Software development

Graded unit 2 scenario 2

*Cameron Morrison 569530*

Contents

[1 Inception 1](#_Toc65414941)

[1.1 Brief 1](#_Toc65414942)

[1.2 Interpretation 1](#_Toc65414943)

[1.2.1 Web application 1](#_Toc65414944)

[1.2.2 Java application 1](#_Toc65414945)

[1.3 Initial functional and non-function requirements 2](#_Toc65414946)

[1.4 Initialise top level use case model 2](#_Toc65414947)

[1.5 Additional information from client 3](#_Toc65414948)

[1.5.1 Questions 3](#_Toc65414949)

[1.5.2 Response 3](#_Toc65414950)

[1.5.3 Revised function and non-function requirements 4](#_Toc65414951)

[1.5.4 Revised top Level use case diagram 5](#_Toc65414952)

[1.5.5 Actors 6](#_Toc65414953)

[1.5.6 Use cases 6](#_Toc65414954)

[1.6 Aims of the project assignment 9](#_Toc65414955)

[1.7 Resources and materials 9](#_Toc65414956)

[1.7.1 Integrated development environment 9](#_Toc65414957)

[1.7.2 Hardware expenses 9](#_Toc65414958)

[1.7.3 Cloud storage 10](#_Toc65414959)

[1.7.4 Resources 10](#_Toc65414960)

[1.7.5 Research 11](#_Toc65414961)

[1.8 Business model 12](#_Toc65414962)

[1.8.1 Business needs, rules, and assumptions 12](#_Toc65414963)

[1.8.2 Determining entities, attributes, and UIDs 13](#_Toc65414964)

[1.8.3 Top level entity relationship diagram 13](#_Toc65414965)

[1.8.4 Revised entity relationship diagram 16](#_Toc65414966)

[1.8.5 Data mapping 16](#_Toc65414967)

[1.8.6 Class diagram 21](#_Toc65414968)

[1.8.7 Sequence diagram 22](#_Toc65414969)

[1.8.8 Activity diagram 23](#_Toc65414970)

[1.9 View model 24](#_Toc65414971)

[1.9.1 User interface designs 24](#_Toc65414972)

[1.9.2 Colours, fonts, and accessibility 31](#_Toc65414973)

[1.9.3 Binding model 33](#_Toc65414974)

[2 Bibliography 37](#_Toc65414975)

Table of Figures

# Inception

I have chosen scenario two for my graded unit project. I have rewritten the brief to make it more readable by eliminating wording unrelated to the design of the program.

## 1.1 Brief

The required application must cover sales invoicing and stock control. It should produce an invoice for each purchase and update the available stock. There must be screen and paper reports covering daily, weekly, monthly, quarterly, and annual sales. As well as VAT analysis, stock turnover and profitability.

Stock reports should include reorder and stock outages. On screen display of customer, product, supplier, and invoice information including transaction logs should be available with an option to print.

Application’s user interface can be achieved through designing the program to run in the console or on a graphic user interface. Application should run on windows XP or newer or Linux or Mac OS X. The use of a database management system is optional.

## 1.2 Interpretation

My initial understanding of the brief is that the client wants a system that controls the operation management of inventory and monitors sales. There are two ways I can imagine designing this program.

### 1.2.1 Web application

The website could be built using HTML, CSS, JavaScript, and PHP. The PHP server-side language would allow for an easy, secure retrieval of information from a database. The use of CSS and JavaScript would give the program a pleasing aesthetic. The benefits of a website is that its portable, every device with internet would be able to access it. However, as we are dealing with client’s information, I would imagine the website needs to utilise PHP for a secure login process.

### 1.2.2 Java application

Another option for designing the application would be to use an object-oriented approach using the popular programming language Java. Java is a portable programming language that prioritises features like security, it is also a statically typed language making its execution speeds faster than others which would be fantastic for an invoicing system. Based on these two key reasons alone: security and speed. I have decided to create my program as an application constructed with Java instead of a web application.

## 1.3 Initial functional and non-function requirements

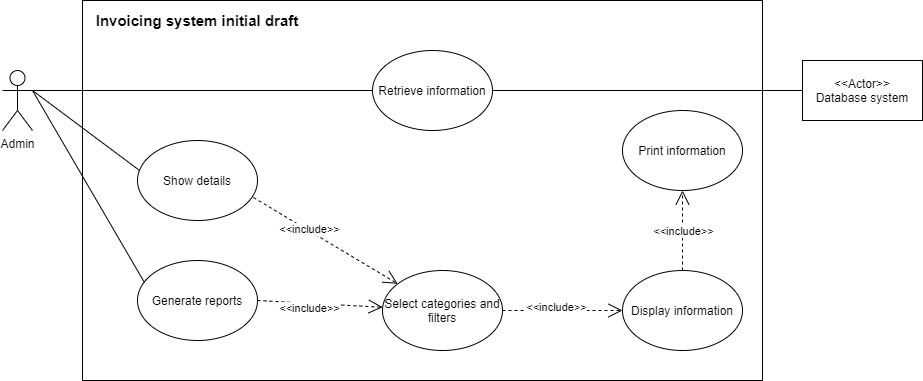
Functional requirements

* Must create an invoice for every purchase.
* Must update stock quantity in real time.
* Must be able to create sales reports (daily, weekly, monthly, quarterly, annually).
* Must be able to create VAT analysis, stock turnover and profitability reports.
* Must be able to create stock reorder and outage reports.
* Must display customer, product, supplier, and invoice information including transaction logs.
* Must be able to print information.
* System must be a console or graphic user interface-based application.

Non-functional requirements

* Global constraints such as budget or time scale not specified in the brief. Further clarification with client is required.
* System should be reliable and secure as managing finance, stock and client’s information carelessly could result in breaches of legislation.
* System could implement data storage, but it is not required.
* System should run on one of the specified operating systems. Windows, Linux, or Mac OS X.

## 1.4 Initialise top level use case model



Here is the initial use case model based off the limited information gained from the brief. Without clarification I assume the system is to monitor sales, stock and generate reports. There is no information specifically addressing how the orders or products are entered into the system. Perhaps the application is purely to filter, display and print reports on the information stored in the database. With clarification from the client, we can better understand the program. I will revisit the use case diagram in detail after our first meeting.

## 1.5 Additional information from client

### 1.5.1 Questions

Here are some clarification questions for the client I created after reading the brief, to ensure the client and I share the same vision for the outcome of the invoicing system.

1. What is the time and budget constraints for the program?
2. Does the client provide stock, supplier, and previous transaction data?
3. What country will the system be designed for? So, we can determine the correct VAT rate and currency?
4. What does the client want the stock reorder and outage reports to look like/achieve, will it show supplier information?
5. Can the user create hard copies of everything?
6. Do you want the program to do be password protected as it includes sensitive information?
7. If the choice of a data storage using a database system is optional, how would you store the data? – If storing the data is not needed then how do you initialize the program?
8. How does the system get new transaction information? Is it entered manually through the system? If so, can the client purchase multiple items at once?
9. Does the client want a way of updating stock quantity, price?
10. Can the client add new products?
11. If I did make a graphical user interface, does the client have preferred colours or navigation style for the applications design?

### 1.5.2 Response

We held out first meeting on the 27th of January 2021 and the client’s responses are listed below, cross referenced to the questions by numeral formatting.

1. Project needs to be finished preferably around the 3rd of May and the budget for the program is £25,000.
2. Stock, client, and previous transaction data should be fake dummy data. The client will not supply this.
3. System will be designed for the United Kingdom initially. Meaning the currency will be the pound and the VAT rate will be 20%.
4. Reports should try to be minimalistic. Stock level and transaction reports should allow date ranges. A reorder report should show all products with 5 or less items remaining. Any product with 5 or less stock should be made obvious. Maximum quantity of stock for an item is 20.
5. Hard copies to be included in a further iteration if possible. If the detail is printed to screen or saved somewhere the client is satisfied.
6. System should be password protected. The system will have two admins with separate logins.
7. Client would prefer database storage rather than a data structure.
8. New purchases must be entered manually through the system each time.
9. Admin must be able to update stock price and quantity easily.
10. Admin can add a new product through the system.
11. The client would like me to design multiple different appearances for the graphical user interface, taking in one consideration for visual impairment and return to her on a later date to discuss the design.

### 1.5.3 Revised function and non-function requirements

After the meeting with the client, I have decided to reconstruct the requirements based on the new information to avoid confusion when designing the program.

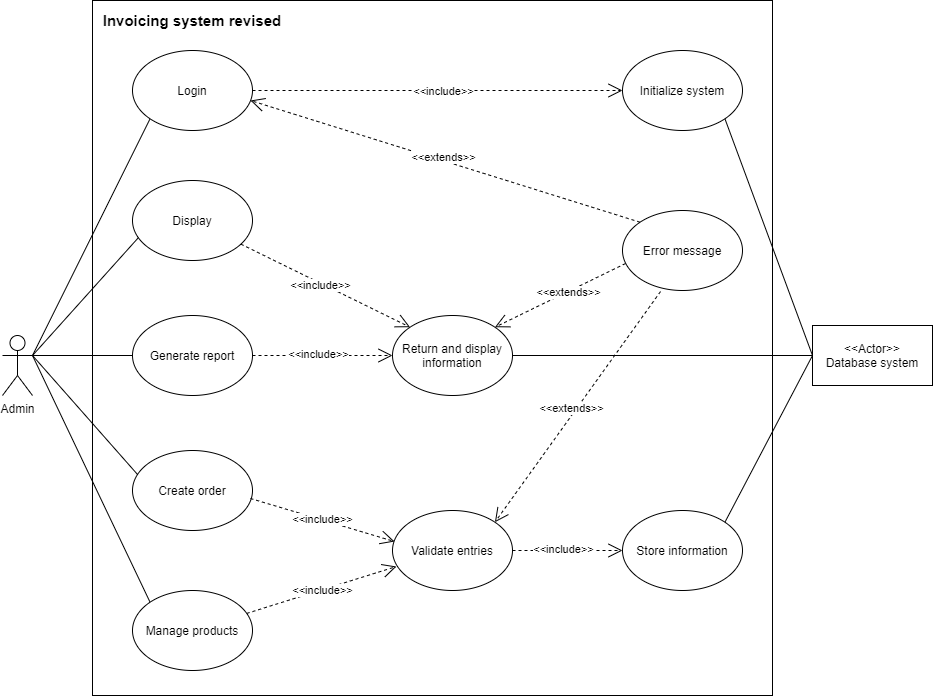
Functional requirements

* Purchases must be entered manually into the system.
* System must create an invoice for every purchase.
* Admin must be able to generate sales reports (daily, weekly, monthly, quarterly, annually).
* Admin must be able to generate VAT analysis, stock turnover and profitability reports.
* Admin must be able to create stock reorder and outage reports.
* System must display customer, product, and invoice information including transaction logs.
* System must allow admin to be able to filter report by date ranges.
* System must be a console or graphic user interface-based application.
* System must be password protected (two unique logins).
* System must be designed for the United Kingdom. Design the VAT rates and currency accordingly.
* System must ensure maximum quantity for an item is 20 and minimum is 5.
* System must use database storage system.
* Admin must be able to update item price and stock quantity.
* Admin must be able to list new products.
* System must highlight all products with less than 5 items remaining in stock reports.

