INVESTMENT PROPOSAL

Master's Thesis submitted to the

Constructor Institute

in partial fulfilment of the requirement for the degree of

Master of Science in Computer Science and Software Engineering

presented by

Leakhena Ellene SUON

under the supervision of

Prof. Dr. Mauro Pezzè

co-supervised by

Mr. Nicolas Kaiser

Widen, June 25, 2023
I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Leakhena Ellene SUON
Widen, 25 June 2023
To my beloved

I would like to express my deepest gratitude to my parents who have always supported and encouraged me to invest in high education in all circumstances. Without their unconditional love and emotional support, this success cannot be achieved.

No words can express my gratitude to my husband who has always stood by my side to help and sort out all problems I faced during my study in this master program. Without his financial and emotional support, I could not undertake this journey.
Abstract

Portfolio optimization is a process defined by asset managers to construct, the most efficient portfolios by using a risk/return point-of-view. The purpose of this analysis is to minimize risks and maximize returns of investment. Basically, there are many methods that can be used for investment decisions, but one of the best methods for portfolio optimization is Mean-Variance Optimal portfolio optimization technique, developed by Nobel prize winner Harry Markowitz. This technique is also known as modern portfolio theory (MPT). Therefore, ETOPS’s engineering team has developed a portfolio optimization tool based on MPT model to assist asset managers in achieving their task of selecting right investments and right asset mix thereof successfully.

The system is based on a modular architecture with two modules called PortOps and Investment Proposal, which are independently developed but can communicate and interact with each other.

Technically, PortOps provides core functionalities relevant to portfolio optimization. Investment Proposal communicates with PortOps via Rest API to provide client data. Finally, it gets the result from the portfolio optimization module and render the output data on different platforms such as mobile and desktop applications.

My responsibility in this project only covers one module that is Investment Proposal module. In this module, I have to collaborate with Etops team for the whole process, starting from analysis design, development testing and integration. Due to the limited timeframe during my practical work in the company, for the purpose of my master thesis, the scope is limited to developing the first iteration including some functionalities as mentioned in section 2.5 and integrating with PortOps module with test data. Nevertheless, it must be developed to a high standard so that it can be integrated with PortOps for next version.
Acknowledgement

I would like to thank Prof. Dr. Mauro Pezzè who has kindly accepted to be my advisor and always provided me with a lot of useful advice relevant to the professionalism of the system’s test. I would also like to show appreciation to all professors for their patience for giving valuable professional education which is of great importance for using in workplaces in Switzerland or anywhere else in the world. I am also grateful to my classmates and cohort members for their help and support.

Likewise, I would like to thank Mr. Nicolas Kaiser, the company advisor, who has always provided me with all the means I need to support my thesis and practicum work in the company.

Words cannot express my gratitude to Etops for providing me an opportunity to be a part of their wonderful team to develop “investment proposal”. It is a very fascinating topic.

Last but not least, I would like to deeply thank Mr. Pius Stucki, Etops CEO, and Mr. Christian Jedlicka, Etops CSO.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Requirement Analysis</td>
<td>6</td>
</tr>
<tr>
<td>2.2 System Overview</td>
<td>6</td>
</tr>
<tr>
<td>2.3 Actors</td>
<td>7</td>
</tr>
<tr>
<td>2.4 Use Case Diagram</td>
<td>8</td>
</tr>
<tr>
<td>2.5 Functional Requirement Summarize</td>
<td>16</td>
</tr>
<tr>
<td>2.6 Non-functional Requirement</td>
<td>17</td>
</tr>
<tr>
<td>High Level Design</td>
<td>18</td>
</tr>
<tr>
<td>3.1 Application Flow</td>
<td>19</td>
</tr>
<tr>
<td>3.2 Application Architecture</td>
<td>20</td>
</tr>
<tr>
<td>3.3 Database Architecture</td>
<td>21</td>
</tr>
<tr>
<td>3.4 Technology Architecture</td>
<td>24</td>
</tr>
<tr>
<td>3.4.1 Infrastructure</td>
<td>24</td>
</tr>
<tr>
<td>3.4.2 Data Architecture</td>
<td>28</td>
</tr>
<tr>
<td>3.4.3 Software Component</td>
<td>29</td>
</tr>
<tr>
<td>3.4.4 System Integration</td>
<td>36</td>
</tr>
<tr>
<td>4.1 Class Diagram</td>
<td>37</td>
</tr>
<tr>
<td>4.2 Development Architecture</td>
<td>38</td>
</tr>
<tr>
<td>Testing</td>
<td>41</td>
</tr>
<tr>
<td>5.1 Functional Testing Methodology</td>
<td>41</td>
</tr>
<tr>
<td>5.1.1 Unity Test</td>
<td>41</td>
</tr>
<tr>
<td>5.1.2 Integration Test</td>
<td>44</td>
</tr>
</tbody>
</table>
5.1.3 System Test........................................................................................................44
5.2 Nonfunctional Testing Methodology ....................................................................44
  5.2.1 Security Test ..................................................................................................44
  5.2.2 Performance Test .........................................................................................44
  5.2.3 Usability and Compatibility Test .................................................................44
Summary & Conclusion ..........................................................................................45
Glossary ....................................................................................................................46
References ...............................................................................................................47
List of Figures

Figure 1. 1: Portfolio Optimization Basic Concept ................................................................. 1
Figure 1. 2: Portfolio Optimization Case Study........................................................................... 2
Figure 1. 3: Portfolio Return and Risk Formula ......................................................................... 2
Figure 1. 4: Sharpe Ratio Formula .......................................................................................... 3
Figure 1. 5: 2-Dimensional Plot of Efficient Frontier and CAL .................................................. 3
Figure 1. 6: Markowitz 3-Dimensional Plot ............................................................................. 4
Figure 1. 7: Iterative Software Development Process .................................................................. 5

Figure 2. 1: Etops Business Solution ......................................................................................... 6
Figure 2. 2: Investment Proposal System Overview ................................................................. 7
Figure 2. 3: Actors Interesting in the System ............................................................................ 8
Figure 2. 4: Use Case Diagram ............................................................................................... 8

