and error correction processes to find out any flaws in the implementation or the design. Any test errors that were found were fixed and the whole process was reiterated.

2.1.5 Maintainence

In this phase the developed application was beta tested and any functional requirements that were not fulfilled were implemented.

2.2 Management

A folder structure was maintained to keep backups on an external hard disk along with the use GitHub repository. GitHub maintains a web-based repository of the codebase and provides the revision control and source code management (SCM) functionalities, using which we can minimize the impact of lost work (drive failure, theft, etc). It also provides previous working versions should any faulty code need reverting to a previous state. The repository can be accessed at https://github.com/echorosso/Markowitz .
Coding standards were adhered to during the development of the project, keeping uniform layouts and styles throughout all source files to increase readability. All variables and functions were designed to follow the CamelCase naming convention.

3 Application Architecture

Earlier most of the applications used to be single tier applications; a centralized mainframe used to contain the presentation, business and data logic all interwined as a single monolitic application. Making changes to any part of the application was a very diffcult and cumbersome task. Over the period of time the applications have evolved to a much more flexible multi-tier models. We have developed our application on the Spring Application Framework since it is one of the stable and robust multi-tier architecture widely used today.

3.1 Why Spring Application Framework ?

Spring is a layered JAVA/J2EE application framework which has following key features:

- A **lightweight container**, which provides centralized, automated configuration and wiring of our application objects. The container provides increase application testability and scalibility by allowing us to develop individual components. These individual components can be tested in isolation and then scaled up.
- Robust **transaction management**, which allows us to configure various transaction managers without needing to deal with the low-level issues.
- A **JDBC abstraction layer**, which simplifies SQL error handling and minimizes the need for writing block JDBC codes.
- Flexible **MVC framework**, which is highly configurable via interfaces and can accomodate multiple view technologies like JSP, Tiles etc.

4 Software Development

4.1 Implementation

The application is built using the Spring Application Framework a J2EE MVC technology and is deployed on Apache Tomcat v7.0 web server. The historical asset data is retrieved using YAHOO! Finance API.
Technologies like JFreeChart API, HyperSQL Database and ojAlgo API were also used for the development. All these are open source technologies and have very active communities.

4.2 Application Design

The figure 7 of the Appendix section, depicts the flowchart of the process that is invoked when the user selects a Market index whose components needs to be fetched.
The figure 9 of the Appendix section, depicts the process of creating the Markowtiz Efficient Frontier when a set of assets are selected by the user.
Please refer to *Software Design Specification* document provided, it gives a much detailed and thorough understanding about the systems's architecture, database design and implementation. The *Sequence Diagrams* document provided will help to thoroughly understand the internal workings of the application. These two documents are critical to getting a complete understanding of the entire application.

4.3 Deployment

To deploy the application, please read the *Deployment Document* provided.

5 Testing

To ensure the software functioned without any issues and all the requirement specifications were met, various methods of testing were performed at each stage of the development cycle.

5.1 Unit and Integration Testing

Unit testing is the process of ensuring the individual components/ functionalities of the application work as intended. This meant that each individual functionality of application could be tested in isolation without interference of any other component. That being the case, we could account for all the possible behaviours of the application.
When the various components were brought together, they were integration tested. This was to ensure that each individual component retain its original behaviour while functioning together as an application. Majority of work involved during integration was to replace appropriate components in place of the proxies and stubs and runtime analysis of the the internal state of each object.

5.2 Requirement testing

Another continuous process that was performed throughout the life of the software development process was requirement testing. This concerned whether or not the software was satisfying the requirements outlined at the start of the project. This kept the development in process and allowed us to determine when the final system was at an acceptable stage of completion.

5.3 User Acceptance Testing

With all of the requirements satisfied and a usable piece of software complete, the project supervisor was requested to test the application. This was done in an attempt to gain an account of the applications usability. After testing the software, the supervisor advised with some basic modifications and alterations. These modifications were implemented and the process was reiterated.

6 Experimentation

With a fully functioning piece of software, the experimentation phase could begin. Below are the screenshots for generating the Efficient Frontier with the following assets:

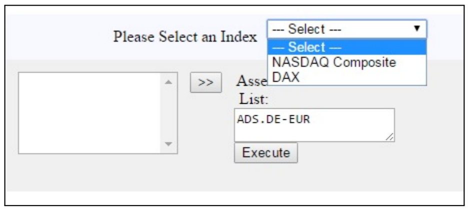


Figure 2: User selecting an index to fetch its corresponding components.