Non-functional requirements

* The project has a budget of £25,000.
* Project must be completed before the 3rd of May 2021.
* Design should take at least one visual impairment into account.
* System should be reliable and secure as managing finance, stock and client’s information carelessly could result in breaches of legislation.
* In further iterations the client would like an option to print but if the details are saved or printed to screen, then the client is satisfied.
* System should run on one of the specified operating systems. Windows, Linux, or Mac OS X.

### 1.5.4 Revised top Level use case diagram



I have made many changes from the initial use case diagram now that information has been clarified by my meeting with the client. All top-level use cases have a precondition that the login use case was valid and the system was initialized. Logging into the system will load information from the database into the program and then allow the administrator to display, query and manage the data in the program. If the login or any information entered the system are invalid, then an extend clause invokes the error message use case to inform the user.

### 1.5.5 Actors

There are two actors. The administrator, the human who signs in then interacts with the program. Then there is the database system that retrieves and stores information.

### 1.5.6 Use cases

1. Login.
2. Initialize system.
3. Display.
4. Generate report.
5. Return and display information.
6. Create order.
7. Manage products.
8. Validate entries.
9. Error message.
10. Store information

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use case title** | **1. Login.** | **2. initialise system.** | **3. Display.** | **4. Generate report.** | **5. Return and display information.** |
| **Description** | Username and password for two unique admins to gain access to the system. | Information is loaded into the system from a database once admin is identified. | Admin chooses what information to display, with optional filters. | Admin chooses what information to generate a report on with optional filters. | System searches information for criteria then returns anything that matches the admins request. |
| **Actors** | Administrator, database system. | Database. | Administrator. | Administrator | Administrator, database. |
| **Pre-conditions** | N/A | Administrator successfully logged in. | Use case 2. | Use case 2. | Use case 2 & (3 or 4). |
| **Trigger event** | Programs start up. | Use case 1 is successful. | Administrator presses button. | Administrator presses button. | Use case 3 or 4. |
| **Post-conditions** | If entry is valid then use case 2 must be called to load information into the system. | Information is loaded into system from database or a new database is created. | System filters and displays entries matching the criteria. | User chose what to report to generate and applied filters. System displayed that information. | Any information found will be returned. |
| **Extension flow** | If entry is invalid, then use case 9 is called to display an error message. | N/A | N/A | N/A | If nothing is found, then use case 9 is called. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Use case title** | **6. Create order.** | **7. Manage products.** | **8. Validate entries.** | **9. Error messages.** | **10. Store information.** |
| **Description** | Admin fills out a form manually with client and purchase information. | Admin chooses whether they would like to create or modify a product then fills out a form. | System validates all new information the admin wants to store and checks it is in the correct format before storing it or displaying an error message. | An error message is displayed to screen to inform the admin what they are trying to do is not valid. | Business logic stores new validated information to the database once it is entered. |
| **Actors** | Administrator. | Administrator. | N/A internal logic. | N/A internal logic. | Database system. |
| **Pre-conditions** | Use case 2. | Use case 2. | Use case 2. | Something has been entered into the system by admin. | Use case 2 and 8. |
| **Trigger event** | Administrator presses a button. | Administrator presses a button. | Use case 6 or 7. | Use case 1, 5 or 8. | Information validated in use case 8 was in the correct format. |
| **Post-conditions** | Information is entered and sent to use case 8 to be validated. | Products are found or created, and new information is sent to use case 8 to be validated. | Information is validated and if there is an error use case 9 is called. Else use case 10 is called to save new information. | Error message is sent to screen for the administrator. | Information is stored to the database and the database is then closed. Boolean value confirms success or failure. |
| **Extension flow** | N/A. | N/A. | Use case 9 if entries are invalid. | N/A. | N/A. |

## 1.6 Aims of the project assignment

I am aiming to create a financial monitoring application in Java for my client. The application will have two logins, one for each Admin. The application will have a button to display information on clients, products, and invoices. There will also be a button to generate reports on sales, inventory stock and transactions. The reports will allow date filters.   
The program will allow the admin to enter a new order. The admin can also update product information as well as adding new products. Any updates or new entries into the system will be validated and stored in the database system.

## 1.7 Resources and materials

The client has graciously allocated us £25,000 funding for the construction of the program. The funding includes all possible expenses such as integrated development environments, hardware for computational power, secure cloud storage, resource wages, and research.

### 1.7.1 Integrated development environment

I have chosen to use Eclipse, a free integrated development environment that supports Java; the language I have chosen to construct the system in. Eclipse provides a great variety of developer tools and being non-commercial, it will help me allocate more funding towards any unexpected expenses.

### 1.7.2 Hardware expenses

The client has agreed that the budget can be spent on any expenses I would deem relevant to the construction of the program. I would like to purchase hardware to ensure the development of the program is moving at the most efficient speed possible and saved appropriately on physical storage.

I would like to purchase a computer that contains an [Intel i7-10700K](https://www.currys.co.uk/gbuk/computing-accessories/components-upgrades/processors/intel-core-i7-10700k-unlocked-processor-10207653-pdt.html) central processing unit as it provides excellent single core performance which is crucial for work in software development as it ensures the code when executing the program will run faster. It also has eight cores that provide superb multicore processing which will make the integrated development environment run smoothly. It is also a lot cheaper than others in the same performance scale sitting at a much lower price of £340.

In our modern times, 16GB of random-access memory (RAM) is considered the new minimum and the sweet spot for running multiple applications that require heavy computation is 32GB of RAM. RAM is a super-fast temporary data storage space that open applications utilise, and I would like to incorporate this quantity into my machine.

For storage I would like to store my operating system, development environments, invoicing system, and other applications on a solid-state drive (SSD). An SSD is more expensive than a typical hard drive but provides better speeds for transferring data which will aid the time it takes to load and save the program. I would like to purchase a 1TB solid state.

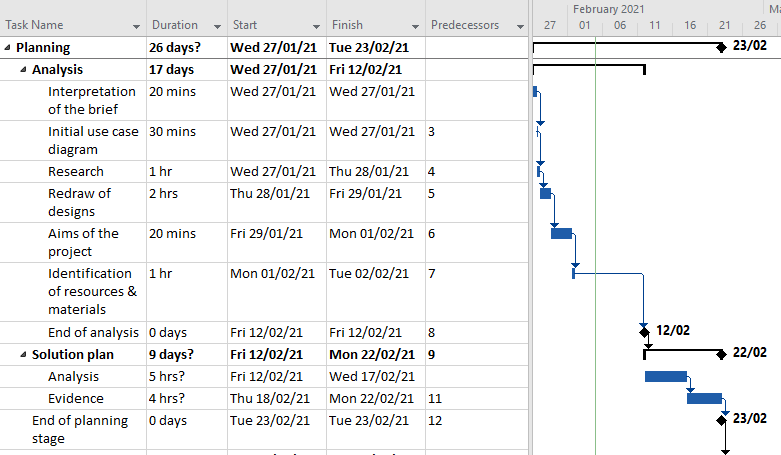
The three mentioned pieces of hardware are the most important for a fast and responsive development environment. There are of course other expenses in a computer such as components like power supply and motherboard. I can calculate this would roughly cost a combined estimate of £1000.

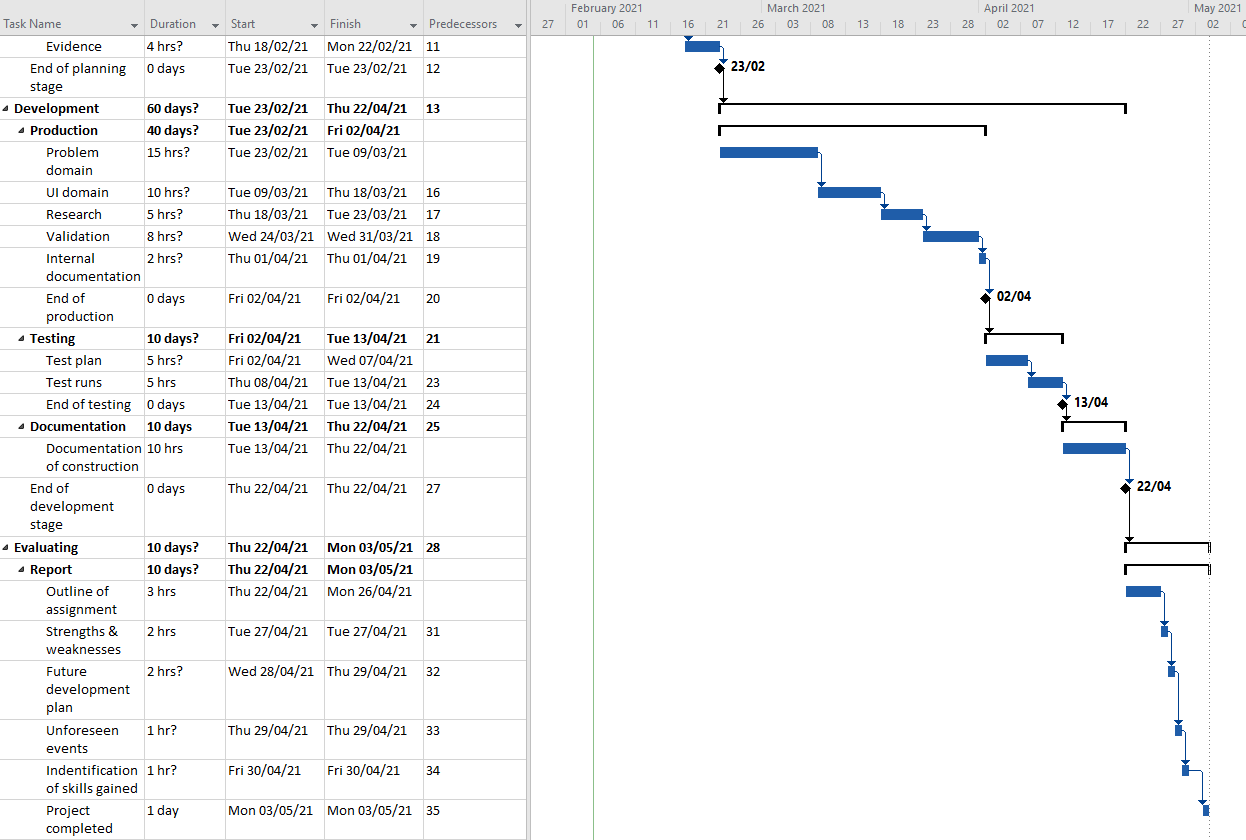
### 1.7.3 Cloud storage

While the application will be stored on my machine. It is crucially important that all programs of importance are backed up consistently to avoid any potential data loss. I would like to use the popular distributed version-control system called [GitHub](https://github.com/), a free service popular amongst programmers. Its goals are to increase collaboration and data integrity. After making any progress on the invoicing system I can push the newest version to [GitHub](https://github.com/), so it is safely backed-up to the cloud. Being a free service, it will help save further costs on the project.