Figure 3. 1: Application Flow .................................................................................................... 18
Figure 3. 2: Application Architecture ...................................................................................... 20
Figure 3. 3: Monolithic Architecture ....................................................................................... 21
Figure 3. 4: Physical Data Model of Investment Proposal......................................................... 22
Figure 3. 5: System Infrastructure .......................................................................................... 24
Figure 3. 6: JSON Web Token (JWT) ..................................................................................... 25
Figure 3. 7: JSON Web Token verification ............................................................................... 26
Figure 3. 8: Secure Login with JWT ....................................................................................... 27
Figure 3. 9: PortOps Data Warehouse ...................................................................................... 29
Figure 3. 10: 4+1 View Model ............................................................................................... 29
Figure 3. 11: Invest Proposal System Context View ............................................................... 30
Figure 3. 12: Logical View ...................................................................................................... 31
Figure 3. 13: Create Prospect .................................................................................................. 32
Figure 3. 14: Edit Prospect ..................................................................................................... 32
Figure 3. 15: List All Prospects ............................................................................................. 33
Figure 3. 16: Create & Display Proposal ............................................................................... 33
Figure 3. 17: Edit Proposal .................................................................................................... 34
Figure 3. 18: Edit Proposal .................................................................................................... 35

Figure 4. 1: Investment Proposal Class Diagram ..................................................................... 37
Figure 4. 2: React, Node.js Express, PostgreSQL Architecture .............................................. 38
Figure 4. 3: React Router Code Snippet ............................................................................... 38
Figure 4. 4: React Service Code Snippet ............................................................................. 39
Figure 4. 5: Node.js Router .................................................................................................. 39
Figure 4. 6: Controller Node.js ............................................................................................ 40
List of Table

Table 2.5. 1: Functional Requirement ........................................................................................................ 16

Table 3. 1: Investment Proposal Logical Model ......................................................................................... 23

Table 5.1.1. 1: Prospect Management Test Suit ........................................................................................ 41
Table 5.1.1. 2: Proposal Management Test Suit ......................................................................................... 42
Chapter 1

Introduction

Portfolio optimization can be a complex task that takes up a large portion of an asset manager’s time, in which the manager has to collect data and generate a portfolio analysis on Excel spreadsheets. As such, this traditional method is not very effective as it requires still to a large extent manual work and time which has a subsequent impact on cost. Therefore, a portfolio optimization software is needed and being developed in order to help asset managers in handily and effortlessly selecting the best portfolios, the most efficient ones to invest in.

Figure 1.1: Portfolio Optimization Basic Concept

The process of portfolio optimization is to provide data such as:
- risk profile,
- thematic bias,
- client restrictions,
- individual client preferences, etc.

to an optimizer software for it to process and then deliver a result of portfolio optimization, which includes a list of the best portfolio. The applied algorithm used for portfolio optimization is called Modern Portfolio Theory (MPT). This theory was created by Mr. Harry Markowitz for his PhD dissertation, written in 1952 and subsequently published in an article in Journal of Finance 1).
CHAPTER 1. INTRODUCTION

His approach helps investors or asset managers to diversify portfolios to get a higher expected yield with a given level of risk or a lower level of portfolio risk with a given level of expected yield. The key element of this modern portfolio theory is Efficient Frontier that can be defined by the position of each portfolio on a two-dimensional axes (x,y), while x-axis represents standard deviation (risk) and y-axis represents expected return or portfolio yield. For instance, to create a two-dimensional (x,y) axes diagram of the Efficient Frontier graph, a real case study of 2 assets—SPY and VXUS—as shown in the following figure will depict the process to define the best portfolio

Figure 1. 2: Portfolio Optimization Case Study

![Figure 1. 2: Portfolio Optimization Case Study](https://www.youtube.com/watch?v=dJipa0K64HI)

Hence, to define the parabolic curve mentioned above, the values of return and standard deviation must be calculated via the formula as follows:

Figure 1. 3: Portfolio Return and Risk Formula

![Figure 1. 3: Portfolio Return and Risk Formula](https://www.youtube.com/watch?v=dJipa0K64)
CHAPTER 1. INTRODUCTION

To plot this portfolio, the value of Shape Ratio must be calculated via a formula as follows:

Figure 1.4: Sharpe Ratio Formula

![Sharpe Ratio Formula](https://www.youtube.com/watch?v=dJipa0K64Hl)

Figure 1.5: 2-Dimensional Plot of Efficient Frontier and CAL

![2-Dimensional Plot of Efficient Frontier and CAL](https://www.youtube.com/watch?v=dJipa0K64Hl)

The maximum value of Sharpe Ratio is 14.03%. Its value of return and standard deviation (risk) is 0.13% and 0.92%. In other words, the best portfolio can be found on the linear Capital Allocation Line (CAL, see above). Its touching point to the Efficient Frontier graph is a point comprising the values of x (standard deviation: 0.92%) and y (return: 0.13%).

The Markowitz Mean-Variance approach only takes 2 dimensions into account such as risk and expected return. Of course, there is the possibility to add another dimension, such as ESG criteria, which, however, makes the calculation of the optimal portfolio much more complex.

This 3-dimensional plot is a factor for the investor or asset manager to take into account when considering the impacts to the company sales based on the criteria related to the environmental, social, and corporate governance, the so-called ESG factor.

Therefore, Markowitz bullet is used to plot in a 3-dimensional space in order to fulfill the need of asset manager or investor who has a concern on this interference. Mathematically, x represents risk, y represents return, and z represents climate score or carbon score and so on.
CHAPTER 1. INTRODUCTION

Figure 6 below shows a 3-dimensional plot according to the carbon level considered as an environmental effect.

Figure 1. 6: Markowitz 3-Dimensional Plot

![Markowitz 3-Dimensional Plot](https://gregorygundersen.com/blog/2022/01/09/geometry-efficient-frontier/)

Besides performing optimization task, the software also provides a service to backtrack the portfolio performance before they are finally placed in the stock trading market.

The method of this performance tracking is to compare the best portfolios found with the benchmark: back test, transition risk caused by temperature, financial resilience, and physical resilience to define if it is overperformance or underperformance over a period of time. Detailed description of this scenarios will be shown in the section below.

The process of portfolio optimization mentioned above is only a brief concept as I am not a core module developer. I have been appointed to develop the investment proposal module interacting with core module (portOps) through JSON data format in order to display the result of portfolio optimization on desktop and mobile platform. Hence, the topic of this thesis is dedicated to the sub system development of the portfolio optimization software called investment proposal.

The development methodology is iterative as the team is working together to refine each iteration until the team is satisfied with the final delivery.
CHAPTER 1. INTRODUCTION

Figure below shows the detailed process of investment proposal development methodology.

