DEPARTMENT OF INFORMATION SYSTEMS

SYSTEMS DESIGN & DEVELOPMENT

SYSTEMS SPECIFICATION FOR POPPEL ORDER PROCESSING

TEAM MEMBERS

Student Number: ***DPLKYL002***	***Student Name:*** ***Kyle Du Plessis***
Student Number: ***SNGSUJ001***	***Student Name:*** ***Sujala Singh***

TEAM NUMBER: 10

PLAGIARISM DECLARATION

1. We know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. This Systems Specification is our own work.
3. We have not allowed, and will not allow, anyone to copy our work with the intention of passing it off as their own work.

Full Name: Kyle Du Plessis Signed: KDP Date: 12 / 10 /2018

Full Name: Sujala Singh Signed: SS Date: 12 / 10 /2018

Table of Contents

1. INTRODUCTION 1
1.1. OVERVIEW OF SPECIFICATION 1
1.2. CONTEXT & SCOPE OF SYSTEM SPECIFICATION 1
1.3. DESIGN ASSUMPTIONS & CONSTRAINTS 3
2. USER INTERFACE & DIALOGUE DESIGN 4
2.1. WIREFRAME DIAGRAMS 5
2.2. SCREEN STANDARDS 8
2.3. DETAILED SCREEN LAYOUT 9
3. DESIGN SEQUENCE DIAGRAMS 16
3.1. PLACING AN ORDER DESIGN SEQUENCE DIAGRAM 17
3.2. EXPIRED PRODUCT REPORT DESIGN SEQUENCE DIAGRAM 19
4. DESIGN CLASS DIAGRAM 20
5. ENTITY RELATIONSHIP DIAGRAM 21
6. REPORT DESIGN 23
6.1. EXPIRED INVENTORY REPORT 30
6.1.1. Detailed Output Requirements 30
6.1.2. Report Layout 31
6.2. ORDER CONFIRMATION REPORT 32
6.2.1. Detailed Output Requirements 32
6.2.2. Report Layout 33
7. INPUT-OUTPUT STANDARDS & CONTROLS 34
7.1. FORMALISED OUTPUTS: 34
7.2. BUILT-IN VALIDATION TO ENSURE REQUIREMENTS ARE MET 34
7.3. INPUT INTEGRITY CONTROLS 34
7.4. OUTPUT INTEGRITY CONTROLS 35
8. IMPLEMENTATION PLAN 36
9. TEST PLAN 37
9.1. TEST ENVIRONMENT 37
9.2. TEST ITEMS 37
9.3. TEST APPROACHES 38
9.4. PROBLEM TRACKING (TEST CASES) 38
9.5. TEST SCHEDULE 40

1. INTRODUCTION

1.1. OVERVIEW OF SPECIFICATION

This document provides a detailed systems specification for a single iteration of the Poppel Order Processing System project. The document structure follows a logical sequence covering various sections which include: User Interface & Dialog Design, Design Sequence Diagrams, Design Class Diagrams, Entity Relationship Diagram, Report Design, Input-Output Standards & Controls, Implementation Plan and Test Plan.

This document outlines the key activities undertaken during the design stage of the Systems Development Life Cycle (SDLC) and serves as a platform on which we will be building the proposed system during the implementation phase, followed by a series of system testing during the testing phase.

The logical ordering of sections follows the reason-centric perspective where the design process is based on a discrete sequence of stages and each stage produces deliverables required for the next stage. This project had arisen from the user requirements specification document which had been developed for the specific business problem at Poppel (i.e. Poppel Order Processing System).

The current completeness of work and documentation for this project has not been reached fully as the focus was only on implementing a segment of the Poppel Order Processing System. The design, construction, testing and implementation of the complete system would be pursued in future project(s).

1.2. CONTEXT & SCOPE OF SYSTEM SPECIFICATION

Poppel is a soft drink and confectionary manufacturer and importer located in Atlantis, Western Cape and maintains several different ordering mechanisms for its wide range of customers. The current order process is time consuming and inefficient, credit authorisation rather cumbersome and the product catalogue becomes outdated quickly - causing immense customer dissatisfaction. This increases the risk of losing customers, allowing other market competitors an opportunity. The large number of inherent problems in the company's current ordering system, as well as dealing with business expansion, has given rise to the Poppel Order Processing System computerisation project.

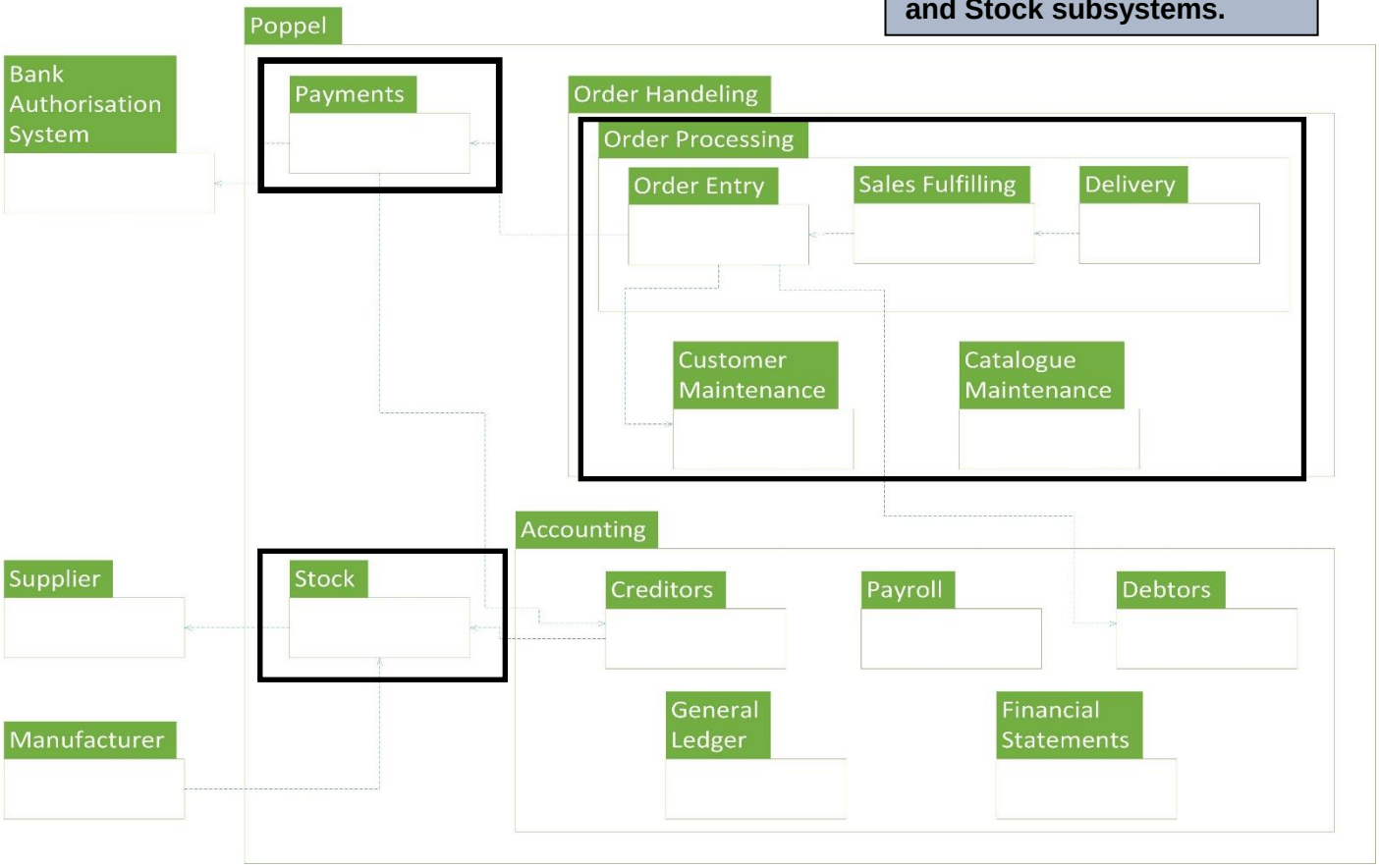
The Order Processing subsystem (package) is being developed in this project as it includes

the critical Order Entry, Customer Maintenance and Catalogue Maintenance subsystems. It is also dependent on the Payments and Stock subsystems as illustrated in the package diagram below showing the entire scope of the Poppel system.

This is essential to achieve the key functionality to be provided in the project solution which include:

1. Create a customer
2. Create a customer order for at least 3 products (including checking if customer is black listed & checking of inventory)
    a. Reserve Inventory for the items
3. Cancel an item that is not invoiced as yet
4. Generate picking list to initiate delivery
5. Print report to identify all expired products in inventory

The Order Processing subsystem is being developed in this project – dependent on the Payments and Stock subsystems.



Note: If this diagram is unclear, please see PackageDiagram.pdf

The preferred solution to this project comprises the development of a custom-built solution which supports all modes of ordering. The order processing system will include the order

entry, picking and dispatch processes together with the catalogue and customer maintenance subsystems. The objective is to improve the efficiency of the current ordering process and catalogue. The custom-built solution improves the efficiency of the current catalogue and ordering system by updating the catalogue automatically for price fluctuations, stock levels, product list and automating the generation of picking lists and inventory expiration reports. This will increase sales and ordering process efficiency by an estimated 20% and 25% respectively in 1 year, and by an estimated 10% per year thereafter.

In Scope:

- Maintain Customer Details
- Maintain Catalogue Details
- Create Customer Order (including checking stock availability, checking customer credit status and handling payments)
- Change or Cancel Existing Orders and Processing Returns
- Enquire on the status of current orders
- Generate monthly Sales Analysis Report
- Generate Order Picking Lists
- Record Order Fulfilment and Invoice Generation
- Schedule Delivery Routes
- Record Delivery Status

Out of Scope:

- Maintain Inventory (STOCKMAN package)
- Maintain Debtors Accounts (SWAT package)
- Customer discounts (to be implemented later)

1.3. Design Assumptions & Constraints

This section describes any constraints in the system design and includes any assumptions made by our project team in developing the system design listed below.

- The focus of this project was only on implementing a segment of the Poppel Order Processing System. The design, construction, testing and implementation of the complete system would be pursued in future project(s).

- The Poppel Order Processing System developed in this project is the “to-be” system (i.e. automated ordering system), not the current (“as-is”) ordering system which is being used by Poppel.

- The Poppel Order Processing System developed in this project is not a web application and does not connect to the Internet. The assumption is that it functions as an offline ordering application that is accessed by employees (i.e. marketing clerks) on behalf of customers wanting to place orders. Our development team is currently constrained by limited application development knowledge, expertise and experience.
- The system assumes all orders are now recorded on the system by the marketing clerk.
- The system processes order payments assuming valid information is entered into the relevant payment fields and assumes payment is made immediately. External bank authorization was not part of the system scope for this project
- The Poppel Order Processing System UI has been designed specifically for the desktop platform and not for mobile devices which require specific design decisions to work within the device hardware and resource constraints (i.e. limited memory, battery life, bandwidth and smaller screen size).
- There are performance constraints imposed on the system which include ideal real-time response to user inputs and fast loading of UI screens, dialogs and report generation. This is essential as it directly affects customer satisfaction with respect to response time as customers expect fast and efficient service.
- Another constraint in the system design is being restricted to develop the entire system using C# as a fixed coding environment and Visual Studio as a fixed development environment. Development team members are more proficient in other programming languages such as Java, Python and C++.
- The project does not have a financial budget; hence no funds are available to cover any development costs that may arise during the project time. There are budget constraints imposed on the system which include some project management and wireframing tools requiring payment, unless used on low features which decreases project quality.
- There is a schedule constraint in the system design which is the fixed, non-negotiable project deadline for all deliverables to be submitted by 12 October 2018 at 12:00. There is also a constraint on team composition as the development team is only restricted to 2 group members allocated to work together on this project.

2. USER INTERFACE & DIALOGUE DESIGN

This section will provide a sense of what the dialog between the user and the system will look like using wireframes. It will also set out the function of each element and the reason for the system's design. The standards by which the system will be designed in terms of appearance, such as layout and colour will also be outlined. Finally, realistic mock-ups of the final dialogue between user and system will be presented.

