Graded Unit 2

THE REPORT

LEE GLEN

Contents

The Project Brief

The client I'm working with owns a small retro gaming store which sells video games and consoles that can be classed as retro to customers that are either collectors, people who want to re-live their nostalgia or to understand why these games where highly regarded. The client will get their game products through either suppliers that still sell these games or reproductions of the games, trade-ins from the customers or very rarely through online sellers like e-bay or amazon and then sell them at a higher price for profit.

How their business currently goes is that the customer brings a product to the till the employees scans the product, the client pays, employee gives the receipt to the customer and then the customer leaves the store. If the client wants a refund the employee checks the details with the details they've got and then either gives them the refund or tells them that they can't. If the customer wants to trade in a game/console the employee takes the game and gives the customer money equivalent to the game/consoles worth.

How their business deals with creating reports is by doing by hand for example if they want to understand what products where the most successful is by checking how many games they've currently have in stock by hand and then writing down the information. Same goes with storing details about customer, product and supplier information is all on-paper and they want to move onto digital for conveyance sake so that rather than going through multiple folders of information trying to find the info that they need.

The Interpretation of The Brief

One way to create this application is to make it a web-based application using HTML/CSS/PHP with a database backend to store information needed using SQL. Another way to create this application is to create a Java application with an integrated SQL database in the backend. Another way is to create a mobile Java application that saves the information stored in the application into a .txt file in the mobile.

I'm going to choose to create this application using Java with an SQL database in the backend as the client is only going to use this on a PC so there's no point of making a website just for one PC and it's the same reason why the mobile app shouldn't be made either as they specifically want it on a computer.

The main reason I'm picking Java to program this application is because it's robust to use as it can find possible errors before any other languages, can be easily moved to any other platform so let's say if the client wants the application to be on mobile as well it won't take a long time to move that application and that it's secure. The reason why I'm using SQL is that it's more secure than just saving the information to a .txt file and a lot easier to get information from a database rather than a .txt file by using SQL commands.

Functional & Non-Functional Requirements

Functional

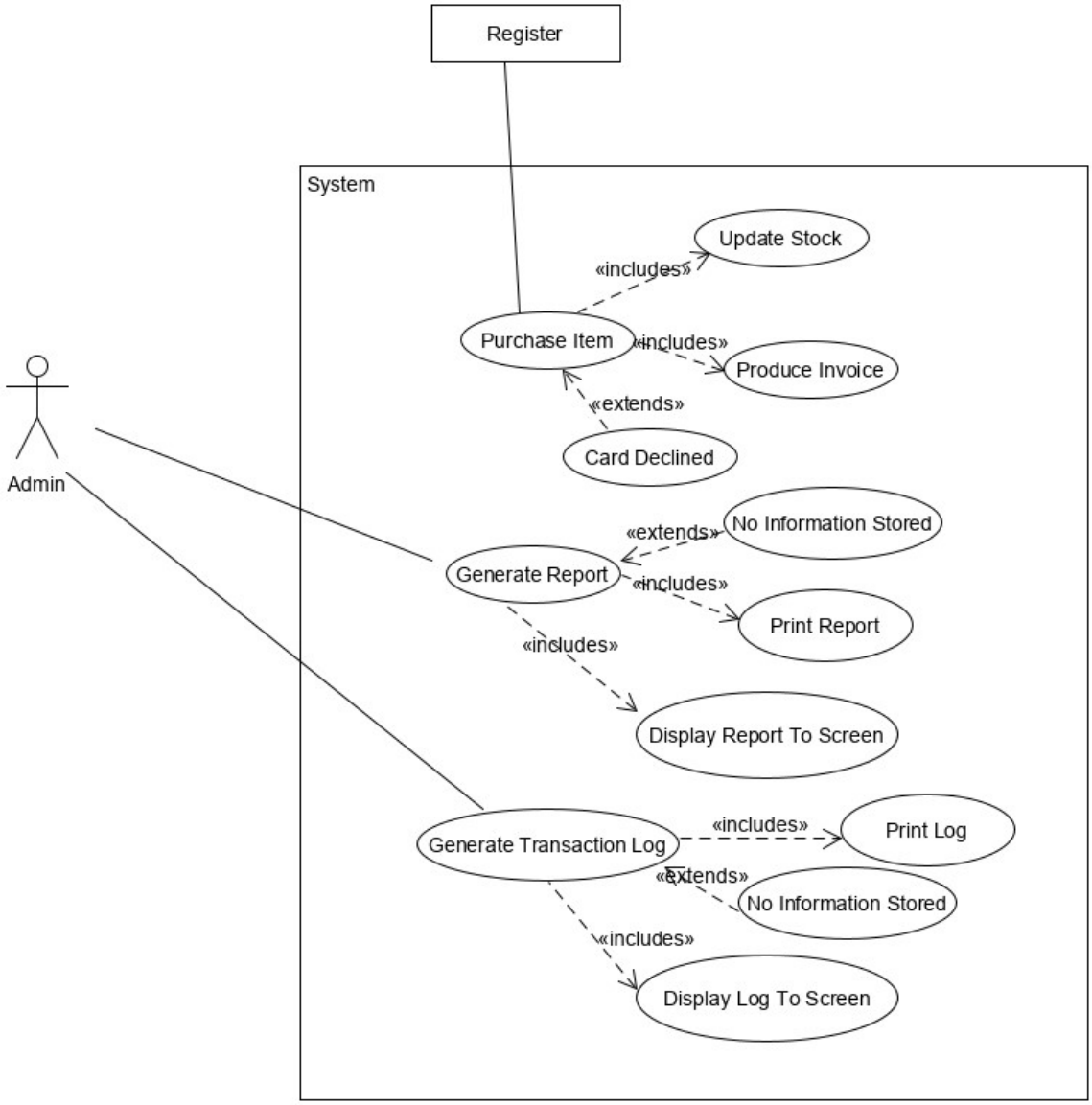
1. Produce a sales invoice for each transaction
   1.1 Takes in a payment
      1.1.2 By cash
      1.1.3 By card.
   1.2 For refunds check the information stored in the database with the receipt the customer will have.
   1.3 If a customer trades in a game:
      1.1.1 The store will take in the game
      1.1.2 Store gives the customer money that is assigned to that game.
2 Update stock after a transaction
   2.1 When an admin checks the stock report the stock for the product if out of stock should have 0.
   2.2 If the customer trades in a game the stock of that product should go up one.
3 Create either screen report or paper report of stock and sales
   3.1 Most of the reports will be on screen and some will be paper.
   3.2 The stock reports will have:
      3.2.1 Stock outages –The products that are currently out of stock that needs to be stocked
      3.2.2 Reorder report – What products that need to be reordered.
4 On-screen displays of customer, product, supplier and invoice information including transaction stock
   4.1 The details of customers, products, suppliers are:
      4.1.1 Customer details like:
         4.1.1.1 name
         4.1.1.2 account number if purchase by card
         4.1.1.3 address for trade-in
      4.1.2 Product details like:
         4.1.2.1 the name of the product,
         4.1.2.2 how many are still in stock
         4.1.2.3 the price of the product
         4.1.2.4 product type.
      4.1.3 The supplier's details like:
         4.1.3.1 suppliers name,
         4.1.3.2 supplier type.

Non-Functional

1. Runs on equipment with either Windows XP or newer or Linux or Mac OS X
2. Either a console-based app or GUI app

3. Doesn’t need a Database Management System but could be if wanted.

Top-Level Use Case Diagram



Information Gathering – Research for Brief

Interview with Client – 6/02/2019

I had an interview with the client talking about the initial functional and non-functional requirements and the top-level use case. I showed the client the top-level use case explaining what I think how the process will go, he said the purchase item use case was what he wanted however he expected there to be an employee to have some way to update the stock through this system. He also mentioned that he currently just wants the reports and transaction logs to display on screen rather than being on paper as he rather it be digital. He also mentions that this system will be used in the back of the store for use for the employees and the admin.

We also talked through the functional/non-functional requirements and explaining to him what I think will happen during usage of this system. He mentioned that there should be a way for an employee of the store to search for a product either through name or number so that they know it's in stock for the customer to buy and this was brought up when we were talking about 1 of the functional requirements.

When talking about 1.2 the client mentioned that I should take into consideration on-long the customer has bought the product and also told me that the shop will give the customer a refund if the product was faulty when they bought it and didn't know following UK law.[1]

When we were talking about 1.3, he mentioned that he needed to know what employee had taken in the game if something happened and needs a way to look up the customers game price for trading-in so that we can give the customer the exact money that is assigned to that game. He also mentioned that he needs to store information about the customer when they've traded-in a game so that if we find out the game was stolen; we know who sold it to us – Their name and address.

When we talked about 2 the client mentioned again that the system needs to track who has updated stock so that they can track down any employee that has either accidently entered or removed too much stock or has done it intentionally.

When discussing 3.1 he mentioned again that all of the reports will be on-screen rather than some of them being paper and we discussed 3.2 the client mentioned that there should be a report for just every item that is currently out of stock so that they know what items they will need to stock up on.

With 4.1 he mentioned that the client wants to have the employee's information held which as well includes their name, address, phone number, job title and salary so that they can

---

1 https://www.gov.uk/accepting-returns-and-giving-refunds - UK law dealing with returns and refunds

know which employee sold an item specified in an invoice. For the product he wants an item number for every product that will uniquely identify them from each other.

Looking at Other Systems Similar to Brief

Through looking around the World Wide Web I have found various websites that are like my client's business which also has some answers to what the client wants in the brief:

Retrogames.co.uk - (Leyawin Media Services Ltd, n.d.)

This business fits with the client's business as both businesses deal with selling retro games and consoles.

How this business does their trade-ins is by either allowing the customer to just send in the game if they just want the game anymore or send in the game and get money in the process via postage.

With the first option they just ask the customer to send in the game to their address and asks them to include a note with the package with the customer's name and address to gain payment back for mail fees or they can just use bank transfer or PayPal.

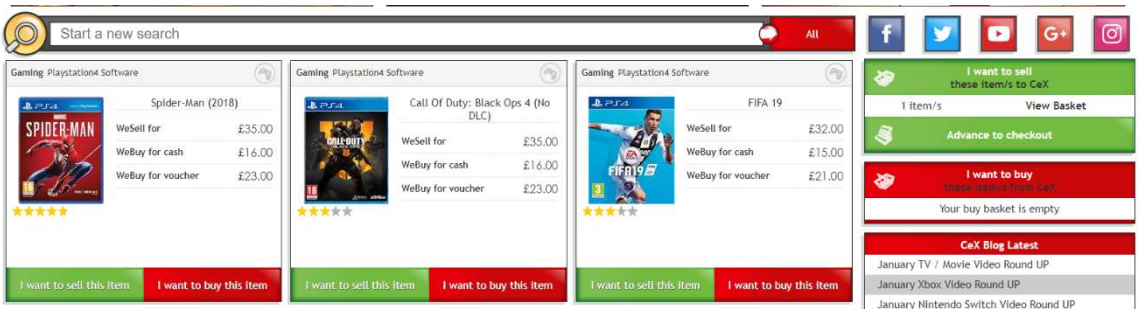
With the second option the customer will need to email the business what they want to sell and what the condition of the game is, and the business will negotiate a deal with the customer which could be high or low depending on the condition of the game and the rarity of the game. Similar to the first option the customer will send the game down to the business and will get the money either by bank transfer, PayPal or cash on delivery.

In the terms and conditions[2] of the website the way they do returns is that if you don't want the product anymore you can send back the product to the business within seven days and in its original packaging and when the package is received by the business, they refund the customer.

The website also has a search function which allows the customer to look for products through all of the products being sold or by a specific console.

CEX - (CEX, n.d.)

This business is similar to the clients as this business sells various forms of entertainment which includes retro gaming.



How this business deals with trade-ins is that on their website you click on the games that you own and want to sell, click on your sell basket, tell the business what why you want to be paid by – PayPal, voucher, bank transfer, cheque – and then the customer sends the products to the business and then receives their payment.

The way they do returns is that if the customer doesn't won't their product anymore, they can take their product to a nearby store with the invoice between seven days after purchasing and get refund there or by sending the product back by package to the business

2 https://www.retrogames.co.uk/Terms_And_Conditions.html - Second last paragraph

and then the company will fully refund the customer through the details given already through the original order.

They again also have a search function which allows the customer to look for a specific product.

Game - (Game Retail Ltd, n.d.)

This business is fairly similar to the client's business even though this business focuses on more modern gaming and the clients is retro gaming the procedures would still be the same between them.

How this business does their trade-ins is by either the customer takes their game into the store and askes the business how much they want for it or do it through online by finding the games they want to sell.[3] It then asks for either the customers PayPal or if they want in-store credit they enter the business card associated with the business for a top-up. Then like the two examples above sends the games to the business and then gets the money up to 5 working days.

How this business deals with returns is that similar to CEX you can come into the customers nearest store up to thirty days after purchasing with the game still being in its original condition when bought and either the invoice or receipt and get their refund through the store or send the product back to the business through postage. The customer will get their refund either by sending it into the card account they purchased with or through PayPal.[4]

The customer can use the search function on the website as well to look for products they want to either buy or trade-in.

3 https://tradein.game.co.uk/ - How to trade-in online

4 https://www.help.game.co.uk/hc/en-gb/articles/360000136465-Returning-an-Item - The process of returning a product.

Conclusion of looking at other systems

Even though some of these businesses deal with trade-ins/returns through postage, the customer can still come in and deal with trade-ins and returns which makes more sense with the client's problem as the client's business doesn't allow customers to buy items online.

Through looking at some of these systems I can see what will be included into this prototype and aligns with what the client needs for example in all three of these systems you'd need to take in the customers personal details as you'd need to log-in to create purchases or selling games/consoles, keep payment information if product handed in was stolen in the first place. The difference is that the customer has to come into the store and give the employee proof of who they say they are, and the employee will need to take in the details of the customer if the customer was going to try and trade in a game.

All of these systems have a search feature which the client asked for during the interview but with our system the admin & employees will be the ones searching for products. With CEX they've got the prices for trading-in in their details which is also what the client wants when discussing about the prototype during the interview.

Most if not all of these systems have some kind of way to detect how long the customer has purchased an item which would be used in the case of returns which the client has asked for during our interview as there is a cut-off date with the returns of a product usually recommended to be seven days.

Both CEX & Game if the customer wants to return the game, they've got an option to take it to a store which ties in with what the client has in their brief so the procedure of those stores dealing with refunds will be similar to how the client's business will deal with it.

Aims of the Brief

The client wants an application created that allows the client to view the stocks of their products to see what needs to be stocked up on, any transactions that have happened on a day to see how many products they've sold to be used on a computer in the back of the store.

Through this application the client will want to take in a payment and store this payment that has been made due to the fact that if the customer wanted to refund a game, they can check the account details of the card and see if the matches with the receipt and through this payment update the stock in real-time by either removing or adding if the customer is returning the game or trading-in a game the customer will need to give the employee their information with proof and that will need to be stored. They'd also like some kind of search system with this application so that if a customer wanted to know if a specific game was in stock, they'd just need to search for the game and see if it is in stock or not.

They also want some kind of way to track what is coming into the system by having the employees name linked to the amount of stock that was added into the system by them so

that it stops the employee from possibly being malicious with adding bogus information about the stock of certain products unless it's accidental.

They'd like some kind of way to display reports based on how well their business has done on that week for example how many consoles they've sold by week, by month and by year to see what consoles are selling are the most popular or to see what games/consoles that are out of stock so that they can stock in those games. They'd also like to have some kind of report on what the most popular games of that month was to understand what products they need to stock more of for the next month.

New Functional/Non-Functional Requirements

Functional Requirements

1. Purchase Item
   1.1 System will take in a purchase and check the name of the item bought
   1.2 System will then search for the product to see if the product is in the database.
   1.3 The system will then take one off of the stock of the product.
2. Generate Reports
   2.1 The client will then select from a list of reports and the system will get the information from the database and then display the information on screen.
   2.2 The kind of reports that can be displayed is:
      2.2.1 All Products
      2.2.2 Products that has no stock
      2.2.3 Most popular products
      2.2.4 Employee sold the most products
      2.2.5 Products sold by a specific:
         2.2.5.1 Week
         2.2.5.2 Month
         2.2.5.3 Year
3. Add Stock
   3.1 The person adding stock will then need to enter their name to verify who entered stock.
   3.2 To find the product to add stock on they'd need to enter the product id or the product name.
   3.3 If the product is found, then they can then enter in the amount of stock into the system.
4. Search Product
   4.1 Similar to 4.2 they can search for an item either by product name or by product id.
   4.2 The system will then search for that name or id in the database to see if it is available
   4.3 If found the information about said product will then display on-screen like.
      4.3.1 Name
      4.3.2 Type
      4.3.3 If game Genre

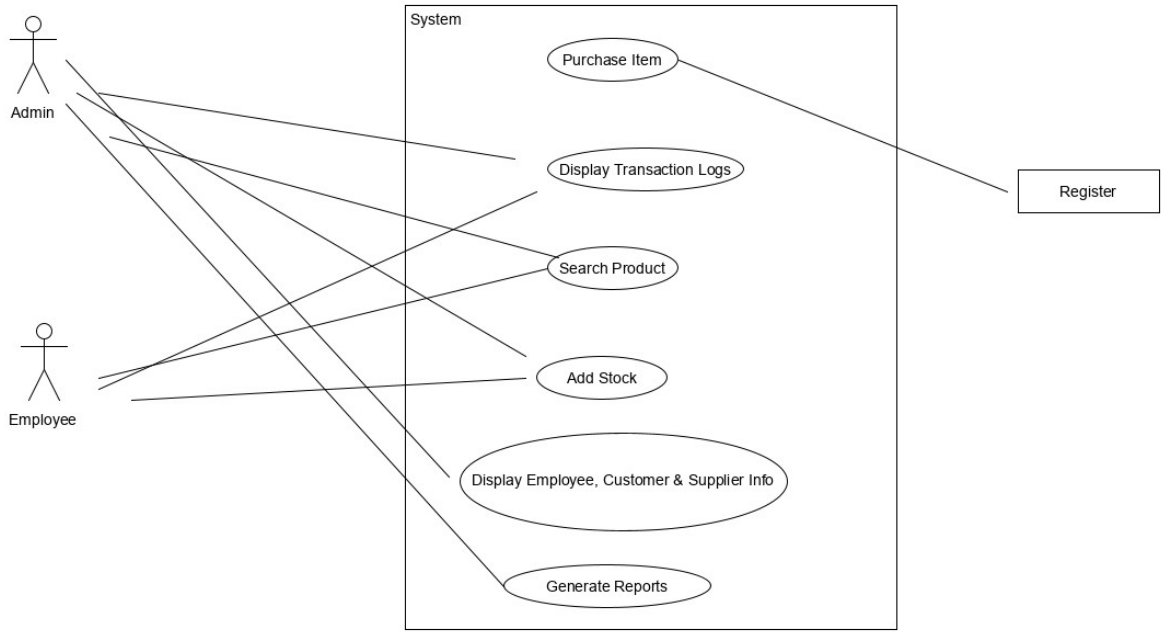
4.3.4 Publisher/developer
4.3.5 Price
4.3.6 Trade-in price

5. Display transaction logs
   5.1 Will display all the transactions on a certain date specified by the client which would include the employee who sold the product to the customer and the products that was sold and the time/date it was sold at and a special id which will be used to check receipts to see if they're valid for a refund.
6. Display employee, customer or supplier details
   6.1 The client will select an option to see either the employee, customer or the supplier details.
   6.2 If the client selects the employees, then the details of the employees will display like
      6.2.1 Name
      6.2.2 Job type
      6.2.3 Phone Number
      6.2.4 Address
      6.2.5 Salary
   6.3 If the client selects the customers, then the details of the customers who've traded-in will display like
      6.3.1 Name
      6.3.2 Address
   6.4 If the client selects suppliers, then the details of the supplier will display like
      6.4.1 Name
      6.4.2 Address
      6.4.3 Type

Non-Functional requirements

1. Runs on equipment with either Windows XP or newer or Linux or Mac OS X
2. Either a console-based app or GUI app
3. Doesn't need a Database Management System but could be if wanted.

New Top-Level Use Case Diagram



Identification of Resources Required

To create the prototype, I'll have to use various software and hardware to design and program the prototype. Hardware wise I will need to use:

- College Computers + Home PC – I need a workspace in college and in my home to design/program the prototype since I won't be in college all the time. I have already got a PC at home and I can use most of the computers that are currently in the computing department of college and the library.
- Laptop – As I live with a relative during college time, I will need to use something portable that is able to program/design the prototype. I already have a laptop bought so I can use that one for the creation of this prototype.
- Internet Access – If I need to look at tutorials or look at the Java API or to store any information onto the cloud then I'll need internet access either at the college or at home. I have WIFI at home and an WIFI adapter in my computer and the college computers are connected to a network which has access to the World Wide Web.
- Printer – I will need a printer to print out any documentation that the client would like to see as the client likes to have physical documentation rather than digital. I can print anything I need in college as there is printers inside the computing department or at the library.
- A USB Flash Drive – Will need this to transfer documents & the prototype itself to other computers or to use software that isn't on certain computers mainly the college computers in the library or a few that aren't installed on the computing departments computers. I have four already at home with various different sizes – The current one I use is 64 GB, so I don't think I need to worry about maxing out size.

Software wise I'll need to use various programs so that I can various things like:

- IntelliJ – As I'm creating the prototype through Java, I'm going to use IntelliJ which an IDE specifically made for Java programming and in my honest opinion is the better option from eclipse and NetBeans as IntelliJ has much more feature wise and is a lot smarter than the other IDEs. Because I'm a student at Forth Valley I can get the Ultimate Edition of IntelliJ for free on their website.
- UMLet – This will be mainly for the design of the prototype for creating the use case diagrams, action diagrams, sequence diagrams and class diagrams. I can get piece of software for free on their website for my home computer and there its already installed in the computing department computers.
- Data Modeler – This piece of software will be used to create ERD's for the prototype as I will be using a database for the storage of data. I can get this software on Oracles website for free. However, it seems that I can't use this software specifically with the college computers as I need to use someone else's computer when they're logged in to use the software, but I can use it on both my laptop and my home computer.
- Chrome – For a web browser I'll use chrome to access the World Wide Web to find information needed like for example using the Java API to help me find an answer to a programming problem, look at old notes to find any code that I have that already answered a problem, potentially looking at tutorials online to learn new information on a certain topic. This web browser is free on googles website and is on both my home pc and laptop and is on all the PC's in the computing department in college.
- Microsoft Word – For creating documents to either keep track on what I've done on this day like a diary or to document everything that I do throughout the lifecycle of this prototype. For college students' word is free on Microsoft's website and again is already installed on both my home PC and laptop and is on every computer in college.
- Adobe Reader/Soda PDF Reader – These will be used to open and view PDFs – Adobe Reader at college and Soda PDF Reader at home or on the laptop. Adobe Reader is what I'll use in college as it is installed on the college computers and is available on adobe's website for free however on my home PC and laptop, I use Soda PDF Reader just because that is what is installed on my pc and is completely free and can be downloaded from Soda's website.
- Notepad++ - Mainly for backing up code specifically if I'm testing various new ways on how to fix a problem, I'll like to keep the initial code there on the notepad as undo has a set limit on most IDEs. Notepad++ is completely free and can be downloaded from their website plus it's on both my home PC and laptop and is installed on all the college computers in the computing department.
- Microsoft Project – For creating and updating the Gantt chart to keep track on what I'm doing, what I've done and what I'll be doing the next procedures. Due to me being a Forth Valley student I can get access to their Microsoft Imagine and download project from there and will give me a free licence key with it as well which will be both installed on my home PC and laptop plus it's already installed on the college computers.

- Windows 10/7 – As I need some kind of operating system to use any of the programs listed above It'll mainly be windows specifically windows 10 and 7 as my PC & laptop have Windows 10 and the college computers have windows 7 installed. Windows 7 is the operating system that is currently installed in most if not all the computers in college and Windows 10 is the operating system that is installed on my home PC and laptop. I can download a version of Windows 10 from Microsoft Imagine.

Information Sources to Be Used

Mainly all my information will come from the internet like:

- Java API - (Oracle, n.d.) – This will mainly be used for looking up stuff about the Java code itself so like looking at different classes or methods and seeing what they do and what their utilized for.
- Youtube - (Google, n.d.) – This will be used mainly for looking up tutorials to further understand topics that I may have no knowledge on or have some knowledge on and want to further improve on it.
- ILearning - (Oracle, n.d.) – This mainly has all of the courses I've been taken where it be SQL and design with databases to Java programming I can use these if I have forgotten anything I can look back to these courses and remember.
- Packt – (Packt, n.d.) - Packt has a lot of information on various programming languages and video tutorials teaching fundamentals on various languages or designs which could be used to gain more knowledge on Java and SQL and how to integrate them.
- Codeacademy – (Codeacademy, n.d.) -This is similar to Packt but the courses that are Codeacademy are more hands-on as you are typing in the code as you're learning which helps in a way as each time you type the code in the more you remember what it is and what it does.
- W3Schools – (W3Schools, n.d.) - This website has a wide selection of programming languages with tons of information with examples like if I wanted to learn how to create a for loop in Java I can look on this website and it can show me an example where this loop can be used.
- Geeksforgeeks – (GeekForGeeks, n.d.) - This website has a backlog of code that can be potentially taken and revamped for a new purpose and similar to W3Schools and Packt it has tons and tons of information on programming languages.
- Tutorials Point - (Tutorialspoint, n.d.) – This website like the examples above has a backlog of information about various programming languages and has tons of video tutorials which goes into detail about what the programming language is, what it can be used for and how to code in this programming language.

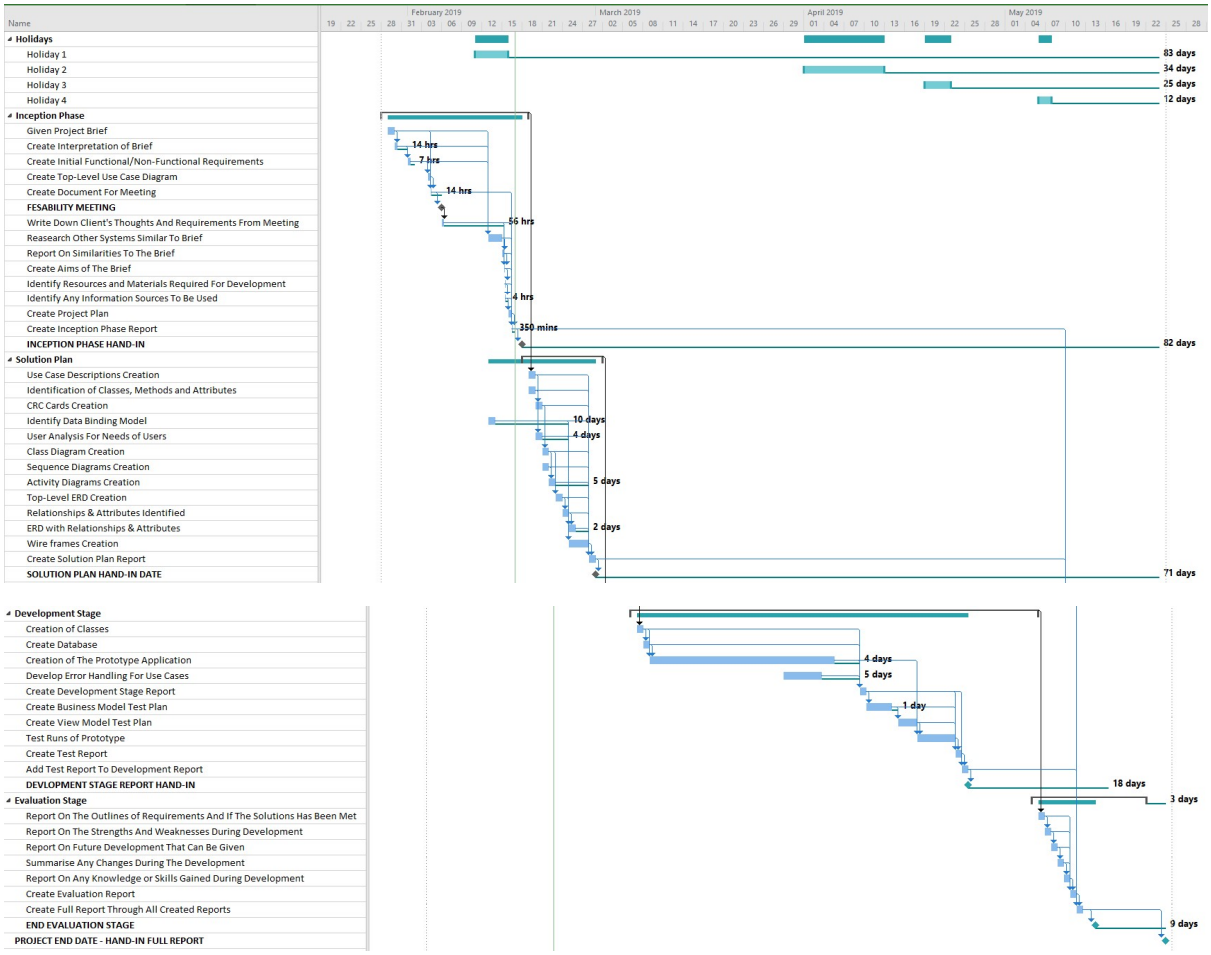
The reason I have various websites to be used for information – Packt, Codeacademy, W3Schools, GeekForGeeks and Tutorial Point – is that if I have a problem specifically with something related on implementation then I don't need to be restricted to just

one website meaning that I can look onto another website and see what how they've tackled that problem.

Project Plan

For the development methodology, I will be following an iterative waterfall method as it would be well suited for a problem like this and waterfall is well suited for milestone-focused development and also, I'm the only person developing the prototype I won't have other people focusing on different parts, so I'll have to do one after another like a waterfall.

