Icon

Description automatically generated

Software Engineering

MOD003263

Contents

[Problem Definition 3](file:///C:\Users\sevda\Downloads\2121058_MOD003263%20(2).docx#_Toc121059821)

[Case Study 3](file:///C:\Users\sevda\Downloads\2121058_MOD003263%20(2).docx#_Toc121059822)

[Current System Overview 3](file:///C:\Users\sevda\Downloads\2121058_MOD003263%20(2).docx#_Toc121059823)

[User and System Requirements 3](file:///C:\Users\sevda\Downloads\2121058_MOD003263%20(2).docx#_Toc121059824)

[Project Aims 3](file:///C:\Users\sevda\Downloads\2121058_MOD003263%20(2).docx#_Toc121059825)

[User Stories 4](file:///C:\Users\sevda\Downloads\2121058_MOD003263%20(2).docx#_Toc121059826)

[Fund Administrator 4](file:///C:\Users\sevda\Downloads\2121058_MOD003263%20(2).docx#_Toc121059827)

[Client 4](file:///C:\Users\sevda\Downloads\2121058_MOD003263%20(2).docx#_Toc121059828)

[Relationship Manager 4](file:///C:\Users\sevda\Downloads\2121058_MOD003263%20(2).docx#_Toc121059829)

[Project Plan (Gantt Chart) 5](file:///C:\Users\sevda\Downloads\2121058_MOD003263%20(2).docx#_Toc121059830)

[Required Resources 5](file:///C:\Users\sevda\Downloads\2121058_MOD003263%20(2).docx#_Toc121059831)

# Problem Definition

## Case Study

The company provides investment services to high-net-worth clients. The advice is tailored to each client, with “relationship managers” handling the client recommendations. “Fund administrators” collate groups of related investments into “investment ideas”, these are the product that is then recommended to the clients.

The company offers investment through many major sectors e.g., renewable energies, blockchain, real estate, bonds, precious metals, etc.

The fact there is seemingly no standardised process for storing and accessing information about clients or ideas, other than emails, suggests that information about the clients and ideas that would help to make informed decisions about the suitability of different investments may not be accessible as conveniently as desired. Lack of convenience means a lack of speed, which in turn leads to lost revenue.

## Current System Overview

* Email based
* No dedicated “hub” for clients, relationship managers or fund administrators
* Relationship managers must switch between emails and current investment idea data to review suitability
* Fund administrators must switch between emails and investment product data to collate products into ideas

# User and System Requirements

## Project Aims

A piece of software will be developed with the aim of streamlining the current system. This piece of software will allow clients, relationship managers and fund administrators to facilitate their current aims within one “hub” (the software). This approach should eliminate the need for switching between wherever it is the company currently store information about products and ideas and an email client, saving time and money. Implementation of algorithms to sort clients by suitability ideas should also take some of the decision-making time away from the relationship manager, allowing for higher turnover, and again, more revenue.

By utilising a login area with different account types, clients, relationship managers and fund administrators will all be able to access tailored content and functionality within the same piece of software.

## User Stories

Based on the problem definition and project aims, these user stories have been generated to highlight the intended functionality of the software for each user.

### Fund Administrator

* As a fund administrator, I want to create an account
* As a fund administrator, I want to be able to login to my account
* As a fund administrator, I want to be able to upload investment ideas for the RM to view
* As a fund administrator, I want to be able to browse products

### Client

* As a client I want to create an account
* As a client I want to be able to login to my account
* As a client I want to be able to store preferences (preferred investment type, etc.)
* As a client I want to browse products by type to match my preferred investments
* As a client I want to be able view RM recommendations
* As a client I want to be able to approve RM recommendations

### Relationship Manager

* As an RM I want to create an account
* As an RM I want to be able to login to my account
* As an RM I want to be able to view all clients
* As an RM I want to be able to view all investment ideas

## Project Plan (Gantt Chart)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Task** | Wk1 | Wk2 | Wk3 | Wk4 | Wk5 | Wk6 | Wk7 | Wk8 | Wk9 | Wk10 | Wk11 | Wk12 | Wk13 |
| Continuous re-evaluation |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Assignment of team roles |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Outline coding standards |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Outline UI design standards |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Problem definition |  |  |  |  |  |  |  |  |  |  |  |  |  |
| MOSCOW analysis |  |  |  |  |  |  |  |  |  |  |  |  |  |
| User stories |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Use case diagrams |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Class diagram |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Database Design |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Database Implementation |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UI Design |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UI Prototype |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Code implementation |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Meeting with client |  |  |  |  |  |  |  |  |  |  |  |  |  |