2.1. Wireframe Diagrams

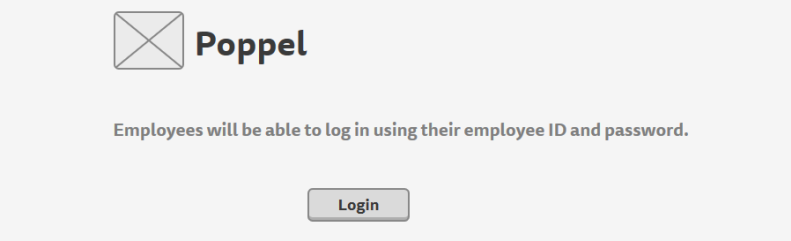
This section provides a rough idea of what the screens the user interacts with will look like.

Employee Log-In

The employee will have to enter their employee number and password into the textboxes to gain access to the site.

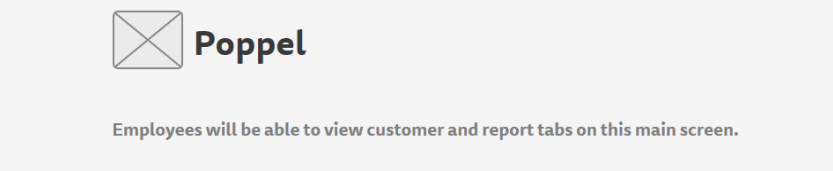
The system will verify the employee details and grant them access to certain options depending on their clearance level.

The system will check if the required fields are filled in and if they are in the correct format.



Home

After log-in, employees will be redirected to this page. This screen will allow employees the options of creating or managing a customer as well as generating a picking list or an expired inventory report.

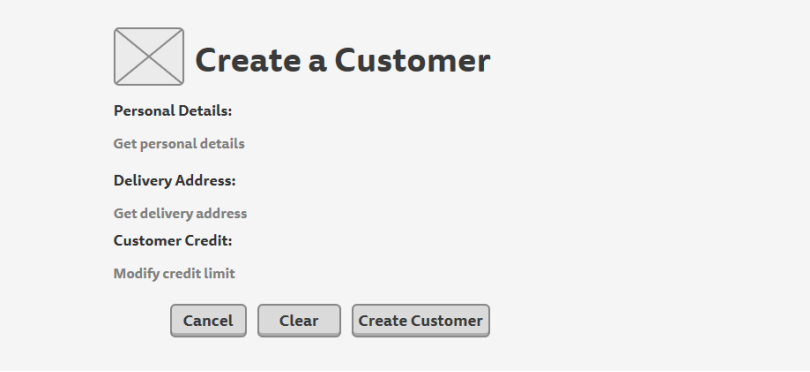


Create a Customer

There are textboxes that the employee will use to fill in with the required details. The

employee can also adjust the customer's credit limit. Once all the details are filled in, they are captured into the system. There is also a "Cancel" button if the customer wishes to quit the registration.

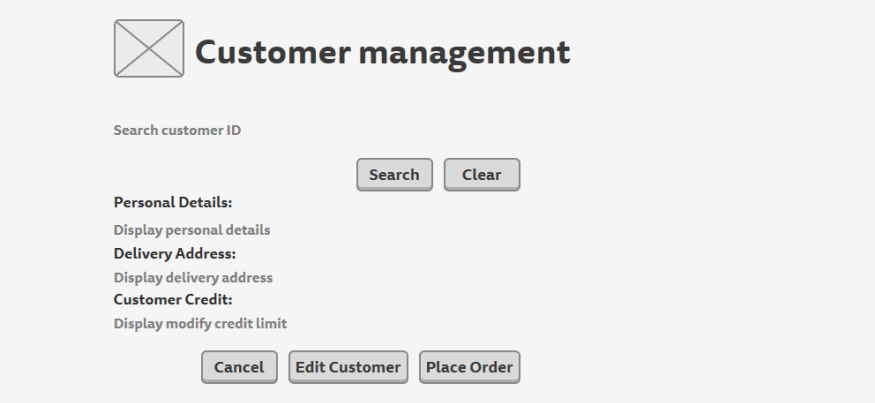
The system will check if all the required fields are filled in and if they are in the correct format. The system will also assign the user with a customer number (generated with the first three letters of their first and last names).



Customer Management

This screen will display the customer details (after the employee enters the customer number into the system).

This screen allows the employ to edit the customer details, delete the customer or place an order on behalf of the customer. This can be done by clicking the appropriate button.



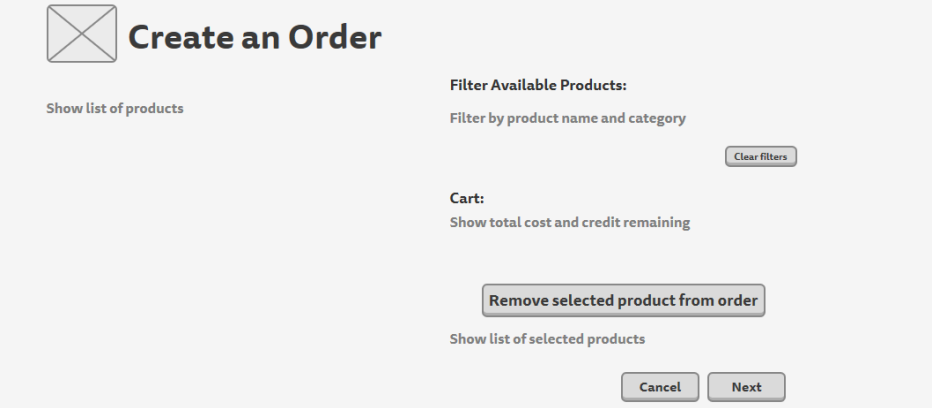
Create an Order

This screen is where the employee can view the product catalogue and select the items the customer wishes to order. This screen is launched when the employee chooses to place an order (from the previous screen).

The items are displayed using an image with its name, description, price and the quantity in stock next to the image. The user will be able to add the item to the cart and choose the quantity the customer requests to order on this page. This makes shopping more efficient and easier to navigate. If an item is no longer in stock, the employee will not be able to add it to the cart.

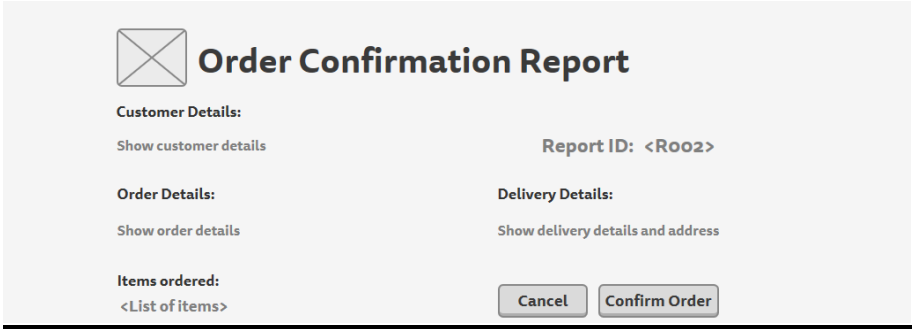
The user will be able to filter through the products to find products faster.

The user can also remove an item from the order that has not yet been invoiced from this page as well.



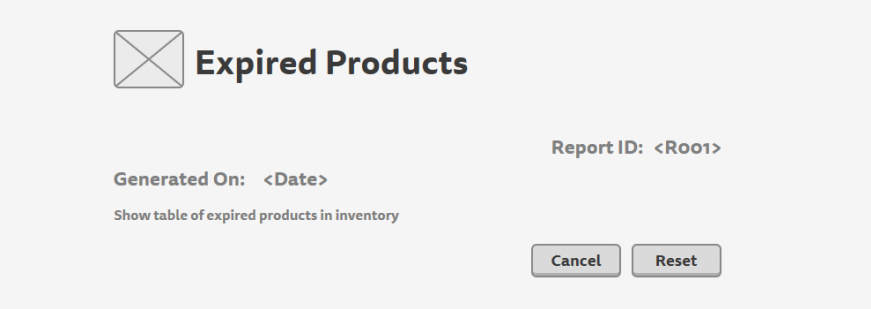
Order Confirmation Report

The confirmation screen will display the customer details, order details, delivery details and a summary of the products that were ordered.



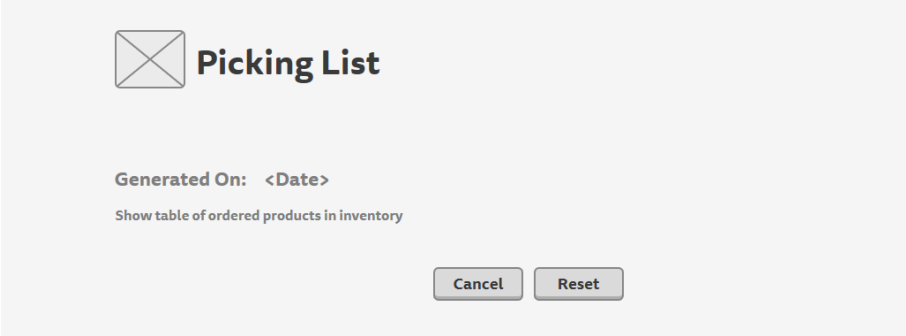
Expired Product Report

This screen will display all the expired products that Poppel has in stock. It will also display the date the report is generated on.



Picking List

This screen will display the products that have been ordered and now need to be picked. It will also display the date the report is generated on.



2.2. SCREEN STANDARDS

The following elements will be standardised across all screens:

- Font

  The font used for the headings will be Poor Richard, in the shade Orchid, sized 22pt and will be bolded and underlined.

  The font used for labels and textboxes will be Microsoft Sans Serif, in Black and sized 8,25pt.

- Logo

  The Poppel logo will appear at the top left-hand corner of each page.

- Background

  The Poppel logo will provide the background image for each screen.

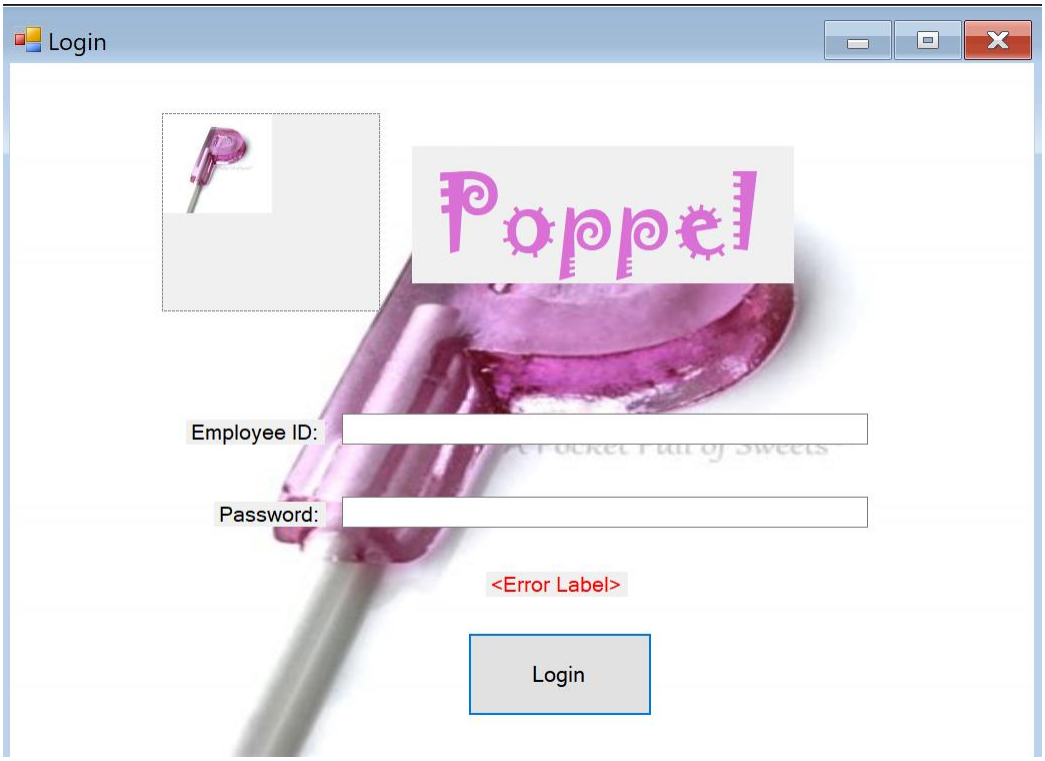
- Buttons

  Buttons will be placed in the bottom right-hand corner of each relevant screen. The sizes of all the buttons will be the same.