Gantt Chart



Main Tasks

Name	Duration	Start	Finish	Predecessor	Resource Initials
Holidays	**76 days**	**Mon 11/02/19**	**Tue 07/05/19**		
Holiday 1	5 days	Mon 11/02/19	Fri 15/02/19		
Holiday 2	12 days	Mon 01/04/19	Fri 12/04/19		
Holiday 3	4 days	Fri 19/04/19	Mon 22/04/19		
Holiday 4	2 days	Mon 06/05/19	Tue 07/05/19		
Inception Phase	**17 days**	**Mon 28/01/19**	**Mon 18/02/19**		
Given Project Brief	1 day	Tue 29/01/19	Tue 29/01/19		L,MW,USB,CC,LP
Create Interpretation of Brief	2 hrs	Wed 30/01/19	Wed 30/01/19	7	L,MW,USB,CC,HC
Create Initial Functional/Non-Functional Requirements	1 hr	Fri 01/02/19	Fri 01/02/19	8	L,MW,USB,CC,HC
Create Top-Level Use Case Diagram	1 hr	Mon 04/02/19	Mon 04/02/19	9	L,HC,UML
Create Document For Meeting	1 hr	Mon 04/02/19	Mon 04/02/19	7,8,9,10	L,HC,UML,USB,MW
FESABILITY MEETING	0 days	Wed 06/02/19	Wed 06/02/19	11	L
Write Down Client's Thoughts And Requirements From Meeting	1 hr	Wed 06/02/19	Wed 06/02/19	12	L,MW,CC,USB
Reasearch Other Systems Similar To Brief	2 days	Wed 13/02/19	Thu 14/02/19	7,8,9	L,HC,MW,CH,WWW
Report On Similarities To The Brief	1 hr	Fri 15/02/19	Fri 15/02/19	14	L,HC,MW
Create Aims of The Brief	1 hr	Fri 15/02/19	Fri 15/02/19	15,13	L,HC,UML,MW
Identify Resources and Materials Required For Development	1 hr	Fri 15/02/19	Fri 15/02/19	16	L,CH,HC,WWW,MW
Identify Any Information Sources To Be Used	1 hr	Fri 15/02/19	Fri 15/02/19	17	L,CH,HC,WWW,MW
Create Project Plan	2 hrs	Sat 16/02/19	Sat 16/02/19	18	L,HC,PT
Create Inception Phase Report	10 mins	Sat 16/02/19	Sat 16/02/19	19,18,17,16,1	L,HC,USB,MW
INCEPTION PHASE HAND-IN	0 days	Mon 18/02/19	Mon 18/02/19	20	
Solution Plan	**12 days**	**Mon 18/02/19**	**Fri 01/03/19**		
Use Case Descriptions Creation	1 day	Tue 19/02/19	Tue 19/02/19	6	L,CC,MW,LP,USB
Identification of Classes, Methods and Attributes	1 day	Tue 19/02/19	Tue 19/02/19		L,CC,LP,MW,USB
CRC Cards Creation	1 day	Wed 20/02/19	Wed 20/02/19	24	L,CH,CC,HC,WWW,USB
Identify Data Binding Model	1 day	Wed 13/02/19	Wed 13/02/19		L,CC,HC,MW,USB
User Analysis For Needs of Users	1 day	Wed 20/02/19	Wed 20/02/19	23	L,CC,HC,MW,USB
Class Diagram Creation	1 day	Thu 21/02/19	Thu 21/02/19	25	L,HC,UML
Sequence Diagrams Creation	1 day	Thu 21/02/19	Thu 21/02/19		L,CH,HC,WWW
Activity Diagrams Creation	1 day	Fri 22/02/19	Fri 22/02/19	29,28	L,CH,CC,HC,WWW
Top-Level ERD Creation	1 day	Sat 23/02/19	Sat 23/02/19	28	L,DM,HC
Relationships & Attributes Identified	1 day	Sun 24/02/19	Sun 24/02/19	31	L,DM,HC,MW
ERD with Relationships & Attributes	1 day	Mon 25/02/19	Mon 25/02/19	31,32	L,DM,HC
Wire frames Creation	3 days	Mon 25/02/19	Wed 27/02/19	26,27	L,CH,HC,WWW,CC,LP,USB
Create Solution Plan Report	1 day	Thu 28/02/19	Thu 28/02/19	34,33,32,31,3	L,HC,MW,CH,DM,WWW
SOLUTION PLAN HAND-IN DATE	0 days	Fri 01/03/19	Fri 01/03/19	35	
Development Stage	**55 days**	**Fri 01/03/19**	**Fri 03/05/19**		
Creation of Classes	1 day	Sat 02/03/19	Sat 02/03/19	22	L,HC,IJ
Create Database	1 day	Sun 03/03/19	Sun 03/03/19	38	L,DM,HC,IJ
Creation of The Prototype Application	23 days	Mon 04/03/19	Mon 01/04/19	38,39	L,HC,IJ,CC,USB
Develop Error Handling For Use Cases	6 days	Mon 25/03/19	Sat 30/03/19		L,CC,HC,IJ,USB
Create Development Stage Report	1 day	Sat 06/04/19	Sat 06/04/19	40,41,38,39	L,IJ,MW,HC
Create Business Model Test Plan	4 days	Sun 07/04/19	Wed 10/04/19	42	L,CC,HC,MW
Create View Model Test Plan	2 days	Fri 12/04/19	Sun 14/04/19	43	L,CC,HC,USB,MW
Test Runs of Prototype	6 days	Mon 15/04/19	Sat 20/04/19	40,43,44	L,CC,HC,IJ,MW,USB
Create Test Report	1 day	Sun 21/04/19	Sun 21/04/19	42,43,44,45	L,HC,MW
Add Test Report To Development Report	1 day	Mon 22/04/19	Mon 22/04/19	46,42	L,HC,MW
DEVLOPMENT STAGE REPORT HAND-IN	0 days	Tue 23/04/19	Tue 23/04/19	47	
Evaluation Stage	**14 days**	**Fri 03/05/19**	**Mon 20/05/19**		
Report On The Outlines of Requirements And If The Solutions Has Been Met	1 day	Sat 04/05/19	Sat 04/05/19	37	L,HC,MW
Report On The Strengths And Weaknesses During Development	1 day	Sun 05/05/19	Sun 05/05/19	50	L,HC,MW
Report On Future Development That Can Be Given	1 day	Mon 06/05/19	Mon 06/05/19	51	L,HC,MW
Summarise Any Changes During The Development	1 day	Tue 07/05/19	Tue 07/05/19	52	L,HC,MW
Report On Any Knowledge or Skills Gained During Development	1 day	Wed 08/05/19	Wed 08/05/19	53	L,HC,MW
Create Evaluation Report	1 day	Thu 09/05/19	Thu 09/05/19	54,53,52,51,5	L,HC,MW
Create Full Report Through All Created Reports	1 day	Fri 10/05/19	Fri 10/05/19	55,47,35,20	L,HC,MW
END EVALUATION STAGE	0 days	Mon 13/05/19	Mon 13/05/19	56	
PROJECT END DATE - HAND-IN FULL REPORT	0 days	Fri 24/05/19	Fri 24/05/19	56	

Resource List

Resource Name	Type	Material	Initials
Lee Glen	Work		L
College PC	Material	Hardware	CC
Home PC	Material	Hardware	HC
Laptop	Material	Hardware	LP
Printer	Material	Hardware	P
USB Flash Drive	Material	Hardware	USB
Umlet	Material	Software	UML
InteliJ	Material	Software	IJ
Chrome	Material	Software	CH
Internet	Material	Hardware	WWW
Word	Material	Software	MW
Project	Material	Software	PT
PDF Reader	Material	Software	PDF
Notepad++	Material	Software	N++
Data Modeler	Material	Software	DM

Design Stage

Changes before going ahead

Before I start I have changed from the iterative waterfall method to agile due to the timescale of having to do all the design in one week would be too much and would be much more sensible to focus one part of the design rather than the full on design and I can just do the design for the second and third iteration while in the implementation stage. That does also mean changing the project plan to fit the new agile methodology with the three iterations.

I also forgot to mention in my identification of resources and materials the use of MySQL with XAMPP which is used to create a server on your computer for use of seeing the websites you've created for potential testing and has MySQL support. That's what I'll use to get the database up and running for testing. You can get the application from the XAMPP website and download the app on the front page.

Looking back at the inception phase has brought up something I completely forgot to mention is that in the functional diagram and in the aims of the brief there should be some form of log-in for the employee and admin so that there is a differentiation between the two so that they can access different methods represented in the use case. Here is what it would represent in a functional requirement.

1. Log-in
   1.1 Employee or admin will enter their details into the two text boxes and then press a button.
   1.2 If the name and password match, then the employee or admin goes to the main menu
   1.3 The details the employee and admin are:
      1.3.1 Full Name
      1.3.2 Password

Techniques Used During Design

Business Model

During the business model I'll be using different design techniques for separate parts of the prototype. For the Java application I'll need to figure out on what person can do certain methods and how those methods will work out so I'll need to use the Use Case Diagrams which will show what person will be allowed to do specific methods and with those methods there will have includes which will show what else will happen during those

methods and extends which will show what will happen if there is a problem that occurs. Then I'll need to write out a description for each use case explaining what the admin/employee will do and what the system will do also. The Use Case Diagrams would be made in Umlet as I have the most experience using this piece of software and that it can also be used for various other diagrams like sequence and activity diagrams, there is others that I can use like draw.io or Visio but the problem is that I have used draw.io before and in my personal experience it has a lot of flaws particularly with movement of shapes and Visio I have no personal experience so I going to stick with what I know best which is Umlet.

I'll need to figure out the classes that will be used and the relationships between those classes and I can figure them out by using CRC cards which identifies what the class is, what it'll do/hold and what it'll relate to and then from those CRC cards I can then use class diagrams which will show the classes with the attributes & methods and their access modifiers that'll be inside the classes and see the relationships between each of the classes wither it be aggregation or composition. The CRC cards will be used with the website CRC card maker (Echeung, n.d.) as I've used it before and is quite convenient with already making the tables to fill in for you without having to make them yourself and the class diagrams will be made by Umlet again due to me having more experience with the software than others.

If I wanted to understand what occurs in the methods, I'll need to use sequence & activity diagrams. Sequence diagrams are used to show the interaction & messages given between objects in the order they'll interact at while an activity diagram is a more in-depth view on what actually happens during that method which it goes step-by-step on what will happen and what is the result. The sequence & activity diagrams will be made in Umlet again and as I said before more experience using Umlet.

For the back end of the prototype I'll need to design an ERD to represent the database using data modeler. An ERD (Entity Relationship Diagram) shows the tables and the relationships between those tables. It'll also show the attributes and data types of the tables so for example if we use customer as an example, they'll have name, e-mail, account number attributes. From that I can convert the physical model – Used to show the relational data like tables, primary & foreign keys - using data modeler and the reason I'm using this besides LucidCharts ERD tool or draw.io or ERDplus is that data modeler is the tool that I've made all of my ERDs on so I know it well enough and there isn't any point to go on and use a completely new thing when I have a timescale to follow and learning something new won't fit in the timescale and would waste time when I could easily create it with data modeler.

View Model

For the data binding model I'm going to use MVC (Model View Controller) pattern when developing the Java application front-end with the model holding data that will be manipulated, controller will be were all the data manipulation and were all the menus and searches will happen and view will be where all the GUI will be stored so alert boxes, choice

picker etc. The model will be used to link the data from the database back end into the java application front-end using JDBC to connect the database to the java app and MySQL for the database storage of customer, employee, supplier details etc. I'll be storing the table using XAMPP as again for many other software I've mentioned it's because I know it much more than say UwAmp and besides that XAMPP is easily usable and adaptable with IntelliJ which is the IDE I use and mentioned in the materials in the inception phase and UwAmp isn't adaptable.

There will be the use of wireframes to represent the GUI of the java application like the alert boxes, the home panel used for the menu. The wireframes will show me the layout of the prototype, where the buttons will be placed, input boxes that will be used, any pictures or specific colours that'll be used. Through Moqups I'll be making these wireframes as again I know this software better than let's say draw.io or using words shapes to make the wireframes and that Moqups has features that are beneficial for colour pallets and such.

I'll also need to use some sort of style guide to research up on and to follow so that possible employees who maybe colour blind or have some kind of problem with their eyesight can still use the program and also research a bit about accessibility and what to implement for example employees who has disabilities with their hands and finding a way to make it easier for them to use the system. What I'll be using to make this style guide specifically choosing the colour pallet is a combination of both the website - (Material, n.d.) & (Toptal, n.d.) – which for the first one will give me a wireframe and then I can pick colours that can match and then go and check the accessibility of being able to read text with these colours telling me whether or not to use either white or black text and the second website allows me to check how these colours will look like for a colour blind person which can be beneficial.

Iterations explanation

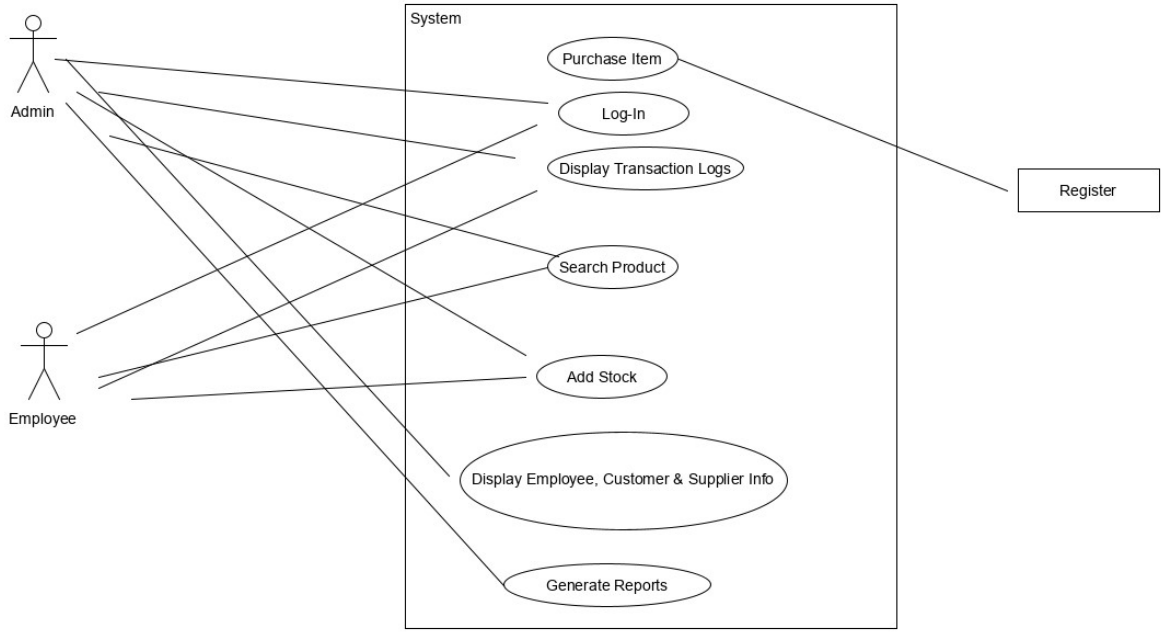
As I am using an agile methodology, I'm going to split the design & implementation into three different iterations and the first iteration will be focused on the database back-end and the UI of the Java application so the ERD's, physical model, wireframes and style/accessibility guide. I'll also include the use case diagram and then splitting the diagram to have more detail and then explaining briefly what each use case will do. This iteration will also include the creation of the database back end through data modeler.

he second iteration will be focused on the Java application which will include the creations of CRC cards, class diagrams, sequence diagrams, activity diagrams and then using the use case diagrams from the first iteration I will be making descriptions for each use case in the diagrams and then finally the start of creating the Java application using IntelliJ.

The third iteration will be connecting the Java front-end to the database back-end pulling out information from the database into the Java application and after the prototype is "finished" that's when the test runs start, and I'll note down any that've passed and that've failed and then potentially fix them if I have time to do so.

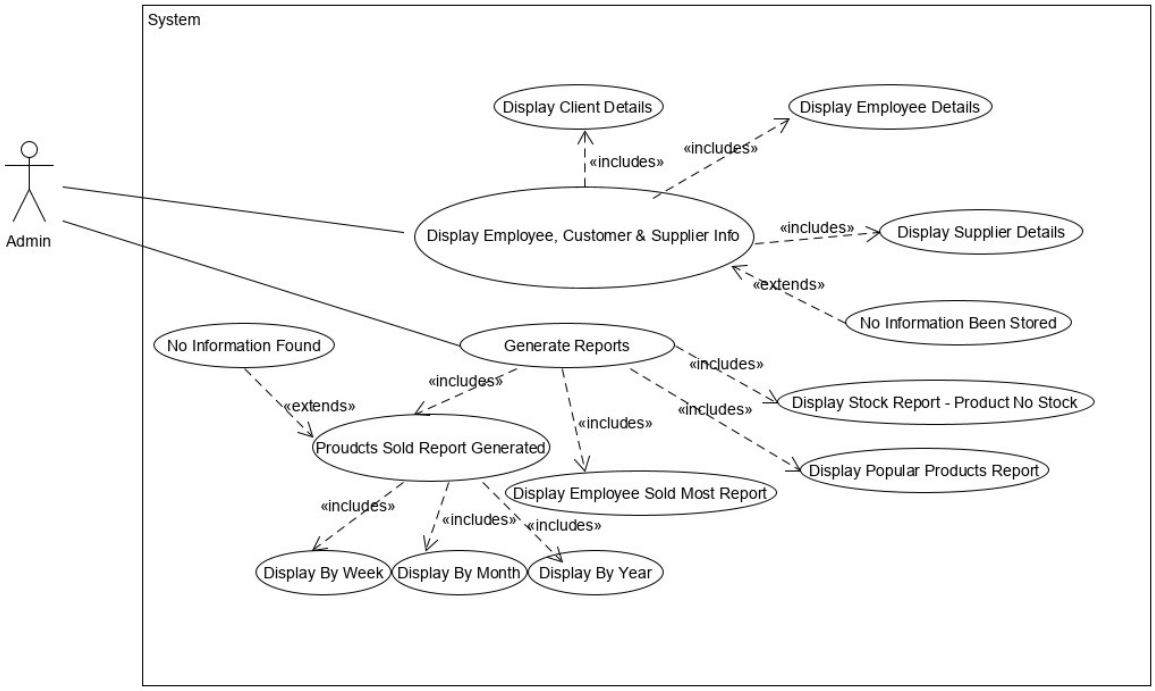
First Iteration

Top-Level Use Case



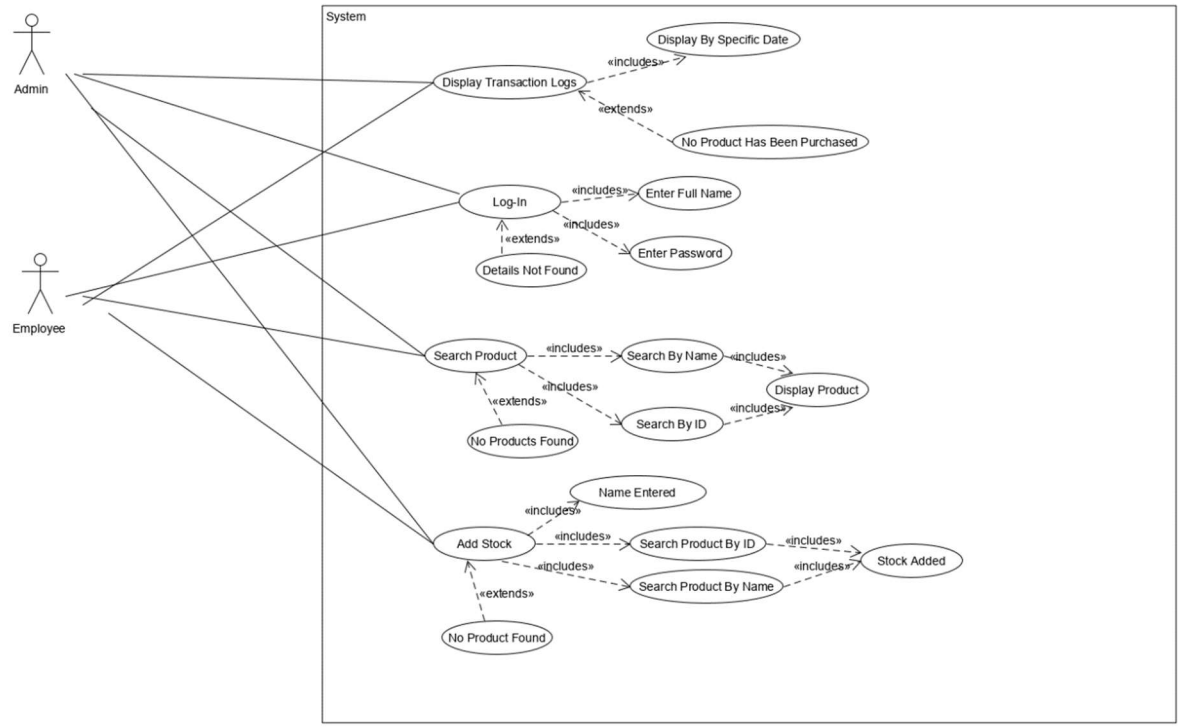
This is the top-level use case diagram that was in the inception phase which details on what the admin and employee will be able to do when using the application. What I'm going to do is break down the use cases by splitting it off to three use case diagrams – one for admin, one for both and one for the purchase item – and then give them a brief description on what the use case will do but not a step-by-step detailed description because that will be in the second iteration. The reasoning of splitting the diagram into three different use case diagrams is due to there being too much clutter if I did it on one big use case with all the includes and the extends inside and makes a lot easier to read and understand on what the employee and admin can do. I also added the log-in use case I mentioned above when talking about things that I forgot to include.

Use Case Diagram – Admin



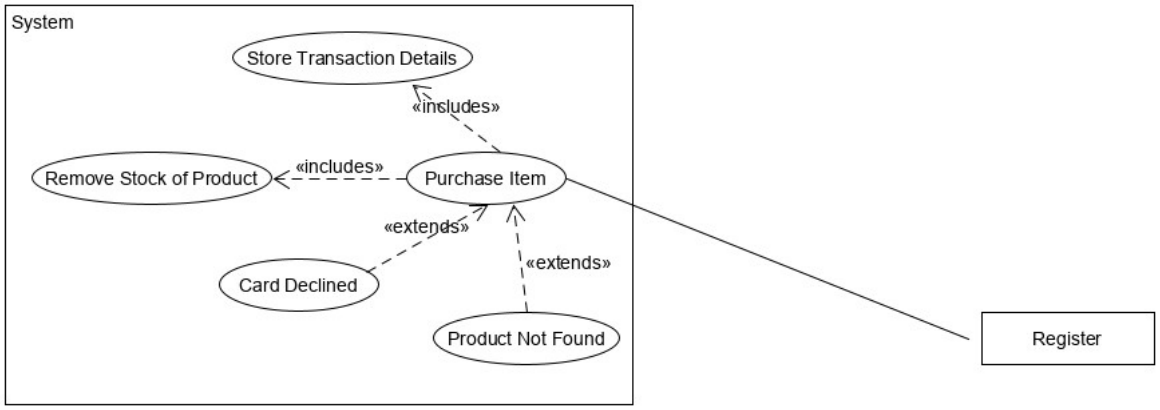
In this diagram it shows what the admin can uniquely do which is showing info and getting reports. The admin can specifically display either employee, customer or supplier info as they might want to contact let's say an employee doesn't come in for their shift the admin might want to get the employees phone number to call/text to find out why they're not in. They can also generate a multitude of reports from finding out what products are out of stock to how many products have been sold by a specific timeframe.

Use Case Diagram – Employee/Admin



In this diagram it shows commonality between the admin and employees on what they can do in the system. Both the admin and employee can display transaction logs for the use of refunds to see if a transaction has been taken on a specific date and the data from that transaction stored on the database is the same as the id given to the customer through a receipt. Both can search a product which can be for looking up a specific product to see how much stock is available for that product or better yet if a customer was wanting to see if they've got the product the customer has specified in their store both can find out. Both can also add in stock if packages have come in and they needed to be added to the stock, they'd need to enter their name just so there isn't any bogus stock added and if so, the admin can trace it back to an employee. For the log-in use case either the employee or the admin will enter their full name and password which if successful will lead them to a menu containing the other use cases, if not then a message will appear telling the employee/admin that they have entered incorrect information and then allow them to enter the information again.

Use Case Diagram – Purchase Item



In this diagram if items have been purchased it will automatically do this use case. This use case involves storing transaction details for example the items have been purchased and the employee who served the customer and then will remove the stock of the product that has been purchased and if there isn't any product found then the system should display message telling the employee to contact the administrator. For testing purposes I'll be using test data just to see if the system picks up the data and then puts the data into the correct places. These diagrams will be further elaborated on in more detail with a step-by-step process of going through these use cases during iteration 2.

Database Design

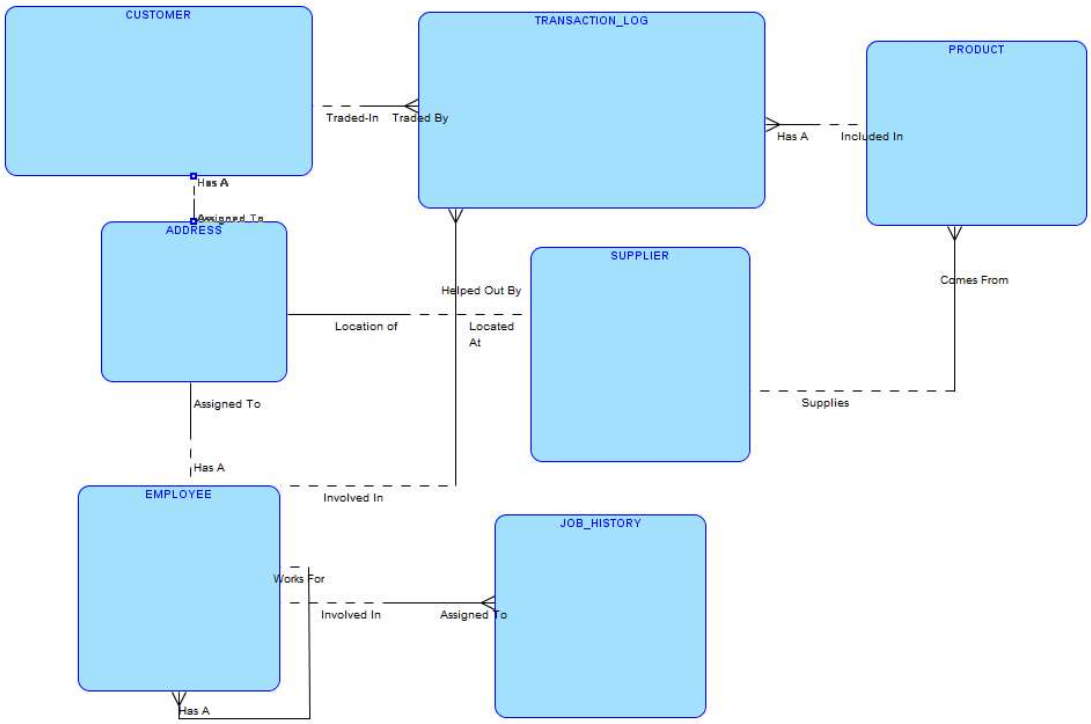
Entities & Relationship Identification

From the inception phase in the last paragraph the client wants to store details about customers, products and suppliers and employees so there will have to be Customer, Product and Supplier entities and when interviewing the client, they'd asked for a way to check transaction logs so the database will need a transaction entity. Also, during the interview with the client, I found out that both the customer and employee stores an address so they're can be an address entity.

01/03/19 – I just went and ask a question specifically about if there is any more job title besides employee and admin and they said yes but have a way to track their job history so there will be a job history entity.

With the specifics of relationships, a supplier could potentially provide more than one product, a customer might have purchased multiple products from the store before, the employee would have more than one involvement with transactions, both an employee and a customer will have one address, a product could be involved in many purchases and an employee could have different jobs during the course of working at the business.

Top-Level Entity Relationship Diagram



This diagram above shows the entities which will be the future tables in the physical model with no attributes that have their relationships. Supplier has a one-to-many to product relationship as the supplier may supply the business more than one of the same products and those products will be coming by the one supplier. The supplier will have a one-to-one address as the client will need to know where the supplier location is and is possibly optional as the supplier.

Product has a one-to-many relationship with the transaction log as a product could be bought more than once and those logs will have the purchased product linked. The employee has a one-to-many relationship with the transaction log as in the interview with the client, they'd like to track what employee done the transaction with the customer so there could be one employee who may have been involved with multiple transactions with the customer.

The employee and customer have a one-to-one relationship with the address as the customer and employee will only have one address and that address will only be linked to that employee/customer. Employee also has a one-to-many relationship with job history as the employee may have had promotions during their time at the business so the employee may have had more than one job role in the business and those jobs were assigned to the employee.

The customer has a one-to-many relationship with transaction log as the customer may trade in a game so we need to take in their details just in case if the customer has traded in a stolen game so the business can give those details to the proper authorities so a customer may trade in multiple games and those games will be linked back to that customer.

Attributes Identified

When discussing what's needed to be stored with the client, they mentioned that they need to store the customer name and address so the name will be an attribute of the customer entity/table usually layout like first, middle and then last name. With address a typical address will have the first line of an address, second/last line of address, postcode and then county which will be used for attributes of address entity/table.

When creating attributes for product it's quite obvious to store the name of the product, type of product detailing it if the products is maybe a console, game, accessory etc. From the original brief of the client they'd like to create a stock report of the products so that means they'll need to store stock of a product as well and will need to also store the price of these products.

When creating attributes for the employee entity/table its quite obvious to store the name of the employee, their address, job type and salary so these will need to be attributes. As there is going to be some kind of log-in system the employee will need to have a password attribute to hold their password. The address will need to be a foreign key of the address entity/table as all the details will be stored there instead.

When talking to the client about the employee's job types the client mentioned that they would like to have an entity/table to hold data like the employee's job name and then start & end date with end date being able to not have any data due to the job might still be worked by the employee.

