Graded Unit-Scenario 2

*Davide Kovac*

*563490*

22/01/2020

**Contents**

1. Introduction 5

2. Planning 6

2.1 Analysis of the Project Brief 6

2.1.1 Interpretation of the Project Brief 6

2.1.2 Natural Language analysis 9

2.1.3 Functional/Non-functional requirements(initial) 11

2.1.4 Use case Diagram(initial) 12

2.1.5 Use cases description(initial) 13

2.1.6 Client Interview 14

2.1.7 Background research 15

2.1.8 2nd Interview(clarification) 15

2.1.9 Aim of the Project 16

2.1.10 Functional/non-functional requirements(revised) 16

2.1.11 Use case diagram(revised) 17

2.1.12 Use case description (revised) 18

2.1.13 Resources and material 19

2.2 Project Stage 20

2.3 Solution Stage 20

2.3.1 Fully Dressed Use cases description 21

2.3.2 CRC cards 28

2.3.3 Model Diagram 30

2.3.4 Class Diagram 30

2.3.5 Communication Diagram 43

2.3.6 Activity Diagram 45

2.3.7 Sequence Diagram 48

2.3.8 Database Design 53

2.3.9 Graphical design & Client meeting 62

2.3.10 Data Binding 76

5. Bibliography 79

**Table of Figures**

[Figure 1:Python vs R through the years 8](#_Toc33393071)

[Figure 2: Agile Methodology 9](#_Toc33393072)

[Figure 3: Initial use case diagram 13](#_Toc33393073)

[Figure 4: Revised use case diagram 18](#_Toc33393074)

[Figure 5: Model diagram 31](#_Toc33393075)

[Figure 6:Package division 32](#_Toc33393076)

[Figure 7:Client Class 33](#_Toc33393077)

[Figure 8:Supplier Class 34](#_Toc33393078)

[Figure 9:Product Class 34](#_Toc33393079)

[Figure 10:Order Class 35](#_Toc33393080)

[Figure 11:NewStock Class 35](#_Toc33393081)

[Figure 12:Report Class 36](#_Toc33393082)

[Figure 13:DatabaseManager Class 36](#_Toc33393083)

[Figure 14:MenuUI Class 37](#_Toc33393084)

[Figure 15:EnterStockChangesUI Class 37](#_Toc33393085)

[Figure 16:NewUserUI Class 38](#_Toc33393086)

[Figure 17:GetInfoUI Class 38](#_Toc33393087)

[Figure 18:DisplayInfoUI Class 39](#_Toc33393088)

[Figure 19:SelectReportUI Class 39](#_Toc33393089)

[Figure 20:DisplayReportUI Class 40](#_Toc33393090)

[Figure 21:Order-Product association 41](#_Toc33393091)

[Figure 22:NewStock-Product association 41](#_Toc33393092)

[Figure 23:.application classes and .access classes relationship(example) 42](#_Toc33393093)

[Figure 24:Inheritance in the .gui package 42](#_Toc33393094)

[Figure 25:.gui JPanel interaction 43](#_Toc33393095)

[Figure 26:Class Diagram 43](#_Toc33393096)

[Figure 27:Update Stock use case Communication Diagram 44](#_Toc33393097)

[Figure 28:Get Info Communication Diagram 45](#_Toc33393098)

[Figure 29:Create Report Communication Diagram 45](#_Toc33393099)

[Figure 30:Update Stock Activity Diagram 47](#_Toc33393100)

[Figure 31:Get Info Activity Diagram 48](#_Toc33393101)

[Figure 32:Create Report Activity Diagram 49](#_Toc33393102)

[Figure 33:Update Stock Sequence Diagram 51](#_Toc33393103)

[Figure 34:Get Info sequence diagram 52](#_Toc33393104)

[Figure 35:Create Report sequence diagram 53](#_Toc33393105)

[Figure 36:Top Level ERD 56](#_Toc33393106)

[Figure 37:Top Level ERD with Attributes 57](#_Toc33393107)

[Figure 38:Orders- Product M:M relationship 58](#_Toc33393108)

[Figure 39:Orders-Products solved M:M relationship 58](#_Toc33393109)

[Figure 40:Products-Suppliers M: M relationship 59](#_Toc33393110)

[Figure 41:Products-Suppliers solved M:M relationship 59](#_Toc33393111)

[Figure 42:Clients-Orders relationship before optionality review 59](#_Toc33393112)

[Figure 43:Clients-Orders relationship with the optionality improved 60](#_Toc33393113)

[Figure 44:Clients-Orders relationship with transferability 60](#_Toc33393114)

[Figure 45:Orders-Ordered Items-Products relationship without optionality 60](#_Toc33393115)

[Figure 46:Orders-Ordered Products-Products relationship with Optionality 60](#_Toc33393116)

[Figure 47:Products-New Stocks-Suppliers relationship without optionality 61](#_Toc33393117)

[Figure 48:Products-New Stock-Suppliers relationship with optionality. 61](#_Toc33393118)

[Figure 49:Products-New Stock-Supplier relationship with transferability 61](#_Toc33393119)

[Figure 50: Final ERD 62](#_Toc33393120)

[Figure 51:Logo Nº1 63](#_Toc33393121)

[Figure 52:Logo Nº2 64](#_Toc33393122)

[Figure 53:Logo Nº3 64](#_Toc33393123)

[Figure 54:Logo Nº4 65](#_Toc33393124)

[Figure 55:Logo Nº5 65](#_Toc33393125)

[Figure 56:Logo Nº6 65](#_Toc33393126)

[Figure 57:Logo Nº7 66](#_Toc33393127)

[Figure 58:Color Scheme 1-"normal view" 66](#_Toc33393128)

[Figure 59: Color Scheme 1-Deuteranomaly view 67](#_Toc33393129)

[Figure 60:Color Scheme 2-"normal view" 67](#_Toc33393130)

[Figure 61:Color Scheme 2-Deuteranomaly view 68](#_Toc33393131)

[Figure 62:Color Scheme 3 68](#_Toc33393132)

[Figure 63:Menu wireframe 69](#_Toc33393133)

[Figure 64:Update Stock wireframe 70](#_Toc33393134)

[Figure 65:New Client/Supplier Wireframe 71](#_Toc33393135)

[Figure 66:New Client/Supplier added wireframe 71](#_Toc33393136)

[Figure 67:Enter Stock changes wireframe 72](#_Toc33393137)

[Figure 68:Stockage outages/shortage wireframe 72](#_Toc33393138)

[Figure 69:Stock successfully updated wireframe 72](#_Toc33393139)

[Figure 70:Get info wireframe 73](#_Toc33393140)

[Figure 71:Select Info wireframe 73](#_Toc33393141)

[Figure 72:Info missing/incorrect wireframe 74](#_Toc33393142)

[Figure 73:Select Report type wireframe 74](#_Toc33393143)

[Figure 74:Report info missing/incomplete wireframe 75](#_Toc33393144)

[Figure 75:Logo selected by the client 75](#_Toc33393145)

[Figure 76:Color Scheme selected by the client 76](#_Toc33393146)

[Figure 77:Data Binding for show product information in an order 77](#_Toc33393147)

[Figure 78:Data Binding for show product information in a new Stock 78](#_Toc33393148)

[Figure 79:Get Client info data binding 79](#_Toc33393149)

[Figure 80:Get Supplier info data binding 79](#_Toc33393150)

[Figure 81:Get Order info data binding 79](#_Toc33393151)

# 1. Introduction

This is the document for my graded unit project where I write a program for a small business where they can update stock live, add client order and produce reports. For this program I decide to use Java for programming side, Java swing using Windows builder to create the GUI and MySQL for the database. Java and MySQL are then connected using JDBC.

# 2. Planning

## 2.1 Analysis of the Project Brief

### 2.1.1 Interpretation of the Project Brief

The second scenario is a program that need to be develop for a company that needs to produce report, update stocks and place orders coming from clients. There are several ways to approach this program. On this section I will try to analyse the pro and cons of three of this option and the reasons I have arrived at the conclusion to undertake the develop of this program in this way. The options that came to my mind when I decided to start to work on this project where: create a website with an implementation of a database in the back end, using a language such as Python or Java and connected to a database, or using a programming language and use a file. The Problem with the web pages is security wise. Since the program is going to be in the internet the security is going to need to be very well coded and functional. Since a program that store stock, client order and client sensible data, cannot be hacked or used by anyone not authorised the security must be one of the aspect of the program where to spend a particular and good part of the programming side. Typically used to transport data in Client and server machine in website using AJAX or JS is the JSON technology. This technology, present a problem which could get exposed using a special attack called JSON Hijacking or JavaScript Hijacking. In quote how the website CAPEC (Common Attack Pattern Enumeration and Classification) describe the Hijack:

“An attacker gets the victim to visit his or her malicious page that contains a script tag whose source points to the vulnerable system with a URL that requests a response from the server containing a JSON object with possibly confidential information. The malicious page also contains malicious code to capture the JSON object returned by the server before any other processing on it can take place, typically by overriding the JavaScript function used to create new objects. This hook allows the malicious code to get access to the creation of each object and transmit the possibly sensitive contents of the captured JSON object to the attackers' server. There is nothing in the browser's security model to prevent the attackers' malicious JavaScript code (originating from attacker's domain) to set up an environment (as described above) to intercept a JSON object response (coming from the vulnerable target system's domain), read its contents and transmit to the attackers' controlled site. The same origin policy protects the domain object model (DOM), but not the JSON.” (CAPEC, JSON hijacking, 2020)

Since this problem can lead to Legal problem with client , I am not that confident JavaScript Skills and I don’t want to intercourse legal problem related to GDPR, see Up Government,2018, Gov.uk, [Online], [23 January 2020], Available at: <https://www.gov.uk/data-protection>.

Eliminated the web option, there were two other ways to undertake the project, one was using and learning an entire new Language, Python since is known as the most Data Science friendly language, the other was to use a known language in which I have more experience and confidence, Java. Why is Python the best language to write for managing Data? First in the last years the use of Python as a language increased constantly and passed the old used R. As can be seen on this diagram.

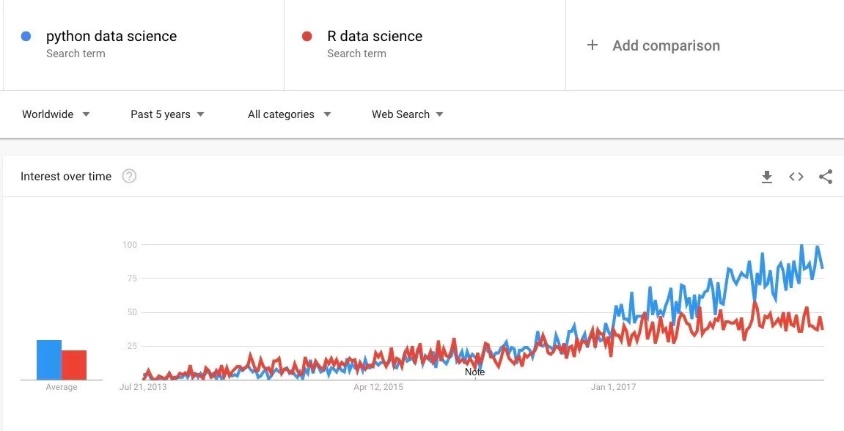


Figure 1:Python vs R through the years

As Said by Tim Peters, in the “The Zen of Python”, “Simple is better than complex”, and Python is known for his versatile and simplicity, making the language easier to learn and use. The many libraries that Python offer for data science are not comparable with the one offered by other languages and this and his simplicity give him a head start, compare to the other language used for Data science. Python is also reliable to process big chunk of data, using the Spark, a tool used by big company such as Intel, idea and EA. Even if the use of Python would make a better and most reliable program I will still stick with using Java. I’m going with Java because the time for producing this program and Graded unit is relative little, and since I want to deliver a working product and at least a decent documentation and using an already known and used language is going to make the efficiency and the final product better, I will use Java. Assumed I’ m going to use Java, I’m also going to use one of the Java Swing graphical tool to make GUI, called Windows Builder, to make the GUI for my Program. Windows Builder is a great add-on that can be added to eclipse, to create GUI. The fact that instead of coding, the tool gives the opportunity to click and drop objects, such as Labels, Buttons, Text Field and etc. Etc., and see right away the impact and how they look like in your program, make the GUI development easier and way more less stressful. Assumed the use of Java and Windows builder for the part of the program that is going to be stored on the client’s machine, now we are going to look thought the Database that I am going to use for this Project. As database I am going to use MySQL since is an open source and free system. Since, I learned the use of SQL using the oracle database, and I am going to learn a little bit of MySQL in our website class, this was the obvious for the database side of the program. For connecting the Java program with the database I am going to use a API, called JDBC. This API is oriented thought the use of database so perfect for working with MySQL. This is how I am going to work in the programming side of the project, while for how I am going to organise the work I am going to working the using the Agile Methodology. There are many advantage of using Agile instead of a Waterfall methodology on your project. First of all, the incremental implementation of features, this give the programmer the chance to test each individual use cases and divided the project in period cycles called iteration, instead of arriving at the end with a final project that may not work like in the waterfall methodology. Agile is also much more adaptable to changes and accident, since everything can be changed or reschedule if needed, while this is not really possible in the waterfall method. Another important advantage is the continuous interaction with the client, that is going to help the developer if changes on budget, requirements or if there are some misunderstood on the product. So in general Agile

methodology is so much better and efficient that the Waterfall one, and this is why I am going to use it.

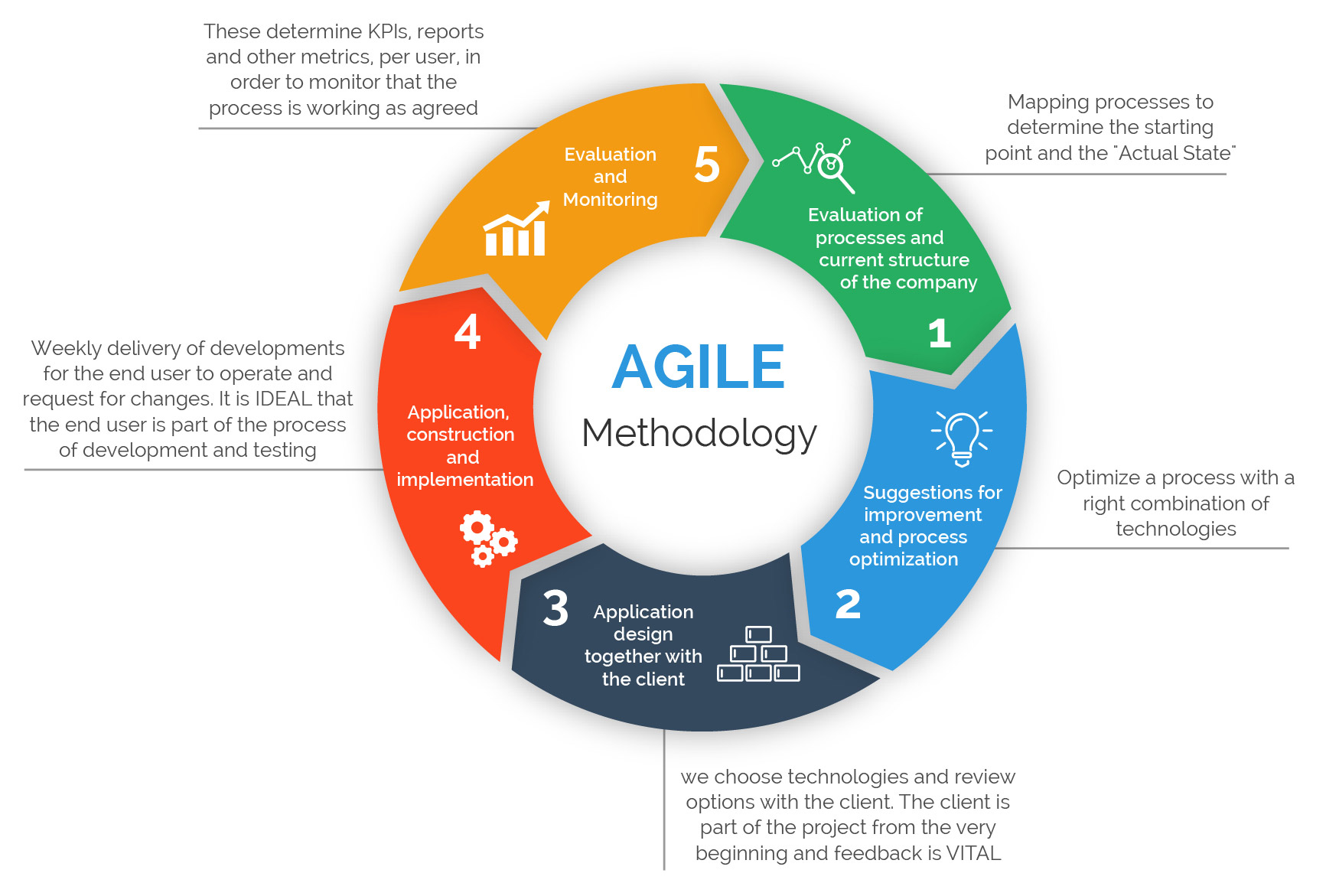


Figure 2: Agile Methodology

### 2.1.2 Natural Language analysis

I am going to perform an NLA (Natural language analysis) since is the first approach to the brief, and it’s going to help to work on my functional and non-functional requirements and on the top-level use cases. The NLA is the process of analyse and breakdown the brief in nouns and verbs to start having an idea of the actors, behaviours and requirements that are going to be involved in the program.

You are required to develop a small business application covering sales invoicing and stock control suitable for use by a typical small retailer operating in a UK context. It should produce a sales invoice for each transaction and update stock in real-time.

There should be a range of screen and paper reports detailing daily, weekly, monthly, quarterly, and annual sales analysis, VAT analysis, stock turnover, and profitability. There should be a suite of stock reports including reorder report and stock outages. General on screen display of customer, product, supplier and invoice information including transaction logs should be available with option for hard copy.

The application should be able to operate on modestly specified equipment running Windows XP or newer or Linux or Mac OS X. Either a console (Text based) application or a GUI application is acceptable.

There is no requirement to implement data storage using a Database Management System but it is not excluded either.

Verbs

Nouns

**Nouns**

1.small business application- The target that application is supposed to satisfy

2.sales invoicing-one of the option of the application, tell the client about a service or how much they owe to the company

3.stock control-application option that give the business to update their stock in real life

4.sales invoice for each transaction-the details for the client/staff of every transition appended

5.stock in real-time-option to update the stock in real time

6.a range of screen and paper report- the report that the app/database should generate

7.daily, weekly, monthly, quarterly, and annual sales analysis-the different type of report that should be generated

8.VAT analysis-the analysis of the change of the VAT ratio, another application feature

9.Stock turnover-another database functionality that should be included in the report, about how many times stock is finished and ordered

10.profitability-how much you profit by selling any sort of product

11.suite for stock report-stock report should be another option

12.report-report in general should be tracked and created by the application and database

13.stock outages-track of stock movement and when stock is finished should be included in the report

14.screen display of customer, product, supplier and invoice information including transaction logs- all this functionality should be available on the report display on screen

15.with option for hard copy-there should be an option to print the report

16.The application- the final product

17.modestly specified equipment-the equipment where the application should be able to run in

18.Windows Xp or newer or Linux or Mac OS X-the machine where the application should run

19.console text based application-the type of application that could be develop, console text based

20.GUI application-the other type, GUI based

21.data storage using a Database System Management-not require but could be helpful to connect a database

**Verbs**

1.develop-what should I develop, that app and the database

2.covering-what the app should be able to cover

3.use by-what the target of the app is likely to be

4.should produce-what the application/database should be able to produce

5.update-what the application/database should be able to update

6.should be-the report that should be included in the application

7.detailing-the details that this report should be displaying

8.should be-other functionality that the app/database should have

9.reorder-the functionality to reorder report or stock outrages in the database/application

10.should be available a hard copy of the report should be available

11.should be able to operate-the generic type of machine that the application should run on

12.running-More specific details of what the machine is likely to be

13. is not excluded-Database is not request but not excluded either

### 2.1.3 Functional/Non-functional requirements(initial)

Functional requirements and non-functional requirements are the features and behaviour that the program should have. The functional are related to the programming side, while the non-functional are related to the other specifications of the program, they don’t need to be implemented.

**Functional requirements**

1.The program needs to be able to update stock in real time

2.The program needs to be able to produce sales invoice for each transition and stock update

3. the Program needs to produce a wide variety (daily, weekly, monthly, quarterly, annual) screen and paper report for sales analysis, VAT analysis, stock turnover and profitability

4.The program needs produce reports when stock needs to be reorder or are close to be finished

5.The program needs produce in screen display of customer information, product transition info, supplier info, and invoice transition.

**Non-functional requirements**

1. The program should be suitable for a UK small retailer.

2. The program should work on machine like Windows XP, Linux and Mac OS x or newer.

3. The program should be either a console application or GUI application.

4. The program should have a database connected to or a file to store information in.

### 2.1.4 Use case Diagram(initial)

Creating use case diagram is essential for the interaction with the client. Since supposedly they client is not going to be an expert about programming so we are going to need to approach them with something that can be easy to perceive and understand. The use case diagram is one of the most important and easy to understand diagram, since the use of actor, representing costumers/staff/devices and the simplicity of the diagram make it understandable to everyone.