### 1.7.4 Resources

I have created a project plan in Microsoft project to visualise my timeline and keep track of the sections I need to complete. I plan to do one hour of work every Monday, Wednesday, and Friday. I also plan on doing two hours of work every Tuesday and Thursday. The reason I chose these times is that it would keep my progress consistent. I have more free time to focus on the project on Tuesdays and Thursdays that is why I will work twice as hard on those days. By my calculations, allocating a generous amount of time to each task, I will complete the project before the 2nd of May. I am currently ahead of estimated schedule, which will allow me to free up more time for the construction phase.





### 1.7.5 Research

There are many free and informative websites online such as [W3Schools](https://www.w3schools.com/java/default.asp), [Stack Overflow](https://stackoverflow.com/) and the [Java 8 docs](https://docs.oracle.com/javase/8/docs/api/) that could be used for research purposes during the construction of the program. Research will not be allocated any of the projects budget. The Java 8 docs by oracle have information on all built in libraries that can be imported. It contains information about the methods and their uses. Stack overflow is an online forum for developers where you can find the answer to most of the questions you require an answer for. It is a great free resource for solving problems others have previously encountered. W3Schools is the world’s largest programming tutorial site that includes Java.

I also intend to work closely with the client. Showing them my progress when it comes to designs. The client will be a very important piece in my research as the program is being designed for them. Getting their feedback of my ideas, designs and programs usability will be key in ensuring the outcome of the program makes the client satisfied.

## 1.8 Business model

### 1.8.1 Business needs, rules, and assumptions

The invoicing system will need to store information for the customer, the product supplier and the staff member who enters the sale. It will also need to store information on the product, transaction, shipping, and address.

**Assumptions**

1. Customer will be entering information using the English language.
2. Customer can create as many purchases as they want.
3. Customers can have the same name but must have different emails to create a different customer ID.
4. If a new order shares an email with a previous customer, they will be counted as the same person and no new ID will be assigned.
5. Product can be shipped to any country.

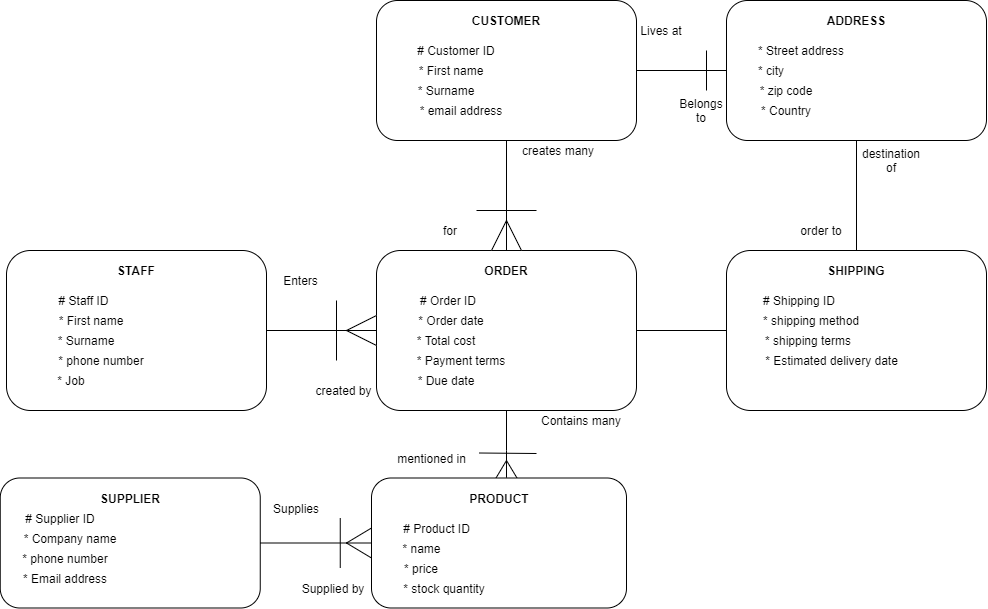
**Business rules**

1. Address cannot exist without a customer.
2. Customers can only have only one address.
3. Customer may have one or more orders.
4. Order can only belong to one customer.
5. Order can only be shipped to one address
6. Order must have at least one product.
7. Order must contain a minimum of one product.
8. Order cannot exist without a customer.
9. Order cannot exist without an invoice item.
10. Order cannot be created without a staff’s interaction.
11. Product must have at least one quantity in stock before it can be sold.
12. Product cannot exist without a supplier.
13. Product quantity cannot exceed 20.
14. Payment is provided by the due date if stated, by the customer.
15. Supplier can have multiple products, but a product can only have one supplier.
16. Shipping cannot exist without an order.

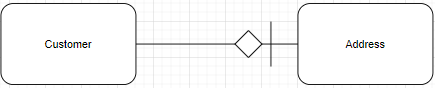
### 1.8.2 Determining entities, attributes, and UIDs

|  |  |  |  |
| --- | --- | --- | --- |
| Customer | Staff | Order | address |
| customer id (PK) | staff id (PK) | order id (PK) | Customer ID (FK) |
| First name | First name | Customer id (FK) | Street address |
| surname | surname | Staff id (FK) | city |
| Email address | phone number | order date | Zip code |
|  | job | Total cost | Country |
|  |  | payment terms |  |
|  |  | Due date |  |
| Invoice item | product | shipping | supplier |
| Invoice id (FK) | Product id (PK) | Shipping id (PK) | Supplier id (PK) |
| Product id (FK) | Supplier id (FK) | Invoice id (FK) | Company name |
| quantity | Product name | shipping method | phone number |
| discount | product price | shipping terms | Email address |
| Item description | Product quantity | Est delivery date |  |

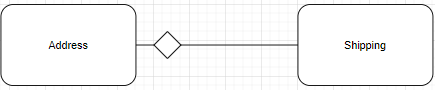
### 1.8.3 Top level entity relationship diagram



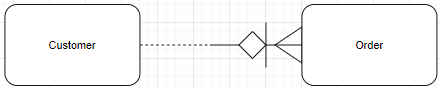
**Identifying relationship cardinality, optionality, and transferability**

****

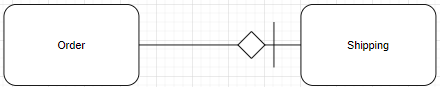
Customer will have a 1:1 relationship with address. For our system it will be non-transferable as an update address option was not required or specified as part of the application. It is mandatory as a customer must have an address to make a purchase. The address entity also obtains the customer ID through a barred relationship so the owner of the address can be identified.

****

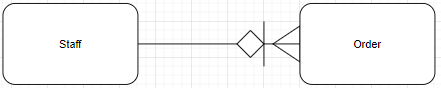
Address will be a 1:1 relationship with shipping. The address is non transferable as address is not changeable and once an item is shipped it is very difficult to change its destination. It is mandatory because when an order is placed it must be sent to the customer.

****

Customer has a one-to-many relationship with order, meaning they can place more than one order. It has also optional for a customer to create an order however order must belong to a customer. An order is not transferable, once it is placed it cannot be a different customers order.

****

Order has a 1:1 mandatory relationship with shipping because an order must be sent out for delivery once ordered and a set of shipping instructions must belong to an order. Shipping instructions are non-transferable and inherit the primary key of the order entity to help identify what order the shipping instructions are for.

****

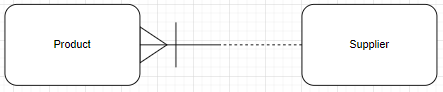
The staff entity has a mandatory one to many relationships with order. An order cannot be placed without the staff’s manual input. A staff will enter many inputs into the system and an order once placed cannot be transferred to a different staff member.

****

Order has a one-to-many barred relationship with invoice item. An order does not have to contain an invoice item, that is optional. However, an invoice item must be included in an order. An order can contain one or multiple items.

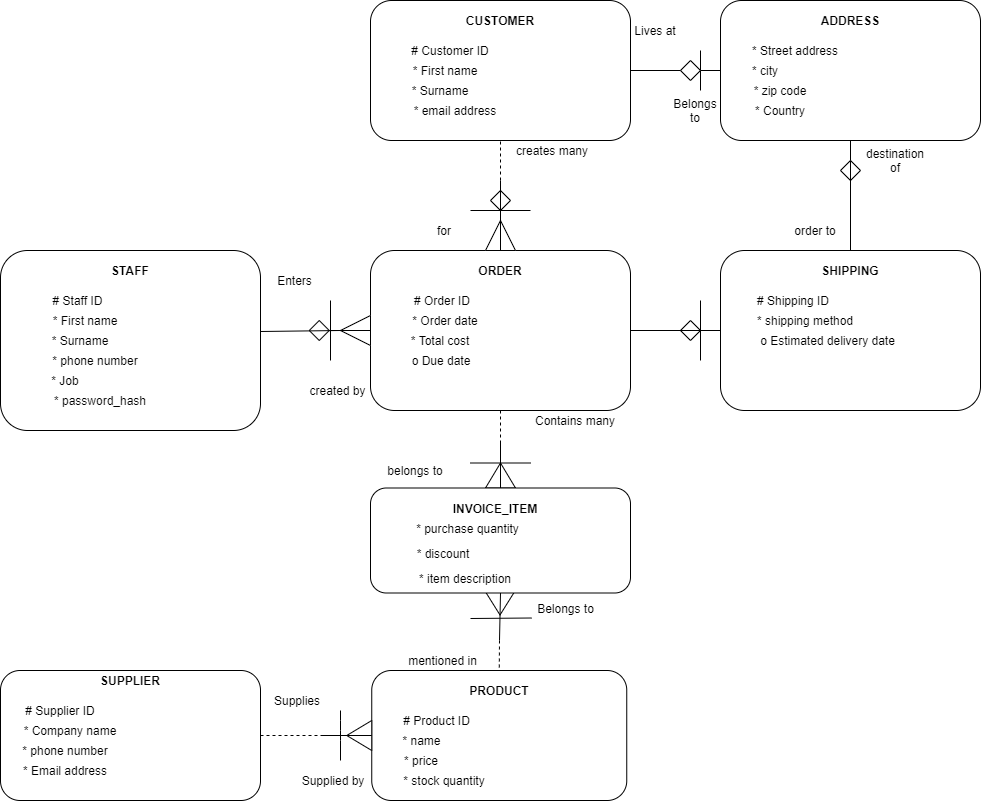
****

One or many invoice items must belong to a product. A product may be one or more invoice items. Invoice item is a bared relationship because it must contain the product ID to correctly identify what products are being purchased.