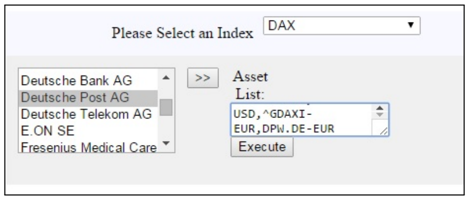
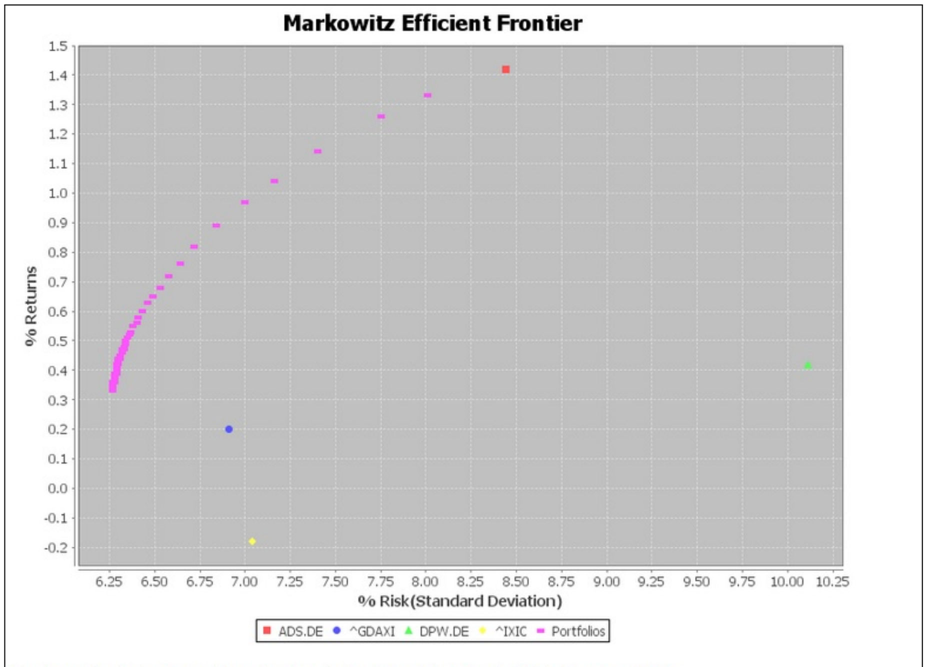
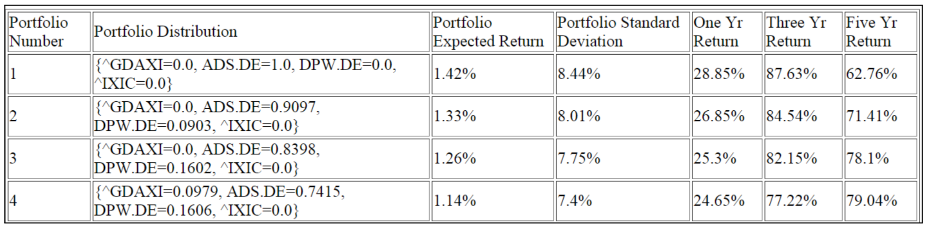


Figure 3: A set of different assets selected by user for portfolio creation.



The above frontier is calculated based on data obtained from November 1, 2000 to January 4, 2010.
The 1 Year, 3 Year and 5 Year Returns are calculated assuming the investment is performed on January 4, 2010.
All the Assets are converted to a common EUR Currency.