2.3. DETAILED SCREEN LAYOUT

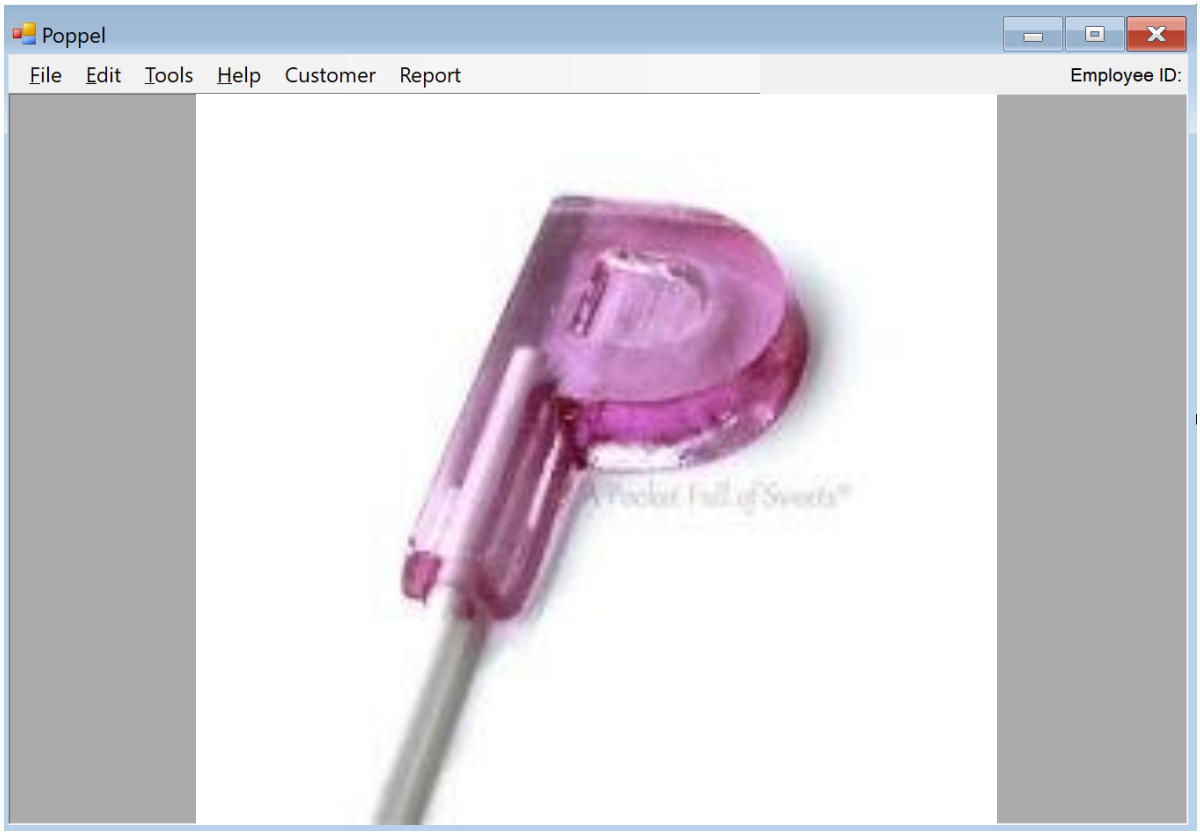
This section depicts realistic mock-ups of the final design of the dialogues between the user and the system.

Employee Log-In



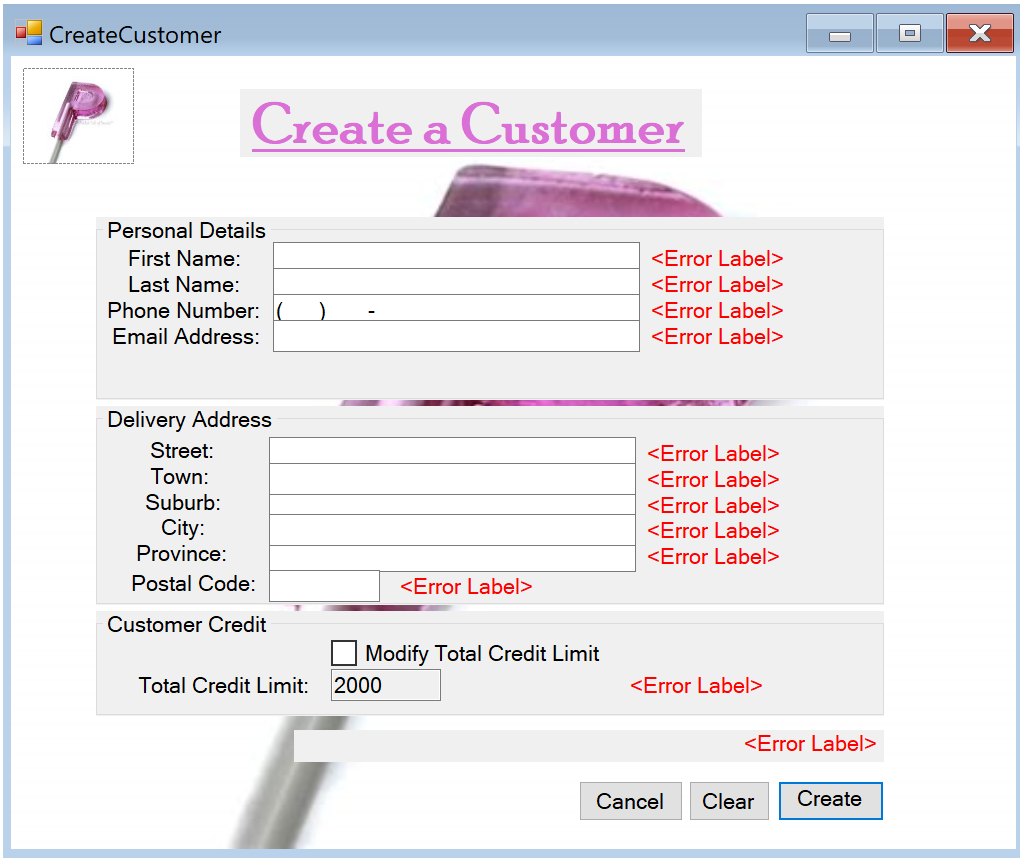
The employee will enter their username and password in the provided textboxes. If there is an invalid entry the error label will change to reflect the error, otherwise the error label is hidden.

Home



The user can use this form to navigate to the Create a Customer form, Customer Management form, Expired Product Report form and Picking List form from this screen. This is done by using the dropdown tab in the toolstrip.

Create a Customer

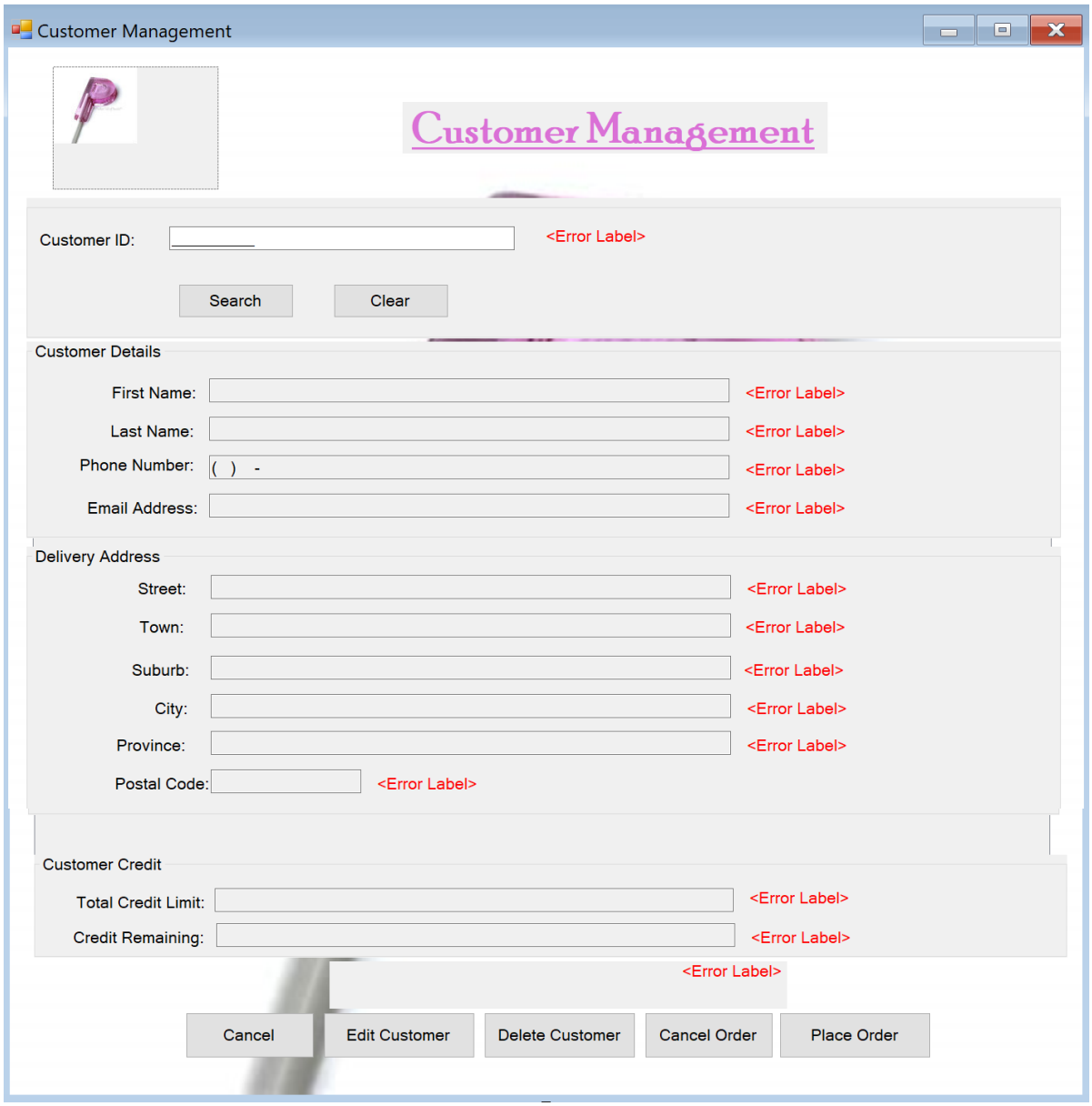


The employee will enter in the relevant details in the allocated textboxes. The credit limit can also be modified by checking the corresponding box which will enable the Total Credit Limit textbox and it can be edited.

Upon clicking “Create”, a message box will display the generated customer code.

The error labels will change to indicate if there have been any errors when entering data, otherwise the error labels are hidden.

