

Point-Of-Sale Project

Name: Theo Plank

Table of Contents

1. Introduction.....	1
2 Analysis.....	2
2.1 Problem Identification.....	2
2.1.1 Stakeholders.....	2
2.1.2 Why this problem is suited to a computational solution.....	3
2.2 Interview	5
2.2.1 Interview Questions	5
2.2.2 Interview	6
2.3 Research	8
2.3.1 Existing Similar Solutions Pros/Cons and Features	8
2.3.2 Evidence of Existing Similar Solution Features	10
2.3.3 Existing Solution Features Conclusion	12
2.3.4 Proposed Solution	14
2.3.5 Further Meetings with Stakeholders	14
2.4 Requirements	16
2.4.1 Software and Hardware Requirements	16
2.4.2 Stakeholder Requirements	17
2.5 Success Criteria	19
2.5.1 Success Criteria	19
2.5.2 Success Criteria for Universal Requirements	22
2.5.3 Success Criteria Justification	24
2.6 Pre-development Limitations.....	28
3 Design	29
3.1 User Interface Design 1.....	29
3.2 User Interface Design 2.....	30
3.2.1 Menu	30
3.2.2 POS	31
3.2.3 Products	32
3.2.4 Orders.....	33
3.2.5 Clock IN/OUT	34
3.2.6 Sales Reports/Analytics.....	35
3.2.7 Admin	36
3.3 Stakeholder input.....	37
3.4 System Design.....	38
3.4.1 Overall System Use-Case Diagram	38
3.4.2 System structure Diagram.....	39
3.5 Stages	40
3.5.1 Notes/Universal algorithms or pseudocode	40
3.5.2 Stage 1/Login	45
3.5.3 Stage 2/Products	48
3.5.4 Stage 3/POS.....	51
3.5.5 Stage 4/Orders	54
3.5.6 Stage 5/Sales	55

3.5.7	Stage 6/Clock in/out.....	57
3.5.8	Stage 7/Admin.....	60
3.6	Database.....	62
3.6.1	Table List	62
3.6.2	Entity Relationship Diagram.....	63
3.6.3	Table design	63
4	<i>Development and Testing</i>	65
4.1	Stage 0/GUI development	65
4.1.1	Login	65
4.1.2	Menu	65
4.1.3	Products	66
4.1.4	Categories.....	67
4.1.5	POS	68
4.1.6	Orders.....	68
4.1.7	Sales.....	69
4.1.8	Clock In	69
4.1.9	Admin	69
4.2	Stage 1/Database	71
4.2.1	Table SQL Create Statements.....	71
4.2.2	userssessions Table.....	71
4.2.3	products Table.....	71
4.2.4	orders Table	71
4.2.5	currentorder Table	72
4.2.6	orderproducts Table.....	72
4.2.7	currentorderproducts Table.....	72
4.2.8	dailyhours Table	72
4.2.9	salesreports Table	73
4.2.10	userhours Table	73
4.3	Stage 2/Login	74
4.3.1	Problem1.....	74
4.3.2	Problem2	76
4.3.3	Developing Session Creation Method code	77
4.3.4	Additional Notes.....	78
4.3.5	Test Plan.....	78
4.3.6	Evidence	79
4.3.7	Success Criteria	79
4.3.8	Review/Stage Evaluation.....	79
4.4	Stage 3/Menu	80
4.4.1	Developing Session Validation Method Code	80
4.4.2	Developing User Access Code	81
4.4.3	Problems	84
4.4.4	Test Plan.....	86
4.4.5	Evidence	87
4.4.6	Success Criteria	88
4.4.7	Review/Stage Evaluation.....	88
4.5	Stage 4/Categories	89
4.5.1	Design Changes	89
4.5.2	Database Design Changes	90
4.5.3	Developing Table Model Code	90

4.5.4	Developing Category Instance Code	91
4.5.5	Developing Add Category Method.....	92
4.5.6	Add Category Button Code	92
4.5.7	Developing Categories Table Data Retrieval Code	93
4.5.8	Developing Populate Table Code	94
4.5.9	Developing Update Category Code	95
4.5.10	Update Button Code	96
4.5.11	Update Category Screen Code.....	96
4.5.12	Developing Delete Category Code	97
4.5.13	Delete Button Code	98
4.5.14	Adding Search Function	102
4.5.15	Adding Categories To Combo Box	104
4.5.16	Test Plan.....	106
4.5.17	Evidence.....	106
4.5.18	Success Criteria	108
4.5.19	Review/Stage Evaluation	108
4.6	Stage 5/Products.....	110
4.6.1	Developing Product Instance Code	110
4.6.2	Developing Product Creation Code.....	111
4.6.3	Add Product Button Code	111
4.6.4	Developing Product Table Data Retrieval Code	113
4.6.5	Developing Populate Table Code	116
4.6.6	Developing Delete Product Code	117
4.6.7	Delete Product Button Code	119
4.6.8	Developing Update Product Code.....	120
4.6.9	Update Product Button Code	121
4.6.10	Test Plan.....	125
4.6.11	Evidence.....	125
4.6.12	Success Criteria	127
4.6.13	Review/Stage Evaluation	128
4.7	Stage 6/Data Validation universal checker	129
4.7.1	Problems	129
4.7.2	Developing Method To Find When User Inputs Data.....	130
4.7.3	Developing Method To Get What Field The Data Came From And Data Type.....	132
4.7.4	Developing Method To Validate Data.....	138
4.7.5	Test Plan	142
4.7.6	Evidence	142
4.7.7	Success Criteria for universal requirements like good data	146
4.7.8	Review/Stage Evaluation.....	147
4.8	Stage 7/POS	148
4.8.1	Updating The Products Section.....	148
4.8.2	Developing POS Screen Fill Product Table Code	151
4.8.3	Developing Column Name Method	152
4.8.4	Adding Category Selection Code.....	153
4.8.5	Adding Refresh Button Code.....	162
4.8.6	Updates in Database and Classes Due To Change In Plan	163
4.8.7	Developing pos Class constructor	165
4.8.8	Developing Add Product To Order Array List Method	165
4.8.9	Developing Fill Order jTable Method	166
4.8.10	Developing Add Product To Order Button Code	167
4.8.11	Developing setOrderColumnNames Method	168

4.8.12	Developing Add Order Method	170
4.8.13	Developing Get Latest Order Primary Key method	171
4.8.14	Developing Add orderProduct method.....	172
4.8.15	Updating Product In Order ArrayList Method	173
4.8.16	Updating Fill Order jTable Method.....	174
4.8.17	Developing Delete Order Product Method	175
4.8.18	Delete Product From Order Button Code	176
4.8.19	Refresh Button Code.....	177
4.8.20	Developing Calculate Total Price Method	177
4.8.21	Developing Method to calculate price after a product is deleted from the order	181
4.8.22	Developing set custom price and set discounted price methods	182
4.8.23	Developing Update Stock Method	183
4.8.24	Confirm Order Button Code.....	184
4.8.25	Test Plan.....	185
4.8.26	Evidence.....	186
4.8.27	Success Criteria	188
4.8.28	Review/Stage Evaluation	188
4.9	Stage 8/Orders	189
4.9.1	Getter and Setter methods	189
4.9.2	Developing order retrieval method	191
4.9.3	Developing Fill Order Table Method	191
4.9.4	Developing Set Column Names method	192
4.9.5	Creating Products In Order jFrame	193
4.9.6	Getter and setter methods + constructor for productsInOrder class.....	193
4.9.7	Developing Order Product Detail Retrieval method.....	194
4.9.8	Developing fill products in order table method.....	196
4.9.9	Adding code to pass the order primary key to the new screen	196
4.9.10	Developing product in order table column names method	198
4.9.11	Test Plan.....	199
4.9.12	Evidence.....	199
4.9.13	Success Criteria	199
4.9.14	Review/Stage Evaluation	200
4.10	Stage 9/Sales Reports/Analytics	200
4.10.1	Design Changes.....	200
4.10.2	Development Plan	200
4.10.3	Developing Report Views in SQL.....	201
4.10.4	Problems	209
4.10.5	Developing Report Parameter Form.....	220
4.10.6	Developing Profit By Month Report	228
4.10.7	Developing Most Sold Product Report	251
4.10.8	Developing Number Of Orders Report	256
4.10.9	Test Plan.....	258
4.10.10	Evidence.....	258
4.10.11	Success Criteria for Sales Reports.....	268
4.10.12	Review/Stage Evaluation	268
4.11	Stage 10 Clock In	269
4.11.1	Username And Password Validation + user primary key retrieval.....	269
4.11.2	Developing Clock In Method.....	270
4.11.3	Clock In Button Code	271
4.11.4	Developing Clock Out User Method	272
4.11.5	Adding Clock Out Button Code	272

4.11.6	Developing Calculate Hours Method	273
4.11.7	Test Plan.....	274
4.11.8	Evidence.....	275
4.11.9	Success Criteria	276
4.11.10	Review/Stage Evaluation	276
4.12	Stage 11/Admin	277
4.12.1	Getter/Setter methods for userHours class + global variables	277
4.12.2	userHours Constructor	278
4.12.3	Developing user hours retrieval Method	278
4.12.4	Developing Fill user hours table method.....	279
4.12.5	Developing set column names method.....	279
4.12.6	Getter/Setter Method + Global Variable Declaration + Constructor for Admin class	280
4.12.7	Developing Create User Method	280
4.12.8	Add User Button Code	281
4.12.9	Developing fill user jTable Method.....	282
4.12.10	Developing Delete User Method	282
4.12.11	Developing Update User Method.....	283
4.12.12	Update User Button Code.....	284
4.12.13	Test Plan.....	285
4.12.14	Evidence.....	285
4.12.15	Success Criteria	286
4.12.16	Review/Stage Evaluation	286
5	Evaluation	288
5.1	Testing For Evaluation	288
5.1.1	Robustness Testing	288
5.1.2	User Testing.....	292
5.2	Overall Evaluation.....	294
5.2.1	Success Criteria	294
5.2.2	Success Criteria Evidence	297
5.2.3	Usability Features.....	303
5.2.4	Limitations.....	307
5.2.5	Bits I Added That Could Have Been Done Better + How?	307
5.2.6	Bits I Could Have Added That Would Improve the Solution / Would Solve the Limitations	309
5.2.7	Maintenance	310
6	Final Code.....	310
6.1	adminScreen.java.....	310
6.2	categoryScreen.java	312
6.3	choiceClockinScreen.java.....	316
6.4	clockinScreen.java.....	317
6.5	loginScreen.java	319
6.6	menuScreen.java.....	322
6.7	newCategoryScreen.java	330
6.8	newProductScreen.java	331
6.9	newUserScreen.java	335

6.10	orderScreen.java	336
6.11	posScreen.java	338
6.12	productScreen.java	355
6.13	productsInOrderScreen.java	359
6.14	reportScreen.java.....	361
6.15	updateCategoryScreen.java	368
6.16	updateProductScreen.java.....	369
6.17	updateUserScreen.java.....	371
6.18	userHoursScreen.java.....	372
6.19	userScreen.java.....	374
6.20	admin.java	378
6.21	categories.java	387
6.22	clockin.java	395
6.23	dataValidation.java	399
6.24	dataVerifier.java	407
6.25	database.java.....	408
6.26	mostSoldByMonth_view.java	410
6.27	orderProducts.java.....	423
6.28	orders.java	430
6.29	products.java	436
6.30	productsInOrder.java	447
6.31	profitByMonth_view.java.....	452
6.32	table.java.....	464
6.33	totalOrdersByMonth_view.java.....	466
6.34	userHours.java	478
6.35	userSessions.java	482

1. Introduction

I will be creating a Swing based windows application point of sale system coded using Java. This POS system will be used by businesses such as McDonalds or other restaurants. The system will allow employees to select certain products to add them to an order, the system will automatically calculate the cost of the order and allow the user to confirm the order. Once the order is confirmed it will be added to an order table which the user can check to see the details of the order. This table will track the orders per day including order details such as what products were on the order, the price, the time of the order, and what number order of the day it was.

When a user first downloads the app there will be no products or categories added. The user can then add categories and products for their business. For example, a restaurant could add a “drinks” category which contains multiple drink products such as “Coke”, “Fanta orange” etc. The user can set the price of each product and edit this price. The user can also add items that contain multiple products for example in McDonalds the user could make an item called a “Hamburger” which contains one bun, one beef burger, one portion of pickles, one portion of onions, one portion of ketchup, and one portion of mustard. In this case when a “Hamburger” was ordered each of the ingredients stock quantity would decrease by 1.

This system will also calculate the daily revenue and profit and the user will be able to view this by clicking on the sales reports tab. This POS system will also notify the user when the stock of any of the products is low. This system will also allow the user to search for items if they cannot find them. There will also be a stock table which will be viewable by the user allowing them to check the stock of their products, this table can be edited so that when they restock certain products, they can update the stock levels.

The reason I chose to do this project is because I work at a restaurant in which we use a similar POS system to the one I am creating when taking orders. The POS system at work only makes orders, however my POS system will have a stock management system as well as an orders table. The reason I made this addition to my POS system is because after every shift we have to restock the drinks and other products which is very time consuming and could be made much easier if we didn't have to calculate how many drinks and products were bought during that shift. The reason I added the order table which stores all the orders from that day is because at the end of the day one of us has to “cash up” which requires looking through around 30 to 50 receipts and calculating how much total revenue we earned that day, whereas this POS system would keep track of all of the orders we've had that day and calculate the revenue for us saving time and money. Furthermore, this system will help remove human error as when an employee takes an order without the POS system they often make mistakes when telling the cooks what the order was for example they might forget to remove a certain item from the dish as per the customer's request or they might tell the cooks the wrong dish, this leads to unhappy customers, less profit due to wasted food, and a possible risk of health as some customers request for certain items to be removed from orders due to allergies. My POS system, in order to solve this problem, will have a notes section where employees can write any requests or changes to the order, this will ensure the customers get the dish they asked for and make for a much lower risk of causing an allergic reaction.

