Graded Unit 2 Solution PLanning

Max Ritchie - 545737

BCSC2/FK

Forth Valley College

Contents

[Second Formal Meeting with Client and Questionnaire 2](#_Toc8164308)

[1. Undertaking the Analysis 3](#_Toc8164309)

[1.1. Business Model 3](#_Toc8164310)

[1.1.1. Textual Analysis – Natural Language Analysis (NLA) 3](#_Toc8164311)

[1.1.2. Written Use Case Descriptions 5](#_Toc8164312)

[1.1.3. CRC Cards 12](#_Toc8164313)

[1.1.3.1. Database Entities 12](#_Toc8164314)

[1.1.3.1. Classes 18](#_Toc8164315)

[1.1.4. ERD 20](#_Toc8164316)

[1.1.4.1. Top-Level ERD 20](#_Toc8164317)

[1.1.4.2. Fully Detailed ERD 21](#_Toc8164318)

[1.1.4.2.1. Fully Detailed ERD and Breakdown 21](#_Toc8164319)

[1.1.5. Activity Charts 25](#_Toc8164320)

[1.1.6. Top Level Use Case Model 28](#_Toc8164321)

[1.1.6.1. Initial Top-Level Use Case Diagram 28](#_Toc8164322)

[1.1.6.2. Next Stage Top-Level Use Case Model 29](#_Toc8164323)

[1.1.7. Initial Class Diagram 31](#_Toc8164324)

[1.2. View Model 32](#_Toc8164325)

[1.2.1. Identification of Data Binding Model 32](#_Toc8164326)

[1.2.2. User Analysis 32](#_Toc8164327)

[1.2.2.1. Initial Wireframes 33](#_Toc8164328)

[1.2.2.2. User Usability 45](#_Toc8164329)

[1.2.2.2.1. Applications Navigation 45](#_Toc8164330)

[1.2.2.2.1.1. Navigation Map 45](#_Toc8164331)

[1.2.2.2.2. Colour Scheme 46](#_Toc8164332)

[2. Evidence for Analysis 47](#_Toc8164333)

[2.1. Business Model 47](#_Toc8164334)

[2.1.1. Static Model 47](#_Toc8164335)

[2.1.2. Dynamic model 48](#_Toc8164336)

[2.1.2.1. Use Case Models 48](#_Toc8164337)

[2.1.1.2. Sequence Diagrams 52](#_Toc8164338)

[2.2. View Model 55](#_Toc8164339)

[2.2.1. UI Design 55](#_Toc8164340)

[2.2.2. Data Binding Design 56](#_Toc8164341)

[3. Information Sources 61](#_Toc8164342)

# Second Formal Meeting with Client and Questionnaire

After receiving feedback after the first Inception Phase, I scheduled a second client interview with a small questionnaire for the client to answer. The questions that I asked the client were to clarify certain areas and aspects, also to confirm with the client that all design that was produced in this stage was satisfactory enough to continue using and to ask if they have any changes they would like to contribute. I will be using this information to help with the creation of diagrams, now knowing what must be added and used. Here is a table that contains all the questions that were asked along with the answers that I received:

|  |  |  |
| --- | --- | --- |
| **No.** | **Question** | **Answer** |
| 1. | Will customer payment details be stored and if so, what will be stored? | Yes, customer credit card details will be stored. The details of this will be:   * Name on Card * Number * Security Number * Expiry Date |
| 2. | Are there any improvements of the created moqups and logo? | Nope, keep it the same as it is. |
| 3. | Are there any other improvements that could be done overall or suggestions for improvement? | Nope, fine as it is. |
| 4. | After seeing the initial top-level use case diagram, is there any additional functional features that are missing or could be added? | Nope, they are prefect. |
| 5. | What are your thoughts on the current layout/design and colour scheme that has been created? | Perfect, keep it the same, it matches the application that you are creating. |

# 1. Undertaking the Analysis

## 1.1. Business Model

Identification of classes, methods and attributes

### 1.1.1. Textual Analysis – Natural Language Analysis (NLA)

I am using the given brief to do an NLA to identify how and where my functional and non-functional requirements were obtained. All key words and sentences that are pulled out of the brief will be highlighted a different colour depending on what key word/sentence is. Initial functional requirements will be highlighted in red, initial non-functional requirements will be highlighted in blue and any users/actors that are identified will be highlighted in green.

I carried out some research on NLA as I needed to be refreshed on how and why it is important to carry out the process. I have referenced the website that I used to help me gain a stronger understand of what is required and what I needed to do. (Contrib.Andrew.cmu.edu, 2019)

Project Brief  
Scenario Two - A Small Business System

You have been sent a proposal, which has you develop a small business application for the company Acquiesce Industries who deal with the selling of music, music merchandise and music memorabilia. The company will have you create a sales invoicing and stock control system suitable for a typical smaller retailer operating within the UK. The application should produce a sales invoice for every transaction and update of stock in real time with the use of a database.

There should be access to multiple different reports that need to be generated, detailed reports of daily, weekly, monthly and annual sales analysis, stock turnover and profit. There should also be reports dealing with stock including reorder reports for certain items and stock outages. Information of customers, suppliers, products and invoice information which includes transaction logs should be available only through employee access.

The system must include a database which will be used to store all stock. The database must be accessible to employee and admin users only to be able to add and remove stock when required. The products name, price, quantity/units and total units sold must be present within the database.

The company are looking for an app that is attractive, friendly and professional looking, going along with the standard requirements to all users, with an interface that is easy to understand and use along with complying with recognized standards. The company are expecting a top-level use case design to be created and presented within the next two weeks.

The application should be compatible to run on different platforms using minimum specs required. The application can either be created with a console-based interface (Text based) or a GUI (Graphical User Interface).

I have taken all the data sourced from the brief and sorted it into their separate sections. These are only the initial points that were taken into the first initial client meeting to discuss how to system will work and to find out more functional features and what must be required. With the use of this is was able to build upon my gathered information and produce a detailed functional and non-functional requirement list of what the client is wanting the and expecting the system to include. This information is shown within each of the separate tables below:

|  |
| --- |
| Actors |
| Employee |
| Admin |
| Customer |
| Database |

|  |
| --- |
| Initial Functional Requirements |
| Create a sales invoicing and stock control system. |
| Should produce a sales invoice for every transaction. |
| Detailed reports of daily, weekly, monthly and annual sales analysis, stock turnover and profit. |
| Should be access to multiple different reports that need to be generated. |
| Should also be reports dealing with stock including reorder reports for certain items and stock outages. |
| Must include a database. |
| Able to add and remove stock. |
| Products name, price, quantity/units and total units sold must be present within the database. |

|  |
| --- |
| Initial Non-Functional Requirements |
| Selling of music, music merchandise and music memorabilia. |
| Update of stock in real time. |
| Attractive, friendly and professional looking, going along with the standard requirements to all user. |
| An interface that is easy to understand and use along with complying with recognized standards. |
| Top-level use case design to be created and presented within the next two weeks. |
| Should be compatible to run on different platforms using minimum specs required. |
| Created with a console-based interface (Text based) or a GUI (Graphical User Interface). |

### 1.1.2. Written Use Case Descriptions

I have taken my initial top-level use case diagram (can be found within section *1.1.5.1. Initial Top-Level Use Case Diagram)* and created tables displaying in detail all use cases required to make the system work. I have included the normal flow of what each use case will do when carried out correctly and included a few extension flow scenarios. These extension flow scenarios describe a scenario that could possible happen when the system runs incorrectly, or an error/fault occurs. Each scenario is walked through on how it would be solved or how the system would handle that error. Include in section *2.1.2.1. Use Case Models* is a breakdown of all uses cases and what functionality these possess. All tables created displaying all the use cases are shown below:

|  |  |
| --- | --- |
| **Use Case One** | |
| **Name** | Creation and use of a database |
| **Primary Actor** | Database |
| **Description** | The database will be created and used to store all stock, invoices, supplier and employee and customer information. |
| **Pre-Condition** | UwAmp must be running and used to bring the database online. |
| **Post-Condition** | Database successfully connected to the system. |
| **Trigger Event** | Automatically connects when the system starts up. |
| **Normal Flow** | 1. The database comes online. 2. On system start up the database and system communicate with each other and connect. 3. The system pulls out and displays required data from the database. 4. The system accesses the database to be modified. 5. The database allows the system to access it and allow the user to modify data. |
| **Extension Flow** | **Scenario One** |
| 1. The database is offline.    1. The system displays an error message informing the user that there are no current databases to connect to.    2. Continue to **Normal Flow Step 1.** |
| **Scenario Two** |
| 1. The system cannot connect to the database.    1. The system displays an error message informing the user that there is a connectivity issue with the database and are unable to connect.    2. Continue to **Normal Flow Step 2.** |
| **Scenario Three** |
| 1. The system displays the wrong data from the database.    1. The system and database must shut down.    2. A trouble shooting process must happened to resolve the problem.    3. Continue to **Normal Flow Step 1.** |

|  |  |
| --- | --- |
| **Use Case Two** | |
| **Name** | Display homepage |
| **Primary Actor** | Employee |
| **Description** | The homepage will be the first page that will be displayed to the screen when the system is started up. |
| **Pre-Condition** | None. |
| **Post-Condition** | Displays homepage to the screen. |
| **Trigger Event** | Automatically displays when the system starts up. |
| **Normal Flow** | 1. System starts up. 2. Homepage automatically gets displayed to the screen. |
| **Extension Flow** | **Scenario One** |
| 1. Homepage doesn’t display.    1. Shut down the system.    2. Check device compatibility, device might not be able to run the application correctly.    3. Continue to **Normal Flow Step 1.** |

|  |  |
| --- | --- |
| **Use Case Three** | |
| **Name** | Display login page |
| **Primary Actor** | Employee/Customer |
| **Description** | The login page is where employees and customer can login into their accounts within the system. |
| **Pre-Condition** | None. |
| **Post-Condition** | Logs the user into the system. |
| **Trigger Event** | Page displays when the user clicks on the login/register meu button. |
| **Normal Flow** | 1. The homepage of the system is displayed to the screen. 2. Login/register menu button has been clicked. 3. The system displays the login page to the screen. |
| **Extension Flow** | **Scenario One** |
| 1. User enters invalid data    1. The system displays an error message informing the user that they have entered invalid credentials (username or password).    2. The system informs the user to re-enter their credentials.    3. Continue to **Normal Flow Step 3.** |
| **Scenario Two** |
| 1. User backs out of login screen.    1. Login screen gets closed.    2. Homepage gets displayed to the screen.    3. Continue to **Normal Data Flow Step 1.** |

|  |  |
| --- | --- |
| **Use Case Four** | |
| **Name** | Display user account page |
| **Primary Actor** | Customer |
| **Description** | The users account page is where customers can see their details and allow them to change their password. |
| **Pre-Condition** | User must be logged into the system. |
| **Post-Condition** | Displays the users personal account details to the screen. |
| **Trigger Event** | Displays when the user clicks on the account button within the menu. |
| **Normal Flow** | 1. The homepage is being displayed to the screen. 2. The user clicks on the account menu button. 3. The system displays the users account page to the screen. |
| **Extension Flow** | **Scenario One** |
| 1. The user is not currently logged into the system.    1. The system does not display the account menu button on the homepage.    2. Continue to **Normal Flow Step 1.** |
| **Scenario Two** |
| 1. The user logs out of the system    1. The user will be redirected to the homepage from their account page.    2. The account menu button will no longer appear in the menu on the homepage.    3. Continue to **Normal Flow Step 1.** |