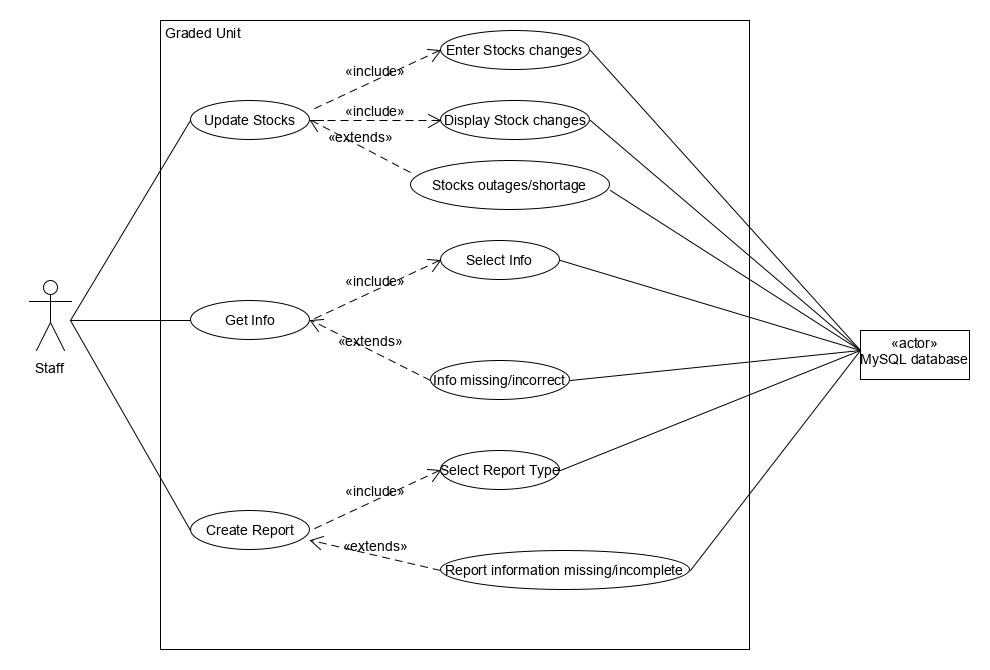


Figure 3: Initial use case diagram

### 2.1.5 Use cases description(initial)

In the use cases description, I am going to explain the functionality and the sequence with the program is going to work. This are important because they give an initial idea to the client and the programmers of how the program is going to work.

1. **Update Stocks-** Use case for updating the stock when an order come in or a client place an order. It’s going to be a menu option that the staff could select
2. **Enter Stocks Changes-**Its going to get activate when the staff select the update stock menu option. It’s going to have an option to select if an order came in or if an order was placed by the client. Its s going to get complete by the client/supplier info and quantity and type of stocks changed
3. **Display Stocks changes-** A Message showing details about the stock change
4. **Stocks outages/shortage-** An error message displayed by the system when stocks are close to outages or empty.
5. **Get info-** A menu option to get the info about client, supplier or transition
6. **Select Info-** With the select info it’s going to be possible to decide what kind of info are needed and read them using an ID
7. **Info missing/incorrect**- Error message show if information cannot be retrieved from the database
8. **Create Report-**Menu option to select the option to create report
9. **Select Report Type-**Select the type report wanted.
10. **Report information missing/incomplete-**Error message If the report cannot be created for the missing of information or the presence of corrupted information

### 2.1.6 Client Interview

Interview are needed and essential to undertake a project with the agile methodology. Interview are important to keep in track with the client of what they would like the project to be and what could be changed and improved.

**Interview Wednesday 29th January 2020**

**1. Q:** *What type of business and what type of product is your business aimed for?*

**A: Our business is a Fruit and vegetable whole shop, which buy products by supplier and receive call by client to place in orders**

**2. Q:** *What kind of functionality would you prioritize on the program?*

**A: The Main functionality that we want you to prioritize are the function to update stocks, to get the information about the clients and the suppliers**

**3. Q:** *What kind of report would you prioritize first?*

**A: The report that we really need to work are the one about stock changes, the weekly and monthly report about our business**

**4. Q:** *Can you explain me the process of the placement of an order by a client?*

**A: the info about the client that we want to store are an ID, the name of the company, their phone number and their e-mail**

**5. Q:** *Can you explain me when you consider your stocks to be close to outages?*

**A: We consider our stocks to be close to outages when the number of it Is less or equal to 8. We want the stocks to be not able to be ordered anymore when it is lower than 3.**

**6. Q:** *What kind of information would like to store about products, suppliers and transitions?*

**A. We would like to store a product ID, the product name, a description of it and price. For the suppliers we would like to store the same info we are storing for the client and for the transition the quantity and products names and the total price of the order**

**7. Q:** *This is my initial idea, does something to change or improve come to your mind (show use case diagram) (see use case diagram(initial)?*

**A: Your idea looks fine and represent what we would like the final product to be**

**8. Q:** *Would be fine to schedule a meeting every two weeks to update you about the project and asking some questions?*

**A: Yes, is fine**

### 2.1.7 Background research

Since my project consist on creating and develop an application for a fruit and vegetable shop, I had to make some background research of what others shop and other wholesale shop look like to have an idea of who selling this kind of product looks like. For a software developer, it is very important to not just have a knowledge about the programming side and language side of the program but also to have acquire knowledge and understanding about what the market of this kind of business require and what other similar shops looks like. First able to understand is how a wholesale company works. Most important part is how our prices should be. Looking online a learning from other wholesale retailer their price a x2.5 the price of production, so if for example an apple tree cost me 40 pound and give me 100 kg of apple a year the price for kilogram of apple should be around 1pound per kilogram. The other thing that retailer shop do is to tell a suggesting retail price which is usually around 2x the price they are buying in it from us. Looking through this would be another stored information that is going to be stored in the program. VAT taxes should not be included in our costumer price since is going to be the retailer to add them on the product. Another thing that I understand looking through some wholesale retailer shop like , [MXWholesales](https://www.mxwholesale.co.uk/), is that if you buy more of a certain product , a discount should be applied. For example, if someone buy 1 kg of apples should pay more per kg that someone that is going to buy 100 kg of them. Looking through veggie and fruit shop another thing that should be clarified with the client is how the shop is going to sell the merch. There are mainly two way to sell products in wholesale vegetables shop, by items or by kg/pound. I would suggest since the shop is a wholesale one to use kg, because dealing with counting every single item, would probably not be ideal for the client and the peoples working at the shop. There could be the option to use both, like in [George Perry](http://www.georgeperry.co.uk/) shop but I think would just be confusing for clients and staff members. Another option to look thought for the business is that there are going to be different product with similar or same names, Indeed in a shop for wholesale the product that are going to be available are going to be many , such as different variety of the same type of fruit and vegetable ,that is going to be essential for the customer to understand what are the difference between the products and what are they going to acquire. Therefore, adding a small description of the product, origin, type and quality would be essential. This should be added by the staff when a new product come in and displayed and told at the client when a purchase of the product is made.

### 2.1.8 2nd Interview(clarification)

After looking through website and had my background research, I get back to the client to ask about feature that I looked and find online that could be helpful for the program.

1. **Q:** *Would adding a suggested retail price would be a function that you look to have?*

**A: We are not interested to add this feature to your product**

2. **Q:** *How much would a discount price be for retailer that buy more than tot kg of a product?*

**A: Our Business idea does not provide any kind of discount for big purchases**

3. **Q:** *How is the shop going to sell the product, by kg/pound or by items?*

**A:We are going to sell our product by Kg**

### 2.1.9 Aim of the Project

Following the [interview with the client](#_2.1.6_Client_Interview) and the [background research](#_2.1.7_Background_research) of the project, I have a better understanding of what the final product would look like. The application is going to help to the client to keep track of the different transaction that the shop must deal with. The app is going to work that when a client call in or a new bunch of stocks came in the staff is going to use the app to manipulate the database through the “Update Stock”([see Use case 1](#_2.1.10_Use_case)) function and it is going to create report for each transition. The information that the app is going to store about the client are going to be as specified by the Client ([see question number 4](#_2.1.6_Client_Interview)), supplier and product ([see question number 6](#_2.1.6_Client_Interview)). The program is also going to generate other report, and as specified by the client, just monthly and weekly ([see question number 3](#_2.1.6_Client_Interview)). Other new feature that I picked up going through my research, is to add a suggested retail price that’s going to be 2x+VAT the selling price and to add a discount to the purchase of large bunch of stocks,but after my second client meeting we have decide that this features are not going to get implemented(see [question 1-2](#_2.1.8_2nd_Interview(clarification))). After having a look through research and interview something in my use case description and functional/non-functional requirements Is going to change, while my use case is going to basically stay the same. My personal aim for this project is to improve as a programmer in java and MySQL, the two languages that I picked for this project. I hope also that this work is going to help me to improve my skill in creating report and use the correct notation in similar cases. I also hope this project is going to give me a better understanding of what working with a client and a future work could possibly look alike, with all the deadline, problem and situation that can be created by this case.

### 2.1.10 Functional/non-functional requirements(revised)

(see [2.1.3 Functional/non-functional requirements(initial)](#_2.1.3_Functional/Non-functional_req))

**Functional requirements**

1.The program needs to be able to update stock in real time when a new bunch of stock arrive, storing number of stocks arrived(in kg), type and information about the supplier (see functional requirement 2 below).

2. The program should store supplier information such as an ID, supplier name, an email and a phone number.

3. The program needs to be able to update stock in real time when a client order arrives, storing quantity of stocks ordered (in kg), type and information about the client (see functional requirement 4 below).

4. The program should store client information such as an ID, company name, an email and a phone number.

5.The program needs to be able to produce sales invoice or report for each transition and stock update, storing supplier or client information and update information or transition. (see functional requirements 1,2,3,4 above)

6. the Program needs to produce weekly, monthly and stocks changes reports.

7.The program needs produce on screen error message when stock needs to be reorder (less than 8) or are close to be finished (less than 3).

8.The program need to store product information such as Type, quantity, price, quality and origin.

5.The program needs produce in screen display of customer information (see functional requirements 4), supplier info (see functional requirements 2).

**Non-functional requirements**

1. The program should be suitable for a UK fruit and vegetable whole shop retailer.

2. The program should work on machine like Windows XP, Linux and Mac OS x or newer.

3. The program should be either a GUI application.

4. The program should have a database connected.

### 2.1.11 Use case diagram(revised)

(see [2.1.4 use case diagram](#_2.1.4_Use_case))

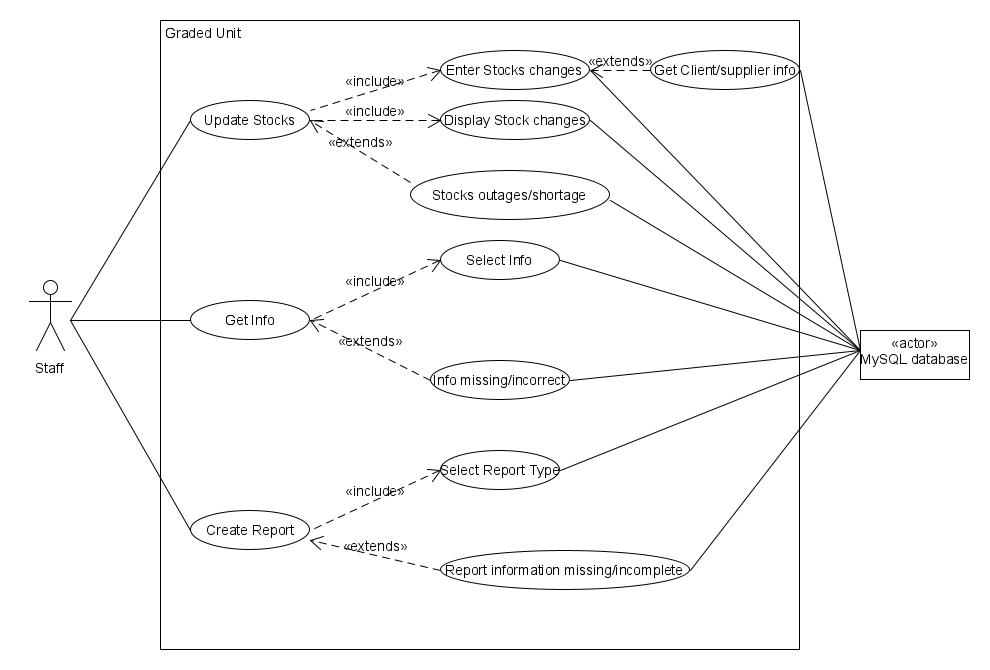


Figure 4: Revised use case diagram

### 2.1.12 Use case description (revised)

(see [2.1.5 Use case description(initial)](#_2.1.5_Use_cases(initial)))

1. **Update Stocks-** Use case for updating the stock when an order come in or a client place an order. It’s going to be a menu option that the staff could select.
2. **Enter Stocks Changes-**Its going to get activated when the staff select the update stock menu option. It’s going to have an option to select if a supplier order came in or if an order was placed by the client. Its s going to ask for an id of the client or supplier and, if the client or supplier does not have one generate one, storing other information such as Company/supplier name, phone number and email (see use case 3). After this or If the client and supplier are already in the system is going to ask for quantity(in kg) and type of stocks changed
3. **Get Client/supplier info-** It’s going to be used if the client or the supplier are not in the system yet. It’s going to auto generate and ID and ask information about the client/supplier such as Company name, email and phone number. After the client/supplier is added to the system the app is going to go back to the Enter stock changes (use case 2) and allow the staff to make a stock change.
4. **Display Stocks changes-** A Message showing details about the stock changes, using the client/supplier information and the transition ID, number of items, type of items and total price
5. **Stocks outages/shortage-** An error message displayed by the system when stocks are lower than 8, the stocks is going to get lock if the number of it is less and equal to 3.
6. **Get info-** A menu option to get the info about client (ID, company name, email, phone number), supplier (ID, supplier name, email, phone number) or transition, info is going to get using IDs.
7. **Select Info-** With the select info it’s going to be possible to decide what kind of info are needed and read them using an ID
8. **Info missing/incorrect**- Error message show if information cannot be retrieved from the database
9. **Create Report-**Menu option to select the option to create report
10. **Select Report Type-**Option to select the type report wanted, weekly or monthly
11. **Report information missing/incomplete-**Error message If the report cannot be created for the missing of information or the presence of corrupted information

### 2.1.13 Resources and material

Resources and material are a list of objects, programs, devices and everything you need during the different stages of the project.

1. **Laptop** – I ‘m going to use my own laptop to do most of the report and program, especially when I am home
2. **Forth Valley College desktop computers**- I’m going to use the Forth valley computers for working on the project when in class.
3. **University of Stirling library desktop computers**- I’m going to use the library computers when I ‘m going to need extra silence to focus in.
4. **My Forth Valley College account**- I’m going to use my account to access to the computers in FVC
5. **My University of Stirling account**- I’m going to use my account to access to the computers in UoS library.
6. **USB stick**-I’m going to store my report and project in the USB so I can carry it with me everywhere
7. **Google Drive**- I’m going to also store my report in google drive to add an extra backup if I lose my USB or something goes wrong inside my laptop HD
8. **GitHub**- I’m going to also save my program on GitHub, for the same reason that I also use google drive
9. **Microsoft Word**-I’m going to use Microsoft word to write down my report
10. **Microsoft Project**-I’m going to use Microsoft project to write down my project
11. **Eclipse IDE**-I’m going to use eclipse to program in Java
12. **Windows Builder**-I’m going to use windows builder libraries to build my GUI
13. **JDBC**-I’m going to use JDBC to connect my database to Java
14. **MySQL**-I’m going to use MySQL for my database
15. **Xamp**-I’am going to use xamp to manage the server
16. **PhpMyAdmin**-I’m going also to use it for managing the server
17. **Google Chrome**-I’m going to use google chrome for making research on the web
18. **UMLet**-I’m going to use UMLet to create my diagrams
19. **Moqups**- website that I’mgoing to use for create dynamic prototype and wireframes
20. **Pen**-I’m going to use pens to write some notes
21. **Paper**-I’m going to write my notes on it.

## 2.2 Project Stage

The Project is where you schedule and organise how your going to manage your time and your going to divide ad allocate the time of your project. Having a project to show to the client is important , to make the client understand when they should except material to be delivered and meeting to be held

The Documents can be found here.

## 2.3 Solution Stage

*//Structure: Class*

*//Behaviour: Communication, Sequence, Activity and State*

Modelling a solution before implementing one is essential. For the same principle that you are not going to build a house without making a plan and draw some sketches, building a software needs to be backed up by the right kind of documentation and design. Planning the solution of a project require different kind of documentation. First thing that needs to be done is the NLA (see [2.1.2 Natural Language Analysis](#_2.1.2_Natural_Language)), which is the first approach and analysis of what the problem given by the client is. The NLA is usually performed on the project brief that is one of the first documents given by the client. After the NLA is performed the next step is to identify the top level use cases(see [2.1.12 Use cases description](#_2.1.12_Use_case)) and create the Use case diagram(see [2.1.11 Use case diagram](#_2.1.11_Use_case)) , with the help of this diagram get back to the client and discuss on what to change about your initial idea. After this part is where the planning of the solution get in depth. With the help of the top-level use cases is important to write and describe the use cases more in depth. Indeed a fully dressed use case description (see [2.3.1 Fully dressed use case description](#_2.3.1_Fully_Dressed)) is going to describe, how any of this use cases are going to work. This description is going to describe and break down the different functions of the program, and how the logic behind is going to be. Another step to get close to plan the solution of the program is to start thinking how the program classes will probably look like. The first step to break down the classes is to use a tool called CRC (Class-responsibility-collaboration) cards (see [2.3.2 CRC cards](#_2.3.2_CRC_cards)). This brainstorming tool is going to give the first draft of what and how the classes in the program are going to be. After this side of the solution planning of the program, still really correlated to the brief, it’s time to start use diagrams to display what our idea and program is going to be. Diagram cannot, however, be created and designed randomly but have to follow specific rules and constrain. For this project, I’m going to use the UML standards to produce my diagrams. Why UML? Because UML diagrams are the one that get closer to represent a simplification of what the project is going to look like, helping you understand complex systems and giving the opportunity to specify structure and behaviour of it. UML diagram are divided in two big category Structure Diagram and Behavior Diagram. The Structure Diagram show what the structure of the program looks like and represent the relation between different parts of the program. Structure diagram do not show how the program and system behave but it may show where and how they are related in the program in an abstract way. Example of this type of diagram are Class, Object, and Package, Model, Network architecture and Deployment diagram. For my project I am going to use Class diagram and Model Diagram. Class diagram(see [2.3.4 Class Diagram](#_2.3.4_Class_Diagram)) is the best way to show the structure of the system and the interaction between the classes, Model diagram(see [2.3.3 Model Diagram](#_2.3.3_Model_Diagram)), instead, show an abstract view of the entire system , so the connection between the different software model , like database , GUI and program. Other diagram such as Network architecture, that show the structure of the network, or the Object diagram where not take into account since obsolete , in case of the object one or in my opinion , not relevant for my project. Another diagram that is going to be develop related to the Structure that is not included in UML is an ERD(See [2.3.8 Database Design](#_2.3.3_Database_Design)). ERDs are diagram used to explain tables and their relationship between each other on a database. This diagram is essential since the use of a back-end database to process data for the business. Behavior diagram are used to show how functions of the program are going to work, and how they are going to effect the system.Example of this diagrams are Use case diagram, Activity, State machine, Interaction, Sequence, Communication and timing. The diagram that I’m going to use are the Use case diagram , which I already have develop(see [2.11 Use case diagram](#_2.1.11_Use_case)), the Sequence diagram (REFERNCE TO SEQUECE) that describe specific interaction between object during different functions, Activity diagram(see [2.3.6 Activity Diagram](#_2.3.6_Activity_Diagram)) , that describe the function of functions, Communication diagrams(see [2.3.5 Communication Diagram](#_2.3.5_Communication_Diagram)) that describe the relationship and interaction between different program and software elements(like database with the program. Other diagram where not take into account because similar to the one selected or because useless for my type of project. Last but not least is documentation related to the GUI. For the GUI , I am going to use Wireframes and a dynamic prototype. Wireframes (see [2.3.9 Graphical design & Client meeting](#_2.3.5_Graphical_design)) provide a visual example of what the GUI is going to look like and are great example to be show to the client to ask opinion and changes about the project. Dynamic prototype (see dynamic prototype [here](https://app.moqups.com/k2MJEHHUbO/view/page/aa9df7b72)) are the evolution of the wireframes. Indeed, the dynamic prototype is a clickable example of the wireframe that show how the GUI of the program is going to work, helping the client to understand the different interaction between the different wireframes, or pages.

### 2.3.1 Fully Dressed Use cases description

The fully dressed use case description is the description of how the main functionality of your program should be act. Even if it does not still use “Programming” terminology, this description should describe in English how the logic of the program is going to be and should help when programming is going to happen. Indeed, you should be able to program, at least function by function/use case by use case, just looking thought this description.

**Use case 1**

**Name:** Update stocks

**Primary Actors:** Staff, Database

**Description:** A member of the staff makes an update of the stock, following a stock outage an order or a new arrival of products.