For the supplier attributes there is going to have to be the name of the supplier to know who is supplying these products. Looking at the brief originally given to me by the client the client mentioned that they get products from a range of suppliers so there will be a supplier type attribute to tell where the products came from and their address can be optional due to the supplier may not be placed in a set location and the type should relate that.

With the transaction log entity/table the client talked about refunds in their brief and a way to detect if their refund is valid is to have a unique id created on the receipt given to the customer and then storing that id to the transaction log so that the id on the receipt can be then checked with the id data in the transaction log entity/table. With all of these entities/tables there will be ids to have ways to uniquely distinguish which data is which specifically with the transaction log when talking about linking the employee name with the transaction as spoken about in the interview with the client.

So, to clearly recap:[5]

Customer

- ID - PK

[5] PK – Primary Key, FK – Foreign Key, M – Mandatory, N/M – Non-Mandatory

- First Name – M
- Last Name - M
- Address – FK

Employee

- ID - PK
- First Name – M
- Last Name – M
- Password - M
- Salary - M
- Address - FK
- Job History - FK

Supplier

- ID - PK
- Name – M
- Type - M
- Address – FK N/M

Product

- ID - PK
- Name - M
- Type - M
- Price - M
- Stock - M

Job History

- ID - PK
- Type - M
- Start Date - M
- End Date – N/M

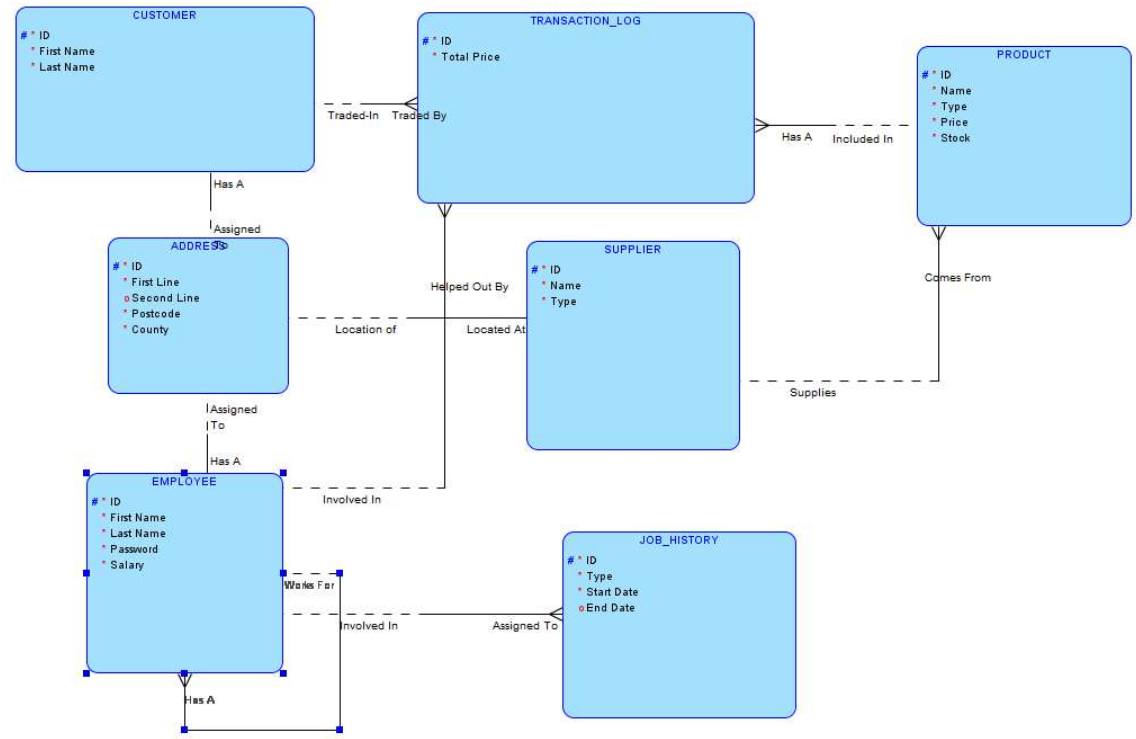
Address

- ID - PK
- First Line - M
- Second Line – N/M
- Postcode - M
- County - M

Transaction Log

- ID - M
- Product ID - FK
- Employee - FK
- Customer - FK
- Total Price – M

ERD With Attributes



In this diagram I have assigned the attributes that I identified above to the correct entities with assignments like if it is mandatory or not and if it is a primary key. With all of the foreign key attributes they don't show up on the diagram itself but they're there in the background and just to prove this is the transaction log entity properties:

	Name	Data type
1	ID	Unknown
2	Total Price	Unknown
3	ID1	Unknown
4	ID2	Unknown
5	ID3	Unknown

The ID's 1, 2 and 3 are the IDs from customer, employee and product and they'll have the tables name next to the ID when developed into the physical model.

Data Type Identification

For the data types of the attributes I will be choosing the relevant data types as seen below:

Customer

- ID – NUMBER – Size - 2
- First Name – VARCHAR2 – Size 20
- Last Name – VARCHAR2 – Size 20

- Address – NUMBER – Size 2

Employee

- ID – NUMBER – Size 2
- First Name – VARCHAR2 – Size 20
- Last Name – VARCHAR2 – Size 20
- Password – VARCHAR2 – Size 20
- Salary – NUMBER – Size 2 to two decimal spaces – 00.00
- Address – NUMBER – Size 2
- Job History – NUMBER – Size 2

Supplier

- ID – NUMBER – Size 2
- Name – VARCHAR2 – Size 15
- Type – VARCHAR2 – Size 10
- Address – NUMBER – Size 2

Product

- ID – NUMBER – Size 4
- Name – VARCHAR2 – Size 30
- Type – VARCHAR2 – Size 11
- Price – NUMBER – Size 4 with 2 decimal spaces – 0000.00
- Stock – NUMBER – Size 2

Job History

- ID – NUMBER – Size 2
- Type – VARCHAR2 – Size 15
- Start Date - DATE
- End Date – DATE

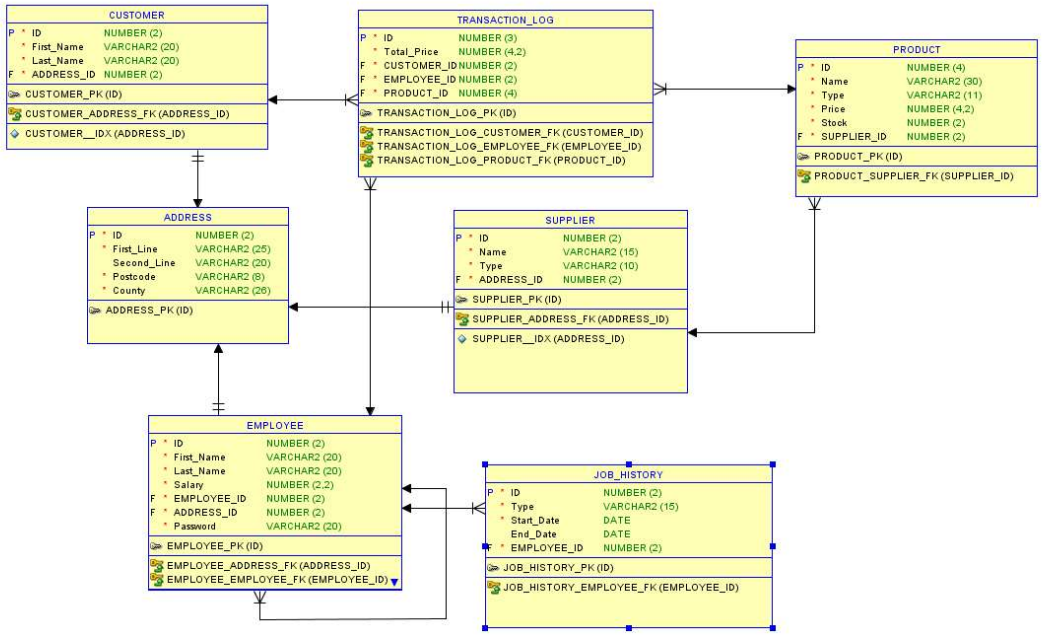
Address

- ID – NUMBER – Size 2
- First Line – VARCHAR2 – Size 25
- Second Line – VARCHAR2 – Size 20
- Postcode – VARCHAR2 - 8
- County - VARCHAR2 - 26

Transaction Log

- ID – NUMBER – Size 3
- Product ID – NUMBER – Size 4
- Employee – NUMBER – Size 2
- Customer – NUMBER – Size 2
- Total Price – NUMBER – Size 4 with two decimal spaces – 0000.00

Physical Model



This physical model came from the ERD that I created and then engineered using Data Modeler to create the diagram above which shows the tables, the attributes and the data types assigned to them and the relationships between the different tables showing the primary key and then the foreign key and where the foreign key links to.

Using this diagram, I can then further turn this physical model into SQL code and then use this code to create the tables I need in XAMPP. Before I show the SQL code, I have to mention first that because I'm using XAMPP which uses MySQL and not Oracle SQL or Microsoft SQL Server versions of SQL, I needed to convert Oracle SQL into MySQL using this website - (Sqllines, n.d.). Here is the MySQL code down below:

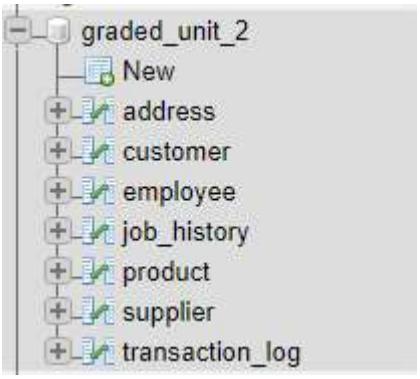
MySQL Code

```
CREATE TABLE ADDRESS
  (
    ID          TINYINT  NOT NULL ,
    First_Line  VARCHAR (25) NOT NULL ,
    Second_Line VARCHAR (20) ,
    Postcode    VARCHAR (8) NOT NULL ,
    County      VARCHAR (26) NOT NULL
  ) ;
ALTER TABLE ADDRESS ADD CONSTRAINT ADDRESS_PK PRIMARY KEY ( ID ) ;


CREATE TABLE CUSTOMER
  (
    ID         TINYINT  NOT NULL ,
    First_Name VARCHAR (20) NOT NULL ,
    Last_Name  VARCHAR (20) NOT NULL ,
    ADDRESS_ID TINYINT  NOT NULL
  ) ;
CREATE UNIQUE INDEX CUSTOMER__IDX ON CUSTOMER
  (
    ADDRESS_ID ASC
  )
  ;
ALTER TABLE CUSTOMER ADD CONSTRAINT CUSTOMER_PK PRIMARY KEY ( ID ) ;


CREATE TABLE EMPLOYEE
  (
    ID          TINYINT  NOT NULL ,
    First_Name  VARCHAR (20) NOT NULL ,
    Last_Name   VARCHAR (20) NOT NULL ,
    Salary      DECIMAL (2,2) NOT NULL ,
    EMPLOYEE_ID TINYINT  NOT NULL ,
    ADDRESS_ID  TINYINT  NOT NULL ,
    Password    VARCHAR (20) NOT NULL
  ) ;
CREATE UNIQUE INDEX EMPLOYEE__IDX ON EMPLOYEE
  (
    ADDRESS_ID ASC
  )
  ;
ALTER TABLE EMPLOYEE ADD CONSTRAINT EMPLOYEE_PK PRIMARY KEY ( ID ) ;
```

```
CREATE TABLE JOB_HISTORY
  (
    ID          TINYINT  NOT NULL ,
    Type        VARCHAR (15) NOT NULL ,
    Start_Date  DATETIME NOT NULL ,
    End_Date    DATETIME ,
    EMPLOYEE_ID TINYINT  NOT NULL
  )
  ;
ALTER TABLE JOB_HISTORY ADD CONSTRAINT JOB_HISTORY_PK PRIMARY KEY ( ID ) ;


CREATE TABLE PRODUCT
  (
    ID          SMALLINT  NOT NULL ,
    Name        VARCHAR (30) NOT NULL ,
    Type        VARCHAR (11) NOT NULL ,
    Price       DECIMAL (4,2) NOT NULL ,
    Stock       TINYINT  NOT NULL ,
    SUPPLIER_ID TINYINT  NOT NULL
  )
  ;
ALTER TABLE PRODUCT ADD CONSTRAINT PRODUCT_PK PRIMARY KEY ( ID ) ;


CREATE TABLE SUPPLIER
  (
    ID         TINYINT  NOT NULL ,
    Name       VARCHAR (15) NOT NULL ,
    Type       VARCHAR (10) NOT NULL ,
    ADDRESS_ID TINYINT  NOT NULL
  )
  ;
CREATE UNIQUE INDEX SUPPLIER__IDX ON SUPPLIER
  (
    ADDRESS_ID ASC
  )
  ;
ALTER TABLE SUPPLIER ADD CONSTRAINT SUPPLIER_PK PRIMARY KEY ( ID ) ;


CREATE TABLE TRANSACTION_LOG
  (
    ID          SMALLINT  NOT NULL ,
    Total_Price DECIMAL (4,2) NOT NULL ,
    CUSTOMER_ID TINYINT  NOT NULL ,
    EMPLOYEE_ID TINYINT  NOT NULL ,
    PRODUCT_ID  SMALLINT  NOT NULL
  )
  ;
ALTER TABLE TRANSACTION_LOG ADD CONSTRAINT TRANSACTION_LOG_PK PRIMARY KEY ( ID ) ;
```

```
ALTER TABLE CUSTOMER ADD CONSTRAINT CUSTOMER_ADDRESS_FK FOREIGN KEY ( ADDRESS_ID ) REFERENCES ADDRESS ( ID ) ;

ALTER TABLE EMPLOYEE ADD CONSTRAINT EMPLOYEE_ADDRESS_FK FOREIGN KEY ( ADDRESS_ID ) REFERENCES ADDRESS ( ID ) ;

ALTER TABLE EMPLOYEE ADD CONSTRAINT EMPLOYEE_EMPLOYEE_FK FOREIGN KEY ( EMPLOYEE_ID ) REFERENCES EMPLOYEE ( ID ) ;

ALTER TABLE JOB_HISTORY ADD CONSTRAINT JOB_HISTORY_EMPLOYEE_FK FOREIGN KEY ( EMPLOYEE_ID ) REFERENCES EMPLOYEE ( ID ) ;

ALTER TABLE PRODUCT ADD CONSTRAINT PRODUCT_SUPPLIER_FK FOREIGN KEY ( SUPPLIER_ID ) REFERENCES SUPPLIER ( ID ) ;

ALTER TABLE SUPPLIER ADD CONSTRAINT SUPPLIER_ADDRESS_FK FOREIGN KEY ( ADDRESS_ID ) REFERENCES ADDRESS ( ID ) ;

ALTER TABLE TRANSACTION_LOG ADD CONSTRAINT TRANSACTION_LOG_CUSTOMER_FK FOREIGN KEY ( CUSTOMER_ID ) REFERENCES CUSTOMER ( ID ) ;

ALTER TABLE TRANSACTION_LOG ADD CONSTRAINT TRANSACTION_LOG_EMPLOYEE_FK FOREIGN KEY ( EMPLOYEE_ID ) REFERENCES EMPLOYEE ( ID ) ;

ALTER TABLE TRANSACTION_LOG ADD CONSTRAINT TRANSACTION_LOG_PRODUCT_FK FOREIGN KEY ( PRODUCT_ID ) REFERENCES PRODUCT ( ID ) ;
```

I then imported this code into phpMyAdmin with XAMPP after creating a database where these tables could be kept in which went successfully without any problems at all and here is proof of the creation of the tables:



Later on, in iteration 3 I will be integrating the data from these tables into the Java application for the reports and the transaction logs, displaying the employee, customer and supplier details, searching for a product and adding stock.

View Model

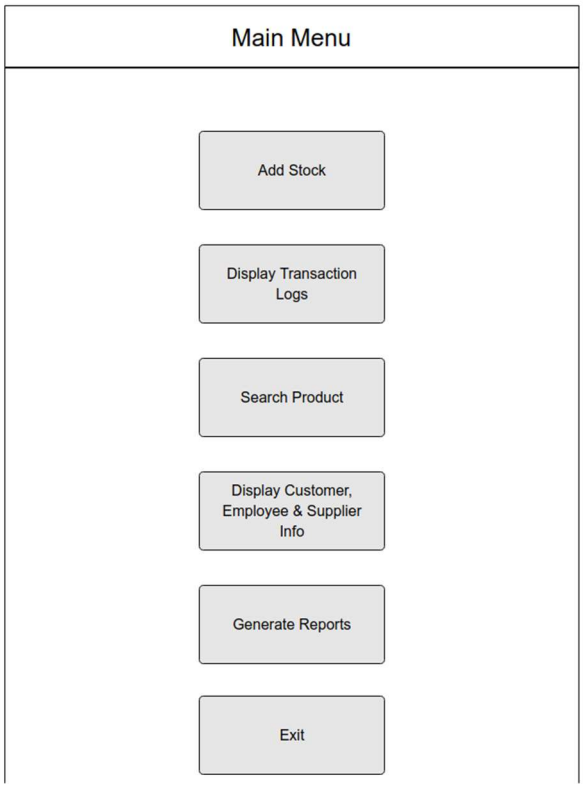
Wireframes

For the GUI I'll need to create wireframes to represent the GUI in a more basic form just to present what the GUI would potentially look like during the implementation stage of the prototype and from what I mentioned above in the techniques part of the design and the resources used in the inception phase I'm using Moqups.

The reason why I'm using Moqups is for both convenience and knowledgeability as Moqups is a web-based application that can be used on many different computers, all I need is to log-into my account on said computer and I have experience using Moqups already so there's no need to go look out for another application to use but just in case Moqups is down or the internet goes down the alternative I'll use is Microsoft Visio which includes wireframe creations.

Before I create the wireframes I done some research on the theory of wireframes and I'm following some advice and tips from this website - (GiveGoodUX, n.d.) – to make the GUI feel more easier to use and to be on point with how easily reachable all the use cases above will be.

1. Main Menu

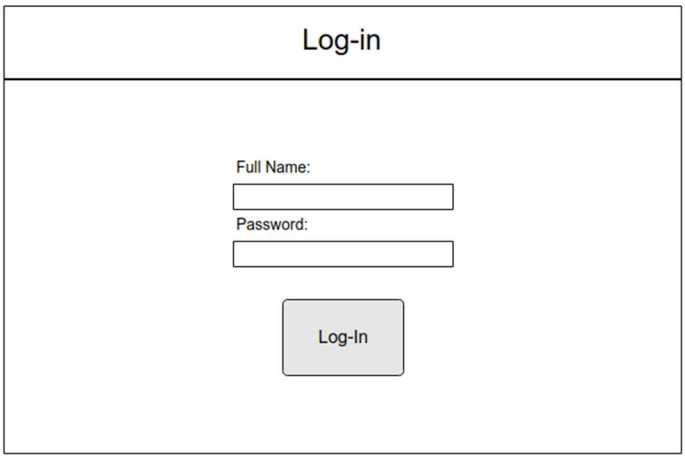


With this and future wireframes, I've made the buttons bigger than usual for people who may have difficulties navigating through applications or general software and also when an employee/admin uses one of the buttons a window will pop-up over the main menu related to the button clicked. I've laid out the window to be vertical because it can then be easily seen on various computer screens due to it not taking up too much space. This above is an example of the admin main menu however the typical employee main menu will not have the display customer, employee & supplier info and generate report at all.

2. Message Box

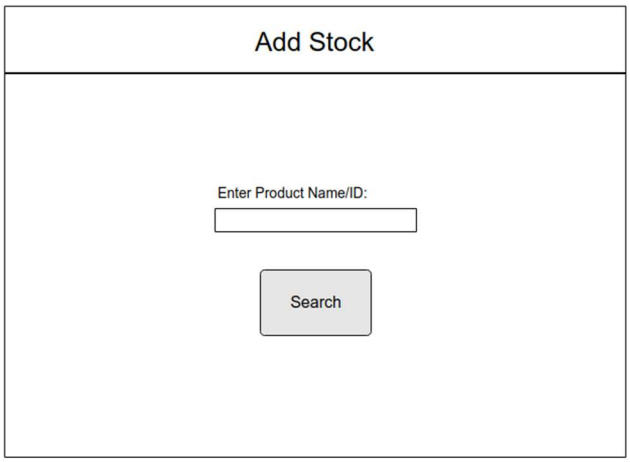
This message box in the current wireframe is for when the admin/employee adds stock successfully the system will display a message telling them so. However, this message box will have variants for different situations that will be mentioned in further wireframe explanations down below.

3. Log-In



This wireframe is for the log-in screen which will be the first screen that would be seen by both the admin and employee. The admin/employee will enter their full name into the first text box and then they'll enter their password in the second text box and then click the log-in button down below. If successful, the employee/admin will go to wireframe 1 to their respected variant and if not then a variant of wireframe 2 will display mentioning that they've entered the wrong details with a back button to go back to the log-in screen.

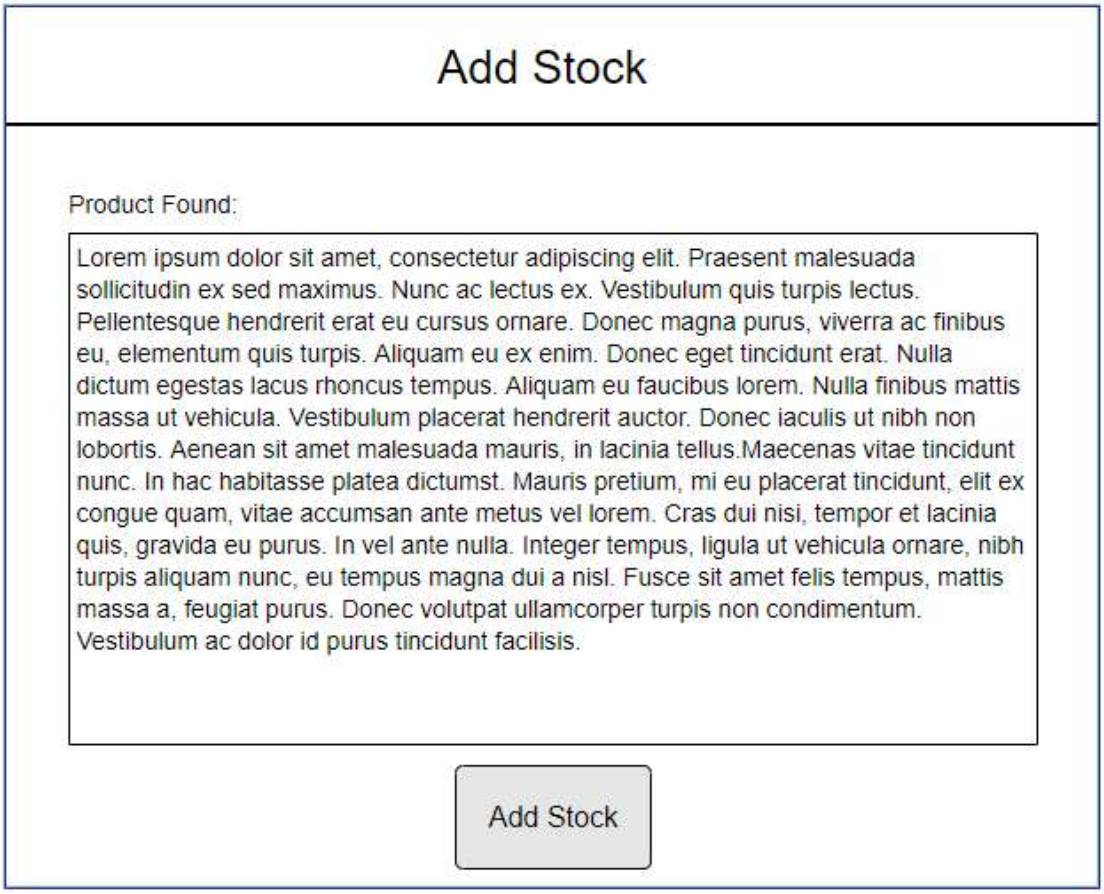
4. Search Box



What this wireframe represents is the add stock button in the main menu and this is the window that will pop-up asking the employee/admin to click and type in either the product id or name into the text box and then clicking on the search button will switch the display to the result.

This wireframe also has another variant for both searching product use case and the adding the actual stock, but the only difference is that the add stock title will change just to search product for search and will change the enter product name to enter number of stock and the message in the button will say add stock.

5. Product Found

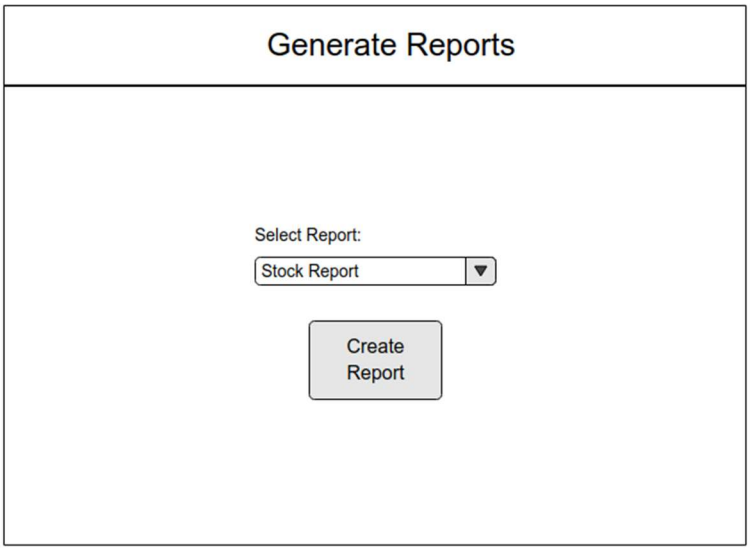


In this wireframe what displays is that if there is a successful search the box will be filled with the information of the product and then the employee/admin can then click the add stock button to then proceed to add the amount of stock they need to add.

However, if there isn't any data on that product then there will be a variant of wireframe 2 that will display a message saying no product found with a back button to go back and search. With this wireframe 5 there will be another variant where the add stock button won't appear and the add stock title would be replaced with the search product title and also with generate report when the admin selects the report they want to see and what will change is that the add stock will remove and the add stock title will be replaced with the name of the report to be its title and will have a scroll wheel as there might be a lot of data outputted by the report and in the text box will be the report.

This also the same with the check details of either customer, employee or the supplier which will display it on the text box with a scroll wheel due to it may have a lot of data to display. And again, there will be another variant with the transaction logs as this screen will pop-up as soon as the employee/admin clicks on the display transaction logs. The difference is that the title will be transaction logs, there is no add stock button and, in the text, box will be information about the transactions.

6. Choice Box Menu

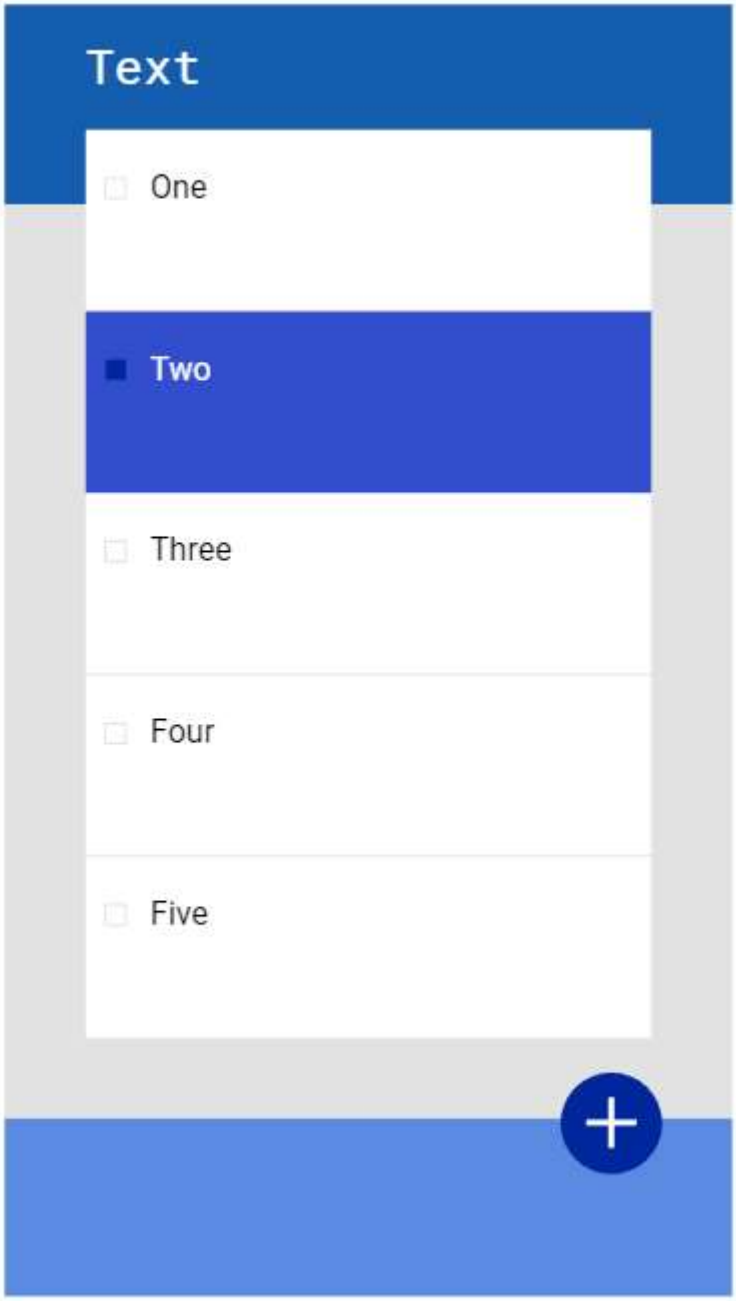


In this wireframe it will ask the admin to click the choice box to choose which report they want to display and then click on the box which will display a variant of wireframe 5 which I explained what will display. This will have a variant for the check details of customer, employee and supplier which will have these choices in the choice box and the title will change to be check details and the text in the button will change to check details.