|  |  |
| --- | --- |
| **Use Case Five** | |
| **Name** | Display products page |
| **Primary Actor** | Customer |
| **Description** | The products page displays all products that can be purchased. |
| **Pre-Condition** | There must be product items stored within the database to be displayed. |
| **Post-Condition** | Displays selected category of products. |
| **Trigger Event** | One of the three categories must be selected within the menu to display that type of product. |
| **Normal Flow** | 1. The homepage is being displayed to the screen. 2. The user clicks on one of the three product category buttons within the menu (music, merchandise, memorabilia). 3. The system displays all products on screen from the chosen category. |
| **Extension Flow** | **Scenario One** |
| 1. There are no product items currently stored in the database.    1. No items will appear on the page.    2. The system will display an error message to the screen to inform the user that there is no current stock being held for that chosen category.    3. Continue to **Normal Flow Step 1.** |
| **Scenario Two** |
| 1. The user selects the wrong category.    1. The user can go back to the home page and select their preferred category from the menu.       1. The user can select the category they want form the menu, which will be displayed on the current page.    2. Continue to **Normal Flow Step 2.** |

|  |  |
| --- | --- |
| **Use Case Six** | |
| **Name** | Display single product page |
| **Primary Actor** | Customer |
| **Description** | A page with only the one product being shown that the user has selected. |
| **Pre-Condition** | There must be product items stored within the database to be displayed. |
| **Post-Condition** | Displays a single item to the screen with a description and an option to add to the users basket. |
| **Trigger Event** | One of the categories must be selected within the menu to display that type of product. |
| **Normal Flow** | 1. The products page is being displayed to the screen. 2. The user clicks on product. 3. The system fetches the required information of the product from the database. 4. The system displays the single items page with the required information. |
| **Extension Flow** | **Scenario One** |
| 1. The item that has been selected is sold out/no units left in the database.    1. The system will display a message to the screen informing the user that the current product is sold out and is unavailable at this current time. |
| **Scenario Two** |
| 1. The user selected the wrong item.    1. The user can select the homepage and select their chosen category again.       1. The user can select back to take them back the products screen.    2. Continue to **Normal Flow Step2.** |

|  |  |
| --- | --- |
| **Use Case Seven** | |
| **Name** | Display users basket page |
| **Primary Actor** | Customer |
| **Description** | The users basket if where all items that the user wants to be will be displayed with the option to remove items or proceed to the checkout. |
| **Pre-Condition** | There must be items added to the users basket to be displayed on the page. |
| **Post-Condition** | Displays the users basket along with all items that have been added. |
| **Trigger Event** | The basket button/icon must be selected in the menu to display the basket page to the screen. |
| **Normal Flow** | 1. The current page that the user is currently on is being displayed to the screen. 2. The user clicks on the basket button within the menu. 3. The system displays the users basket to the screen. 4. The users basket displays all items that the user has added. |
| **Extension Flow** | **Scenario One** |
| 1. The user has no items added to their basket.    1. The system will display the page but with no items on screen and with the message informing the user that they haven’t added any items to their basket.    2. The user is unable to proceed to the checkout page.    3. Continue to **Normal Flow Step 1.** |
| **Scenario Two** |
| 1. The user wants to remove an item from their basket.    1. With each item being displayed there is also a remove button displayed.    2. The user clicks the remove button and that item will be removed from the screen and the basket.    3. Continue to **Normal Flow Step 4.** |

|  |  |
| --- | --- |
| **Use Case Eight** | |
| **Name** | Display checkout page |
| **Primary Actor** | Customer |
| **Description** | The checkout page will allow the customer to pay for and confirm their order of item(s). |
| **Pre-Condition** | There must be product items present within the users basket. |
| **Post-Condition** | Displays the checkout page with the total price and the option to buy. |
| **Trigger Event** | When product items are present within the users basket, the checkout button will now be available for the user to proceed to the checkout page when clicked. |
| **Normal Flow** | 1. The users basket is being displayed to the screen. 2. The user clicks the proceed to checkout button displayed on screen. 3. The system displays the checkout screen, displaying the total cost, payment details and items being purchased. 4. The user clicks the confirm order button to confirm the purchase of the items. 5. The system generates an invoice which is displayed to the screen and saved into the database. |
| **Extension Flow** | **Scenario One** |
| 1. The user cancels the order.    1. The transaction process is cancelled.    2. The user is redirected back to their basket.    3. Continue to **Normal Flow Step 1.** |

|  |  |
| --- | --- |
| **Use Case Nine** | |
| **Name** | Display employee page |
| **Primary Actor** | Employee |
| **Description** | The page an employee will see when they log into the system, only employees have access to this page. |
| **Pre-Condition** | An employee account must be logged into the system. |
| **Post-Condition** | Page displays when employee account gets used to login. |
| **Trigger Event** | Page automatically appears when an employee logs into the system. |
| **Normal Flow** | 1. Employee logins into the system. 2. The system automatically displays the employees account page to the screen. |
| **Extension Flow** | **Scenario One** |
| 1. Trying to access an employee account page using a customer account.    1. Only employee accounts have access to an employee account page, customer accounts have their own separate account page.    2. Continue to **Normal Flow Step 1.** |

|  |  |
| --- | --- |
| **Use Case Ten** | |
| **Name** | Display database edit screen |
| **Primary Actor** | Employee |
| **Description** | A screen that allows the user to edit the contents of the database. |
| **Pre-Condition** | An employee account must be logged into the system. |
| **Post-Condition** | Displays the edit database pop up screen. |
| **Trigger Event** | The pop-up displays when the edit database button has been clicked. |
| **Normal Flow** | 1. Employee account page displayed to the screen. 2. User clicks edit database button. 3. The system displays the pop-up screen. 4. The pop-up screen displays different options that the user can select to edit the database. |
| **Extension Flow** | **Scenario One** |
| 1. Database not connected.    1. The system will display an error message to the screen to inform the user that there are currently no databases connected to the system.    2. Continue to **Normal Flow Step 2.** |

### 1.1.3. CRC Cards

I have decided to create two sets of CRC cards. One set will help me identify my entities that are required for the database that will be created and the second set will help me identify and give me a greater understanding of what classes are required for the creation of the application. I carried out a some research to help me understand what is required with CRC cards and I found a website very useful with understand and creation on my CRC cards, the website is : (Agilemodeling.com, 2019).

#### 1.1.3.1. Database Entities

All entities that I have come up with that are required for the database. I have laid them out in tables along with descriptions to help understand why they are required. All attributes are set to mandatory, any attributes that are not are specified within the attributes and description and the reasoning why. The tables that have been created are shown below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity Number** | | 1. | |
| **Entity Name** | | Employee | |
| **No.** | **Attribute** | | **Description** |
|  | Employee ID | | The employees unique identifier to identify the employee (no two are the same) |
|  | Given Name | | The employees given/first name |
|  | Last Name | | The employees last/surname |
|  | Username | | The employees generated username using first three letters of their given name and adding a number at the end |
|  | Password | | The employees password which is assigned and stored, can be changed at any time by the employee |

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity Number** | | 2. | |
| **Entity Name** | | Customer | |
| **No.** | **Attribute** | | **Description** |
|  | Customer ID | | The customers unique identifier to identity the customer (no two are the same) |
|  | Given Name | | The customers given/first name |
|  | Last Name | | The customers last/surname |
|  | Username | | The customers username, created by themselves |
|  | Password | | The customers password, created by themselves and stored |
|  | Email Address | | The customers email address |
|  | Phone Number | | The customers phone number |
|  | Address ID | | The customers address ID linked to another entity |
|  | Payment ID | | The customers payment ID linked to another entity |

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity Number** | | 3. | |
| **Entity Name** | | Customer Address | |
| **No.** | **Attribute** | | **Description** |
|  | Customer Address ID | | The customers address unique identifier to identity the customers address (no two are the same), this is used to link to the customer |
|  | Address Line One | | The customers address line, this will be mandatory |
|  | Address Line Two | | The customers address line two, this will be optional, if the customer requires more space for address information |
|  | City | | The customers city that their address is located |
|  | House Number | | The customers house number |
|  | Postcode | | The customers postcode |

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity Number** | | 4. | |
| **Entity Name** | | Customer Payment Details | |
| **No.** | **Attribute** | | **Description** |
|  | Customer Payment Details ID | | The customer payment details unique identifier to identity the customers address (no two are the same) |
|  | Name on Card | | The customers name that is displayed on the credit card |
|  | Security Number | | The customers credit card security number, the three or four digits usually displayed on the back of the credit card |
|  | Expiry Date | | The customers credit cards expiry date displayed on the credit card |
|  | Customer ID | | The customers id, this links the payment details to the customer with the use of a foreign key |

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity Number** | | 5. | |
| **Entity Name** | | CD | |
| **No.** | **Attribute** | | **Description** |
|  | CD ID | | The CDs unique identifier to identity the customer (no two are the same) |
|  | Name | | The CDs name |
|  | Price | | The CDs price |
|  | Quantity | | The CDs amount/number of units of the product currently being held |
|  | Description | | The CDs description, this is optional |
|  | Image | | The CDs image |
|  | Artist | | The CDs artist |
|  | Genre | | The CDs genre |
|  | No of Tracks | | The CDs No of tracks |
|  | Year Released | | The CDs year of release |

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity Number** | | 6. | |
| **Entity Name** | | Vinyl | |
| **No.** | **Attribute** | | **Description** |
|  | Vinyl ID | | The vinyl’s unique identifier to identity the customer (no two are the same) |
|  | Name | | The vinyl’s name |
|  | Price | | The vinyl’s price |
|  | Quantity | | The vinyl’s amount/number of units of the product currently being held |
|  | Description | | The vinyl’s description, this is optional |
|  | Image | | The vinyl’s image |
|  | Artist | | The vinyl’s artist |
|  | Genre | | The vinyl’s genre |
|  | No of Tracks | | The vinyl’s No of tracks |
|  | Year Released | | The vinyl’s year of release |

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity Number** | | 7. | |
| **Entity Name** | | Merchandise | |
| **No.** | **Attribute** | | **Description** |
|  | Merchandise ID | | The merchandises unique identifier to identity the customer (no two are the same) |
|  | Name | | The merchandises name |
|  | Price | | The merchandises price |
|  | Quantity | | The merchandises amount/number of units of the product currently being held |
|  | Description | | The merchandises description, this will be optional for some products |
|  | Image | | The merchandises image |
|  | Size | | The merchandises size, clothing sizes |
|  | Colour | | The merchandises colour, colour of clothing |