**Pre-conditions:** The system is start-up, a client as placed an order or new stock has arrived, the menu option “update stock” was selected

**Post-conditions:** Stocks is updated

**Trigger Event:** The staff member selected “update stocks” in the menu

**Scenario:**

*Normal flow*

1. The Staff member select the Update stocks option
2. The Staff member select the type of stock changes
3. The Staff member select new client/supplier or old client/supplier

3.1 the staff enter information of the new client/supplier (see use case 2)

1. The staff member enters the requirements info about the stock changes and client/supplier making the change (see use case 3)

5. The stocks get updated on the database

6. The Computer show an on-screen message about the changes happened (see use cases 4)

*Extension Flow*

4. The information added are wrong

4.a. Error message for missing stock or low (See use case 5)

4.1.a The staff enter the wrong info

4.1.1.a The staff member receives a pop-up with an error message

4.1.2.a The app goes back to insert client info option

5. Error message for missing stock or low (See use case 5)

**Use case 2**

**Name:** Enter Stocks changes

**Primary Actors:** Staff, Database

**Description:** A member of the staff insert information to makes an update of the stock, following a stock outage an order or a new arrival of products.

**Pre-conditions:** The system is start-up, a client as placed an order or new stock has arrived, the menu option “update stock” was selected

**Post-conditions:** Stocks is updated

**Trigger Event:** The staff member selected “update stocks” in the menu

**Scenario:**

*Normal flow*

1. The staff member select if is a new client/supplier or an old client/supplier

1.a The staff member compile information of new Client/supplier (see use case 3)

2. The staff member select if is a client or a supplier ticking the right box

3.a The staff member inserts client ID

3.1.a The staff member inserts the product name requested by the client

3.2.a The staff member inserts the quantity of the product

3.3.a The Staff click the add button

3.b The staff enters the supplier ID

3.1.b The Staff enters the product name

3.2.b The Staff enters the provenance of the product

3.3.b The staff enters the Quantity of the product that was bought

3.4.b The staff enters the price of retail of the product per kilogram

3.5.b. The staff enters a brief description

3.6.b The staff select the type(vegetables/fruit)

3.7.b The staff press adds to add the product to the stock.

4. The product get added to the order or to the stock, and the product info are displayed.

5. the staff press the submit button

6.The Computer show an on-screen message about the changes happened (see use cases 3)

*Extended Flow*

3.a The client id is wrong

3.1.a the product is not found in the database

3.2.a the quantity is not a valid number, must be retype

3.3.a the product is shortage or out of stock (see use case 5) can’t be added to the order

3.b the supplier Id is wrong

3.1.b The product name is in the wrong format, must be retype

3.2.b the provenance is in the wrong format, must be retype

3.3.b the quantity is in the wrong format, must be retype

3.4.b the price is in the wrong format, must be retype

**Use case 3**

**Name:** Get Client/Supplier info

**Primary Actors:** Staff, Database

**Description:** The staff add information about a new client or supplier, they get add to the database and an ID is generated by the system

**Pre-conditions:** The system is start-up, a client as placed an order or new stock has arrived, the menu option “update stock” was selected and the new client/supplier button was pressed

**Post-conditions:** The staff can make an order/add stock from the new supplier/client.

**Trigger Event:** The button add new client/supplier was pressed.

**Scenario:**

*Normal flow*

1. The staff clicks the box for client or supplier
2. The Staff enters the company name of the client or supplier
3. The Staff enters a reference e-mail address for the client or supplier
4. The Staff enters a reference phone number for the client or supplier
5. The staff enter the submit button
6. The database generates an ID for the client/supplier
7. The new user gets added to the database
8. A pop-up message gets displayed with the ID of the user.

*Extended flow*

2. Company name of the wrong format, must be retyped

3. e-mail wrong format, must be retyped

4. Phone-number of wrong formats, must be retyped

**Use Case 4**

**Name:** Display Stock changes

**Primary Actors:** Staff, Database

**Description:** After a stock changes has been made a pop-up message get show whit the details and the assign ID to order or ne stock income.

**Pre-conditions:** A order has arrived, and the database has been changed. (see use case 2).

**Post-conditions:** An order Id is displayed, and the staff can get back to the menu

**Trigger Event:** Submit button has been clicked on the update stock change.

**Scenario:**

*Normal Flow*

1. An order gets placed
2. The order is added to the database
3. An order id is created
4. Order info and order ID are displayed on screen with a pop-up message

*Extension Flow*

1. Problem with the connection with the database and order cannot be place

**Use Case 5**

**Name:** Stock outages/shortage

**Primary Actors:** Staff, Database

**Description:** Error message when the stock is too low o

**Pre-conditions:** A change of stock was tried

**Post-conditions:** The Stockage can be changed and the product can’t be added to the order.

**Trigger Event:** A change of stock was tried

**Scenario:**

*Normal Flow*

1. The staff try to add a product to the order

2.a The Product quantity is lower than 8, a message telling the stock is close to finish will pop-up, the product will be added to the order anyway (if less than 5 kg)

2.b. The Product is lower than 3 the stocks get blocked and not added to the order, an error message will pop-up.

3. The staff get back to add other products or to finish the order.

**Use Case 6**

**Name:** Get Info

**Primary Actors:** Staff, Database

**Description:** The staff get some information that they need from the database

**Pre-conditions:** information is needed, and the staff pressed the button “Get Info” on the Menu

**Post-conditions:** The Staff get the information and get back to the Menu

**Trigger Event:** The Staff select the button “Get Info”

**Scenario:**

*Normal Flow*

1. The Staff press the “Get Info” button
2. The Staff select the type (see use case 7)
3. The staff receive the on-screen message with the info that he requested.
4. The Program get back to the menu page

*Extension Flow*

1. Information error (see use case 8)

**Use Case 7**

**Name:** Select info

**Primary Actors:** Staff, Database

**Description:** The Staff insert the data of the info it would like to display

**Pre-conditions:** The Staff selected “Get Info” button

**Post-conditions:** The information is displayed on screen

**Trigger Event:** The Staff selected “Get Info” button

**Scenario:**

*Normal Flow*

1.The staff select the type of information needed clicking the right type (Client, order, Supplier)

2. The staff insert the id correspondent

3. the database search of the information stored with that id

4. The database send the information to the program

5. The information get displayed on screen

*Extension Flow*

1. The id is incorrect (See use case 8)
2. The database can’t find the right ID (see use case 8)

**Use Case 8**

**Name:** Info missing/incorrect

**Primary Actors:** Staff, Database

**Description:** The information that the staff tried to obtain from the database are not existent or corrupted

**Pre-conditions:** The staff tried to get some Information from the database using the Get info (Use case 6) and Select type (Use case 7) use case

**Post-conditions:** The staff must re enter the information or get back to the menu

**Trigger Event:** An Information Was tried to get retrieve

**Scenario:**

*Normal Flow*

1. The Staff tried to retrieve information

2a. The id enter is invalid

2b. The Id is in the correct format

2c. The Information stored with that id are corrupted

3. The Error message pop up

4. The Program get back to the select type information

**Use Case 9**

**Name:** Create report

**Primary Actors:** Staff, Database

**Description:** Create a report, weekly or monthly, that show the business movement for that period.

**Pre-conditions:** The Staff decide to create a report, press the “Create report”, and select the type of report wanted

**Post-conditions:** A report get created and showed

**Trigger Event:** The Staff press the “Create report” button

**Scenario:**

*Normal Flow*

1. Staff select the “Create report Type”
2. Staff select report type (see use case 10)
3. The report gets showed on the screen

*Extension Flow*

2.Report information missing or incomplete (see use case 11)

**Use Case 10**

**Name:** Select Report type

**Primary Actors:** Staff, Database

**Description:** The Staff select the type of report wanted

**Pre-conditions:** The Staff selected the report for

**Post-conditions:** The Report get display on screen

**Trigger Event:** The Staff selected the “Create Report” Button

**Scenario:**

*Normal Flow*

1.The Staff select the type of report (weekly or monthly)

2. The Database look at all the transactions in the last 7 weeks or 30 days.

3. The database send the information to the app

4. The information are show on the screen

*Extension Flow*

1. the information are missing or corrupted (Use case 11)

**Use Case 11**

**Name:** Report information missing/incorrect

**Primary Actors:** Staff, Database

**Description:** The information that the Staff tried to acquire are missing or corrupted

**Pre-conditions:** The Staff has clicked the “Create Report” button and selected the type of report

**Post-conditions:** The App gets back to the menu

**Trigger Event:** Selection of the report type (use case 10)

**Scenario:**

*Normal Flow*

1.A type of report get Selected

2a. The information for that time period are incomplete or corrupted

2b. There is not enough time from the start of the database

2c. The database has a problem

3. An error message get shown

4. The Application get back to the menu.

### 2.3.2 CRC cards

CRC card (Class/responsibility/collaborator) cards are a useful tool to use when developing an Object-oriented program. The CRC describe what the class was supposed to do, their behaviour and responsibilities and the other class they interact with, their “Collaborators”. This is very helpful when laying down the class diagram, because it gave the first view at what the class are probably going to be.

|  |  |
| --- | --- |
| **Class:** Client | |
| **Responsibilities** | **Collaborators** |
| Create statement for client | Order  DatabaseManager |

|  |  |
| --- | --- |
| **Class:** Order | |
| **Responsibilities** | **Collaborators** |
| Create statement for orders | Client  DatabaseManager |

|  |  |
| --- | --- |
| **Class:** New Stock | |
| **Responsibilities** | **Collaborators** |
| Create statement for new coming stock | Supplier  DatabaseManager |

|  |  |
| --- | --- |
| **Class:** Product | |
| **Responsibilities** | **Collaborators** |
| Create statement for new products or old product | New Stock  Order  DatabaseManager |

|  |  |
| --- | --- |
| **Class:** Supplier | |
| **Responsibilities** | **Collaborators** |
| Create statement for supplier | New Stock  DatabaseManager |

|  |  |
| --- | --- |
| **Class:** Report | |
| **Responsibilities** | **Collaborators** |
| Create statement for creating report | DatabaseManager |

|  |  |
| --- | --- |
| **Class:** DatabaseManager | |
| **Responsibilities** | **Collaborators** |
| Communicate with the Database and manage the interaction with it | Client  Order  New Stock  Product  Report  Supplier  MenuUI |

|  |  |
| --- | --- |
| **Class:** MenuUI | |
| **Responsibilities** | **Collaborators** |
| It’s the menu GUI, should communicate with the other panels | DatabseManager  EnterStockChangesUI  GetInfoUI  SelectReporUI |

|  |  |
| --- | --- |
| **Class:** EnterStockChangesUI | |
| **Responsibilities** | **Collaborators** |
| GUI to change the stocks | MenuUI  NewUserUI |

|  |  |
| --- | --- |
| **Class:** NewUserUI | |
| **Responsibilities** | **Collaborators** |
| GUI to enter a new client or supplier | EnterStockChangesUI |

|  |  |
| --- | --- |
| **Class:** GetInfoUI | |
| **Responsibilities** | **Collaborators** |
| GUI to select information | MenuUI  ShowInfoUI |

|  |  |
| --- | --- |
| **Class:** ShowInfoUI | |
| **Responsibilities** | **Collaborators** |
| Show the information retrived from the Database | GetInfoUI |

|  |  |
| --- | --- |
| **Class:** SelectReportUI | |
| **Responsibilities** | **Collaborators** |
| Select the report needed by the client | MenuUI  ShowReporUI |

|  |  |
| --- | --- |
| **Class:** ShowReportUI | |
| **Responsibilities** | **Collaborators** |
| Show the selected report | SelectReportUI |

### 2.3.3 Model Diagram

When program start to get more complex, a good programming habit that should be follow is to divide logical and business model from the user interface, and in general to divide the program in smaller units, aka packages. Using this technique, the model of this program must be identified and explained, and the easier way to do that is package diagram or model diagram. The use of different package also gives your application flexibility, since the java programming core could be use in different platform, changing only the GUI part. This diagram defines how the application is modelled and it’s dependence between the different packages. I divided the application in three main layers: A GUI one, that contains the graphical user interface package, the business one, which contain the application bit, and the Data layer, with contain the package that communicate to external database.

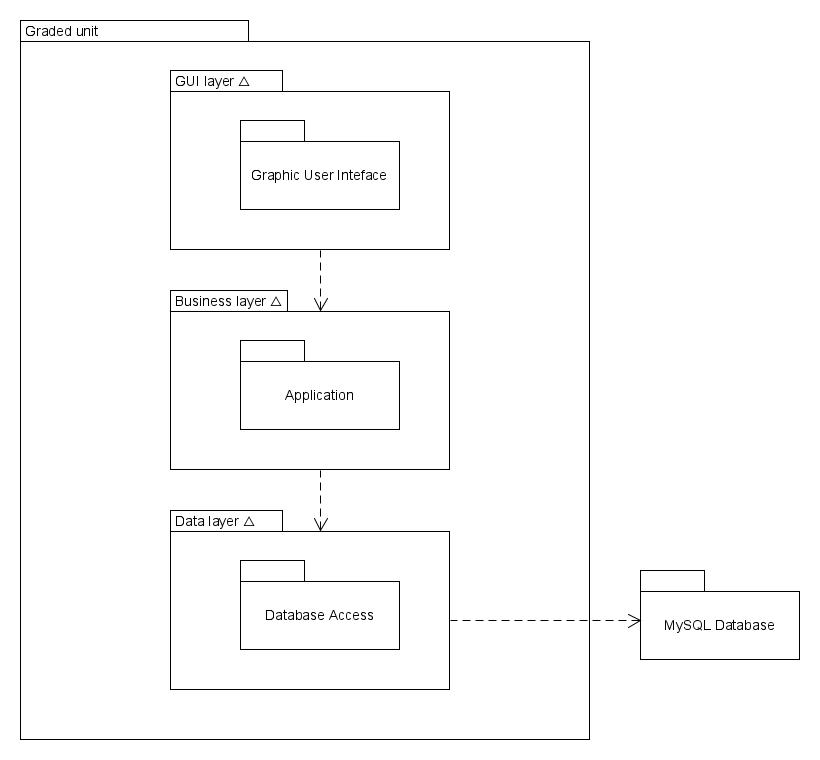


Figure 5: Model diagram

### 2.3.4 Class Diagram

Class Diagram explain in and display, how the class interact between each other and their behaviour. This is essential to describe the structure of the program. The purpose of the class diagram is to show the static structure of the system , to help in the creation of other diagram in the dynamic model, giving them a basic notation to work with, to help other people to understand how the program is laid down without having to look through the code. So, a class diagram is formed by classes, which are divided in attribute and methods. Attributes are value or data members of the class, while methods are behaviour that change the program flow. Other part of the class diagram is the relationship which have different type and strength. The weakest relationship between classes is Dependency, which is represent by a dashed line open arrow, which define that one of the classes could change the other class. Another of the weak relationship is association, which is draw as a solid line connecting the two classes, that means that a structural link between the two classes exist. The middle of the road one is aggregation, that is represent with unfilled diamond and a solid line, which represent that one class is part of the other, so an object may be stored in the other class. The second strongest is Composition which as the same style of aggregation but with a filled diamond and mean that the class is completely depend by the other, class 2 exist only is class 1 does. The strongest class is inheritance, which mean that a class is a subclass of a main one, for example dog is a subclass of the animal class. Using the classes I have identified in the CRC cards I started laying down a class diagram.

**The Packages**

As said before on the Model diagram section ([2.3.3 Model Diagram](#_Model_Diagram)), I’m going to divide the program in different packages. One for the GUI, one for the application core, and one to the access to the database. So, in this image below I displayed how the package are divided and how the class are distributing between them. For the naming I am following the java convention. First the “company” or institution domain, uk.ac.forthvalley, after the project bit, gradedunit, and after the different package purpose, application/gui/dataaccess. This how the package division looks like.

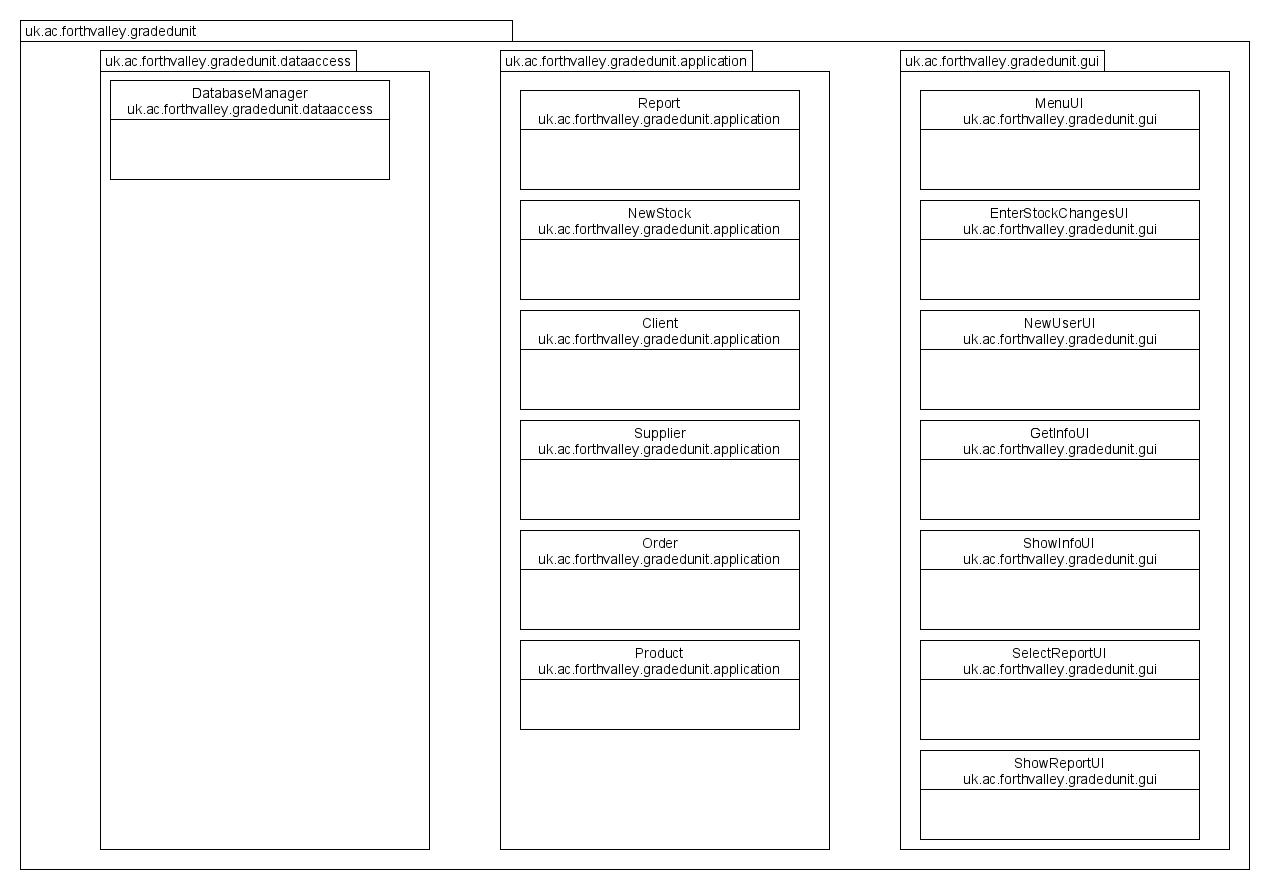


Figure 6:Package division

**Class (Attributes and Behaviour)**

After having breakdown the structure in packages and classes (with the [CRC cards](#_2.3.2_CRC_cards)) it’s time to have a look at how the classes are going to be develop. As said before classes are formed by properties or attributes, and behaviour or methods. Class in object-oriented programming also, almost always, has a “Special” method, also known as constructor, which define the blueprint of the object. So, in this stage before looking through the relationship between classes and the diagram, I am going to break down the classes. Here some notion about what some stuff in the class diagram means. The – sign represent private attribute or methods (cannot be accessed or modify outside the class), the + sign represent a public attribute or method (can be accessed and modify outside of the class), and the # sign a constructor. The data type of methods and attributes is writing and specify on the side , after the : sign. Set methods or Setter are the one the set a value for the attributes specified , they are a void method , since they are used to set the value and they don’t need to return anything. Get or Getters methods are the one use to access to the value from outside the class, this is why they are public and not private. toString methods are used to display and format the class attributes.

**uk.ac.forthvalley.gradedunit.application**

**Client Class**

This class is going to create the statement that call or insert information about the client from the database. The attributes are going to be private but there are going to be called in the Database manager using methods , calling a prepared statement , and insert the information get from the GUI, passing the value true the function in the client class.

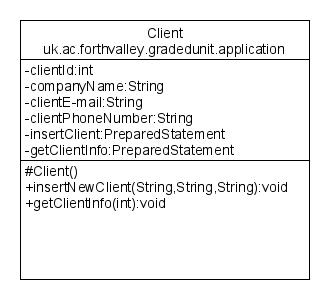


Figure 7:Client Class

**Supplier Class**

Supplier class is going to be very similar to the client class. It’s going to have the same function and idea , since the two users are very similar.

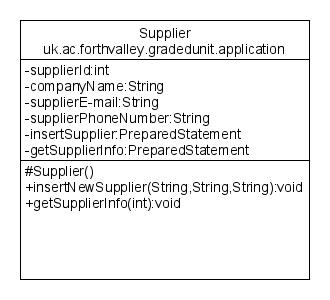


Figure 8:Supplier Class

**Product Class**

This class is going to deal with the adding and removing of product from the database. The methods for removing and adding products , and the statement , are going to be called in the Order class methods and New Stock class methods. Information about product are not needed to be display singularly .