By using this system, the chance for human error such as forgetting dishes would be much lower as the employee can select the requested dishes much quicker than having to write it down on paper, this would also eliminate the risk of the chefs misreading the employees handwriting which would make the cooking of food much more efficient meaning the customers get their meals sooner.

2 Analysis

2.1 Problem Identification

2.1.1 Stakeholders

My POS system is aimed towards restaurant or cafe front of house employees and managers. The stakeholders can be any age and gender however it would be easier for someone who is comfortable with tech to use such as someone in the age range of 18-35 years old. The employees would use this POS system when taking orders instead of having to write down the orders on paper or remember the orders. This system would be much more efficient than writing down orders on paper as it would have categories and product so if a customer ordered a “cheeseburger” the employee could select the “burger” category and then select the “cheeseburger” and add it to the order. This system will also automatically calculate the cost for the order, so the employee doesn’t have to calculate it themselves.

Restaurant managers would also use this system when evaluating the days earnings by looking at revenue and profit as well as viewing orders and looking at what dishes are most popular, this knowledge can be used to improve the restaurant. This POS system will have the orders clearly represented in a table which will calculate the revenue and profit and show this clearly, this table will also store the time of the order. Warehouse managers could also make use of this system as they could view the database which will have a table that tracks stock levels for the products, they could use this table to predict and prepare for when the restaurant re-orders their products from the warehouse. The stock table would be clear and have many fields such as “product name”, “price per unit”, “re-order link”, “product warehouse code” etc, this will allow the manager to check the stock levels and have the products that will be re-ordered ready to be shipped, this will also help them calculate the order price. Lastly, the company or restaurant accountant(s) could use this system to analyse the restaurants profits and revenue by viewing the orders table and using this information to provide financial advice.

2.1.2 Why this problem is suited to a computational solution

Method	Justification	How I will be using it
Concurrency	<p>Concurrency must be used to ensure that a system runs smoothly while also performing other procedures and algorithms.</p> <p>It is not feasible to create a system where the whole system stops working and pauses while a procedure is run.</p>	<p>As my POS system is used it cannot stop and pause while a procedure is completed it has to keep running so the user can use the system while procedures are being completed.</p> <p>The user must be able to navigate the system which means the system will have to be checking for user input constantly; while also allowing the user to create products and take orders while the system is performing calculations on stock, sales, shift hours etc.</p>
Decomposition/Divide and Conquer	<p>Decomposition is where you break a problem down into multiple smaller problems which are simpler and easier to solve.</p> <p>This is very effective when working with complex and large problems and makes it much easier for a developer to solve the problem.</p>	<p>I can split my system up into multiple smaller functions. The system as a whole will have a login, menu and various different main menu buttons. Therefore, I can split the system up into various different functions that I can develop and code separately such as login, menu, POS, products, orders, sales, clock in/out, admin.</p> <p>By splitting the system up into different functions I can look at coding each one separately and then integrate them together once they are all coded.</p> <p>For example, I can code the login screen on its own and treat it as its own problem then code the menu screen then integrate the two by having a successful login open the menu screen. This makes the system as a whole much easier to develop and code.</p>

Analysis

Planning in advance	<p>This refers to planning the project in detail and evaluating which parts of the project will take longer.</p> <p>This also refers to planning the order in which it would be most efficient to code the project as well as predicting what code can be re-used and how/where it can be re-used.</p>	<p>I believe that the most time and effort efficient way to code my project would be to code the project in a somewhat linear fashion starting with the login, then the menu, then the products section and so on.</p> <p>By doing this I can build a strong foundation of working code in the login and menu sections before moving on to the next sections.</p> <p>In addition, the products section includes features such as adding a product, deleting a product, editing a product and displaying the products in a table. Once I have written the code for these features it can be re-used in many other sections such as the orders section, or when altering the users.</p> <p>By coding the project in this order and manner I can save time as I will be able to re-use lots of the code. This will avoid lots of redundant code.</p>
Data mining	<p>Data mining is the process of searching through very large amounts of data to find patterns or relationships between data to help solve a problem or improve a business.</p>	<p>Data mining can be used in the sales section of my project however on a much smaller scale.</p> <p>For example, I could use data mining to find out the most sold product on a specific day and relay this information to the user. The user can prepare for that day by stocking large amounts of that product.</p> <p>This would help increase profits for the user as they will have more of that product in stock to sell.</p>

2.2 Interview

2.2.1 Interview Questions

I will be interviewing my boss Mark who is the manager of a restaurant and one of the waiters Luke. Mark uses a POS system daily, manages stock levels and manages pay for employees. The questions I ask will aim to find an experienced opinion on how my stakeholders would use the proposed solution and what features and functions they would find useful.

2.2.1.1 *Restaurant Managing*

Questions for Mark:

1. Are there any problems with your current POS system?
2. Are you dissatisfied with any of the features in your current POS system?
3. As a manager are there any features you wish your current POS system had?
4. How long have you been managing restaurants?

Questions 1 and 2 help identify any problems with the POS system. This is important as it will help direct me when designing the proposed solution as I will know what not to do and I will also have a better idea of how to design certain features in a way that ensures the stakeholders will be satisfied.

Question 3 helps identify what additional features I should add to the proposed solution in order to make it more useful and a better developed solution for both the front of house employees taking orders, and the managers.

Question 4 establishes the stakeholder's history and experience with POS systems as a manager, this is useful as it establishes whether their opinion is based on past experience with POS systems.

2.2.1.2 *Waitering*

Questions for Luke:

1. Are there any problems with your current POS system?
2. What additional features would you find useful in a POS system?
3. How long have you been waiting?

Question 1 helps identify what problems with the POS system affect Luke as a waiter, this is important as it outlines what to avoid doing as well as what features are more problematic than useful.

Question 2 helps identify what additional features I should add to the proposed solution in order to make it more useful for waiters and better at aiding waiters at their job.

Question 3 establishes the stakeholder's history and experience with POS systems as a waiter, this is useful as it establishes whether their opinion is based on past experience with POS systems.

2.2.2 Interview

2.2.2.1 *Restaurant managing – Mark*

1. Are there any problems with your current POS system?

“It is difficult to navigate which causes delays when taking orders or making general use of the app. This is annoying and can lead to customer complaints.”

2. Are you dissatisfied with any of the features in your current POS system?

“No all the features work well.”

3. As a manager are there any features you wish your current POS system had?

“There are various managerial features I wish the app had for example, tracking stock, allowing employees to clock in and out, and possibly producing sales reports so I can identify our most sold products as well as other analytics.”

4. How long have you been managing restaurants?

“I have been managing this restaurant for 6 years.”

2.2.2.2 *Analysis – Mark*

Mark stated that there is an issue with the navigation of the system, this will be something I will focus on when designing the proposed solution in the hopes that it will be easy to navigate and use. Mark also proposed some features that would be useful in the proposed solution such as stock tracking, clock in/out function, and producing sales reports; these are features that I will aim to include as Mark is very experienced with restaurant managing.

2.2.2.3 *Waiting – Luke*

1. Are there any problems with your current POS system?

“It can be difficult to find certain products as I have to scroll through a long list which takes time and delays the customer’s order.”

2. What additional features would you find useful in a POS system?

“It would be useful if it tracked stock as it takes ages for us to count up the stock of all the drinks at the end of the shift for restocking.”

Analysis

3. How long have you been waiting?

"About a year and a half."

2.2.2.4 Analysis – Luke

Similar to Marks answer Luke also stated that there is a problem with navigation as it takes a lot of time and is difficult to find products, I plan to solve this problem by adding categories so the user can add products to categories so during use they would be able to select a category and search a much smaller list which should reduce the amount of time it takes and make it much easier to find the product. I might also add a sorting function for the list so the user can choose for the list to be alphabetically sorted in order to find the product faster. Luke also stated that a stock tracking feature would be useful as it would save time therefore this is a feature I will definitely aim to add as both stakeholders suggested it.

2.3 Research

2.3.1 Existing Similar Solutions Pros/Cons and Features

Similar POS systems -

- Square
- Zettle
- Loyverse POS

Square – Pros

- User friendly UI.
- Simple payment and order tracking system.
- Keypad.
- Understandable and detailed sales analytics.
- Easy to add products and categories.
- Customizable UI.

Square – Cons

- Complicated discount creation system.
- No notifications for low stock.
- Bad presentation of stock levels (hard to find out the stock levels of items).
- Complicated inventory system.

Zettle – Pros

- Good presentation of stock levels (very easy to scroll through a list of items and view stock levels).
- Notifies user of low stock levels for items in the basket.
- The user can set a custom low stock level for each item.
- Calculates change owed if the customer overpays.
- Efficient and well-designed receipt system.
- Simple discount creation system.
- Simple inventory system.
- Product/item menu structure is very good.
- Good presentation of item list including filters, and sort selections.
- Very efficient use of barcodes when searching for products, updating product stock levels, editing products etc.

Zettle – Cons

- Less detailed sales analytics.
- Based off reviews, this app frequently has card reader connection problems.

Analysis

Loyverse POS – Pros

- There is a split option so people can pay separately.
- There is a clock in function.
- Tracks starting cash when the shift started, cash payments, paid in, paid out, gross sales, refunds, discounts, and net sales.
- Timesheet tracks how long the shift was for each employee.
- Negative stock alerts stop employees from selling more of a product than is in stock.
- Displays stock clearly.

Loyverse POS – Cons

- Only offers one option (email) for sending a receipt.
- Less user-friendly UI.
- Non customizable UI.
- No low stock level notifications.

Analysis

2.3.2 Evidence of Existing Similar Solution Features

Loyverse Shift Earnings System

Zettle Receipt System

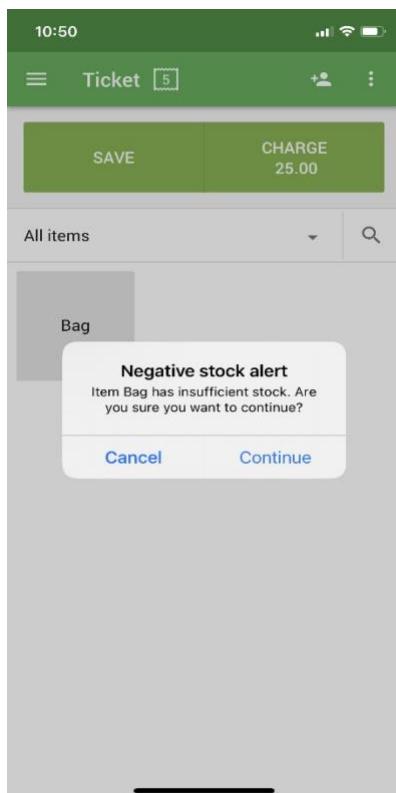
Zettle Item Menu

The image displays three side-by-side screenshots of mobile point-of-sale (POS) applications:

- Loyverse Shift Earnings System (Left):** Shows a shift summary for shift number 4, opened by 'Owner' at 22/06/2022, 10:50. It details cash drawer transactions: Starting cash £0.00, Cash payments £5.00, Cash refunds £0.00, Paid in £0.00, and Paid out £0.00. The expected cash amount is £5.00.
- Zettle Receipt System (Middle):** Shows a receipt for a transaction dated 20 June 2022. It includes a search bar for 'Find transaction' and a section for 'Amount, receipt no., card last digits'. Transaction details: £30.00 at 19:39, £70.00 at 19:21, and £40.00 at 18:31.
- Zettle Item Menu (Right):** Shows a menu for 'Sell' with a 10% discount applied. It lists items: 'Bl' (blue), 'Pr' (purple), 'Ln' (light blue), 'Ts' (grey), 'Colours' (grey), and '5 of each colour bought' (grey). A purple button at the bottom right says 'Charge £15.00'.

Analysis

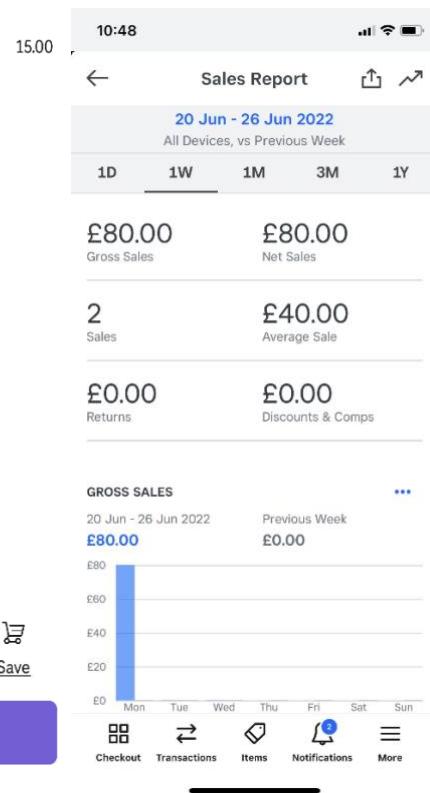
Loyverse Negative Stock Alert



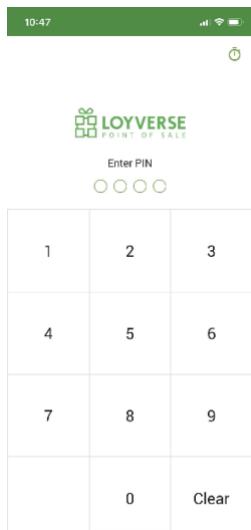
Zettle Low Stock Alerts

3 Lynx
2 left in stock

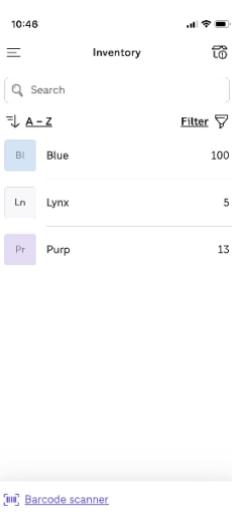
Square Sales Analytics



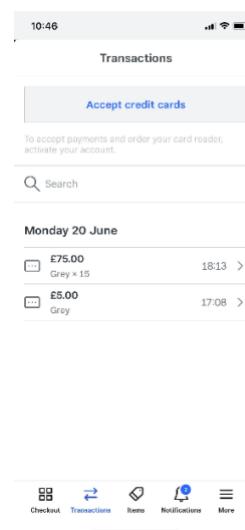
Loyverse Clock in Function System



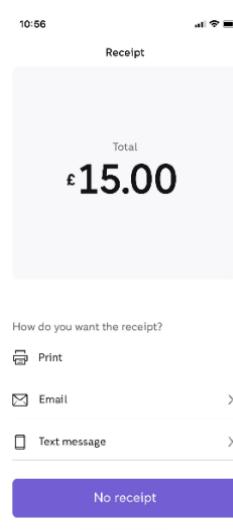
Zettle Inventory System



Square Order Tracking System



Zettle Receipt System



2.3.3 Existing Solution Features Conclusion

I have researched several different POS systems that have a similar function to mine, I have outlined the key features that these POS systems have done well and key features that they haven't done as well. I will use my research to plan the features that I intend to add to my POS systems as well as the ones that I intend not to add.