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity Number** | | 8. | |
| **Entity Name** | | Memorabilia | |
| **No.** | **Attribute** | | **Description** |
|  | Memorabilia ID | | The memorabilia’s unique identifier to identity the customer (no two are the same) |
|  | Name | | The memorabilia’s name |
|  | Price | | The memorabilia’s price |
|  | Quantity | | The memorabilia’s amount/number of units of the product currently being held |
|  | Description | | The memorabilia’s description, this will be optional for some products |
|  | Image | | The memorabilia’s image |

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity Number** | | 9. | |
| **Entity Name** | | Invoice | |
| **No.** | **Attribute** | | **Description** |
|  | Invoice ID | | The invoices unique identifier to identity the customer (no two are the same) |
|  | Date | | The invoices date of transaction |
|  | Time | | The invoices time of transaction |
|  | Total Price | | The invoices total price of transaction |
|  | Customer ID | | The invoices customer ID, used to identify the customer that carried out the transaction |
|  | Order ID | | The invoices order ID, used to identity the order it came form |

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity Number** | | 10. | |
| **Entity Name** | | Order | |
| **No.** | **Attribute** | | **Description** |
|  | Order ID | | The orders unique identifier to identify the order (no two are the same) |
|  | Date | | The orders date of transaction |
|  | Time | | The orders time of transaction |
|  | Quantity | | The orders quantity amount |

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity Number** | | 11. | |
| **Entity Name** | | Item | |
| **No.** | **Attribute** | | **Description** |
|  | Item ID | | The items unique identifier to identify the item (no two are the same) |
|  | Name | | The items name |
|  | Category | | The items assigned category |
|  | Description | | The items description, this will be optional for some products |

|  |  |  |  |
| --- | --- | --- | --- |
| **Entity Number** | | 12. | |
| **Entity Name** | | Supplier | |
| **No.** | **Attribute** | | **Description** |
|  | Supplier ID | | The supplier’s unique identifier to identify the supplier (no two are the same) |
|  | Company Name | | The suppliers company name |
|  | Address | | The suppliers address, this is set to optional |
|  | Phone Number | | The suppliers phone number |
|  | Email Address | | The suppliers email address, this is set to optional as phone number is already a mandatory piece of contact information |

#### 1.1.3.1. Classes

These are my second set of CRC cards; these cards help me with the identification of the base classes that are required to make the application function. These are only first initial classes, they may be subjects to slightly change or even additional classes may be added to help with functionality. I have created tables and displayed each class with responsibilities and collaborators that I think they will have to comply with, the tables that have been created are shown below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Number** | | 1. | |
| **Class Name** | | GUI (Graphical User Interface) | |
| **No.** | **Responsibility** | | **Collaborators** |
|  | What is displayed to the screen, what the user sees | | * Database * Driver |
|  | Allows the user to interact with the application | |
|  | Display information to the screen form the database | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Number** | | 2. | |
| **Class Name** | | Driver | |
| **No.** | **Responsibility** | | **Collaborators** |
|  | Controls the application | | * Customer * Database Editor * Employee * Graphical User Interface * Item * Invoice |
|  | Calls other classes to perform operations | |
|  | Creates objects | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Number** | | 3. | |
| **Class Name** | | Employee | |
| **No.** | **Responsibility** | | **Collaborators** |
|  | Interacts with database to allow information to be stored into the program | | * Database * Driver * Graphical User Interface |
|  | Generates reports with the use of the database | |
|  | Ability to edit the database | |
|  | Stores information about the employee and validates it from the database | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Number** | | 4. | |
| **Class Name** | | Customer | |
| **No.** | **Responsibility** | | **Collaborators** |
|  | Stores information about the customer retrieved from the database | | * Database * Driver |
|  | Interacts with the data base to validate customer information | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Number** | | 5. | |
| **Class Name** | | Item | |
| **No.** | **Responsibility** | | **Collaborators** |
|  | Retrieves and stores information for a single in from the database into the program | | * Database * Driver * Graphical User Interface |
|  | Displays information to the screen | |

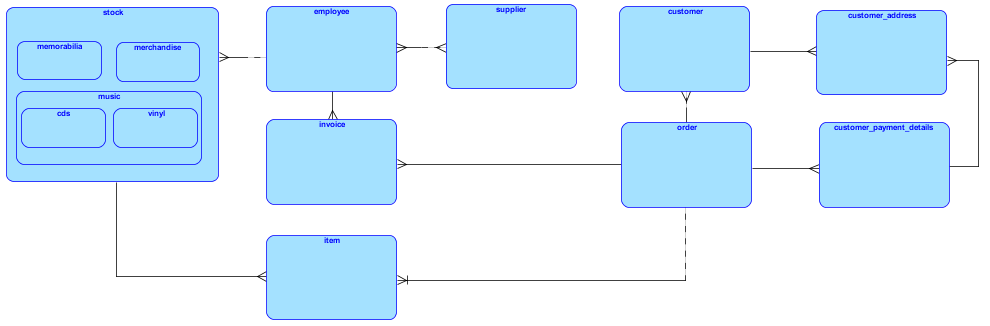
|  |  |  |  |
| --- | --- | --- | --- |
| **Class Number** | | 6. | |
| **Class Name** | | Invoice | |
| **No.** | **Responsibility** | | **Collaborators** |
|  | Generates and stores information into the database | | * Customer * Database * Driver * Graphical User Interface * Item |
|  | Retrieves information form the database when required | |
|  | Displays to the customers screen after transaction and when selected by employees | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Class Number** | | 7. | |
| **Class Name** | | Database Editor | |
| **No.** | **Responsibility** | | **Collaborators** |
|  | Allows the employee to access the database | | * Employee * Database * Driver * Graphical User Interface |
|  | Connects to the database | |
|  | Displays pop up the screen | |

### 1.1.4. ERD

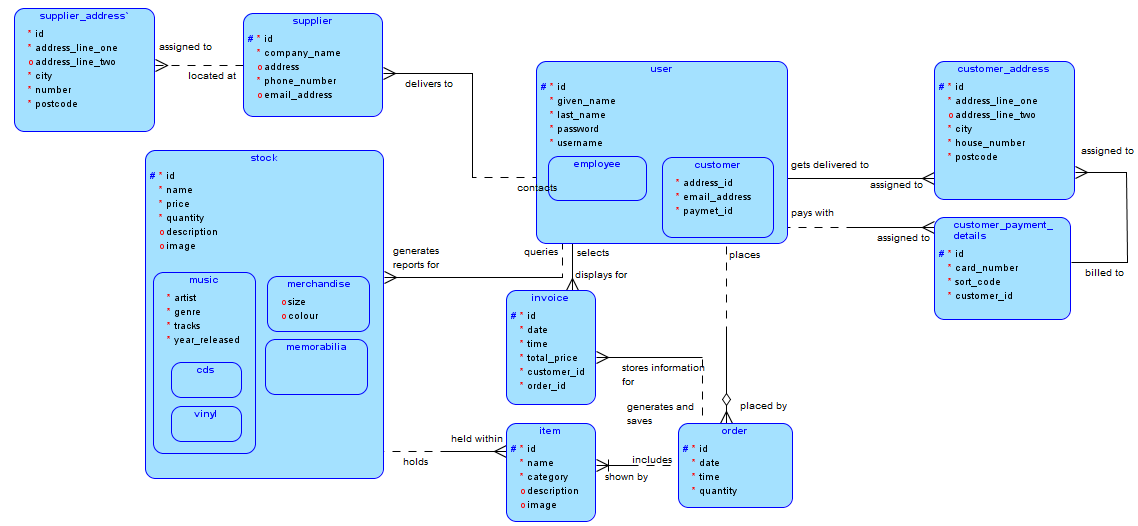
#### 1.1.4.1. Top-Level ERD

Using my entities that I identified using my CRC cards (found in section *1.1.3.1. Database Entities*), I have created a simple top level ERD to display what the database system will look like, work and how each entity connects to each other. I have created a supertype called ‘stock’ which will contain product subtypes, I added this to the diagram to give a better understanding of how products will be stored. I will also be adding another supertype called ‘user’ which will contain the subtypes ‘employee’ and ‘customer’. I have not included this into the diagram as I want to show the relationships the two separate entities have other entities. The reason I am putting these two entities together is because they share the same attributes with only a couple of differences. I will be going into greater detail about the ERD in the next section *1.1.4.2. Fully Detailed ERD* where I have the fully finished and detail ERD to talk about. The initial diagram of the ERD is shown below:



#### 1.1.4.2. Fully Detailed ERD

Below is a diagram of what the fully detailed ERD looks like. The entities shown in the diagram may be subject to change or additional entities might be added when the actual database is created.



##### 1.1.4.2.1. Fully Detailed ERD and Breakdown

In section *1.1.4.2. Fully Detailed ERD* the diagram ERD of the database shows all relationships between each table. Each relationship that is displayed shows the optionality and mandatory status of each relationship. I have also included parent and child tables that use the same entities, this stops the same entity names being duplicated. I have used this for the user table and the stock table which is shown in the above referenced section. I have also displayed all tables along with all entity information. The tables that I have created are shown below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name:** | customer | | |
| **No.** | **Attribute** | **Data Type** | **Opptinaity** |
| 1. | id | NUMBER | Mandatory |
| 2. | given\_name | VARCHAR2 | Mandatory |
| 3. | last\_name | VARCHAR2 | Mandatory |
| 4. | password | VARCHAR2 | Mandatory |
| 5. | username | VARCHAR2 | Mandatory |
| 6. | address\_id | NUMBER | Mandatory |
| 7. | email\_address | VARCHAR2 | Mandatory |
| 8. | payment\_id | NUMBER | Mandatory |
| 9. | phone\_number | NUMBER | Optional |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name:** | customer\_address | | |
| **No.** | **Attribute** | **Data Type** | **Opptinaity** |
| 1. | id | NUMBER | Mandatory |
| 2. | address\_line\_one | VARCHAR2 | Mandatory |
| 3. | address\_line\_two | VARCHAR2 | Optional |
| 4. | city | VARCHAR2 | Mandatory |
| 5. | house\_number | NUMBER | Mandatory |
| 6. | postcode | VARCHAR2 | Mandatory |
| 7. | customer\_id | NUMBER | Mandatory |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name:** | customer\_payment\_details | | |
| **No.** | **Attribute** | **Data Type** | **Opptinaity** |
| 1. | id | NUMBER | Mandatory |
| 2. | card\_number | NUMBER | Mandatory |
| 3. | sort\_code | NUMBER | Optional |
| 4. | name\_on\_card | VARCHAR2 | Mandatory |
| 5. | expirty\_date | DATE | Mandatory |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name:** | employee | | |
| **No.** | **Attribute** | **Data Type** | **Opptinaity** |
| 1. | id | NUMBER | Mandatory |
| 2. | given\_name | VARCHAR2 | Mandatory |
| 3. | last\_name | VARCHAR2 | Mandatory |
| 4. | password | VARCHAR2 | Mandatory |
| 5. | username | VARCHAR2 | Mandatory |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name:** | supplier | | |
| **No.** | **Attribute** | **Data Type** | **Opptinaity** |
| 1. | id | NUMBER | Mandatory |
| 2. | company\_name | VARCHAR2 | Mandatory |
| 3. | address\_id | NUMBER | Mandatory |
| 4. | phone\_number | VARCHAR2 | Mandatory |
| 5. | email\_address | VARCHAR2 | Optional |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name:** | supplier\_address | | |
| **No.** | **Attribute** | **Data Type** | **Opptinaity** |
| 1. | id | NUMBER | Mandatory |
| 2. | address\_line\_one | VARCHAR2 | Mandatory |
| 3. | address\_line\_two | VARCHAR2 | Optional |
| 4. | city | VARCHAR2 | Mandatory |
| 5. | number | NUMBER | Mandatory |
| 6. | postcode | VARCHAR2 | Mandatory |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name:** | order | | |
| **No.** | **Attribute** | **Data Type** | **Opptinaity** |
| 1. | id | NUMBER | Mandatory |
| 2. | date | VARCHAR2 | Mandatory |
| 3. | time | VARCHAR2 | Mandatory |
| 4. | quantity | VARCHAR2 | Mandatory |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name:** | item | | |
| **No.** | **Attribute** | **Data Type** | **Opptinaity** |
| 1. | id | NUMBER | Mandatory |
| 2. | name | VARCHAR2 | Mandatory |
| 3. | category | VARCHAR2 | Mandaroty |
| 4. | description | VARCHAR2 | Optional |
| 5. | Image | BLOB | Mandatory |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name:** | invoice | | |
| **No.** | **Attribute** | **Data Type** | **Opptinaity** |
| 1. | id | NUMBER | Mandatory |
| 2. | date | DATE | Mandatory |
| 3. | time | TIME | Mandatory |
| 4. | total\_price | MONEY | Mandatory |
| 5. | customer\_id | NUMBER | Mandatory |
| 6. | order\_id | NUMBER | Mandatory |