**Figure 1.7: Iterative Software Development Process**

![Iterative Software Development Process](https://www.javatpoint.com/software-engineering-iterative-model)

This thesis is composed of the following sections:

- Chapter 1: Introduction
- Chapter 2: Requirement Analysis
- Chapter 3: High Level Design
- Chapter 4: Low Level Design
- Chapter 5: Testing

4+1 architectural view model is the standard model used for describing the architecture of the software.
Chapter 2

Requirement Analysis

2.1 Business Solution

Before getting through the system overview, first I need to understand how the system provides business solutions to their customers to fulfil their business needs regarding the portfolio optimization.

The business flow starts from Etops company selling the software license to their customers. Once both parties agree to all regulations, the asset manager’s account is created by the Etops team. Then asset manager can log into the system and create their future customers who were previously defined as prospects. Through software usability, asset manager can define the best portfolio optimizing for their customers to invest. Figure 8.1 below shows the detailed workflow.

2.2 System Overview

Investment Proposal is defined as a separated module which interacts with PortOps in order to render the information related to the result of portfolio optimization to the users. The idea of this separation is to provide users very simple and useful information by just viewing the final result of the best optimal portfolio.
CHAPTER 2. Requirement Analysis

Moreover, this architecture provides more secured and better performance because the core operations cannot be accessed directly by external users. Another benefit is that the operation related to the read and write to database is not overloaded because these 2 modules have different database.

Figure 8.2 illustrates the operations that can be accessed by asset managers to manage prospects and proposals. The details of actors and each use case description will be shown in the actor and use case description section.

2.3 Actors

The actors interacting with the system are:

➢ Administrator: refers to the user working in Etops and having the right to manage the asset manager's account.
➢ Asset Manager: refers to the user working in the bank and getting the account from administrator to create prospects and manage investment proposals.
CHAPTER 2. Requirement Analysis

Figure 2.3: Actors Interesting in the System

![Actor Diagram]

Admin  Asset Manager

2.4 Use Case Diagram

The use case diagram below shows all features to be accessed by the asset manager.

Figure 2.4: Use Case Diagram

![Use Case Diagram]

1. Manage Prospect module consists of several use cases as follows:
   1. Create Prospect
   2. List All Prospect
   3. Update Prospect
CHAPTER 2. Requirement Analysis

(a) Create Prospect

(1) Name: Create Prospect

(UC-CPR-01)

(2) Brief Description:

This use case allows the asset manager to create the prospect.

(3) Flow of Events

a. Basic Flow:

The Prospect is created with the following information

- Salutation
- Name
- First Name
- Middle Name
- Title
- Place of Birth
- Date of Birth
- Legal Capacity
- Investment Profile
- Nationality
- Tax Domicile
- Language

b. Alternative Flow:

The administrator can check first if the person is an old prospect by searching their names in the search box, if the

it is the case, a list of all prospects matching the names will be show up.

(4) Special Requirement:

Login as asset manager is mandatory
CHAPTER 2. Requirement Analysis

(5) Pre-Conditions

All fields below must be valid.
- Salutation
- Name
- First Name
- Legal Capacity
- Investment Profile
- Nationality
- Tax Domicile
- Language

(6) Post-Conditions

After pre-conditions are successfully completed. Then, asset manager can move on to the next phase to create the proposal matching with that prospect.

(b) List ALL Prospects

(1) Name: List All Prospects

(UC-LPR-02)

(2) Brief Description:

This use case allows the asset manager to view the prospect list.

(3) Flow of Events

a. Basic Flow:

The asset manager clicks on Existing Prospect link to view all prospects

b. Alternative Flow:

Not applicable

(4) Special Requirement:

Login as asset manager is mandatory

(5) Pre-Conditions: Not applicable

(6) Post-Conditions

All created prospects are selected from Investment Proposal database to display on the front end.
CHAPTER 2. Requirement Analysis

(c) Edit Prospect

(1) Name: Edit Prospect
   (UC-EPR-03)

(2) Brief Description:
   This use case allows the asset manager to edit the prospect.

(3) Flow of Events
   a. Basic Flow:
      The asset manager clicks on the Edit Prospect link. Then the form of
      prospect information is displayed so that the asset manager can modified.
   b. Alternative Flow:
      Not applicable

(4) Special Requirements:
   Login as asset manager is mandatory

(5) Pre-Conditions: The modified field must be valid.

(6) Post-Conditions
   New edit information of the prospect must be saved to database and the
   successful edit message must be popped up.

2. The Manage Proposal module consists of several use cases as follows:
   1. Create Proposal
   2. Update Proposal

(d) Create Proposal

(1) Name: Create Proposal
   (UC-CPR-04)

(2) Brief Description:
   This use case allows the prospect create the proposal.
CHAPTER 2. Requirement Analysis

(3) Flow of Events

a. Basic Flow:

To create the proposal, asset manager has to define the following steps as below:

1. Define Risk Profile

1.1: the prospect selects one of 5 predefined strategies:

- Very conservative: 80% Fixed Income / 20% Equities
- Conservative: 60% Fixed Income / 40%
- Equities
- Balanced: 40% Fixed Income / 60% Equities
- Dynamic: 20% Fixed Income / 80% Equities
- Very dynamic: 100% Equities

1.2: the prospect can add a Cash and an Alternative Investment share in the following fields:

- Equities (%)
- Fixed Income (%)
- Cash (%)
- Alternatives (%)

1.3: the prospect defines currency allocation and asset allocation in the following fields:

- Currency Allocation
  - EUR (from % to %)
  - USD (from % to %)
  - CHF (from % to %)
  - GBP (from % to %)
  - Other (from % to %)

- Asset Allocation
  - Cash (from % to %)
  - Equities (from % to %)
  - Fixed Income (from % to %)
  - Alternatives (from % to %)

2. Define Thematic

The prospect can choose which investment sectors they are interested in. By default, the system provides the following thematic as follows:
CHAPTER 2. Requirement Analysis

- energy transition
- circular economy
- water & forestry
- digitalization
- fold & healthcare
- Knowledge society

3. Define Restriction
The asset manager can choose the type of investment that they are not interested in. By default, the system provides the investment restriction such as:

- Pesticides
- Fossil fuels
- Weapon
- Tobacco
- Alcohol
- Meat
- Animal testing

b. Alternative Flow:
If the asset manager has already created at least one proposal, he/she can view the existed proposal and edit it.

(4) Special Requirement:
Log in as the asset manager is mandatory.