Firstly, I intend to add a payment and order tracking system and an inventory system similar to that of "Square". The payment and orders were both presented in a very clear and streamlined manner, although I didn't like how it was difficult to search for an order. In my POS system I plan to add a search feature that allows the user to enter the cost of the order, name of the order etc to find a specific order. On "square" it was hard to find out the stock level of the items as you had to click on each individual item which is a very inefficient way of checking stock levels and would waste lots of time when a restaurant has lots of items. It was also unnecessarily complicated when altering stock levels; because of this I plan to add a stock management system similar to that of "Zettle" which had a very simple stock management system where there was a list of products with stock levels displayed next to each product. This makes it very easy to check stock levels of all items quickly as well as alter the stock levels easily. Lastly, I like how "Zettle" has a list of items that has filters and a sort selection so that the user can easily look for products that for example have low stock or are from a certain category. This is a feature I plan to add to my POS system.

Secondly, although I like the UI of the "Square" app I plan to create my own UI however: I like how on the "Square" app the UI is customizable. This is a feature I would like to add however it is of minimal importance and won't affect the functionality of the app, so it won't be a priority when designing and developing my POS app. One feature of the "Square" app that I do intend to add to is the detailed sales analytics. In the "Square" app it calculates average sale cost, daily revenue, and has a bar graph that represents clearly how the revenue varies from one day/week/month to another. If I have time and if possible, I would like to add even further detailed sales analytics which would include the features I have outlined above and more features such as reports on what the most popular item was, and what dishes were sold most at specific times. This would allow the user to maximise the possible profits for the restaurant. I would also like to explore the idea of adding an AI that predicts the most-sold-dish/profits/revenue/amount-of-tables etc based off factors such as weather, day of the week, whether or not it's a national holiday or bank holiday and past experience. I think this would put my POS system above many others however it is not necessary for a good POS system to have this feature, and it would be extremely complicated to develop and therefore would only be added if I have the time.

Thirdly, I like the fact that "Zettle" notifies the user of low stock in the basket, and I like how "Loyverse" has negative stock alerts so that the user won't issue an order for more quantity of a product than they have in stock. I plan to add both of these features however I will alter them both so that the user gets a pop-up notification when they try and oversell stock and I plan to make it clearer to see when the user has low stock. In addition, I plan to add notifications so that when an item has reached a certain stock level, a level which will be determined by the user, there will be a pop up in order to notify the user of the low stock levels for that item.

Analysis

In addition, I like “Zettle’s” product menu structure, although I will design my own. I also like how easy it is to add products and categories on all three of the apps. This is a feature that I fully intend to add to my own POS system. This will make ordering much more efficient and allow the POS system to be much more compatible and be suited to any restaurant, café, or fast-food shop’s needs. One feature of the “Zettle” product-menu system that I like is the use of barcodes, this feature allows you to enter a barcode to search for a product, change stock levels etc. However, I feel that this feature could be altered to use an SKU number instead of barcodes or any other unique identifier as barcodes would be very difficult to do. Because of this I intend to add an option so that the user can add a unique identifier number to each product that they can search for to find a specific product.

The “Zettle” receipt system is very efficient, and I intend to add a system similar to this to my POS system so that when an order is confirmed it gives the user the option to send a receipt via email or message. The receipt will outline what they ordered, the time and date of the order, and the cost of the order as well as a few other details. Also, I like how easy it is to create a discount on the “Zettle” POS app, I intend to add a similar feature to my app so that the user can create a discount that will either be based off of percentage or a set amount of money discounted. The discount could have a custom name and can be added to an order and to automatically deduct the correct amount of money from the order. The discount would also show up on the receipts. If I have the time, I might add an automatic discount system so that if a specific number or choice of products is ordered then a discount will be automatically applied to the order. The discounts will also be customizable by the user so that it will only be applied to an order under specific conditions, and they can customize the discount percentage or amount and name. Lastly, I plan to add a feature that calculates how much change the user owes the customer if they overpay in cash and a feature that allows user to split the bill if the customer requests it to be split.

Lastly, I fully intend to add a clock-in/clock-out system so that the main user can create new employees that have a code which they can enter to clock in and out. This system would also track the time of their shift and how much money was made during their shift and the tips were received during their shift. This would help the manager calculate how much they owe each employee based off the tips they earned throughout the shift assuming the tips are split between everyone in the restaurant or café. This system would be similar to the “Loyverse” system and would track the employee’s hours and calculate a running total that would reset at the end of each month. The manager can multiply that number of hours worked by an employee’s hourly wage in order to calculate their pay. I’m also thinking of adding a feature so that the manager could enter each employee’s hourly wage and have the system calculate an employee’s pay automatically. In addition, my clock in/out system would expand upon the “Loyverse” system as I intend to add a function so that employees can start lunch breaks as well as clock in and out.

2.3.4 Proposed Solution

2.3.4.1 Initial Concept Considering this Research

My solution will be a windows-based application that when opened a menu screen will open allowing the user to navigate the POS system. There will be various buttons that lead to features. There will be a POS button which takes the user to the area where they will take orders, there will also be a button that takes you to the products list where you can view stock and alter or add new products, and a button to clock in and out; once an order has been taken and confirmed it will be recorded in the “Orders” section, there will also be a sales reports section.

2.3.4.2 Limitations of my Solution

The main limitation of my solution will be that it won't automatically print out orders to the kitchen, this means an employee would have to manually print out an order and deliver it to the kitchen so that the chefs know what to cook. I believe I could add this feature however it would be difficult and would require extra hardware and would take lots of time. In addition, my solution is not meant only for restaurants or cafes it is intended to also be used in regular shops therefore it would be somewhat unnecessary to add a feature that is only intended for use in restaurants or cafes. Although, there could be a use for this feature in regular shops in the way of printing receipts and while this would be useful, I plan to allow the user to send receipts electronically to the customer by email.

2.3.5 Further Meetings with Stakeholders

I met again with both stakeholders and explained to them the initial concept as described above and outlined any features I planned to add.

Responses:

2.3.5.1 Restaurant Managing – Mark

“Yeah that sounds good the menu screen seems much easier to navigate than the design on our current POS I like the sound of the orders section as I often have to filter through receipts to find specific orders. One thing you could add is a feature which gives employees access only to what they need as I wouldn't want all of my employees being able to check our sales.”

2.3.5.2 Analysis – Mark

Mark seemed to be happy with the menu design and found it easier to navigate than his current POS so I will aim to stick with that design, he also said that it would be useful to have a feature which gives employees access only to the features they need, in order to do this I could add a feature that allows the admin or manager to create employee accounts and give each employee a type which determines what features they can and cannot access.

2.3.5.3 Waiting – Luke

“Yes that sounds much easier to navigate than the one we have now which should save me a lot of time when taking orders, also the stock tracking feature will be very useful. You could also add a

Analysis

feature that locks the user out if they haven't been active in a certain amount of time as I have had times where I was still logged in and a drunk customer started messing about with the POS system."

2.3.5.4 Analysis – Luke

Luke also seemed to be happy with the menu design and said it would be much easier to navigate and also said that he would find the stock tracking feature useful so I intend to add this feature. He also suggested I add a time out feature that logs the user out if they haven't been active so I might add this feature if I have time.

2.4 Requirements

2.4.1 Software and Hardware Requirements

2.4.1.1 *Hardware*

A computer capable of running the software – Most laptops and computers are capable of running the software. They would need a fast enough processor and enough RAM. Standard peripherals would be a mouse, keyboard, and monitor.

2.4.1.2 *Software*

Windows, Mac, or Linux OS – All these systems are supported by Java.

Java Interpreter and Swing Libraries – The program will be written in Java using swing libraries and a GUI builder so the computer the software is being run on must have the swing runtime libraries downloaded and have a Java interpreter.

2.4.2 Stakeholder Requirements

2.4.2.1 Design

Requirement	Explanation
Lightweight Design	The design needs to be easy to navigate and simple to use.
Centralized Menu Screen	A menu screen from which all other major functions/sections can be accessed.
Product Categories	Products can be put into categories to make the POS design more efficient and lightweight

2.4.2.2 Functionality

Requirement	Explanation
Form and confirm orders	This is the main function of the system, forming an order from product created by the user and confirming the order.
Track product stock	The user will be able to set the stock amount for a product and when that product is sold the amount will decrease by the quantity of the sale.
Track Orders	When an order is confirmed, it will be recorded in the orders section.
Create sales reports	Display monthly sales, profits, average sales, most sold product.
User can create products and categories	The user should be able to create products and add details about the products such as name, picture, sell price, cost and be able to add the product to a category which can also be created and named.
Clock in/out	The user can enter a password (chosen by them) and they will be given the option to clock in, when they clock in the time will be recorded. When they enter the password again they will be given the option to start a break or clock out, once done the time will be recorded and the hours between the two entries calculated and recorded.
Verify data validity (universal requirement)	The user mustn't be able to enter invalid data. Invalid data can make the Java UI crash, or it can put bad data in the database and make the database storage crash. If the system needs a number entering, then the user must not be able to enter a character value. This is the same for integer, decimal, and date type data.

Analysis

	<p>My system will use a relational database so it is important that the user can't delete a parent record that relates to a child record otherwise the child record becomes an orphan.</p> <p>This is called referential integrity. The system will prevent records being deleted if they are parents of other records in other tables.</p> <p>This is done by having a primary key in one table and having this key be a foreign key in another table. Then if the user tries to delete a record with a primary key that has a foreign key constraint on a record in the second table, it won't be allowed.</p> <p>All the valid data types are stored in the database so I might try and get them from the database for the validity checking. Then if the database is ever changed like a field for a number is increased from 4 digits to 6 digits then the system may not need to be manually changed as it will change automatically to match the database.</p>
--	---

2.4.2.3 Hardware and Software

Requirement	Explanation
Windows, Linux, or Mac operating system	All of these operating systems are supported by Java.
Java with Swing and MySql	Java is the programming language I am using. Swing is the GUI builder I am using to build the user interface. MySql is where the database is that has all the tables that are accessed In the program.
Computer Specs: Enough to run the software	The computer must be fast enough and powerful enough to send data to and receive data from the database (any modern day computer should be able to do this)
Computer with a mouse/touchpad, keyboard, monitor/screen	The user will need a computer to run the software and will need the peripherals to navigate the software.

2.5 Success Criteria

2.5.1 Success Criteria

Number	Criteria	How to evidence/test	Justification
	Must		
1	Track stock (automatically decrease stock per purchase)	Screen recording of order confirmation and decrease in stock after order + screenshot of code	All the POS systems I reviewed had this feature + stake holders said it would be useful
2	Form orders (calculate price, add meals or products to an order)	Screen recording of adding products to order and confirming order	It is the main feature of a POS system + all reviewed solutions had this feature
3	Allow user to add an item with custom name, picture, price, multiple products	Screen recording of adding an item	All reviewed solutions had this feature
4	Have a Keypad	Screen shot of keypad	Some of the reviewed solutions had this feature + stakeholder said it would be useful
5	Allow user to add items to categories	Screen recoring of adding a category	All of the reviewed solutions had this feature
6	Track orders	Screen shot of orders list	All of the reviewed solutions had this feature
7	Create and display Sales Reports/analytics	Screen shot of sales reports	Some of the reviewed solutions had this feature
8	Notify the user when there is low stock for a product (with a customizable low stock level which the user can change from product to product) and have a negative stock alert	Screenshot of the notification used to alert the user	All of the reviewed solutions had this feature, although they implemented it in different ways
9	Clock in/out function (calculate hours between clock in and out)	Screen recording of clocking in then waiting a minute or more then clocking out	Some of the reviewed solutions had this feature
10	Card/cash payment	Screen shot of card entry page	Necessary for a POS system

Analysis

11	User Specific Menu Screens (users will have different user types which determine what functions they have access to)	Screen recording of logging in as different user types to show how access is limited for specific users	Stakeholder requested this feature
12	Login System	Screen recording of logging in	Necessary so that a non-employee cannot use the system
	Could		
13	Barcode Usability (so the user can make a product with a specific barcode, then they can scan this barcode with their camera and will be given options to either add the product to the order or edit/alter the product)	Screen recording of this in use	Some of the reviewed solutions had this feature
14	AI that predicts various things such as how busy it will be, what your most sold product will be on a certain day based off factors such as weather, if it's a national holiday, what day of the week it is etc	Screen shot of code facilitating this	Would be a very useful feature provided it is accurate
15	Calculate employees' wages based off hours worked in the week/month and hourly wage	Screen shot of calculate wage with hours worked and hourly wage included	Would be a useful for some of the stakeholders
16	Have a filter/sort selection function for the product list/order list	Screen recording of this feature in use	All of the reviewed solutions had this feature
17	Have a search bar for the orders list/products list, this will allow the user to search by order name, item name, order cost, item cost.	Screen recording of this feature in use	Some of the reviewed solutions had this feature
18	Customizable UI	Screen recording of this feature in use	Some of the reviewed solutions had this feature
19	Automated discount system	Screen recording of this feature in use	Would be a very useful feature for the stakeholders