Style Guide

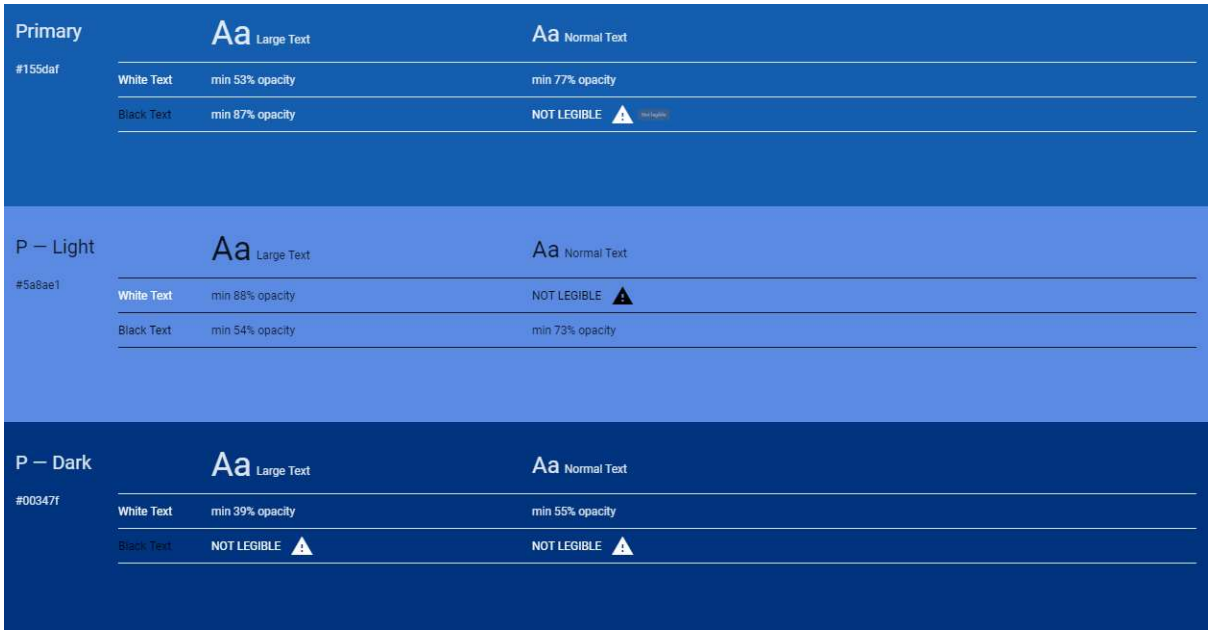
Before I create the style guide for what colours I'll use during the implementation of the prototype I did a bit of research beforehand to understand what colours I should watch out for and what colours to use and what fonts are best suited for dyslexic people etc. Firstly, this website - (Usabilla, n.d.) – gave me quite a bit of tips and helpful advice on what to do when thinking about creating a style guide particularly with the colour blind and what colour combinations are bad and suggestions on what to use.

I also did some research on what font is best when discussing about dyslexic people and according to this website - (BDatech, n.d.) – is some Microsoft fonts like Arial, Verdana, Calibri and some open-source fonts like Lexia Readable, Dyslexia etc. What I'll use to create the colour scheme that will be used for the prototype is a website called material tools colour tool which is mainly used for android application design, but the tool is quite good for applications like this given details on what colour complements another and what colour is font is better for the other etc - (Material, n.d.)



Through this wireframe given to be by the colour tool this is what the colours will be throughout the entire prototype as with this colour scheme I used this website - (Toptal, n.d.) – which allowed me to take the weblink of the colour scheme and let me see the colour blind versions of the colours and for both the protanopia and deuteranopia colour blindness they were pretty much the same and for tritanopia it looked like a greener colour was coming out but looked completely fine. Here is the colour scheme with colour names


Primary
#155daf
P
RESET
P — Light
#5a8ae1
P — Dark
#00347f
Secondary
#324ece
S
S — Light
#717aff
S — Dark
#00279c
Text on P
#ffffff
T
Text on S
#ffffff
T

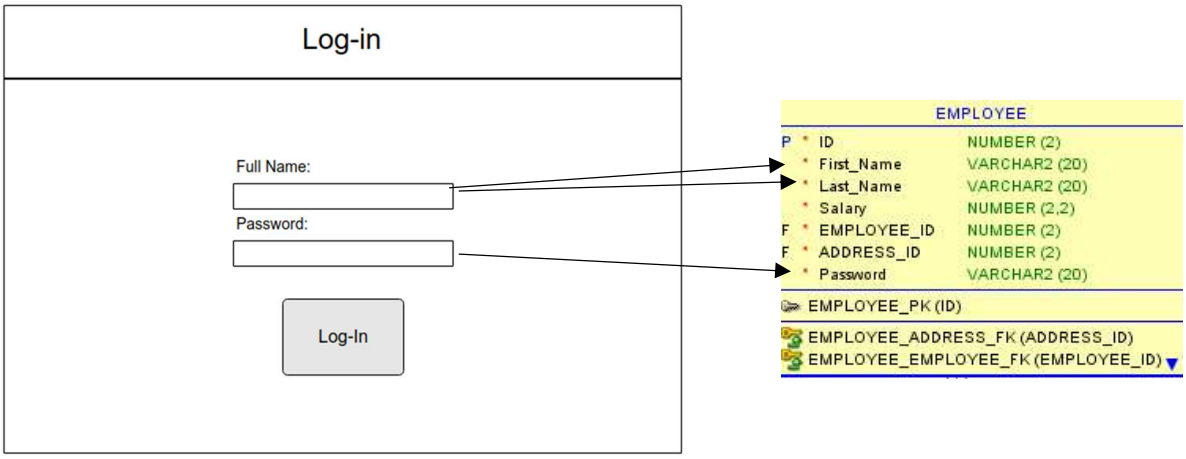

Primary
#155daf
Aa Large Text
Aa Normal Text
White Text
min 53% opacity
min 77% opacity
Black Text
min 87% opacity
NOT LEGIBLE
P — Light
#5a8ae1
Aa Large Text
Aa Normal Text
White Text
min 88% opacity
NOT LEGIBLE
Black Text
min 54% opacity
min 73% opacity
P — Dark
#00347f
Aa Large Text
Aa Normal Text
White Text
min 39% opacity
min 55% opacity
Black Text
NOT LEGIBLE
NOT LEGIBLE

In these pictures the tool told me to use white and black text at different times, for the primary normal version it tells me the best way to read text is to use black text, for a light version I'd use the white text as it is easier to read, when using the dark version use white text as the black text is not legible. With the secondary colours the normal secondary told me to best use black text instead of white text, the light version uses a black text as using white text for normal text is not legible and for the dark version to use white text like the primary colours.

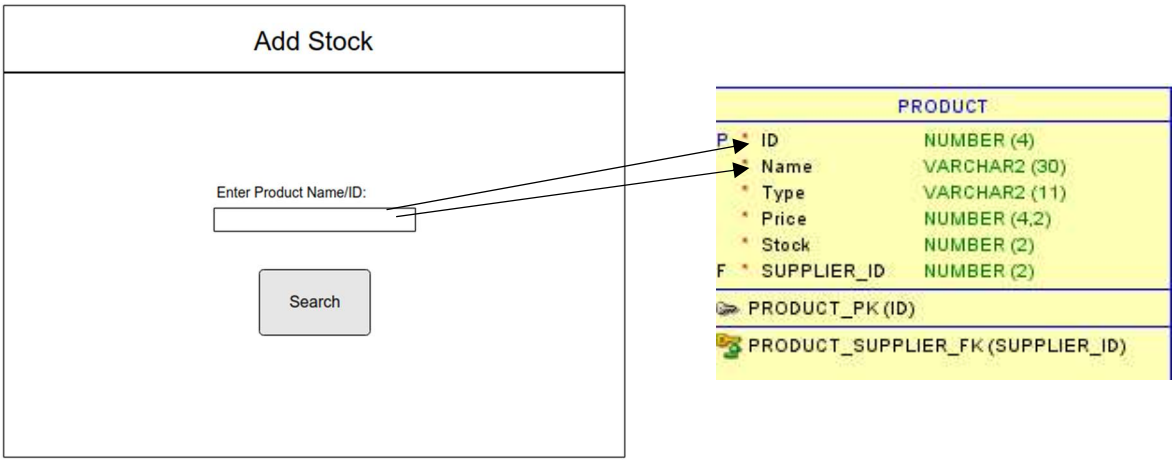
For the font I'm going to see if I can use Arial as it was described in the website that Verdana was one of the go-to fonts as its much bigger making it much easier to read the font and if I aren't available to use Verdana I'll use the open source Dyslexia instead due to maybe not being able to configure Verdana with the IDE I'm using.

Data Binding Model

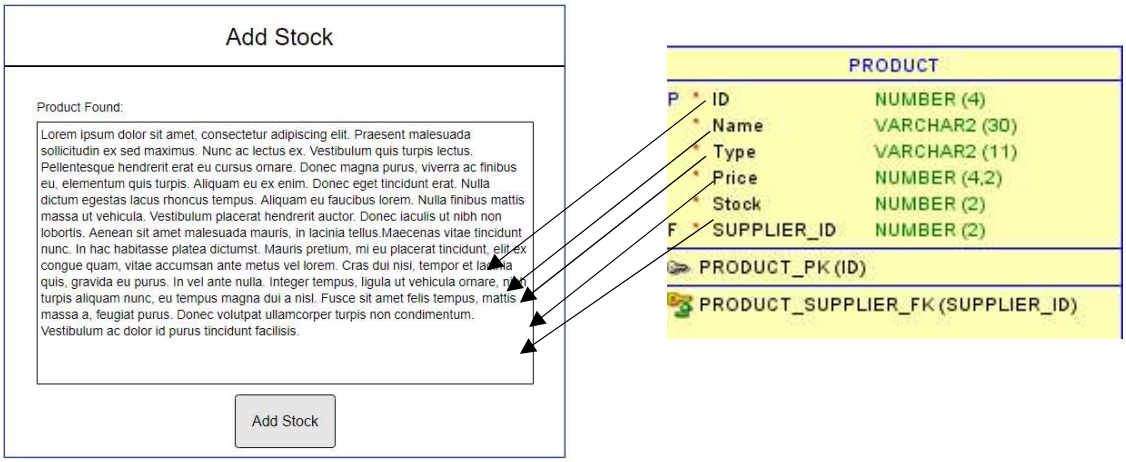
This is the reason I did the database back-end first is to then be able to show where the data from the backend will appear in the front-end of the prototype so what I'm going to do is using wireframe 3, 4 and 5 and the physical model of the database I'll use arrows to point out what specific attributes will be used for that particular part.



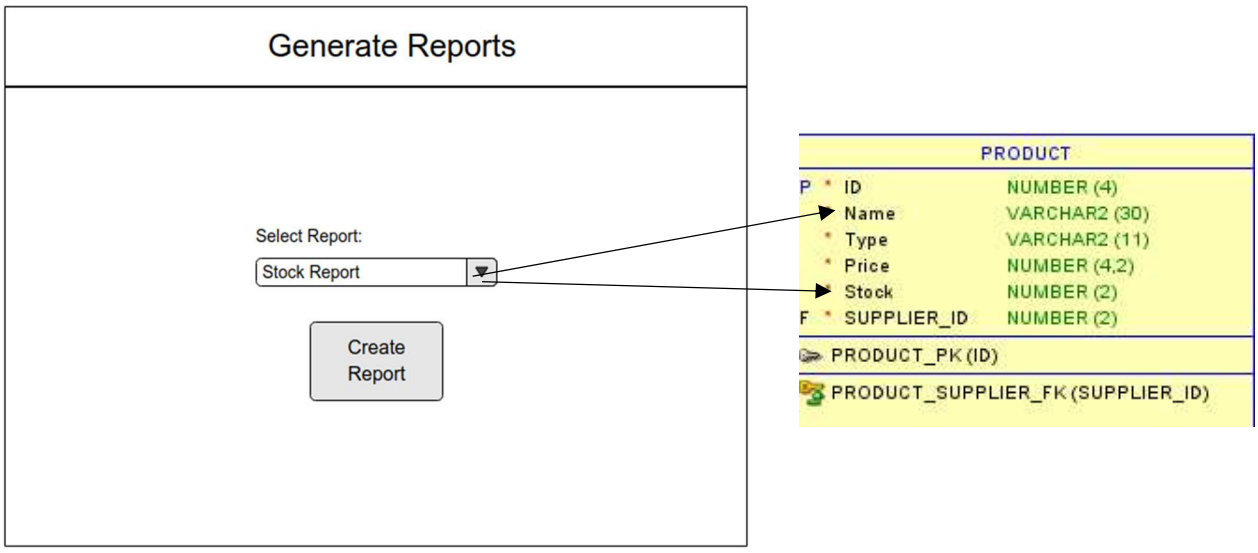
With the employee/admin enters their details and then presses the log-in button the data from the employee table in the database will be taken out specifically the first & last name and the password to then be checked to see if the name and password given matches up with any name and password in the database.



When the employee/admin enters a product name or id and then clicks the search button the data from the product table will be pulled out specifically the name of the product or the id of the product and will then check to see if there is any product with the id or name given by the employee or admin and if so will then proceed to take the info about that product out to produce the wireframe 5.



What this shows is that when a product has been found that is the specific product the employee/admin has searched for then the products information will then be pulled out of the table and into this text box detailing what the product is, how much it costs etc. However, when talking about this wireframe I stated that this wireframe will also be used for displaying the reports so this will be the same for displaying stock, popularity and most popular product reports. I also mentioned about the displaying employee, customer and supplier information so it will be the same as above with the data from the table be it the employees name, salary, job title or suppliers name, type which will be displayed into the text box represented in the wireframe or specific information about the transactions which will take from multiple tables like the employee, product, customer if trade-in and the transaction log table.



For this wireframe the admin selects what report they'd like to see which in this case is a stock report and then clicks on create report which will then get the data from the table

product specifically the name and the stock of the product and will display the information in the text box of wireframe 5.

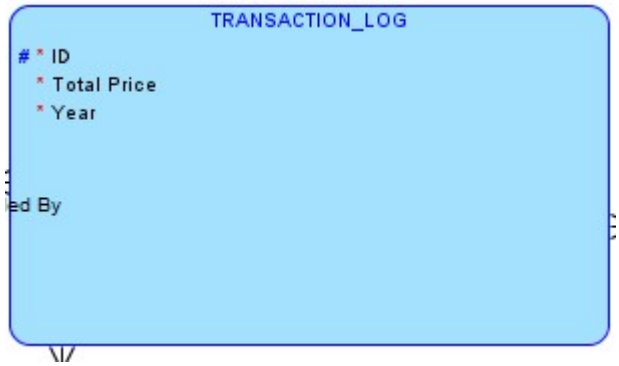
Next Iterations

For the second iteration I'm going to focus on designing the front-end Java application by creating the descriptions of the use case diagrams that I made and then make the CRC cards specifically identifying what classes I'll need to have and what they'll do, class diagrams to represent how the classes will look like and what data they'll contain and what they can do. I'll also need to make sequence and activity diagrams representing the functionality of the use cases and then finally start programming the Java application. With the third iteration I'll make the link between the back-end database with the front-end Java application by using JDBC and then create the test plans to use when I have "finished" the prototype so that I can find any errors in the code or any misplaced validation that I need to correct.

Second Iteration

Changes Before Going Ahead - 2

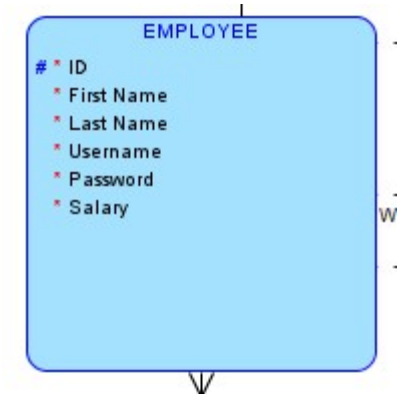
In the transaction log entity, I forgot to add a specific attribute to that entity called date of purchase to determine when that transaction took place which I then added to the ERD as seen down below:



	Name	Data type
1	ID	NUMERIC (3)
2	Total Price	NUMERIC (4, 2)
3	Year	Date
4	ID2	NUMERIC (2)
5	ID3	NUMERIC (2)
6	ID1	NUMERIC (4)

Also, in the employees table I've realised that I didn't have a username attribute. I cleared up on what the username's format will be like with the client and they told it me it should be the first initial of the employees first name, the full last name and their id as the numbers

after the last name so for example lee glen and id is 25, the username should look like this – Lglen25 which I then changed in the design as seen down below:



	Name	Data type
1	ID	NUMERIC (2)
2	First Name	VARCHAR (20)
3	Last Name	VARCHAR (20)
4	Username	VARCHAR (23)
5	Password	VARCHAR (20)
6	Salary	NUMERIC (2, 2)
7	ID1	NUMERIC (2)
8	ID2	NUMERIC (2)

Reasoning behind the length is due to the first character of the employee's first name adding the fact that their id could have two digits plus their last name with some leeway for people with longer last names will be 23.

I then re-engineered the ERD with these changes and then re-created the database in the same way I did it from page 14 to midway of page 17.

I also forgot to mention when talking about the search box wireframe, that will be first thing you'll see when clicking on the display transaction logs as the title will be display transaction logs and then the label above the search bar will say search by a specific date and then it will lead into the products found wireframe.

Use Case Descriptions

As I mentioned back in the first iteration, I mentioned that I will need to create the descriptions of the use cases that where specified in the diagram. These descriptions will be used to determine what I believe what that use case will pretty much do when the application is being made plus extension on an unexpected response as seen below with the log-in and then the extension being that the details are wrong.

Admin/Employee

Log-in

Description: Either the employee or the admin wants to use some of the features of the application but will need to log-in to the application first. The employee/admin will use their

ID as the username and then their password to access the application to use the features that they're allowed to use.

Actor: Admin, Employee

Trigger: The Admin/Employee starts the application

Pre-conditions: The application isn't already started; the database is up and running.

Post-conditions: The application pops up a menu with the functions that is specified to them.

Normal Flow of Events:

1. Admin/Employee loads up the app
2. The application pops up a menu displaying a log-in screen.
3. The Admin/Employee will enter their username specified by a text box.
4. The Admin/Employee will enter their password specified by a text box.
5. The Admin/Employee will click on the submit button
6. A query is sent to the database to get the usernames & passwords
7. The app will then do a comparison between the username & password given with the usernames & passwords stored in the database.
8. The username & password checks out.
9. A menu displaying functions that's specified to the admin/employee displays.

Extension Flow

8. Details given are incorrect and doesn't match
   8.1 Username is wrong
      8.1.1 A prompt will tell the admin/employee that either username is wrong
      8.1.2 Step 2 will happen again
      8.1.3 The admin/employee will re-enter username with correct username
      8.1.4 The admin/employee will re-enter password.
      8.1.5 Step 5 onwards will happen again until step 7
      8.1.6 If username given is correct in step 8 then step 9 will happen
      8.1.7 If the username is not correct, then step 8.1.1 and onwards will happen again until the admin/employee enters correct username.
   8.2 Password is wrong – Is the same as 8.1, just the username is changed with the password.

Search Product

Description: Once the Admin/Employee is logged-into the application the admin/employee will then click on the search product button which will pop-up a menu with a text box and a button to allow them to search a product by name or id. The admin/employee will then enter either the product name or the id and then the admin/employee will hit the submit button. The product will then be returned from the database and displayed to the admin/employee.

Actor: Admin/Employee

Trigger: The Admin/Employee clicks on the search product button

Pre-Conditions: The Admin/Employee is logged in.

Post-Conditions: The Product Details Are Displayed.

Normal Flow of Events:

1. Use Case – Log-In happens
2. The Admin/Employee clicks on the search product button
3. The application pops up a screen with a text box and a button
4. The Admin/Employee enters either a product name or ID they want info on.
5. The Admin/Employee then clicks on the button
6. All the names and IDs of every product from the database is transferred to the application
7. A search is then made using the product name or ID is given with the data from the database.
8. Data matches with the data of the product.
9. A new screen will pop up with a big text box which will contain the details of the product.

Exception Flow

7. Product Not Found
   7.1 A prompt will appear which will inform the Admin/Employee that the product hasn't been found.
   7.2 The Admin/Employee will then re-enter either a different product or possibly re-spelled the product name due to maybe misspelling might happen.
   7.3 Step 5 onward until step 7
   7.4 If the data matches, then step 9 will happen
   7.5 Else it will return back to doing 7.1 until it reaches step 7 again.

Display Transaction Logs

Description: Once the Admin/Employee logs-into the application the Admin/Employee will click on the button related to displaying transaction logs which will lead them to a screen where they'll be able to search by a date. When the Admin/Employee searches by a date another screen will pop-up displaying the transaction logs of that date.

Actor: Admin/Employee

Trigger: The Admin/Employee clicks on display transaction log

Pre-condition: The Admin/Employee is logged in.

Post-Conditions: The transactions are displayed.

Normal Flow of Events:

1. Use Case – Log In happens
2. The Admin/Employee clicks on display transaction log button
3. The application will pop up a screen with a text box and a button
4. The Admin/Employee will enter the date of the purchase inside the text box
5. The Admin/Employee clicks on the button
6. The transaction logs from the database will then be stored inside the application related to the date entered by the Admin/Employee.
7. A new screen will pop up with a big text box with all the transaction details.

Exception Flow

6. No Product Has Been Purchased
   6.1 A prompt will pop up saying that there have been no purchases made.
   6.2 The application would be redirected back to the main menu.

Add Stock

Description: The Admin/Employee will be able to add stock to a product by searching its id or name and then adding a specific amount of stock which will be added to the product with a date and the name of the Admin/Employee that added the stock will be stored.

Actor: Admin/Employee

Trigger: The Admin/Employee clicks on add stock

Pre-condition The Admin/Employee is logged-in

Post-Conditions: The stock has been added and the name and date of stock being added has been stored.

Normal flow of events:

1. Use Case – Log In happens
2. The Admin/Employee clicks on add stock button
3. The application will pop up a screen with a text box and a button
4. The Admin/Employee will enter the name or id of the product inside the text box
5. The Admin/Employee clicks on the button
6. All the names and IDs of every product from the database is transferred to the application
7. A search is then made using the product name or ID is given with the data from the database.
8. Data matches with the data of the product.
9. A new screen will pop up with a text box and a button.
10. The Admin/Employee will then enter the amount of stock being added into the text box
11. The Admin/Employee will then click on the button
12. The stock will then be added onto the stock that is in the product table.
13. The name of the Admin/Employee will be stored alongside the date that the stock was added.

Exception flow:

7. Product Not Found

7.6 A prompt will appear which will inform the Admin/Employee that the product hasn't been found.
7.7 The Admin/Employee will then re-enter either a different product or possibly re-spelled the product name due to maybe misspelling might happen.
7.8 Step 5 onward until step 7
7.9 If the data matches, then step 9 and onwards will happen

Else it will return back to doing 7.1 until it reaches step 7 again.

Admin

Display Employee, Customer & Supplier Info

Description: The Admin will be able to display client, employee or supplier details to get more detail either on a person, company or an employee.

Actor: Admin

Pre-Condition: The Admin has logged-in

Post-Conditions: Either employee, customer or supplier's info is displayed

Normal flow of events:

1. Use Case – Log In happens
2. The Admin clicks on display employee, customer & supplier info button
3. The application will pop up a screen with a choice box and a button
4. The admin can choose three options: employees, customers and suppliers
5. After choosing one of these options the admin will then click on the button
6. Depending on the choice the admin has chosen the information related to one of these options will be pulled from the database.
7. A new screen will pop up with a big text box with the details the admin has chosen.

Exception Flow:

6. No Information Been Stored

6.1 A prompt will appear which will inform the admin that there is no information stored related to the choice.
6.2 The application will then redirect back to the main menu.

Generate Reports

Description: The Admin will be able to choose from a multitude of reports to display from what employee has sold the most, what products are the most popular, what product hasn't got any stock etc.

Actor: Admin

Pre-condition: The Admin has logged-in

Post-conditions: A report has been displayed in the application

Normal flow of events:

1. Use Case – Log In happens
2. The Admin clicks on the generate report button
3. The application will pop up a screen with a choice box and a button
4. The Admin can choose from multiple options – Products sold, employee most sold, most popular products and stock report of the products that have no stock (For these events Products sold will be picked)
5. The Admin will click the button
6. The application will pop up a new screen with a choice box and a button
7. The Admin will choose from these options – sold by week, month or year.
8. The admin will click the button
9. A query will be sent to the database that will get the most sold product depending on what timeframe the Admin has chosen
10. The information calculated by the query will then be sent back to the application
11. A new screen will pop up with a big text box with the information inside the text box

Alternative Flow:

The Admin has chosen employee most sold:

1.-3. Is the same as the normal flow of events

4. The Admin chooses employees most sold

5. A query will be sent to the database finding how much the employees have sold

6. The information calculated by the query will then be sent back to the application

7. A new screen will pop up with a big text box with the information inside the text box

The Admin has chosen most popular products

1.-3. Is the same as the normal flow of events

4. The Admin chooses most popular products

5. A query will be sent to the database finding the most popular products of all time

6. The information calculated by the query will then be sent back to the application

7. A new screen will pop up with a big text box with the information inside the text box

The Admin has chosen stock report

1.-3. Is the same as the normal flow of events

4. The Admin chooses stock report – product no stock

5. A query will be sent to the database finding the products that have no stock
6. The information calculated by the query will then be sent back to the application
7. A new screen will pop up with a big text box with the information inside the text box

Exception Flow

6. No information found
    6.1 A prompt will appear which will inform the admin that there is no information found related to the choice.
    6.2 The application will then redirect back to the main menu.

Purchase Item

Use Case name is the same as the title.

Description: From the register after a purchase is made the application will pick up on the purchase and then remove the stock of the item being purchased and then store the transaction details.

Actor: register

Pre-condition: The customer is purchasing a product

Post-condition: The stock has been removed and transaction details have been stored.

Normal flow of events:

1. When a purchased has been made the details from what products have been purchased and what employee has dealt with the purchase comes into the application.
2. The product that has been purchased will then be checked to see if the product is in the database
3. If the product has been found, then the stock of the product will then be removed and how much will be removed will depend on how much of that product has been purchased.
4. The details will then be stored in the transaction log table in the database
5. The id of the transaction log will be used in the receipt to be used as reference for refunds.

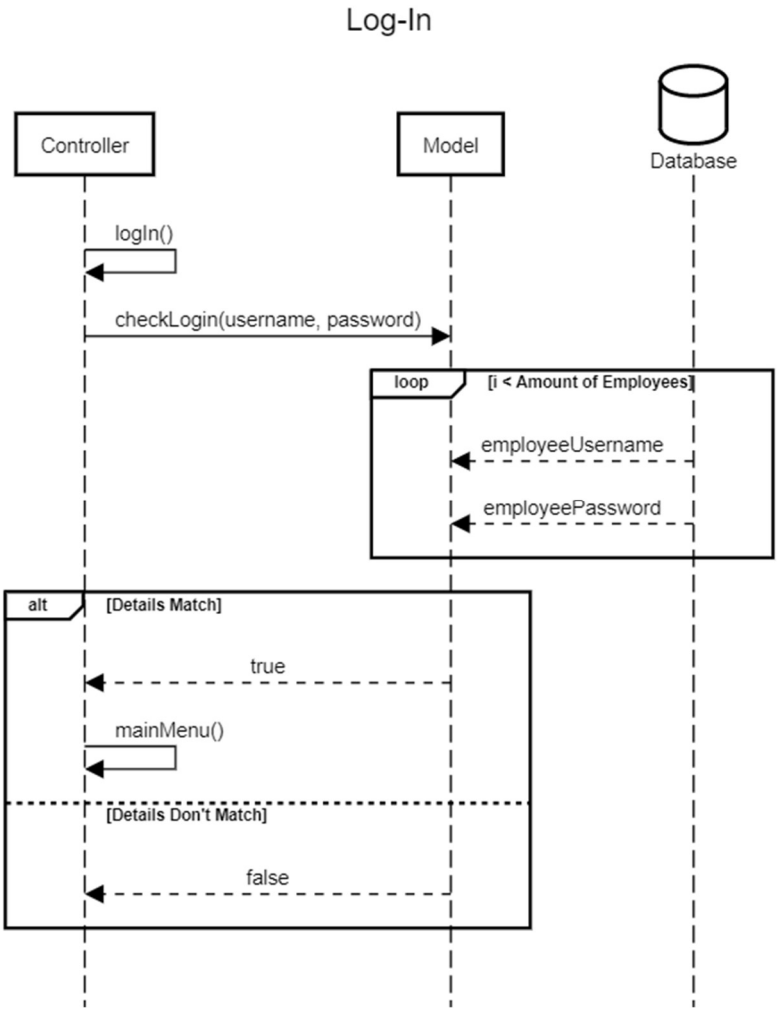
Exception Flow:

2. Product Not Found
    2.1 An alert with a button will pop-up informing the admin that a product that has been purchase hasn't been found.
    2.2 This alert will stay on screen until someone presses the button.
1. Card Declined
    1.1 If the card that customer uses to purchase the product declines, then the process in the normal flow won't happen at all.