Figure 4: The plot of frontier and the corresponding individual assets



Portfolio Number	Portfolio Distribution	Portfolio Expected Return	Portfolio Standard Deviation	One Yr Return	Three Yr Return	Five Yr Return
1	{^GDAXI=0.0, ADS.DE=1.0, DPW.DE=0.0, ^IXIC=0.0}	1.42%	8.44%	28.85%	87.63%	62.76%
2	{^GDAXI=0.0, ADS.DE=0.9097, DPW.DE=0.0903, ^IXIC=0.0}	1.33%	8.01%	26.85%	84.54%	71.41%
3	{^GDAXI=0.0, ADS.DE=0.8398, DPW.DE=0.1602, ^IXIC=0.0}	1.26%	7.75%	25.3%	82.15%	78.1%
4	{^GDAXI=0.0979, ADS.DE=0.7415, DPW.DE=0.1606, ^IXIC=0.0}	1.14%	7.4%	24.65%	77.22%	79.04%

Figure 5: Different portfolios and their 1, 3 and 5 year performance.

7 Conclusion

With all requirements and objectives achieved, I am satisfied with the work done on this project. Working on this project has proved to be a challenging yet extremely rewarding experience. The experience of learning about the domain and applying the knowledge gained over the course has been benificial in the development of the project and I have also learnt new Software development practices along the way.

8 Future Work

The work carried out in on this project opens itself to a possibility of future improvements. The structure of the software was designed with code reuse and future extensibility in mind. Future work recommendations include incorporating other portfolio creation models into the application and using custom data streams rather than relying on Yahoo! Finance API.