Analysis

20	Receipt system	Screen recording of entering email and receiving notification	Some of the reviewed solutions had this feature
21	Customizable colour theme	Screenshot of this feature in use	Some of the reviewed solutions had this feature
22	Discount creation	ScreenShot of option to create a discount	Some of the reviewed solutions had this feature
23	Time out function	Screen recording of timing out	Stakeholder requested it

2.5.2 Success Criteria for Universal Requirements

Number	Criteria	How to evidence/test	Justification
Must			
UR1	Notify the user if they try to enter an invalid integer	<p>Screen recording of a product being added and the user entering a character in the Stock field which is an integer.</p> <p>Screen recording of a product being added and the user entering a number in the Stock field which is too many digits.</p> <p>Screen recording of a product being added and the user entering a null or nothing in the Stock field which is required.</p>	The user mustn't be able to enter invalid data.
UR2	Notify the user if they try to enter an invalid decimal	<p>Screen recording of a product being added and the user entering a character in the Cost field which is decimal.</p> <p>Screen recording of a product being added and the user entering a number in the Cost field which is too many digits.</p> <p>Screen recording of a product being added and the user entering a null or nothing in the Cost field which is required.</p>	The user mustn't be able to enter invalid data.
UR3	Notify the user if they try to enter an invalid character field	<p>Screen recording of a product being added and the user entering too many characters in the product name field.</p> <p>Screen recording of a product being added and the user entering no characters in the product name field which is required.</p>	The user mustn't be able to enter invalid data.

Analysis

UR4	Notify the user if they try to enter an invalid date field	Screen recording of a product being added and the user entering a bad date	The user mustn't be able to enter invalid data.
-----	--	--	---

2.5.3 Success Criteria Justification

My POS system will be windows based meaning it will be made for use on a windows OS, there are many reasons I plan to make this POS system for windows. Firstly, I have access to a windows system to build, test and run this app making it very convenient. Secondly, windows systems are very common and readily available meaning it is very easily accessible for someone to use my POS system.

The reason I decided to add the stock tracking feature and the reason it is a must add feature is because it was a feature in all three of the POS systems that I tested. I believe stock tracking is a very useful feature for a business owner to have in order to help them keep track of their stock levels for their products without manually counting stock. This feature would minimise the possibility for human error when checking stock levels and re-ordering stock and would also save time.

The reason there will be a form order function is because it is the primary function of any POS system and is required for the user to be able to write up customers when they want to buy a product or make an order. Since this function will calculate the price, it will also save time as there won't be a need for an employee to calculate how much the customer's order is based off of what they ordered.

The reason that allowing the user to create items with a name, picture, price and possibly multiple products is a must add feature is because all three of the POS systems I tested had this feature. Furthermore, by adding this feature it ensures that my POS system can be used by anyone and is not limited by what products can be added.

The reason there will be a keypad is because this will be very useful for writing up customers and two out of the three POS systems I reviewed had this function. The keypad allows the user to create a custom price for the order regardless of what products were sold so the POS app can be used in any situation making my system more versatile.

The reason there will be a category addition feature is because by allowing the user to add categories which they can add products to, will mean the user can search for a product much faster than if they had to search for each item individually making this system more time efficient during actual use. Being time efficient is very important for a server at a restaurant or shop.

The reason that my app will track the orders is because this allows the user to look back on older orders for any information needed. This function is similar to how a business would store receipts, this way is much more time efficient and is much more feasible long term.

The reason my app must have a system that creates and displays sales reports/analytics is because this is very useful for the user as it allows them to view how much revenue or profit, they made per day as well as what their average order price was. In addition, it allows the user to visually see how a certain statistic has changed over the days, weeks, months, or years such as how many orders they have, how much profit they made, how much revenue they made. This can help them identify the effects of any changes they have made. One of the main reasons I decided to add this feature is because all three of the POS apps that I reviewed had sales analytics and reports.

Analysis

The reason my POS app will notify the user when there is low stock and have a negative stock alert system, is because the “Zettle” POS app that I reviewed notified the user in the basket when there was low stock and had a customizable low stock level. Also, the “Loyverse” app had negative stock alerts for if the user tried to add a larger quantity of a product than they had in stock. By adding this feature, it would remove the chance for the user to charge the customer for a certain quantity of a product and then have to refund the order if they didn’t have enough of that product in stock. Furthermore, this feature would give the user advanced notice on a products low stock before it runs out so they can re-order more of that product.

The reason my POS system will have a clock in and out function and will calculate the number of hours between when an employee clocks in and out is because the “Loyverse” POS app I reviewed had this system. This makes it much easier to calculate the wages for each employee without having to remember when they clocked in and out and calculate the amount of hours themselves. This way the user only has to multiply the calculated hours by the employees’ wages to calculate how much to pay them.

My POS system will give two options for payment when taking an order which will be cash and card. For the card option my system will simply take the card details entered by the user and store them however it won’t actually take money from the card if the user would like to take card payment they would need a card reader. The cash option would just be entering an amount of cash paid to keep track.

The reason my POS system will have a login function is so that employees will have access to certain features based off what their role is. The admin user will be able to create employee logins where each employee can enter a username and a password, and the user can select what role each employee has. For example, if a waiter logs in they will have access to the POS function but won’t have access to the sales reports function, if a manager logs in they will have access to all the functions; by adding this feature it means the manager can stop the waiters or other employees from checking sales or altering products if they don’t want them to.

The reason there might be a discount creation system is because 2 out of 3 of the POS apps that I reviewed had this feature and I believe it is a very useful feature for a POS system to have. It allows the user to set up deals to incite people to buy more products and apply the discount to the order. This way they don’t have to give the order a custom price using the keypad and so the user can quickly discount orders by a percentage without having to calculate the order amount on the go which can be very difficult and lead to mistakes such as over or under charging the customer. This feature also makes this system more time efficient as the user doesn’t have to calculate the discounted order price.

The reason my POS system might have a barcode orientation system is to help the user find products and edit them. However, this feature is not necessary as my POS system will likely have a search bar so that users can search for a specific product. One reason I might add barcode orientation is

Analysis

because “Zettle” had this feature. Although I probably won’t add this feature as my app will be windows based and lots of windows-based systems are laptops or computers rather than tablets or phones so they might not have a camera and I would also have to learn how to make a barcode scanner which is very difficult and would be very time consuming.

I might add an AI that uses old data in order to predict the number of orders the user might get on a certain day, what your most sold product will be on a particular day etc, this would be based on various factors such as weather, day of the week, time of day, if it is a national holiday etc. I could add this because it would separate my POS system from all the other POS systems that I have reviewed as none of them had this feature, also it would give the user notice so they can prepare for certain situations such as calling in more staff if it is predicted to be busy. However, I do not have an in depth enough knowledge of AI to create this so I will only add this feature if I have time to learn about AI and learn how to code AI and integrate this within my system.

The reason I might add a feature that calculates the employees’ wages by multiplying their hours by their hourly wage is because it would save time as the manager wouldn’t have to calculate it themselves. Also, this would remove the risk of human error such as the manager over or under paying employees by calculating their wages wrong. However, as this feature is not necessary to make my POS system a high quality one; I will only add it if I have time.

The reason I might add a filter/sort is so that the user can find products and orders quicker by filtering them by a condition such as order amount or stock level. Another reason I might add this feature is because “Zettle” had this feature, and it is very useful. I also will likely add a search bar so that the user can search for products and orders either by name or order price, however I will not add these features if I run out of time.

The reason I might add a customizable UI is because the “Square” app had this feature and it is helpful to customize the UI to make it more unique and suited to the users own needs however, this is an ease of life feature and I might struggle to add this feature or run out of time so it’s not a priority.

An automated discount system is where if a certain selection of products are ordered a discount is automatically applied. The reason I might add this feature to my system is because this would be quicker than having the user manually add the discount making taking orders much more time efficient. I also might add a receipt system that sends the customer an electronic receipt either by text or email. The reason I will likely add this feature is because both “Zettle” and “Square” had this feature and I believe it is very important as customers often ask for receipts.

The reason I might add a customizable theme colour for the system is because one of the POS systems that I have reviewed had this feature and it allows the user to make their POS system unique to their business. This feature would not affect or improve functionality of the POS system

Analysis

and because of this it will not be a priority when I am coding my POS system and I will only add it if I have time and all the more important features have already been added. Another reason I might not add this feature is because if I allow the user to change the background/theme colour I would either have to limit the number of colours they can use, allow them to change text colour as well, or automatically change the text colour so that they can still read text. This would be necessary because if they make the background black, normally text would be black so the text would be invisible to the user. This was a problem I found when using this feature with the “DailySalesRecord” POS system as when I changed the background to white I couldn’t see any of the menu or text.

2.6 Pre-development Limitations

Barcode orientation system

The reason I will probably not add this feature to my POS app is because it is beyond my technical ability and although I have an understanding of how barcode scanners work, I wouldn't know how to implement or create the code to make the camera scan a barcode. In addition, for me to learn the skills I would need, it would take lots of time and I would not have enough time to add other more important features.

AI feature

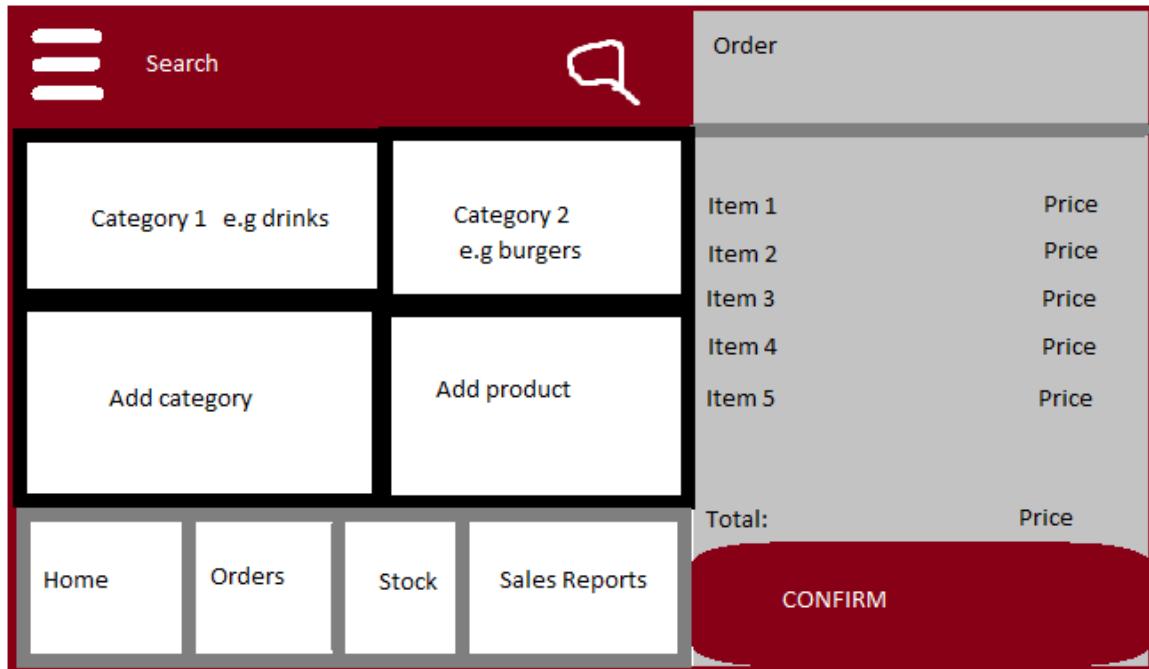
The reason I probably won't add an AI is because it is beyond my technical ability, I have a very limited knowledge of how AI works and I have no knowledge of how to code AI so in order to learn the skills I would need, it would take too much time and therefore I wouldn't have time to implement other more important features. If I have time, I will attempt to learn how to create this AI and attempt to implement it as I do believe it is very possible and there are lots of resources on how to code AI.

Card Payment System

The reason I might not add a system that takes payment from a card is because I do not have the technical knowledge nor the technical ability to code this system and I believe this would be very difficult to learn how to do therefore it is unlikely that I will add this feature.