Customer Management

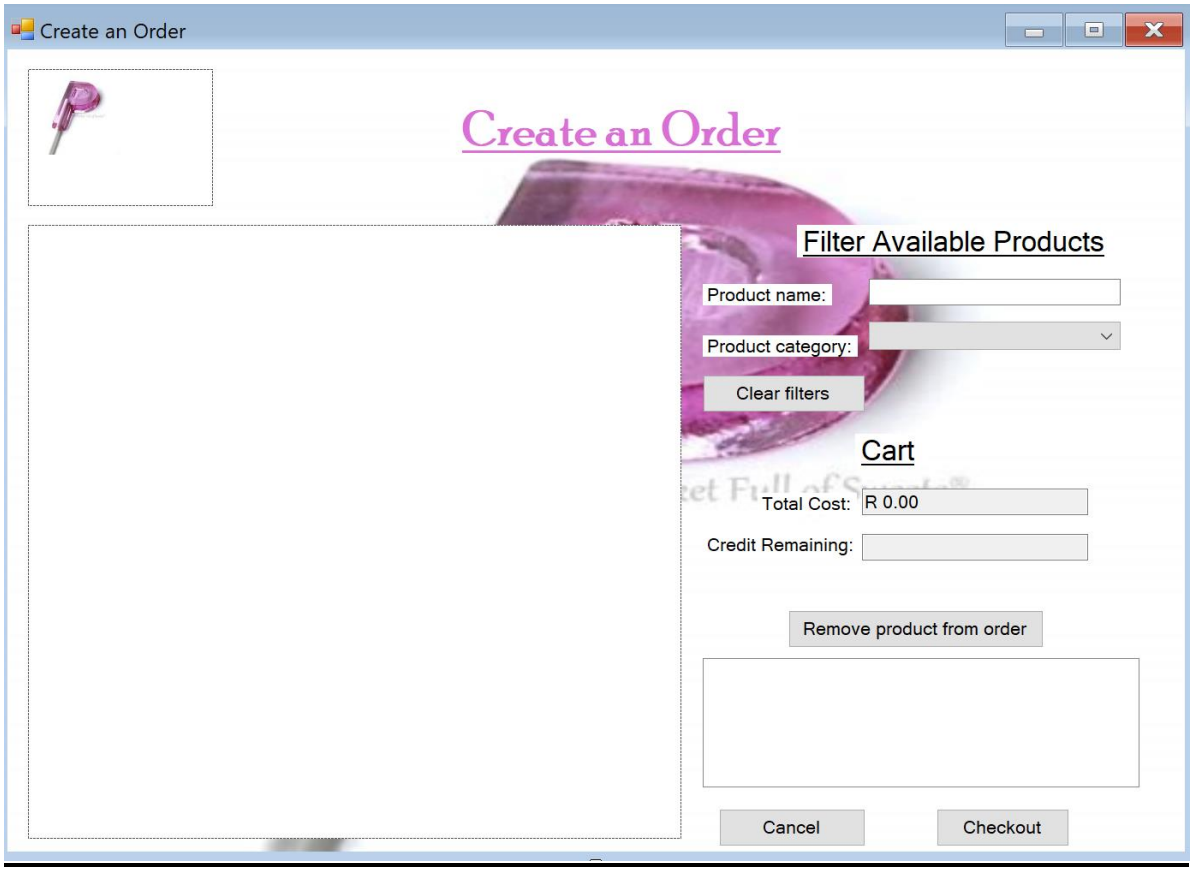


On this screen, the user enters in a customer number, clicks “Search” and the fields will be populated with the customer’s details.

From this screen, the customer’s details can be edited, the customer can be deleted, an order can be cancelled, and an order can be placed by clicking on the relevant buttons.

The error labels will change to indicate if there have been any errors when entering data, otherwise the error labels are hidden.

Create an Order

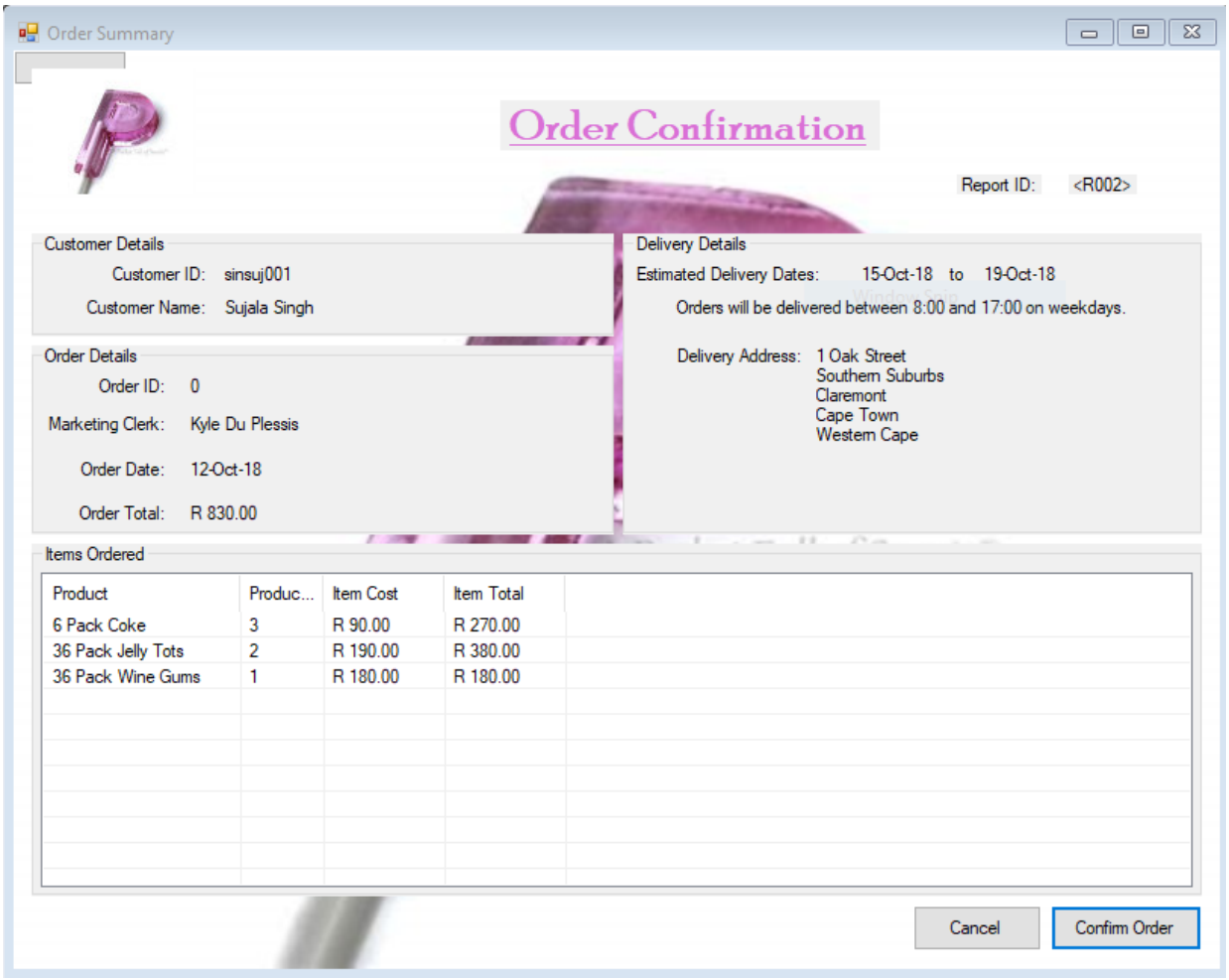


Here the user can choose an item to add to the order. The products are displayed with their description, price and number in stock. Products can also be filtered using the product name or category.

An item can be added to the order by selecting “Add” below the item image, quantities can also be changed.

Once added, the items will be displayed in the list view on the right-hand side of the page. From here items can be selected to be removed by checking the box next to the item and selecting “Remove product from order”.

Order Confirmation Report



This screen displays the completed order.

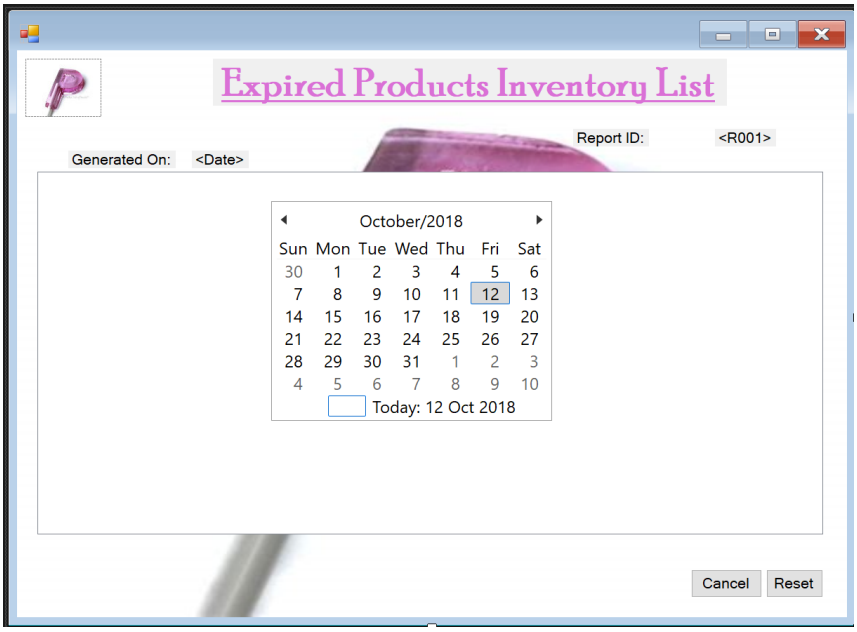
The customer's details are displayed along with the order details and delivery details.

The items ordered are displayed in a list at the bottom of the screen.

Once "Confirm Order" is clicked a message box shows up indicating that the order has been confirmed.

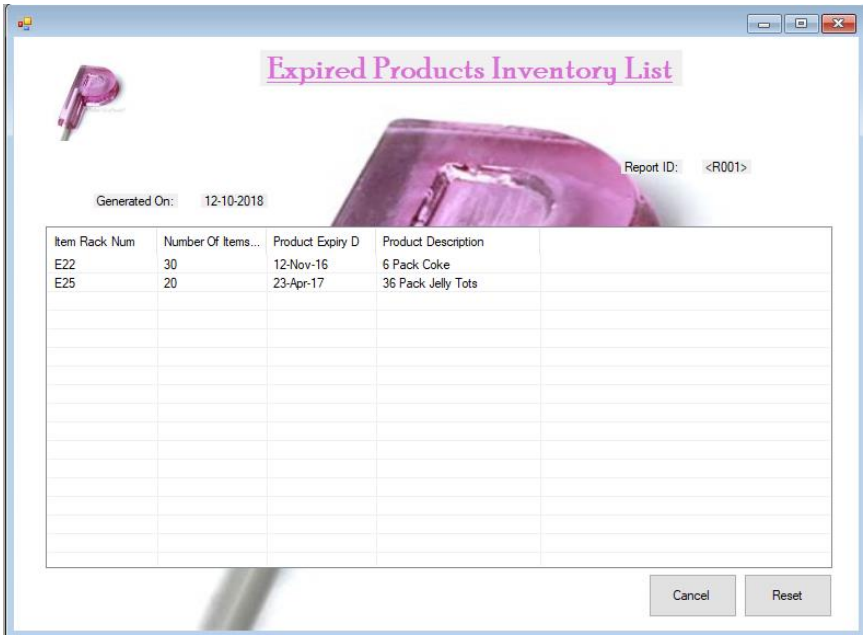
Expired Product Report

1. First Screen



On this screen the user must select a date for the system to check which products are expired.

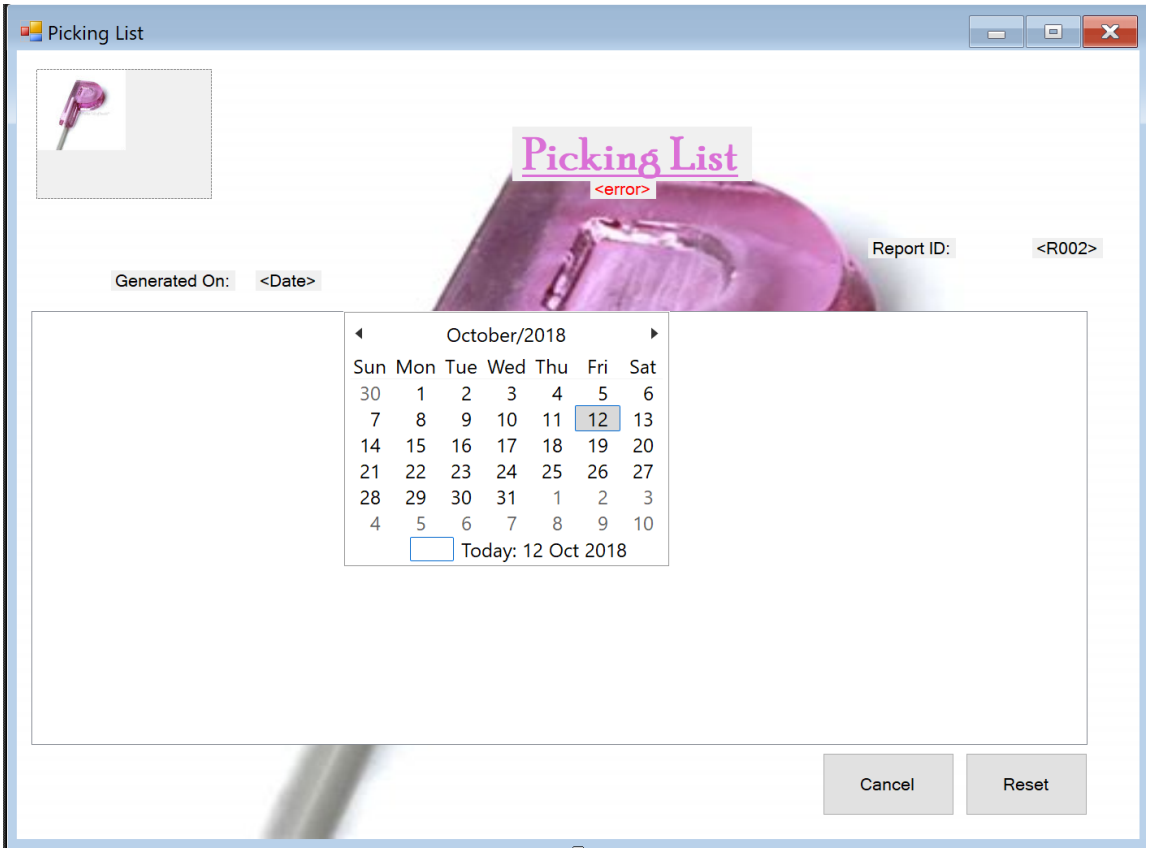
2. Second Screen



The expired products are then displayed in a list.