Table 1. Project Gantt Chart

## Required Resources

The following resources are required to effectively deliver this product in accordance with the project plan:

* Access to a workstation each at all times
* Visual Studio 2022 for delivering that product as C#-based WinForms program
* Adobe Creative Cloud (for creation of some UI elements)
* Drawi.io diagram software
* GitHub for collaboration and source control

# Software Maintenance

For the maintenance of the plan is to have multiple branches for the sysadmin to maintain with there being the main branch the beta branch the dev branch and the nightly branch. The main branch being the one in which is released for the public which can also be known as the stable version. The beta branch will be the one which is used for the new features that may be added so that they can be tested for issues, it is also used to determine whether the userbase likes these features. The dev branch is the internal testing branch which is similar to the beta branch and will be used to instead test functions that could help in the creation of ideas, products or accounts. The nightly branch is for those which are the bravest and is for the most unstable changes being much further down the roadmap so that these features have a lot more time to bake in the oven.

The way that these functions will be added up or down will be through the usage of GitHub and the way that those branches be modified would be through the usage of GitHub merging. The way that you would want to merge down would be to go to the upper branch and do a pull request to the lower branch whereas with the lower branch and then if there is aby issues you would have to deal with them individually. Whilst with merging up it is similar except instead of going to the upper you go to the lower and then pull to the upper one.

# Software Implementation Documentation

## Database Connectivity

A screenshot of a computer

Description automatically generated with medium confidenceText

Description automatically generatedOne of the functionalities contributed to the team is creating a connection between the database and the application. The class that does that is called DBFactory, it is using an interface DBConnection which will be inherited by other classes that are database providers (example: SQLite, MySQL, Oracle, DynamoDB). This allows for further software maintenance as we can decide later to add new databases with new providers.

To start with, a singleton is used in the constructor to access the class, avoiding creating new instances of it, this way there will be running only a single connection when needed and it will be closed once the program does not read from the database.

Text

Description automatically generatedThe string propfile will be used to access a text file which contains information about the database provider (SQLite) and the database name (Database.db).

Text

Description automatically generatedThe function getProperties() is used in the getConnection() method to read the properties.dat file and return the database provider and name. Then the getConnection() method checks if the provider exists and creates a connection between the database and application, if the provider does not exists throw an error message. getConnection() also checks if the file exists, if it does not throw an error message.

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generatedThe class SqLiteCon inherits the functions from DBFactory Interface and uses them to create a connection with a SQLite database. The method initialise() finds the database and sets up the connection. The method setConnection() gets the database read from initialise() and sets up the connection by using SQLiteConnection imported from System.Data.SQLite.

There are open and close methods implemented to manipulate the connection when it is needed and when it must be closed after the function is done. There is a select function which gets a string SQLite style query and reads the database, this will be explained further on the BusinessMetaLayer class.

Text

Description automatically generatedThe BusinessMetaLayer class is the layer between the database and application, where in database terms, connection and queries are considered and application is object oriented. In BusinessMetaLayer functions which read database elements and return objects are created, in this example there is an empty list of users, a connection is opened and a function to read the database using a query is created, each line read is assigned to a user object and each object is added to create a new user. Each user created is added to the list and then is returned when the method getUsers() is called.

Integrating database functionality was key to the project, as we use it to log in, create users, check client preferences, create investment ideas and suggest ideas to clients.

## Application Forms

Graphical user interface, application

Description automatically generatedText

Description automatically generated

Text

Description automatically generatedText

Description automatically generatedLog in form- the grey text on the username and password fields is used as a place holder, when a user clicks the text clears and changes to black. The event handlers check from the database if the username and password match any of the users and flag the Login form as DialogRresult.OK if they do match, else they log in as guests. Previously the foreach statements were checking for Relationship Manager, Clients, and Fund Administrator separately, because they were on separate tables, however the code was later updated by another member of the team to search for all users and get their user type. Once the user is authenticated, the main form appears with the relevant to user type menu.

Text

Description automatically generatedMain form- this form was designed to access all menu items using a single form. This is achieved using panels. Each panel will be displayed to the user thanks to the buttons provided on the left. Clicking on the buttons will invoke event handlers to bring the corresponding panel to the front.