3 Design

3.1 User Interface Design 1



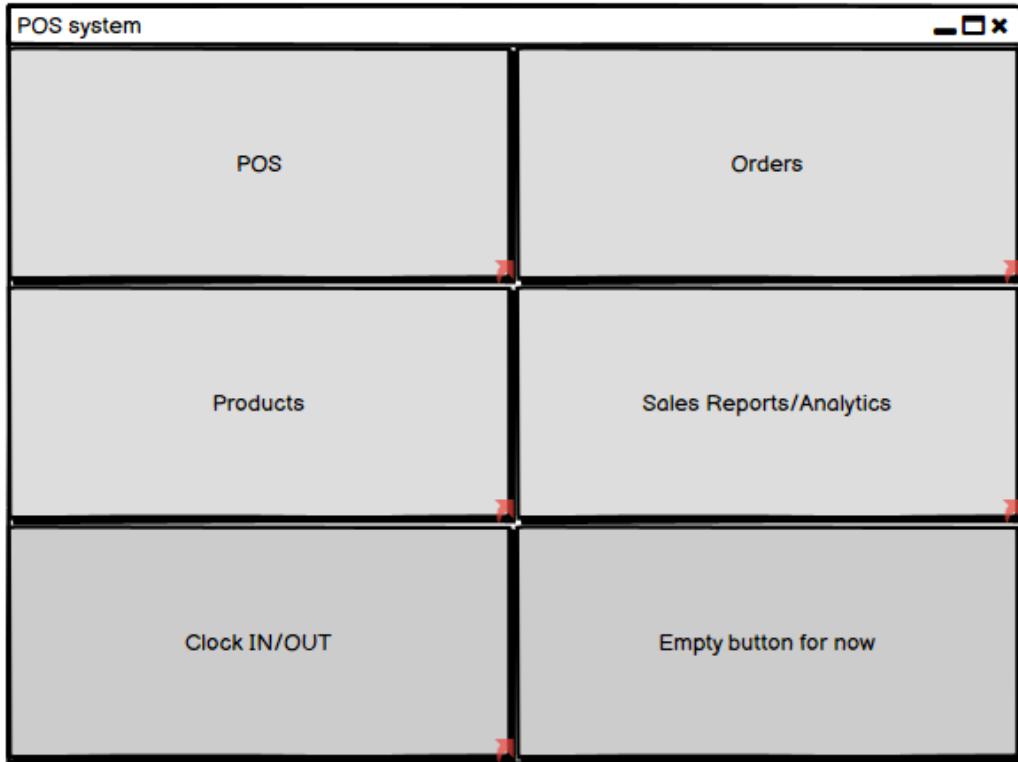
The design above was my initial design, after reviewing another POS system called “DailySalesRecord” and talking to my stakeholders I have decided to change the GUI to design 2. I believe design 2 is much easier for the user to navigate which is important for a POS system as the user will be serving customers real time so they must be able to navigate the system quickly and with ease.

3.2 User Interface Design 2

3.2.1 Menu

This is the menu screen which opens after entering a valid username and password at the Login screen. Clicking on any of the titled buttons brings up a new page corresponding to the titled designs below.

3.2.1.1 *Menu Screen*



3.2.1.2 *Usability*

- The buttons will be large and clearly labelled so the user can select each function with ease.

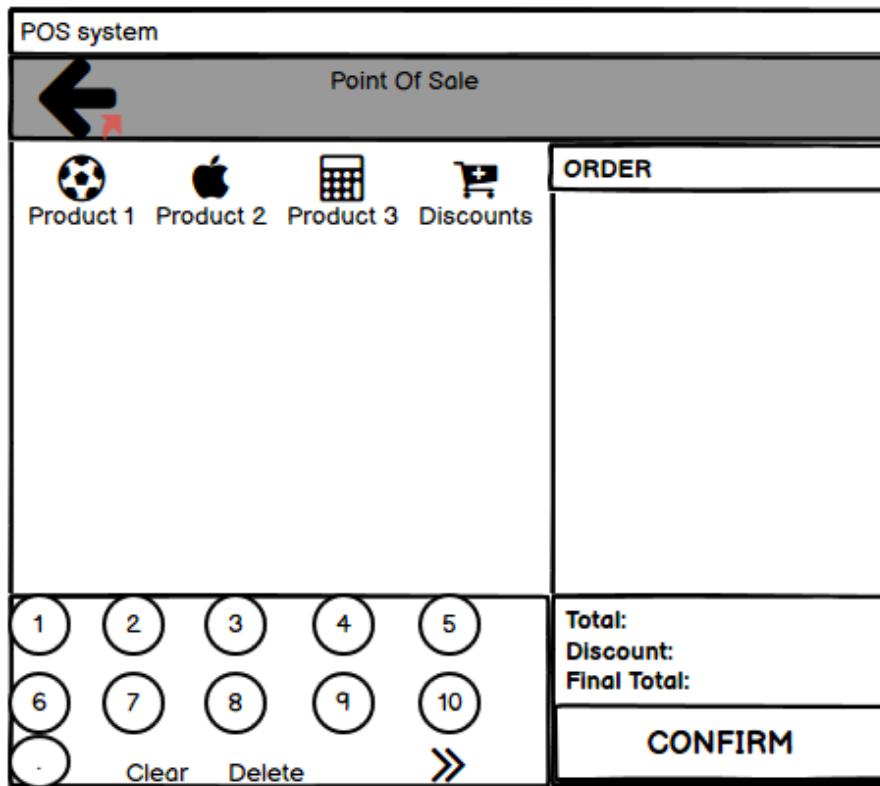
3.2.2 POS

Clicking on the products will add them to the order section on the right and update the total fields at the bottom.

The user can use the keypad to create a custom price.

Clicking on “Discounts” will allow the user to add a discount to the order which can be done either by percentage or flat deduction.

3.2.2.1 POS Screen



3.2.2.2 Links to Success Criteria

- Form Orders
- Has a keypad
- Can create Discounts

3.2.2.3 Usability

- Clear buttons for confirm, products, and keypad so the user can form and confirm an order with ease.
- Have a clear and delete button so it is easier for the user to alter the order total.

3.2.3 Products

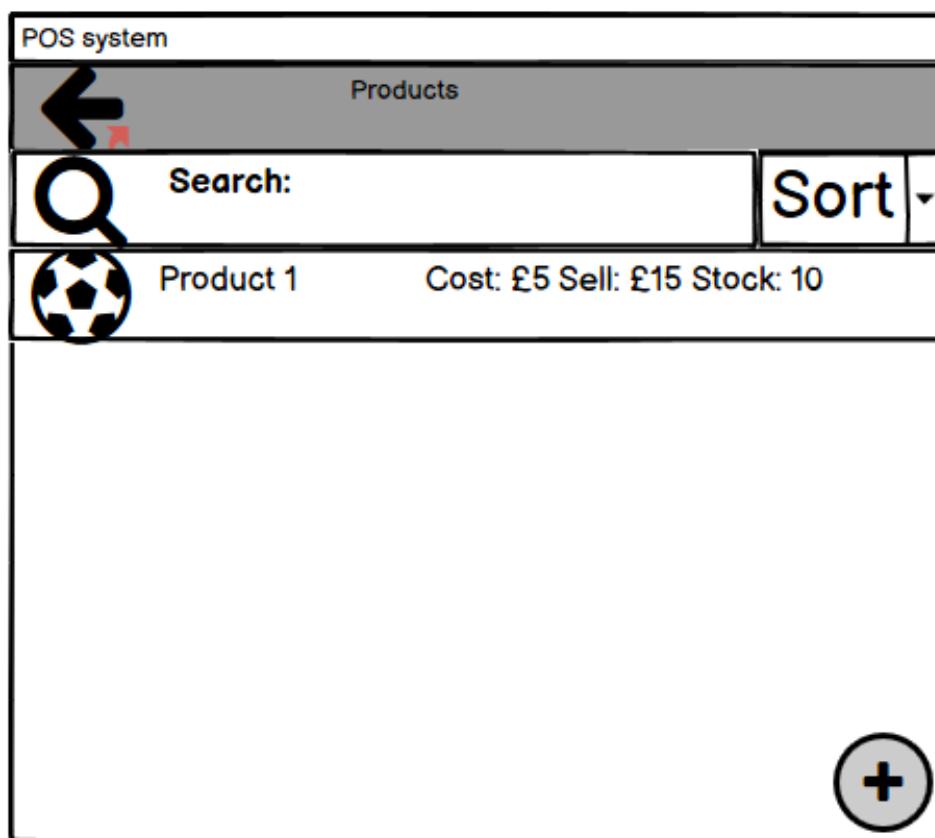
Clicking on the product will allow the user to edit various features of the product such as name, picture, cost, sell price, stock, low stock level etc.

The user can use the search box to search for a specific product by name.

The user can use the sort box to order the products in a certain manner such as alphabetically.

Clicking on the + symbol creates a popup which can be used to create a new product and add it to a category; the user can also create a category in this popup.

3.2.3.1 Products Screen



3.2.3.2 Links to Success Criteria

- Track Stock
- Allow user to create products/items
- Allow user to add items to categories

3.2.3.3 Usability

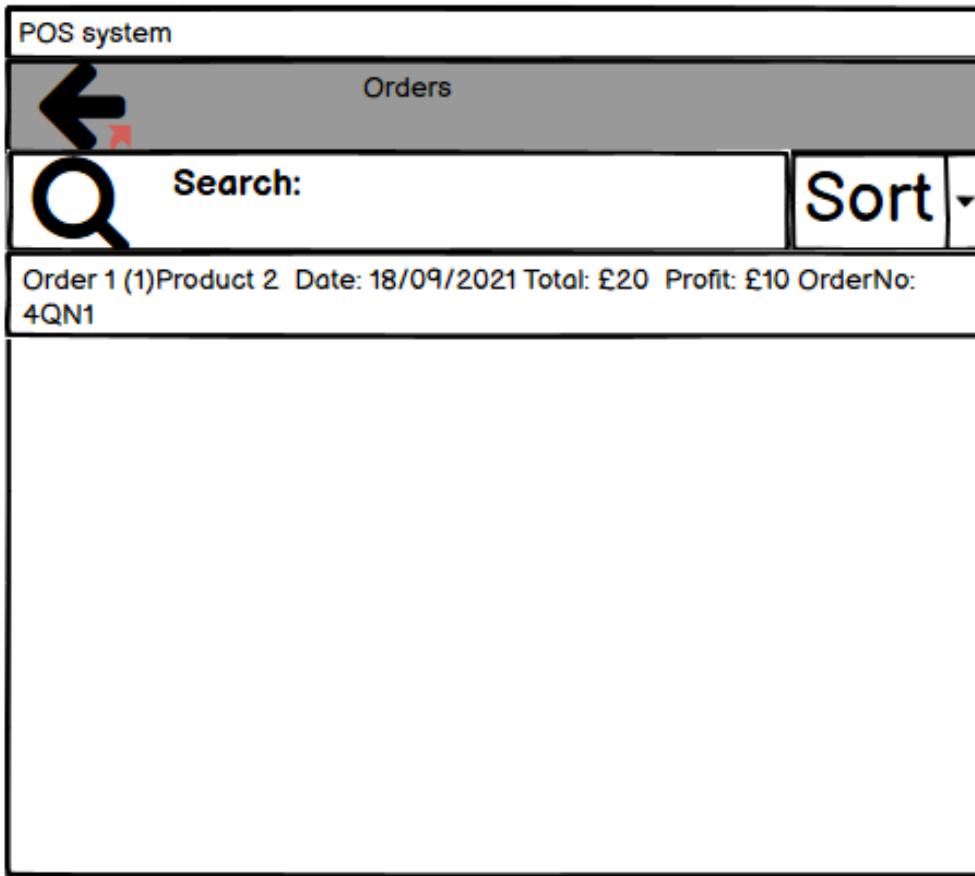
- Clear + button to add a product.
- Search and Sort function to allow the user to find products with ease.

3.2.4 Orders

The user can use the search box to search for a specific product by name.

The user can use the sort box to order the products in a certain manner such as alphabetically.

3.2.4.1 *Orders Screen*



3.2.4.2 *Links to Success Criteria*

- Track Orders

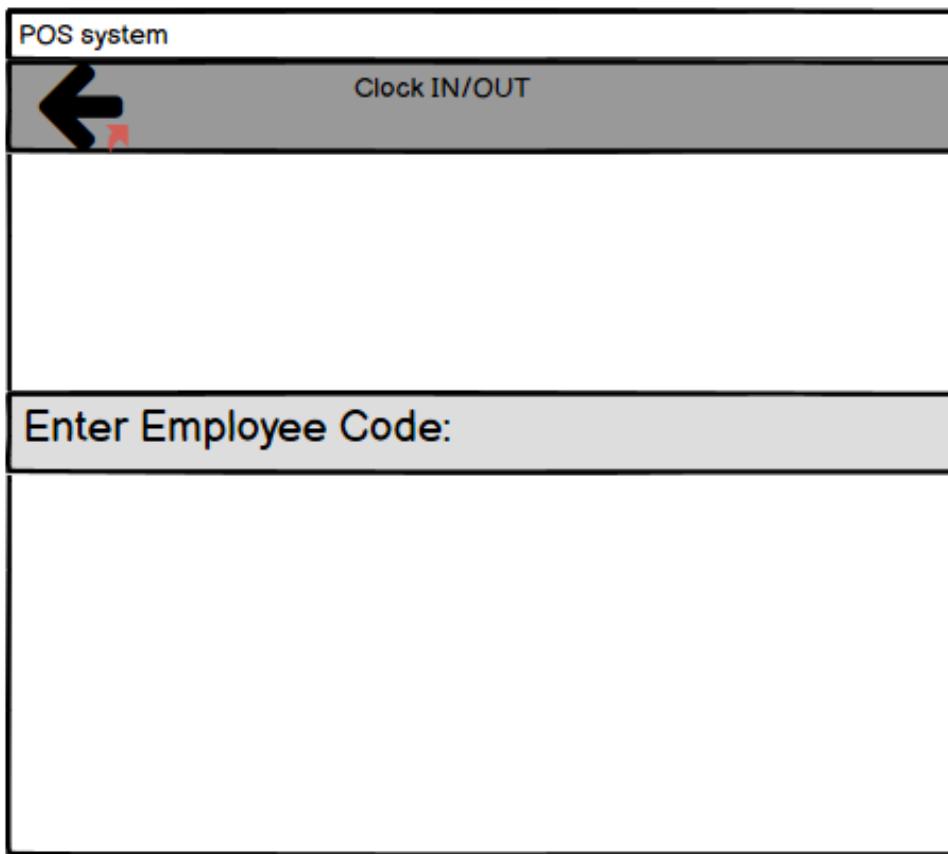
3.2.4.3 *Usability*

- Search and Sort function to allow the user to find products with ease.
- Clearly displayed orders and order details.