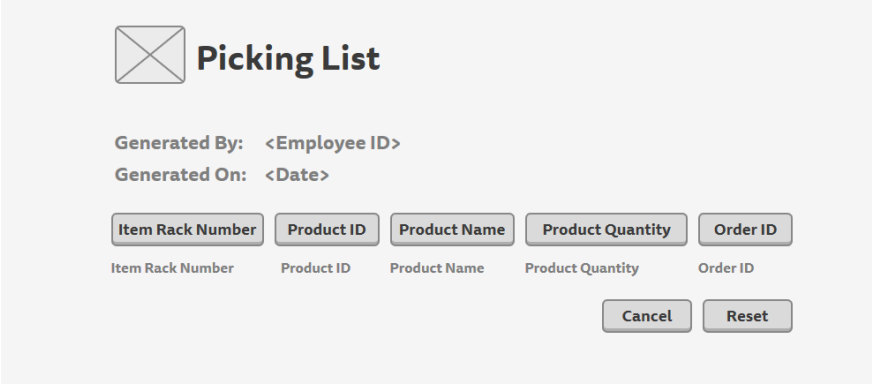
Picking List

1. First Screen



On this screen the user must select a date for the system to display the products which have been ordered so that they can be picked.

2. Second Screen



The ordered products are then displayed in a list for the pickers to follow.

3. DESIGN SEQUENCE DIAGRAMS

This section depicts two design sequence diagrams. The first being the sequence of placing an order and the second the sequence of generating an inventory report.

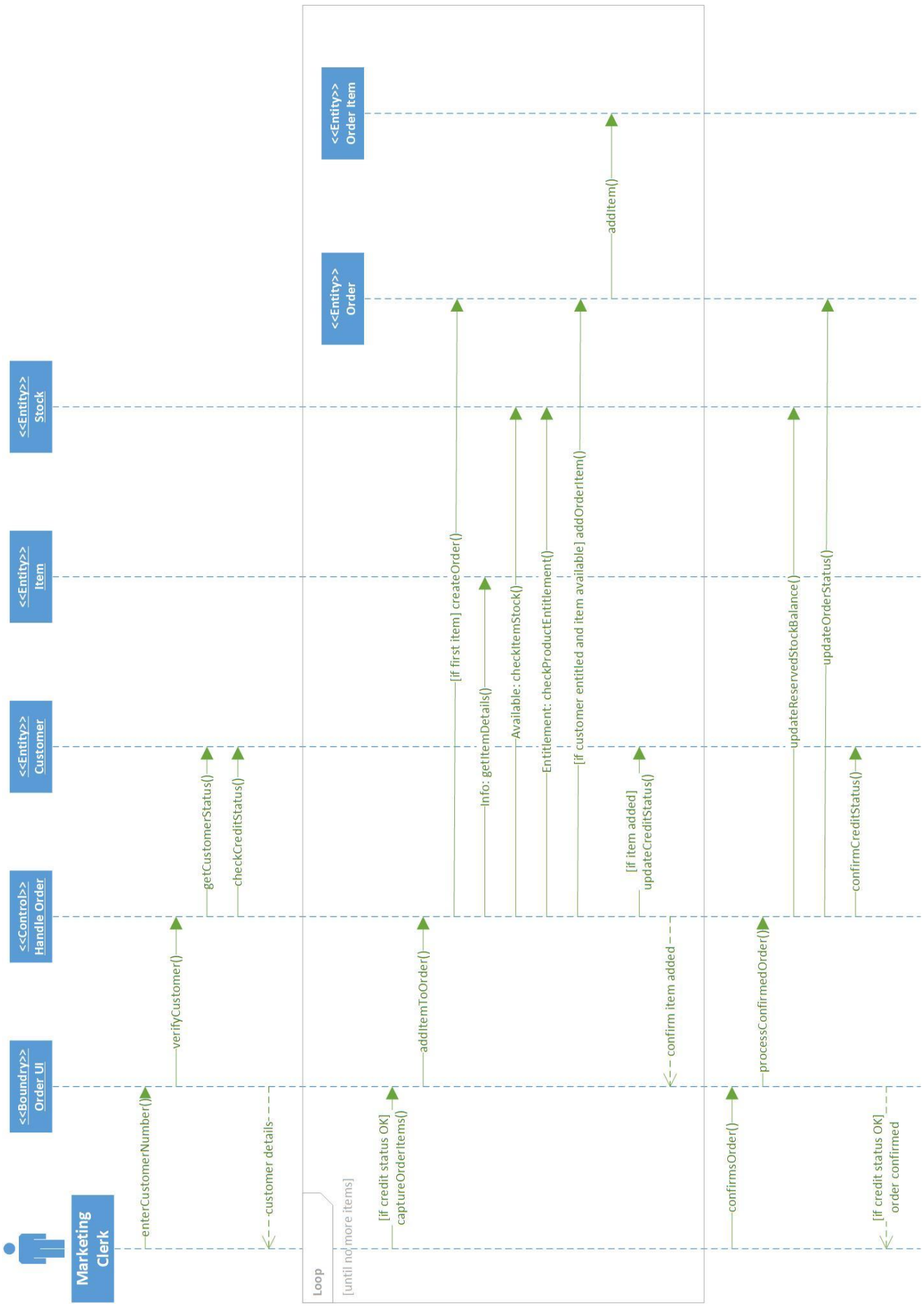
3.1. PLACING AN ORDER DESIGN SEQUENCE DIAGRAM

The process starts when a marketing clerk logs in to the system. They enter in a customer number the system verifies the customer and checks their credit status. If the credit status is okay, the clerk can proceed with placing the order.

If it is the first item selected, a command to create an order will be passed. The system will retrieve the product's information and check if there is stock available. If there is stock, the system will check the customer's entitlement status to see if they are eligible to purchase the product. If the product is available and the customer is entitled to purchase it, the item is added to the order. The system will automatically update the customer's credit status after the item is added to the order. The system will then confirm with that the order has been added. This process is repeated for other items to be ordered.

Once all items are added, the clerk confirms the order and the system updates the reserved stock balance and the order status. The system also confirms the customer's credit status, if the credit status is okay the order is confirmed and processed.

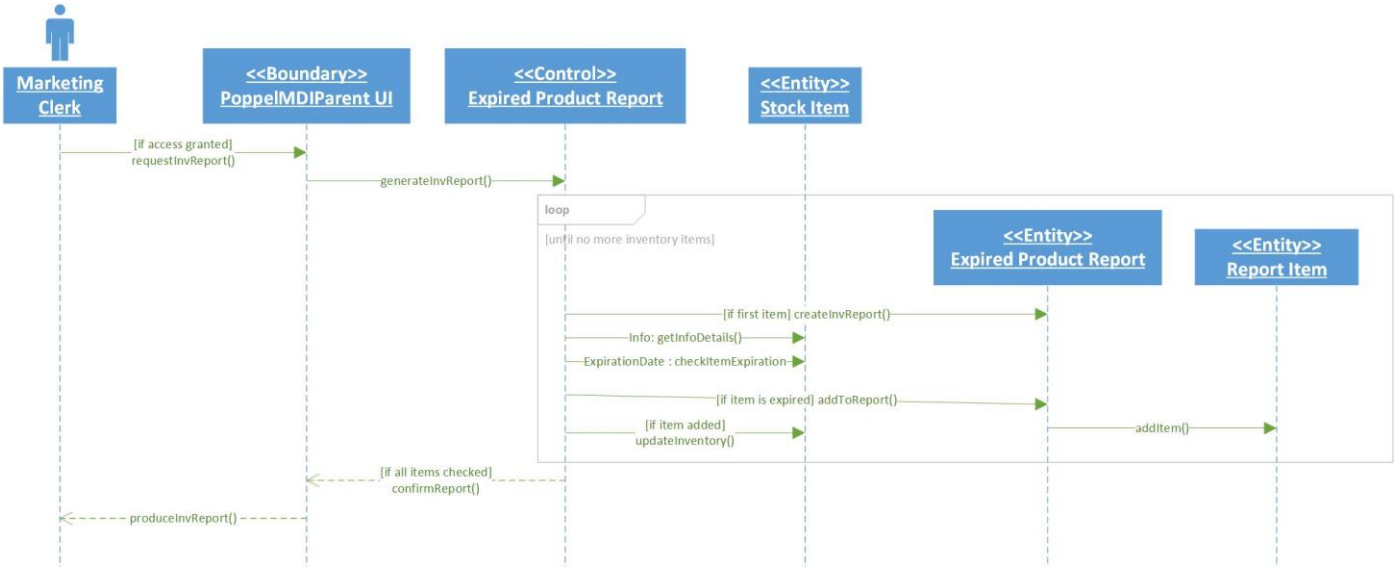
The design sequence diagram on the next page depicts the flow of events when a marketing clerk places an order.


Marketing Clerk
<<Boundry>> Order UI
<<Control>> Handle Order
<<Entity>> Customer
<<Entity>> Item
<<Entity>> Stock
<<Entity>> Order
<<Entity>> Order Item
enterCustomerNumber()
verifyCustomer()
getCustomerStatus()
checkCreditStatus()
customer details
Loop
[until no more items]
[if credit status OK] captureOrderItems()
addItemToOrder()
[if first item] createOrder()
Info: getItemDetails()
Available: checkItemStock()
Entitlement: checkProductEntitlement()
[if customer entitled and item available] addOrderItem()
addItem()
[if item added] updateCreditStatus()
confirm item added
confirmsOrder()
processConfirmedOrder()
updateReservedStockBalance()
updateOrderStatus()
confirmCreditStatus()
[if credit status OK] order confirmed

3.2. Expired Product Report Design Sequence Diagram

The process starts when an employee requests an Expired Product Report through the UI which instructs the Expired Product Report (controller) class to generate the report. The controller class instructs the Stock Item class to run through all its current inventory and check the expiration date on the products. If the product is expired, it will be added to the report. Once all the products have been checked, the system will print an Expired Product Report for the employee to see all the expired products. This will include the names of the products as well as the quantity of products expired.