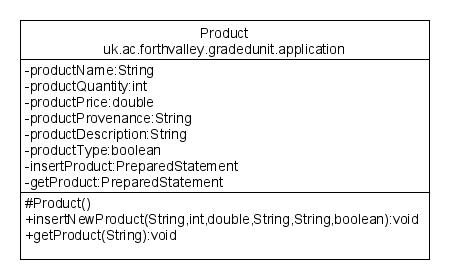


Figure 9:Product Class

**Order Class**

The Order class is going to manage the statement to view order and add new order. Inside the insertNewOrder method , it is going to be called the getProduct, using the name to call the product and remove it from the database.

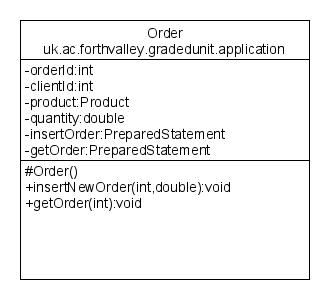


Figure 10:Order Class

**NewStock Class**

Just like the order class the new stock class is going to interact with the product class to call some of their specific statement.

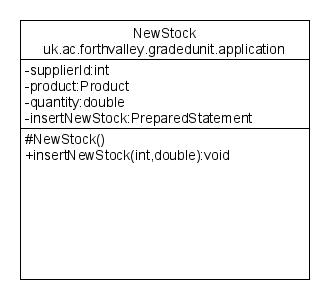


Figure 11:NewStock Class

**Report Class**

The Report class has as only two attributes , the two prepared statement, since no information from the user need , except for the requested report that is going to be selected using button in the GUI.

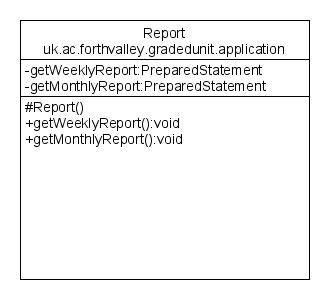


Figure 12:Report Class

**uk.ac.forthvalley.gradedunit.dataaccess**

**Database Manager**

This class is going to connect to the database, call and use the statement getting the info from the GUI, being the getaway between application,GUI and database. Having ma statement private and in other classes is going to help the abstraction of the program, so the statement and information about the table , and attributes are not going to be visible.

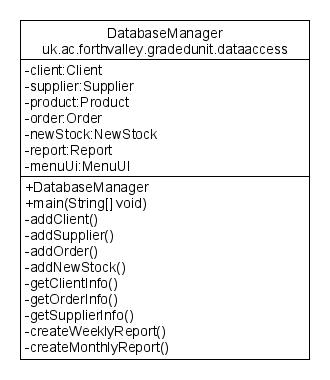
****

Figure 13:DatabaseManager Class

**uk.ac.forthvalley.gradedunit.gui**

For the GUI I am going to have a JFrame, or main page, MenuUI() which would inheritance all the other JPanel , for the other GUI parts. This class are going to store all their action and component in constructor and main method, so not explanation class per class Is needed. The MenuUI is going to have an initialize() method , because is a JFrame and not a JPanel like everyone else. To look how they are going to look in the design style , go to Wireframes in the [design](#_2.3.6_Graphical_design) bit.

**MenuUI**

(see Figure 63:Menu wireframe for design reference)

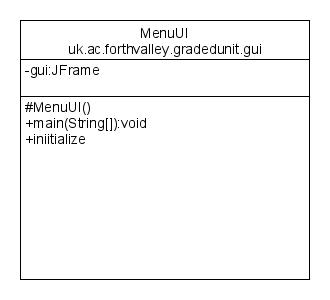


Figure 14:MenuUI Class

**EnterStockChangesUI**

(See Figure 67:Enter Stock Changes wireframes/Figure 64:Update Stock wireframe for design reference)

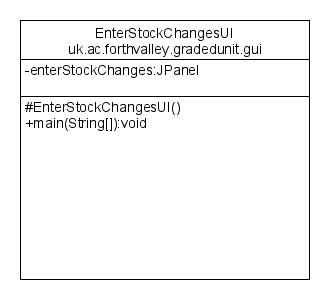
****

Figure 15:EnterStockChangesUI Class

**NewUserUI**

(see Figure 65:New Client/Supplier Wireframe for design reference)

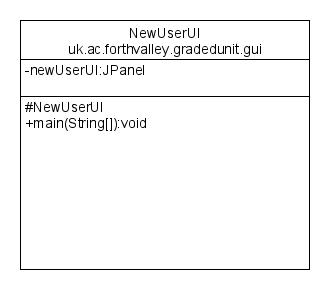
****

Figure 16:NewUserUI Class

**GetInfoUI**

(see Figure 70:Get Info wireframes for design reference)

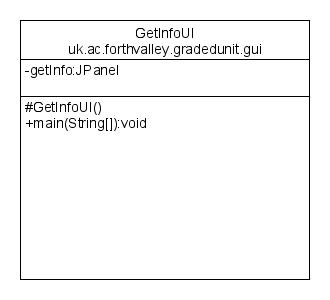
****

Figure 17:GetInfoUI Class

**DisplayInfoUI**

(see Figure 71:Select Info wireframes for design reference)

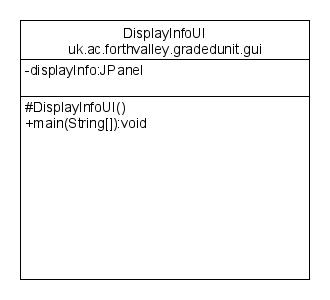


Figure 18:DisplayInfoUI Class

**SelectReportUI**

(see Figure 73:Select Report type wireframe for design reference)

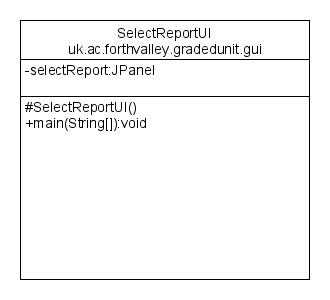
****

Figure 19:SelectReportUI Class

**DisplayReportUI**

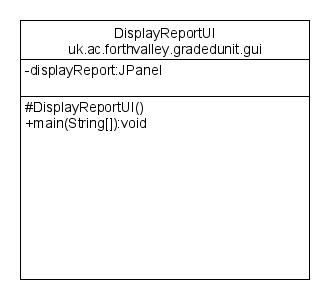
****

Figure 20:DisplayReportUI Class

**Relationship between classes and packages**

The different packages in the classes are going to interact in different way , and inside the packages there is also going to be interaction between each other. In this section I will look through what kind of interaction are happening in the program and how they are going to be represented in the class diagram. As explained before in programming there are different kind of interaction and they mean different thing and represent in different way. For more detail’s explanation look into the introduction session of this paragraph.

**Interaction in the .application package**

In the application package there are two interaction between the different classes , the one between Order and Product and the one between Product and new Stock.

**Order-Product**

The Order needs to various time the product statement and getProduct() method to remove the products from the database and add them to the order. Since It’s not an object that is stored or use , and a method is just called an association relationship should be applied. The Product are part of the Order, as can be seen in the diagram, and the number 1,… means that more Order and Product can be part of each other’s.

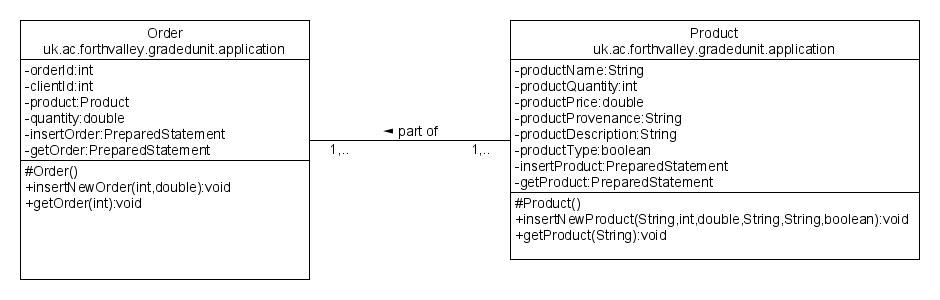


Figure 21:Order-Product association

**NewStock-Product**

Same as the order one, product to be added to the database must have to pass through the NewStock class. So, for adding new stock into the database the function insertNewProduct() must be called in the NewStock class. So same has the other relationship , this is going to be an association. Product are also part of the NewStock coming ,and multiple newstocks can be associated with multiple product and vice versa.

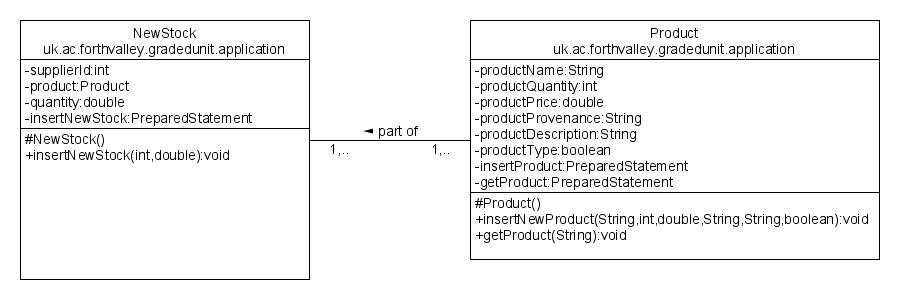
****

Figure 22:NewStock-Product association

**Interaction between classes in .application and .dataaccess**

All the application packages classes must be use and manipulated by the Database Manager class , which also manage the connection between the program and the Database. All the classes are going to be aggregate with the DatabaseManager class. The DatabaseManager class is going to call the methods , adding the information needed to call and use the queries , getting the data from the GUI. Below an example.

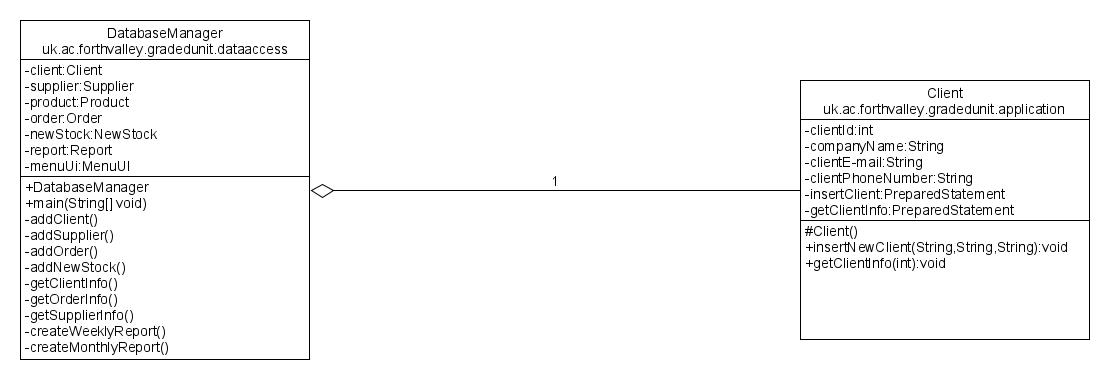


Figure 23:.application classes and .access classes relationship(example)

**Interaction in .gui package**

There are two type of interaction inside the .gui package. The first between all the other GUI JPanel class and the main GUI class the JFrame , the MenuUI, and the one between certain JPanel Class.

**MenuUI Inheritance**

For having a better functionality of the GUI and having the other panel to be able to be called and interact , there is a class that must be declared as a Frame. The other GUI classes are panel that extends the Frame and inheritance through it , the Menu class is going to communicate with the manager class. Below is how it looks like.

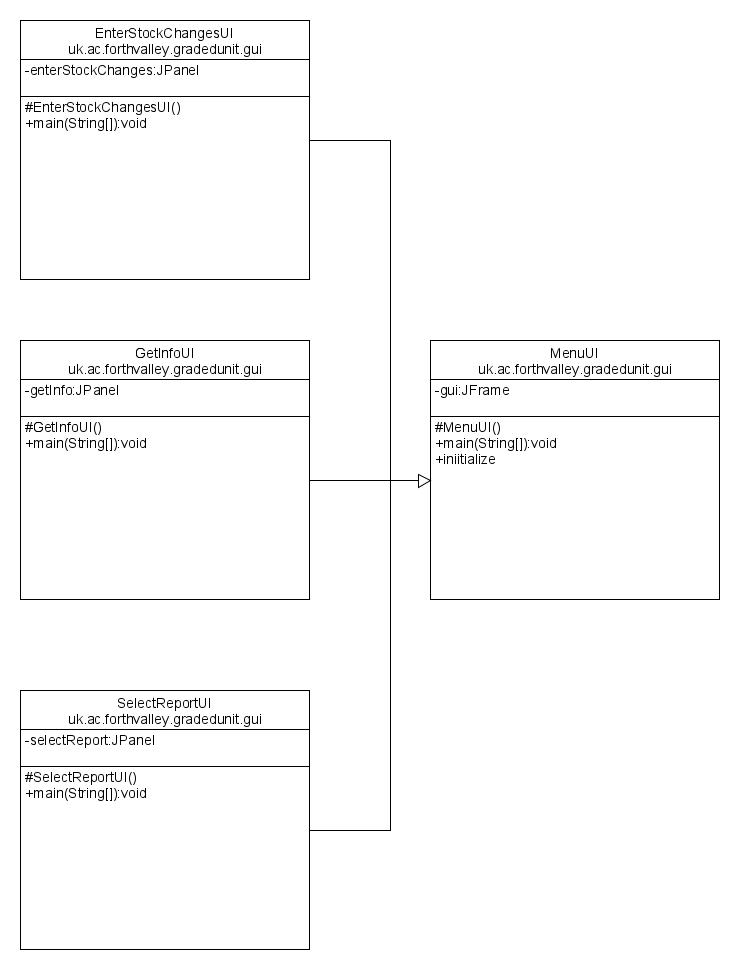


Figure 24:Inheritance in the .gui package

**Communication between the JPanels**

Some JPanel are also communicating between each other’s. Since this JPanel needs the other precedent panel to exist and they also need to have reference in the other panel , this relationship is going to be a Composition. The class that are involved in this relationship are NewUserUI with EnterStockChangesUI, the DisplayInfoUI to GetInfoUI and the DisplayReportUI with the SelectReportUI. Below the image of the relationship

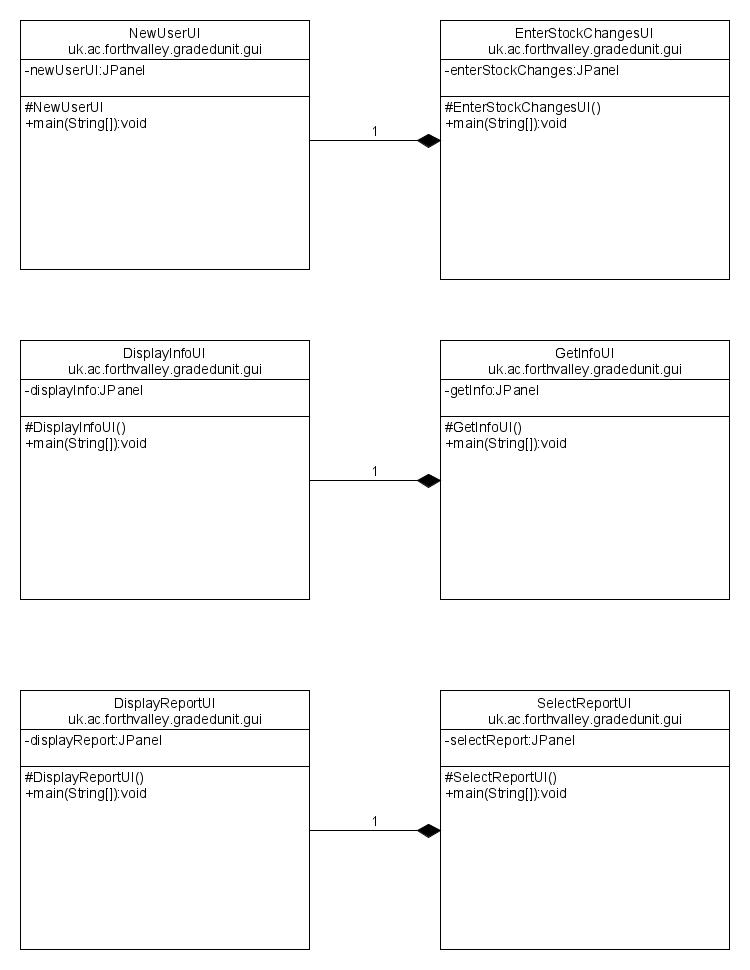
**

Figure 25:.gui JPanel interaction

**Class diagram**

After working through it , identify packages, classes and their interaction between this is the final class diagram (better quality image in the folder)

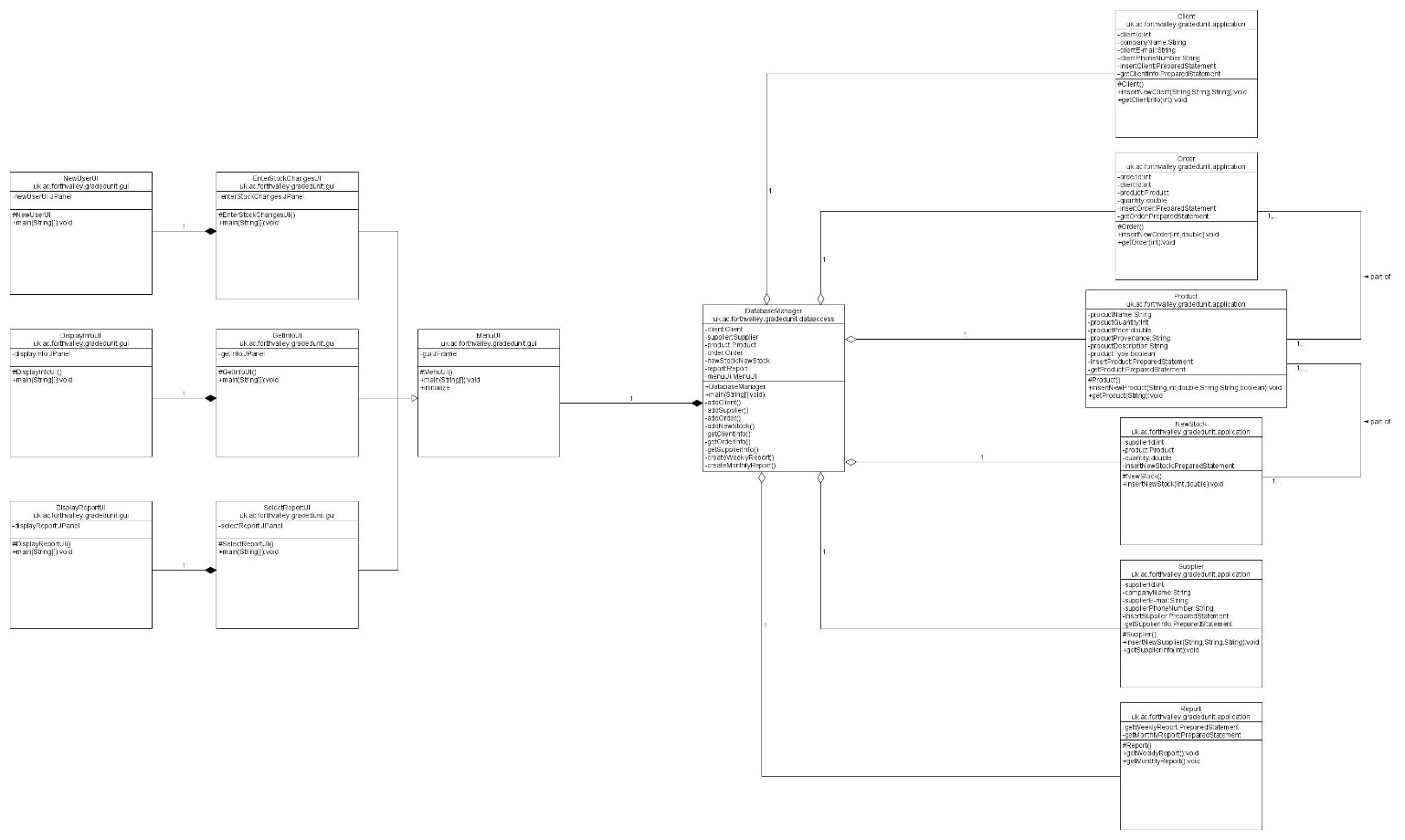


Figure 26:Class Diagram

### 2.3.5 Communication Diagram

Communication diagram explain the interaction between objects, parts and actors. This is made through sequenced messages; indeed, this diagram is also known as a simplified sequence diagram .The Database is not on the diagram but is going to give the information to display and change to the Database Manager class. The Three diagram that I created represent the three main and top-level use cases in the use case Diagram.

**Update Stock communication diagram**

(see [2.3.1 Fully dressed Use Case description 2-3](#_2.3.1_Fully_Dressed))

This Communication diagram explain the basic of how the program is going to act when updating the Stock is needed, adding new Client/supplier and managing the products and stock. The function is going to be called in the main class DatabaseManager that’s going to call them from the specific object , and after deal with the database.