(5) Pre-Conditions
Risk profile, thematic, and restriction must be defined.

(6) Post-Conditions
After the prospect completely finished for the process of defining risk profile, thematic, and restriction, a new proposal will be saved to the database and the preference page will be displayed with the result of the best optimal portfolio with the following information:

➢ Instrument (Asset Class)
➢ Business Sector
➢ Currency
➢ Weight
➢ Graph of preference parameters (risk, return, climate, impact)

Moreover, the system allows the prospect to evaluate the performance of portfolio optimization by performing the following operations:
CHAPTER 2. Requirement Analysis

➢ Strategy Back Test
  • The prospect can view graphs which compare the values of portfolio and benchmarks according to their preferred time frame such as the last 1 year, 3 years, 5 years or current year.
➢ Transition Risk (to be defined in the next version of the system)
➢ Financial Resilience (to be defined in the next version of the system)
➢ Physical Resilience (to be defined in the next version of the system)

(e) Edith Proposal

(1) Name: Edith Proposal
  (UC-EPR-05)

(2) Brief Description:
This use case allows prospect to edit the proposal.

(3) Flow of Events
  a. Basic Flow: The prospect performs the following operation during Edit Proposal.

  1. Add securities to a wish list: the prospect can add new security (financial instrument) to the same business sector.
  2. Add securities to an exclusion list: the prospect can add security to the exclusion list.
  3. Change the optimization criteria: the prospect can change the values of preference parameters for the portfolio optimization. There are 4 values of preference parameter such as risk, return, climate, and impact.
  4. Change the weights distribution manually: the prospect can manually change the weight of distribution but the sum of all weight must not exceed 100%.

  b. Alternative Flow: Not applicable

(4) Special Requirement:
Log in as asset manager is mandatory.

(5) Pre-Conditions:
The new input value must not be null, and the sum of all distribution weight must not be greater than 100%.

(6) Post-Conditions:
After the prospect completely provides information to be edited, a message signalling successful edit will pop up and those data will be updated in the database.
CHAPTER 2. Requirement Analysis

(f) View Portfolio Performance

(1) Name: View Portfolio Performance
   (UC-VPP-06)

(2) Brief Description:
   This use case is mentioned in Post-Conditions Section (on page 17).

(g) Evaluate Future Performance

(1) Name: Evaluate Portfolio Future Performance
   (UC-EFP-07)

(2) Brief Description:
   This use case allows the asset manager to forecast portfolio performance in the future.

(3) Flow of Events
   a. Basic Flow: The prospect can view more future forecasting of the entire portfolio performance based on the criteria as follows:
      1. Estimated asset development: shows the predicted value of the entire portfolio's grow rate based on the specific period of time. See source below to get the formula to calculate the entire grow rates. https://www.wikihow.com/Calculate-Growth-Rate
      2. Estimate annual yield: shows the predicted value of the entire portfolio's annual average growth rate. The formula to calculate this value is the summation of each portfolio growth rate divided by number of total portfolios.
      3. Climate alignment: shows the predicted value of the climate alignment. Normally, if the value found is below 2°C, it means that it is aligned with the climate targets of the Paris Agreement in order to limit the global warming. See source below for more detailed information. https://www.infras.ch/media/filer_public/a0/36/a03639c7-102a-4ca9-aed5d-deae01a2dced/report_portfolio_climate_alignment_infras_hsg_220621.pdf
      5. Physical value at risk: forecasts the value of the maximum loss related to the physical impacts of the climate change.
CHAPTER 2. Requirement Analysis

6. Biodiversity footprint forecasts the value of biodiversity footprint by using the below method:

\[ \text{Footprint} = \sum (\text{ha area in use}_i \times [1 - \text{MSA pressur} \text{e factor}_i]) \]

See source below for more detailed information.


7. Impact thematic: shows the forecasted value of thematic impact

8. Portfolio Positioning

b. Alternative Flow: Not applicable

(4) Special Requirement:

Log in as admin is mandatory.

(5) Pre-Conditions

Not applicable

(6) Post-Conditions

After the asset manager clicks “Next”, a proposal page will appear with the following information. The system will connect to PortOps via RestAPI to render a result to be sent to the front end.

2.5 Functional Requirement Summarize

The table below lists all the functional requirements to be developed according to their priorities. If the level of priority is low, that functionality is developed in the next version.

Table 2.5. 1: Functional Requirement

<table>
<thead>
<tr>
<th>#</th>
<th>Function Name</th>
<th>Use Case</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Login</td>
<td></td>
<td>Very High</td>
</tr>
<tr>
<td>2</td>
<td>Create Prospect</td>
<td>(UC-CPR-01)</td>
<td>Very High</td>
</tr>
<tr>
<td>3</td>
<td>List All Prospect</td>
<td>(UC-LPR-02)</td>
<td>Very High</td>
</tr>
<tr>
<td>4</td>
<td>Edit Prospect</td>
<td>(UC-EPR-03)</td>
<td>High</td>
</tr>
<tr>
<td>5</td>
<td>Create Proposal</td>
<td>(UC-CPR-04)</td>
<td>Very High</td>
</tr>
<tr>
<td>6</td>
<td>Edit Proposal</td>
<td>(UC-EPR-05)</td>
<td>Very High</td>
</tr>
<tr>
<td>7</td>
<td>View Portfolio Performance</td>
<td>(UC-VPP-06)</td>
<td>Next Iteration</td>
</tr>
<tr>
<td>8</td>
<td>Evaluate Future Performance</td>
<td>(UC-EFP-07)</td>
<td>Next Iteration</td>
</tr>
</tbody>
</table>
CHAPTER 2. Requirement Analysis

2.6 None Functional Requirement

The system must provide users with the user-friendliness and usability conditions. Especially, it must adhere to the following criteria:

- correctness
- robustness
- reliability
- scalability
Chapter 3

High Level Design

Figure 3.1: Application Flow
CHAPTER 3. High Level Design

3.1 Application Flow

Figure above shows the detailed process of the application flow starting from a user with a role of an asset manager logging into the system. If login is correct:

1. They can create a new prospect by providing all required information. If the prospect already existed based on the information provided, the asset manager will see a list of the existing prospects where the manager can edit the prospects and add new or edit proposals attached to the prospects.