Sequence Diagrams

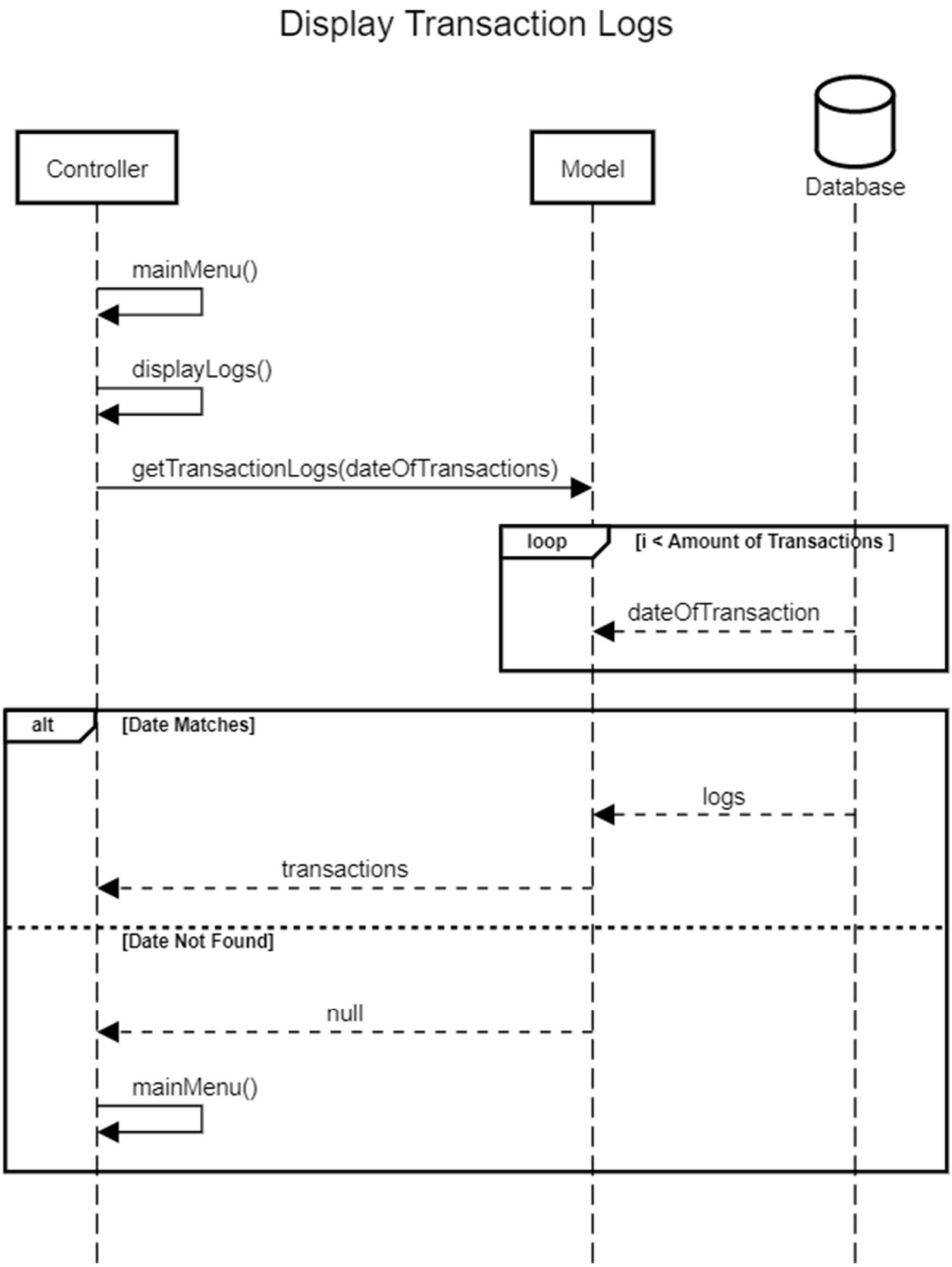
These diagrams represent how the objects will interact with each other and in basic way how the use cases would work. I have made a rough sequence diagram for each use case but I'm not entirely sure that these diagrams are correct.

Log-In



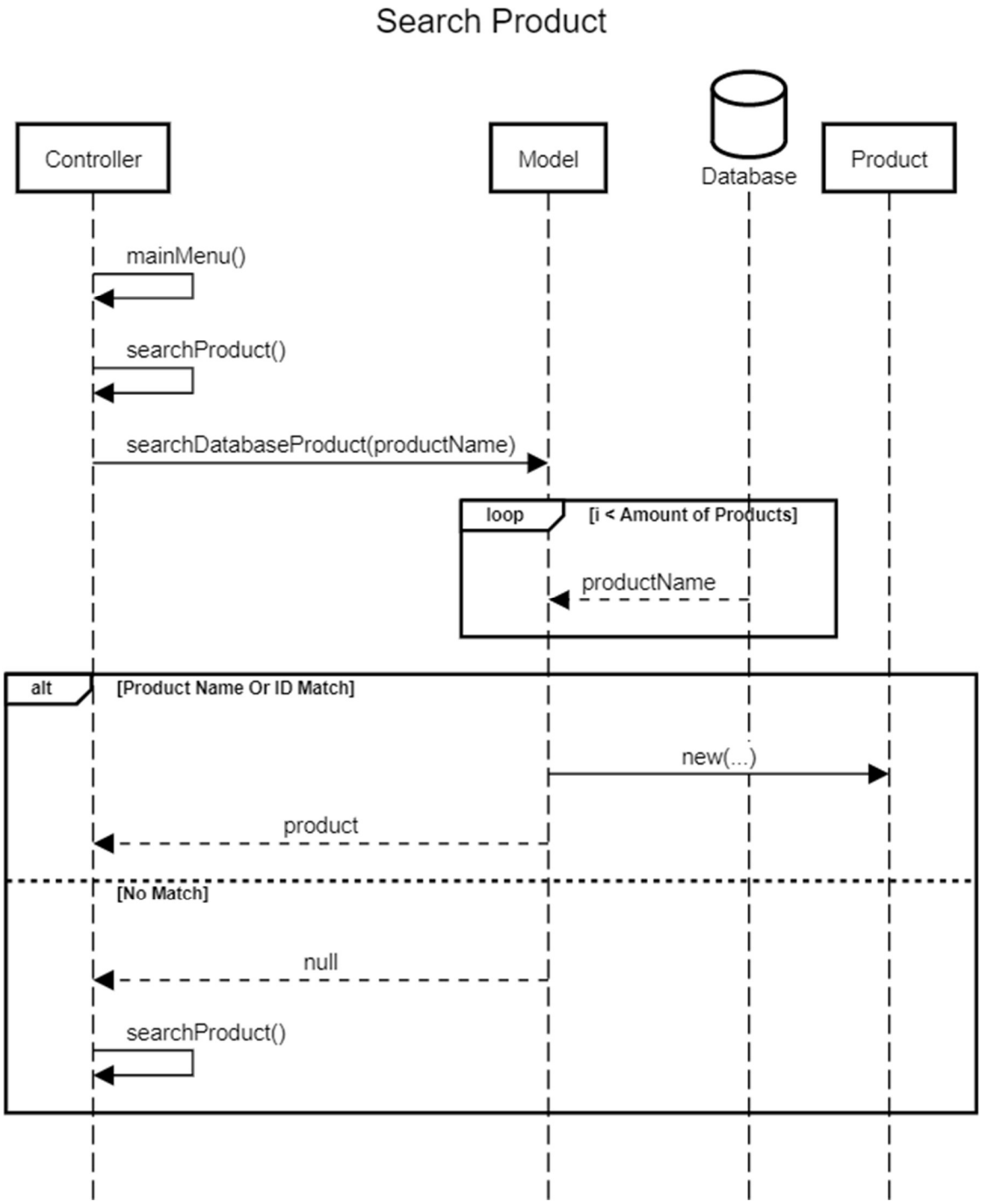
This diagram represents how the log-in will take place from what was described in the use case description of log-in but a bit rougher.

Display Transaction Logs



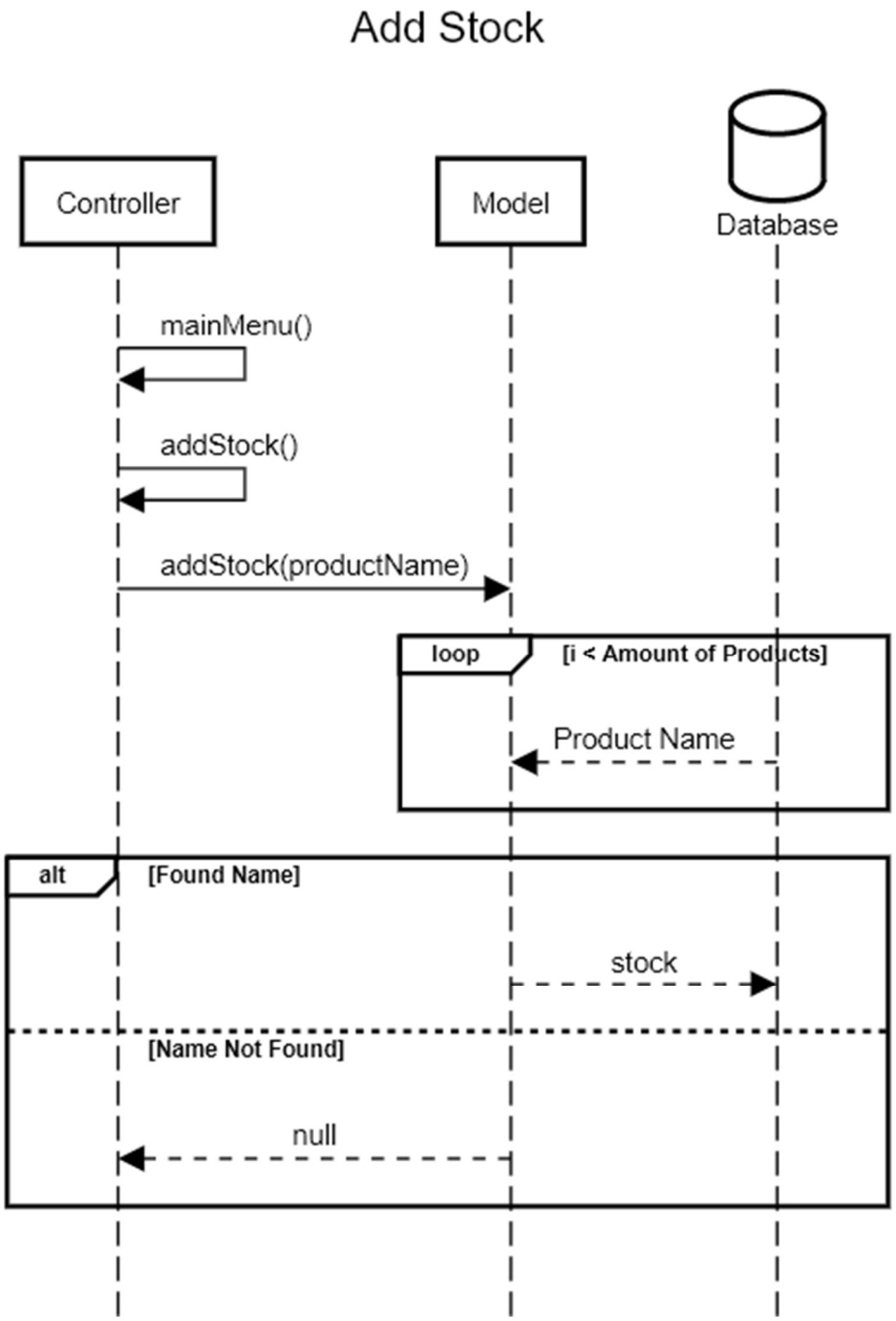
This represents how the search product is what was roughly said from the use case description related to the transaction log.

Display Transaction Logs



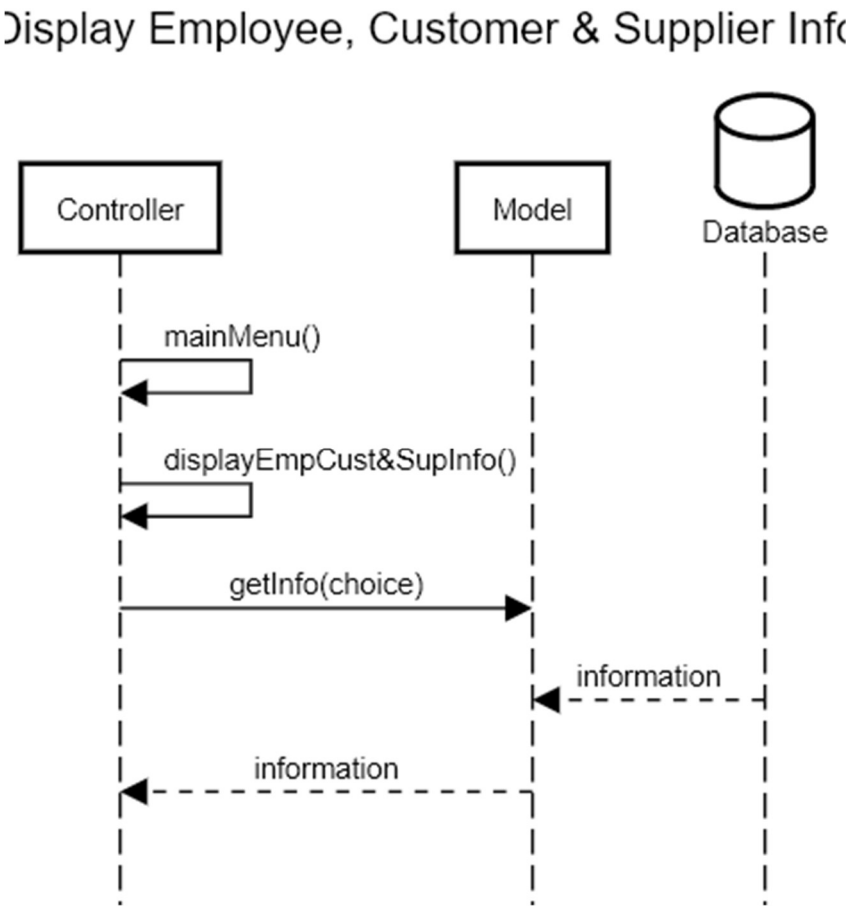
This is the sequence diagram for searching a product which was created from the use case description describing how searching a product would work out.

Add Stock



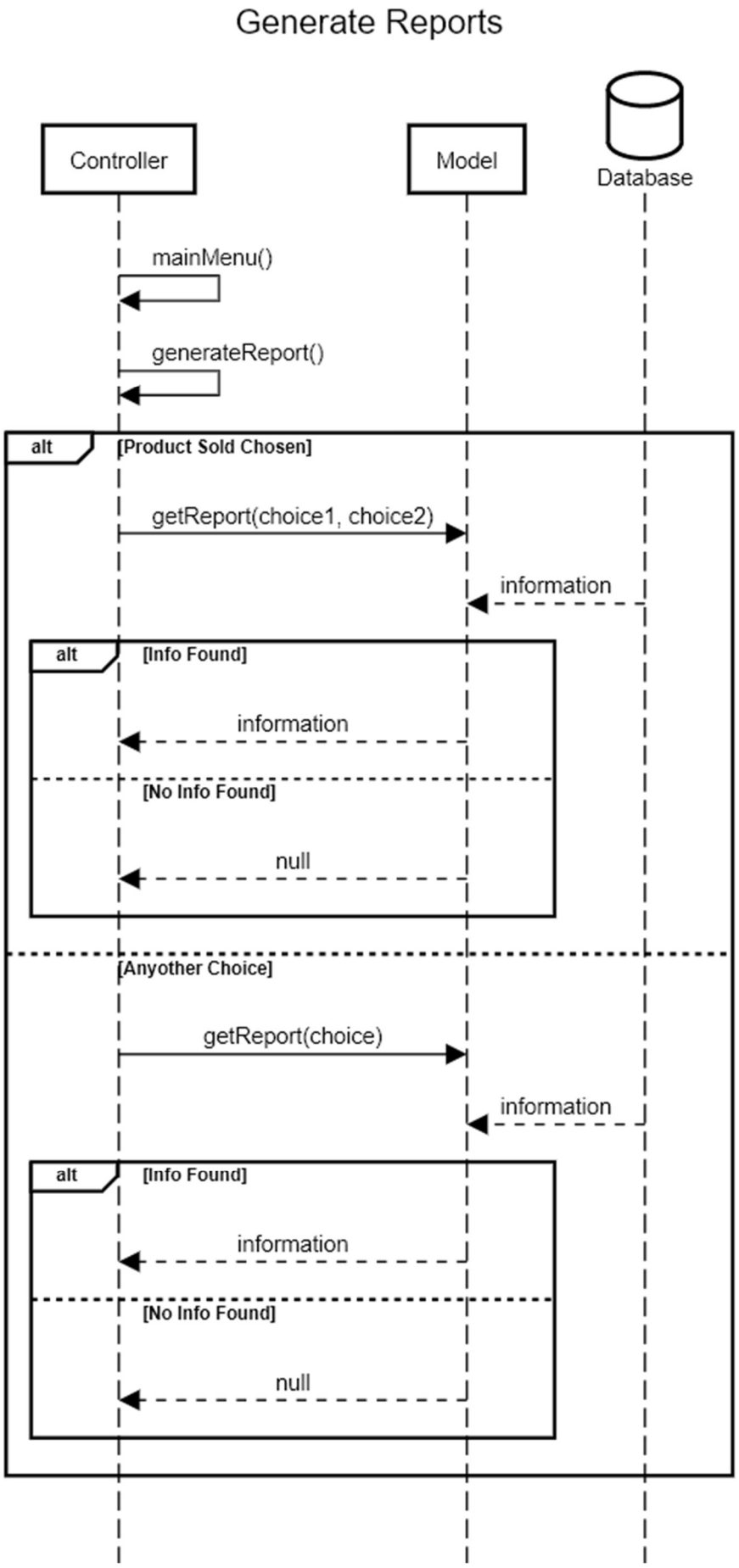
This is the sequence diagram adding stock to a product which was created from the use case description describing how adding stock to a product inside the database would work out.

Display Employee, Customer & Supplier Info



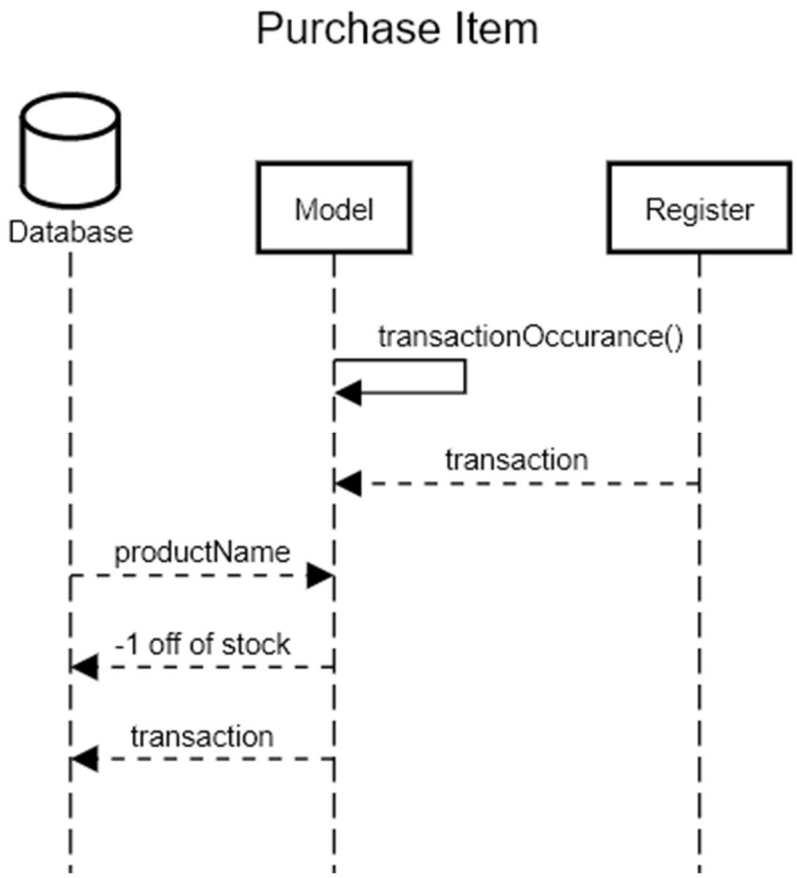
This is the sequence diagram for displaying either the employees, customers or the supplier's info which was created from the use case description describing about getting the information and then displaying that information to the Admin.

Generate Report



This is the sequence diagram for generating a report which was created from the use case description describing how the information would be gotten from the database and then depending on the choices the Admin is made, returning information for a specific report to display for the Admin to read.

Purchase Item



This is the sequence diagram for purchasing an item which was created from the use case description describing the procedure of when a customer has purchased an item by taking in the transaction details and then taking in the information to then be stored by the database. For the first iteration of the program this will be represented when the client clicks on a button on the main menu.

All sequence diagrams are subject to change in the future as I don't guarantee these will be the same when I start creating the application.

Next Iteration

For the next iteration I'll be creating the application by starting off with getting the UI sorted and then checking to see if the database and the application will work with each other. After the fact I will return to make the class diagram and activity diagram due to the fact that I'm still unsure on how the class will be layout and connected due to me using hibernate which

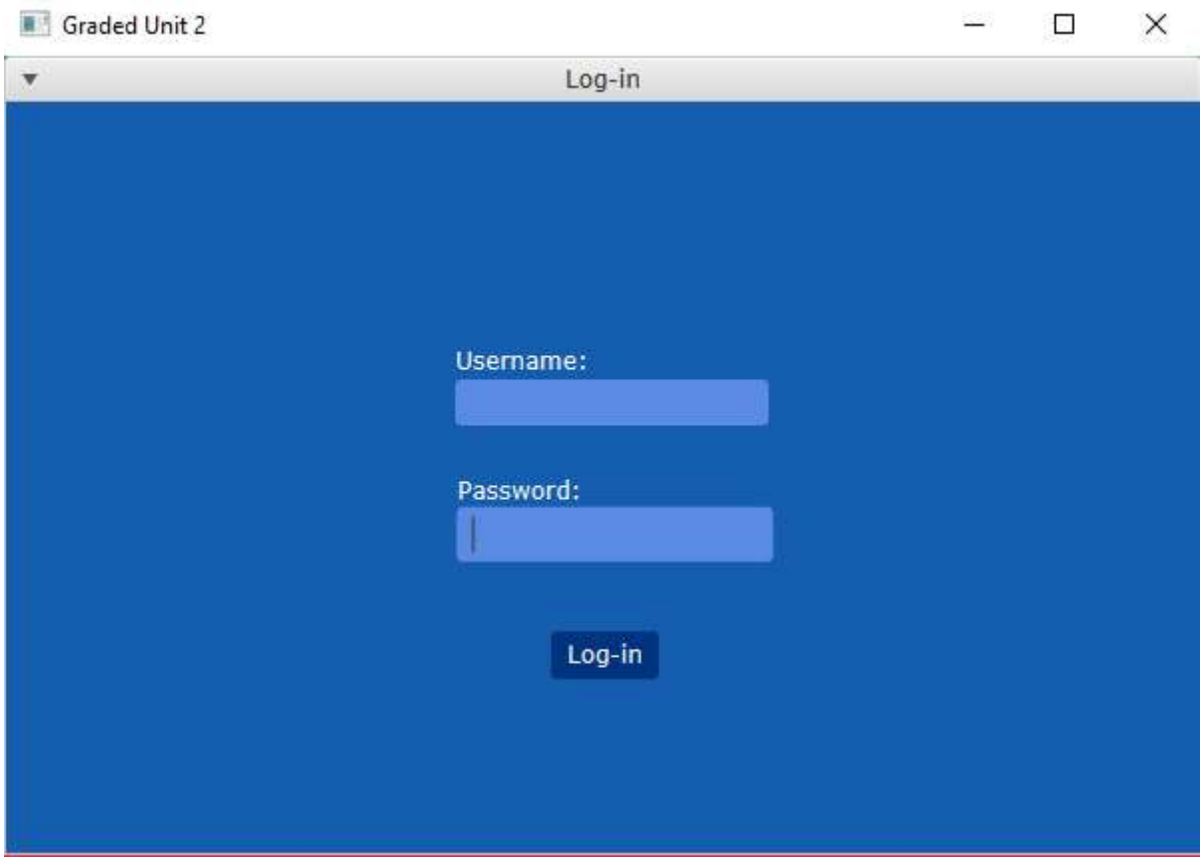
is one of my unfamiliar libraries I'm going to use and how my methods in my application might not translate in the future with my activity diagrams if I created them now. Also probably expand on the sequence diagram descriptions below the actual diagrams.

Third Iteration

Changes Before Going Ahead – 3

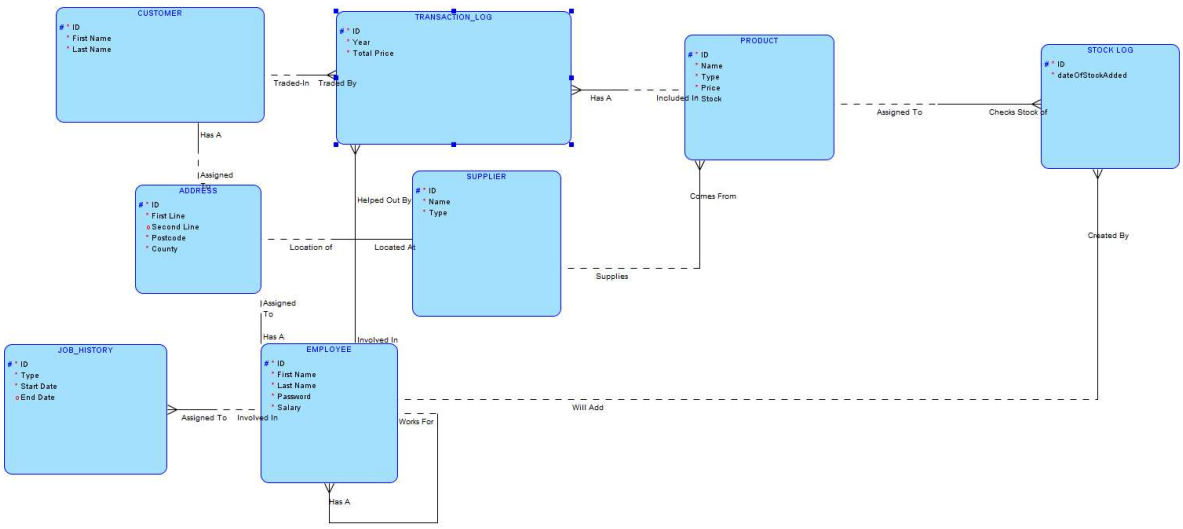
A lot has changed since the last iteration. I didn't just create the UI and the connection between the Java and database, but I created the entire prototype that is defined in the design from the use cases. From this developed prototype I then created a class diagram from the functionality I have. I then made a few activity diagrams describing how Log-In, add stock and search product works.