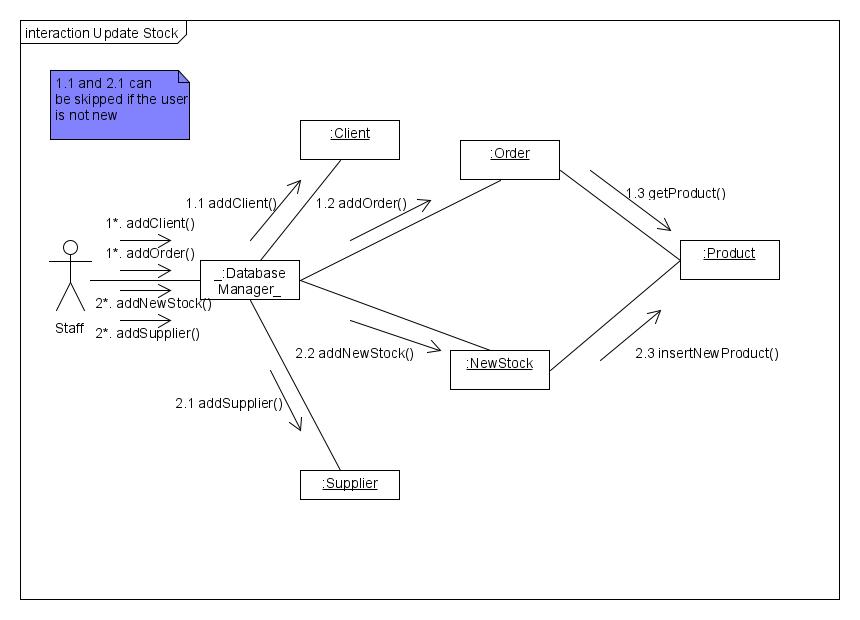


Figure 27:Update Stock use case Communication Diagram

**Get Info communication diagram**

(see 2.3.1 [Fully Dressed Use Case Description 6-7](#_2.3.1_Fully_Dressed))

This is for the Staff to retrieve info about the Supplier, Client or the past Order .The function are going to be called in the main class DatabaseManager that’s going to call them from the specific object , and after deal with the database.

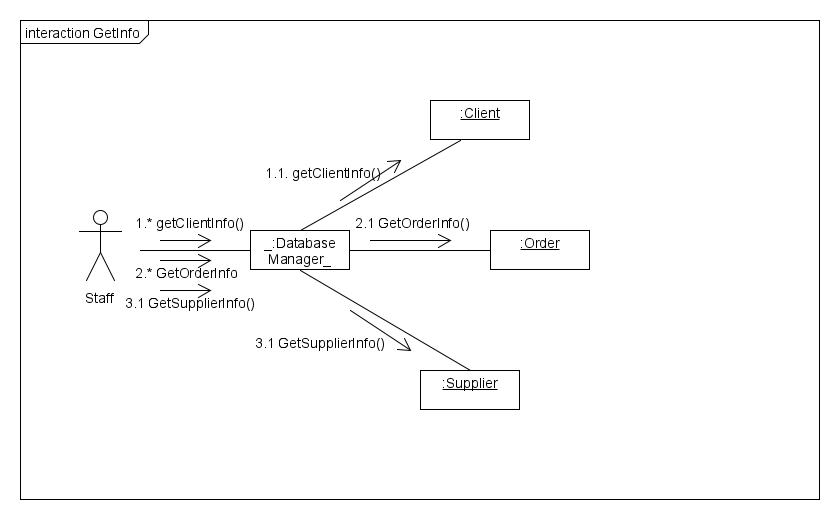


Figure 28:Get Info Communication Diagram

**Create Report communication diagram**

(see [2.3.1 Fully Dressed Use Case Description 9-10](#_2.3.1_Fully_Dressed))

Communication Diagram when the Staff needs to create different type of Reports. The function is going to be called in the main class DatabaseManager that’s going to call them from the specific object , and after deal with the database.

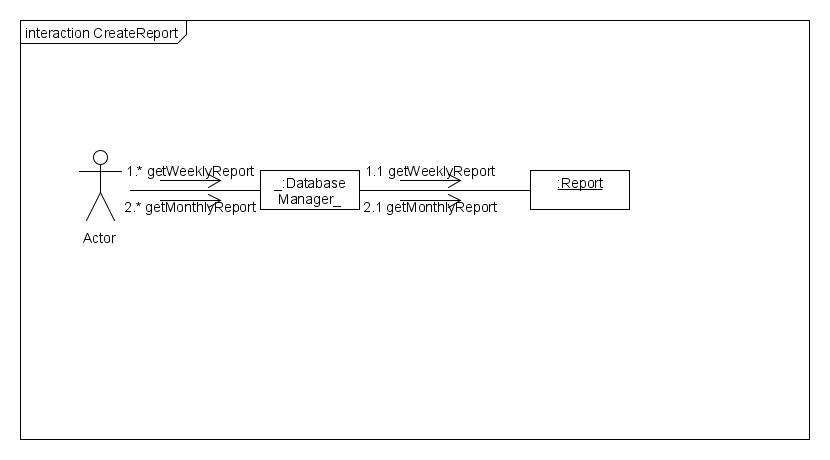


Figure 29:Create Report Communication Diagram

### 2.3.6 Activity Diagram

Activity diagrams are behaviour diagram that explain and show how the flow of the program or certain use case should be. They are not really use any technical language and they are the logical explanation of the program flow. In this section , like in the one before I am going to show how the three main use case get represent in this diagram.

**Update Stock activity diagram**

(see [2.3.1 Fully dressed Use Case description 2-3](#_2.3.1_Fully_Dressed))

(see Figure 27:Update Stock communication diagram above)

This activity shows how the program deals with the request to make change on the stock. First it asks if the Client/supplier(user) is new or not to the system. If old it’s going to skip updating the stock, if new is going to ask to add the user to the system, and display an id generated by the database. After is going to ask to select for the type and insert the id. Following is going to be the process to insert a product, finalised and inserted all the product , the session is going to end.

(Images for zooming in, in the image section of the folder)

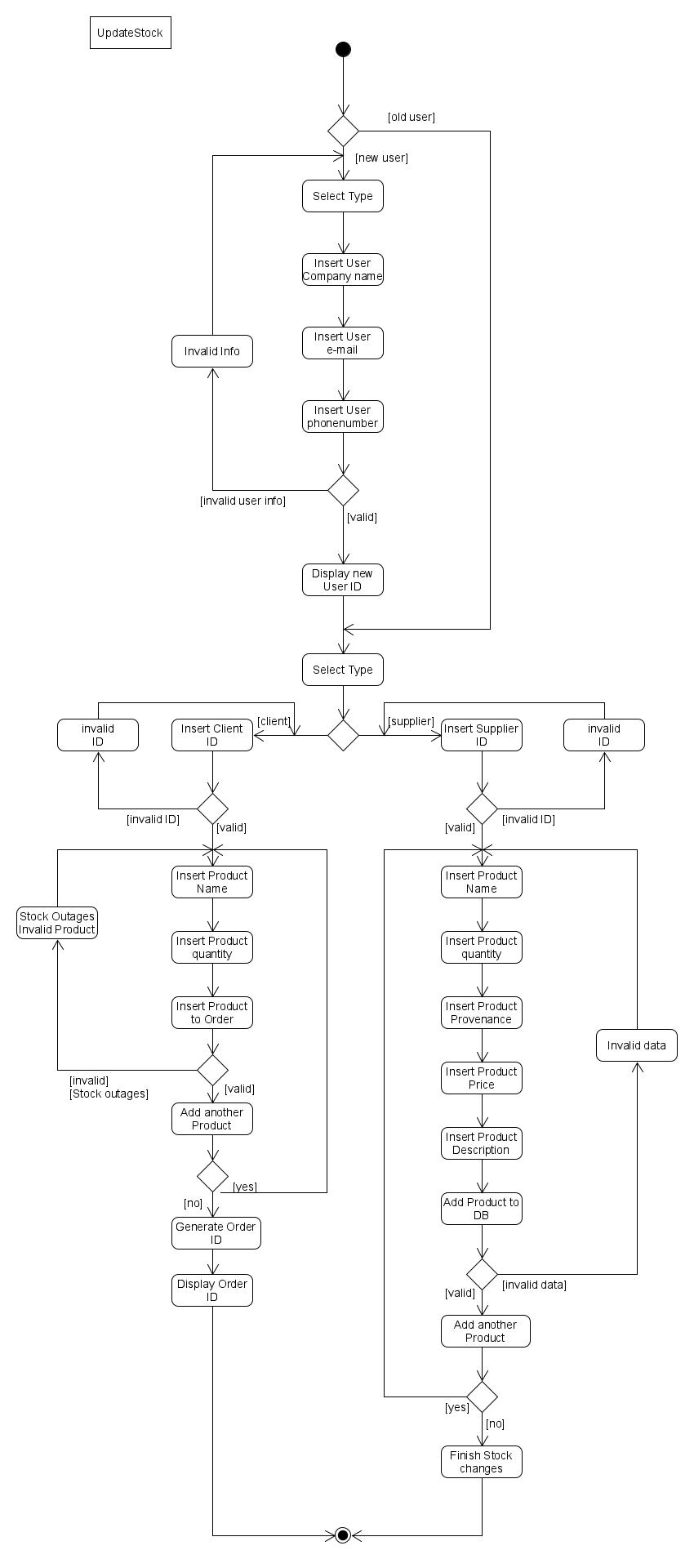


Figure 30:Update Stock Activity Diagram

**Get Info activity diagram**

(see 2.3.1 [Fully Dressed Use Case Description 6-7](#_2.3.1_Fully_Dressed))

(see Figure 28:Get Info Communication Diagram Above)

This activity diagram shows the flow of how the information are going to get retrieve from the database and how the logic behind this process is going to work. After selected the type , radio button on the GUI, the id is going to be inserted and after the information collected from the database, they are going to be displayed.

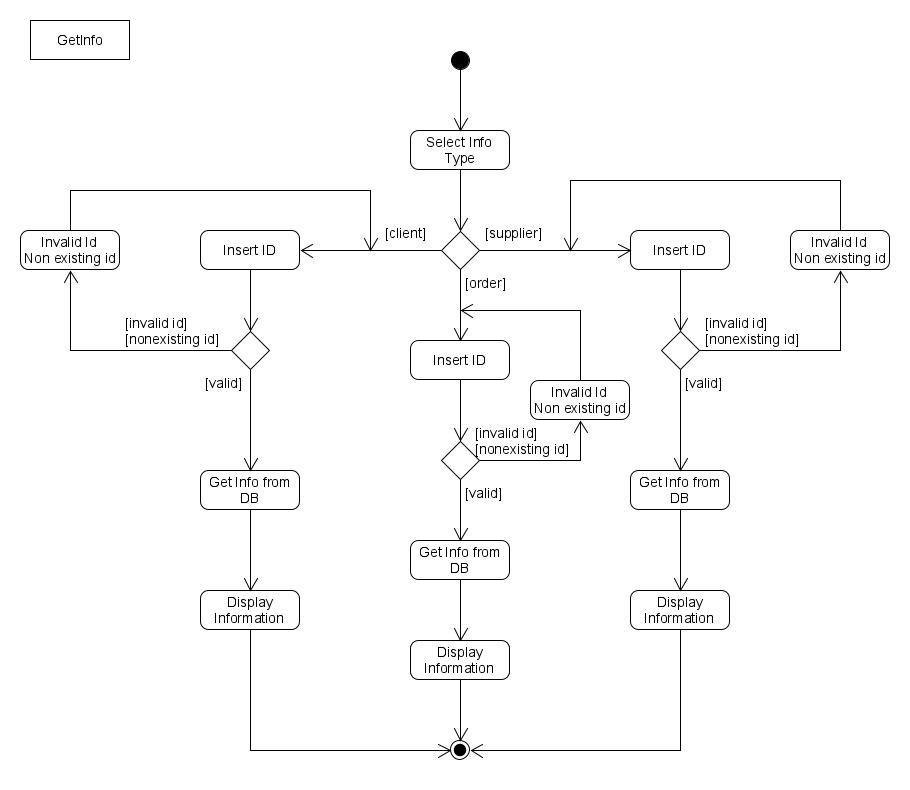


Figure 31:Get Info Activity Diagram

**Create Report activity diagram**

(see [2.3.1 Fully Dressed Use Case Description 9-10](#_2.3.1_Fully_Dressed))

(see Figure 29:Create report Communication Diagram above)

This diagram shows the flow of how the report are going to get created. After the type of report is selected the program is going to ask the Db for the information and changes of stock in the period of time requested, create a report and store the information in and , finally , display it for the Staff.

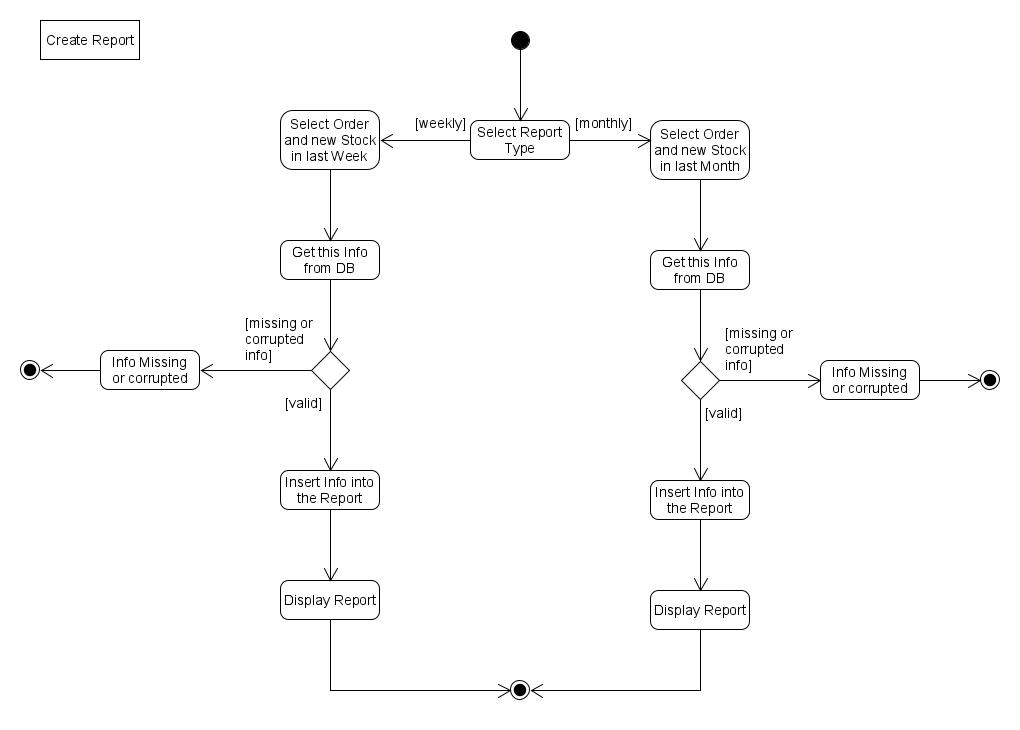


Figure 32:Create Report Activity Diagram

### 2.3.7 Sequence Diagram

Sequence diagram are behavioural diagram that show the sequence of how object interact between each other in certain use cases or algorithm. They also show the function and message that are exchanged between the objects. There are three type of arrow that explains different type of message. The straight line with full arrow indicated synchronous messages, while asynchronous message are represented by an empty arrow. The return messages are represent by dotted line. The sequence diagram are a more complicate diagram then the communication, since the explain in a better and more precise way all the error,message and alternative on the object and message communication.

**Update Stock sequence diagram**

(see [2.3.1 Fully dressed Use Case description 2-3](#_2.3.1_Fully_Dressed))

(see Figure 27:Update Stock communication diagram above)

(see Figure 30:Update Stock Activity above)

This diagram show and explain the communication between classes and database , when a change or update to the stock is needed. The First crossroad is when the Staff select if the user is new or old. In case the user is already in the system , the system would jump the all *alt 1*. To get to the other crossroad where the staff indicate if is a new order or a coming of new Stock. In case of new user the system, after having the information inserted, will generate and id for a Client(*alt 1.a*) or Supplier(*alt 1.b*). After this the other option are if is going to a new Stock(2.b) or a Order(2.a) . Both of this process are going to call a loop to add or remove products from the database. This diagram also give explanation of how the program is going to deal with error handling.

(image also available in the image folder)

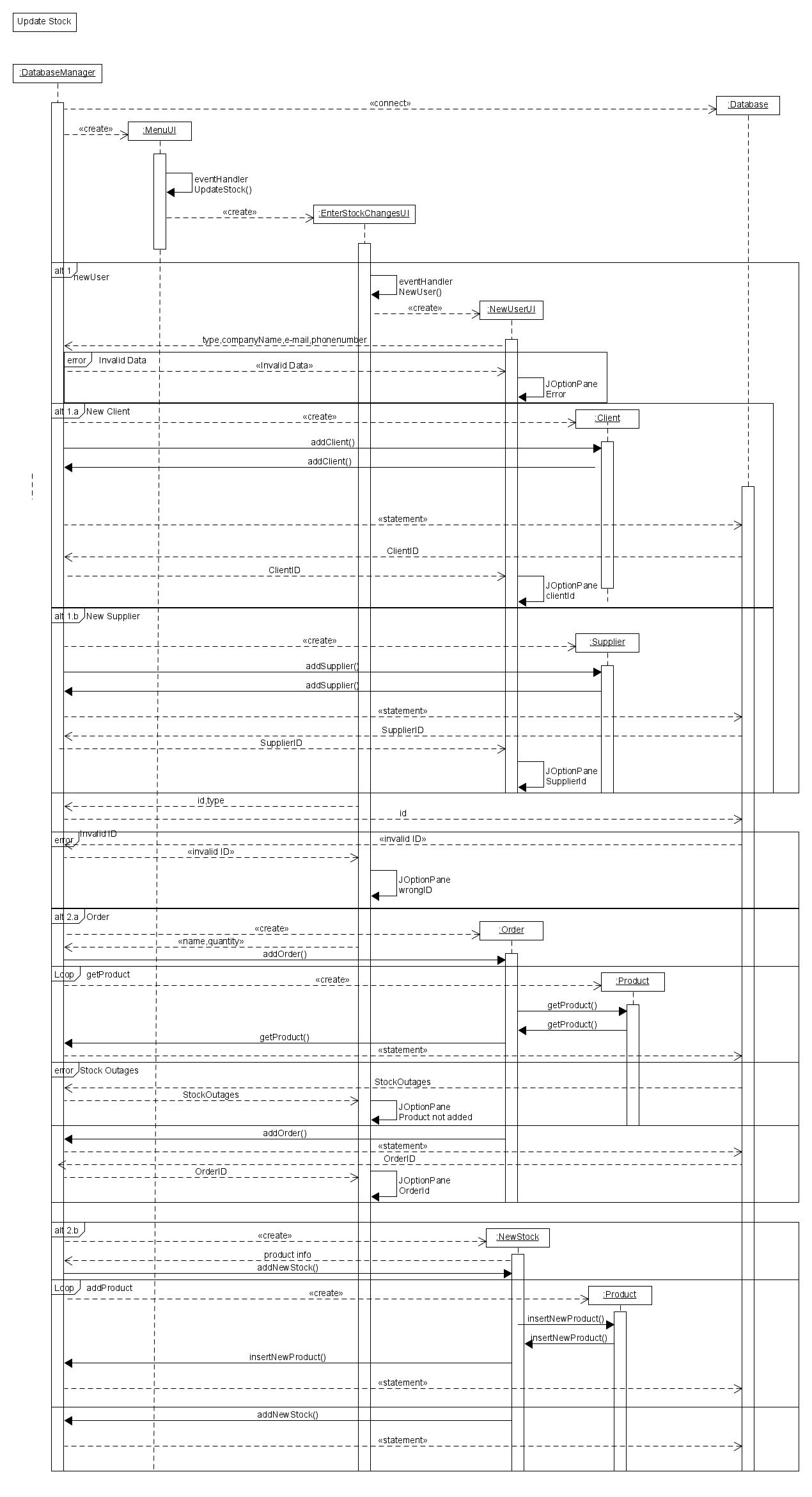
****

Figure 33:Update Stock Sequence Diagram

**Get Info sequence diagram**

(see 2.3.1 [Fully Dressed Use Case Description 6-7](#_2.3.1_Fully_Dressed))

(see Figure 28:Get Info Communication Diagram Above)

(see Figure 31:Get Info activity Diagram Above)

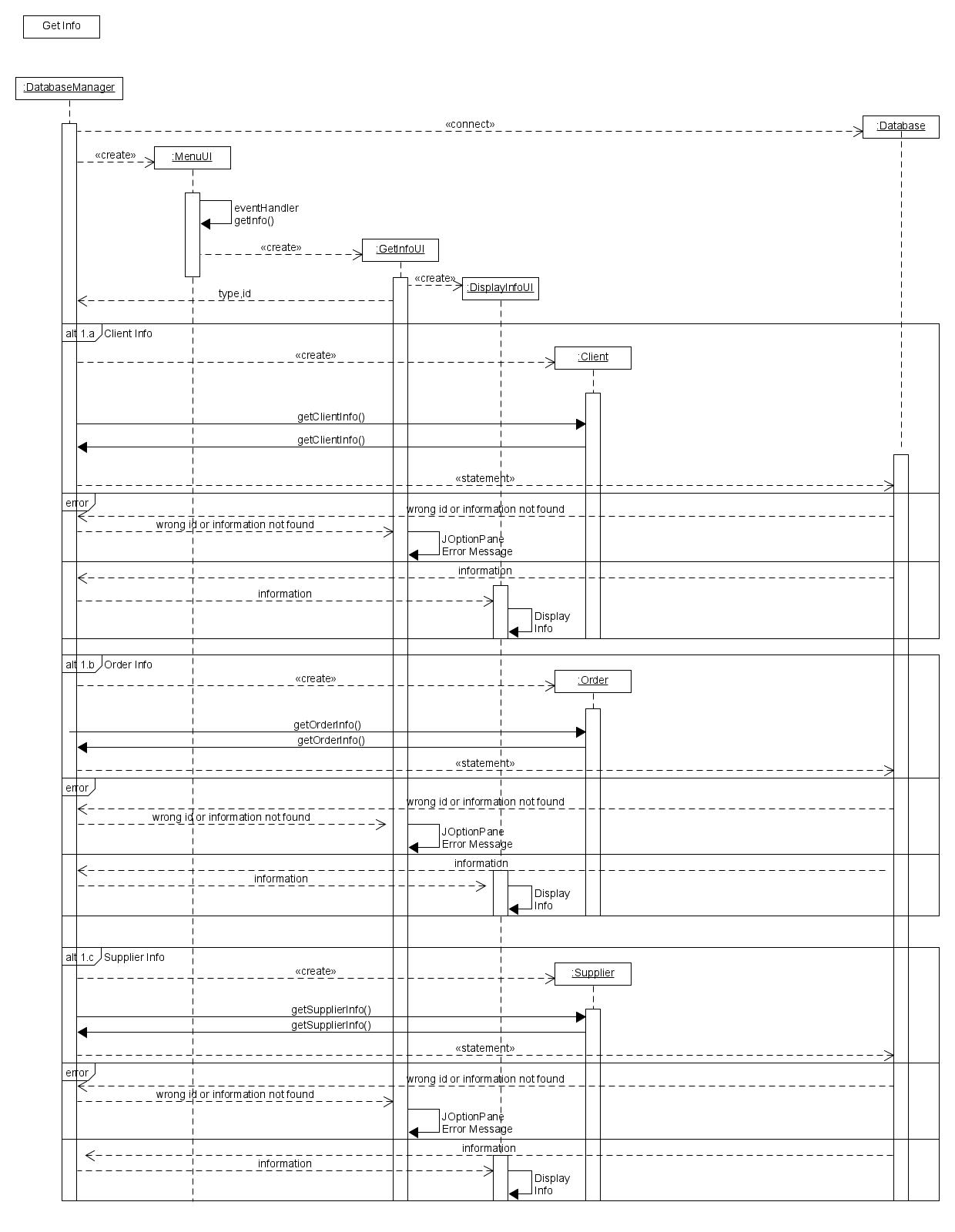
This sequence diagram explains the flow of data and methods between the classes and the database when a staff member tries to retrieve information from the database using an id. There are three possible path : one is to get information of the Client(*alt 1.a)***,** One to get information of a certain order (*alt 1.b*) and one of the Supplier(*alt 1.c*).They all use the id for retrieve the information. The diagram also show the flow of error messages. 