****

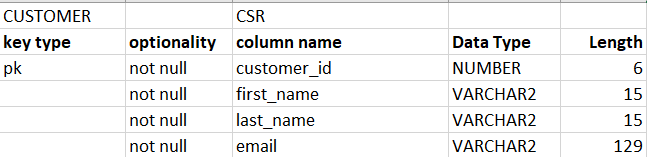
Many products must be supplied by a supplier. A supplier may supply one or more products. The relationship is barred as supplier ID will be needed to identify who supplies the products.

### 1.8.4 Revised entity relationship diagram

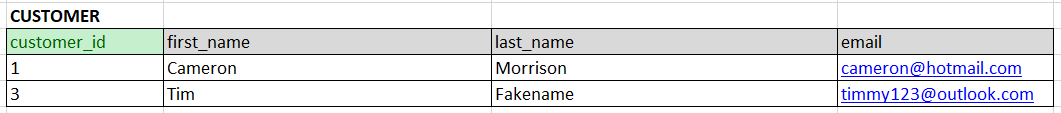


Here I have changed the optionality and transferability of the initial diagram. I have also added a password hash field to staff to securely store the staff’s password which will be used validate the login form. I have also added an intersection entity called invoice item which will allow order to contain multiple different purchases, the reason I did not just make a one-to-many relationship direct to product is because products could be discounted or purchased in bulk. I figured it did not make sense to contain this information in order as there could be multiple different values. I have also changed estimated delivery date and payment due date to optional.

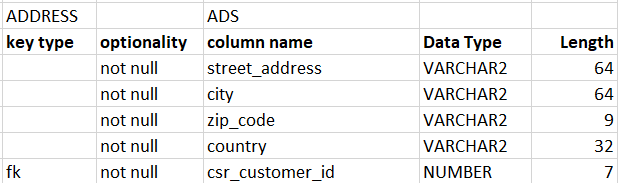
### 1.8.5 Data mapping

****

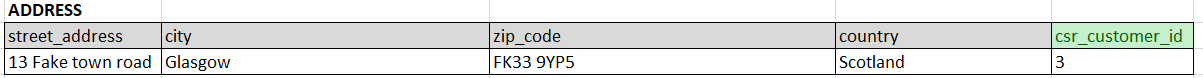
Here we have the customer table. Customer id is the primary identifier, this will be autogenerated by the program when a new customer is entered. We also have fields for names and an email address. All attributes are VARCHAR2 except for id which is a number. I have chosen to make the customer ID 6 in length as it allows for just under 1 million customers. I have decided to allocate a length of 15 characters to both first name and last name. For the email I have allocated a gracious 129 characters which will allow 64 characters for both username and domain name with 1 character for the at sign.

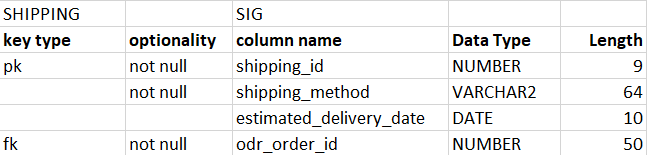
****

Here is sample data of what should be expected. First name and last name are both lowercase with the first letter capitalised. I can add external programming to make sure the information is in the correct format when saved to the database.

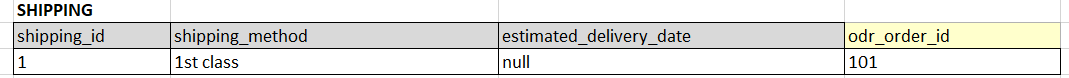
****

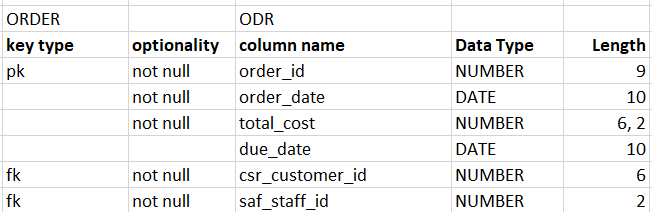
All the attributes in the address table are mandatory optionality. All are VARCHAR2 except the foreign key which is the id from the customer entity. For the street address and city name I have chosen a generous 64 characters. Zip codes in the UK are a maximum of [9 characters](https://en.wikipedia.org/wiki/Postcodes_in_the_United_Kingdom#:~:text=The%20postcodes%20are%20alphanumeric%2C%20and,and%20the%20inward%20code%20respectively.) in length (including the space).

****

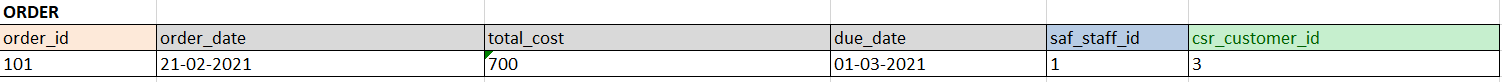
****

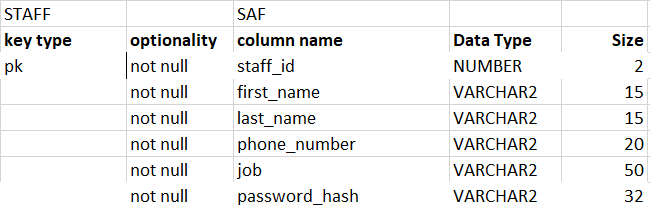
I have decided to make the length of the shipping ID 9, to allow for almost 10 million orders to be shipped. All the attributes are mandatory except estimated delivery date which can be null. This attribute is the first DATE type I have in my mapping document; I have allowed a length of 10 so the format can be 01-01-2021 for example.

****

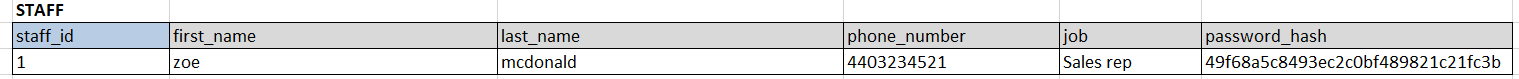
****

For the order table we have an ID, a date, total cost, due date of the payment and the ID of both the staff and customer as foreign keys. All attributes are mandatory except due date of the payment. The total cost attribute has 6,2 length. Meaning that it can be 6 digits with two decimal places.

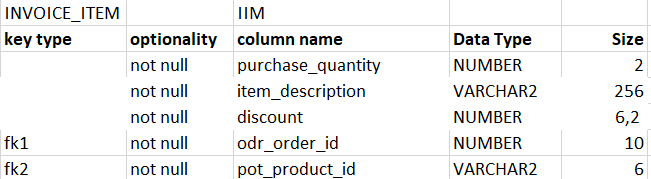
****

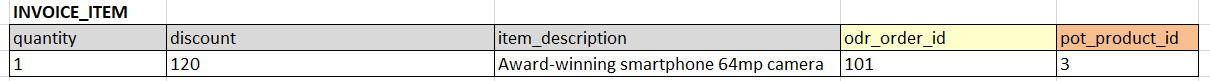
****

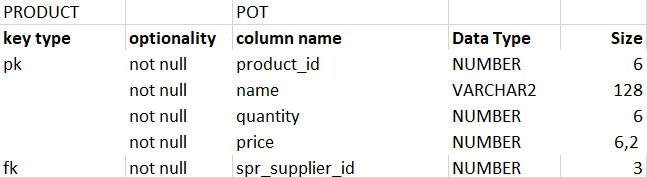
For staff ID I have set the length to two digits as the client specified there are currently only two employees. I left room for more employees to maintain future scalability. Staff also have a phone number attribute so that the customers can contact the staff and the staff can interact with the suppliers; I have given this a size of 20 characters. I have given the password hash attribute a size of 32 as it is a secure 32-character hash value I will be generating.

****

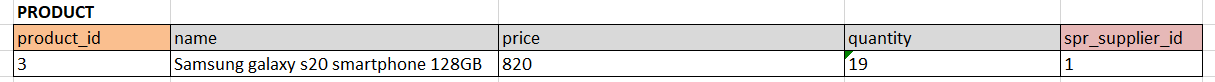
Hash values cannot be decrypted making it very secure to store in a database.

Invoice item is an intersection entity between order and product, hence the two foreign keys. All columns are mandatory. I have set the purchase quantity to a length of 2 as 20 is the maximum stock a product can have. For item description I have decided to allow 256 characters, as the description will be displayed on the invoice it would be silly to have the length any longer. Discount has a size allowing for two decimal places as I plan to have the discount price as the individual products total cost for buying one or more quantity.

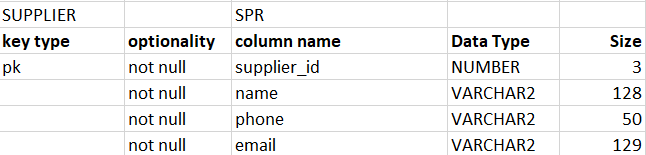
****

****

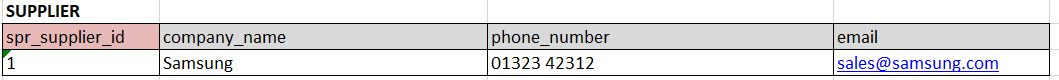
All the columns in the product table are mandatory, we have the foreign key of the ID from supplier. Price is again modelled as a number data type with 2 decimal places. For the product name I have allocated 128 characters as the name can have a lot of description in it.