2. If the asset manager clicks a link to add a new proposal, the manager has to provide the information such as risk profile, thematic, and restriction step-by-step, so that the external module called PortOps can use the new information to optimize the best portfolio shown in the Preference page. Moreover, the user can also evaluate the current portfolio performance based on methods such as Portfolio Backtest. There are many Backtest strategies the system can use, but for this version only one strategy is applied. It is Strategy Backtest.

There is a formula to calculate benchmark's value over the timeframe for us to use it for comparison with portfolio values starting from the past to the current date. If the benchmark value is greater than the portfolio's return value, this means that we underperform the custom benchmark.

See the following source to see detailed explanation on how to calculate benchmark value and do an evaluation.

https://www.google.com/search?q=portfolio+benchmark&sxsrf=APwXEdcbB9pNtv4gNl3UDrsrrNCQtf6tg:1682372301800&source=lnms&tbm=vid&sa=X&ved=2ahUKEwi61ravcrbvcP-AhWBhf0HYh1BRoQ_AUoBHoECAEOBg&biw=1487&bih=698&dpr=1.25#fpstate=ive&vld=cid:8c611cb2,vid:udgg_Fg-GNM

The system is not only eligible to evaluate the present performance of the portfolio optimization, it can also evaluate the future performance by using the strategy mentioned in Evaluate Future Performance use case.

3. If the user clicks the link to edit the proposal, they can update the information such as proposal name, wishlist, exclude wishlist, and weight.

The operation relevant to future performance’s evaluation needs external data such as market data to estimate asset development. The carbon and finance data are needed to evaluate the transition risk, financial resilience, and physical resilience. Market data is stored in CSS database, while carbon and finance data are stored in Azure Blob Storage, Microsoft's cloud solution for the massive unstructured object data storage. The sftp gateway filemage is a file transfer.
CHAPTER 3. High Level Design

Solution seamlessly connects SFTP and FTP(S) protocols to Azure Blob Storage. See the link below to get more information relevant to the concept of filemage.

https://github.com/filemage

The Azure Data Factory (DF) pulls data from Azure BLOB Storage by transforming, filtering and enhancing, before moving it along to another destination. In this project the destination that the data is moved to is PortOps database.

See the link below to get more information relevant to Azure DF.

https://towardsdatascience.com/azure-data-factory-basic-concepts-e10448e54023

3.2 Application Architecture

Figure 3. 2: Application Architecture

The application is designed with microservice architecture which consists of two microservices, API Gateways and PortOps. They have their own technology stack and database. API Gateways provide a mid-tier service which handles the client requests related the investment proposal such as creating prospect, editing prospect, creating proposal, and communicating with PortOps microservices via Rest API in order to get the result of portfolio optimization to render that json response to the clients. PortOps is the core business logic. It handles portfolio optimization tasks such as showcasing best portfolio optimization, performing portfolio back test and showing detailed proposal attached with future portfolio performance.
CHAPTER 3. High Level Design

This architectural design provides more security than the combination of Investment proposal and PortOps in one business logic because it prevents external clients from connecting directly to the core system, which is a high risk for information security including confidentiality, integrity, and availability.

Another reason behind choosing the microservice architecture is that it can solve the problems that monolithic architecture is encountering such as code complexity, highly interdependent components, limited scalability, and new barrier of technology.

Figure 13 below showcases the disadvantage of monolithic architecture as all components are bundled in one box.

![Figure 3.3: Monolithic Architecture](https://www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-business/#:~:text=While%20a%20monolithic%20application%20is,well%20as%20perform%20specific%20functions.)

3.3 Database Architecture

Figure 3.4 below shows entity relationship of Investment Proposal.
CHAPTER 3. High Level Design

Figure 3. 4: Physical Data Model of Investment Proposal
CHAPTER 3. High Level Design

Based on physical data model above, we can convert to a logical model represented by the table as follows:

Table 3. 1: Investment Proposal Logical Model

<table>
<thead>
<tr>
<th>No.</th>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>USER</td>
<td>store user data</td>
</tr>
<tr>
<td>2</td>
<td>ROLE</td>
<td>store role data</td>
</tr>
<tr>
<td>3</td>
<td>PROSPECT</td>
<td>Store prospect data</td>
</tr>
<tr>
<td>4</td>
<td>PROPOSAL</td>
<td>store proposal data</td>
</tr>
<tr>
<td>5</td>
<td>RISK_PROFILE</td>
<td>store proposal data</td>
</tr>
<tr>
<td>6</td>
<td>INVESTMENT_STRATEGY</td>
<td>store investment strategy data</td>
</tr>
<tr>
<td>7</td>
<td>PREFERENCE</td>
<td>store preference data</td>
</tr>
<tr>
<td>8</td>
<td>ASSET_ALLOCATION</td>
<td>store asset allocation data</td>
</tr>
<tr>
<td>9</td>
<td>ASSET_CLASS</td>
<td>store asset class data</td>
</tr>
<tr>
<td>10</td>
<td>CURRENCY_ALLOCATION</td>
<td>store currency allocation data</td>
</tr>
<tr>
<td>11</td>
<td>CURRENCY</td>
<td>store currency data</td>
</tr>
<tr>
<td>12</td>
<td>RESTRICTION_PROPOSAL</td>
<td>store restriction proposal data</td>
</tr>
<tr>
<td>13</td>
<td>RESTRICTION</td>
<td>store restriction data</td>
</tr>
<tr>
<td>14</td>
<td>LANGUAGE_RESTRICTION</td>
<td>store language restriction data</td>
</tr>
<tr>
<td>15</td>
<td>THEMATIC</td>
<td>store thematic data</td>
</tr>
<tr>
<td>16</td>
<td>PROPOSAL_THEMATIC</td>
<td>store thematic proposal data</td>
</tr>
<tr>
<td>17</td>
<td>LANGUAGE_THEMATIC</td>
<td>store language thematic data</td>
</tr>
<tr>
<td>18</td>
<td>SECURITY_ADD_PROPOSAL</td>
<td>store security adds to Wishlist data</td>
</tr>
<tr>
<td>19</td>
<td>SECURITY_EXCLUDE</td>
<td>store security excludes from Wishlist</td>
</tr>
<tr>
<td>20</td>
<td>LANGUAGE</td>
<td>store language data</td>
</tr>
</tbody>
</table>
CHAPTER 3. High Level Design

3.4 Technology Architecture

Technology architecture showcases the technical blueprint related to interaction and interdependent of components, so that system’s requirements are met.