Figure 34:Get Info sequence diagram

**Create Report sequence diagram**

(see [2.3.1 Fully Dressed Use Case Description 9-10](#_2.3.1_Fully_Dressed))

(see Figure 29:Create report Communication Diagram above)

(see Figure 32:Create report activity Diagram above)

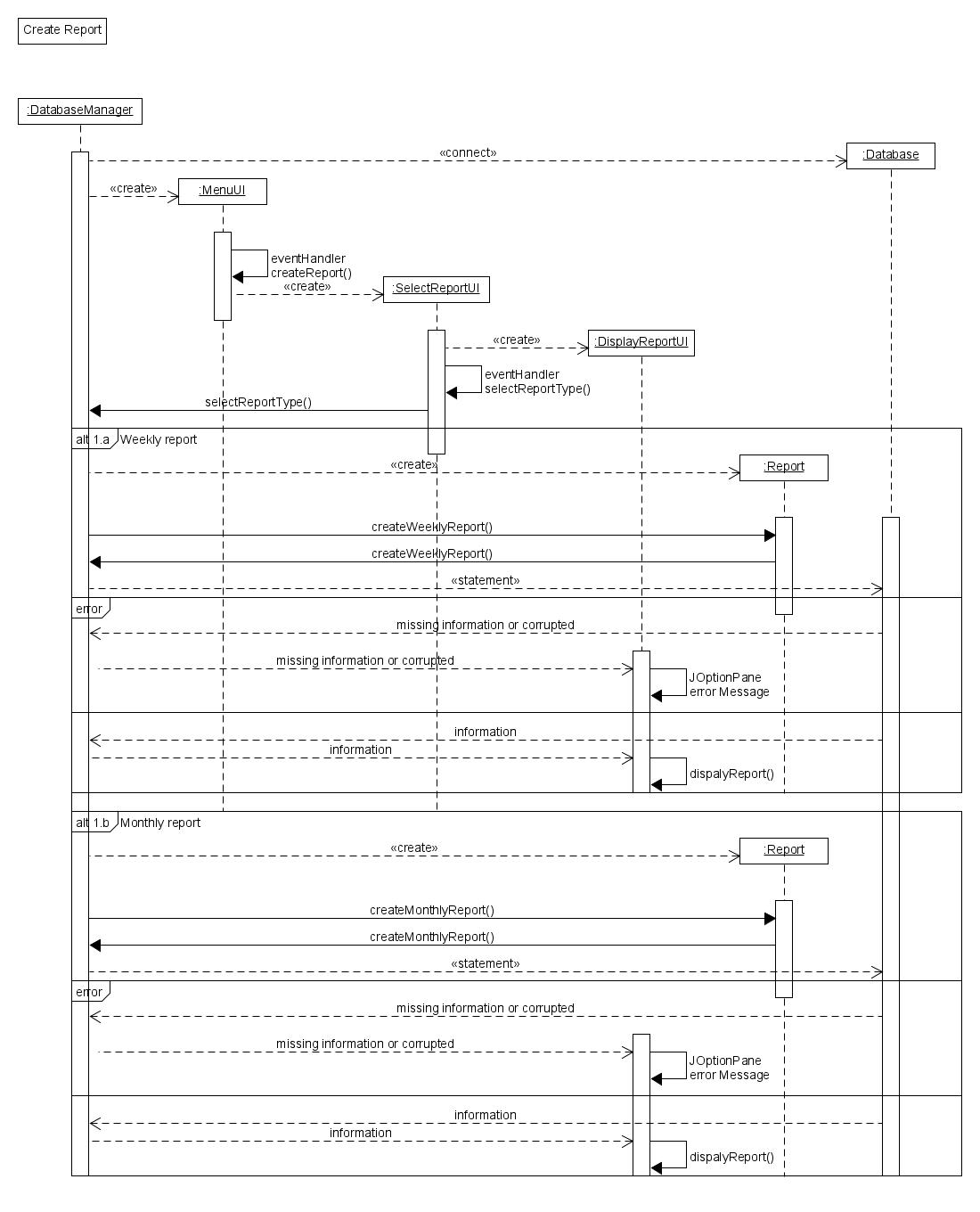
This diagram explains the sequence of messages and methods that happen, when a new report is tried to be created. There are just two type of report implemented , a weekly(*alt 1.a*) and a Monthly(*alt 1.b*). The program knows what report to create thanks to the button in the UI, that are going to call different methods from the DatabaseManager class. 

Figure 35:Create Report sequence diagram

### 2.3.8 Database Design

ERD (entity and relationship diagram) are essential to define the structure of databases. ERDs define entity, their relationship between each other and what type of relationships are. Relationship can be of three different types: one to one, one to many and many to many. ERD also store entity and the attributes that define them, storing if they are UID (unique identifier), mandatory or optional. First thing to do to create a database is to look through the data that the database must store.

**Initial database entity and attributes**

The entity and attributes that can be defined after the initial client interview and brief are the following

**Clients**: client ID(UID), company name(mandatory), phone number(mandatory) and e-mail (mandatory)

**Products**: product ID(UID), Product name (Mandatory), description(mandatory) and price(mandatory)

**Suppliers**: Supplier (UID), company name(mandatory), phone number(mandatory) and e-mail(mandatory)

**Orders**: Order ID(UID), items(mandatory), quantity(mandatory), total price(mandatory)

Looking at this entities, different changes come to my mind. First the product element should be divided in two subclasses, Vegetable and Fruits, so if is needed to display just the vegetables or the fruits is going to be possible. They are going to have a code to reference if they are a fruit or a vegetable, 0 for fruit and 1 for vegetables. Other problems are in the Orders entity where attributes like items, quantity and price shouldn’t be stored since they are going to get retrieved from the products entity. This are the new entity after reviewing them.

**Clients**: client ID(UID), company name(mandatory), phone number(mandatory) and e-mail (mandatory)

**Products**: Product ID(UID), Product name (Mandatory), quantity(mandatory),description(mandatory) and price(mandatory), type code (0 for Fruit and 1 for vegetables)

Vegetables (product subclasses)

Fruit (product subclasses)

**Suppliers**: Supplier (UID), company name(mandatory), phone number(mandatory) and e-mail(mandatory)

**Orders**: Order ID(UID)

**Initial relationship**

After having look through the entities and attributes, is time how these entities interact between each other and laid out the first idea of what relationship may be. First the relationship is going to be describe using normal language and after, they are going to be displayed using a Matrix Diagram. This are the initial relationship:

**Clients and Orders**

Clients are going to be able to make different orders, while a singular order shouldn’t be able to be shared or have many that one client.

-**Clients** may have one or many **orders** (one to many(1:M))

-**Orders** must have one and only **client** (one to many(1:M))

**Suppliers and Products**

Different suppliers could be able to sell the different products, and different products may be bought by many suppliers

-**Suppliers** may sell one or many **Products** (many to many(M:M))

-**Products** may be sold by any or more **suppliers** (many to many(M:M))

**Orders and Products**

Different products should be able to be inserted in the same orders and orders may contain different products.

-**Orders** must include one or more **products** (many to many(M:M))

-**Products** may be included in one or more **Orders** (many to many(M:M))

**Matrix Diagram**

Matrix Diagram explain an analyse relationship between entities. They are a useful management tool that help the design and the identification of how the information are related and the different kind of relationship. Matrix diagram also help to visualize easily the relationship and understand how to improve them and fix them in case of any error. There are many types of matrix diagram, going from the less complex L-shaped diagram to the complicated X-shaped diagram. For this project I am going to use the L-shaped diagram since no more complexity is required than the A-B relationship and using more complicated would just result tedious. The diagram doesn’t include the cardinality of the relationship. The diagram must be read row to column, and not every row has to have a relationship with a column.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Clients | Suppliers | Orders | Products |
| Clients |  |  | May have |  |
| Suppliers |  |  |  | May sell |
| Orders | Must have |  |  | Must have |
| Products |  | May have | May have |  |

**Top Level ERD**

Identified the primaries relationships and entities, the next step is to put them together in the first design of the database, the top level ERD. ERD are the best way to describe the Interaction between the entities. This first top level ERD, describe in a basic way how the database should be. This database presents the 4 main entities: Clients, Suppliers, Orders and Products, their relationship between each other and no attributes. The biggest problem that can be found in this database is the presence of Many to many relationships. Many to many relationships are not suggested to be implemented, since they would ask the database to store multiple value in a single column, making the process of maintenance and writing queries very hard for the database manager. For solving this type of relationship, the database needs to have another table or entities, called joining table or intersection entity, which store a reference to both the primary key and unique identifier of the two entities. To create this new entity in an erds a one to many barred relationships is created between the new entity and the two old one that were creating the many to many relationships. In this ERD, there are two many to many relationships, one between Orders and Products and one between Products and Suppliers. Two solve this M:M relationship is also needed to define and include in the ERDs the attribute that are stored, to define if the intersection entities are going to store some attributes or not. The implementation of the attributes can be seen from Figure 36 to Figure 37. Other thing to improve are Normalization and to check the transferability of the different relationship.

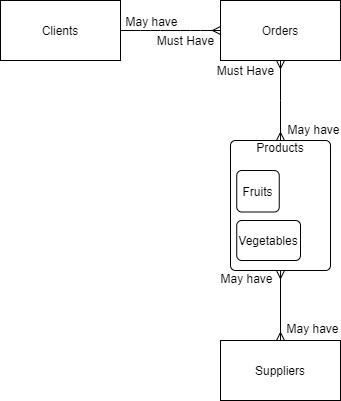
****

Figure 36:Top Level ERD

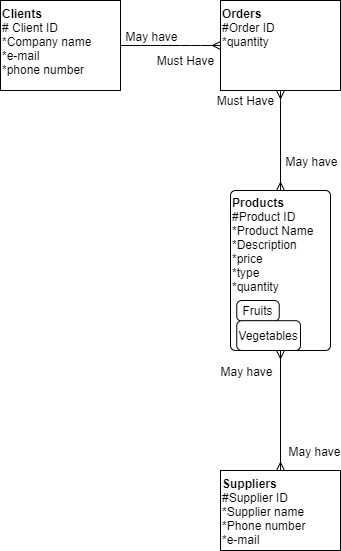


Figure 37:Top Level ERD with Attributes

**Solving M:M relationships**

So, as said before, to maintain the functionality of the Database solving Many to many relationships is essential. To solve many to many relationships, a third entity as to be created, called intersection entity, that is going to be connected to the two entities with a one to many relationships. This third entity is also going to store as Foreign keys, the two or more primary key of the two entities that where in to the many to many relationships. In my ERDS there are two M:M relationship to solve: The orders-products one and the suppliers and products one.

**Solving Orders-Products M:M relationship**

An order must be able to have one or more product, but also a product cannot be exclusive for a single order if not the database would be pointless since there would be a limited number of order and after that you would need to reset it. To solve this problem, I am going to create an entity called Ordered Products, that is going to store the ID of the order and the Different id of the ordered products, solving in this way the M:M relationship. Foreign keys are not show in this type if ERDs, so the entity is going to be “empty”. The differences can be seen in the two pictures below.

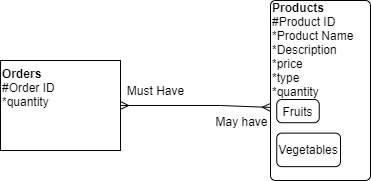


Figure 38:Orders- Product M:M relationship

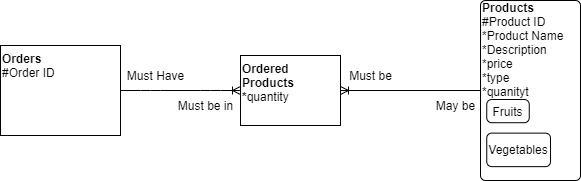


Figure 39:Orders-Products solved M:M relationship

**Solving Products-Suppliers M:M relationship**

Products may be selling by more than one supplier, meaning that sometimes when new Stock must be ordered does not have to come from the same supplier/farmer that sold the stock before. Also, obviously, Supplier are probably going to sell more than one product. To solve this, I am going to create a similar entity to the one used to solve the other M:M relationship above. I am going to create an intersection entity called new stock that is going to store the Stock ID and Supplier ID as foreign keys, knowing which supplier supplied that type of stock. This entity is also going to store the quantity of the new stock income. The changes between the solved and unsolved can be seen in the picture below.

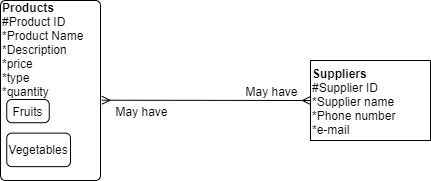


Figure 40:Products-Suppliers M: M relationship

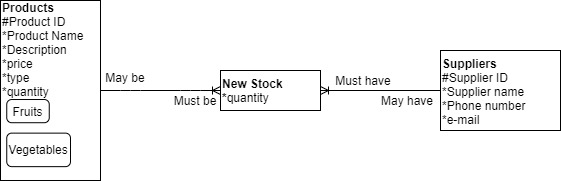


Figure 41:Products-Suppliers solved M:M relationship

**Relationship Transferability and Optionality**

Relationship are composed by 3 principles: Cardinality, Optionality and Transferability. So far, we have solved the cardinality of the relationship, setting the different relationship to One to many or one to one, and solving the many to many relationships in one to many. The different relationship optionality is showed with the different use of line, a continuous line is used to show a mandatory relationship, while a dotted line is for the optional. In the English version of the relationship a Must or have to make the relationship mandatory, while a may or can usually refer to an optional one. Transferability, indeed, refer if the relationship can be transfer to another row or entity. For example, if an order can be transferred between clients. A diamond on the relationship refer to a non-transferable relationship.

**Clients-Orders relationship optionality and transferability**

The Client must have an order to be inserted in the system and the order must have a client to exist so this relationship should be mandatory in both ends. The first time I laid down the ERD(See Figure 5-6) I did the error to think that the client could be generated without having any order , but after looking through the program functionality and idea I realised that the client exist, in the database, just if an order is placed by him. The changes can be seen between Figure 42 to Figure 43.

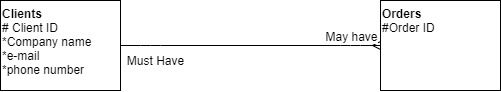


Figure 42:Clients-Orders relationship before optionality review



Figure 43:Clients-Orders relationship with the optionality improved

The different client should not be able to transfer their order between each other’s so the relationship should be not transferable.



Figure 44:Clients-Orders relationship with transferability

**Orders-Ordered Items-Products transferability and cardinality**

I put these three entities and their relationship together since the third entity, was created by solving the M:M relationship between products and orders. Orders to exist they must have ordered products if not they wouldn’t be an order and ordered items to exist, they need to be part of an ordered, making the both end of this relationship mandatory. Instead products don’t have to be part of a set of ordered products, while ordered products must be a product, making the relationship optional on the products side and mandatory on the ordered products side. The changes can be seen between figure 45 and figure 46.

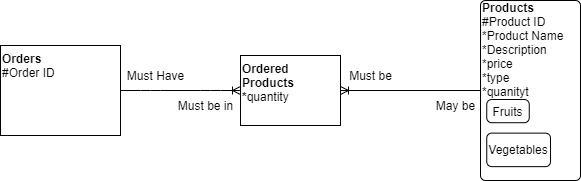


Figure 45:Orders-Ordered Items-Products relationship without optionality

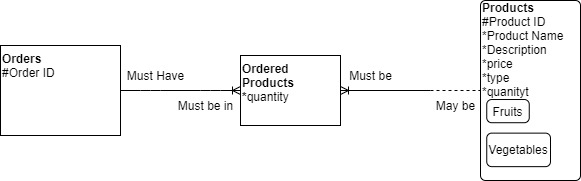


Figure 46:Orders-Ordered Products-Products relationship with Optionality

The transferability of this relationship should be transferable. Maybe not in the first implementation of the program, but in the future can be implemented the possibility to change an order just made, so this option should still be left available.

**Products-New Stocks-Suppliers Optionality and Transferability.**

As it was for the client-orders relationship, in the first ERD (see Figure 5-6) I made an error on the supplier relationship, since a Supplier to exist in the database must had sell at least one item to the shop. This said the new stocks supplier’s relationship must be mandatory, while the Products new stock relationship should be optional on the products end. Indeed, the database is going to be generated after some stocks are already been purchase to the shop without taking track on the database, making the products been already in the system without being new stocks. The changes can be seen between figure 47 to figure 48.

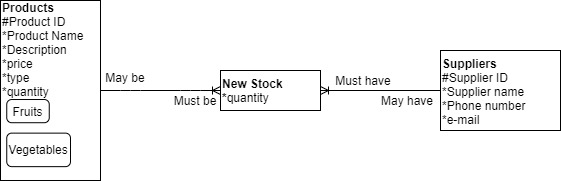


Figure 47:Products-New Stocks-Suppliers relationship without optionality

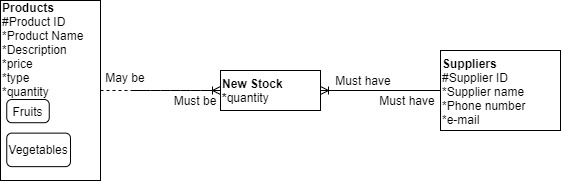


Figure 48:Products-New Stock-Suppliers relationship with optionality.

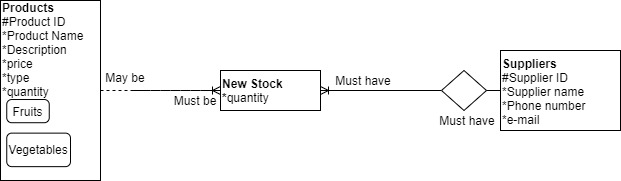
As for Transferability a Supplier should not be able to transfer is supply to anyone else, so not transferable, and for the New Stock the same problem that for the orders, could be needed to be able to be changed so I will leave it transferable.

Figure 49:Products-New Stock-Supplier relationship with transferability

**Matrix Diagram (revised)**

The First Matrix diagram that I created (see Matrix diagram at page 54) was incomplete and does not explain the new and changed relationship happened during the development of the ERD. So, I decided to make a new and improved one.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Clients | Orders | Ordered Items | Products | New Stock | Suppliers |
| Clients |  | Must have |  |  |  |  |
| Orders | Must Have |  | Must Have |  |  |  |
| Ordered Items |  | Must Have |  | Must be |  |  |
| Products |  |  | May be |  | May be |  |
| New Stock |  |  |  | Must be |  | Must have |
| Suppliers |  |  |  |  | Must have |  |

**Normalization**

The only normalization that came to my mind for this project is to break down the description in the Product, keeping the description but adding a new attribute called provenience.

**ERD**

After all this process this is the final ERD. This ERD is the product of all the process and change made in the pages above from the first ERD (see ERD page 55-56).