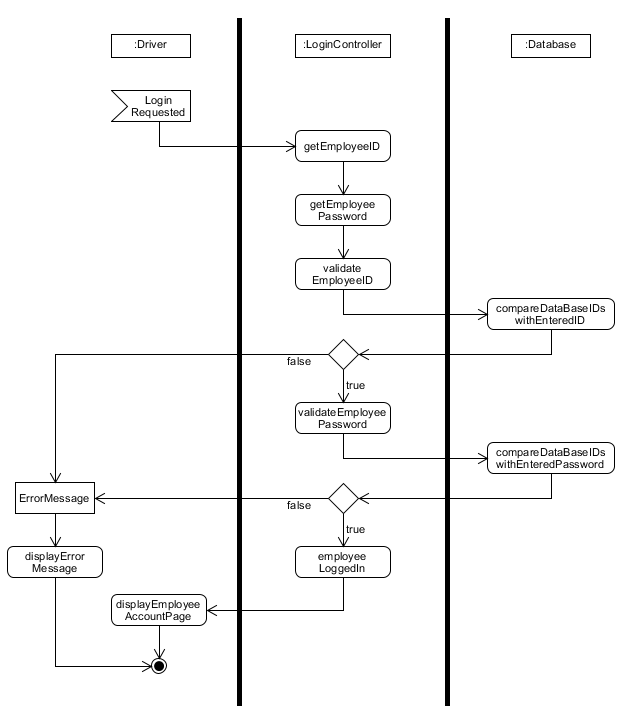
|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name:** | cds | | |
| **No.** | **Attribute** | **Data Type** | **Opptinaity** |
| 1. | id | NUMBER | Mandatory |
| 2. | name | VARCHAR | Mandatory |
| 3. | price | MONEY | Mandatory |
| 4. | quantity | NUMBER | Mandatory |
| 5. | description | VARCHAR2 | Optional |
| 6. | image | BLOB | Optional |
| 7. | artist | VARCHAR2 | Mandatory |
| 8. | genre | VARCHAR2 | Mandatory |
| 9. | tracks | NUMBER | Mandatory |
| 10. | year\_released | DATE | Mandatory |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name:** | vinyl | | |
| **No.** | **Attribute** | **Data Type** | **Opptinaity** |
| 1. | id | NUMBER | Mandatory |
| 2. | name | VARCHAR | Mandatory |
| 3. | price | MONEY | Mandatory |
| 4. | quantity | NUMBER | Mandatory |
| 5. | description | VARCHAR2 | Optional |
| 6. | image | BLOB | Optional |
| 7. | artist | VARCHAR2 | Mandatory |
| 8. | genre | VARCHAR2 | Mandatory |
| 9. | tracks | NUMBER | Mandatory |
| 10. | year\_released | DATE | Mandatory |

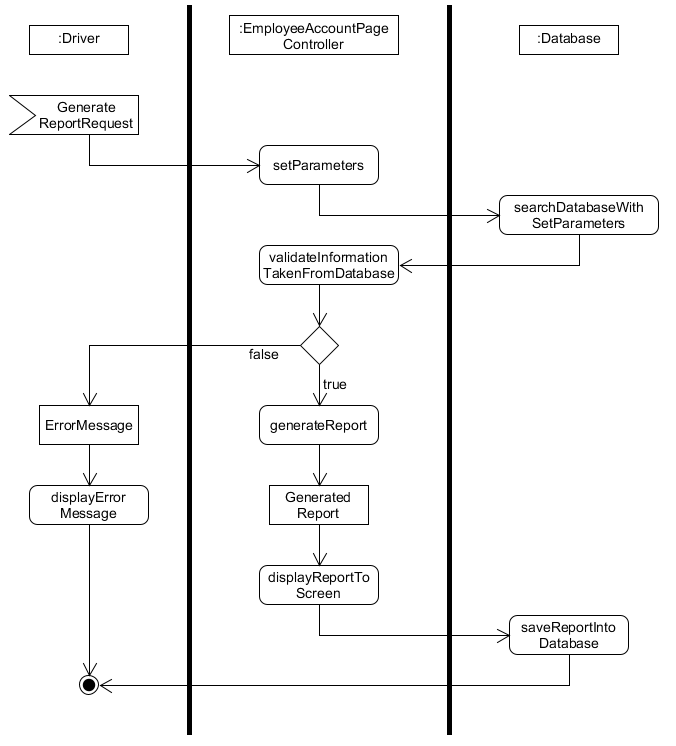
|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name:** | merchandie | | |
| **No.** | **Attribute** | **Data Type** | **Opptinaity** |
| 1. | id | NUMBER | Mandatory |
| 2. | name | VARCHAR | Mandatory |
| 3. | price | MONEY | Mandatory |
| 4. | quantity | NUMBER | Mandatory |
| 5. | description | VARCHAR2 | Optional |
| 6. | image | BLOB | Optional |
| 7. | size | VARCHAR2 | Optional |
| 8. | colour | VARCHAR2 | Optional |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name:** | memorabilia | | |
| **No.** | **Attribute** | **Data Type** | **Opptinaity** |
| 1. | id | NUMBER | Mandatory |
| 2. | name | VARCHAR | Mandatory |
| 3. | price | MONEY | Mandatory |
| 4. | quantity | NUMBER | Mandatory |
| 5. | description | VARCHAR2 | Optional |
| 6. | image | BLOB | Optional |

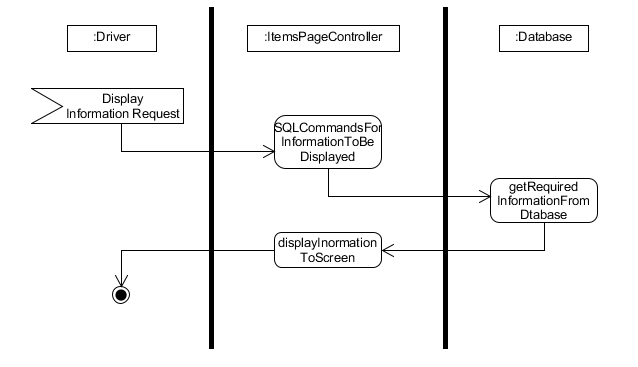
### 1.1.5. Activity Charts

Employee Login  
The first diagram that is show ‘Employee Login’ which shows how the application interacts with the database with the validation of what is entered into the required fields. The diagram shows what happens when valid data is imputed, allowing the user to login to the system and how the application handles invalid data, with the necessary error handling.

Generate Report  
In this activity chart it shows how reports will be generated and displayed to the user. The application firstly prompts for required parameters after being initialized. With the entered parameters, the application access the database and searches with the set parameters and pulls the required information found out of the database. The information is then displayed to the screen in the set report format that will crated to be shown to the user, if the database cannot find anything matching the set requirements, the necessary error handling will be initiated and a error message will be displayed to the screen to alert the user.

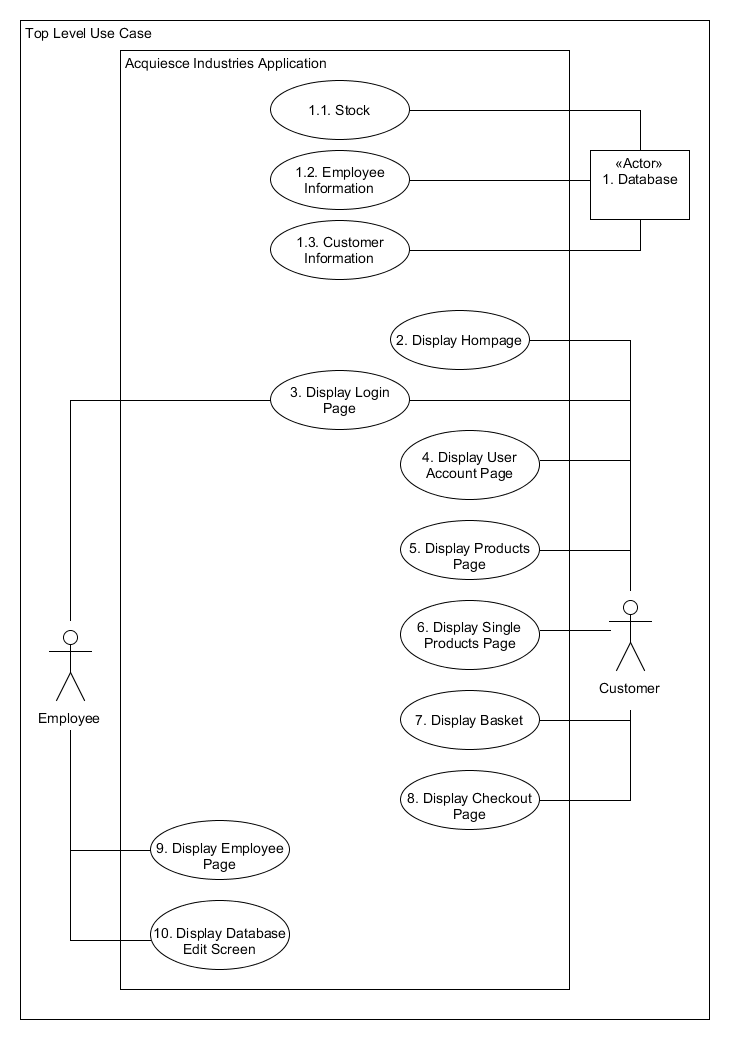


Display Information  
The ‘Display Inoframtion’ chart shows how the application is going to intercat with the database and how it will pull out required information and display it were required. This will be used everthing that had to be displayewd onto the screen when called up. SQL commands will mainly be used to call required information from the database.