3.4.1 Infrastructure

Figure 3. 5: System Infrastructure

The above figure illustrates the communication between webserver and application server via protocol http. First application server deployed Investment Proposal microservice, while another one deploys PortOps. This provides a core business logic relevant to portfolio optimization. These two application servers can connect to and communicate with each other via Rest API.

This architectural design provides a good solution for system’s security and performance as explained in details in the section below.

3.4.1.1 Application Security

Figure 14 illustrates how Investment Proposal is an API gateway that interacts with communication between the clients and PortOps server. This architecture provides more security for the information.

Another solution for application security is the secure authentication and authorization based on JSON Web Token (JWT) technology.
JWT provides more security than traditional session-based authentication which does not pass any encryption method before sending the session id to the browser's cookie. HTTP cookie is vulnerable to the information security. Hence, authorization and authentication with JWT tackle this problem. The process below dispatches the interaction between client and server with JWT mechanism.

1. Client sends credential information with a username and a password to the server.
2. Server authenticates the client's credential. If the username and password are correct, the server creates JWT with an encrypted secret key and a signature for a specific period of time.
3. Server provides JWT with the information such as a secret key signed, user information and authorization to the client's browser which stores it for future access to the website visits.
4. Client sends the request including JWT to the server.
5. Server verifies if JWT has not been changed since the time that it has been signed. The process how JWT do the verification is explained in details in Figure 15.
6. If everything is fine, the client is authorized to get access to the content.
CHAPTER 3. High Level Design

Figure 3.7: JSON Web Token verification

JWT creates an encoded secret key with the signature to send to the client. To be the authorized user, the client sends the encoded key to the server for decoding process which 3 distinct parts

1. HEADER: after decoding the red text of the encoded part, header is determined as follows:
   a. alg: the algorithm used for encoding and decoding
   b. typ: token type

2. PAYLOAD: information stored in the token
   c. sub: ide of the authenticated user
   d. name: user name
   e. iat: expire at the specific period of time

3. VERIFY SIGNATURE:
   f. base64UrlEncode(header): encode the header section with the secret key (your-256-bit-secret).
   g. base64UrlEncode(payload): encode the payload section with the secret key (your-256-bit-secret).
CHAPTER 3. High Level Design

When the server retrieves the client’s request with JWT, it will decode both sections, then combine it together, and hash it with a particular algorithm in order to check the result of matching the last portion of the key. If the result fails, there will be a change on header or payload. As a result, the server will not authorize that client to gain access to the resources.

Figure 3.8 below shows the development view of security login with JWT authentication and authorization.

Figure 3.8: Secure Login with JWT

![Secure Login with JWT](https://www.bezkoder.com/react-express-authentication-jwt/)

The backend development for secure login is implemented with Node.js and PostgreSQL. This is explained in the following process.

1. After the server retrieves HTTP request from the client, CORS Middleware via Express.js will match the rout. If the rout is not matched, CORS will throw an error and a message will be sent to HTTP response.

2. Otherwise JWT Authentication Middle ware will proceed with token verification.

3. If the encrypted key with signature fails to verify, JWT Authentication Middle ware will throw an error, and a message will be sent to HTTP response.
CHAPTER 3. High Level Design

4. Otherwise, the level security is passed and Authorization Middleware checks the user’s roles in the database. If he or she is not an authorized user, Authorization Middleware will throw an error, and a message will be sent to HTTP response.

5. Otherwise, controllers will interact with PostgreSQL Database via Sequelize and send HTTP response including token, user information, data based on role to the client.

3.4.1.2 Application Performance

This design also provides a scalable system in order to reduce the workload. Obviously, 2 application servers handle 2 different tasks as mentioned above, and each of them connect with their own PostgreSQL database.

Another solution for system performance is server-side caching. It is an approach where the server stores a copy of webpage (Static Data Store) in a specific amount of time after the first visit, so the next time the user revisits the website, the server will send back the already saved web page without reconstructing or regenerating new contents from its database.

Last but not least, JWT mechanism helps to improve a performance as well because there is no any session stored on the server database or memory like the session-based authentication. All the JWT information is systematically stored on client’s browser.

3.4.2 Data Architecture

There are three types of data architecture, but this project covers only two types including application database and data warehouse.

3.4.2.1 Application Database

Application database stores clients’ information such as the prospect, risk profile, thematic, preference, etc. Technically, those data are stored in PostgreSQL database management system.

3.4.2.2 Data Warehouse

Detailed description of this data architecture is done in Section 3 of the application flow on page 23 and 24.
CHAPTER 3. High Level Design

Figure 3. 9: PortOps Data Warehouse

3.4.3 Software Component

4+1 view model is used as an architectural blueprint to convey the system design from logical view to physical view. Figure 17 below shows how the fourth different view describes the software based on the use cases description as mentioned in above section.

Logical and process views are mentioned in this section, while development and physical views will be described in the low-level architecture below.

Figure 3. 10: 4+1 View Model
CHAPTER 3. High Level Design

3.4.3.1 System Context View

Figure 18 bellow recaps the system context starting from asset manager access, to interface of investment proposal, to create the prospect and to proposal. The use case description and scenarios of those features are explained in details in the requirement analysis section. The component, investment proposal GUI, uses the services that implement the create prospect and create proposal functionalities to save the client data to the database. PortOps provides portfolio optimization service to investment proposal GUI to render the result of portfolio analysis for the asset manager.

Figure 3. 11: Invest Proposal System Context View

The Logical and process view will be illustrated in details for each use case as follows.

3.4.3.2 Proposal Management

To simplify the logical view, the use cases including creating prospect, listing all prospect, editing prospect, creating proposal and editing proposal are grouped together in one use case. They are implemented in component and investment proposal service.
CHAPTER 3. High Level Design

3.4.3.2.1 Logical View

Figure 3. 12: Logical View

User Interface component interacts with the System manager component via REST API to render a result, which is sent as the JSON data at client side. System manager is responsible for managing all client's requests and interacting with the component, Investment Proposal Service, which provides the business logic operation to compute the client's request such as create prospect, list prospects, edit prospect, create proposal, edit proposal and display proposal. Finally, business logic component interacts with Client Data component via DB driver to perform database the database operation.

System manager component is also responsible for communicating with PortOps Service component to get portfolio optimizing service and render the result of portfolio optimization as the JSON data at client side.

**Styles:**
Microservice Architecture
Data-Centric (Persist data)
Client/Server

**Patterns:**
Model-View-Controller
CHAPTER 3. High Level Design

3.4.3.2.2 Process View

Sequence diagram dispatches the process view for each use case.