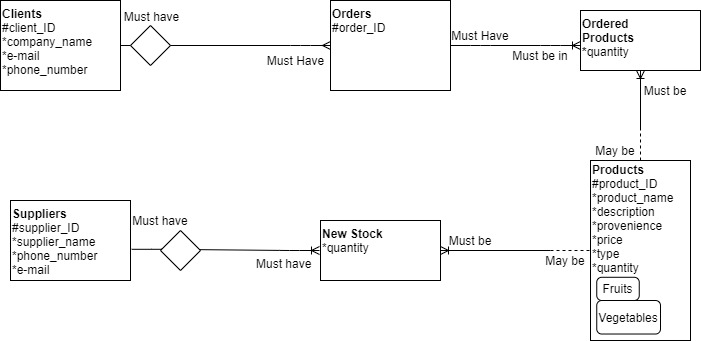


Figure 50: Final ERD

### 2.3.9 Graphical design & Client meeting

Graphical design is an important part of the project as well. It is where all the graphical part of the application gets built and is where in the solution stage the client gets actually and actively involved. In this section of the documents , I’m going to explain what I have done for the Logo and name of the company, the colour scheme and how this is going to be applied and how is going to deal with people with colour blindness problem , the accessibility with people with different disability, and the first prototype made with wireframes of the GUI of the application. In this phase I am going to show to the client different solution and at the end, with a client meeting ask for his opinion and approval about the different design. I’m going to create different logos, colour scheme, and a dynamic prototype and wireframes.

**Logos and Company name**

With the application I am also trying to rebrand the company, giving them new name and logo. So, for the company name I have decided for “**5 a day, Veggie and Fruit Wholesale”** and had the task to come up with different logos to show to the client that represent the idea of the shop. For making this logos for the company, I used the help of two free website that help you making logos for free, since I am not a graphic designer and I don’t want to pretend to be one. The first one is [Hatchful](https://hatchful.shopify.com/)  and the second is [FreeLogoDesign](https://www.freelogodesign.org/). I use their idea and modify them to create this logos for the company.

**Logo Nº1**



Figure 51:Logo Nº1

This is the simplest logos and it’s just the simple title in a green background. This logo is also available in different color of background and the text.

**Logo Nº2**

****

Figure 52:Logo Nº2

This logo has a text and an image of a carrots ,since is a wholesale shop of fruit and vegetables and also follow the color scheme that I have in mind for the app using different bright shade of green and yellow, that in my opinion better represent fruit and vegetables.

**Logo Nº3**

****

Figure 53:Logo Nº3

This logo represents the tree and on the back something that could be a sun or other tree leaves. This combination well represents the business since tree are where majority fruit come from. The colors are also bright and of the right color scheme I would like to implement in the business brand. I think this logo well represent my design idea for the brand.

**Logo Nº4**

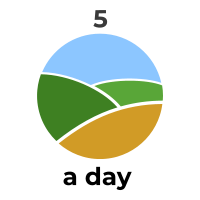
****

Figure 54:Logo Nº4

This logo represents cultivated hills and give the message of that our products come from the producer direct to the shop which we provide to. The Bright color also follow the idea that want to give to people of the shop, making looking stuff fresher and of better quality.

**Logo Nº5**

****

Figure 55:Logo Nº5

This logo represents a spike of wheat, and even if the wheat is not related to fruit and vegetable, in people mind still represent agriculture at its most basic form. This logo could probably be improved changing the “5” color to a brighter one

**Logo Nº6**

****

Figure 56:Logo Nº6

This logo has a different color scheme of the other, but it’s probably what it makes it special. Indeed, the tomato and the turnip dictated a more red based color scheme, but I think it still effective with the green and yellow color scheme.

**Logo Nº7**

****

Figure 57:Logo Nº7

This is the last logo and one of my favorite one. Simple but Include everything you need, the vegetables at the top, the company name and a description of the shop.

**Color Scheme**

For the application color scheme, I thought to make something using very bright colors, since fresh vegetables and fruit are usually very bright and colorful, and when they are like that, they are also more appetible to all. While I was doing creating this color scheme I also take as accountable the possibility of having color blind people as employee. Using information found online and on the [National eyes institute website](https://www.nei.nih.gov/learn-about-eye-health/eye-conditions-and-diseases/color-blindness/types-color-blindness) there are different kind of color blindness and the most common is called deuteranomaly, and it’s a red-green color blindness. Working with the website [Paletton](https://paletton.com/#uid=1000u0kllllaFw0g0qFqFg0w0aF) I created three different Color scheme to show to the client , showing also the deuteranomaly version for color blindness. The color I focused on are green and yellow, and for the last a black and white version for the one that are affected by achromatopsia and for a more simple and basic design.

**Color scheme 1**

This is the first and better color scheme. It’s a combination of bright yellow and green and I personally think is the best representation and best fit for the business. Above the normal view (figure 27) and what would look like for color blind people (figure 28).

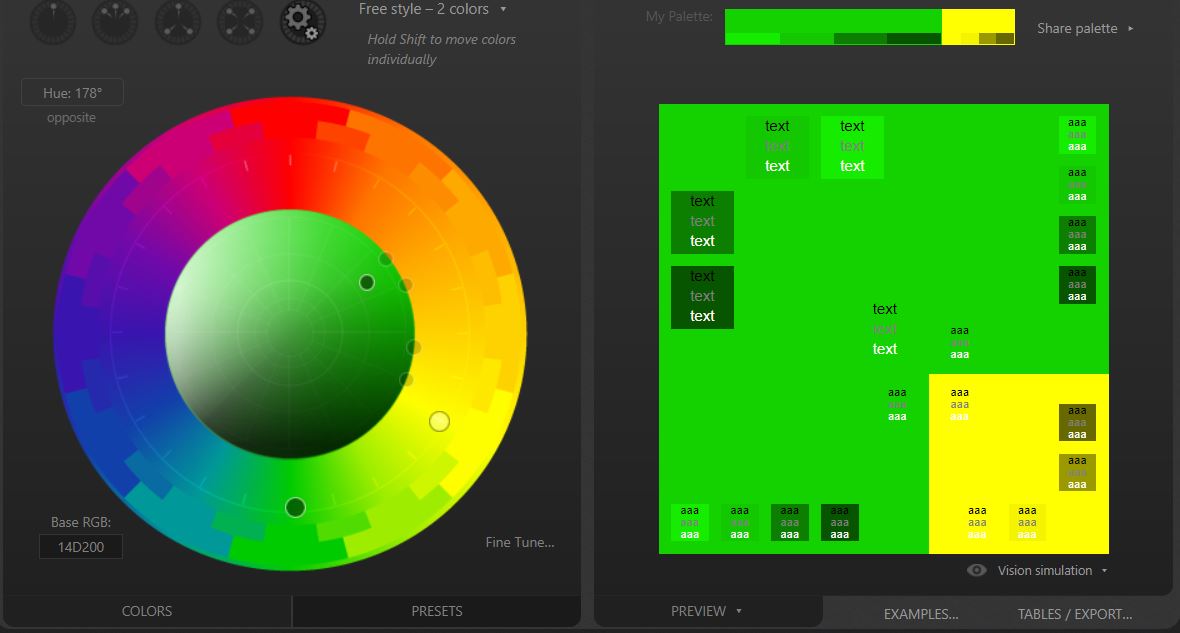


Figure 58:Color Scheme 1-"normal view"



Figure 59: Color Scheme 1-Deuteranomaly view

**Color Scheme 2**

This color scheme it’s a darker alternative of the first color scheme. The combination is still green and yellow, but this time a little more on the dark shadow. Above the normal view (figure 29) and what would look like for color blind people (figure 30).



Figure 60:Color Scheme 2-"normal view"

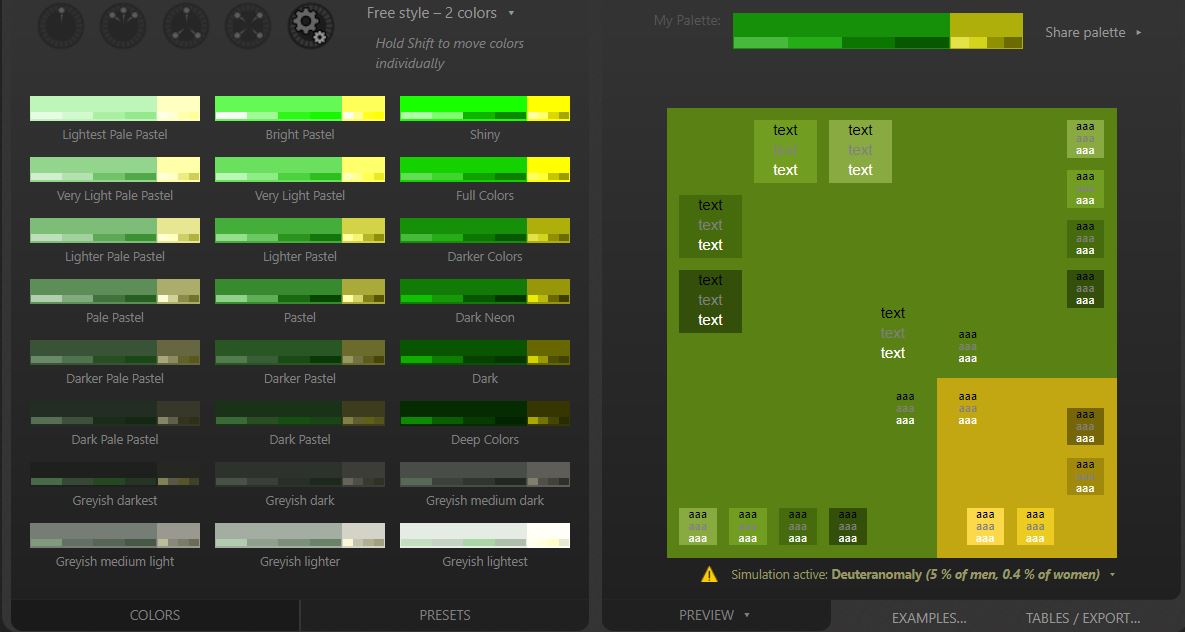


Figure 61:Color Scheme 2-Deuteranomaly view

**Color Scheme 3**

The third scheme is a black and white one. It’s a simple color scheme design for people that cannot see colors, or for an easier design, if the bright colors are no approved by the client.

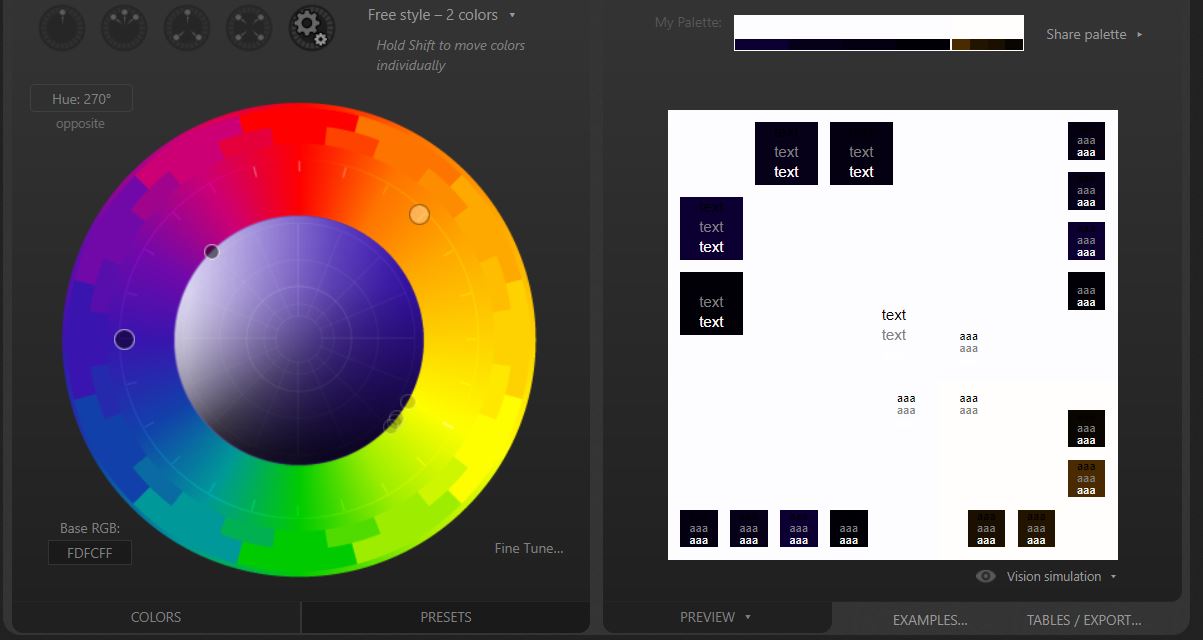


Figure 62:Color Scheme 3

**Accessibility**

For recent studies, up to the 25% of users need accessibility support. These stats make accessibility a key part to talk and deal with during the planning stage of making an application or website, or anything that is going to be use by clients and users. One part of the accessibility is about different scheme for color blind people, which we already deal on the color scheme section. In a second moment a button to change the visibility of the app is going to be implemented. This button will make the text and button bigger, so people with motion problem or eyesight problem won’t have a hard time trying to click and read the different option. Also adding more contrast between text-background and button could also be helpful. Having a written description of the UI would help people with blindness or with difficulty in reading to be able to use it anyway. Adding a description would allow the screen readers to the functionality out loud. Also, a vocal assistant could be implemented to allow the users with disability that don’t give them the opportunity to type well or type at all to use the app. There is not legislation yet that set specific guidelines about what kind of accessibility should be included in mobile apps, but the one the regulate the use for website can also be applied into application. Also it’s important to give equal job opportunity to everyone , and follow the [Equality act 2010](http://www.legislation.gov.uk/ukpga/2010/15/contents.).

**Dynamic Prototype and Wireframes.**

(Dynamic prototype available [here](https://app.moqups.com/k2MJEHHUbO/view/page/aa9df7b72))

Dynamic prototype and wireframes show the initial idea of the GUI to the user. Dynamic prototypes are particularly effective since they are interactive, and they give an all-around view and idea of what the app would be. Wireframes are not interactive, but they are still helpful for the GUI design. The dynamic prototype of the wireframes below can be found at the hyperlink above.

**Menu wireframe**

(See Figure 14:MenuUI Class)

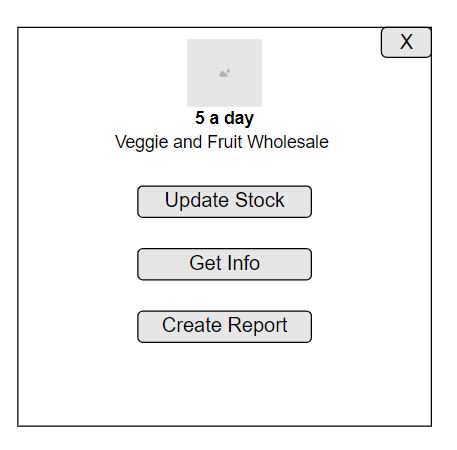
****

Figure 63:Menu wireframe

This is the menu wireframe, then is going to popup when the application is started, the image is going to be the logo and the button are going to lead to the three function of the app.

**Update Stock Wireframe**

(see [2.3.1 Fully dressed Use case 1](#_2.3.1_Fully_Dressed))

(see Figure 15:EnterStockChangesUI Class)

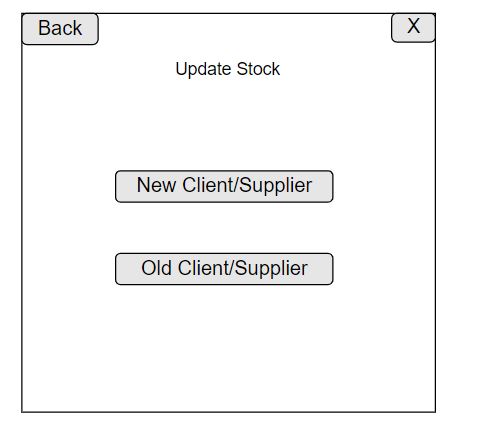


Figure 64:Update Stock wireframe

This will allow the user to select if the people placing the order or selling the stock is a new or old client/supplier and if he is needed to be added to the system. This refer to the Use case 1(Hyperlink above under the title). Back button is going to bring you back to the menu.

**New Client/Supplier Wireframe**

(see [2.3.1 Fully dressed Use case 3](#_2.3.1_Fully_Dressed))

(see Figure 16:NewUserUI Class)

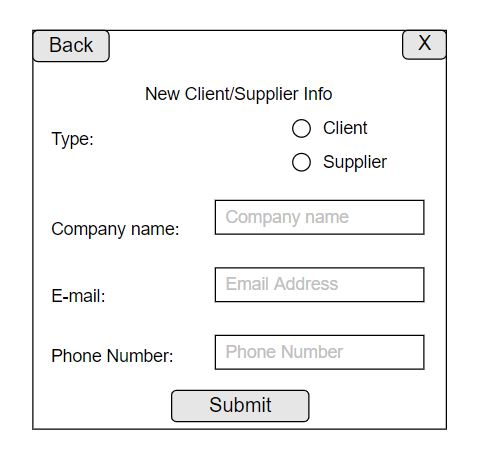


Figure 65:New Client/Supplier Wireframe

This page allows the staff to add new client and supplier to the database. The staff is going to select the type of user, client or supplier, and add the info requested by the database(see [ERD](#_2.3.3_Database_Design)). This refer to the Use case 3(link above).

**New Client/Supplier added wireframe**

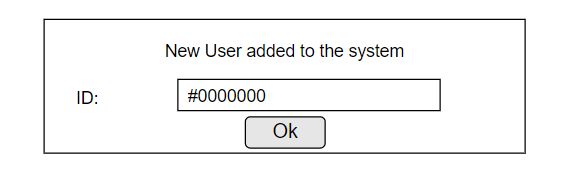
****

Figure 66:New Client/Supplier added wireframe

This popup message will appear when the submit button is pressed and a type of user is successfully added to the system. This would show the ID that the system gave to the client/supplier.

**Enter Stock changes**

(see [2.3.1 Fully dressed Use case 2](#_2.3.1_Fully_Dressed))

(see Figure 15:EnterStockChangesUI Class)

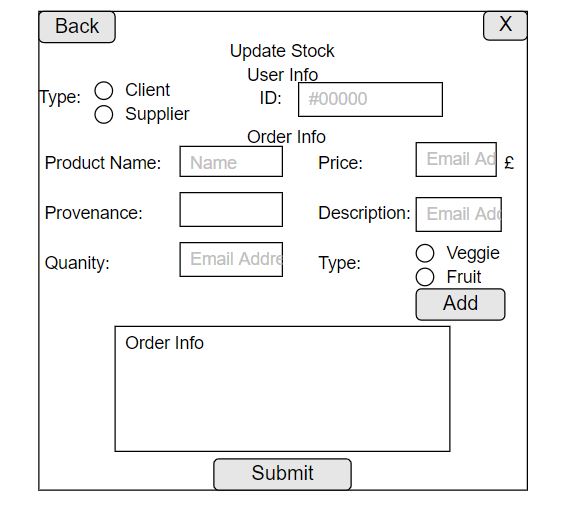
****

Figure 67:Enter Stock changes wireframe

This is going to allow the staff to insert the changes made at the stock, a new order has arrived, or a new stock has been bought. Every order is going to be add to the system pressing and the information plus the id are going to show in the box with order info inside. This refer to use case number 2(link above).

**Stockage outages/shortage wireframe**

(see [2.3.1 Fully dressed Use case 5](#_2.3.1_Fully_Dressed))

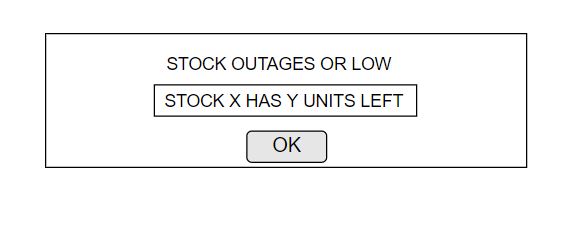
****

Figure 68:Stockage outages/shortage wireframe

This message is going to pop up if one of the stocks is close to outages or is low, so cannot be add to the order.

**Stockage successfully update wireframes**

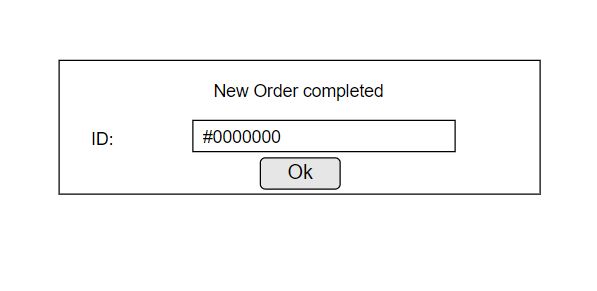
****

Figure 69:Stock successfully updated wireframe

Message that comes up when the order/new stock is successfully placed. This message is going to show the ID of the order/stock update.