### 1.1.6. Top Level Use Case Model

### 1.1.6.1. Initial Top-Level Use Case Diagram

In the first stage (Inception Phase) is created a simple top-level use case diagram of how the system will be structed and how it may look like. I went into greater detail of how the model will work in the previous inception stage. I am going to continue to use this diagram and build upon it for the next stage. Shown below is my top-level use case model that will be getting used for further development:

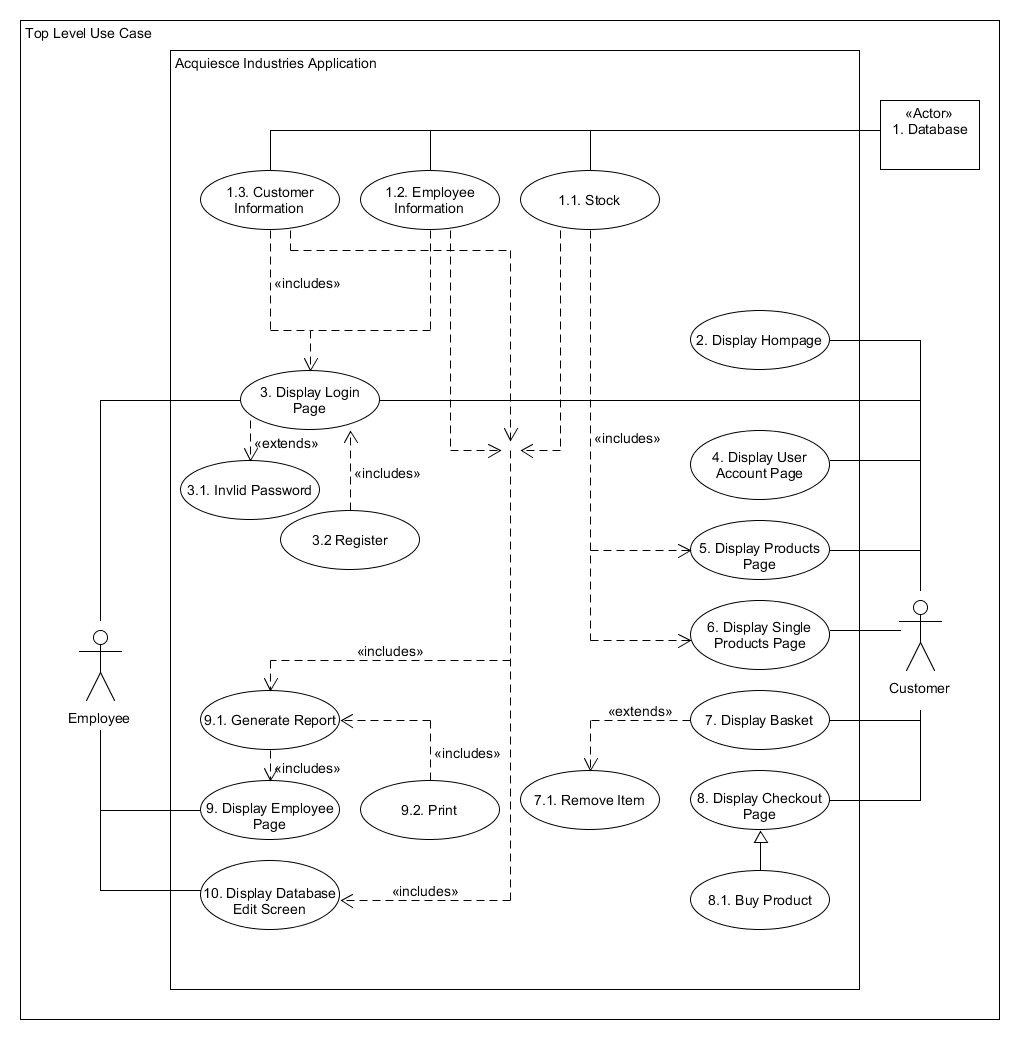
### 1.1.6.2. Next Stage Top-Level Use Case Model

I have taken my first top level use case diagram and produced a more technical top-level use case model. I have added additional functionality by adding includes and extends where necessary. I have tried to keep the diagram simple as well as including a high amount of detail as possible. Although parts of the diagram look messy, it is very easy to understand navigate. I will explain certain areas of the diagram that may cause confusion with overlapping of lines and arrows.

The first area of the diagram I am going to breakdown is **Use Case 3 – Display Login Page**. This use case has three includes connected to it: **Use Case 1.3 – Customer Information**, **Use Case 1.2 - Employee Information** and **3.2. Register**. Use case **1.2** and **1.3** are taken from the database actor and are needed for logging into the system, all login credentials as stored within the database and are used in this use case. The other include of **3.2** allows a user to register a new account through this use case, it is included in the process of logging into the system. This use case also includes an extend: **3.1 Invalid Password**, this function extends the use cases functionality whenever invalid data has been entered into the system.

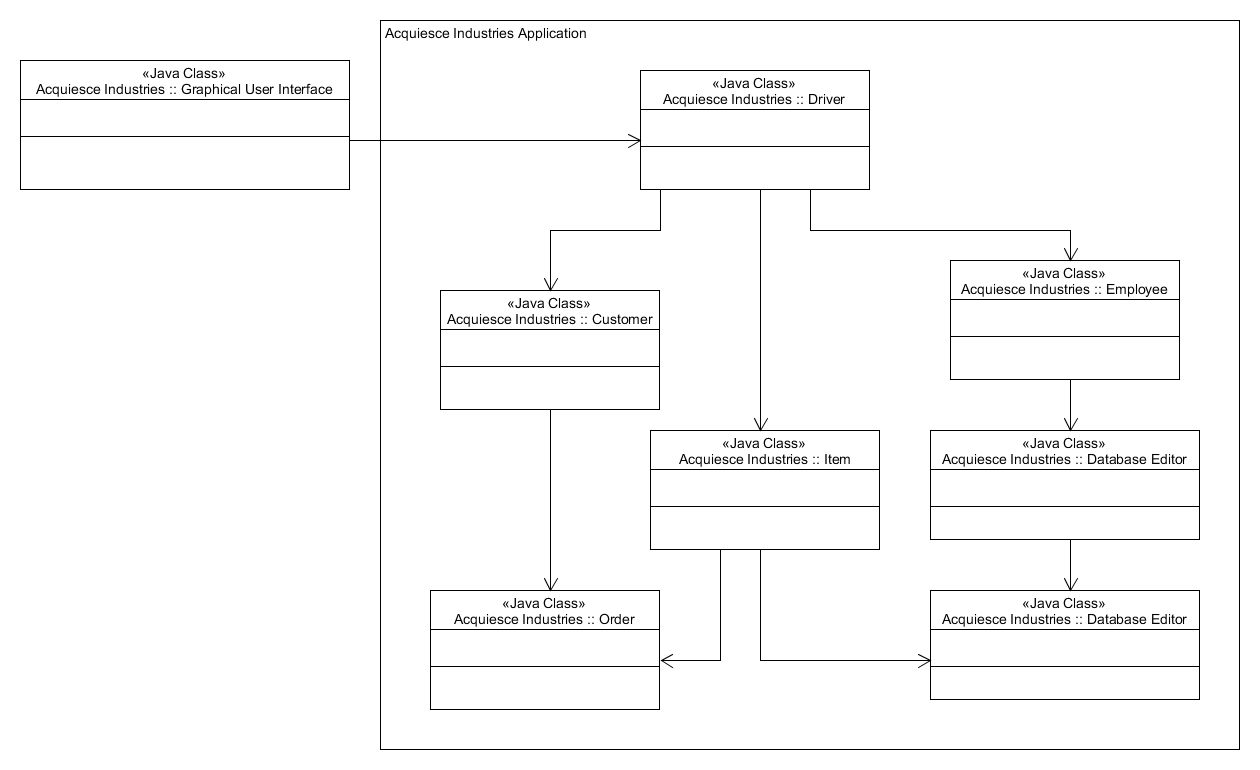
The second area I want to discuss about the diagram is everything that is going on with: **Use Case 9 – Display Employee Page** and **Use Case 10 – Display Database Edit Screen**. Use case **9** has an include called **9.1 Generate Report** which has all use cases from the actor **1. Database** (Use Cases included - **1.1, 1.2, 1.3**) included into it to allow different types of reports to be generated based on information being stored within the database. There is also an additional include connected called **9.2 Print** which will allow the reports that have been generated to be printed out as paper copies. Use Case **10** also has all use cases from the actor **1. Database** (Use Cases included – **1.1, 1.2, 1.3**) which allows access to the database, enabling the ability to edit the database (add and remove items).

The third and final area that I am going to talk about and explain is to do with: **Use Case 5 – Display Products Page**, **Use Case 6 – Display Single Product Page** and **Use Case 7 – Display Basket**. Use cases **6** and **7** both have the same include, they both have **1.1 Stock** connected to them. They both have this because all products are stored within **1.1** in the database. These use cases take the required products/information and display it to the screen in the correct areas. Use case **7** only has one extend connected to it: **7.1 Remove Item**. This has been added in the case that the user wishes to remove an item form their basket before checking out. This is an additional function that needs to be carried out, hence why it has been added as an extended feature.  
  
Within section *2.1.2.1. Use Case Models* I discuss and go more in-depth about each use case and how they will function within the system. It will also give a clearer understanding of what the system is carrying out at each stage.  
  
I have created this next stage use case model and included all these changes; this is shown within the diagram below:



### 1.1.7. Initial Class Diagram

I have created a top-level class diagram for this section. I have crated this because it gives me starting point and a greater understanding of what is required to create the program. With the help of my CRC cards (*1.1.3.1. Classes*) I able to use the classes that I come up with to create this diagram. I will be using this diagram to build upon and later create a fully detailed class diagram. The initial diagram will only be displaying the class and how they connect to each other, I will be going into further detail with the fully finished detailed diagram which is found in section *2.1.1. Static Model*. The top-level class diagram is show below:



I have also included additional classes that weren’t produced in CRC cards. I added these as it makes the program make more sense, having these additional classes could help with production of the application instead of having multiple functions held within the one class. This is only the initial top-level diagram so it may be subject to change.

## 1.2. View Model

### 1.2.1. Identification of Data Binding Model

For my data binding model, I intend to use a database for the development of my application. I have chosen this because I want to have a database connected to the backend of the application, doing so will allow the use to interact with the database though the use of the application. The database will have the ability to display information to the screen whenever called upon to do so, the user will be able uses the application to carry out these commands. The application will pull out required information stored within in the data, store it temporary within the application and then display that information to the screen for the user to see. Also, the database will have the ability to save data and information that has been directly inputted by the user through the application.

### 1.2.2. User Analysis

In the Inception Phase, I created some initial wireframes of what the application may look like that were presented to the client. After having a short discussion with the client and asking a few questions (shown in section Second Formal Meeting with Client and Questionnaire), I was able to get feedback and clarification on what the client thought of the layout and designs. I was informed that the designs I produced are what the client is looking for and are pleased to allow me to continue and elaborate on the current designs in place. I also have a working pdf moqups prototype which will be presented to the client for further discussion and evaluation of the design stage. I will be showing the initial wireframe designs with more in-depth details in the next section *1.2.2.1. Initial Wireframes*.