Figure 3. 13: Create Prospect

1. User creates prospect by entering the prospect information.
2. After browser validates the input, system manager invokes service *add prospect*.
3. Investment proposal service invokes the save prospect service to insert the prospect to the client database.

Figure 3. 14: Edit Prospect

1. User selects which prospect to be edited.
2. After the browser validates the input, system manager invokes service *edit prospect*.
3. Investment proposal service updates the prospect to the client database.
CHAPTER 3. High Level Design

Figure 3.15: List All Prospects

1. User inputs prospect information.
2. System manager invokes the service list prospect based on the information provided.
3. Investment proposal service gets the list prospects from data base and return the prospect list to system manager. System manager forward the list to user interface.

Figure 3.16: Create & Display Proposal
CHAPTER 3. High Level Design

1. User creates a proposal by entering the information: risk profile, thematic and restriction. For prospect_id, proposal_id and user_id, they are systematically provided.
2. After all input are validated, system manager invokes service to create proposal.
3. Investment proposal service saves proposal to the client database.
4. System manager interacts with the remote PortOps to get the best portfolio based on the specific user id and prospect id.
5. Remote PortOps returns the best portfolio to the system manager.
6. System manager return the best portfolio to the user.
7. User analyses the portfolio backtest based on proposal id
8. System manager interacts with the remote PortOps to get the portfolio backtest
9. Remote PortOps returns the portfolio backtest to the system manager.
10. System manager renders the backtest result and sends it to the user.

Figure 3.17: Edit Proposal
CHAPTER 3. High Level Design

Figure 3.17 above illustrates the sequence diagram of editing the proposal based on add new security in the wishlist or exclude the security from the wishlist. Adding security and removing security from the wishlist are optional operations according to the need of the user to change the values of these parameters. If one or both operations are done by the user, the added new security will be saved to wishlist table, and the security exclusion will be saved to the exclusion wishlist table of client database before old security has been changed in the wishlist. Finally, the result of the best portfolio will be optimized again based on the security in the wishlist.

Figure 3.18: Edit Proposal

According to the sequence diagram above, user can optionally:

1. Edit the risk profile with the parameters including risk, return, impact and climate. After editing those value, the new portfolio optimization will be manipulated and the result of the best portfolios will be shown to the user.

2. Change the weight of distribution and update the weight to the client database if the total weight is less than 100. Otherwise, an error message will be shown on the client side.
CHAPTER 3. High Level Design

3.4.4 System Integration

Investment proposal system must be integrated with PortOps to build a complete system of portfolio optimization. Even though, system integration is not an easy task but according to the advantage of the architectural design paralleling to the principles, low coupling and high cohesion, which allows investment proposal module interact with PortOps through the Rest API without getting concern with the implementation of the PortOps module. The investment proposal itself is designed with the appropriate class's relationship in order to manipulate the relevant operations correctly. Another important solution to avoid the integration issue is each class contains the attribute and method that represent the system behaviour. For example, a Prospect class represents the prospect behaviour and contains a method to validate the email address. In short, it does not contain unrelated elements.
The diagram above illustrates the process and relationship between each class of the investment proposal domain model. By converting this class diagram to the development based on node.js and react technology, we can see that each class represents one react component whose attribute needs to validated before invoking the method developed as the API with node.js technology in order to render the result to the client via URL end point.
CHAPTER 4. Low Level Design

According to the system’s scope for the first version, this class diagram is limited to only English as the default language. However, the design can also be extended in the next version to accept other languages without having a problem of high dependency.

4.2 Development Architecture

Figure 4. 2: React, Node.js Express, PostgreSQL Architecture

Source: https://www.bezkoder.com/react-node-express-postgresql/

CRUD (create, read, update, and delete) operation for the investment proposal development follows the architecture as shown in the above figure. React is the front-end technology applying the Router to map one page to one component. Figure below shows code snippet that navigates to different pages.

Note: Code snippets below do not present the development structure of investment proposal, but it is the structure to be followed for the development of the investment proposal API.

Figure 4. 3: React Router Code Snippet

```html
<div className="container mt-3">
  <Routes>
    <Route path="/" element={<TutorialsList/>} />
    <Route path="/tutorials" element={<TutorialsList/>} />
    <Route path="/add" element={<AddTutorial/>} />
    <Route path="/tutorials/:id" element={<Tutorial/>} />
  </Routes>
</div>
```
CHAPTER 4. Low Level Design

React Service contains the URL end-point to be sent via HTTP client (axios) according to different operation such as get, post, put and delete. Axios has ability to send the http request and receive the response data.

Figure below shows code snippet of the react service.

Figure 4. 4: React Service Code Snippet

```javascript
class TutorialDataService {
    getAll() {
        return http.get("/tutorials");
    }

    get(id) {
        return http.get("/tutorials/${id}");
    }

    create(data) {
        return http.post("/tutorials", data);
    }

    update(id, data) {
        return http.put("/tutorials/${id}", data);
    }

    delete(id) {
        return http.delete("/tutorials/${id}");
    }

    deleteAll() {
        return http.delete("/tutorials");
    }
}
```

Node.js is the technology chosen for API development. According to the http request sent from the client by the react service, application server can route which controller to manipulate the client request by invoking the ORM (object relational mapping) to connect to the database and perform the insert, select, update, and delete operation. This description is defined by the following code snippet.

Figure 4. 5: Node.js Router

```javascript
var router = require("express").Router();

// Create a new Tutorial
router.post("/", tutorials.create);

// Retrieve all Tutorials
router.get("/", tutorials.findAll);

// Retrieve all published Tutorials
router.get("/published", tutorials.findAllPublished);

require("./app/routes/tutorial.routes")(app);

// set port, listen for requests
const PORT = process.env.PORT || 8000;
app.listen(PORT, () => {
    console.log("Server is running on port ${PORT}.");
});
```
CHAPTER 4. Low Level Design

Figure 4.6: Controller Node.js

```javascript
const db = require("../models");
const Tutorial = db.tutorial;
const Test = db.test;
const Op = db.Sequelize.Op;

// Create and Save a new Tutorial
exports.create = (req, res) => {

  // Validate request
  if (!req.body.title) {
    res.status(400).send({
      message: "Content can not be empty!",
    });
    return;
  }

  // Create a Tutorial
  const tutorial = {
    title: req.body.title,
    description: req.body.description,
  };
```
Chapter 5

Testing

5.1 Functional Testing Methodology

Functional testing refers to the different types of tests which includes unity test, integration test and system test. It is a black box testing technique which ensures that the system responses to the customers’ requirement.