3.2.5 Clock IN/OUT

The user can enter their employee code in the box and a popup will open with the option to “clock in”. Once the user has clocked in the system will record the time and if they enter their code again another popup will open with two options “start a break” and “clock out”. If the user clocks out the time will be recorded and the hours between clocking in and out will be calculated and recorded; if the user starts a break, then the time will be recorded and when they enter their code again they will be given the option to “stop break” and the time will be recorded once pressed. The total hours worked once the user has clocked out will be recorded and displayed.

3.2.5.1 Clock IN/OUT Screen



3.2.5.2 Links to Success Criteria

- Clock in/out function

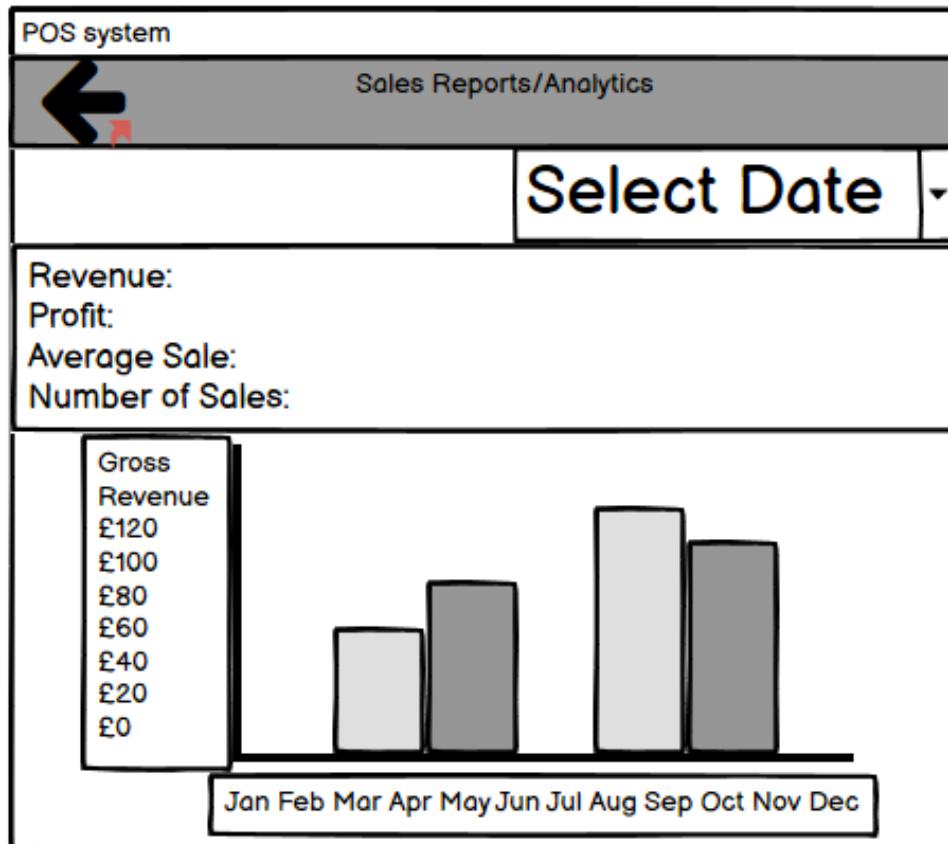
3.2.5.3 Usability

- Clear type box for user to enter their code in.

3.2.6 Sales Reports/Analytics

The user can use the “Select Date” dropdown box to select a date or month for which they want to see the sales analytics.

3.2.6.1 Reports Screen



3.2.6.2 Links to Success Criteria

- Create and Display sales analytics

3.2.7 Admin

The system will also need an admin button which will take the empty button space on the menu screen. The admin button will open a new screen from which the admin can create, delete and edit users and also choose what user type each user is. I'm planning on having 4 user types, admin: has access to all functions and buttons, sales: has access to POS-products-orders-clock in/out, kitchen: has access to orders-clock in/out, and warehouse: has access to products-clock in/out. In addition, all users will have access to the login feature and the menu feature as they will need to navigate the system and be able to login so the system knows their user type and can limit access.

3.2.7.1 *Usability*

- Clearly displayed analytics such as profit and number of sales.
- Clearly displayed and labelled graph.
- Select date function to sort so only one specific month of sales analytics is showing so it's not too cluttered and hard to read.

3.3 Stakeholder input

I met again with both stakeholders Luke and Mark and showed them the screenshots above and explained how the system would work as is written above these were their responses:

3.3.1.1 Restaurant Managing – Mark

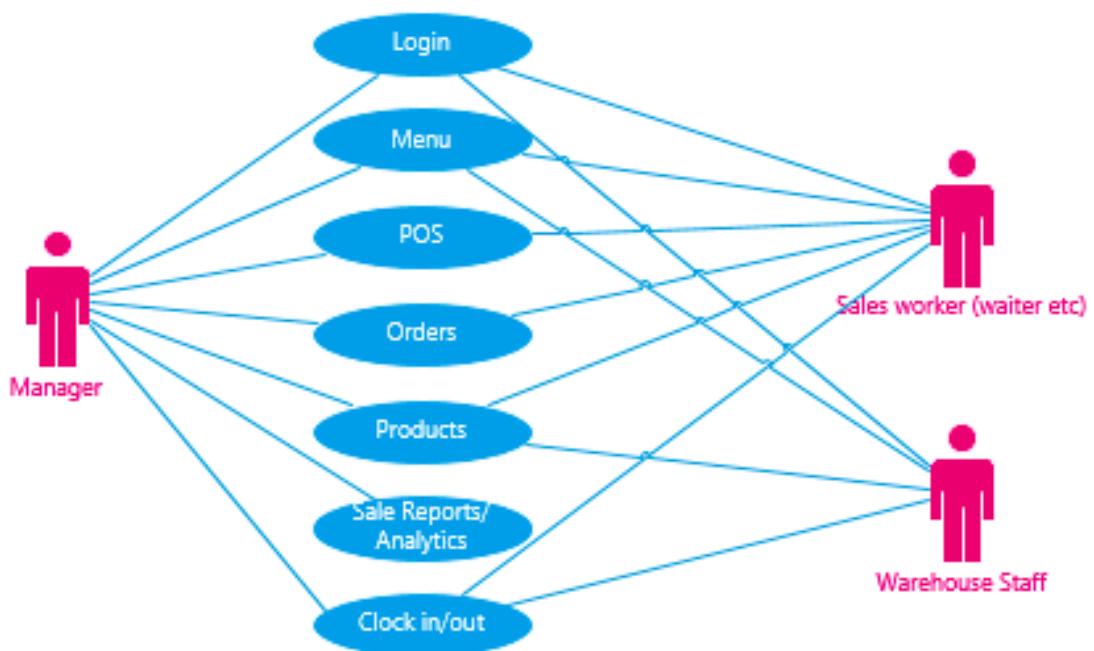
“These designs look great; it looks very easy to navigate and sounds easy to use. I like the menu screen it’s pretty much exactly what I would’ve asked for on our current app.”

3.3.1.2 Waiting – Luke

“Yeah it looks like it’ll be very easy to use which is perfect, taking orders seems to be pretty ideal in terms of the design and having the keypad at the bottom for easy access.”

3.4 System Design

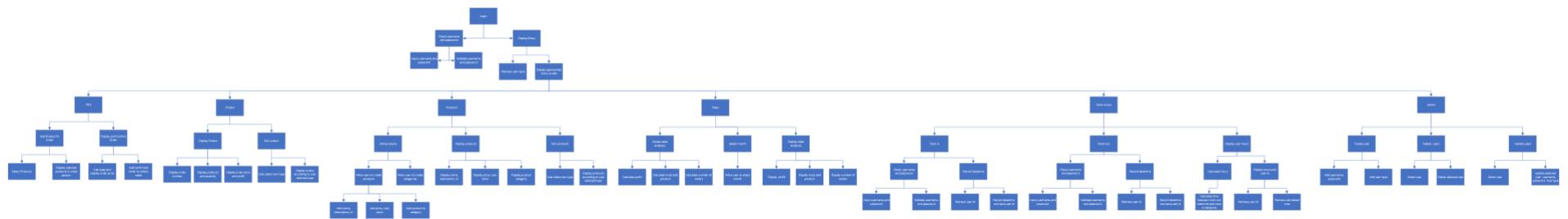
3.4.1 Overall System Use-Case Diagram



The Manager would also have access to the “admin” button not shown on this diagram.

Design

3.4.2 System structure Diagram



This structure diagram is broken down into parts below.

3.5 Stages

3.5.1 Notes/Universal algorithms or pseudocode

I won't put any algorithms or write any pseudocode for features that aren't in the **must** section of the success criteria.

I won't write the pseudocode in every stage for algorithms that are going to be used in a lot of stages I will write them below:

Whenever there is a jTable that is being populated with data from a table from the database for example in the products code, the code below will be necessary aswell.

FillArrayWithRecordsFromTable - Pseudocode

DataArray – pseudocode

```
recordList ← Array(element.Model)[]  
queryStatement ← “SELECT * FROM tablename”  
dataResults ← queryStatement.results  
recordAmount ← dataResults.length()  
FOR i = 0 TO recordAmount  
    productList[i] ← dataResults[i]  
END FOR
```

DisplayDatabaseTableInJtable - Pseudocode

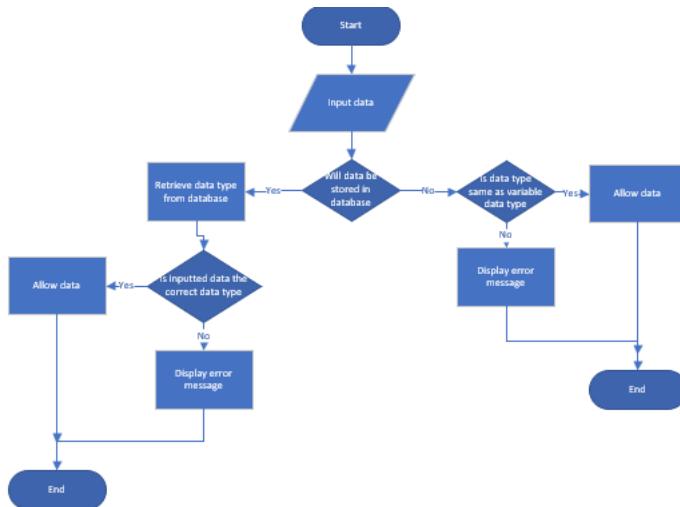
```
For i = 0 TO recordAmount  
    Jtable.Row[i] = recordList[i]  
END FOR
```

3.5.1.1 *dataVerifier – Flowchart*

I need a universal algorithm to check data entered. Something I can call whenever a user inputs data to a screen and that data needs to be validated. If possible, the algorithm should be quite generic so I can use it everywhere and not have to make a separate validation algorithm for each screen and data input and for each type of data being validated. The algorithm should check the data input and either accept the data or provide an error message to tell the user to amend the data they just typed.

All the data types that are needed to store the data in my SQL database are in the database meta data – this is the data inside MySQL that describes the data in my POS database. All my tables and the columns in the records in the tables are described in this meta data including the name of each column and the type of data that is stored in the columns. I will try to use this information when I validate the data input. For example, I can see if a data input is going to be stored in the database as an integer, so I know to validate the data input as a valid integer. If I can't do this then I will have to make separate validation for each type of data which is a lot more code. Also if I can make the validation work from the database meta data then if the database table structure is changed then the validation will stay up to date. For example if an integer field is changed from length 4 digits to 6 digits my validation would allow the 6 digits without needing me to change the Java code.

If a data input isn't for a database table data item then I might still want to validate it. So I need to check if it is a database validation or a static (non-database) validation. If it's a database validation I will try to get and use the datatype from the database. If it is static I will do the validation in Java without using the datatype from the database.



3.5.1.2 *Success Criteria Satisfied*

Number	Description
UR1	Notify the user if they try to enter an invalid integer
UR2	Notify the user if they try to enter an invalid decimal

UR3	Notify the user if they try to enter an invalid character field
UR4	Notify the user if they try to enter an invalid date field

3.5.1.3 Algorithms

I need to know in the validation code how to validate the value. I will check this and then validate. If it is going to be a database validation then I'll use the database table definition in the database for the data type otherwise I will not go to the database but will do it in the Swing code statically.

ValidateInputtedData – Pseudocode

```

String field_value ← INPUT
String validation_method ← CONSTANT
IF (validation_method=="DATABASE") THEN

    Get the database datatype for the column that will store the field value inputted
    Make the validation code from the datatype
    Validate the data value

    IF field_value IS INVALID THEN
        Error message ← OUTPUT
        STOP
    END IF

ELSE IF (validation_method == "STATIC") THEN

    IF field IS NOTNULL AND field_value ==NULL THEN
        Error message ← OUTPUT
        STOP
    END IF

    IF field IS CHARACTER THEN
        IF field_value has too many characters THEN
            Error message ← OUTPUT
            STOP
        END IF
    ELSE IF field IS INTEGER THEN
        IF field_value NOT NUMERIC THEN
            Error message ← OUTPUT
            STOP
        ELSE IF field_value has too many digits THEN
            Error message ← OUTPUT
            STOP
        END IF
    ELSE IF field is a decimal THEN
        IF field_value NOT NUMERIC THEN
            Error message ← OUTPUT
            STOP
        ELSE IF field_value SIGNIFICANT DIGITS too many digits THEN
            Error message ← OUTPUT
        END IF
    END IF

```

```

STOP
ELSE IF field_value DECIMAL DIGITS too many digits THEN
    Error message ← OUTPUT
    STOP

ELSE IF field is a date THEN
    IF field_value NOT VALID DATE THEN
        Error message ← OUTPUT
        STOP
    END IF
END IF
END IF

```

3.5.1.4 Test Plan

Test Number	How to Test	Expected Result
1	Enter a character data that is too long	Error message
2	Enter nothing when a character data is required	Error message
3	Enter a bad integer data – put a character in the number	Error message
4	Enter a bad integer data – put nothing if a number is required	Error message
5	Enter a bad integer data – put too long a number	Error message
6	Enter a bad decimal data – put too long a number	Error message
7	Enter a bad decimal data – put nothing if a number is required	Error message
8	Enter a bad integer data – put a character in the number	Error message
9	Enter a bad date data	Error message
10	Enter a bad date data – put nothing if a date is required	Error message
11	Enter a good data for characters, integers, decimals, and date	Allow the data input
12	Change a database table so a data type is changed from character length 100 to 150. Test with >100 characters to see error then make change to 150 then test again >100 characters	Error message when >100 characters then when database changed >100 characters will be allowed

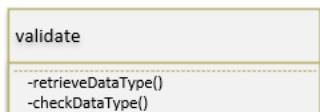
3.5.1.5 Evidence Plan

- Screen Record: entering the bad character data shows an error.
- Screen Record: entering the bad integer data shows an error.