With the UI I spoke to the client to show what the UI will look like based off the wireframes and the colour scheme showing them the log-in screen and he was happy with the look and told me to continue on with that design. Here is what I showed the client:

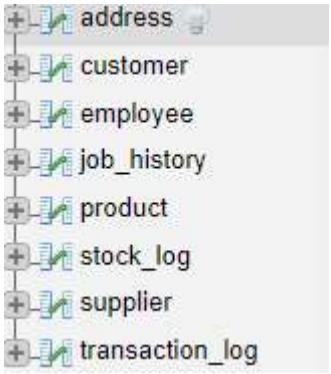


Database

So, when developing the program particularly storing who's added stock onto a product and when searching for a specific date for transactions, I didn't create a table of some sort that would store stock information and didn't make an attribute in the Transaction Log entity that will hold the date that the transaction took place so I went and did that as you can see from the ERD below:

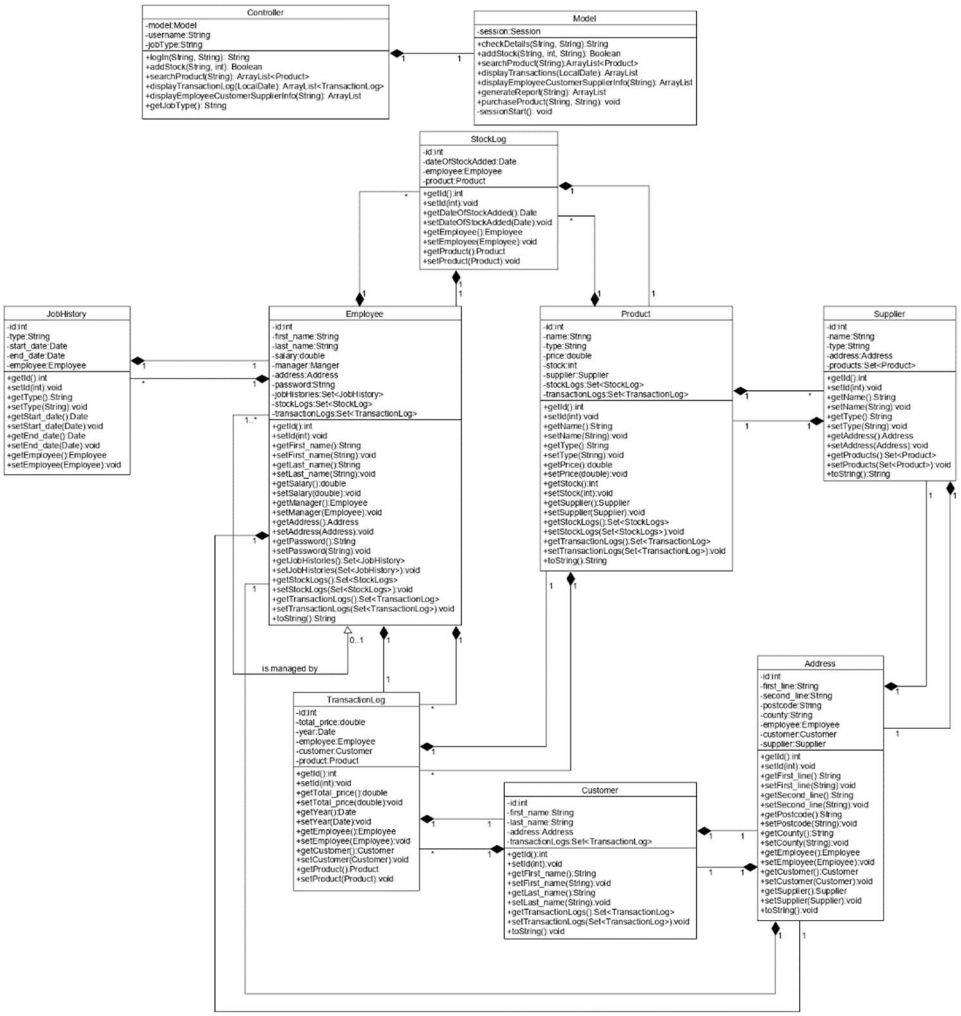


I then did the following procedure as I did when I created the database by converting this ERD into the physical model and then generating the SQL code to then import into my database. The stock log entity has the attributes of id which is number data type with a restriction of 3 whole numbers and dateOfStockAdded is date data type. The relationships the stock log has is employee to stock log with one employee can be involved in stocking many products whilst those stock logs will be linked to one employee. Also, in transaction log I added in a new attribute called Year which stores the date of the transaction with it storing the data type of date. This is the final number of tables that are in my database:

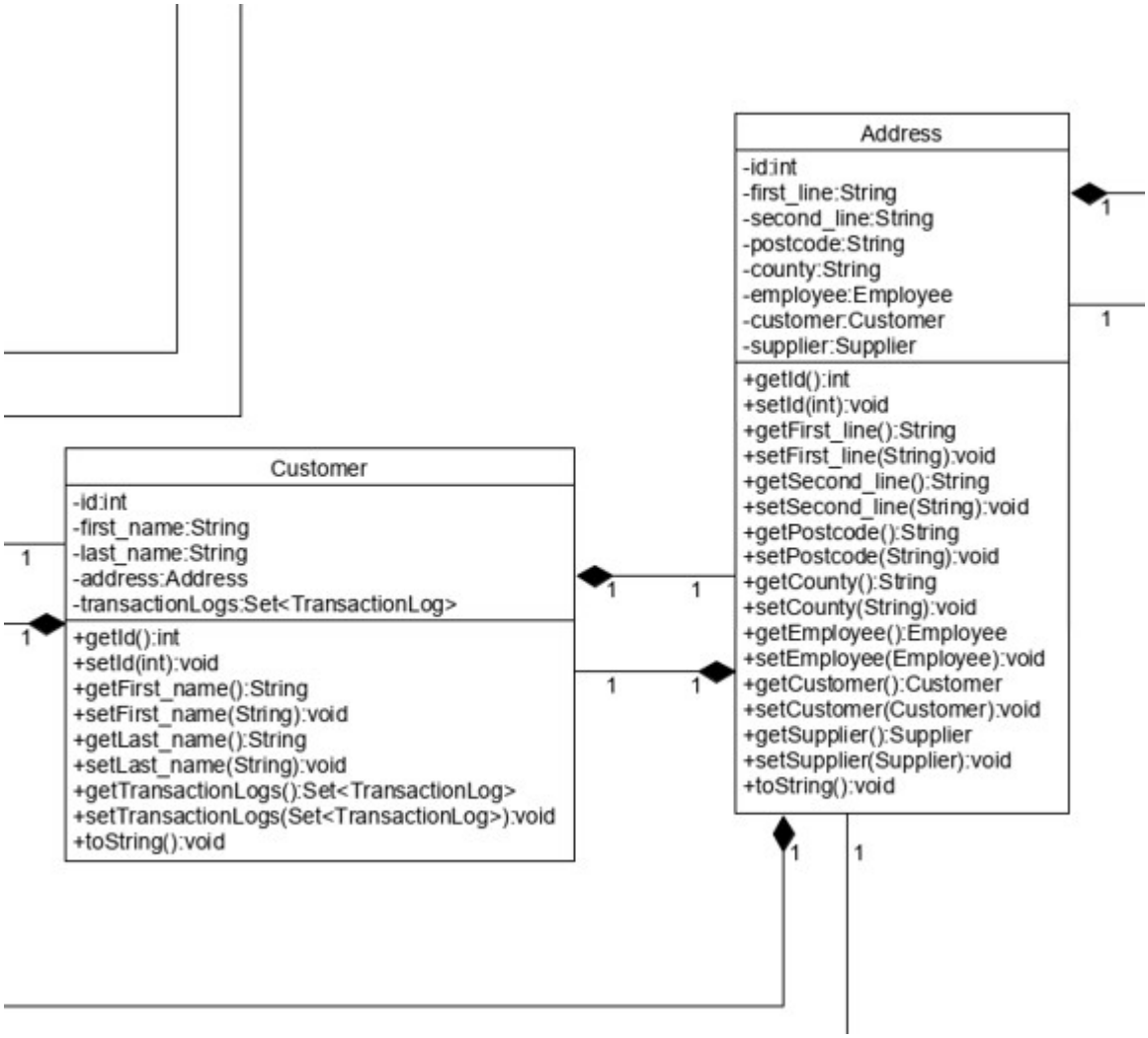


Class Diagram

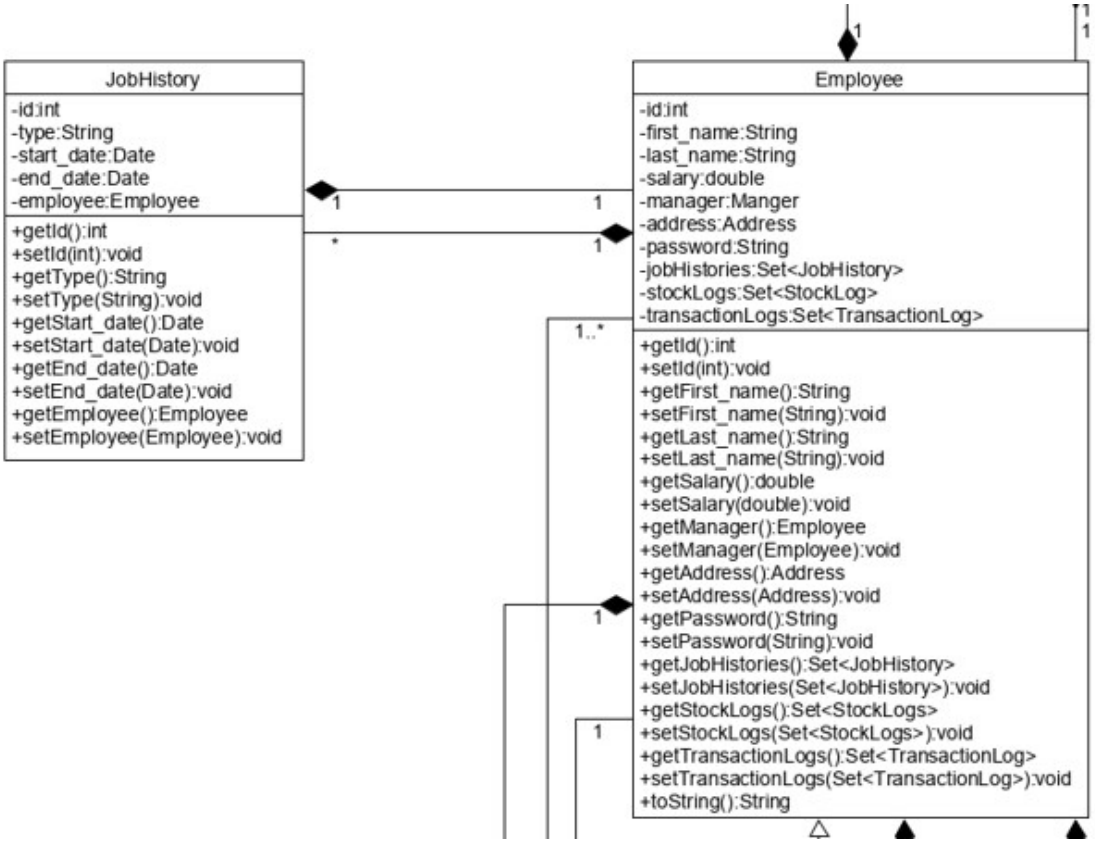
As I said above, I mentioned that I created a class diagram from the prototype that I developed from the functionality only. What I've defined in the class diagram I've defined the classes that will store the data from the database with the attributes and methods that are in that class with their access modifier showing if either that method or attribute can be accessed from another class or not. Here is that diagram:



It also shows the relationships between the classes which all of the classes have composition which means that the classes depend on the class that has the relationship with. Because I'm using Hibernate which represents tables in the database with class diagrams the composition relationships between the classes besides the controller and model are supposed to represent the relationships between tables. From the class diagram above the relationship in the table between customer and address is one-to-one so in the class diagram the relationship needs to be one-to-one from customer to address and in the other will be one-to-one from address to customer as seen below:



With one-to-many relationships one way will be one to many and the other will be one-to-one so for example employee could have many jobs so in a class diagram you'd represent it by having one way one-to-many with 1 to * and then the other way will be one-to-one as seen below:

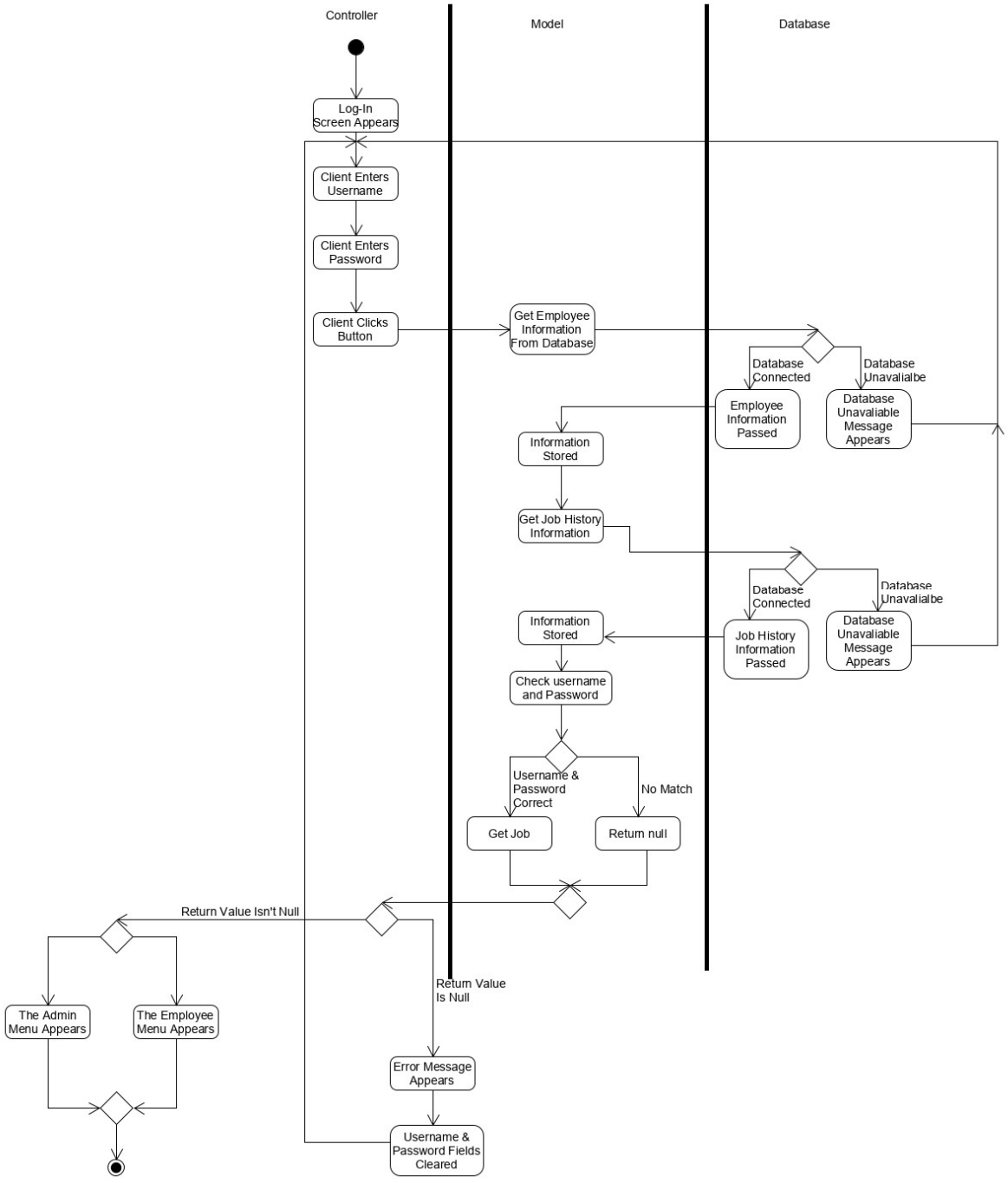


All of the data manipulation that involves calling the data from the database and then storing that data into objects will all happen in the Model class and then all the data will be passed into the controller class which will then be displayed.

Activity Diagrams

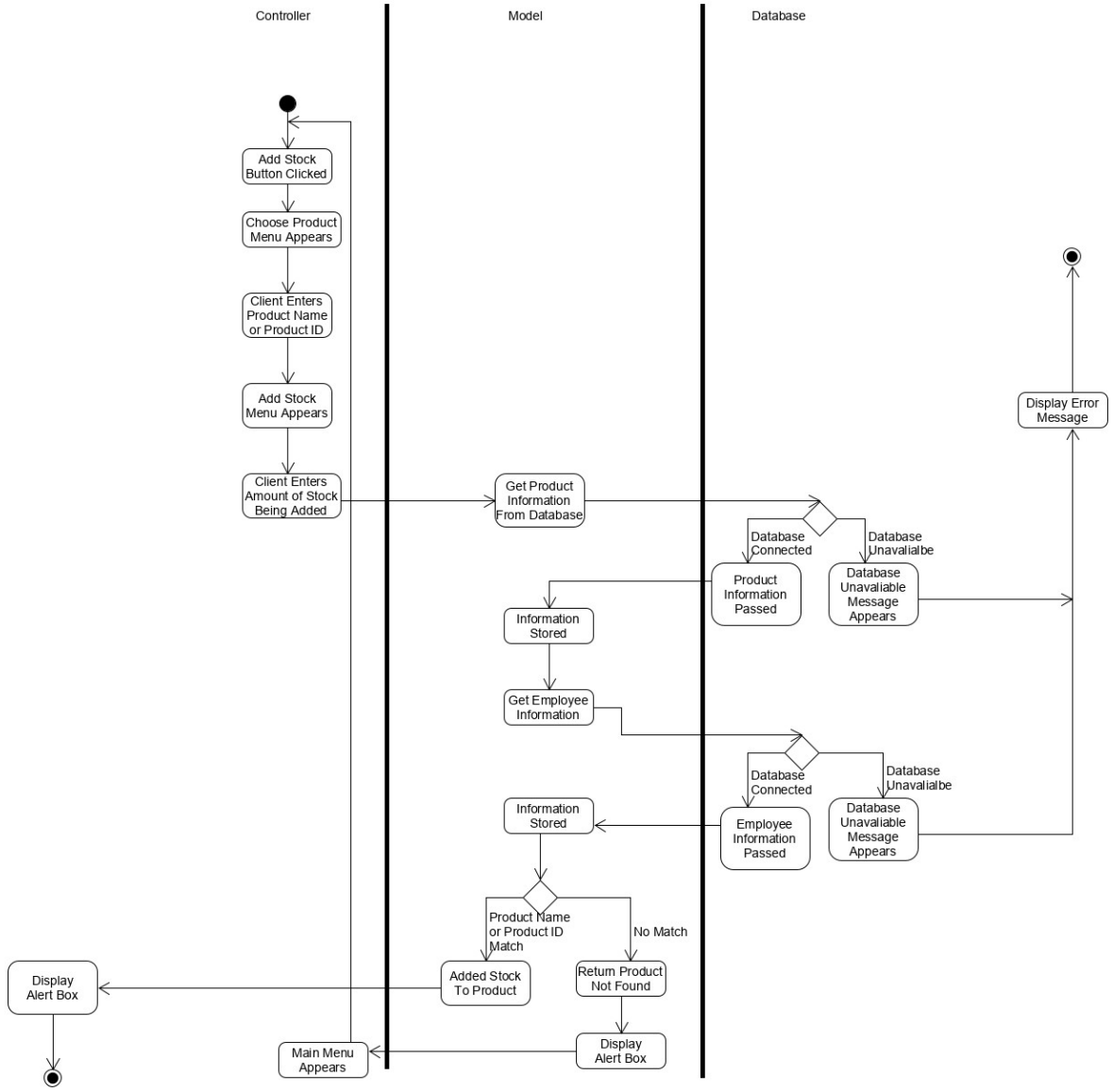
As I said earlier, I created activity diagrams which will represent the functionality step-by-step on what will happen. However due to time constraints I could only make a few of these diagrams due to a switch on what to prioritise. When this prototype is being further developed in the future, I will make the activity diagrams but for right now I'll describe what's going on in the diagrams that I made.

Log-In



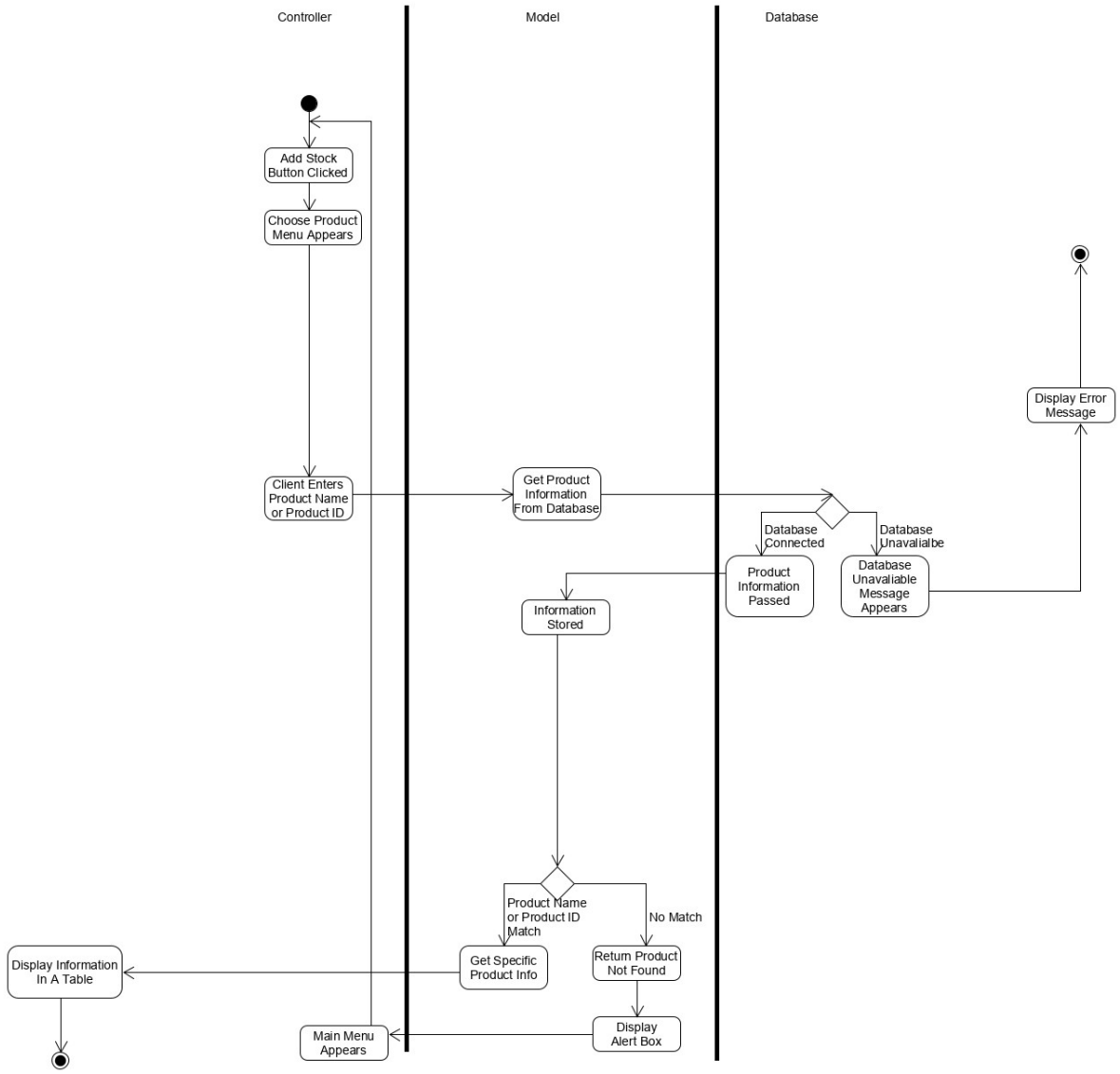
What this diagram shows is the step-by-step process on how the log-in works by getting the info from the client, sending the info into the model to then get info from the database and then checking the clients username and password to see if it matches with an employee and then when it doesn't the job the employee does is then passed into the controller and depending if there an admin or not a menu specific to the job type appears. If it doesn't match it will return back to the start displaying an error message telling the client that the username or password is incorrect. At any point when the database gets accessed and can't access the database it goes back to start stating that the database is unavailable.

Add Stock



This represents when the client clicks on the add stock button to then find the product, they want to add stock too and then specify the amount of stock they want to add. Same as the log-in the data is then passed into the model to then check if the product exists or not and same as before the data will come from the database and from that data it checks if the product specified matches then an alert box pops up displaying that stock has been added. If there isn't a match, then an alert box will display saying so and then will return to the main menu until the client clicks on add stock again. Same with log-in if the database is unavailable a message will display saying so but instead of returning to the beginning it closes the program.

Search Product



This represents when the client clicks on the search product button which will entail the client entering the product they want to look up and then the data will be passed into the model which will take in the data from the database specifically the products which then will be compared to what the product the client is looking for and if it matches then the data is then displayed into a tableview. Else similar to add stock an error message appears telling them that there isn't a product matching what the client defined and will return back to the main menu. How the database deals with the database being unavailable is the same as add stock.

As I said previously, I won't have time to finish creating the other activity diagrams now but maybe later on in a later iteration.

Development Stage

Unfamiliar Libraries

For my unfamiliar libraries I have used JavaFX using Scene Builder - (Gluon, n.d.) – for creating the GUI of the program, Hibernate - (Hibernate, n.d.) – to create the link between the database and the Java program & Spring Security - (Spring, n.d.) – for checking the password hash to see if the users input for a password matches the password from the hash.

Scene Builder

The reason I went and used Scene Builder is due to having previous experience with Swing using its counterpart with windows builder so when I found out about JavaFX, I assumed there had to be something similar which is of course Scene Builder, and this is how I discovered it also. The difference however is that with Scene Builder you'd need to create FXML files to create the UI of a certain scene and then after creating the scene you create a controller class specific to that FXML and then create the UI attributes like for example your buttons and text fields etc. Then you'd link the attributes by putting @FXML on top of the declaration inside the controller class. Here is the code for the Log-in screen to give an example:

FXML Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Defines the version of XML-->

<!--Imports all of the UI attributes being used-->
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.PasswordField?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.control.TitledPane?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>

<!--Creates the elements and then given them their attributes - with TitledPane it links the controller class specific to the log-in using fx:controller-->
<TitledPane alignment="CENTER" animated="false" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0"
            style="-fx-background-color: green;" text="Log-in" xmlns="http://javafx.com/javafx/8.0.172-ea" xmlns:fx="http://javafx.com/fxml/1" fx:controller="View.LoginController">
  <content>
    <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="180.0" prefWidth="200.0" style="-fx-background-color: RGB(21,93,175);">
      <children>
        <PasswordField fx:id="passwordField" layoutX="226.0" layoutY="203.0" prefHeight="27.0" prefWidth="158.0" style="-fx-background-color: RGB(90,138,225);" />
         <!--The Log-in button has an onAction attribute which links to a method so when the button is pressed that method will run.-->
        <Button fx:id="loginButton" layoutX="273.0" layoutY="265.0" mnemonicParsing="false" onAction="#buttonPressed" style="-fx-background-color: rgb(0,52,127);" text="Log-in" textFill="WHITE">
          <font>
            <Font name="Verdana" size="12.0" />
          </font></Button>
        <TextField fx:id="usernameField" layoutX="225.0" layoutY="139.0" style="-fx-background-color: RGB(90,138,225);">
          <font>
            <Font name="Verdana" size="12.0" />
          </font></TextField>
        <Label fx:id="passwordLabel" layoutX="226.0" layoutY="187.0" style="-fx-font: Verdana;" text="Password:" textFill="WHITE">
          <font>
            <Font name="Verdana" size="12.0" />
          </font></Label>
        <Label fx:id="usernameLabel" layoutX="225.0" layoutY="122.0" text="Username:" textFill="WHITE">
          <font>
            <!--Defines the font of the text in this attribute-->
            <Font name="Verdana" size="12.0" />
          </font></Label>
        <Label fx:id="errorMessage" alignment="CENTER" layoutX="154.0" layoutY="240.0" prefHeight="17.0" prefWidth="294.0" textFill="RED" />
      </children></AnchorPane>
  </content>
  <font>
    <!--Defines the font of the text-->
    <Font name="Verdana" size="12.0" />
  </font>
</TitledPane>
```

This is the FXML code for the log-in screen which declares the UI elements that's going to be displayed and the attributes associated.

Java Code:

```
package View;

import ...

//Declares Class
public class LoginController {

//Declares object of Controller
    private Controller controller = new Controller();

    /**
     * Button Used To Start Method
     */
    @FXML
    public Button loginButton;
    /**
     * Field To Enter Password
     */
    @FXML
    public PasswordField passwordField;
    /**
     * Field To Enter Username
     */
    @FXML
    public TextField usernameField;
    /**
     * Label To Show Where To Enter Password
     */
    @FXML
    public Label passwordLabel;
    /**
     * Label To Show Where To Enter Username
     */
    @FXML
    public Label usernameLabel;
    /**
     * Label To Display Error Messages
     */
    @FXML
    public Label errorMessage;
```

```
     * Takes in the data from the username and password field and checks to see if valid.
     * @param actionEvent
     * @throws IOException
     */
    public void buttonPressed(javafx.event.ActionEvent actionEvent) throws IOException {
        try {
            //Initialise variable
            String jobType;
            //Will pass through the username and password the user has entered
            // and then will take in the value coming back from the method.
            jobType = controller.logIn(usernameField.getText(), passwordField.getText());
            //If the employee is the admin
            if (jobType.equals("Admin")) {
                //Displays The Admin's Menu
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(getClass().getResource( name: "MainMenuAdmin.fxml"));
                Parent mainMenuParent = loader.load();
                Scene mainMenuScene = new Scene(mainMenuParent);

                //Passes through the controller made here so that no value is lost.
                MainMenuAdminController mainMenuAdminController = loader.getController();
                mainMenuAdminController.initData(this.controller);

                Stage window = (Stage) ((Node) actionEvent.getSource()).getScene().getWindow();
                window.hide();
                window.setScene(mainMenuScene);
                window.show();

                //If either the employee's username or password does not match then
                //the fields are cleared and then an error message will display using a label
            } else if (jobType.equals("ERROR")) {
                usernameField.clear();
                passwordField.clear();
                errorMessage.setText("Either Username or Password is incorrect! - Try Again!");
            } else  {
                //Displays the menu for employees only
                FXMLLoader loader = new FXMLLoader();
                loader.setLocation(getClass().getResource( name: "MainMenuEmployee.fxml"));
                Parent mainMenuParent = loader.load();
                Scene mainMenuScene = new Scene(mainMenuParent);

                MainMenuEmployeeController mainMenuEmployeeController = loader.getController();
                mainMenuEmployeeController.initData(controller);

                Stage window = (Stage) ((Node) actionEvent.getSource()).getScene().getWindow();
                window.hide();
                window.setScene(mainMenuScene);
                window.show();
            }//END IF/ELSE
```

```
        } catch (HibernateException e){
            //If the database cant connect then the error message displays in a label.
            if (e.getMessage().equals("Connection refused: connect"));
            {
                errorMessage.setText("Database Connection Is Unavailable!");
            }//END IF
        }//END TRY/CATCH
    }//END METHOD buttonPressed

}//END CLASS LoginController
```

This is the controller class that will have the UI elements defined in the FXML file declared with the action event as well which will check the username and password and depending on the value coming back displaying a menu or an error message.

Hibernate

I used Hibernate to link my database to my Java application. What is used for is to map the data in a table to an object so for example I have a table in the database called Customer and if I wanted to get data from that table I would need to make a class called Customer with the attributes matching the attributes inside Customer table including the foreign keys. How I mapped the classes to the tables is that I firstly needed to create a configuration XML

file so that it knows where the database is, the username and password of the database and the connector to use so for what I'm using is MySQL so I'll need to use the connector specific for MySQL.

After that I had to make a configuration file for each class I had to make so using Customer again I specified what class is going to link to the table, define what the id will be which is id, the properties of the class which in this case its first name and last name which is linked to the attributes in the table and then finally define the foreign keys which in this case is a one-to-one foreign key to address which is stored in an Address object and a one-to-many foreign key for transaction log which is stored in a set of Transaction Log.

I would then put a mapping tag in the original config file which maps the customer config file to the class. To start up hibernate I made a method that will use the session factory that is defined in the configuration file so I can start a session at any time I need to. Using HQL which is hibernates own query language to query the table and then get the information from the table so using customer taking in the information into an ArrayList of Customer.

Here is an example of what I've just previously talked about using the customer class, config file and customer config file:

Hibernate Config File:

```
<?xml version='1.0' encoding='UTF-8'?>
<!--Defining what XML is being used-->
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<!--Defines that this is a hibernate configuration file-->


<hibernate-configuration>
    <!--This defines what database and SQL is being used and the mapping of the classes-->
    <session-factory>
        <!--Defines the database and SQL-->
        <property name="hbm2ddl.auto">update</property>
        <property name="dialect">org.hibernate.dialect.MySQL55Dialect</property>
        <property name="connection.url">jdbc:mysql://localhost:3306/graded_unit_2</property>
        <property name="connection.username">root</property>
        <property name="connection.password">root</property>
        <property name="connection.driver_class">com.mysql.jdbc.Driver</property>

        <!--Defines the mapping between the classes and tables-->
        <mapping resource="HibernateXML/employee.hbm.xml"/>
        <mapping resource="HibernateXML/address.hbm.xml"/>
        <mapping resource="HibernateXML/customer.hbm.xml"/>
        <mapping resource="HibernateXML/jobhistory.hbm.xml"/>
        <mapping resource="HibernateXML/product.hbm.xml"/>
        <mapping resource="HibernateXML/stocklog.hbm.xml"/>
        <mapping resource="HibernateXML/supplier.hbm.xml"/>
        <mapping resource="HibernateXML/transactionlog.hbm.xml"/>
    </session-factory>
</hibernate-configuration>
```

This is the hibernate configuration file that defines what database and what kind of database I'm using plus mapping the config files for each class & table.

Customer Config File:

```
<?xml version='1.0' encoding='UTF-8'?>
<!--Defines the XML version-->

<!--Defines that this is a hibernate mapping file-->
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
    <!--Defines what class is linking to what table-->
    <class name="com.LeeGlen.Customer" table="customer">
        <!--Defines the ID-->
        <id name="id">
            <generator class="assigned"/>
        </id>
        <!--Defines the linking attributes-->
        <property name="first_name" column="First_Name" type="java.lang.String"/>
        <property name="last_name" column="Last_Name" type="java.lang.String"/>
        <!--Defines the relationships-->
        <one-to-one name="address" cascade="all"/>
        <set name="transactionLogs" table="transaction_log" inverse="true" cascade="save-update" lazy="true">
            <key>
                <column name="ID" not-null="true"/>
            </key>
            <one-to-many class="com.LeeGlen.TransactionLog"/>
        </set>
    </class>
</hibernate-mapping>
```

This is the Customer configuration with the definitions of what class is linked to this, what attributes it has and the relationships between the classes/tables.

Customer Class:

```java
package com.LeeGlen;

import java.util.HashSet;
import java.util.Set;

//Declares the class
public class Customer {

    //Declares the attributes
    private int id;
    private String first_name;
    private String last_name;
    private Address address;
    private Set<TransactionLog> transactionLogs = new HashSet<>();

    //Declares the getters and setters
    public int getId() { return id; }

    public void setId(int id) { this.id = id; }

    public String getFirst_name() { return first_name; }

    public void setFirst_name(String first_name) { this.first_name = first_name; }

    public String getLast_name() { return last_name; }

    public void setLast_name(String last_name) { this.last_name = last_name; }

    public Address getAddress() { return address; }

    public void setAddress(Address address) { this.address = address; }

    public Set<TransactionLog> getTransactionLogs() { return transactionLogs; }

    public void setTransactionLogs(Set<TransactionLog> transactionLogs) { this.transactionLogs = transactionLogs; }

    //Prints the needed information
    @Override
    public String toString() {
        return id + " " + first_name + " " + last_name + " " + address.toString();
    }
```

This is the customer class that has the attributes of the class defined and the getters and setters of the attributes which will be used by hibernate to get and set the data in the database. The address and the transactionLogs are used to store the foreign data from both the Address and Transaction Log tables.

Customer Table:

#	Name	Type	Collation	Attributes	Null	Default
1	ID	tinyint(4)			No	*None*
2	First_Name	varchar(20)	latin1_swedish_ci		No	*None*
3	Last_Name	varchar(20)	latin1_swedish_ci		No	*None*
4	ADDRESS_ID	tinyint(4)			No	*None*

This is the customer table to show the representation of the attributes between the table and the class. The transaction log foreign key isn't in the table as it is in the transaction log table using the customer id.

How I found out about Hibernate was due to the fact I wanted an alternative to just using JDBC as I thought at the time JDBC would be hard to use for what I was going to do and already in my head I wanted to do something similar to what ORM does. So I had a look at what you could use in a Java program through looking at IntelliJ's libraries you could import and I saw Hibernate so I researched what it was and then later on looked at Youtube videos on its use and that when I wanted to use Hibernate because it looked real easy to use.

Spring Security

Spring Security is a framework that is used create security to a Java application by giving it authentication and authorization possibilities. How I found out about this is that I heard about BCrypt which is encryption and I wanted to use this with the passwords so that when someone enters their username and password, the password hash is pulled from the database and is compared to the password provided by user to see if they match and from my knowledge base Java doesn't have this so I googled what Java framework has BCrypt and sure enough Spring Security has BCrypt. I've only used it once to check the password from the user matches the password hash from the database which is shown here:

```
if (username.equals(employee.getId() + employee.getFirst_name()) && BCrypt.checkpw(password, employee.getPassword())) {
```

The Code of the business model

The program has been created and has matched the design. The descriptions and sequence diagrams match what happen in the program. The class diagram matches the classes for the functionality as it was pretty much taken from it besides two methods in the model class which was openSession() and closeSession() which will open or close a hibernate session which is what is used to get access to the database.

I went online to get a regular expression to check if the date for the transaction is correctly formatted - https://www.regextester.com/96683.

Here is a sample of my code which is how the program checks log-in details which is from the model class:

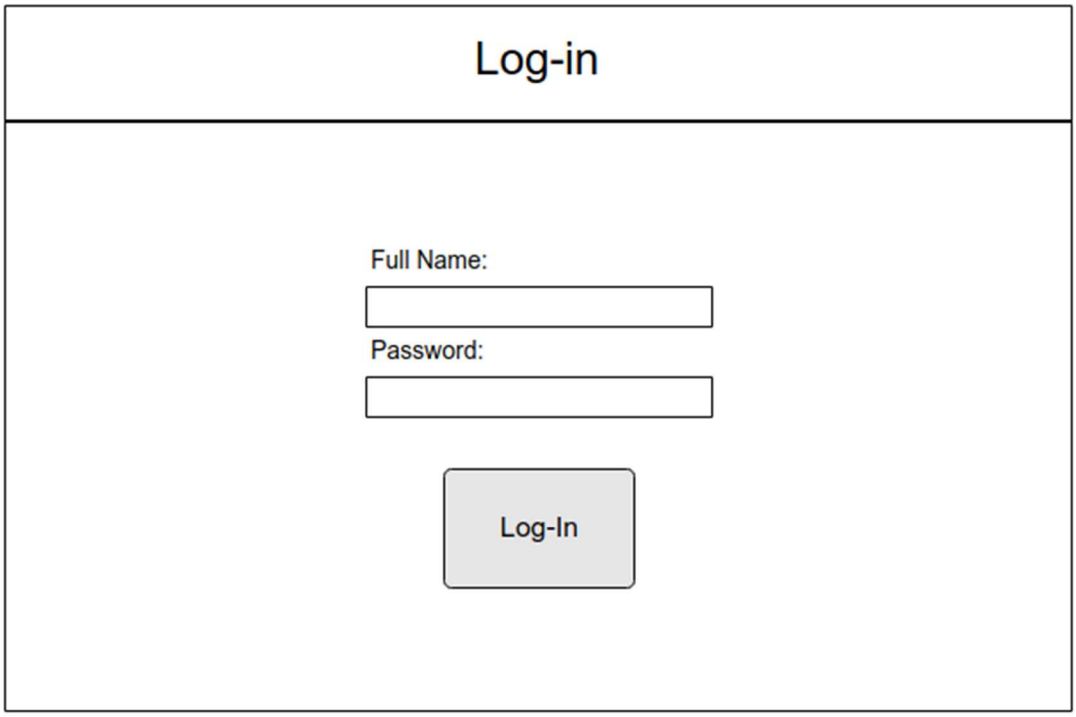
```
/**
 * Checks to see if user defined details match details with the data in employee table.
 *
 * @param username The username the user specified.
 * @param password The password the user specified.
 * @return Either a job type or null.
 * @throws HibernateException If the database isn't running.
 */
public String checkDetails(String username, String password) throws HibernateException {
    //Starts the hibernate session
    session = sessionStart();
    session.beginTransaction();
    //Runs a query to get all data from the employee table
    String HQL = "from Employee";
    Query q = session.createQuery(HQL);
    List<Employee> employees = q.getResultList();
    //Runs a query to get all data from the job history table
    HQL = "FROM JobHistory";
    q = session.createQuery(HQL);
    List<JobHistory> jobHistories = q.getResultList();
    //Finds the matching employee and then return their job type
    for (Employee employee :
            employees) {
        if (username.equals(employee.getId() + employee.getFirst_name()) && BCrypt.checkpw(password, employee.getPassword())) {
            for (JobHistory history :
                    jobHistories) {
                if (history.getEmployee().getId() == employee.getId() && history.getEnd_date() == null) {
                    String jobType = history.getType();
                    return jobType;
                }//END IF
            }//END FOREACH
        }//END IF
    }//END FOREACH
    //If there is no matching employee returns null
    return null;
}//END METHOD checkDetails
```

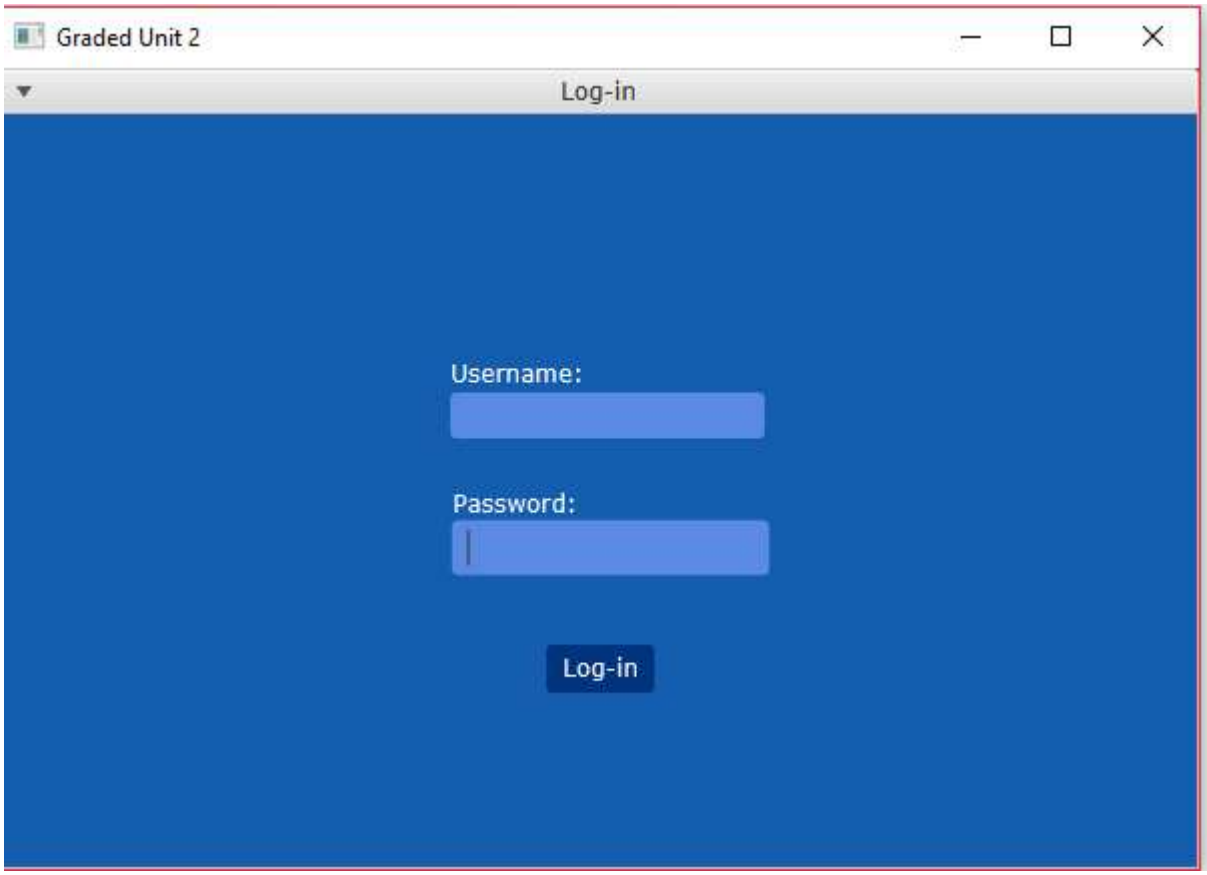
Coding of the UI

I have developed the UI based of the wireframes and colour scheme I made in the first iteration using scene builder with FXML so I made all of the text Verdana as I said in the colour scheme to help people with dyslexia to be able to read the text plus I followed the colour scheme by making the UI have the colours I chose.

The changes I made where that when displaying employee, customer, supplier, product or transaction data I used a table instead of a text box because it looked a lot neater and easier to read and understand what that data is by using the columns.

Here's a sample of what the UI is and how it follows the design by showing the Log-in screen wireframe and the actual program:





Test Plan

So, for how I'm going to test this program is that I'm to do unit testing and functionality testing.

Unit testing will be used for the functionality of the program by testing each part of the program with test data depending if it has user input or not like for example checking to see if a button goes to its destination or does the program take a correct product name to search for just to find any errors that where possibly unseen by the programmer.

Functionality testing will be used to check to see if the UI of the program functions as expected like for example the page can take in text for the log-in or that the choices in a choice box can be used. I'll also be checking to see if other people can do some of the functionality as well.

Test Runs

Unit Testing

In the folder called tests I've created a test plan, test data and test log. The test plan is what is going to be tested in the program, what test data is going to be used if at all and what is my expected result. I then created the test data that'll be used to test the program with and are all linked back to the tests in the test plan. The test log is used to see what the actual result was, if the tests passed as in, they did what is expected or failed and then a comment if necessary on what may have either caused this error or that they may be further explanation needed.

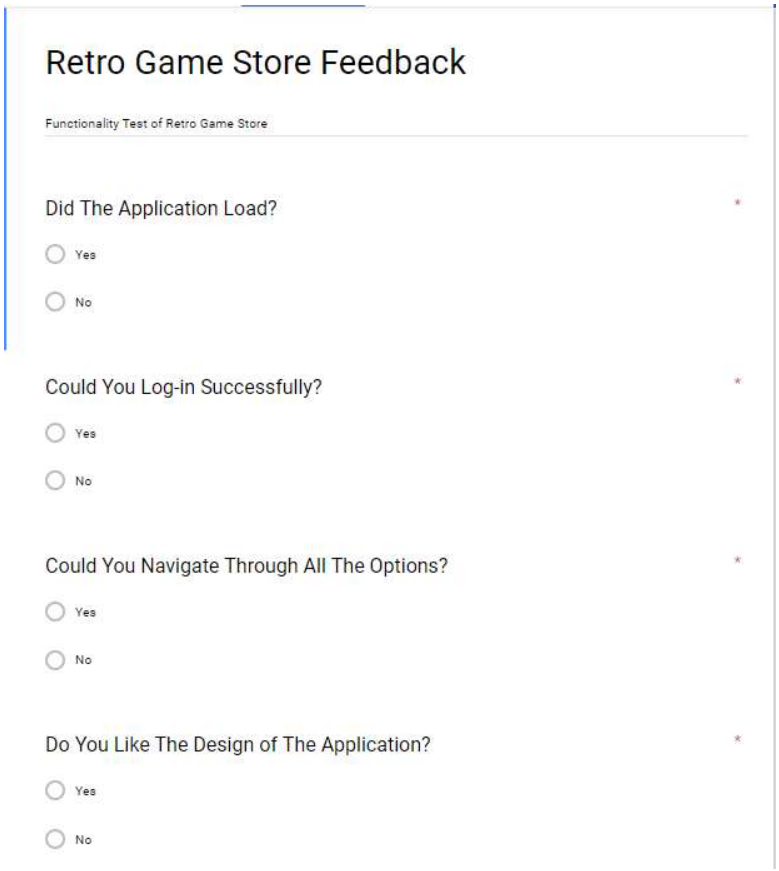
I did encounter errors –

- The database wouldn't give an error message to then close the page. What I think happened is that I've incorrectly handled the wrong exception.
- The stock could 0 even though this may not be bad it still should give an error saying that the added stock shouldn't be 0
- The main menu button from the display employee, customer, supplier, product and transaction logs wouldn't return back to the menu.

These errors will be fixed in the later iterations of the program.

Functionality Testing

I got three people to test my program by navigating the program, clicking on buttons and doing the functionality of the program and the test data in the database isn't real information or is information that is public. They then went and filled out a form based on the testing. This is the form:



Retro Game Store Feedback

Functionality Test of Retro Game Store

Did The Application Load? *

- Yes
- No

Could You Log-in Successfully? *

- Yes
- No

Could You Navigate Through All The Options? *

- Yes
- No

Do You Like The Design of The Application? *

- Yes
- No

Do You Like The Color Scheme of The Application?

- Yes
- No

Could You Read The Text Easily? *

- Yes
- No

Did The Products You Searched For Return Details? *

- Yes
- No

If Selected No What Happened?

Long answer text

Did You Successfully Add Any Stock? *

- Yes
- No

If Selected No What Happened?

Long answer text

Did The Employees, Customers And Suppliers Display Details? *

- Yes
- No

If Selected No What Happened?

Long answer text

Did The Transaction On The Given Date Appear? *

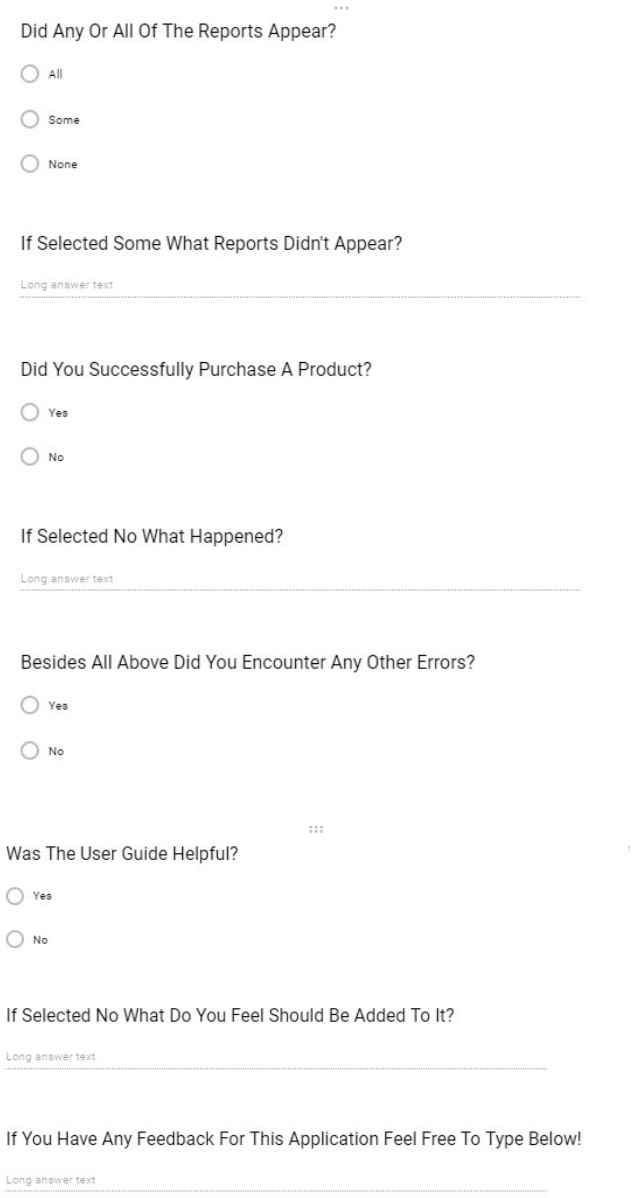
- Yes
- No

If Selected No What Happened?

Long answer text

Did Any Or All Of The Reports Appear?

- All
- Some
- None



From the answers all of them could load the application, 2 could log-in but only one couldn't, everyone said they could navigate the options, they all also said that they liked the design and the colour scheme and could read the test easily so the UI is staying until more people test the program.

Two could get the products they searched for besides one as they said that nothing happened and when I looked further into this I found out that the search product would always call null if the name couldn't parse an int so what I did is make so that it will return null when the id of the product matches the amount of products stored.

Everyone got to successfully add stock however due to one person not being able to log-in as an admin they couldn't display any of the employees, customers or suppliers but the other can. This is due to me giving out the wrong password for admin but correcting it later.

All three could get a transaction and display all of the reports however only one could successfully mimic a purchase product as both couldn't see a transaction be stored in the

transaction log and the other had a database error happen when they entered a id but on my end it would store a transaction log and didn't get any database error but I assume that this was done on the employee main menu so that'll be checked to see if that's causing the issue.

None said they encountered any errors besides the ones they've already said, said that the user guide was helpful and the feedback was that one person couldn't log in into the admin but that was my fault but the other suggestion was a back button which the person is right because if a button in the main menu is clicked on by accident you can't go back until that section is finished so that'll be added into the next iteration.

In the tests folder is the results from this form.

Users guide

With the user's guide I made two – one for the tester and one for the user. The difference between the two are that the testers has information on how to use UwAmp which is the software that is used to make a local server so that the program can get the data inside the database whilst when this program has finally finished then the database will be somewhere else and talks about having a computer that can connect to the internet by using either ethernet or WIFI.

Outline of the Assignment

The requirements that where finalized in the inception phase from the various talks with the client and research done and the requirements that were added before design started – (Changes before going ahead) - where:

- Log-in – Logs the user into the system.
- Purchase Item – Detects a purchase has been made.
- Generate Reports – Generates various reports.
- Add Stock – Adds' stock to a product.
- Search Product – Searches for a product
- Display Transaction Logs – Displays the transaction logs
- Display Employee, Customer or Supplier Info – Displays the details of either Employee, Customer or Supplier.

Log-in

With log-in what happens is that either the employee or the admin will enter the details which is their full name and password and then the program will validate the data received to check if both the full name and password matches any in the database and if so then the program will load the main menu. In the actual program how the program logs in is the same but the details being entered is different.

Besides the full name being entered what actually is entered is a username based of the id of the user and then the first name of the user. The program will then send them to two different menus depending if there an admin or just an employee. I had troubles trying to create the log-in by having an issue on how BCrypt is used, with BCrypt when trying to

decrypt the password given by the user I would get an invalid salt revision so how I fixed this was creating the hash inside the program and then taking that hash and then inserting the hash to the employee it's going to be assigned to.

Purchase Item

With Purchase Item what was meant to happen was that when a purchase has been made by the customer the program should automatically detect that a product has been purchased and that one stock would be removed from the product and a log would be saved. In the program because I didn't have an idea on how this would work out I made a button on the main menu for both the employee and admin to simulate how this would work out by making the admin/employee enter the product name/id to then remove stock by one and then to make a transaction log just to show the process of how a product is purchased. However this also means that only one product is being purchased rather than multiple products so let's say a customer is purchasing a console with multiple games, the transaction log is going to be made for each product being purchased and the total being the products price rather than all three being in the log with the total being those products added up.

I had a problem when developing this part of the program due to a fix I did in the display transaction logs function which was that once a product been purchased and if another product was going to bought afterwards the product would be purchased. This was due to the program already committing a transaction that folded of purchasing the product and then storing a transaction log so what needed to happen was I needed to do was close the session after the transaction log was stored and then checking to see if the session was closed at the start of each function besides log-in to then open the session to get data from the database.

Generate Reports

With Generate Reports its purposes are to generate reports of different variety of reports to the user by allowing them to pick a choice to display and then the screen should be their choice with the data related to that choice. The client can pick from all products, products that has no stock, most popular products, employees who've sold the most products and products that've been sold in a specific time frame from week, month and year. In the program I knew how all of this would work by taking the information from the database and then displaying them which I did so but the only difference is that I don't have all products and the reason is that I forgot that was one of the reports it needed to do however this would be really easy to develop as I've already done similar things with display employee, customer and supplier info as it would just take all of the info from the database specifically about the product and then displaying into a table.

With generate reports at the time of development I had problem getting specific data from the database due to being inexperienced with what data is being pulled from the database so the problem was that I needed to take in a count of data so for the getting every single product that's been sold into the program, what I did to fix the problem was that I needed to store the data into an ArrayList of an array object – the first index of this array is the

product and the second index is the count – and then using this ArrayList I then displayed the data into a TextField.

Add Stock

With add stock its purpose is to add stock to a specific product specified by the user by having the user to enter the products name or id and then the program will check to see if the products exists and if so then the user will enter the amount of stock will be added. This was fairly easy to program and how the program functioned in the end with this is function is that the user will enter the product name or id and then clicks enter, however the validation won't happen until the user enters the stock and then will check to see if the product exists and then if so the stock will be added to the product and then will create a stock log using the employees name and the product to see what employee has entered the stock in case of an error of false stock.

I had a problem when developing this part of the program due to a fix I did in the display transaction logs function which was that once a product has added the stock and if another product was getting stock added to the program wouldn't add the stock. This was due to the program already committing a transaction that folded of adding the stock so what needed to happen was I needed to do is do what I did in purchase product which was closing the session after stock being added and then checking to see if the session was closed at the start of each function besides log-in to then open the session to get data from the database.

Search Product

With search product the user should be able to enter the product name or id and should check to see if the product exists and if so the information of that product specifically its name, type, genre, publisher, price, trade-in price should be displayed so the user could see the information. The program functioned exactly as I set out to do however the information displayed about the product was the id, name, type, price and then stock instead of what I specified above due to me forgetting what the information was going to be stored in the database as I was designing the database and due to this I was using the databases info about the product instead of the functional requirements info.

I did have troubles with creating this function as it was the first time actually using HQL to get the products data and then figuring out a way to store the data which I figured out by getting the query to store the data into an ArrayList and then searching through the ArrayList to find the product the user is looking for.

Display Transaction Logs

With display transaction logs what happens is the user should be able to enter a specific date a transaction may of occurred and then the transaction information should display showing the employee who sold the product, the customer that bought the product and the products that was sold and the time/date the product was sold and a special id that is used to check receipts. In reality the program does get the data from a specific date however the information is different as in the customer may not display as in reality the customer wouldn't really give out information to the company until potentially if they were returning

the product or trading in the product however the rest does display so the employee, the product and the date with the id.

This was difficult to display at first as when I was developing the program, I was using hibernate to open a session to get the data for the transaction logs and then closing the session after getting that information. However, in doing so deleted the data related to the other tables linked to transaction logs so customer, product and employee as it couldn't find the data linked to the transaction log meaning I couldn't display the data so what I did was kept the session open after logging into the program so that when the transaction data is gotten and when the data is being passed back to where it will be displayed the data of the employee, customer and product isn't lost. This change caused quite a bit of errors in other places such as the add stock and purchase product in which I discussed about.

Display Employee, Customer & Supplier Info

With display employee, customer & supplier info what should happen is that depending on the user's choice either all employees, customers or supplier's information should be displayed. For the employee the information of the employee displayed should be name, job type, phone number, address and salary. For customer the data should be their name and their address. Finally, the supplier data should be name, address and type.

The program does display all three and there is only one difference which is the employees doesn't have a phone number because It's the same reason why there isn't a display all products report is that I forgot. Didn't really have difficulties with this as it was just pulling all the data from a specific table and then displaying that information. The only problem at the time was displaying the info into a table which turns out wasn't really that difficult as you specify what table you're displaying and the data type of the info coming in and then just assigning the data.

Overall almost all the functional requirements were met besides purchase product however I did represent it in the program showing how it would work but it was pretty shit and the differences are little like the information is a tiny bit different when displaying the data or the products data coming out being quite different where validation occurs etc.

Strengths and Weaknesses

In this section of the evaluation I'll be focusing on my strengths and weaknesses during the whole project.

Inception Phase

During the inception phase I had trouble at the start with trying to getting an understanding on what the project will do like for example I thought you could buy products using this product and I was getting frustrated on my interpretation of the brief as I thought it was the aims of the brief at first on what the program was going to do out of assumption but in reality it was what that program will be – either a website or an application. I realised this after the interview with the client talking about the use case diagram and the functional requirements of what the program will be able to do because I got an understanding on what the client wanted in the first place.

What I did quite good on was that the initial functional/non-functional requirements as I got certain functional requirements dead on like the displaying customer, supplier information and update stock even though update stock in the initial function requirement does something different I had the idea there that there was going to be some involvement with updating stock. Even the initial use case which had three of the use cases that appears in the updated top-level use case was pretty spot on what the admin has potential in doing at the time.

I felt like the interview with the client went really well as the client gave me really informative answers based certain questions on the functional requirements I had created telling what they specifically wanted, further ideas on what certain things should do, telling me what specific things that were in the functional/non-functional requirements and use case diagram that they didn't need like a printable report and overall getting the answers I really needed to advance the project.

Next up I did the research on how other systems that are similar to the brief to see how other businesses deal with a similar problem to the project which I felt quite good doing as I have knowledge on this industry and know quite a few businesses that do something similar to how the clients business is and then from those businesses I looked for the similarities of how that business will do certain things like how they would deal with trade-in's but In all honesty it didn't really help in the long term when designing and then developing the program as all it does is validate what I'm doing is similar to other businesses.

The aims of the brief shows what the client needs the program to do in the end which was easy to create and the only difference is the aims of the brief to the program created is that the customers payment details hasn't been stored but other than that everything else has been programmed into the final program. This was also a cake walk as I accidently created this when I was making the interpretation of the brief thinking this was the interpretation.

With the new functional/non-functional requirements all the functional requirements were quite simple to change as I just used the answers I received from the interview with the client and some of the research I did by looking at other businesses and pretty much keeping the non-functional requirements.

All these functional requirements have been represented in the actual program in one way or another and how the employee/admin would use some of the requirements is how it works in the program too. From those functional requirements I then re-created the basic use case diagram to use the updated functional requirements which was easy to create however the register system wasn't created during the construction of the program as the program itself represented the functional requirement of purchase item to mimic if a real register was being used.

The research on what resources I would need to use was a cake walk as I knew what I'll be needing to use throughout the elaboration and construction stages which most of it was obvious. The things I didn't use was the printer during the entire project and that is it, everything I identified was used at one point or another. I then went onto talk about what resources I would be using when I was needing information about how to do certain things

when constructing the program in which I was quite easy to list off as most of these sources I've used beforehand. In all honesty most of these where used besides Codeacademy and Packt and the reason they were specified was I needed more sources to make all the sources fit a page.

I had no clue during the entire time I was making my project plan as I have barely any experience in making my Gantt chart so all of the times in between the deadlines were all just flung in just so that there something there and all of the things that are supposed to happen on those days didn't happen. Although the resource list is basically correct and what I used to do those tasks. Out of all of the inception phase this was the one I was weak due to the lack of experience of using project to create projects.

Elaboration Phase

Before I even started on designing the program what I done was explain what's been changed since the inception plan which was I changed from waterfall to agile, the resources I'm using which is MySQL and XAMPP which then changed back to UwAmp later on – reason is that XAMPP wasn't working properly on the college computers plus the fact that it is dependent on where XAMPP is installed - and the functional requirement of log-in as the employees and admin should be able to access two different menus which do different things. Later on, I've also had other changes in the other iterations which leads on to my weakness of forgetting things like how I forgot that there's meant to be a log-in.

With the techniques being used talked about the different design diagrams that I'll be making for the business model which is the back-end of the program and then the view model which is the GUI of the program – the business model will involve creating ERDs, use case diagrams + descriptions, sequence and activity diagrams and a class diagram and view model will involve wireframes and style guide. I knew what I needed for the most part as I would need to have an ERD of some sort for the database to understand the entities and then relationships behind the database plus using data modeler I can convert the ERD into an physical model which can then get be further converted into SQL code. I needed the wireframes and the style guide as I needed some kind of guideline on how the program will look as I would probably make look so inconsistent and unpleasant to the naked eye and could cause problems for people who have disabilities.

Before I created the diagrams I explained what will happen in each iteration of the design which for the first iteration I did the database and then the wireframes + style guide for the GUI of the design, second iteration was the creation of the Java application by creating CRC cards, class diagram, sequence and activity diagrams and the third phase of connecting the database with the application and then testing the application.

The first iteration was correct as I did do what was said, the second iteration I didn't create the class diagram or the activity diagram in this iteration and I forgot to create the CRC cards but that was due to not knowing how to layout the design for the class diagram and the third iteration was that's when I started actually creating the application, linking the database and then testing the application. This was due to how bad my time management was as I left quite a bit of the design done at the last minute like the activity and the class

diagram but for the class diagram I needed the application done due to me using an unfamiliar library which could have a unique way to represent in a class diagram and the activity diagrams due to how close the deadline was for handing in the elaboration plus construction phases.

First Iteration

The first thing I done for the first iteration was I did more detailed use cases based on what the admin can individually do and then what both the employee and the admin can do in this application and my assumption on how the purchase product use case will go. This was quite simple as I already had the information on what should happen the updated functional/non-functional requirements.

The next thing I did was the database design which involved me creating an ERD which is used to represent the tables, the tables' attributes and the relationships between those tables which then was converted into a physical model and then from this physical model it is then converted into SQL which is then imported into UwAmp into a database which was honestly the easiest thing I done in this whole project as it was just pretty much clicking an import button in the database and then clicking on the SQL file to import everything that was associated with the physical model.

The process of creating the ERD and converting the ERD into SQL was fairly easy to do but actually thinking about what tables will be made and what attributes those tables will have was quite difficult for me as at first I thought I had all the tables and attributes I needed in the first place but when I got further with the design I started to realise I needed more so Transaction Log, Stock Log and their attributes due to me not giving a good look through the requirements or the answers from the interview and missing out on possible tables or attributes for certain tables like phone number for employees.

After creating the database, I went onto creating the wireframes and the style guide of the application. The wireframes where made to give me a guide on how the application will look when constructing the application but was also made to give the client a look at how the layout of the application may look like when developed and I was able to create these wireframes pretty easily and didn't real.

However, I forgot to ask the client about the wireframes until I got to third Iteration due to poor communication. The style guide was made to show what colours and fonts will be used for the GUI so that people with disabilities like dyslexia could understand and be able to use the application and similar to the wireframes show what the style of the application may look like. Similar situation with the wireframes I forgot to show the client the style guide until the third iteration of the design.

Second Iteration

For the second iteration I have more changes since the first iteration like transaction log I mentioned earlier and everything that will need like its own display transaction log page and that employee has a username attribute. Turns out I never even used or developed a

username attribute in the actual database and application due to forgetfulness which was also the case further down the line as I then changed the username again to just using the employees id and first name put together as a username instead.

I then created the descriptions for the use cases which describes who will use this use case, how this use case will trigger, the pre-conditions of the use case, what happens after this use case and then a step-by-step basis on what will happen during the use case which is fairly helpful during development as I can see what these use cases should be able to do.

I had a good understanding on what will happen during each of these use cases as I already had a good starting point with the functional requirements as they had already showed a bit how it would work and it was fairly easy to describe what will happen and any potential errors that would happen.

I then went onto creating sequence diagrams which shows the objects interactions between each other in the use cases and what data is being pulled from the database. To be honest I only made these diagrams so that I had something with diagrams in my design and it didn't help in one bit at all although in some sort of strange way that's how the use cases actually worked in the application so I must have gotten lucky.

Third Iteration

With the third iteration I spoke about the changes that happened after the second iteration which let's just say I did the whole application so most of the third iteration is what I've missed out diagram wise so class diagram and activity diagrams.

What's changed however is that the database got a new table called stock log which needed to take in the stock changes an employee has made which is what the client wanted during the interview I had with them and just completely forgot to implement so I went back and designed the stock log in the ERD and then added it into the database.

The class diagram I created was based of the finalized application created which made it easier to design in a sense as I know for a fact what the classes are and what the relationships between them are. At the time I didn't make the class diagrams in the previous iteration as I didn't have a understanding on how I would layout the class diagram based on what I was using at the time which was hibernate because of hibernate being ORM – Object Relational Mapping – and how I would show that so I went and constructed the application and then from that application generated a class diagram using IntelliJ's class diagram creator and then cleaned up that class diagram removing what didn't need to be shown. This in a way was better as I already had a good understanding on how this application will work out and it guarantees that the class diagram is correct.

The activity diagrams where created to show what will happen during the use cases similar to a flow chart however I only created them as I wanted more diagrams in my design. I only made three due to time management as when I was making these I had three days till the deadline and I shifted focus onto other things I needed to do plus they weren't helping me in the slightest as these where made after the app was created and they were shite honestly and I would never use them at all if I did create them before the app.

Construction Phase

During the development of the application I flied through developing the actual programming of the application as I knew what I was doing for each of the use cases as I had my descriptions made and in a way the sequence diagrams helping me to what each use case will do. The problem was learning these new unfamiliar libraries which was hibernate and scene builder and how to use them correctly during the development of the program.

Firstly, let's talk about hibernate which was used to model the classes I made to the tables in the database to then pull data from the tables. It was a pain to actually use this at the start as I needed to learn how to configure hibernate on what database I'm using and then make configuration files for each class in my program as I was quite poor at doing XML which looking back wasn't that bad and has actually made me understand XML a little more.

After configuration of hibernate, actually using hibernate to start sessions and then pull the data from the data base wasn't hard at all and was quite easy as you would just start the session, run a query to get the data you want and then store the data and pass it back to wherever you need it. I had to use Hibernates' own SQL language called HQL – Hibernate Query Language – which was quite simple and very similar to MySQL and Oracle's SQL which I've used previously. I learnt most of this stuff from Youtube - (Brains, 2014) - and Tutorialspoint - (Tutorials Point, n.d.) - and from what I've noticed one of my strengths was picking up stuff really quickly as most of the configuration, creating sessions and using HQL only took me two days to actually develop.

Now let's talk about Scene Builder which was used to create the UI of the application. At first I hated scene builder as I didn't understand how the fuck I was meant to use it as all the tutorials where only using one scene but then I soon realised that I would just need to make controllers for each and every FXML file to program the functionality for every scene.

I had another problem with how to call the constructor I was using for the functionality and then use the data that is stored in the data like for checking if the user is an admin or employee by storing the job type in the controller class so what I initially had for every controller in the GUI's I created controller object but turns out I'm not using the same controller so what I needed to do is pass the controller created in log-in's GUI controller into wherever it needed to go so log-in to main menu then to any of the use cases which was figured out quite quickly. Creating the GUI was very easy to do as it was as simple as drag-and-drop the attributes onto the scene and then program what the action listeners will do. Again, from what I've seen is that my strength is that once I get an understanding on what to do, I can quickly do whatever I need to do and then replicated it.

I then made a user's guide on the requirements needed for running the application, the installation process and then how to use the application. The only thing I was unsure of was the requirements so I just used Java's basic requirements as everyone using the application will need to have Java. I made two guides – one for testers as they'll need to use UwAmp and the one for the client when the application is finished as it assumes that the database is on an external server.

The testing part of the application was easy as I just needed to create a test plan that checks the scenarios that part of the application that could happen like for example the database isn't available and making a questionnaire so that other people can test the program. This was easy due to me already understanding how to test a program from previous experience testing my own programs and I've made questionnaires already for programs.

Overall I'd say that my strengths are that I can create the diagrams and the application quite easily as I know how to and that I'm quite good at picking up things quite quickly like for example how I picked up how to use hibernate and scene builder with quite a good knowledge of both in just a couple of days after being introduced to them and that my weaknesses are time management as most of the design work could've been done a lot earlier than when it was actually done and if I did have better time management skills I could've done a lot more than that was currently given and that I'm quite forgetful so some of the things that I mentioned in either the inception phase like for example the data from the functional requirements specifically about the employee and the product isn't the same as the data in the actual database or the elaboration phase with like certain tables or attributes which leads onto more development on something that could've been done ages ago.

Recommendations for the Future

One recommendation is that there should be a back button on every page besides the main menu as when I was testing the application I was getting frustrated as I would accidently click on a button I wasn't wanting to click and I couldn't go back I would always need to either close down the application or go through with what I needed to do. It isn't that hard to code as well because it would be like how the go to main menu button works with the button going to the main menu associated with the employee. This is easy to do as it's just going back a scene and would cost next to nothing to implement.

Another recommendation is to give the admin more functions to do like for example adding new employees, adding products, updating products, adding new customers, removing customers, adding suppliers, removing suppliers, changing how the purchase product works, how transaction log stores its data, changing the product data with what it's supposed to have and more reports.

Adding products would be a good suggestion as there may be a situation when a customer may try to purchase a game that doesn't exist in the system so rather than the admin going to possibly a database manager to add these new products the admin can do it himself so that business can run as usual much quicker rather than the database manager going into the database and then adding the data. This is quite quick to do so the cost of doing this is low.

Updating product would be needed if we go down the route of having the admin being able add a product as the admin could possibly make a mistake inserting a new product so they would need to update the product in some way and rather than them going to an administrator the admin can just do it there on the application and also if a games price has

been changed then the admin can easily go and change the price with no troubles at all. This would take next to nothing in time and the price to implement it not that much.

Adding new customers would be necessary as if we talk about trade-ins the business needs to have some kind of way to keep data on a customer as if in the case of that person trading in the game has stolen that game then the business has the data on the person trading in the game to give to the law if necessary. Again, takes next to nothing to implement and would cost barely anything.

How I think this would work out is that it would happen at the register as what would happen is the employee would ask for their name and address plus proof of their address and proof of identity in which the employee will enter the customers details and then from there the register would send the information to the application which would add the customers details. I should suggest that the system should also take in the phone number of the customer so that if the scenario I mentioned above happens then the law will have some way to contact them. Again, takes next to nothing to implement and would cost barely anything.

Removing customers is necessary suggestion if the business is storing data on a customer due to the data protection act you would need to get rid of data after it's no longer needed so what should happen is that after a certain time has passed the application should remove the data that is been in the database longer than the specific time and should happen every time the system starts and when the system closes like for example if the shop is closed to guarantee that the data that should be held anymore is definitely gone. Again, takes next to nothing to implement and would cost barely anything.

Adding suppliers would be a good suggestion as it would be easier for the administrator to just add the supplier's information through the application and not through as I said previously the database administrator and asking them to do it as it would just be easier for the admin to quickly do it. Again, takes next to nothing to implement and would cost barely anything.

Removing suppliers is necessary suggestion due to maybe a supplier for certain products goes bankrupt the business wouldn't be able to get products anymore from that supplier or that the supplier has finished business with the current business so the admin can easily just remove the supplier information with just them entering the id of the supplier which will completely delete the data of that supplier. Again, takes next to nothing to implement and would cost barely anything.

Big recommendation is to further work on the purchase product as what I envisioned is that the customer goes to the employee at a register, the customer buys a product/s, the employee scans the product/s and then the total price is created, the customer pays the total price and then the information from that register should then be picked up automatically by the application and then puts the transaction info into the transaction log table. This could have the potential to take up the most time as you'd need to get people with experience developing applications with tills and possibly concurrency to automatically

do the purchase product in the background and because this could take up the most time it will cost the most and will probably be fairly high price.

Which leads onto the next recommendation is to change how transaction logs work because in its current state it only will store only one product that has been purchased rather than multiple so what should be changed is that the transaction log should have a recursive relationship as in the first product being purchased has no product foreign key but the other products being purchased that is part of this transaction should have the foreign key be the primary key of the first product being purchased so that the admin/employee can understand what products are related to the transaction.

How to work out the final cost as well will need to be changed as currently it'll only take in one products price which is wrong and should take in multiple if there are multiple products being purchased. What I'd suggest is that when the data is being taking in a variable is made to hold in the total and each time a product is being passed down the total is added up. However, when actually storing the total price using what I've mentioned above should the total price be only for the first product and the total price for each product afterwards linked to that transaction be blank or that the total price is on every single bit of the transaction, I'd think the former would make more sense but the later could be easier to understand products have been purchased during the transaction. This is probably be the second thing that will take up the most time, but it won't be that long to actually develop, and the cost won't be that much.

Another suggestion is to change the data being stored inside the product table as I forgot about the data I was supposed to store in the functional requirement linked to the search product use case and then what should also happen is to then change the columns inside the products displayed scene when the product has been searched so that the new data can then be displayed. Ideally, you'd probably want to know what kind of product it is or what kind of genre it is as well. Depending on how many products is already in the database it would take not that long and would cost not that much to do.

Another suggestion to add in is a log-off button on the main menu cause if the business is going to have multiple people working with the application at different points of time then you don't want to close the application each time to log-off so how I would do it is firstly close the session that is currently open unless the session has already been closed due to add stock or purchase product and then just call the log-in controller without passing in the main functionality controller as this is a new log-in. I just did this on my free time, and it would take fuck all too actually develop and implement and basically cost barley anything.

Another suggestion is to add more reports the admin can use as the application is supposed to have the products data involve genres, developers/publishers so I'd suggest checking for the most popular product based on the genre or the developer/publisher making it, having a report that would check for the most traded-in product as well, what products has the supplier given to the business. This would probably take quite a bit as you'd need to develop some kind of way to track the products being supplied by maybe having a link between stock log and suppliers as you can specify where the stock came from or have some way to

track what products have been traded-in which would be quite simple as the transaction log could have a column that is like a Boolean to see if the game has been taken in and then adding one to a stock basically the reverse of purchase product.

Another suggestion is to maybe switching from a Java application to an in-house website instead as from the project brief it seems like the Java application will only be used on one computer which could potentially be problematic as multiple people may want to use the application at the same time so a website that is only accessible through a possible intranet so that many people can access to get what they need at the same time and also prevents people from just taking a copy of the application from the computer making it more secure. It could also cause problems particularly at multiple people all accessing the data at the same time which could cause a possible server crash as multiple people are trying to access data from the database. I wouldn't go for this suggestion however if you don't have the money to actually get something like this going as the cost for this would probably be very high and would probably guessing at least of development of this system.

A big suggestion is to fix the errors that've been discovered through testing. One of the big errors was that when the database crashes or is not on the application wouldn't give and error message and then close the application. What I think was the cause of the problem is that the exception that was supposed to throw isn't the right exception being catched so every time the application tries to access the database It wouldn't display anything. This seems like lack of info on the exceptions of Hibernate so it wouldn't take that long to research and see the exceptions that've been pulled out and wouldn't cost that much to develop a fix.

I think this was caused due to changing how the session starts and ends due to how add stock and purchase product was made as it closes the session every time either stock was added or when a product has been purchased because before I made these changes the exception would be catched and then would give an error and then close the application. Honestly this could take a couple of hours of testing and would cost next to nothing to actually develop.

Another error was with the employee section as every time the application would display data be it the transaction logs or product data the back to main menu buttons for that bit wouldn't work as I guess it doesn't check if the job type is employee in the if/else statement or that an exception is happening but isn't being caught at all. This would cost next to nothing for developing a fix for this error plus it would take fuck all of time to fix the error.

A problem I've been having with the application that I suggest should be fixed is that it won't run on the computer I've been making this application on without being ran first using a .bat file to run the actual app but would be able to run on the college computers fine without the .bat file which I honestly have no fucking clue on why the fuck it does that. My best guess is that the JDK's I've used on my home computer and the college's computer probably are different versions of Java 8 which could cause conflictions on what version of Java you need. This could also take quite of time due to the fact that you'd need to have

multiple computers to test the application which would also be quite costly of actually obtaining these multiple computers to be tested on.

Modifications during the Project

What's changed during the elaboration part of the project is that the database got a new table called stock log which needed to take in the stock changes an employee has made which is what the client wanted during the interview I had with them and just completely forgot to implement so I went back and designed the stock log in the ERD and then added it into the database.

It was needed because the client specifically wanted to store every stock that had been added as there could've been stock accidently been added or been maliciously been added to spite the business so the admin can easily see what employee has added that stock to be able to find out who it was. The impact of it was that I had to go back and change the ERD which that could've been in the ERD originally as It was specified in the inception phase of the project and took time off from other diagrams, I could've made more of like the activity diagrams.

I have more changes like creating a new table for the transaction logs and everything that will need like its own display transaction log page as the use cases specifically ask for the admin and the employees to be able to see what transactions has happened on a specific date and that originally I didn't have a table at all in the ERD but further down the line I remembered that I needed to hold data about specific transactions that've happened so that people who want to refund their product can do so by checking the id on the receipt with the primary key of the transaction log. The impact it had was quite a significant change as I had to re-do relationships between certain entities and create a whole new entity and then re-create the database which all of this could've been prevented if I looked at the inception phase a little more than I did rather than wasting time on fixing something that should've been already in.

The username for the log-in changed so much so originally with the log-in using the full name of the employee which then lead onto me adding a username attribute to the ERD that will store the employees user name which is supposed to have the employees first initial and then the second name of the employee with their id at the very end all joined up which then lead onto me not even putting the attribute for the username in the table and then just using the id at the start and then the employees full first name joined together as the username that is in the application currently.

This is probably the thing that changed the most besides the ERD which looking back is quite funny as I really was clueless that I had already specified my username twice once in the functional requirement for the log-in and then again when discussing the second set of changes in the elaboration phase. Honestly this impacted a little bit as I had to re-do certain parts of the ERD which again could've saved time as it wasn't really necessary come to think it as I could've just checked the username by taking the first initial of the first name, the last name and id and then join that up and check to see if it matches what the user has entered.

Other changes where that I changed from waterfall to agile which waterfall meant that I had to do that certain stage before I could move on and couldn't move back a stage meaning that If I didn't create something in the design that I said I would do I couldn't move on until I do that design while agile allows me to go back and change what is needed like the design while I'm constructing the application. This had a massive impact as I was no longer forced to do one part of the project but could do multiple parts at the same time as develop the application and then create a class diagram based of the finished application.

Another change was switching from XAMPP to UwAmp as the reason is that XAMPP wasn't working properly on the college computers because it's the fact that it is dependent on where XAMPP is installed so because my XAMPP was installed on a USB at my home computer with a specific drive letter which was E: and then when I put it into the college computers the drive would become F: and when that happens XAMPP wouldn't find specific information it needed for it to work.

What I could've done is installed XAMPP on the college computers and then I could use it, but the problem is that I'm using two different versions of XAMPP which could cause errors trying to transfer databases over both. This didn't have a big impact as one it was really early in development so no time was lost and plus, I've had previous experience using UwAmp so I knew what I was doing with UwAmp anyhow and plus it didn't have any of those sorts of problems XAMPP was having either.

A big change happened with introduction of the functional requirement of log-in as the employees and admin should be able to access two different menus which do different things. This had a huge impact as I needed to add in a password attribute for when I was creating the ERD so that the password given from the employee can be stored to then be called when the log-in happens plus the fact that this will be the first thing either the employee or admin ever sees and the whole fiasco with the username as I've previously talked about. This was also the reason why I went and used the unfamiliar library of Spring Security as I needed some way to encrypt the employees password when storing the password using the database as you don't want the password to be plain text as hackers can see what the password is and with encryption they can't see anything. This was the biggest change that needed to happen.

Another big change happened where internally I switched from doing JavaFX without using scene builder to actually use scene builder because at the time of switching the deadline for both the elaboration and construction phases was coming up in at the time 6 days so I went fuck it and just started to use scene builder and the reason I picked scene builder is because I looked at it previously and it looked like windows builder which is the swing equivalent and I've used windows builder before and was really easy to use to build a GUI so I thought the same with scene builder as that would be my ticket for a quick and easy GUI for my application.

An unforeseen event happened during development of the project as on certain days the lecturers of Forth Valley college went on strike which prevented me from getting any communication I may of needed on one of those days with one of the lecturers and my

client or maybe needed one of the college's computers as I could've had malfunctions with my machines I have at home.

Knowledge and Skills, I've Learned

What I've gained from doing this project is how ORM works and improving my skills with XML and having understanding on where XML could be used in an Object-Orientated program and improving on my JavaFX in some form.

What I learned is that ORM can be really good but also quite tricky to use as when developing the project when I was at the elaboration phase I was looking at videos on what can be used besides JDBC because at the time I was originally going to use JDBC which is Java Database Connector which just connects the Java to a database to do queries to pull the information. However at the time I wanted to somehow represent the tables as objects in the program as I thought there must be some way to do something like that and I thought hypothetically you could probably pull data and set data using its getters and setters so I had a look around and found out about Java Persistence and looked at Spring Framework which I had a look at and it didn't fit what I wanted.

However through IntelliJ it had Hibernate on it and I google what It was and looked at it and it was what I wanted as It was doing what I envisioned plus looking at Youtube tutorials - (Brains, 2014) - and then saying to myself that this looks easier to use than JDBC on its own as it looked like just using the getters and setters of the object to pull data from the database and then vice versa but I looked further and saw that hibernate will cascade for you which means it will store any foreign data like for my project storing a customer's details with an address. Turns out not really as what I had to do was firstly create classes to represent every single table in the database and then configuration files for each class to then link the tables to the classes and then create the a configuration file for creating sessions and to define what JDBC to use to connect to the database which honestly I found an example of one on the internet and used it and changed to what I needed which was using the MySQL version as I was using MySQL for the database which were all created using XML which was quite surprising because I thought it was many to hold data but this showed me that it can be used for configurations which has now given me more ideas on how to specifically create projects.

The configs in hindsight wasn't as bad as I originally thought because it was quite easy to understand as to define a link between an attribute in a class to a table you'd just specify the name of the attribute in the class, the column name inside the table and then the type of data that will be pulled however the only problem was the relationships because at the time I didn't know how to represent the one-to-many but through some research online and a bit of help found on Tutorialspoint - (Tutorials Point, n.d.) - I figured out that to define a one-to-many so let's use employee for example employee has the one-to-many relationship to stock logs so what I need to do is to create a set with the data type of StockLogs as I need to take in all of the stock logs that the employee has been involved with so in the config for employee it would have a set tag specifically defining where the data will be coming from which is the stock logs table and then specify that the employee id is the foreign key for the

stock logs employee and in the stock log config it specifies that there is a relationship with employee that is one-to-many and this honestly took a while to really figure out because it got quite confusing but after getting it was quite simple afterwards.

I got to see the benefits of using hibernate as when I was trying to assign data to a database I could literally just use the setters of the object to set data so for example when adding a new stock log after stocks been added I just create a new StockLog object, assigned the data and then saved it into the session and committed which is quite quick rather than doing an insert query to the database through JDBC.

Another benefit is that all the data from the database can just be queried to then be put into an Collection which is quite useful plus you can set parameters inside the query which is hibernates own query language which is quite different than SQL as there's some operations in HQL that you can't do like date operations which was a pain when trying to get the most popular products from a specific time frame for a report I needed to specify the time being passed by taking in the current date of the machine using the product to then go back a certain time frame – week, month, year – and then create the query and give the query a parameter to then assign it to the date that has been taken back a certain timeframe.

However, from what I've noticed its quite slow to get data from a database as it could take a couple of seconds to log-in when I was testing the program and using it in general to see it would work but this could be due to my machine being a bit slow so that could be tested a bit further but overall I'd probably use ORM again if I was using database again with Java.

FXML was created because of scene builder which is JavaFX's equivalent to Swing's windows builder which creates a design that uses JavaFX as when I use scene builder to add buttons and such the FXML changes so that it will assign the properties of the button created with scene builder to the declared button in the controller class for the FXML which at the time confused the hell out of me because I didn't know if you would use one controller class for every single one of the FXML created or to just give every FXML file its own controller which I did because it made more sense at the time but looking back it duplicated a lot of code which could've been dealt with if it was in one controller class.

What improved with my JavaFX is that I got to use parts of JavaFX that I've never before or got to correctly implement attributes that I've used before but never gotten to use for example this was the first time I used TableView which was quite confusing at the time as how I created my controllers was through initData which would store the controller usually but when needed the data of a table from the database but with TableView I needed to create an observable list and then assign the data of the table to the observable list and then assign the columns of the table to the attributes of the object to display the table data and then assign the observable list to the TableView. The reason why I had to do it there is because if I tried it outside of that method it wouldn't work as the data from the table would just get erased as when the scene changes the data gets deleted.

The attribute I got use properly was the DatePicker which I've tried before hand in another project but it didn't work as I tried to pass the data back it wouldn't allow me to but for when the application takes in the users date to look for transactions I just get the date from

the DatePicker and then assign the date to a variable with a data type of LocalDate. However due to date being LocalDate this I needed to convert that date into a string to check if the date was a correct date and to match the date with the dates coming from the database.

This project could've been improved by a multitude of ways which I'll specify in the list down below:

- More people working on the project.
- Better Time Management.
- Being more furrow during development.

More people working could improve the process as much more stuff could've been done much quicker as multiple people could've been working on the design of the project like one person was working on the UI whist another would work on the database design which would quickly get to the construction phase faster. With the construction phase the developers could be working on different parts of the application at different times so one could be working on the log-in whilst another working on the GUI and controllers of the GUI and another working on one of the functional requirements.

Due to this other parts of the project would be able to have more work done so that for example the Purchase Product if had more people could've been a reality with multiple people working on a way to link a register to the application and another group could be working on the specific part in the application. This also shows that having multiple people can bring in knowledge from different areas of programming as someone could have knowledge on concurrency whilst another may have knowledge on database programming.

If I had better time management skills and didn't fuck around at home and doing nothing at all, a lot of the things I missed out on doing could have been fucking done like for example the rest of the activity diagrams would be done if I did a lot of things faster or didn't leave things nearer the deadline because honestly if I did have good time management skills I could've done this using waterfall instead of agile as I would be specifying what to do on what days and then doing them without any hesitation.

If I had been a lot more furrow and not be a forgetful fuck, I would have gotten a lot more things related to the inception phase with the functional/non-functional as for example the username for the employee and admin changes about three times during this whole project because I didn't look through the inception phase enough to realise my mistake which would make the project more authentic to what I specified in the inception phase.

What I've noticed is that during the construction I've made quite a bit of repeated code specifically about returning back to the main menu which could've been its own class where it would just take in the controller and then do the repeated code which would clean up the code making it more easier to understand and be much, much faster as it can be accessed wherever it is needed.

References

BDatech. (n.d.). *Typefaces for Dyslexia*. Retrieved from BDatech: https://bdatech.org/what-technology/typefaces-for-dyslexia/

CEX. (n.d.). *CEX Homepage*. Retrieved from CEX: https://uk.webuy.com/

Codeacademy. (n.d.). *Codeacademy Learn*. Retrieved from CodeAcademy: https://www.codecademy.com/learn

Echeung. (n.d.). *CRC Card Maker*. Retrieved from Echeung: https://echeung.me/crcmaker/

Game Retail Ltd. (n.d.). *Game Home Page*. Retrieved from Game: https://www.game.co.uk/

GeekForGeeks. (n.d.). *GeekforGeeks Homepage*. Retrieved from GeekforGeeks: https://www.geeksforgeeks.org/

GiveGoodUX. (n.d.). *Right and Wrong Way To Wireframes*. Retrieved from Give Good UX: https://www.givegoodux.com/the-right-and-wrong-way-to-wireframe/

Gluon. (n.d.). *Scene Builder*. Retrieved from Gluhon: https://gluonhq.com/products/scene-builder/

Google. (n.d.). *Youtube Homepage*. Retrieved from Youtube: https://www.youtube.com/

Hibernate. (n.d.). *Hibernate ORM*. Retrieved from Hibernate: http://hibernate.org/orm/

Leyawin Media Services Ltd. (n.d.). *Homepage*. Retrieved from Retro Games: https://www.retrogames.co.uk/

Material. (n.d.). *Colour Tool*. Retrieved from Material: https://material.io/tools/color/#!/

Oracle. (n.d.). *Java 8 API*. Retrieved from Oracle Docs: https://docs.oracle.com/javase/8/docs/api/

Oracle. (n.d.). *Oracle Academy Home*. Retrieved from Oracle Academy: http://ilearning.oracle.com/ilearn/en/learner/jsp/user_home.jsp

Packt. (n.d.). *Packt Homepage*. Retrieved from Packt Publisher: https://www.packtpub.com/

Spring. (n.d.). *Spring Security*. Retrieved from Spring: https://spring.io/projects/spring-security#overview

Sqllines. (n.d.). *SQL Converter*. Retrieved from Sqllines: http://www.sqlines.com/online

Toptal. (n.d.). *Colour Filter*. Retrieved from Toptal: https://www.toptal.com/designers/colorfilter/

Tutorialspoint. (n.d.). *Tutorialspoint Homepage*. Retrieved from Tutorialspoint: https://www.tutorialspoint.com/

Usabilla. (n.d.). *How To Design For Colour Blindness*. Retrieved from Usabilla: https://usabilla.com/blog/how-to-design-for-color-blindness/

W3Schools. (n.d.). *W3Schools Homepage*. Retrieved from W3Schools: https://www.w3schools.com/

Brains, J. (2014, 06 30). *Hibernate Tutorial*. Retrieved from Youtube: https://www.youtube.com/playlist?list=PL4AFF701184976B25

Tutorials Point. (n.d.). *Hibernate*. Retrieved from TutorialsPoint: https://www.tutorialspoint.com/hibernate/index.htm