#### 1.2.2.1. Initial Wireframes

|  |  |
| --- | --- |
| **Wireframe No.** | **Wireframe Name** |
| 1. | Homepage |
| **Description** | |
| The first wireframe I am showing is the homepage, as it will be the first page that will displayed to the screen when the application is started up. The wire frame shows what the layout of the page will look like, following the requirements from the brief and client input, I went with a simple and organised design and layout, this making it easy for the user to navigate through the application.  The menu bar located at the top of the screen will be present on all pages throughout the application, on certain pages there will be only be certain options displayed in the menu depending on what pages is being displayed to the screen. I have decided to go with this idea as it makes the navigation through the application a lot easier, the user will only have to click buttons that will take them to their required page. Within the menu, I have implemented a dropdown feature for music, I have done this because it saves space in the menu and makes sense to have music items stored together.  Displayed on the page will be clear image related to the application and a small paragraph describing and explain the purpose and some additional information about the application. I have also included a ‘Featured Music’ section located at the bottom of the page, I still need to discuss this with the client if they would like this featured implemented or changed to something else. The featured music section would display the images of the albums along with the name of the album and the artist. The album images displayed will also be clickable links that will redirect the user to that album’s product page. This section doesn’t need to be featured music, it could also change to ‘New Albums’ or ‘Best Selling Albums’ for example. This would come down to the client decision on what they would like to see. | |
| **Image** | |
|  | |

|  |  |
| --- | --- |
| **Wireframe No.** | **Wireframe Name** |
| 2. | Merchandise and Memorabilia Product Page |
| **Description** | |
| This wireframe is what is displayed when the merchandise or the memorabilia buttons within the menu are clicked. I have only created the one wire frame for these two pages as they will be displaying products the exact same way but with the different product types for each page. All products will be displayed in a list format with a scroll bar to scroll down the page to view more products. With all products, there will also be an image of the product, a small description of what the product is and the name of the product. The name and the images will be clickable, allowing the user to be redirected to that products item page which is shown in **Wireframe No.5 – Single Item Page**. Each product that is being displayed will be evenly laid out to make it easier to understand and read the information presented to the user on the screen. | |
| **Image** | |
|  | |

|  |  |
| --- | --- |
| **Wireframe No.** | **Wireframe Name** |
| 3. | Music Page |
| **Description** | |
| The music page wireframe shows what will be displayed when either the ‘CDs’ or ‘Vinyl’ button is clicked in the music down menu. Both pages (CDs and Vinyl) will have the same layout and design. Just different type of form of music. This page will include the ‘Featured Music’ just like the Homepage, which may be subject to change (see **Wireframe No. 1 – Homepage** for additional details).  All albums will be displayed with an image, artist and name of the album. This will all be clickable and link to the corresponding albums single products page, the design for that page is shown in **Wireframe No.5 – Single Item Page**. The layout of the page will be very simple, each item will be displayed in grind like layout, making it very clear and easy to read each item that is being display. There will also be different sections the page, which will make it easier to fins the album the user is looking for. The genre section will take the user to a different page which will only be focused on the genre selected (**Wireframe No.4 – Genre Page**).  I have also come up with other ideas that will not be implemented in this prototype but may be used in the future. The implementation of a search feature. This would help with user navigation instead of manually looking through every item, they could just search for that item and have it displayed to the screen. | |
| **Image** | |
|  | |

|  |  |
| --- | --- |
| **Wireframe No.** | **Wireframe Name** |
| 4. | Genre Page |
| **Description** | |
| The genre wire frame is showing the layout of what will be displayed when a genre has been clicked on by the user. All music that is held within that specific genre will be displayed to the screen in the same format as **Wireframe No.3 – Music page**, as well as everything on the page being clickable like the Music Page. When clicked on, each item will be linked to their own product page (**Wireframe No.4 – Single Item Page**) where more details and the option to buy will be displayed. | |
| **Image** | |
|  | |

|  |  |
| --- | --- |
| **Wireframe No.** | **Wireframe Name** |
| 5. | Single Item Page |
| **Description** | |
| The single item wireframe shows what the user will be displayed with when they click on a product. The page will display the image of the item and the name along with additional information and details is applicable. This page is where the user can add the item to their basket if they wish to buy the item. When the ‘Add to Basket’ button is clicked, the basket menu button will update depending on how many items are currently in the user’s basket (**Wireframe No.6 – Users Basket Page**). There is also a back button included on this page, this allows the user to go back to the previous page they were on.  All items (music, merchandise and memorabilia) will have the same single item page, they will all include the same clean and simple layout with just the one item being shown. Another idea that could be implemented on this screen could be similar products be advertised at the bottom of the screen. This will not be implemented in the current prototype at this stage, but maybe in future designs. | |
| **Image** | |
|  | |

|  |  |
| --- | --- |
| **Wireframe No.** | **Wireframe Name** |
| 6. | Users Basket Page |
| **Description** | |
| This wireframe shows what the users basket may look like when the ‘Basket’ menu button is clicked. The user will be presented will a list of the item(s) that they have added to their basket. There will also be ‘Remove Item’ buttons along side each item, which will remove that item from the users basket and update the ‘Basket’ button in the menu to represent the new number of items in the users basket. If the user is happy with the items that they currently have, they have the option to select the ‘Checkout’ button at the bottom of the page. This button will only be available when there is at least one item present in the basket. When clicked, the user is redirected to the checkout screen where they are shown the total price, their payment details, the items they are purchasing and the option to cancel the order or complete the transaction (this is shown in **Wireframe No.7 – Checkout Page).** | |
| **Image** | |
|  | |

|  |  |
| --- | --- |
| **Wireframe No.** | **Wireframe Name** |
| 7. | Checkout Page |
| **Description** | |
| This is the last page that the customer will see when they complete their order. The checkout page will display the items they are about to purchase, their payment details and the address the items will be delivered to. If any information is incorrect, the user has the option to change their details with the click of the edit information button which will take the user to their account page where they can update and change their details (**Wireframe No.10 – Customer Account Page**). To complete the transaction, all the user must do is click the place order button, the order is placed, and an invoice is then generated for that order. The invoice is displayed to the screen for the customer to see and is also stored into the database. An example of what an invoice is shown below along with the customer checkout page wireframe. | |
| **Image** | |
|  | |

|  |  |
| --- | --- |
| **Wireframe No.** | **Wireframe Name** |
| 8. | Login Page |
| **Description** | |
| The login page will be used by the customer and employee to login into the system. The two different account types will be directed to different pages in the application, customers will be directed to the homepage (**Wireframe No.1 - Homepage**) and employees will be shown their accounts pages (**Wireframe No.11 – Employee Account Page**). If a user were to forget their password, they can select the forgotten password link which will show a page that will allow the user to change their password and enable them to login. I have made this page very simple as there is no need to have a lot on this page, making it very easy to understand and carryout actions. I the user doesn’t currently have an account, they have the ability to register for an account by clicking the register button. The button will take the user to the register page where they can input their details and create an account (The register page is show in **Wireframe No.9 – Register Page**).  If the user were to enter any invalid log in details, the system will display an error message stating invalid input has been entered and ask the user to re-enter their details. | |
| **Image** | |
|  | |

|  |  |
| --- | --- |
| **Wireframe No.** | **Wireframe Name** |
| 9. | Register Page |
| **Description** | |
| The register page is what is shown when the register button is clicked on the login page (**Wireframe No.8 – Login Page**) and allows the user to create a ne account on the system. The user must fill in all mandatory (input boxes with \*) fields with valid information to allow them to proceed in creating their account. After the account has been created the user can now log into their new account and purchase items.  I have split up all required information in sections, making it easier for the user to know what information is required and what different sections need to be filled out. | |
| **Image** | |
|  | |

|  |  |
| --- | --- |
| **Wireframe No.** | **Wireframe Name** |
| 10. | Customer Account Page |
| **Description** | |
| This wireframe is what the customer sees when they click on the account button with in the menu. The page displays the customers information and the customers order history with the user of the generated invoices that link to the customer (an example of the invoice is show in **Wireframe No.7 – Checkout Page**). The customer will also have the ability to change their details here with the use of buttons and links, the wireframe does not currently show all features that would be implemented but they will look very similar to the Change Password’ link that is currently featured. There is also a return home button which will take the user back to the homepage when clicked (**Wireframe No.1 -Homepage**). | |
| **Image** | |
|  | |

|  |  |
| --- | --- |
| **Wireframe No.** | **Wireframe Name** |
| 11. | Employee Account Page |
| **Description** | |
| When an employee logs into the system, this is the first page that will be displayed to them. Their details will be displayed along with a ‘Change Password’ link, the same as the customers. But employees have the option to generate reports using parameters that they set, they also have the ability to print the report if they wish. Once generated, the report will pop up onto the screen with all required information, this is shown below along with the employee account page wireframe. The headings and information will change depending on what parameters are set by the employee. Another feature employees have is the ability to edit the database, a pop up appears and allows the employee to add or remove items depending on their selection and input. These are shown in **Wireframe No.12 – Edit Database pop up**. The final feature that is available for employees is that they can view all generated invoices that are currently stored in the database. The invoices will be displayed to the screen as a pop up, similar to how the report pop up is presented by with different titles and information. | |
| **Image** | |
|  | |

|  |  |
| --- | --- |
| **Wireframe No.** | **Wireframe Name** |
| 12. | Edit Database Pop up |
| **Description** | |
| This is what is displayed to an employee when they click on the edit data base button in **Wireframe No.11 – Employee Account Page**. The two options that are shown ‘Add’ or ‘Remove’ display different pop ups. The ‘Add Item’ pop up allows the user to enter information on what they want to add to the data base, all fields must be filled with valid data otherwise the employee with be unable to add the data into the database. The ‘Remove Item’ pop up will allow the employee to search the database using either an id or the name of an item (this could change to where the employee can select what to search for with the use of a dropdown menu list that can be selected from) and delete it from the database. The employee also has the option to cancel the process at any point on ether pop up with the click of the cancel button which will take the employee back to the ‘Edit Database Pop up’. This wireframe is only a very basic input screen and will be worked on in the near future closer to the development stage. | |
| **Image** | |
|  | |

#### 1.2.2.2. User Usability

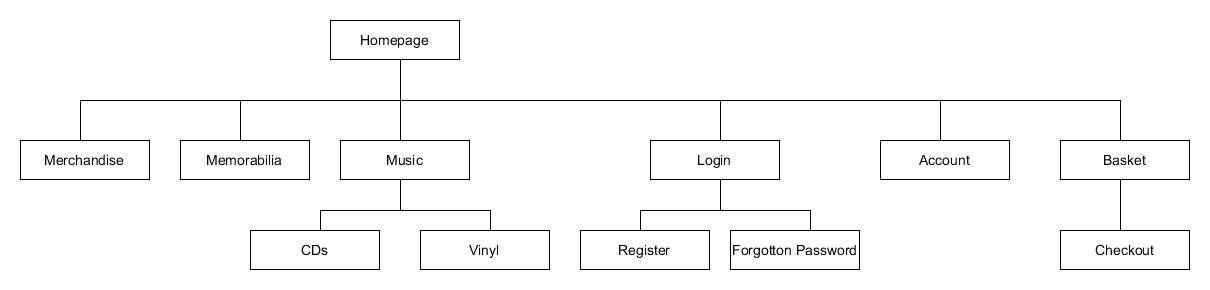
##### 1.2.2.2.1. Applications Navigation

I have planned out the application with the user in mind. I have tried to make the application as simple as possible so that anyone can use it and easily understand what to do and where to go. I have enforced the fact that everything that is displayed to the screen must be laid out correctly, this make it easier for users to navigate throughout the application and find what they are looking for.