- Screen Record: entering the bad decimal data shows an error.
- Screen Record: entering the bad date data shows an error.
- Screen Record: entering good character, integer, decimal, date data is allowed.
- Screen Record: entering bad character >100 shows error message then extend database table, then test >100 characters and data will be allowed.

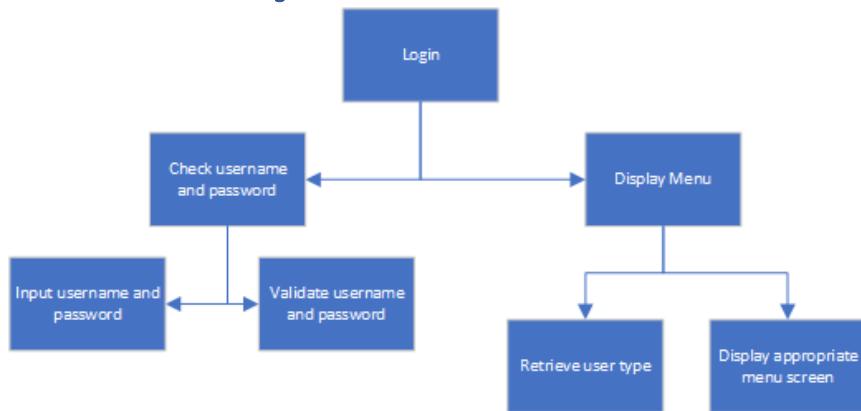
3.5.1.6 Class Diagram

Universal data type validation class

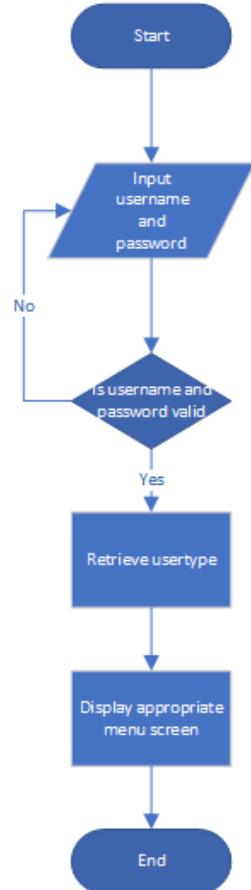


3.5.2 Stage 1/Login

3.5.2.1 Structure Diagram



3.5.2.2 Flowchart



3.5.2.3 Success Criteria Satisfied

Number	Description
11	User login screen

3.5.2.4 Algorithms

ValidateUsernamePassword – Pseudocode

```

String username ← INPUT
String password ← INPUT
IF (username.length() < 6 OR password.length() < 6) THEN
    ("Invalid, username and password must be over 6 characters long") ← OUTPUT
ELSE THEN
    queryStatement ← "SELECT username, password, userpk FROM users WHERE 'username' =
        username AND 'password' = password"
    dataResults ← results.executeQueryStatement
    IF (dataResults == TRUE) THEN
        CREATE menu
        CLOSE login
    ELSE THEN
        ("Username or password incorrect") ← OUTPUT
    END IF
END IF

```

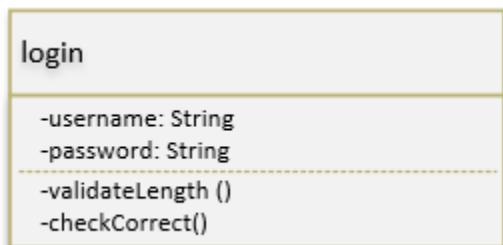
3.5.2.5 Test Plan

Test Number	How to Test	Expected Result
1	Enter username and/or password under 6 characters	Error message
2	Enter incorrect username and/or password	Error message
3	Enter correct username and password	Menu screen opens and login screen closes

3.5.2.6 Evidence Plan

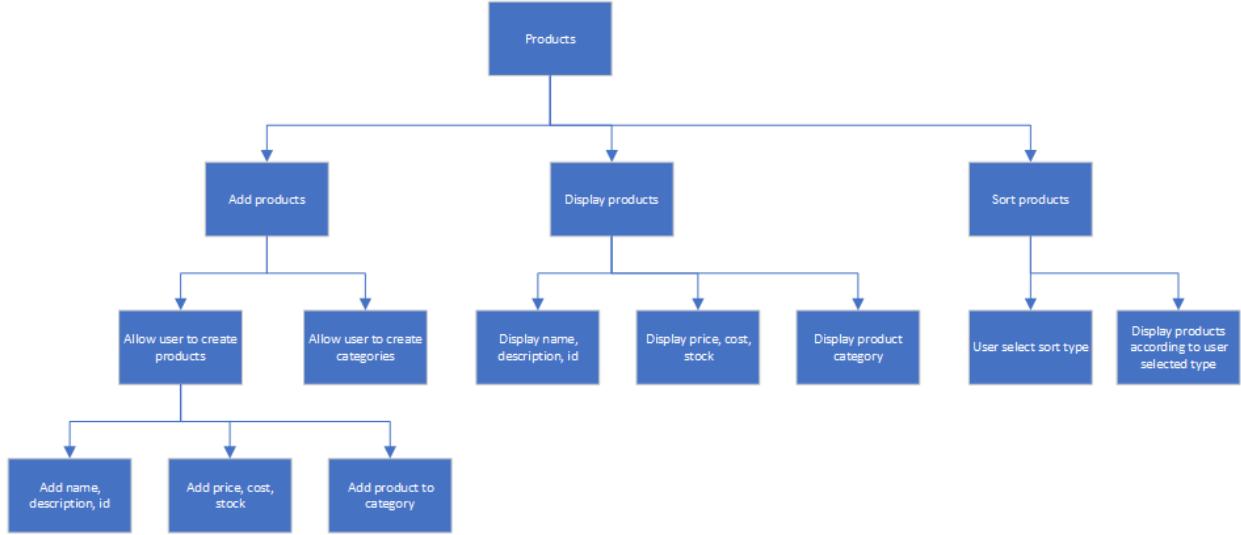
- Screen Record: entering the incorrect username and password shows an error.
- Screen Record: entering the correct username and password opens the menu screen and closes the login screen.
- Screenshots: users table to show that the username and password I entered was correct and to show what user type I logged in as.
- Screen Record: attempting to access features that as a specific user type I don't have access to.

3.5.2.7 Class Diagram



3.5.3 Stage 2/Products

3.5.3.1 Structure Diagram



The structure diagram would also have a delete and update products section, the update section would be very similar to the add product section and the delete section would just be selecting a product and deleting it.

3.5.3.2 Success Criteria Satisfied

Number	Description
3	Allow user to create an item/product
5	Allow user to create categories and add items to categories
8	Set low stock level

3.5.3.3 Plan

- Use a jTable to display products table from database

3.5.3.4 Algorithms

AddAProduct – pseudocode

```

queryStatement ← "INSERT INTO products productName = nameInput, productDescription = descriptionInput, productPrice = priceInput, productCost = costInput, productCategory = categoryInput, productStock = stockInput, lowStockLevel = lowStockInput, productCreatedBy = userPrimaryKey"
  
```

DeleteAProduct – pseudocode

```

queryStatement ← "DELETE FROM products WHERE product_pk = selectedProductPK"
  
```

UpdateAProduct – pseudocode

Design

```
queryStatement ← “UPDATE products SET productName = nameInput, productDescription = descriptionInput, productPrice = priceInput, productCost = costInput, productCategory = categoryInput, productStock = stockInput, lowStockLevel = lowStockInput ,productCreatedBy = userPrimaryKey WHERE productPK = selected.productPK”
```

FillArrayWithRecordsFromTable

DisplayDatabaseTableInTable

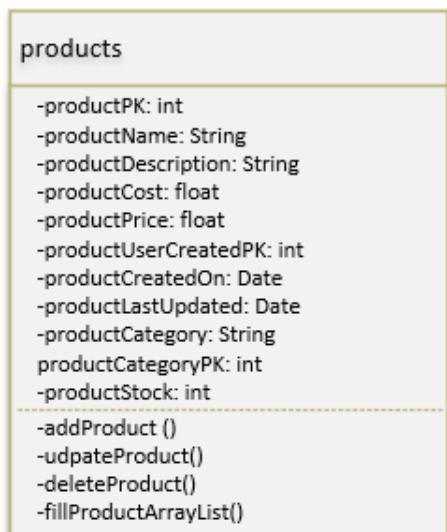
3.5.3.5 Test Plan

Test Number	How to Test	Expected Result
1	Press add product button	add product screen should pop up
2	Enter product details and confirm product	New product should be added and displayed
3	Select a product and press update button	Update product screen should pop up
4	Enter new details and confirm	Product should be updated with new details
5	Select a product and press the delete button	Product should be deleted

3.5.3.6 Evidence Plan

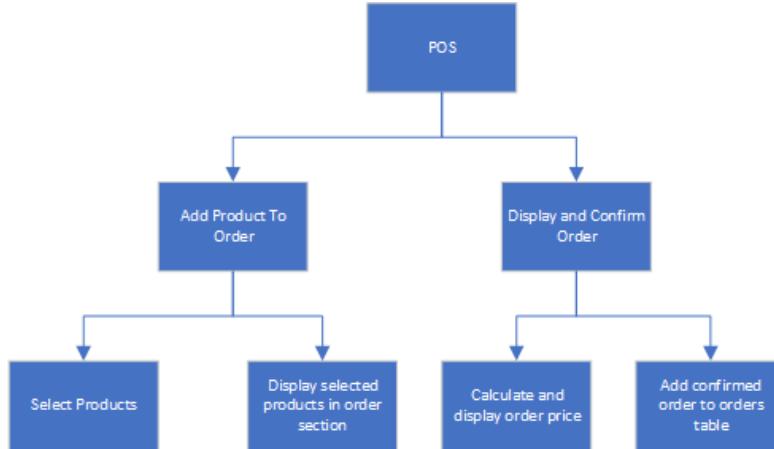
- Screen Record: pressing the add products button and a screen pops up that allows you to add a product.
- Screen Record: entering all the inputs such as name and adding the product.
- Screenshot: added product displayed.
- Screen Record: selecting a product to update and update screen pops up.
- Screen Record: entering all inputs and updating the products.
- Screenshot: updated product.
- Screen Record: selecting a product, pressing the delete button.
- Screenshot: deleted product not displayed.
- Screen Record: product stock before and after order.

3.5.3.7 Class Diagram



3.5.4 Stage 3/POS

3.5.4.1 Structure Diagram



3.5.4.2 Success Criteria Satisfied

Number	Description
2	Form orders
4	Have a keypad
1	Track stock
8	Low stock notification and Negative stock alert
10	Pay with card/cash function

3.5.4.3 Plan

- Use two jTables
- jTable1 will display products and allow the user to select the products, it will display the products table from the database however will only display necessary information.
- jTable2 will display the products added to the order, it will display the currentorder table from the database.
- There will be two buttons, one will add the selected product from the products table and insert it into the currentorder table.
- The other will remove the selected product in jTable2 from the currentorder table and in turn remove it from jTable2.

3.5.4.4 Algorithms

AddProductToOrder – pseudocode

```
queryStatement ← "INSERT INTO currentorder productName, productPrice WHERE productPK = selected.productPK"
```

```
queryStatement ← "SELECT productStock, lowStockLevel FROM currentOrder WHERE productPK = order.ProductPK"
```

```
dataResults ← results.queryStatement
```

```
stock ← dataResults(1)
```

```

lowStockLevel ← dataResults(2)

IF stock <= lowStockLevel THEN

    "You are low on stock for this product" ← OUTPUT

ELSE IF stock <= 0 THEN

    "Warning there are none of this product in stock" OUTPUT

END IF

```

RemoveProductFromOrder – pseudocode

```
queryStatement ← "DELETE FROM currentorder WHERE productPK = selected.productPK"
```

ConfirmOrder – pseudocode

```
queryStatement ← "INSERT INTO orders productPK, productName, productDescription,
productPrice, productCost, productStock, productCategory WHERE currentorderPK =
selected.currentorderPK"
```

```
queryStatement ← "UPDATE products SET productStock = productStock-OrderQuantity WHERE
productPK = orderProductPK"
```

```
queryStatement ← "DELETE * FROM currentorder"
```

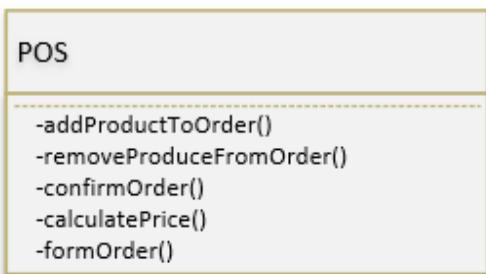
3.5.4.5 Test Plan

Test Number	How to Test	Expected Result
1	Select products and add them to order	Products displayed in order and price calculated
2	Confirm order by pressing "confirm" button	Order added to the orders table
3	Add a product with low stock to the order	Pop up message notifying the user of the low stock
4	Add a product with 0 stock to the order	Pop up message notifying the user of the negative stock

3.5.4.6 Evidence Plan

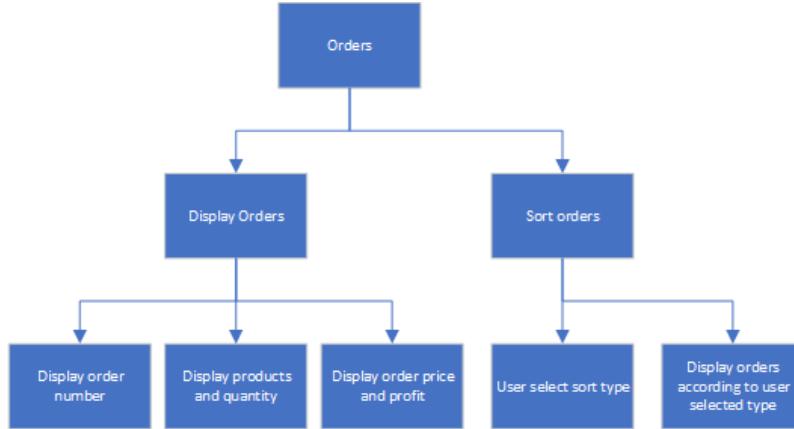
- Screen Record: adding and deleting products to and from orders.
- Screenshot: products in order displayed.
- Screen Record: keypad changing order price.
- Screen Record: low stock and negative stock notification.