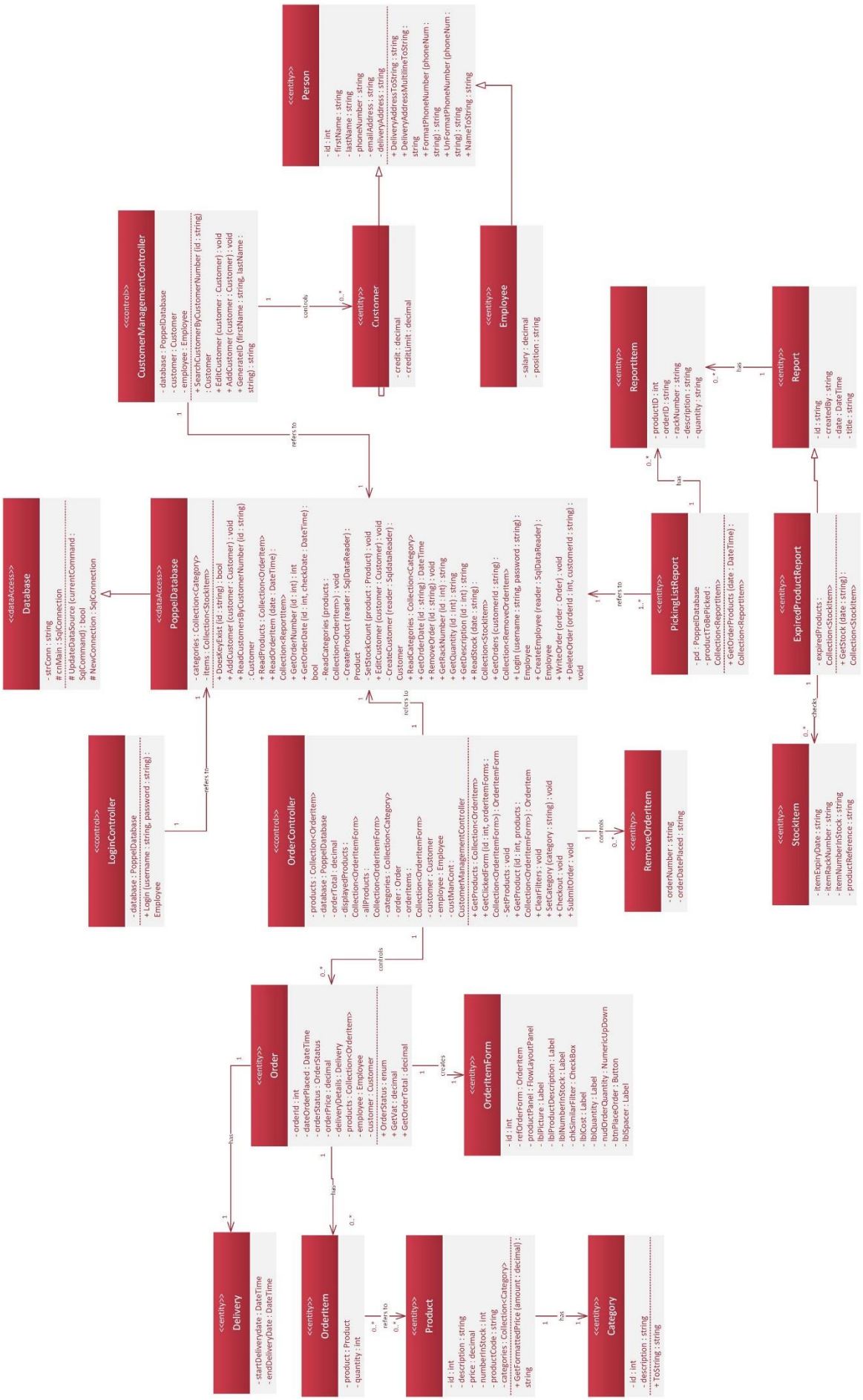
The following design sequence diagram depicts the process of generating an expired product report.



4. DESIGN CLASS DIAGRAM

It is important to have a detailed design class diagram before the programming of a system commences as it helps structure the system and shows the developers what classes to create. The diagram shows the class' attributes, properties, methods and their relationships with each other. it also depicts encapsulation as well as the visibility of the attributes and methods within the classes.

On the following page is the design class diagram for the Poppel Order Processing subsystem.
Note: if the diagram is unclear, please see ClassDiagram.pdf

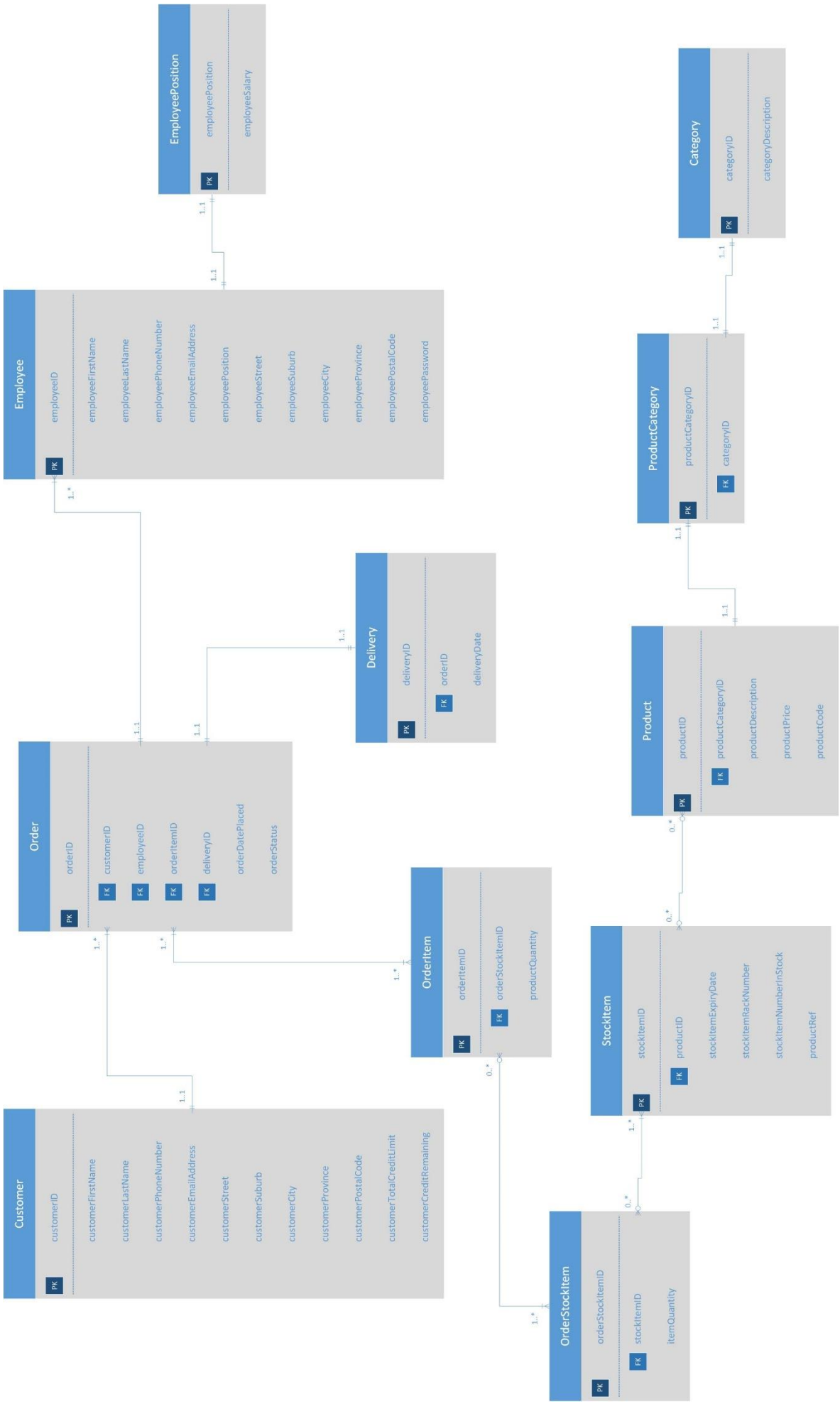

Database
- strConn : string
cnMain : SqlConnection
UpdateDataSource (currentCommand : SqlCommand) : bool
NewConnection : SqlConnection
PoppelDatabase
- categories : Collection<Category>
- items : Collection<StockItem>
+ DoesKeyExist (id : string) : bool
+ AddCustomer (customer : Customer) : void
+ ReadCustomersByCustomerNumber (id : string) : Customer
+ ReadProducts : Collection<OrderItem>
+ ReadOrderItem (date : DateTime) : Collection<ReportItem>
+ GetOrderNumber (id : int) : int
+ GetOrderDate (id : int, checkDate : DateTime) : bool
- ReadCategories (products : Collection<OrderItem>) : void
- CreateProduct (reader : SqlDataReader) : Product
- SetStockCount (product : Product) : void
+ EditCustomer (customer : Customer) : void
- CreateCustomer (reader : SqlDataReader) : Customer
+ ReadCategories : Collection<Category>
+ GetOrderDate (id : string) : DateTime
+ RemoveOrder (id : string) : void
+ GetRackNumber (id : int) : string
+ GetQuantity (id : int) : string
+ GetDescription (id : int) : string
+ ReadStock (date : string) : Collection<StockItem>
+ GetOrders (customerId : string) : Collection<RemoveOrderItem>
+ Login (username : string, password : string) : Employee
+ CreateEmployee (reader : SqlDataReader) : Employee
+ WriteOrder (order : Order) : void
+ DeleteOrder (orderId : int, customerId : string) : void
CustomerManagementController
- database : PoppelDatabase
- customer : Customer
- employee : Employee
+ SearchCustomerByCustomerNumber (id : string) : Customer
+ EditCustomer (customer : Customer) : void
+ AddCustomer (customer : Customer) : void
+ GenerateID (firstName : string, lastName : string) : string
Person
- id : int
- firstName : string
- lastName : string
- phoneNumber : string
- emailAddress : string
- deliveryAddress : string
+ DeliveryAddressToString : string
+ DeliveryAddressMultilineToString : string
+ FormatPhoneNumber (phoneNum : string) : string
+ UnFormatPhoneNumber (phoneNum : string) : string
+ NameToString : string
Customer
- credit : decimal
- creditLimit : decimal
Employee
- salary : decimal
- position : string
LoginController
- database : PoppelDatabase
+ Login (username : string, password : string) : Employee
OrderController
- products : Collection<OrderItem>
- database : PoppelDatabase
- orderTotal : decimal
- displayedProducts : Collection<OrderItemForm>
- allProducts : Collection<OrderItemForm>
- categories : Collection<Category>
- order : Order
- orderItems : Collection<OrderItemForm>
- customer : Customer
- employee : Employee
- custManCont : CustomerManagementController
+ GetProducts : Collection<OrderItem>
+ GetClickedForm (id : int, orderItemForms : Collection<OrderItemForm>) : OrderItemForm
- SetProducts : void
+ GetProduct (id : int, products : Collection<OrderItemForm>) : OrderItem
+ ClearFilters : void
+ SetCategory (category : string) : void
+ Checkout : void
+ SubmitOrder : void
RemoveOrderItem
- orderNumber : string
- orderDatePlaced : string
ReportItem
- productID : int
- orderID : string
- rackNumber : string
- description : string
- quantity : string
PickingListReport
- pd : PoppelDatabase
- productToBePicked : Collection<ReportItem>
+ GetOrderProducts (date : DateTime) : Collection<ReportItem>
Report
- id : string
- createdBy : string
- date : DateTime
- title : string
ExpiredProductReport
- expiredProducts : Collection<StockItem>
+ GetStock (date : string) : Collection<StockItem>
StockItem
- itemExpiryDate : string
- itemRackNumber : string
- itemNumberInStock : string
- productReference : string
Order
- orderId : int
- dateOrderPlaced : DateTime
- orderStatus : OrderStatus
- orderPrice : decimal
- deliveryDetails : Delivery
- products : Collection<OrderItem>
- employee : Employee
- customer : Customer
+ OrderStatus : enum
+ GetVat : decimal
+ GetOrderTotal : decimal
OrderItemForm
- id : int
- refOrderForm : OrderItem
- productPanel : FlowLayoutPanel
- lblPicture : Label
- lblProductDescription : Label
- lblNumberInStock : Label
- chkSimilarFilter : CheckBox
- lblCost : Label
- lblQuantity : Label
- nudOrderQuantity : NumericUpDown
- btnPlaceOrder : Button
- lblSpacer : Label
Delivery
- startDeliveryDate : DateTime
- endDeliveryDate : DateTime
OrderItem
- product : Product
- quantity : int
Product
- id : int
- description : string
- price : decimal
- numberInStock : int
- productCode : string
- categories : Collection<Category>
+ GetFormattedPrice (amount : decimal) : string
Category
- id : int
- description : string
+ ToString : string
refers to
controls
has
creates
checks

5. ENTITY RELATIONSHIP DIAGRAM

The entity relationship diagram (ERD) depicts the relationship between the classes by showing how they communicate with each other using primary and foreign keys. The ERD makes use of Crows-foot Notation and includes multiplicities and cardinalities to indicate the types of relationships between classes.

The ERD on the following page depicting Poppel's order processing subsystem has been normalized to the Third Normal Form.