****

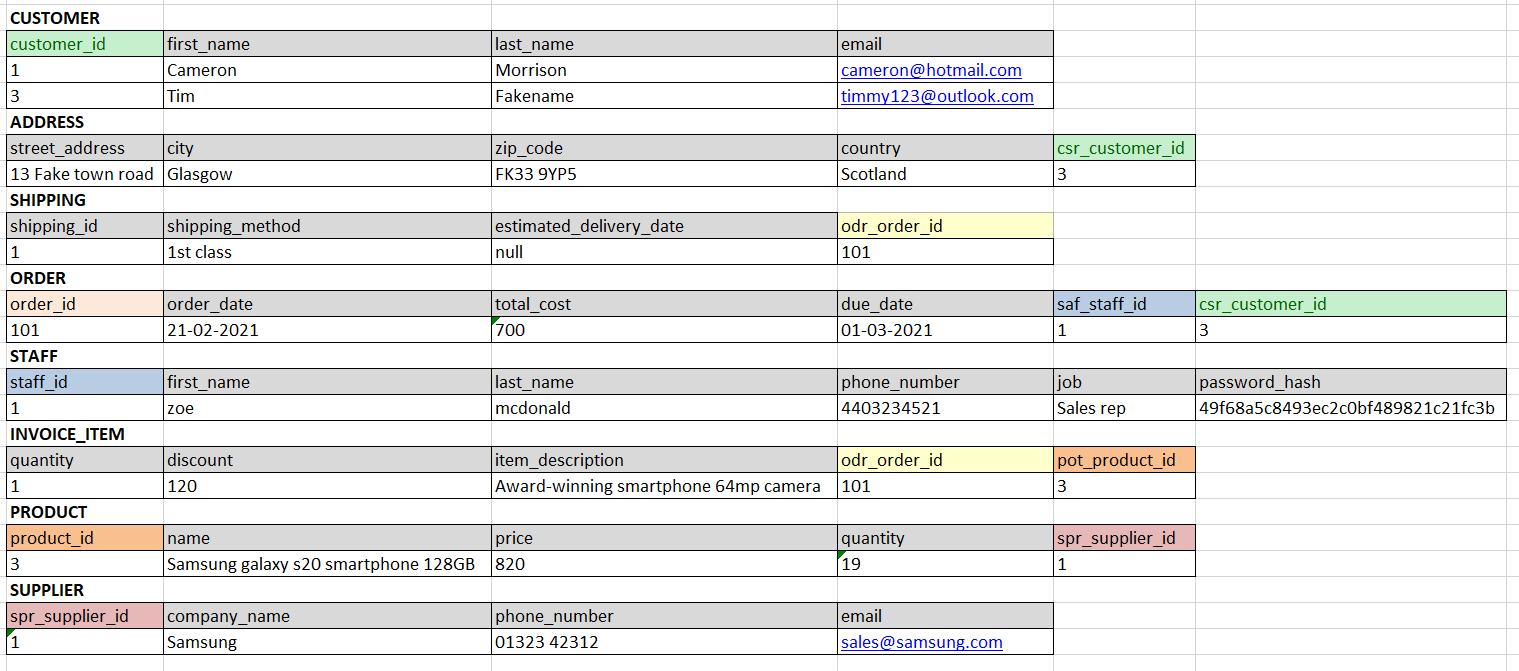
For this example, I used Samsung as an example of how the name of a product can be quite large. Sometimes the features are mentioned.

****

In the supplier table all the columns will be mandatory. Supplier ID is a size of 3, initially allowing for up to 999 suppliers. For the name I have allowed 128 characters. The email will be 64 characters for both the user and domain name with one character for the at sign.

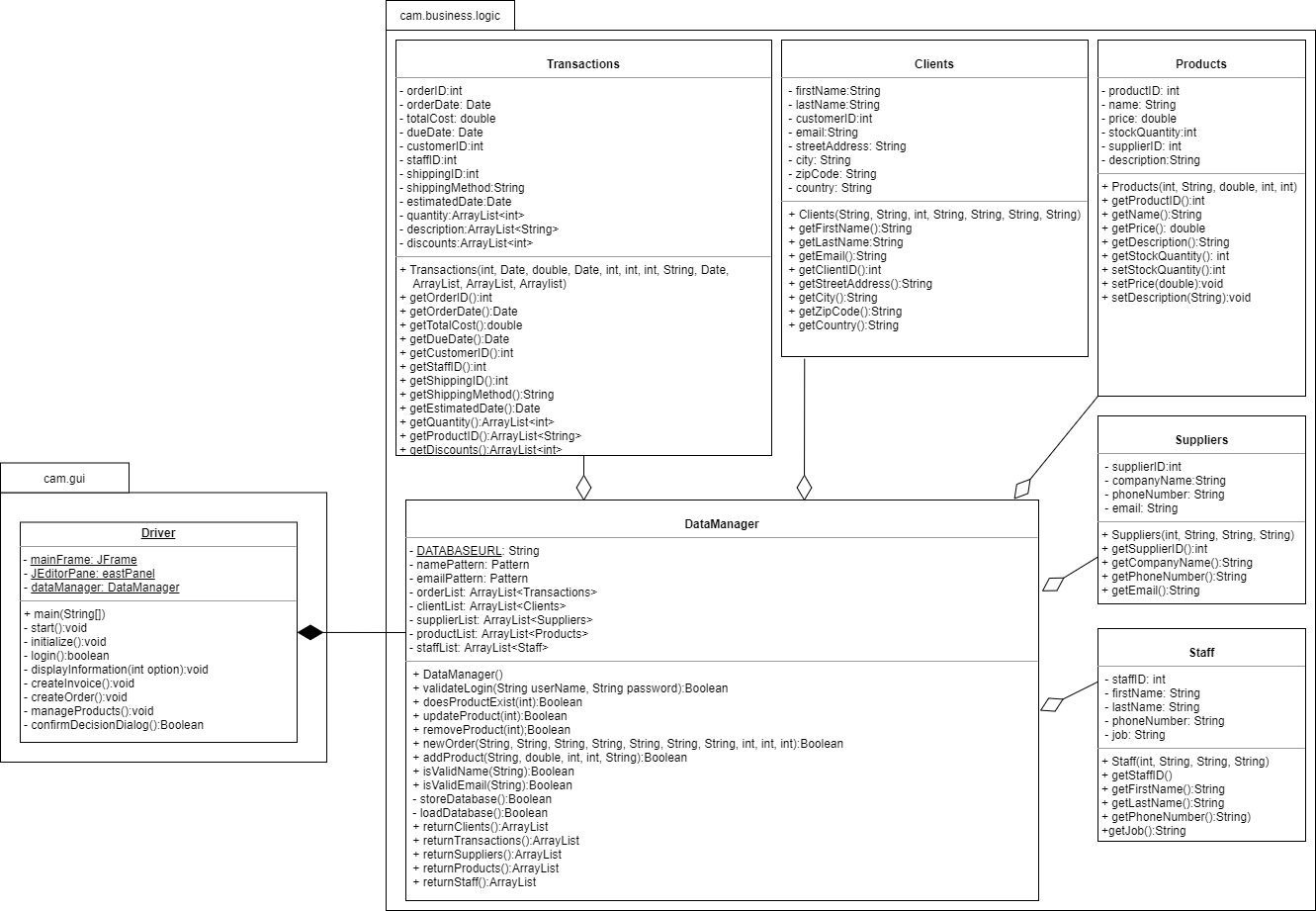
****

**Mapping document overview.**

****

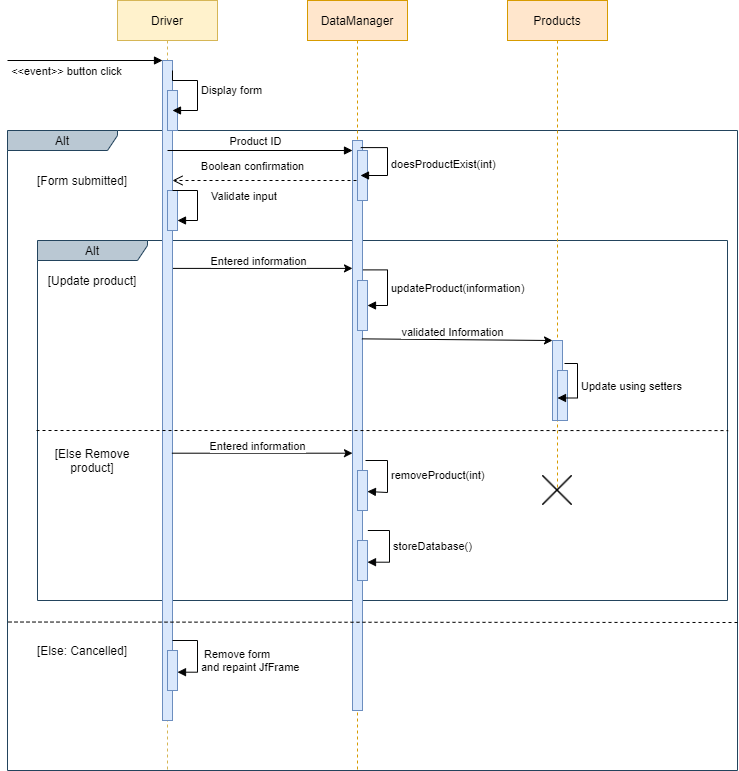
I have colour coded the primary and foreign keys to help show how the tables relate to each other.

### 1.8.6 Class diagram



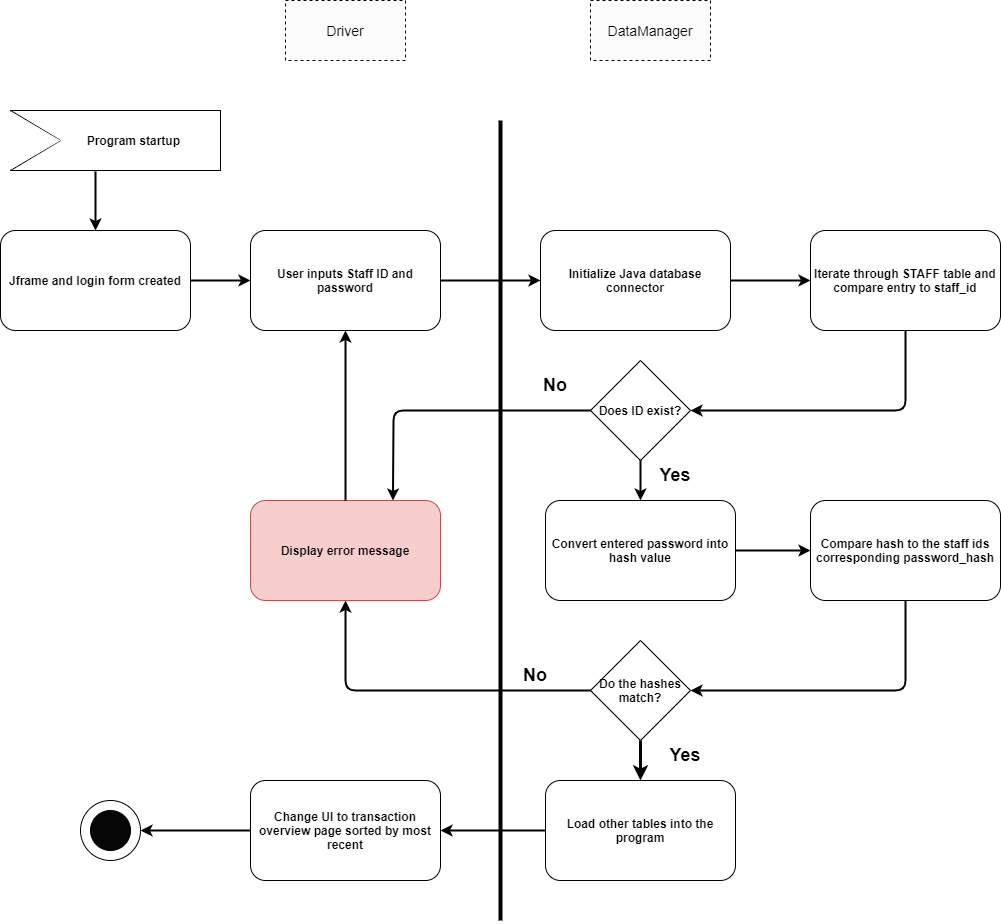
Here we have a class diagram I have constructed to model the system. I have separated the classes into two packages one which handles the user interface and one that handles the business logic. Data Manager will control the flow of the program and return information to the driver so it can be displayed. Transactions class is made up of the order, shipping, and invoice items entities from the ERD.