9 Appendix

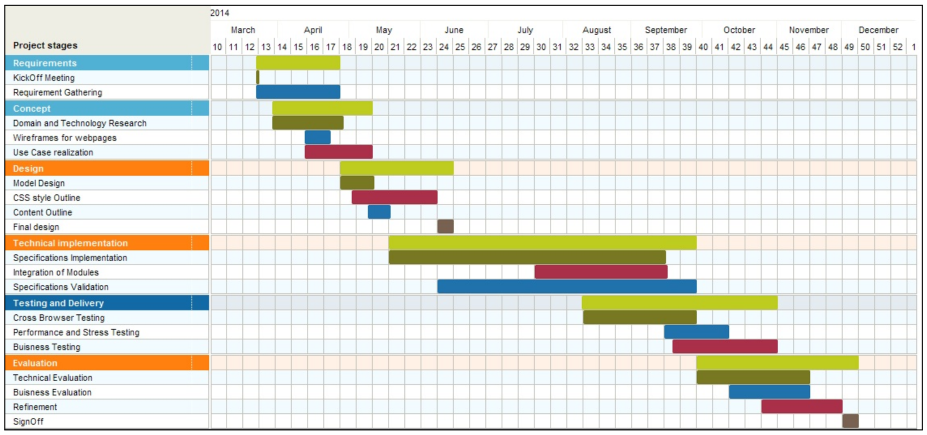


Figure 6: Gantt chart of tasks described in section 2.1