The user will have the ability to click almost everything that is going to be displayed to the screen. All text and images that show products will be clickable and take the user to the products page. Each page will be titled with the current page name to help the user identity where they are in the application, the menu bar will also highlight to the current corresponding that the user is on the further help them understand where they currently are. The will also be dropdowns implemented when applicable in the menu, the music menu will the main one in the menu bar, which keeps the menu looking clean, simple and not too crowded. This can be seen in *1.2.2.1. Initial Wireframes* – *Wireframe No.1 – Homepage*.

My reasoning for these decisions is to keep the application very simple and easy to use but at the same time have a professional layout and feel. With the use of the menu bar being displayed on almost all screens, this helps hugely with the navigation around the application.

###### 1.2.2.2.1.1. Navigation Map

I have created a simple navigation map from the customers perspective to help with the understanding of how the application will be navigated and to show how different pages connect to each other. The navigation map is shown below:

##### 1.2.2.2.2. Colour Scheme

I began by carrying out research on how to use colours for colour blindness. I found a very god website which gave me a greater understanding and knowledge of what I can create that will help with the effect. The website is studied was (Usabilla.com, 2019) which got me thinking about how I want to carry out my colour scheme design.

My first idea was to create a darkish themed colour scheme, this would not only help with the effect of colour blindness, it also fits the theme of music, gives the application a rock and roll theme which I am looking to achieve. With the use of dark colours, this reduces the contrasting of colours which a lot of colour-blind users or even users with sight problems can find very uneasy and stressful when trying to read.

I have decided to go with only 2-3 colours for the colour scheme as I know that having to many colours on the one screen may produces stress on the eyes and makes it hard to read for some user. This will make the application look more professional with the use of the same colour scheme throughout the whole application. I created a fully coloured moqup of what the homepage may look like with the colour scheme that I have produced at this current stage; the example is shown below:

I received feedback from the client that they like how the colour scheme looks (Second Formal Meeting with Client and Questionnaire) and they stated how the colours help with colour blindness. I chose the three colours – Black, White and Red because all three work well together. I wanted to pick colours that aren’t too bright or over the top, hence the dark theme creating a simple as well as a professional layout.

In the next stage of the project I am going to be considering the ability to change the colours of the application to suit users that may have sight problems and have colours that are more comfortable for them to use. I will need to carry out further research and collected different types of colours sets that could be used.

# 2. Evidence for Analysis

## 2.1. Business Model

### A close up of text on a white background Description automatically generated2.1.1. Static Model

As I discussed in section *1.1.7. Initial Class Diagram*, I am going break down and discuss what is going on in the above diagram and the though process I went through to come up with the initial designs. I have decided to split the diagram into three different parts: Database (on the left-hand side), User (down the middle) and Stock (on the right-hand side). As I said in the previous section, this is only the first diagrams for the prototype so not all of the classes that are required for a fully functional application are displayed in the diagram, I have only chosen to include the class that are need to have a functional prototype application.

The first section I am going to discuss is the required database classes. These classes are used to establish the connection to the required database, the connection classes solely purpose is to find and connect to the database. The database manager will take care of any required error handling that may occur in the process of connecting to the database. The manager will also be used to control the login process, where it must communicate with the database to match the correct credentials, this is where the SQL queries are held for the login.

The second section that I am going to talk about is the user classes. This is where the employees and customers will be created, called and stored into the database. Both employee and customer use basically the same values with slight differences which is shown with the customer class where it holds more methods than the employee class. Both classes serve the same purpose but will different user types, they both user getters and setters to take in information then assign and stored the given information in the correct location within the database.

The third and final section I am going to talk about is the stock section. In this section, all stock type will be stored under ‘Stock’ as the all share similar values. I have done this as it stops the duplication of declaring variables in each of the classes. All these classes carryout the same process but with different values being passed through and set. All classes use getters and setting to get the required information and set the received information into the database in the correct required stock type.

### 2.1.2. Dynamic model

#### 2.1.2.1. Use Case Models

I have produced use case models on some of the key features that need to work in order for the application to run correctly. I have used the top-level user case diagram found in section *1.1.6.2. Next Stage Top-Level Use Case Model* to create these separate diagrams. I have practically dissected each stage that I need to have functional for the prototype of the application to work. I will have a small discussion and explanation of each user case model which are shown below:

A picture containing text

Description automatically generatedLogin

For the login use case, I have decided to show the customer and employee in the one diagram as they both go through the same process to login into the application. The login system doesn’t require much as it’s only checking the inputted username and password with the saved values stored within the database. If invalid credentials are entered, the system will display an error which is shown above with the ‘extends’.

A close up of text on a white background

Description automatically generated Generate Report

For the diagram of ‘Generate Report’, I have included all the different type of reports that will be present for generation in the applications prototype. Each report will be carrying out the same actions of pulling a tables information out of the database using SQL commands. Each report generated will display their own headings and each have different formats ready for printing.

A close up of a piece of paper

Description automatically generatedUpdate

The update model shows the process of how the application will deal with the updating of stock held and other information stored within the database. The process is very simple with the different categories that can be updated linked to the required table in the database. When selected, the user will be prompt to enter the ID and the new information that will be prompted to enter. Each category will display different input boxes depending on what category the user selects (for example: employee information will prompt the user for given name, last name, etc… while stock will ask what type of stock (CD, Memorabilia and Merchandise) that needs to be updated and depending on what is selected the user will be prompt with price, quantity, etc…). After the update process is complete, the database saves the changes and is ready to display the new additions that have been added/updated.

A close up of a piece of paper

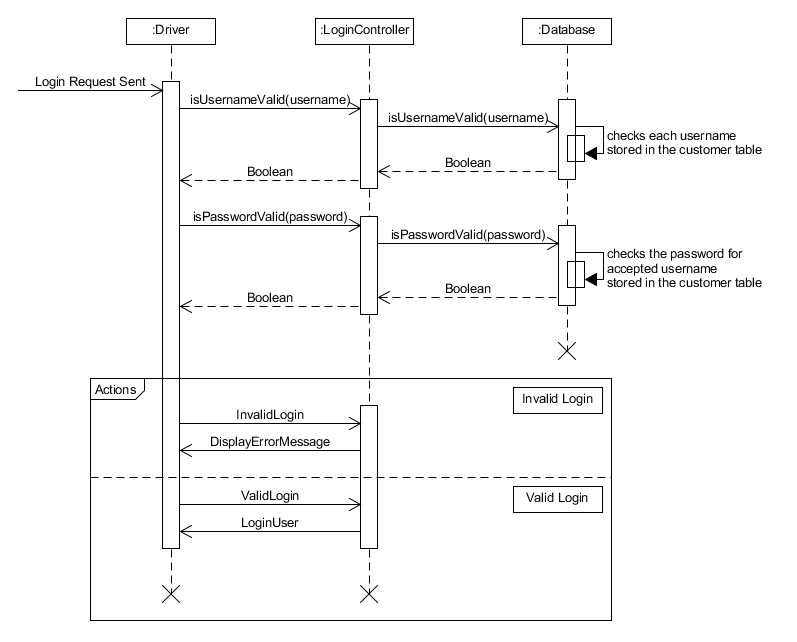
Description automatically generatedPrint

The print use case model is very simple as all it does is check is a printer is available when the print feature has been initialized, if a printer is available the process will work, and the required report will be printed out. On the other hand, if an error were to occur or no printers are available, an error message will occur address the problem to the user which is shown in the above diagrams extends statement.

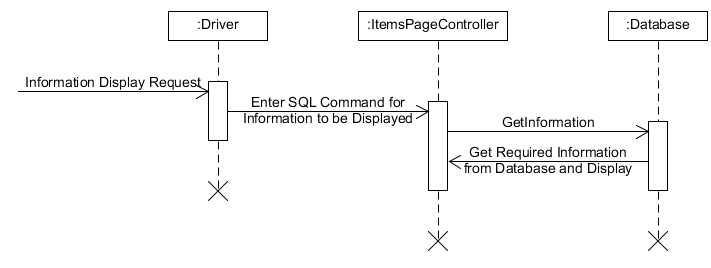
#### 2.1.1.2. Sequence Diagrams

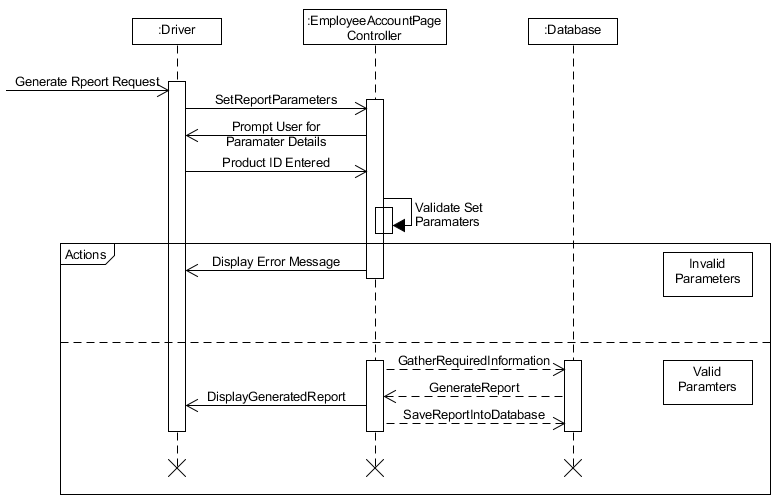
The sequence diagrams that I have chosen to create some of the most important functions to the application that must function on order for the application to work correctly. Each diagram is very self-explanatory with each heading for every action easy to understand and read what is being carried out.

Customer Login  
This diagram shows the interaction between the two required classes and how the application will interact with the database, how and what information will be pulled out of the database. Employee logins will be the same but the table in the data base will be different, the employee table will be searched and checked.

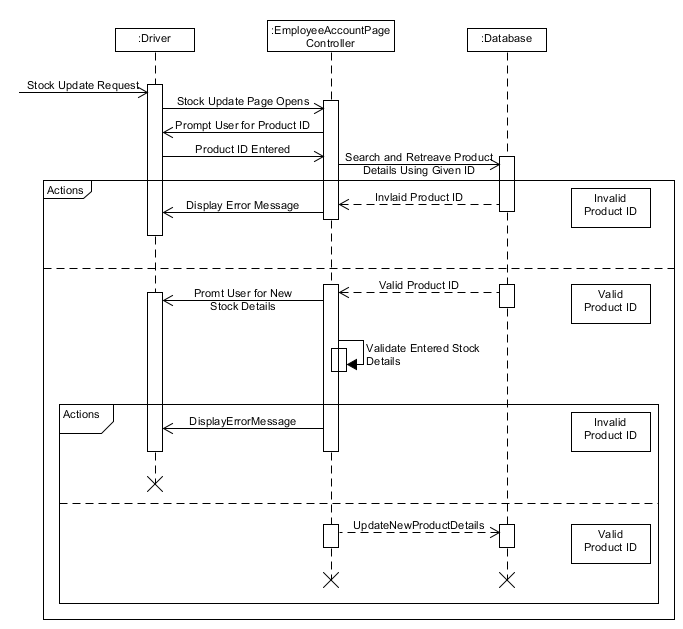


Display Information  
In the application, majority of information required for products including images will stored within the created database. SQL commands would be the most sufficient way to call and display required information to the displayed page in the application.