### 1.8.7 Sequence diagram



This diagram is showing what happens when the user clicks the “Manage product” button. A form is displayed then the entered information is validated to check firstly if the productid exists and secondly whether the related field information is in the correct format. If valid and the update product button is clicked, then the information is sent to data manager where it is updated using the product classes setters. If the button clicked on the form was instead the remove product button, then a method in data manager is called using the product ID and then the product is removed from the database. If the user instead cancels the form, the panel will be removed, and the frame will be updated.

### 1.8.8 Activity diagram



In this diagram I am demonstrating how the login feature will work. On system initialization the main frame of the application is constructed. The login form is also generated then displayed to the user. The user attempting to gain access will then enter information. That information is passed to the data manager class in the business logic, that handles all the validation and control of important data. The java database connector will be created, and the staff table will be used to check if the entered staff ID is valid. If it is not then an error message will be displayed, and the user will have to login again.

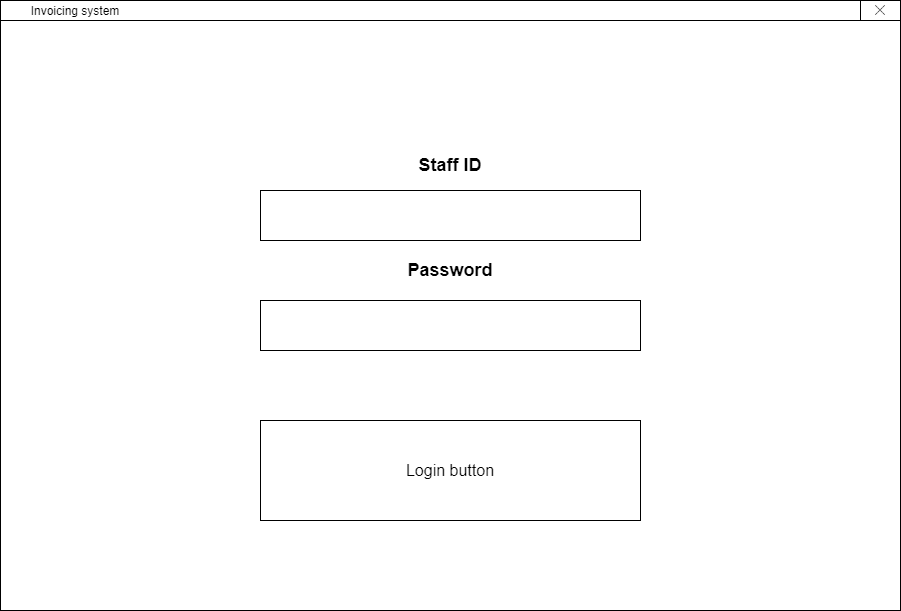
If the staff ID is in the staff table, then the entered password will be converted into a hash and compared to that staff ID’s corresponding password hash, if it does not match then the user will again be displayed an error message and sent back to the login form. If it is correct, then the program will load in the other tables and tell the driver that the login was successful so it can update the interface for the user and allow the staff member to progress to the overview page which will initially display transactions by recent.

## 1.9 View model

### 1.9.1 User interface designs

Not all these designs will make it into the program, these are just templates for discussion with the client to get ideas flowing. Colour will be eventually incorporated into the design and the layouts shown in this document may be scrapped.

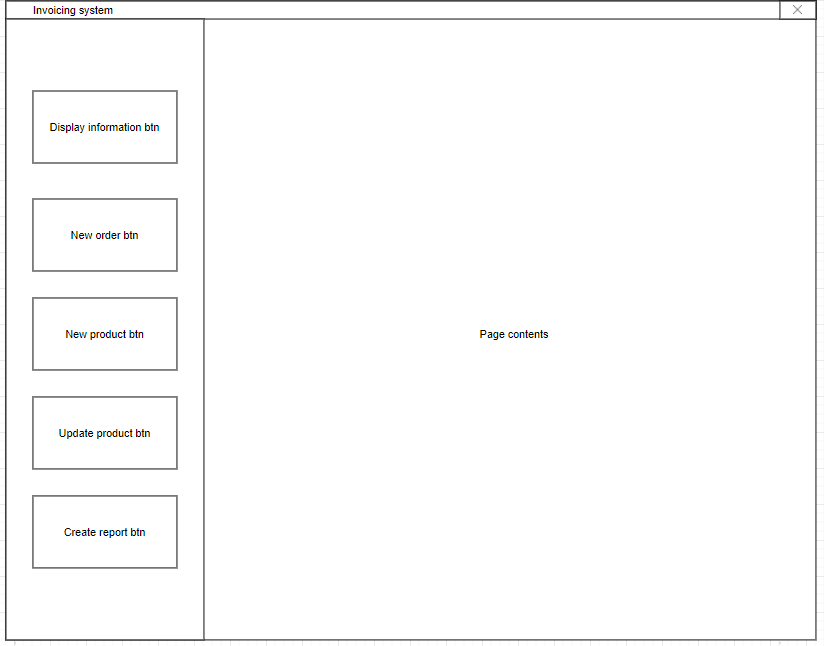
**Login view**

****

Here we have a simplistic security feature when the program is started no client or transaction information will be accessed until the users input matches the correct fields in the staff table. I plan to create a password for each of the two staff and store it in the database as a 32-character hash value. I would rather use hashing than encryption as encryption is bi-directional, it can be ciphered and deciphered. Hashing cannot be deciphered; it is an almost unique representation of a piece of data. This means the only way to validate the user’s input is to convert their entry into a hash and see if it matches the value stored with the associated staff ID.

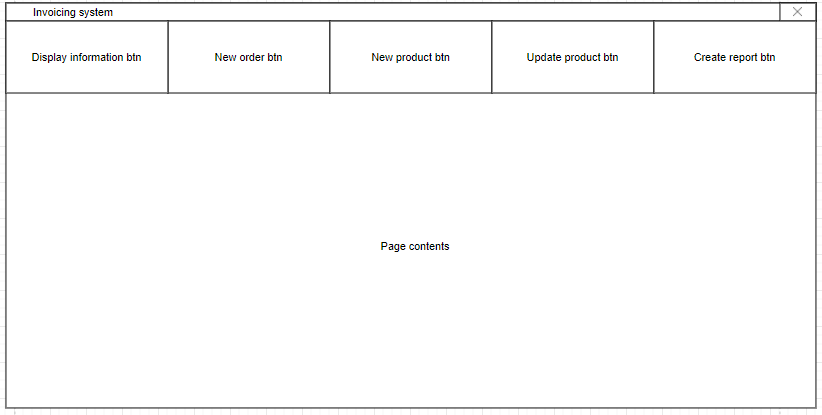
**Application navigation ideas**

I have chosen to incorporate large buttons into my navigation designs for any current or future staff with a visual impairment.

**Vertical navigation**

This left-aligned vertical design maximises usability and accessibility as it is large, simplistic, always in view and familiar. For a new staff member getting used to the new system, this will make it easier. I figured using drop down navigation would decrease the staff’s efficiency in creating orders and managing stock. After reflecting upon both designs, the client has agreed with me that this remains the best option and would like me to incorporate a vertical navigation layout into the application.

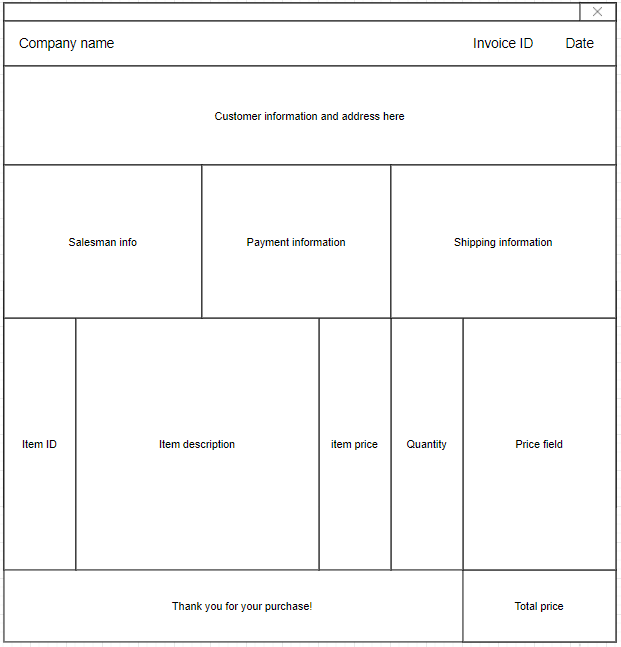
**Horizontal navigation**

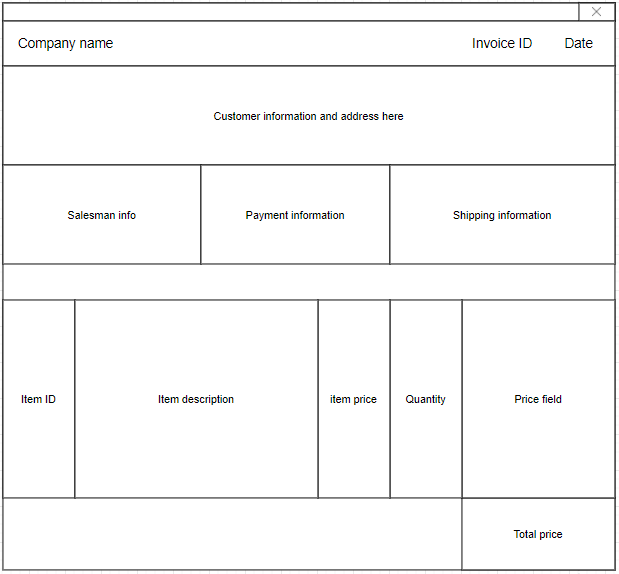
****

This was an alternative option for the navigation, which is as simple as having the bold buttons horizontal rather than vertical. However, I mentioned to the client that as monitors tend to be positioned in landscape view, a horizontal navigation would take up too much of the little vertical space on the screen. That would decrease usability as the staff member working the application may have to deal with a cumbersome amount of scroll bars.