5.1.1 Unity Test

Regarding to the unity test, the following test suit and test case are defined as follows.

Table 5.1.1. 1: Prospect Management Test Suit

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1. Create Prospect | 1. Validation test: check the mandatory inputs if they are empty.  
2. Functional test: new prospect must be saved in database. |
| 2. List All Prospect | 1. Validation test: all prospects must be successfully selected from database. |
| 3. Edit Prospect   | 1. Validation test: check the mandatory inputs if they are empty.  
2. Functional test: prospect must be changed in database. |
Table 5.1.1. 2: Proposal Management Test Suit

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1. Create Proposal | 1.1 Validation test for risk profile:  
➢ Check if investment strategy is selected.  
➢ Check if cash and alternative investment share text box is empty.  
➢ Check if currency allocation is not null.  
➢ Check if the asset allocation is not null.  
1.2 Functional test: risk profile for this proposal must be saved to the database and rendered to the next page to define thematic. |
|                  | 2.1 Validation test for thematic:  
➢ Check if investment sector is selected. |
|                  | 2.2 Functional test: thematic for this proposal must be saved to the database and rendered to the next page to define restriction. |  
|                  | 3.1 Validation test for restriction:  
➢ Check if investment type restriction is selected. |
|                  | 3.2 Functional test: restriction for this proposal must be saved to the database. |
|                  | After all, 3 definitions step are defined, new proposal must be saved to the database and rendered to the next preference page. |
### CHAPTER 5. Testing

<table>
<thead>
<tr>
<th>Test Case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Edit Proposal</td>
<td>1.1 Validation test for add wishlist:</td>
</tr>
<tr>
<td></td>
<td>➢ Check if new financial instrument is provided when users decide to add a new security to the wishlist.</td>
</tr>
<tr>
<td></td>
<td>1.2 Functional test: a new security must be saved to the table wishlist in the database.</td>
</tr>
<tr>
<td></td>
<td>2.1 Validation test for exclusion list:</td>
</tr>
<tr>
<td></td>
<td>➢ Check if financial instrument excluded in the wishlist is listed in the exclusion list.</td>
</tr>
<tr>
<td></td>
<td>2.2 Functional test: new exclusion security must be saved in table exclusion list in the database.</td>
</tr>
<tr>
<td>2. Edit Proposal</td>
<td>1.1 Validation test for changing the optimization criteria:</td>
</tr>
<tr>
<td></td>
<td>➢ Check if value of preference parameter for the portfolio optimization is provided.</td>
</tr>
<tr>
<td></td>
<td>➢ Check the weight</td>
</tr>
<tr>
<td></td>
<td>1.2 Functional test: new security must be saved to the table wishlist in the database.</td>
</tr>
<tr>
<td></td>
<td>2.1 Validation test for exclusion list:</td>
</tr>
<tr>
<td></td>
<td>➢ Check if financial instrument excluded in the wishlist is listed in the exclusion list.</td>
</tr>
<tr>
<td></td>
<td>2.2 Functional test: new exclusion security must be saved in table exclusion list in the database.</td>
</tr>
</tbody>
</table>
CHAPTER 5. Testing

5.1.2 Integration Test

Investment proposal module must be well integrated with the PortOps module in order to build up a complete system of portfolio optimization.

According to the requirement analysis mentioned in above section, 3 test cases are found as shown below in order to verify if the rendered results through the communication of investment proposal with PortOps via web service end-point are correct:

1. View best portfolio optimization shown on the preference page
2. View Portfolio Performance
3. Evaluate Future Performance

5.1.3 System Test

The 23rd of June is the date to arrange a demo for the stake holders including the top management of Etops. To reduce the complexity, they define the first iteration to be developed and tested is the integration between investment proposal and PortOps module with the sample data. After collecting their feedback, development team will know where to make improvement.

5.2 Nonfunctional Testing Methodology

The purpose of this testing methodology is to cover the security test, performance test, usability and compatibility test.

5.2.1 Security Test

The test case for the security test is the attempt to login to the system with the correct username and password but with the null or unverified token. So, the result of the test must show that the system does not redirect to the user's dashboard. Instead, it redirects to the homepage.

5.2.2 Performance Test

It requires the simulation test tool to check if the system will slow down or halt when there are multi concurrent connections to the web server and application server.

5.2.3 Usability and Compatibly Test

It is mandatory to get the feedback from the users for the usability test in order to make sure that the user interface is user-friendly and prevents frustration. Last but not least, it is mandatory to test the system performance running on different browsers and operating systems.
Summary & Conclusion

I have always wished for an opportunity to get involved in a project relevant to investment or the fin tech system, so this is a huge opportunity for me to be accepted by Etops to join this interesting project. Throughout this project, I can apply my existing experience and the recent knowledge I have acquired at my institute—Constructor—to build a qualified software. Moreover, this project allows me to work with tasks I really like which includes software analysis, software design, and development with node.js and react technology.

The presentation of the development part was set to be demonstrated to the top management on the 23rd of June. However, the first iteration could not be completed in time with approximate completion of 30% of the overall mock up screen due to some problems related to family, social events, thesis writing and technology. Therefore, the demonstration shall be rescheduled afterward.

Nevertheless, I have made a lot more progress after spending more time working with the front-end development part. Now, about 70% of the first iteration is completely done.

The last demonstration shall be done during my defence day at Constructor Institute.
Glossary

UC-CPR-01: Use Case Create Prospect
UC-LPR-02: Use Case List All Prospects
UC-EPR-03: Use Case Edit Prospect
UC-CPR-04: Use Case Create Proposal
UC-EPR-05: Use Case Edit Proposal
UC-VPP-06: Use Case View Portfolio Performance
UC-EFP-07: Use Case Evaluate Portfolio Future Performance
ESG: Environmental, social, and corporate governance
References

url: https://www.bezkoder.com/node-js-jwt-authentication-postgresql/

url: https://www.bezkoder.com/react-node-express-postgresql/


url: https://www.indeed.com/career-advice/career-development/testing-methodologies


url: https://edgemesh.com/blog/difference-between-server-side-caching-and-client-side-caching-and-which-is-good-for-your-website

url: https://esg.conservice.com/esg-scores-why-they-matter/#:~:text=An%20ESG%20score%20is%20a,to-day%20work%20and%20operations