Generate Report  
For the application to generate a report, parameters must be set which will be passed into the data base to be used as a search (SQL WHERE command) to find the required data. This diagram shows what steps will be in place and what will be carried out to pull the required data out of the database and displayed to the screen in a report format. It also displays the required error handling in place if the parameters are invalid to the databases held data.

Updating Stock  
The updating stock diagram shows that the user has to only enter the product ID to have the ability to update their required item. This process is showing how the application asks the user for input and finds the required products. Their will also be necessary error handling if required which is shown in the diagram and what the outcome will be.



## 2.2. View Model

### 2.2.1. UI Design

As I discussed in section *1.2.2. User Analysis* about how I will be presenting the user interface. As I mentioned in that section, I have created wire frames in section *1.1.6.1. Initial Top-Level Use Case Diagram* where I have gone into great detail about how the application will look and I have also talked about the user interface in the previous documents *Stage One - Inception Phase Report* where I briefly discussed what the interface may look like with the subject to change in the future.

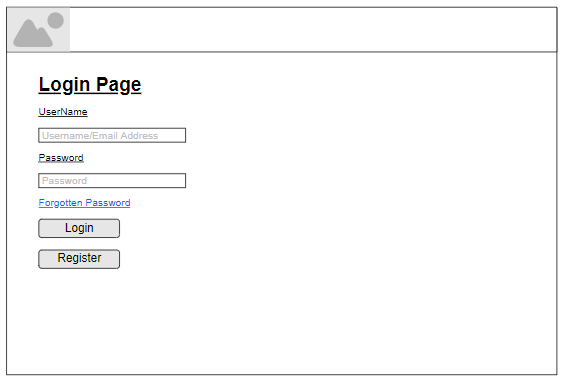
The wireframes that I created are pretty much the final design of what the application will look like. I have also thought of different ways of how I will be able to implement the interface. When implementing the design, I am going to slightly change how the application will work and the layout of the pages. Instead of having a pop-up screen for the generation of reports, I am going to have it placed on the employees accounts page making the layout a lot cleaning rather than have the page clustered with too many buttons. Other than that main change that I plan on implementing, everything else that I have discussed and shown will likely stay the same of the design documents. With this also being a prototype, I also plan on only creating the important aspects of the application that the employee account needs to use and access, so some features that I have discussed will not be implemented in this current build of the application.

### 2.2.2. Data Binding Design

With the use of the data table that I previously created to show all entities stored within the database, I have created diagrams displaying how the data binding between the application and the database will work. The diagrams will be displaying where user input is required and where required information needs to be pulled out of the database.

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name:** | employee | | |
| **No.** | **Attribute** | **Data Type** | **Opptinaity** |
| 1. | id | NUMBER | Mandatory |
| 2. | given\_name | VARCHAR2 | Mandatory |
| 3. | last\_name | VARCHAR2 | Mandatory |
| 4. | password | VARCHAR2 | Mandatory |
| 5. | username | VARCHAR2 | Mandatory |

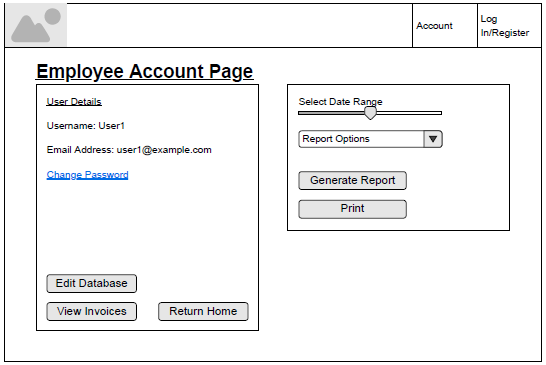
Login Page



With the login screen, it only requires two entitities – username and password. This will be the same for customer logins using the customers table. The user is required to enter their log in details(in this instance an employee is logging in), where the program then takes the entered values and searches the database for the corisponding values. If the details that have been entered match and are valid, a successful loggin will occur and the user will be logged into the application. If there is no match of data and the entered details are invalid, an error message will be displayed to the screen stating that invalid details have been entered and to re enter their details.

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name:** | employee | | |
| **No.** | **Attribute** | **Data Type** | **Opptinaity** |
| 1. | id | NUMBER | Mandatory |
| 2. | given\_name | VARCHAR2 | Mandatory |
| 3. | last\_name | VARCHAR2 | Mandatory |
| 4. | password | VARCHAR2 | Mandatory |
| 5. | username | VARCHAR2 | Mandatory |

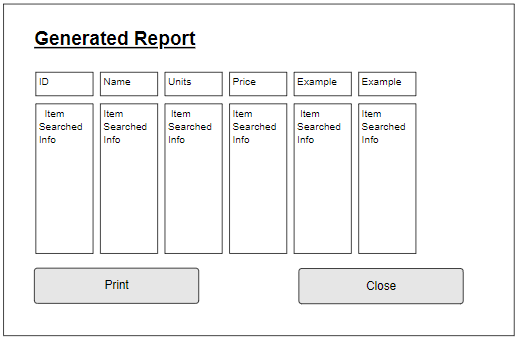
Employee Account Page



On the employee account page only a few entites are required for the display of the employees details. I plan on chaning the design sightly to show the employees full name and username, rather than the email being displyed. The layout of how the reports are generated and shown will be disffrernt, I plan on having the database viewer on the page rather than having it on a separate pop up screen when a button is clicked.

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name:** | cds | | |
| **No.** | **Attribute** | **Data Type** | **Opptinaity** |
| 1. | id | NUMBER | Mandatory |
| 2. | name | VARCHAR2 | Mandatory |
| 3. | price | VARCHAR2 | Mandatory |
| 4. | quantity | VARCHAR2 | Mandatory |
| 5. | description | VARCHAR2 | Optional |
| 6. | image | BLOB | Mandatory |
| 7. | artist | VARCHAR2 | Mandatory |
| 8. | genre | VARCHAR2 | Mandatory |
| 9. | tracks | VARCHAR2 | Mandatory |
| 10. | year\_released | VARCHAR2 | Mandatory |

Generated Report



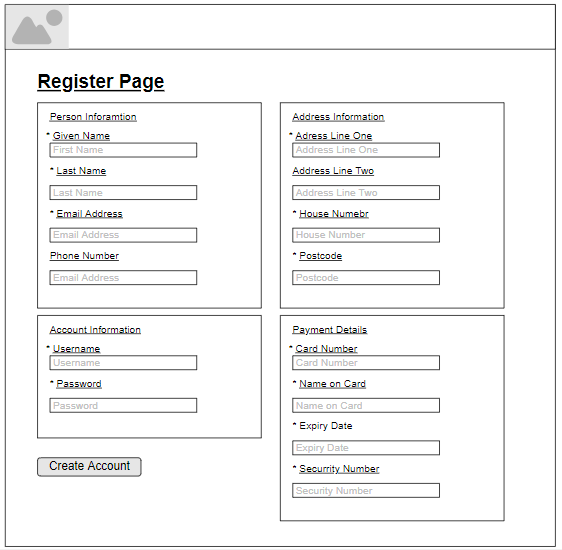
For the generation of reports, I have used the cds table as the example of how the data will be displayed to the screen. This format will be the same with all other tables that will have the ability to be generated into a report. When a paramter is set, the applcaition will connect to the database and run the set SQL query that goes with the parameter. Each tablw that will be displayed will be displayed with its own headings, the same as it’s displayed in the database. My plan for displaying the reports was going to be on a separate pop up screen when the user clicks a button after setting the paramters. I have revised my options for this and decided that I will have this inclcued in the employee account page. I have done this as will look a lot nier having a dropdown option for all avaliable tables instead fo having to change and set paraters, all the user is required to do is set an option and click a button to display the required table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name:** | customer | | |
| **No.** | **Attribute** | **Data Type** | **Opptinaity** |
| 1. | id | NUMBER | Mandatory |
| 2. | given\_name | VARCHAR2 | Mandatory |
| 3. | last\_name | VARCHAR2 | Mandatory |
| 4. | password | VARCHAR2 | Mandatory |
| 5. | username | VARCHAR2 | Mandatory |
| 6. | address\_id | NUMBER | Mandatory |
| 7. | email\_address | VARCHAR2 | Mandatory |
| 8. | payment\_id | NUMBER | Mandatory |
| 9. | Phone\_number | NUMBER | Optional |

Register Page

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name:** | customer\_address | | |
| **No.** | **Attribute** | **Data Type** | **Opptinaity** |
| 1. | id | NUMBER | Mandatory |
| 2. | address\_line\_one | VARCHAR2 | Mandatory |
| 3. | address\_line\_two | VARCHAR2 | Optional |
| 4. | city | VARCHAR2 | Mandatory |
| 5. | house\_number | NUMBER | Mandatory |
| 6. | postcode | VARCHAR2 | Mandatory |
| 7. | cts\_id | NUMBER | Mandatory |

|  |  |  |  |
| --- | --- | --- | --- |
| **Table Name:** | customer\_payment\_details | | |
| **No.** | **Attribute** | **Data Type** | **Opptinaity** |
| 1. | id | NUMBER | Mandatory |
| 2. | card\_number | NUMBER | Mandatory |
| 3. | sort\_code | NUMBER | Optional |
| 4. | name\_on\_card | VARCHAR2 | Mandatory |
| 5. | expirty\_date | DATE | Mandatory |



The register page requires multiple tables that will take in and store information entered by the user. I have displayed the three different tables that will be used when a new account is created by a new user, I have colour coded each table to match up with the moqup image above. All info that is required to be entered by the user will added to the database using SQL insert queries. This is how customer accounts will be created and stored. Employee accounts might be crated like this in the future but for the protototype the register feature might not function and the only way that accounts will be through the update feature on the account page.

# 3. Information Sources

Agilemodeling.com, 2019. *agilemodeling.com - CRC Cards.* [Online]   
Available at: http://agilemodeling.com/artifacts/crcModel.htm  
[Accessed 28 02 2019].

Contrib.Andrew.cmu.edu, 2019. *contrib.andrew.cmu.edu - Natural Language Analysis.* [Online]   
Available at: http://www.contrib.andrew.cmu.edu/~dyafei/NLP.html  
[Accessed 28 02 2019].

Usabilla.com, 2019. *usabillablog - How to Design for Color Blindness.* [Online]   
Available at: https://usabilla.com/blog/how-to-design-for-color-blindness/  
[Accessed 02 28 2019].