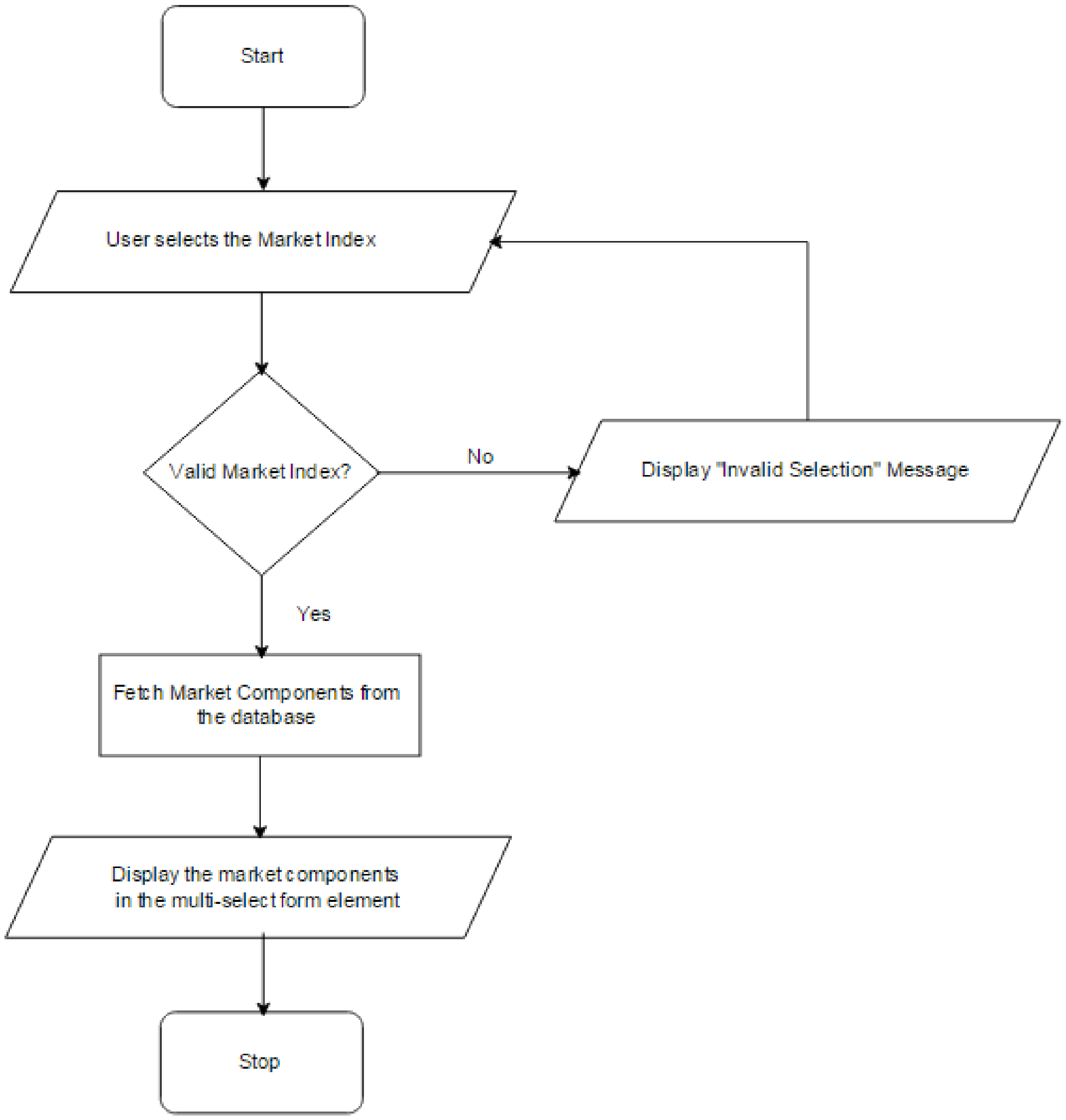
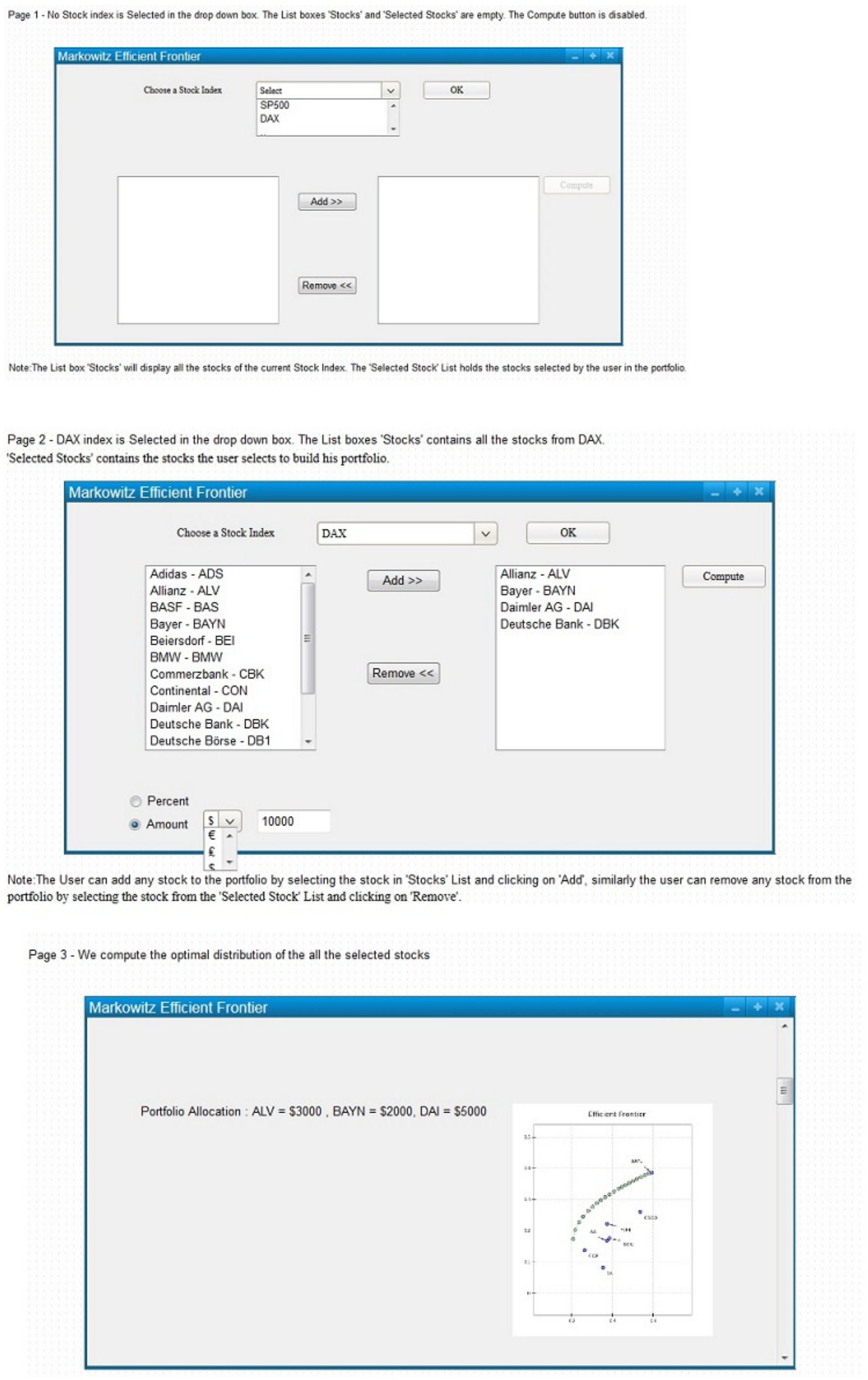