Note: if the diagram is unclear, please see ERD.pdf


EmployeePosition
employeePosition
employeeSalary
PK
Category
categoryID
categoryDescription
Employee
employeeID
employeeFirstName
employeeLastName
employeePhoneNumber
employeeEmailAddress
employeePosition
employeeStreet
employeeSuburb
employeeCity
employeeProvince
employeePostalCode
employeePassword
ProductCategory
productCategoryID
categoryID
FK
Delivery
deliveryID
orderID
deliveryDate
Product
productID
productCategoryID
productDescription
productPrice
productCode
Order
orderID
customerID
employeeID
orderItemID
deliveryID
orderDatePlaced
orderStatus
OrderItem
orderItemID
orderStockItemID
productQuantity
Stockitem
stockItemID
productID
stockItemExpiryDate
stockItemRackNumber
stockItemNumberInStock
productRef
Customer
customerID
customerFirstName
customerLastName
customerPhoneNumber
customerEmailAddress
customerStreet
customerSuburb
customerCity
customerProvince
customerPostalCode
customerTotalCreditLimit
customerCreditRemaining
OrderStockItem
orderStockItemID
stockItemID
itemQuantity
1..1
1..*
0..*

The following are data dictionary tables that describe the attributes, primary keys, foreign keys, data types and field sizes found in the ERD.

Order:

Key	Column Name	Data Type	Null Value	Description
Primary Key	orderID	int	Not Null	A unique, automatically generated sequence of letters and numbers. Used to identify the order.
Foreign Key	customerID	nchar(10)	Not Null	A unique ID number, used to identify the customer that placed the order.
Foreign Key	employeeID	nchar(10)	Not Null	A unique ID number, used to identify the employee that processed the order.
Foreign Key	orderItemID	int	Not Null	A unique ID number, used to identify the order item added to the order.
Foreign Key	deliveryID	int	Not Null	A unique ID number, used to identify the delivery designated to the order.
	orderDatePlaced	datetime	Not Null	Stores the date the order was placed.
	orderStatus	nvarchar(50)	Not Null	Records the status of the order.

Employee:

Key	Column Name	Data Type	Null Value	Description
Primary Key	employeeID	nchar(10)	Not Null	A unique ID number, used to identify an employee.
	employeeFirstName	nvarchar(50)	Not Null	Stores the employee's first name.

	employeeLastName	nvarchar(50)	Not Null	Stores the employee's last name.
	employeePhoneNumber	nchar(10)	Not Null	Stores the employee's phone number.
	employeeEmailAddress	nvarchar(50)	Not Null	Stores the employee's email address.
	employeeStreet	nvarchar(50)	Not Null	Stores the employee's street address.
	employeeTown	nvarchar(50)	Not Null	Stores the customer's town.
	employeeSuburb	nvarchar(50)	Not Null	Stores the employee's suburb.
	employeeCity	nvarchar(50)	Not Null	Stores the employee's city.
	employeeProvince	nvarchar(50)	Not Null	Stores the employee's province.
	employeePostalCode	nvarchar(4)	Not Null	Stores the employee's postal code.
	employeePassword	varchar(50)	Not Null	Stores the employee's password so they can login to the system.
Foreign Key	employeePosition	nvarchar(30)	Not Null	Stores the employee's position.

EmployeePosition:

Key	Column Name	Data Type	Null Value	Description
Primary Key	employeePosition	nvarchar(30)	Not Null	Identifies the different positions available at Poppel.
	employeeSalary	money	Not Null	Stores the salary that corresponds with the position.

Customer:

Key	Column Name	Data Type	Null Value	Description
Primary Key	customerID	nchar(10)	Not Null	A unique ID number, used to identify the customer that placed the order.
	customerFirstName	nvarchar(50)	Not Null	Stores the customer's first name.
	customerLastName	nvarchar(50)	Not Null	Stores the customer's last name.
	customerPhoneNumber	nchar(10)	Not Null	Stores the customer's phone number.
	customerEmailAddress	nvarchar(50)	Not Null	Stores the customer's email address.
	customerStreet	nvarchar(50)	Not Null	Stores the customer's street address.
	customerTown	nvarchar(50)	Not Null	Stores the customer's town.
	customerSuburb	nvarchar(50)	Not Null	Stores the customer's suburb.
	customerCity	nvarchar(50)	Not Null	Stores the customer's city.
	customerProvince	nvarchar(50)	Not Null	Stores the customer's province.
	customerPostalCode	nvarchar(50)	Not Null	Stores the customer's postal code.
	customerTotalCreditLimit	money	Not Null	Stores the customer's allotted credit limit.
	customerCreditRemaining	money	Not Null	Stores the customer's remaining credit.

Delivery:

Key	Column Name	Data Type	Null Value	Description
Primary Key	deliveryID	int	Not Null	A unique ID number, used to identify the delivery designated to the order.
	deliveryDate	datetime	Not Null	Stores the dates the delivery will take place.

OrderItem:

Key	Column Name	Data Type	Null Value	Description
Primary Key	orderItemID	int	Not Null	A unique ID number, used to identify the order item added to the order.
Foreign Key	orderStockItemID	int	Not Null	A unique ID number, used to identify the order stock item added to the order.
	productQuantity	int	Not Null	Stores the quantity of the order item on the order.

OrderStockItem:

Key	Column Name	Data Type	Null Value	Description
Primary Key	orderStockItemID	int	Not Null	A unique ID number, used to identify the order stock item added to the order.
Foreign Key	stockItemID	int	Not Null	A unique ID number, used to identify the stock item added to the order.
	itemQuantity	int	Not Null	Stores the quantity of the order stock item.

StockItem:

Key	Column Name	Data Type	Null Value	Description
Primary Key	stockItemID	int	Not Null	A unique ID number, used to identify the stock item added to the order.
	stockItemExpiryDate	datetime	Not Null	Stores the expiry date of the item.
	stockItemRackNumber	nvarchar(20)	Not Null	Stores the rack number the item is found on.
	stockItemNumberInStock	int	Not Null	Stores the number of items in stock.
Foreign Key	productID	int	Not Null	A unique ID number, used to identify the product.

Product:

Key	Column Name	Data Type	Null Value	Description
Primary Key	productID	int	Not Null	A unique ID number, used to identify the product.
Foreign Key	productCategoryID	int	Not Null	A unique ID number, used to identify the product category.
	productDescription	nvarchar(50)	Not Null	Stores the description of the product.
	productPrice	money	Not Null	Stores the price of the product.
	productCode	nvarchar(20)	Not Null	Stores the code for the product.

ProductCategory:

Key	Column Name	Data Type	Null Value	Description
Primary Key	productCategoryID	int	Not Null	A unique ID number, used to identify the product category.
Foreign Key	categoryID	int	Not Null	A unique ID number, used to identify the category.

Category:

Key	Column Name	Data Type	Null Value	Description
Primary Key	categoryID	int	Not Null	A unique ID number, used to identify the category.
	categoryDescription	nvarchar(50)	Not Null	Stores the description of the category.

6. REPORT DESIGN

This section describes two different electronic reports the Poppel Order Processing System will generate – an expired inventory report and order confirmation report – which the user can view once logged into the system.

6.1. EXPIRED INVENTORY REPORT

The system generates this report to identify all expired products in inventory. This is valuable as it can be periodically generated by Poppel management/employees to sell off products at discounted prices (if still completely safe for human consumption) or instead dispose them as it may cause health risk(s) which Poppel will be held accountable for.

6.1.1. Detailed Output Requirements

1. **Output type & ID:**
   This is an electronic exception report (ID: R001) as it provides detailed information about transactions / operating results that fall outside a predefined normal range of values and is used when non-standard conditions experienced at Poppel (i.e. expired products in inventory).
2. **Report Objectives:**
   This report lists all expired products in inventory to allow Poppel management / employees to make strategic decisions - to sell off products at discounted prices (if still completely safe for human consumption) or instead dispose them as it may cause health risk(s) which Poppel will be held accountable for.
3. **Audience:**
   Poppel management / employees
4. **Content:**
   Generated By, Generated On, Item Rack Number, Number of Items in Stock, Product Expiry Date, Product Description
5. **Layout:**
   Columns are used for the various content fields in table.
6. **Selection:**
   The report can be generated periodically by clicking the “Inventory Report” button on the employee Home screen.
7. **Sequence:**
   The list of expired products is displayed in descending order of item rack number.

8. **Comparison:**

   Expiry date < Today's date

9. **Grouping:**

   Fields are grouped by item rack number.

10. **Media to be used:**

    Electronic / digital

11. **Frequency, Timing, Delivery:**

    The report can be generated periodically by Poppel management / employees however, best practice would be to generate the report on a daily or weekly basis.

12. **Distribution:**

    The single report is only displayed to the specific Poppel management member / employee identified by login who generates the report and is automatically displayed on screen and distributed by email to this specific user.

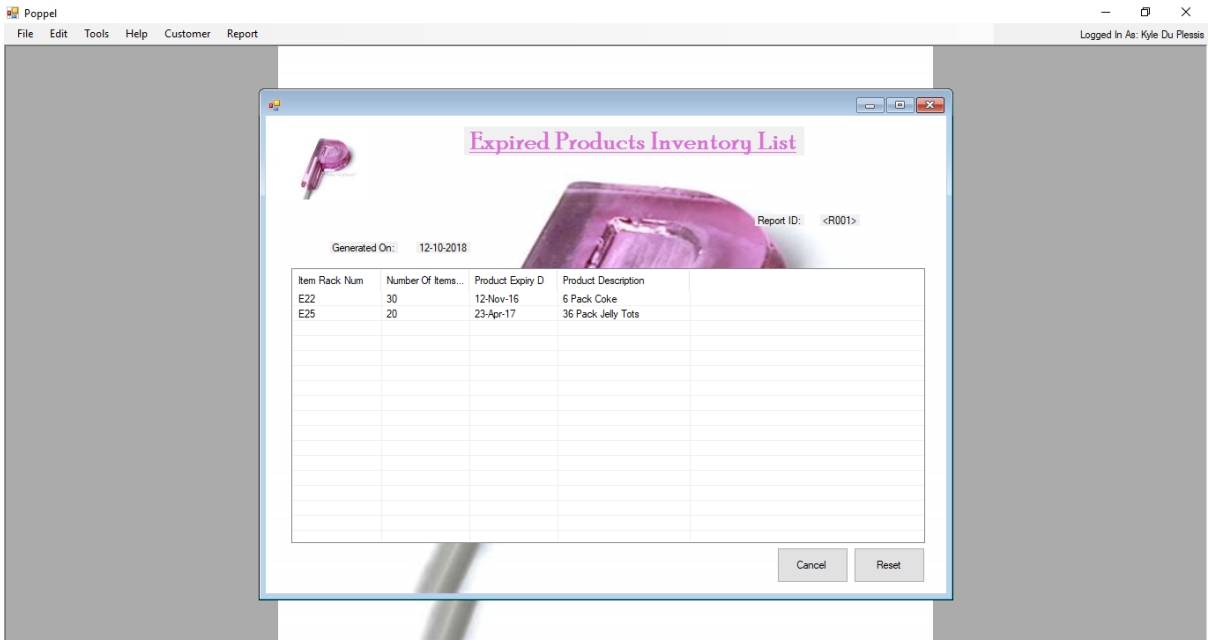
13. **Privacy, security & integrity requirements:**

    This report can only be generated by specific Poppel management / employees identified by login details (valid username and password).

    The generated report will display detailed expired product information that is accurate, complete, and current.

    The report can only be accessed by authorized Poppel employees.

6.1.2. Report Layout



6.2. Order Confirmation Report

The system generates this report to confirm an order placed for a customer. This is essential as it verifies that the order has been successfully placed and recorded, reducing misunderstandings in case the order does not match the original purchase offer.

6.2.1. Detailed Output Requirements

1. **Output type & ID:**
   This is an electronic detailed report (ID: R002) as it contains specific information on day-to-day business transactions at Poppel (i.e. order confirmations).
2. **Report Objectives:**
   This report confirms an order placed on behalf of a customer to verify that the order has been successfully placed and recorded, reducing misunderstandings in case the order does not match the original purchase offer.
3. **Audience:**
   Employees and customers.
4. **Content:**
   Customer ID, Customer Name, Order ID, Marketing Clerk, Order Date, Order Total, Estimated Delivery Dates, Delivery Time, Delivery Address, Items Ordered
5. **Layout:**
   Sections for each set of details: Customer Details, Order Details, Delivery Details. The ordered items are displayed in a list at the bottom of the page.
6. **Selection:**
   The report is generated automatically for all orders where payment has been successfully received and if the customer is not blacklisted, by clicking the “Checkout” button on the Create an Order screen.
7. **Sequence:**
   The list of ordered products is displayed in ascending order of product name.
8. **Comparison:**
   First product name < last product name
9. **Grouping:**
   Fields are grouped by product name.
10. **Media to be used:**
    Electronic / digital
11. **Frequency, Timing, Delivery:**
    The report is generated automatically for all orders where payment has been successfully received and if the customer is not blacklisted.