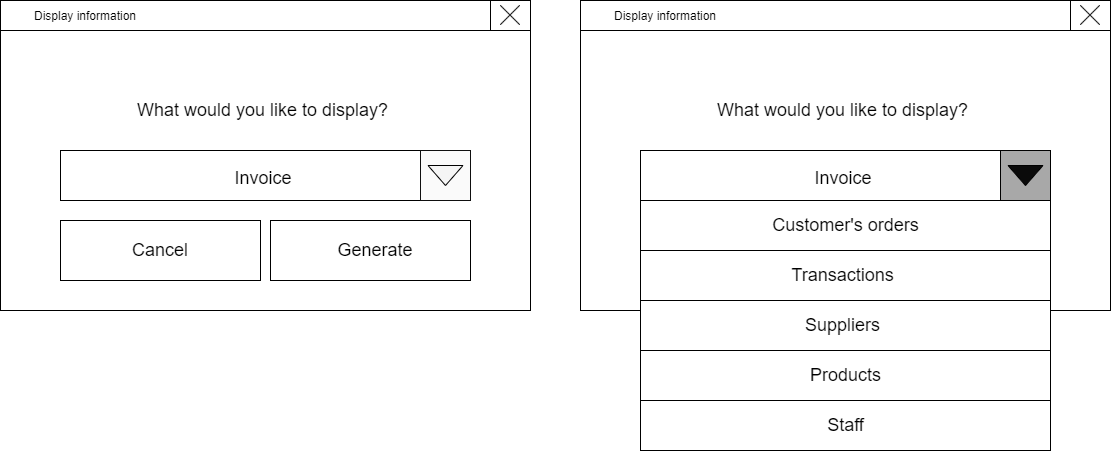
**Individual invoice section design ideas generated from the “create report” button.**

The two examples of how we could model the invoice interface are quite similar however this seems to be a [universal standard](https://www.freshbooks.com/hub/invoicing/design-an-invoice). I have tried to compact the design as much as possible so that it fits on a user’s screen. Here we have two separate wireframes, one has more spacing than the other, but both are similar as the application is to be usable and familiar. Overcomplicating an app that monitors financial transactions and inventory amount is a horrible idea, so usability to improve the client’s business has been my focus for the design of the application.

Along the top of the invoice the company name, invoice ID and date of transaction will be displayed in bold as they are the primary identifiers of the document. Then beneath that we have the customers information, things such as name, email, address, and ID. Then in three columns we will have specification of the sale such as staff present, payment information and shipping methods. At the bottom of the document, we have a tabular view that will contain all the necessary information on the invoice items in the client’s transaction. The individual price will conveniently be aligned on the right hand so that the summed up total price including VAT can be displayed at the very bottom of the document as [internationally standard](https://www.sage.com/en-gb/blog/invoice-cheat-sheet/).

Here is the second design of the invoicing system that is strikingly similar. They must be to comply with the expected layout. In this design I tried to imagine how I could compact so much important information into a small area whilst maintaining readability and usability. Adding white space between the tabular view of items purchased and the sale details at the top will go a long way to improve readability. I have also removed the thank you message at the bottom left as it clutters the design. This invoice layout was much preferred by the client over the alternative design. So, I have decided to go with this as it has a simplicity to it that is familiar.

**Display information dialog**



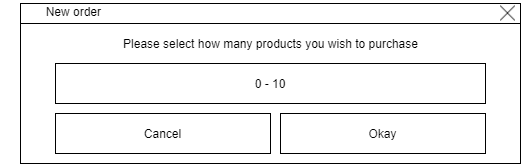
I will prompt the user to select what they would like to display. Choosing one of these options will return all valid information in a tabular layout. Choosing the invoice option or a customer’s orders option will prompt the user again asking for a customer ID which when entered correctly will return either an invoice or a list of all the clients orders.

**Display information overview page**

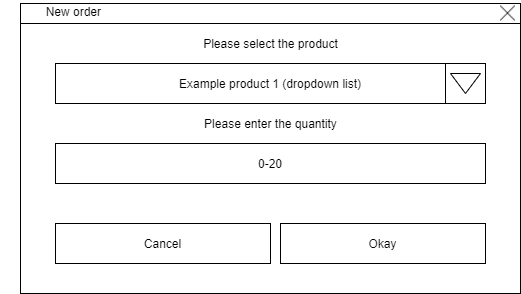
I strongly believe it would be a great idea to have the overview page (retrieved by the display information dialog) to give the admin an option of what to display in a drop-down list then show the selected result in a table format. Here is a very simplistic example of what I am thinking. The values shown in red indicate low inventory levels as the client requested. This section of the application will be populated based on the amount of compatible data stored.

****

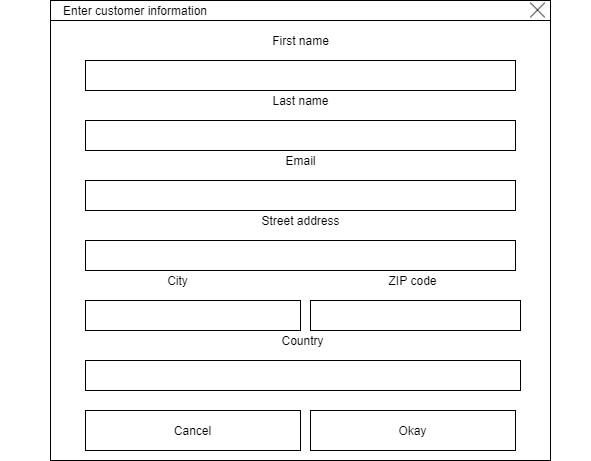
**Create order button**

****

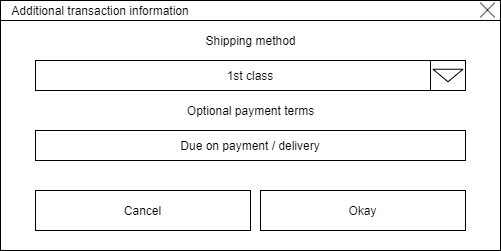
When the user clicks the create order button, they will be asked how many products they would like to purchase. Then the system will use that number to iterate over the new order prompt.

****

The new order prompt will ask the user what product they want to buy from a drop-down list. Then the user will have to enter a quantity based on 0-20. If the quantity is 0 then the order will not be saved. This form will appear once for every product the user wanted to buy in the first form.

****

When all the products have been selected then the user will be shown a customer information form which the user will fill in for the customer. All the purchased items will be associated with the customer information entered in this form.



Finally, the user will be asked to fill in additional information such as the shipping method from a drop-down list and an optional text field about when the payment is due. For example, the admin could say the payment is due on delivery or instantly upon purchase.

**Update product button**

For update product, new order, and new product buttons I would like to use a similar design, below is an idea of what It may end up looking like. This example is a pop-out modal. I will design it so the main panel of the application displays relevant information such as a list of products and the popup shows up next to it so you can quickly cross reference the product id.

****

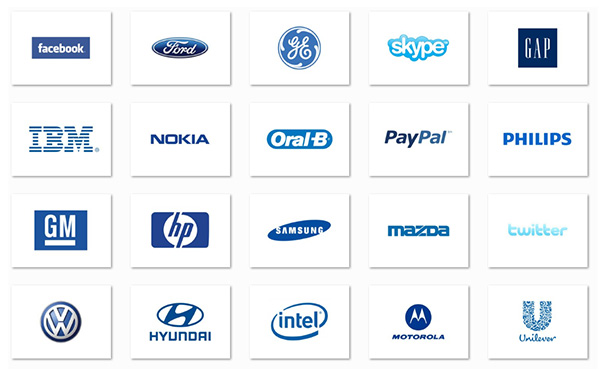
If the user wishes to remove a product, they simply need to enter a valid product ID into the correct text field and the valid product ID will be removed.

If the update product button is clicked then the system will check if the product ID is valid, before updating the other fields. Any fields left empty will not be changed in the database.

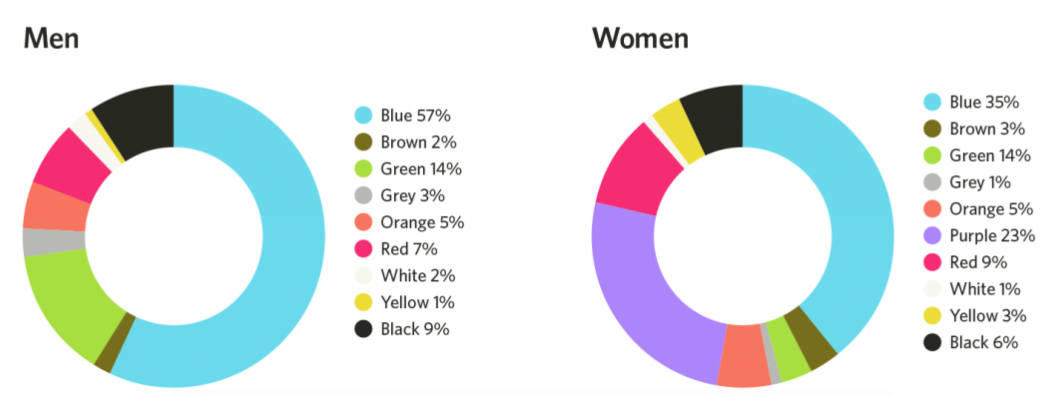
### 1.9.2 Colours, fonts, and accessibility

There are many visual impairments that we need to consider, one example of such is Deuteranomaly, a form of colour blindness which affects an individual’s impression of the colours red and green. Deuteranomaly is incredibly common, as many as 6% of males and 0.4% of females are affected by this form of colour-blindness. I will also briefly cover monochromacy as it is important to not rule out any form of visual impairment.

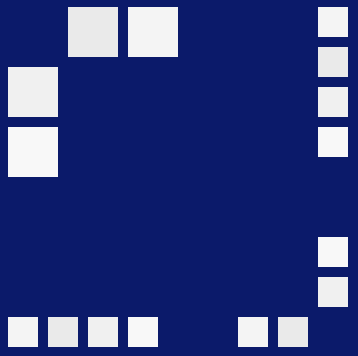
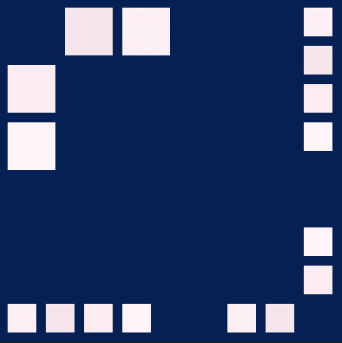
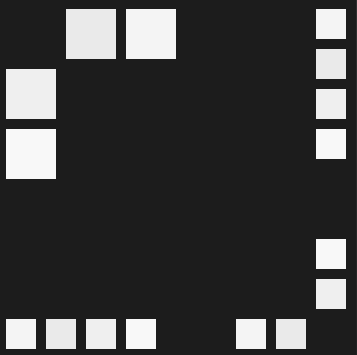
Many large companies design their websites and applications around accessibility by considering disabilities such as visual impairments. They improve their designs by occasionally adding text to speech, improving contrast, and increasing font size. The W3 organisation recommends buttons are large and bold, usually using a contrasting colour and making the button appear 3D with CSS rules is sufficient. Large companies like amazon and Microsoft designed their colour palettes around the colour blue as blue is statistically the most popular colour internationally for both genders. Here is an example of some big names that designed their logo around the colour blue.