**Get Info wireframe**

(see [2.3.1 Fully dressed Use case 6](#_2.3.1_Fully_Dressed)-7)

(see Figure 17:GetInfoUI Class)

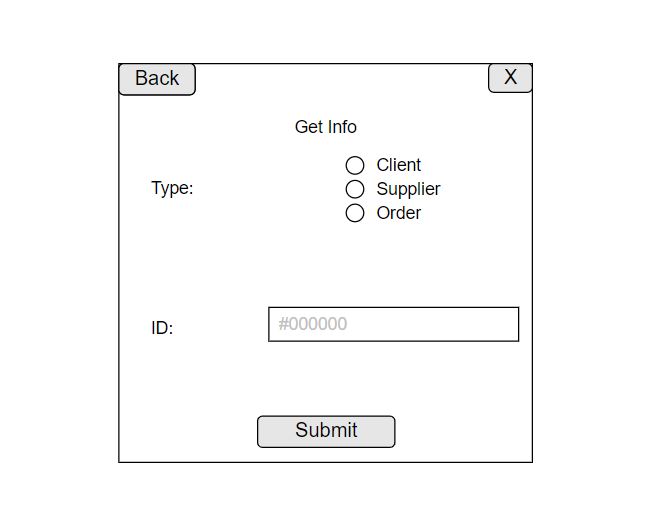
****

Figure 70:Get info wireframe

This wireframe shows how the info are going to be retrieve from the system. The staff is going to select which type of info what to get and insert the ID related to. See use case number 6.

**Select Info wireframe**

(see [2.3.1 Fully dressed Use case 6-7](#_2.3.1_Fully_Dressed))

(see Figure 18:DisplayInfoUI Class)

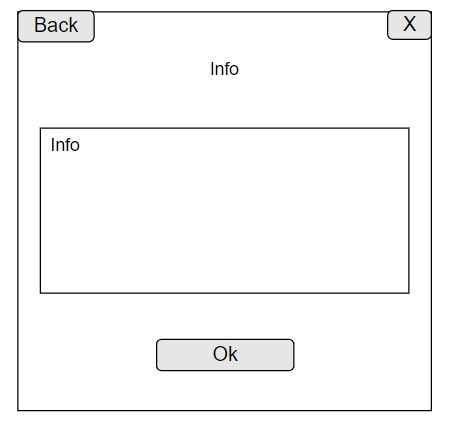
****

Figure 71:Select Info wireframe

Show the info that the staff asked for.

**Info missing/incorrect wireframes**

(see [2.3.1 Fully dressed Use case 8](#_2.3.1_Fully_Dressed))

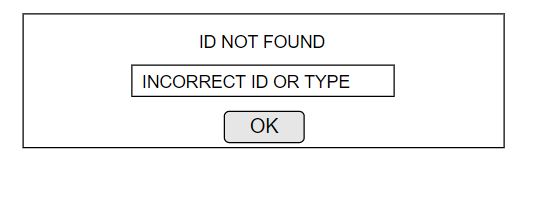


Figure 72:Info missing/incorrect wireframe

Error message for missing info or incorrect Id. See use case number 8.

**Select report type wireframes**

(see [2.3.1 Fully dressed Use case 9-10](#_2.3.1_Fully_Dressed))

(see Figure 19:SelectReportUI Class)

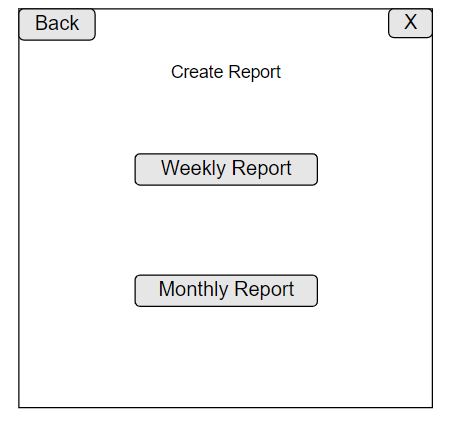


Figure 73:Select Report type wireframe

Allow the staff to create and select the type of report.

**Report info missing/incomplete**

(see [2.3.1 Fully dressed Use case 11](#_2.3.1_Fully_Dressed))

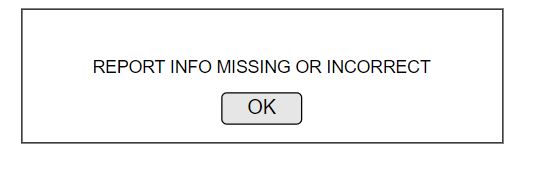


Figure 74:Report info missing/incomplete wireframe

Error message if the report missing are info or incomplete. See use case 11.

**Client Meeting and Client design choice**

(Audio of the client meeting [here](Client%20Interview%20Design.m4a) or in the folder under Client Interview Desing)

As first thing in the client meeting, I show to her the name that I thought for the business (“5 a day”) which get approved. After I show how the dynamic prototype works and how the conceptual idea of the system is. The client agrees with the design and the messages and concept of the GUI, saying that it is “nice and clean”. After I show her the different logos I made, and she select to go for Logo Number 5. After I show her the color scheme and she select the first one for the green and yellow, which she defines “Perfect fit for the fruit and vegetables shop” and “Spot on”. As last we discussed about the accessibility of the GUI, and what could be implemented (see above for the accessibility bit). Below the selected Color scheme and logo.



Figure 75:Logo selected by the client

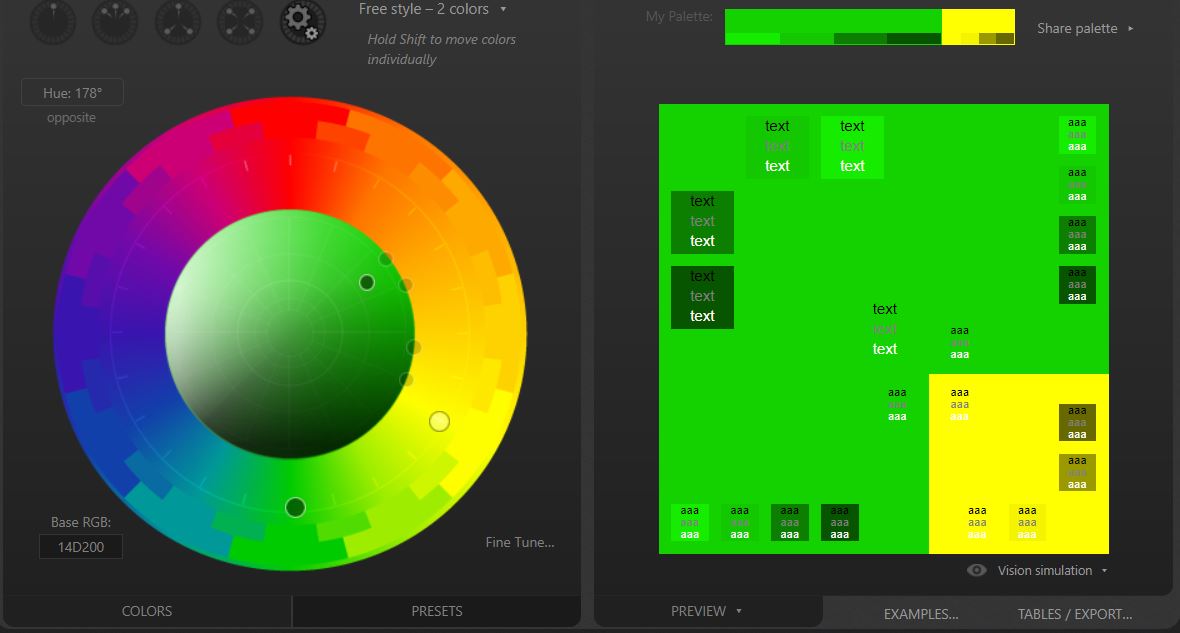


Figure 76:Color Scheme selected by the client

### 2.3.10 Data Binding

Data binding represent the interaction between the UI and the data it displays, so where the application and app get the information from. This process should represent how the data arrive and how the text field that show them are actively updated. In this application specific case the data binding is between the application and the database. The program when needs to display it call the specific function that call the statement to the database and send the information back to the UI. This section is just going to have the Update stock and get info part, because the Created Report are going to be displayed in a different way,outside the GUI.

**Update Stock(Product Information)**

These two images ,roughly represent, how the data travel between application , Gui and database, when a new product is added to an order or a newStock. This show just the data display and not the changes that the function makes to the database.

Check the diagrams sections(from [2.3.3](#_Model_Diagram) to [2.3.8](#_2.3.8_Database_Design)) to see specific functionalities.

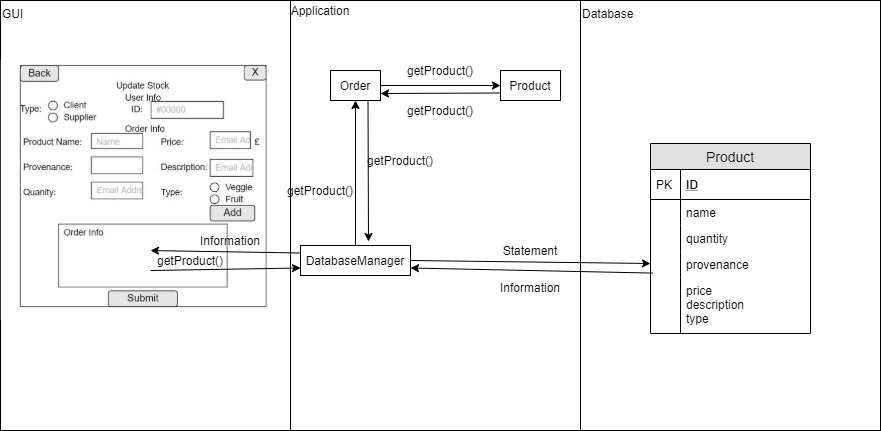


Figure 77:Data Binding for show product information in an order

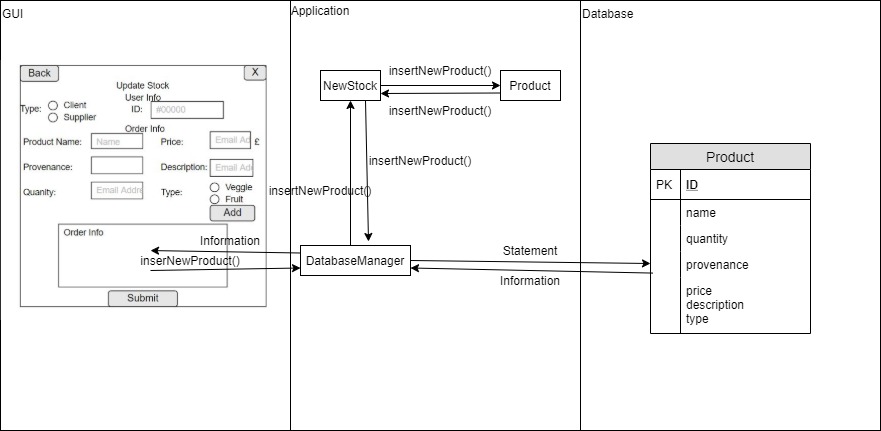


Figure 78:Data Binding for show product information in a new Stock

**Get Info data binding**

This part shows how the information shows for displaying the information need by the Staff are retrieve. The logic and specific functionality are explained and displayed in the different diagram , in the different section of this document. This part wants just to graphically and briefly explain how the communication work showing the GUI and database.

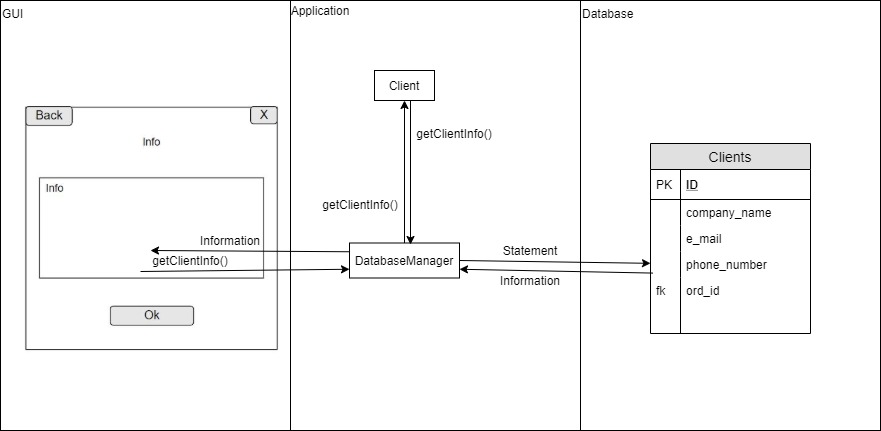


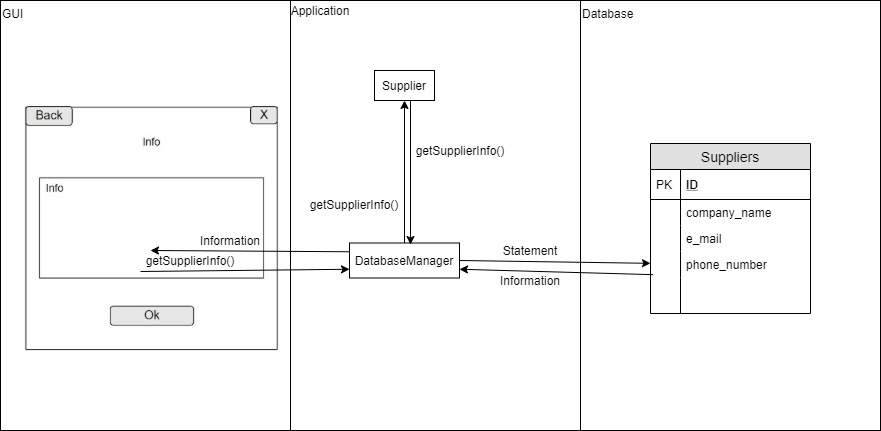
Figure 79:Get Client info data binding

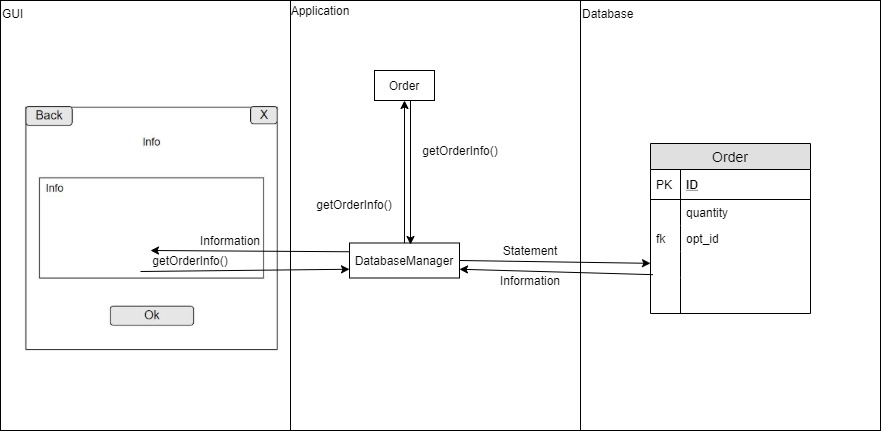
Figure 80:Get Supplier info data binding

Figure 81:Get Order info data binding

# 5. Bibliography

Government Digital Service (2011). *Data protection*. [online] GOV.UK. Available at: https://www.gov.uk/data-protection.

‌Mitre.org. (2014).*CAPEC - CAPEC-111: JSON Hijacking (aka JavaScript Hijacking) (Version 3.2)*. [online] Available at: https://capec.mitre.org/data/definitions/111.html [Accessed 25 Jan. 2020].

Basel, K. (2018).*How Did Python Become the Language of Choice for Data Science?*[online] Netguru.com. Available at: <https://www.netguru.com/blog/how-did-python-become-the-language-of-choice-for-data-science>.

John Paul Mueller and pwpadmin (2018). *Java vs. Python: Which Programming Language Is Best?* [online] New Relic Blog. Available at: https://blog.newrelic.com/engineering/java-python-comparison/ [Accessed 25 Jan. 2020].

‌DEV Community (2019). *Why Java is the best Programming language to Learn Coding for beginners?*[online] The DEV Community. Available at: https://dev.to/javinpaul/why-java-is-the-best-programming-language-to-learn-coding-for-beginners-n89.

‌Wikipedia Contributors (2019). *Java Database Connectivity*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Java\_Database\_Connectivity [Accessed 2 Aug. 2019].

Wikipedia Contributors (2019b). *MySQL*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/MySQL.

‌www.javatpoint.com. (2011). *Java Database Connectivity with MySQL - javatpoint*. [online] Available at: https://www.javatpoint.com/example-to-connect-to-the-mysql-database.

‌Wikipedia Contributors (2018). *Agile software development*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Agile\_software\_development.

‌Agilenutshell.com. (2019). *What is Agile?* [online] Available at: http://www.agilenutshell.com/.

‌ Handshake.com. (2019). How to Wholesale and Sell Products to Retailers | Handshake. [online] Available at: https://www.handshake.com/blog/how-to-wholesale [Accessed 1 Feb. 2020].

MX Wholesale UK Discount Wholesale and Pound Shop Supplier. (2019). MX Wholesale UK Discount Wholesale and Pound Shop Supplier. [online] Available at: https://www.mxwholesale.co.uk/ [Accessed 1 Feb. 2020].

George Perry Ltd. (2019). George Perry Ltd. [online] Available at: http://www.georgeperry.co.uk/ [Accessed 1 Feb. 2020].

Fresh Direct. (2014). Fresh Direct - We deliver fresh fruit, veg and wholesale food directly to you. [online] Available at: https://www.freshdirect.co.uk/ [Accessed 1 Feb. 2020].

Real Foods 2015 (2020). Shop fresh fruit and veg online at Real Foods. [online] Realfoods.co.uk. Available at: https://www.realfoods.co.uk/shop/fresh-fruit-and-veg [Accessed 1 Feb. 2020].

Veg, F. (2020). Fruit & Veg. [online] Greendale Farm Shop. Available at: https://www.greendale.com/fruit-and-veg/ [Accessed 1 Feb. 2020].

Wikipedia Contributors (2019). Class-responsibility-collaboration card. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Class-responsibility-collaboration\_card [Accessed 6 Feb. 2020].

Kirill Fakhroutdinov (2013). UML 2.5 Diagrams Overview. [online] Uml-diagrams.org. Available at: https://www.uml-diagrams.org/uml-25-diagrams.html [Accessed 6 Feb. 2020].

Wikipedia Contributors (2019b). Unified Modeling Language. [online] Wikipedia. Available at: <https://en.wikipedia.org/wiki/Unified_Modeling_Language>.

GeeksforGeeks. (2017). Unified Modeling Language (UML) | An Introduction - GeeksforGeeks. [online] Available at: https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/.

Visual-paradigm.com. (2019). Why UML Modeling? [online] Available at: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/why-uml-modeling/>.

martinfowler.com. (2020). Presentation Model. [online] Available at: https://martinfowler.com/eaaDev/PresentationModel.html [Accessed 6 Feb. 2020].

Thraka (2019). Data binding overview - WPF. [online] Microsoft.com. Available at: https://docs.microsoft.com/en-us/dotnet/desktop-wpf/data/data-binding-overview [Accessed 6 Feb. 2020].

Uregina.ca. (2014). Data Modeling and Entity Relationship Diagram (ERD). [online] Available at: http://www.cs.uregina.ca/Links/class-info/215/erd/ [Accessed 10 Feb. 2020].

Asq.org. (2019). What is a Matrix Diagram or Chart? Data Relationship Analysis | ASQ. [online] Available at: https://asq.org/quality-resources/matrix-diagram [Accessed 20 Nov. 2019].

Lucidchart.com. (2017). ER Diagram Tutorial | Lucidchart. [online] Available at: https://www.lucidchart.com CapTech Consulting. (2014). Resolve your Many-to-

Manys for Accurate Requirements. [online] Available at: https://www.captechconsulting.com/blogs/resolve-your-many-to-manys-for-accurate-requirements [Accessed 10 Feb. 2020].

‌/pages/er-diagrams.

FreeLogoDesign. (n.d.). Logo Maker - Create Your Own Logo, It’s Free! [online] Available at: https://www.freelogodesign.org/. [Accessed 15 Feb. 2020].

Hatchful. (2018). Hatchful | Your digital logo designer. [online] Available at: https://hatchful.shopify.com/.

Nih.gov. (2019). Types of Color Blindness | National Eye Institute. [online] Available at: https://www.nei.nih.gov/learn-about-eye-health/eye-conditions-and-diseases/color-blindness/types-color-blindness.

paletton.com. (n.d.). Paletton - The Color Scheme Designer. [online] Available at: https://paletton.com [Accessed 15 Feb. 2020].

Kalyan, S. (2016). Accessibility for mobile apps. Just go for it. [online] Medium. Available at: https://blog.prototypr.io/accessibility-for-mobile-apps-just-go-for-it-aeb746b87ac5 [Accessed 16 Feb. 2020].

Android Developers. (n.d.). Make apps more accessible. [online] Available at: https://developer.android.com/guide/topics/ui/accessibility/apps [Accessed 16 Feb. 2020].

Legislation.gov.uk. (2010). Equality Act 2010. [online] Available at: http://www.legislation.gov.uk/ukpga/2010/15/contents.

Visual-paradigm.com. (2019). What is Class Diagram? [online] Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/.

Rumbaugh, J., Booch, G. and Ivar Jacobson (2010). The unified modeling language reference manual. Boston: Addison-Wesley.

Reese, G. (2000). Database programming with JDBC and Java. Cambridge, Mass.: O’reilly.

‌

‌Thraka (2019). Data binding overview - WPF. [online] Microsoft.com. Available at: https://docs.microsoft.com/en-us/dotnet/desktop-wpf/data/data-binding-overview.

‌

‌

‌

‌

‌

‌

‌

‌

‌

‌

‌

‌

‌

‌