12. Distribution:

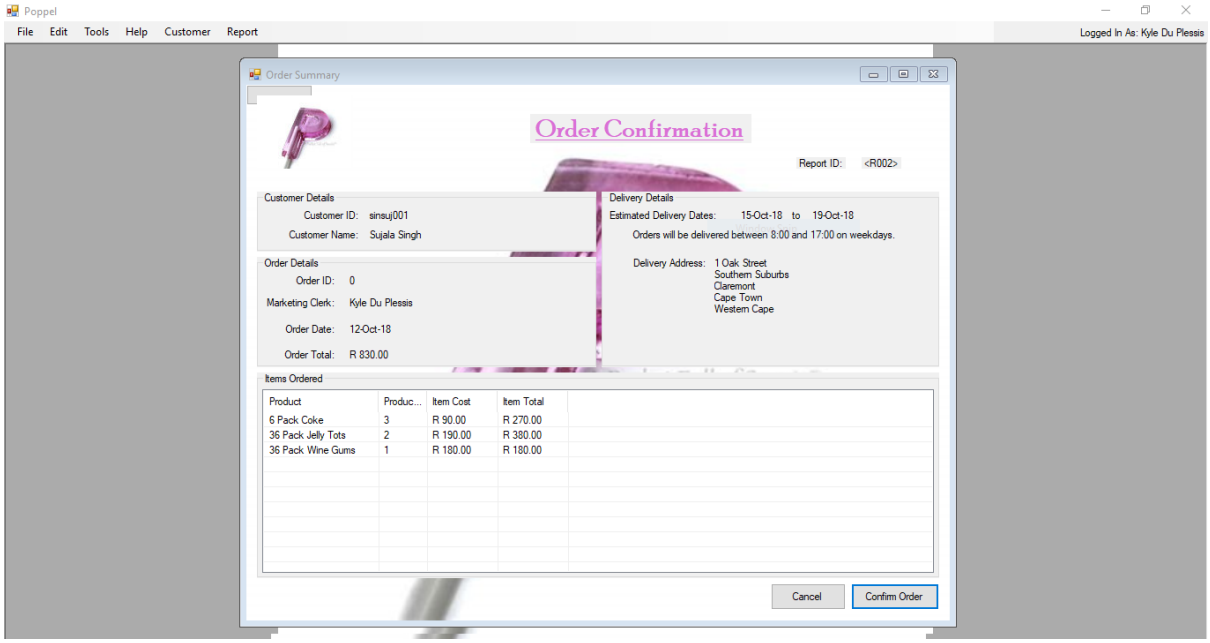
The single report is displayed to the specific customer identified by customer ID and the employee that processed the order. It is distributed by email to this specific user when the employee confirms the order by clicking the appropriate button.

13. Privacy, security & integrity requirements:

This report is only displayed for specific customers identified by login details (valid username and password) after successful payment processing and black list checking. The generated report will display detailed ordered product information that is accurate, complete, and current.

The report can only be accessed by authorised Poppel employees and the specific customer.

6.2.2. Report Layout



7. INPUT-OUTPUT STANDARDS & CONTROLS

This section provides the detailed design of the system and subsystem inputs and outputs relative to the user (i.e. customer or Poppel management member / employee).

7.1. FORMALISED OUTPUTS:

Formalized forms of output included in the Poppel Order Processing System include the order confirmation report, expiry list inventory report, product catalogue list, product checkout list, and picking list to initiate delivery. It also includes various output pop-up dialogs: unsuccessful login, successful and unsuccessful customer creation, incomplete or invalid fields, cancel confirmation, remove selected items confirmation, proceed to payment confirmation, back confirmation, invalid promotion code and payment confirmation.

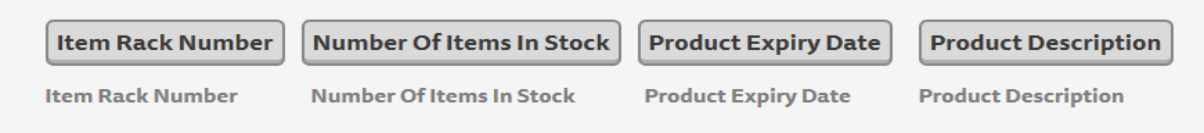
7.2. BUILT-IN VALIDATION TO ENSURE REQUIREMENTS ARE MET

User authentication and permissions had been implemented as a security measure to restrict access of critical data items and information to only those access types required by users (i.e. customer and employee login). There are verification processes for additions, deletions, or updates of critical data (e.g. customer, order and inventory information) and confirmation messages are displayed to users before destructive actions. All input fields are checked for correct, complete and valid user information. Input entry limits to fields are incorporated where appropriate and required fields are highlighted. Correct user information input in fields are saved to prevent repeat data entry. Good field defaults are used to help reduce user error and allows users to refine their choices.

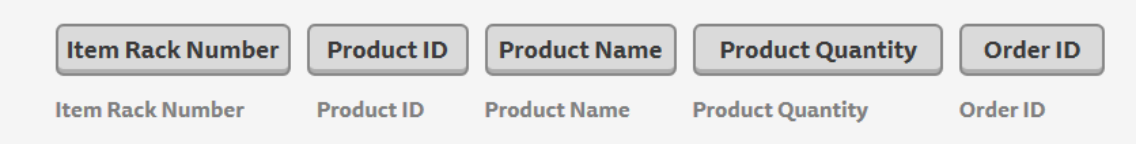
7.3. INPUT INTEGRITY CONTROLS

On the Create a Customer screen, a format is provided for the phone number to follow. The input is required to be in this specific format, otherwise an error message is generated to inform the user.

7.4. Output Integrity Controls



On the Expiry List Inventory Report screen, only expired products in inventory are displayed in the output table where the expiry date has been reached or is before today's date.



On the Picking List screen, only products that have been ordered are displayed.

8. IMPLEMENTATION PLAN

This section includes a detailed implementation plan scheduling the tasks our development team has completed over approximately 3 weeks to move from the design stage to the final delivery of this phase of the system to the users.

Implementation Stage	Task	Duration (Days)	Start Date	End Date
Class Implementation	Create and add relevant entity classes to the business layer of the application	1	15/07/2018	15/07/2018
Database Manipulation	Create a database with one table per concrete class	2	16/07/2018	17/07/2018
	Create and add relevant database classes to the database layer to allow data manipulation			
	Create and add controller classes to the business layer of the application			
User Interface Design	Create forms and add these to the presentation layer of the application	2	18/07/2018	19/07/2018
Complete Poppel Order Processing System	Create a customer	3	20/07/2018	22/07/2018
	Create a customer order for at least 3 products (including checking if customer is black listed and checking of inventory) • Reserve Inventory for the items			
	Cancel an item that is not yet invoiced Generate a picking list	3	23/07/2018	25/07/2018
	Print report to identify all expired products in inventory	3	26/07/2018	28/07/2018

9. TEST PLAN

A system is tested to determine if there are any development issues or defects that exist. This is an important stage as good testing practices have good chances of finding any undiscovered errors. Testing identifies weaknesses with the system, proves the system's quality in that it is both usable and efficient, and it also determines if the system has sufficiently met the user requirements.

9.1. TEST ENVIRONMENT

The system has the potential to expand if it is implemented successfully. Therefore, future expansion and growth must be taken into consideration.

The minimum hardware requirements are an Intel Core i5, 8GB RAM and a 500GB hard drive.

The minimum software requirement is to have programs powerful enough to simulate engagement with the system.

9.2. TEST ITEMS

The features of the system that will be tested are relate to the ordering process. The features to be tested are:

- Employee login
- Creating a customer
- Create an order
    - Check inventory
    - Check if a customer is black listed
    - Reserve inventory items
- Cancel an item that has not been invoiced
- Generate a picking list
- Generate an inventory report for expired products

9.3. TEST APPROACHES

The following tests will be performed:

- Unit testing: this test conducted to verify the implementation of the design for one software element. It tests each unit of code, methods and functions of the system to ensure that each unit is working properly. This is performed throughout the entire development process. It ensures that there is thorough testing of each unit. In pair-programming, one-member codes the function while the other member tests the completed code.
- System testing: this will test the functionality and completeness of the system as a whole as well as testing the functionality of each subsystem. It will test the integrated hardware and software system to verify that the system meets the specified requirements. This test will identify any defects that occur once the system is running as a whole. This includes testing the system's performance, security, configuration, sensitivity, start-up and recovery from failure modes.
- Integration testing: progressive testing in which software, hardware or both are combined and tested to evaluate the interactions until the entire system is integrated. It tests that the interfaces between the subsystems are working correctly. It also tests if any interactions with external interfaces work correctly.
- Performance testing: this is a Black Box test which tests whether the overall performance of the system is acceptable and if it meets the performance criteria.
- Alpha testing: an acceptance test performed by the development team in an environment controlled by the developers. It tests the entire system with the developer there to observe the test and to record any errors and usage problems that occur. It is necessary to follow Alpha testing with Beta testing.
- Beta testing: an acceptance test performed by the customer in a live application of the software in an environment not controlled by the developer. At this stage, the online ordering system will be implemented in the head office and will be tested by external testers.
- Acceptance testing: tests whether the system satisfies the acceptance criteria and will enable Poppel to decide whether to accept the system. This test is conducted using real data.

9.4. PROBLEM TRACKING (TEST CASES)

A test case is documentation that specifies inputs, predicted results and a set of execution conditions for a test item.

The test cases will pass if they meet the specified requirements. The test case fails if the specified requirements are not by the system.

Test Case Name	Step	User Action	Expected Results	Pass/Fail
Employee login	1	Try to login with an invalid username and/or password.	System informs user that there is an invalid entry	
	2	Try to login with a valid username and password.	System will direct user to the home screen	
Create a customer	1	Fields filled out in wrong format e.g. first name has numerical characters.	System should inform the user that the entry is invalid.	
	2	Fields left blank.	System should inform user that the field must be populated with valid data.	
	3	Credit limit not set.	System should inform user that a credit limit has not been set for the customer.	
	4	Credit limit set and all fields populated with valid data.	System should create a customer record in the database and inform user of customer ID.	
Create an order	1	Products not selected.	System should inform the user that no products have been selected.	
	2	Quantity of products selected exceed number in stock.	System should inform user that the quantity selected is too high.	
	3	Total exceeds credit limit.	System should inform the user that the total exceeds the customer's credit limit.	
	4	Products selected, quantity is appropriate, and total does not exceed credit limit.	System should update inventory reserve and display confirmation of order.	
Cancel an item (not yet invoiced)	1	No item is selected to be cancelled.	System should inform the user that no item has been selected to be removed from the order.	
	2	Item is selected to be cancelled.	System should confirm with user that the item is	

			going to be removed and update inventory reserve.	
Generate picking list	1	No date is selected.	System should inform user that a date must be selected before the list is generated.	
	2	Date is selected.	System should display picking list.	
Generate expired inventory report	1	No date is selected.	System should inform user that a date must be selected before report is generated.	
	2	Date is selected.	System should display expired inventory report.	
	3	No inventory is expired.	System should inform user that there is no expired inventory in stock.	

9.5. Test Schedule

The tests are scheduled to be performed after each part of the system is implemented. This will allow developers to discover errors early in the development process and address them accordingly. Each error will be addressed in the implementation phase that it is discovered in.

SA CRM	Duration (Days)	Start Date	End Date
Customer Subsystem	14	29/07/2018	11/08/2018
Unit Testing	10	29/07/2018	07/08/2018
Integration Testing	4	08/08/2018	11/08/2018
Inventory Subsystem	16	12/08/2018	27/08/2018
Unit Testing	11	12/08/2018	22/08/2018
Integration Testing	5	23/08/2018	27/08/2018
Order Subsystem	16	28/08/2018	12/09/2018
Unit Testing	10	28/08/2018	06/09/2018
Integration Testing	6	07/09/2018	12/09/2018
Reports and Queries	14	13/09/2018	26/09/2018

Unit Testing	9	13/09/2018	21/09/2018
Integration Testing	5	22/09/2018	26/09/2018
System Testing	16	27/09/2018	12/10/2018
Beta Testing	10	27/09/2018	06/10/2018
User Acceptance Testing	6	07/10/2018	12/10/2018