Start
User selects the Market Index
Valid Market Index?
No
Display "Invalid Selection" Message
Yes
Fetch Market Components from the database
Display the market components in the multi-select form element
Stop

Figure 7: Flowchart To Fetch Market Components.


Page 1 - No Stock index is Selected in the drop down box. The List boxes 'Stocks' and 'Selected Stocks' are empty. The Compute button is disabled.

Note:The List box 'Stocks' will display all the stocks of the current Stock Index. The 'Selected Stock' List holds the stocks selected by the user in the portfolio.

Page 2 - DAX index is Selected in the drop down box. The List boxes 'Stocks' contains all the stocks from DAX.
'Selected Stocks' contains the stocks the user selects to build his portfolio.

Note:The User can add any stock to the portfolio by selecting the stock in 'Stocks' List and clicking on 'Add', similarly the user can remove any stock from the portfolio by selecting the stock from the 'Selected Stock' List and clicking on 'Remove'.

Page 3 - We compute the optimal distribution of the all the selected stocks

Portfolio Allocation : ALV = $3000 , BAYN = $2000, DAI = $5000

Figure 8: Wireframes during the Design Phase.

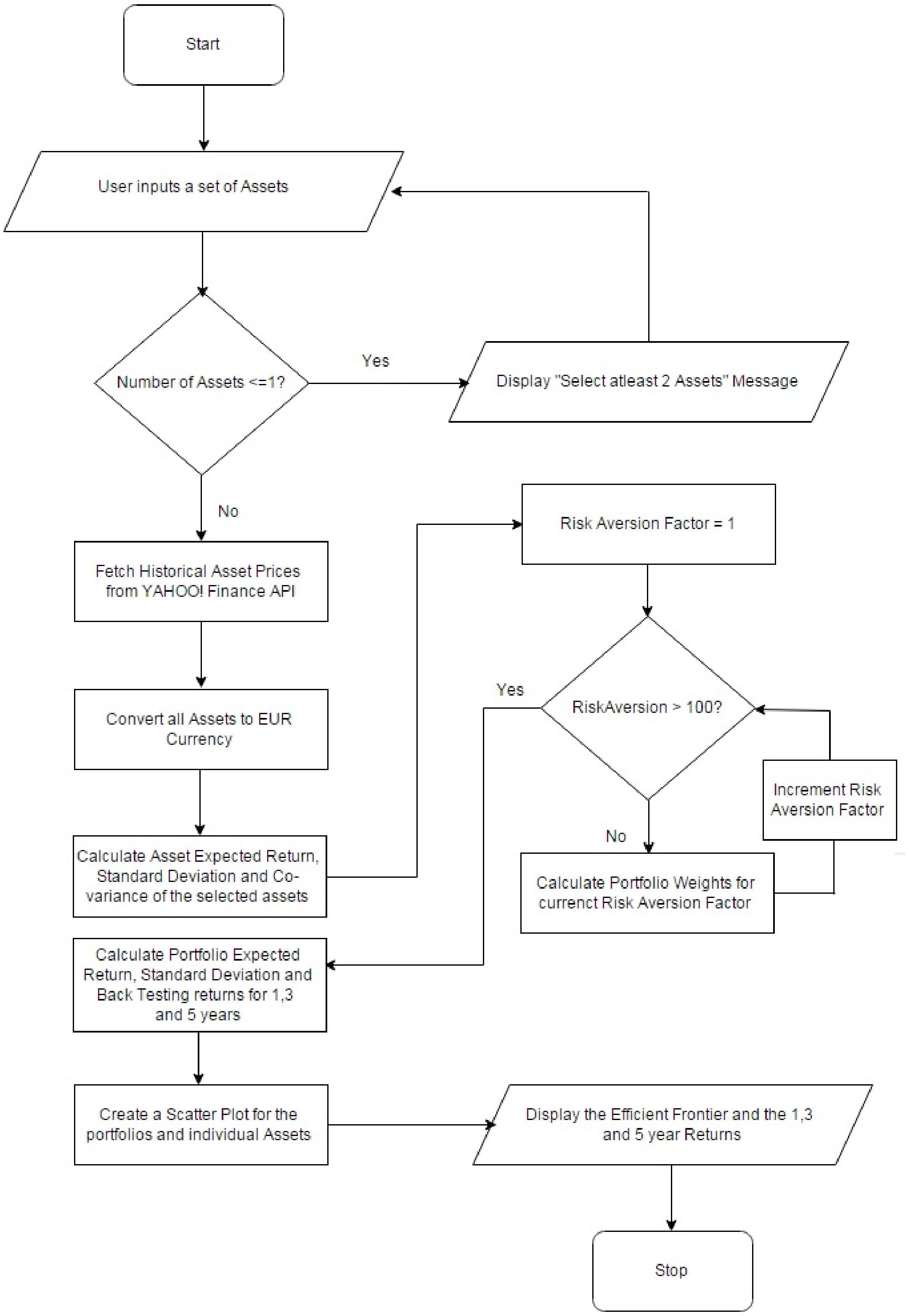

Start
User inputs a set of Assets
Number of Assets <=1?
Yes
Display "Select atleast 2 Assets" Message
No
Fetch Historical Asset Prices from YAHOO! Finance API
Convert all Assets to EUR Currency
Calculate Asset Expected Return, Standard Deviation and Co-variance of the selected assets
Risk Aversion Factor = 1
RiskAversion > 100?
Yes
No
Increment Risk Aversion Factor
Calculate Portfolio Weights for currenct Risk Aversion Factor
Calculate Portfolio Expected Return, Standard Deviation and Back Testing returns for 1,3 and 5 years
Create a Scatter Plot for the portfolios and individual Assets
Display the Efficient Frontier and the 1,3 and 5 year Returns
Stop

Figure 9: Flowchart To Calculate the Efficient Frontier and 1, 3 and 5 Year Returns(Back Testing).

References

[1] Markowitz, Harry. ”Portfolio selection*.” The journal of finance 7, no. 1 (1952): 77-91.

[2] Witt, Stephen F., and Richard Dobbins. ”The Markowitz Contribution to Portfolio Theory.” Managerial Finance 5, no. 1 (1979): 3-17.

[3] West, Graeme. ”An introduction to Modern Portfolio Theory: Markowitz, CAP-M, APT and Black-Litterman.” Parktown North: Financial Modelling Agency (2006).