[](https://thedesignlove.com/top-20-famous-logos-designed-in-blue/)

Blue is a very distinct colour; it contains many shades. Shades like navy blue can easily compliment lighter colours of text if used correctly. To an individual with another form of colour blindness such as monochromacy (unable to see colour), navy can appear like black which is a complete opposite on the three-dimensional colour space to white hence sufficient contrast.

[](https://marvelapp.com/blog/important-color-ui-design/#:~:text=Without%20a%20doubt%2C%20blue%20is,%2C%20Shazam%2C%20Safari%2C%20etc.)

I would encourage my client to use this shade of navy blue or a similar contrast of colours. As navy blue is a great example of how dynamic a colour can be. A user who is fortunate enough to not have a visual impairment may enjoy the colour and the aesthetic it provides and with this colour palette so can individuals with deuteranomaly. Additionally, the contrast is so sufficient that even people with monochromacy can clearly read the text on the website. Navy blue’s contrast is amplified the more colour-blind the individual is, making it an indubitable choice for any application to include. (See colour palettes below, white boxes represents foreground such as text). Navy - #0b1a6a / rgb(11, 26, 106)

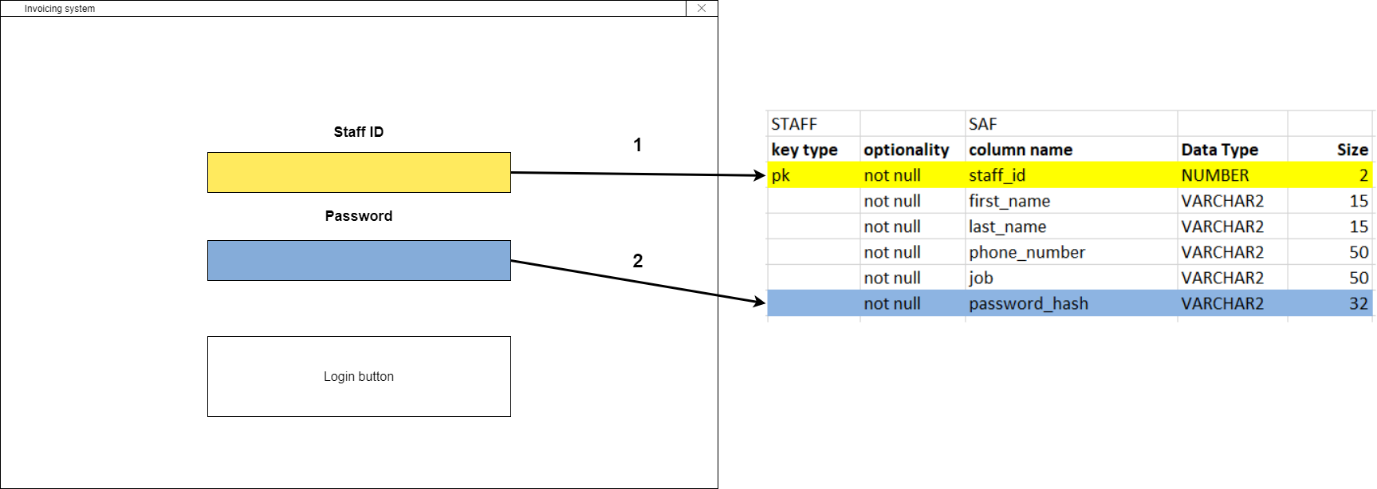
  

Normal view Deuteranomaly Monochromacy

For font choices I would recommend Arial as it is one of the core windows fonts making it recognisable. It is an often-overlooked font, but it is clear and readable typeface. It is important the font size is a minimum of 16px as there are strict guidelines by organisations created to ensure accessibility to most users.

### 1.9.3 Binding model

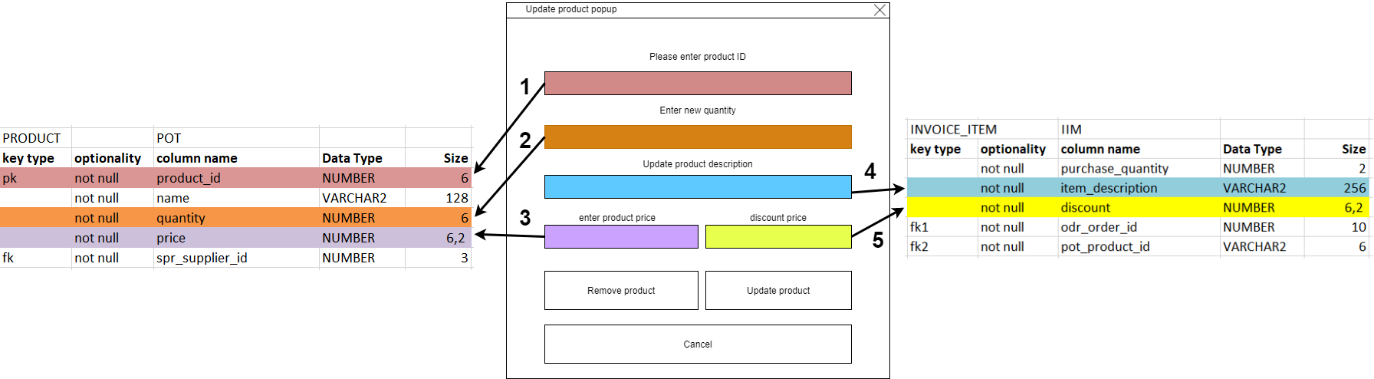
**Login page**



1. When login button is clicked the staff table will be checked to see if the id is valid.

2. If the entered staff id is a value in the table then the password will be hashed and compared to the hashed value stored with the associated staff id. If it matches the user can progress, else they will be displayed an error message.

**Update product popup**



1. When the remove product or update product button are clicked, the system will validate the input and then check the product table in the database to see if the product exists.

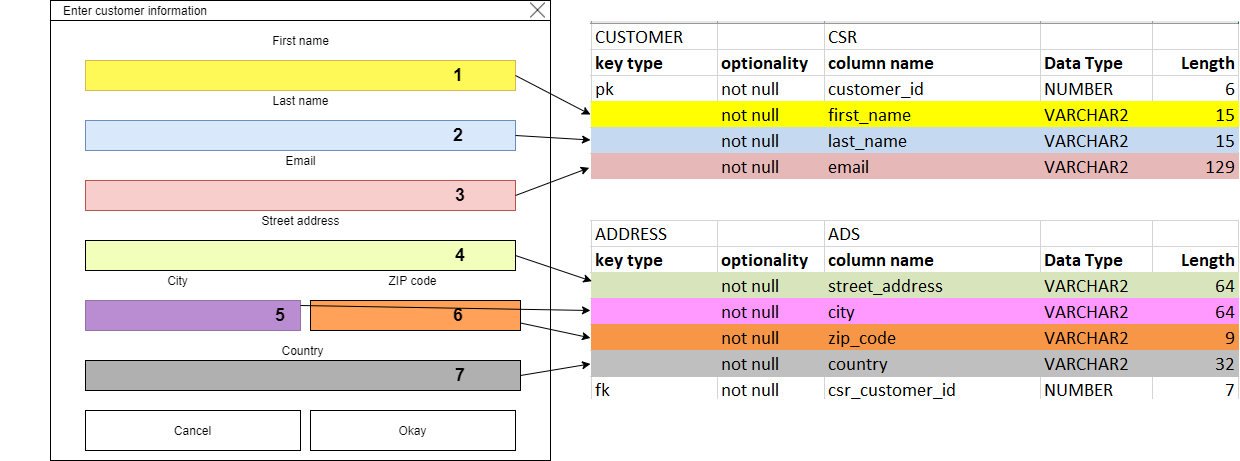
2. If the update product button is clicked and the quantity field contains information, if it is a numeral value between zero and twenty then the corresponding quantity row for the product id will be updated.

3. If the product ID is valid and the entered price is in the correct format when the update product button is clicked then the price field in the product table will be updated.

4. If the product ID is valid and the description for the item is no longer than 256 characters then the update product button will store the updated description into the INVOICE\_ITEM table.

5. If the product ID is valid, discount amount is less than price and price is in the correct format then the discounted price will be entered into the INVOICE\_ITEM table.

**Entering customer information**



1. The first name of the customer will be validated, converted to lower case, and its first letter capitalised before eventually ending up in the customer table in the first\_name field. First name cannot be longer than 15 characters.

2. The first name of the customer will be validated, converted to lower case, and its first letter capitalised before eventually ending up in the customer table in the last\_name field. Last name cannot be longer than 15 characters.

3. Email will be validated. It must include an at sign. It cannot be longer than 129 characters. It will eventually end up in the email field of the customer table.

4. Street address will be entered and will be validated to ensure it is no longer than 64 characters. It will then end up in the address table in the street\_address column. All address information entered here will be associated with the auto generated customer ID.

5. City input will be stored in the address table as a varchar2 with a maximum of 64 in length.

6. ZIP CODE will be validated by the program to ensure it is only 9 characters in length before ending up in the zip\_code column in the address table.

7. Country must be within 4 (shortest country name: Chad) to 32 characters in length. It will also end up in the address table as a VARCHAR2.

# 2 Bibliography

Susan Gardner (27th January 2021). First client interview. Discussion of project.

Susan Gardner (3rd February 2021). Second client interview. Clarification of application.

Susan Gardner (24th February 2021). Third client interview. Discussion of design.

Hardware expenses <https://www.currys.co.uk/gbuk/computing-accessories/components-upgrades/processors/intel-core-i7-10700k-unlocked-processor-10207653-pdt.html>

Cloud storage <https://github.com/>

Research resources <https://stackoverflow.com/> <https://docs.oracle.com/javase/8/docs/api/> <https://www.w3schools.com/java/default.asp>

Colour recommendation <https://paletton.com/#uid=1000u0kllllaFw0g0qFqFg0w0aF>

<https://marvelapp.com/blog/important-color-ui-design/#:~:text=Without%20a%20doubt%2C%20blue%20is,%2C%20Shazam%2C%20Safari%2C%20etc>.

https://thedesignlove.com/top-20-famous-logos-designed-in-blue/

<http://web-accessibility.carnegiemuseums.org/design/color/>

https://www.invisionapp.com/inside-design/color-accessibility-product-design/

Invoice layout <https://www.freshbooks.com/hub/invoicing/design-an-invoice>

<https://www.sage.com/en-gb/blog/invoice-cheat-sheet/>

https://en.wikipedia.org/wiki/Postcodes\_in\_the\_United\_Kingdom#:~:text=The%20postcodes%20are%20alphanumeric%2C%20and,and%20the%20inward%20code%20respectively.