3.5.4.7 Class Diagram



3.5.5 Stage 4/Orders

3.5.5.1 Structure Diagram



3.5.5.2 Success Criteria Satisfied

Number	Description
6	Track orders

3.5.5.3 Plan

- Use a jTable to display the orders

3.5.5.4 Algorithms

FillArrayWithRecordsFromTable

DisplayDatabaseTableInJtable

3.5.5.5 Test Plan

Test Number	How to Test	Expected Result
1	Confirm an order	Order should be displayed

3.5.5.6 Evidence Plan

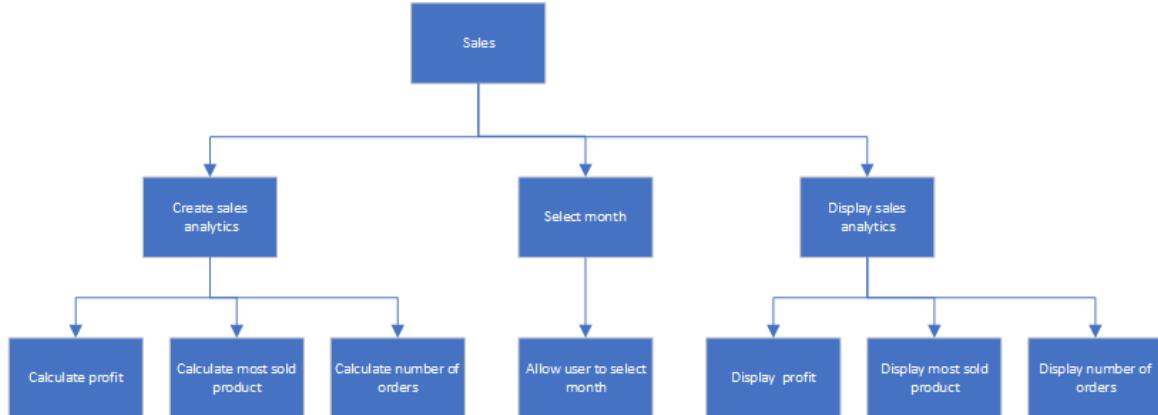
- Screenshot: orders displayed

3.5.5.7 Class Diagram



3.5.6 Stage 5/Sales

3.5.6.1 Structure Diagram



3.5.6.2 Success Criteria Satisfied

Number	Description
7	Create and display sales reports/analytics

3.5.6.3 Plan

- Use a jTable to display sales

3.5.6.4 Algorithms

```
selectedMonth ← JComboBox.getSelectedItem
```

calculateProfit

```
queryStatement ← "SELECT SUM(cost), SUM(price) FROM orders WHERE date≥ selectedMonth"
dataResults ← results.queryStatement
totalCost ← dataResults(1)
totalPrice ← dataResults(2)
totalProfit ← totalPrice – totalCost
totalProfit ← OUTPUT
```

calculateMostSoldProduct

```
queryStatement ← "SELECT productFK, (products.productName WHERE productFK = productPK),
COUNT(productFK) AS MOST_FREQUENT FROM orders WHERE date≥ selecteMonth"
dataResults ← results.queryStatement
mostSoldProduct ← dataResults(2)
```

```
mostSoldProduct ← OUTPUT
```

CalculateNumberOfOrders

```
queryStatement ← "SELECT MAX(orderPK) FROM orders"
dataResults ← results.queryStatement
totalOrderNum ← dataResults(1)
queryStatement ← "SELECT BOTTOM(orderPK) FROM order WHERE date >= selectedMonth"
dataResults ← results.queryStatement
firstOrderOfMonth ← dataResults(1)
MonthlyOrders ← totalOrderNum – firstOrderOfMonth
MonthlyOrders ← OUTPUT
```

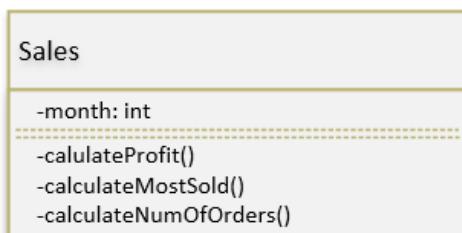
3.5.6.5 Test Plan

Test Number	How to Test	Expected Result
1	Create and confirm multiple orders	Correct sales analytics generated

3.5.6.6 Evidence Plan

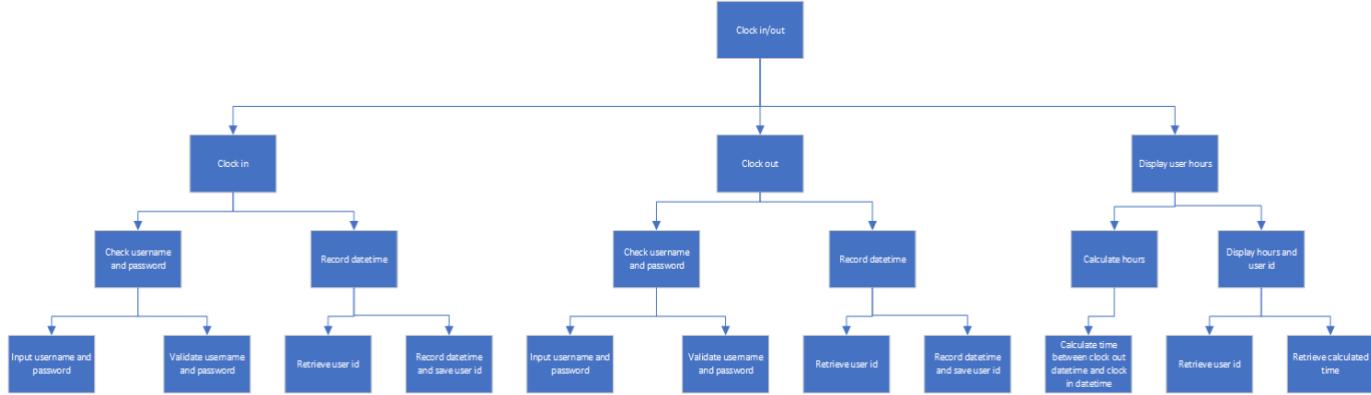
- Screenshot: generated sales analytics.
- Screen Record: new sales analytics being generated as a new order is confirmed.
- Screenshot: sales analytics displayed.

3.5.6.7 Class Diagram



3.5.7 Stage 6/Clock in/out

3.5.7.1 Structure Diagram



3.5.7.2 Success Criteria Satisfied

Number	Description
9	Clock in/out function + calculate hours

3.5.7.3 Plan

- Use cost and price of orders to calculate profit
- Use jTable to display user hours

3.5.7.4 Algorithms

ClockIn/Out – Pseudocode

```
String username ← INPUT https://www.ocr.org.uk/Images/514655-programming-project-set-a-high.pdf
```

```
String password ← INPUT
```

```
IF (username.length() < 6 OR password.length() < 6) THEN
  ("Invalid, username and password must be over 6 characters long") ← OUTPUT
```

```
ELSE THEN
```

```
  queryStatement ← "SELECT username, password, userpk FROM users WHERE 'username' = username AND 'password' = password"
```

```
  dataResults ← results.executeQueryStatement
```

```
  IF (dataResults == TRUE) THEN
```

```
    queryStatement ← "SELECT clockedin FROM userhours WHERE userFK = currentUserPK"
```

```
    dataResults ← results.executeQueryStatement
```

```
    clockedIn ← dataResults(1)
```

Design

```
IF (clockedIn == FALSE) THEN
    queryStatement ← "INSERT INTO dailyhours
                      clockintime = currentDateTime() WHERE userFK = currentUserPK"
ELSE IF (clockedIn == TRUE) THEN
    queryStatement ← "UPDATE dailyhours SET
                      clockouttime = currentDateTime() WHERE userFK = currentUserPK"
END IF
ELSE THEN
    ("Username or Password is incorrect") ← OUTPUT
END IF
END IF
```

CalculateHours

```
queryStatement ← "SELECT TIMEDIFFERENCEBETWEEN(clockintime, clockouttime) FROM userhours
                  WHERE userFK = currentUserPK"
dataResults ← results.queryStatement
totalShiftHours ← dataResults(1)
queryStatement ← "UPDATE userhours SET monthlyhours = monthlyhours+totalShiftHours WHERE
                  userFK = currentUserPK"
```

FillArrayWithRecordsFromTable

DisplayDatabaseTableInJtable

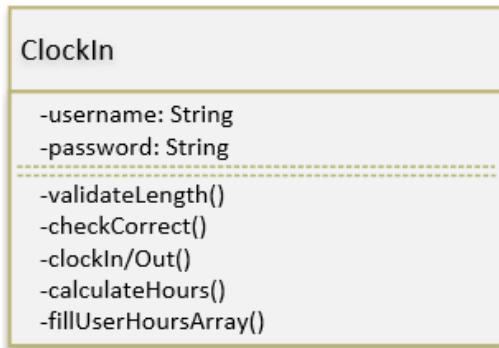
3.5.7.5 Test Plan

Test Number	How to Test	Expected Result
1	Enter username and password and clock in	Clock in time should be saved in the clock in table
2	Enter username and password and clock out	Clock out time should be saved in the clock in table
3	Clock in and out	Correct hours should be calculated between clock in and out time

3.5.7.6 Evidence Plan

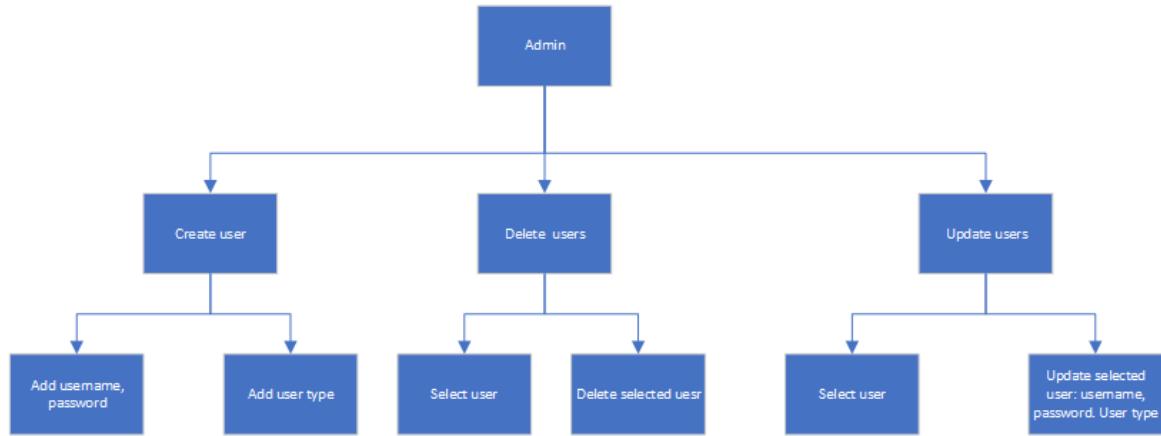
- Screen Record: entering username and password and clocking in.
- Screen Record: entering username and password and clocking out.
- Screenshots: clock in table showing the clock in/out datetime.
- Screenshots: clock in table showing calculated hours.

3.5.7.7 Class Diagram



3.5.8 Stage 7/Admin

3.5.8.1 Structure Diagram



3.5.8.2 Plan

- Use a jTable to display users

3.5.8.3 Algorithms

AddAUser – pseudocode

```
queryStatement ← “INSERT INTO users userName = nameInput,
userType = typeInput, userPassword = passwordInput”
```

DeleteAUser – pseudocode

```
queryStatement ← “DELETE FROM users WHERE userPK = selectedUserPK”
```

UpdateAUser – pseudocode

```
queryStatement ← “UPDATE products SET users.userName = nameInput, userType = typeInput,
userPassword = passwordInput WHERE userPK = selectedUserPK”
```

FillArrayWithRecordsFromTable

DisplayDatabaseTableInJtable

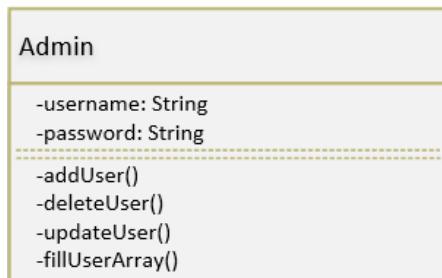
3.5.8.4 Test Plan

Test Number	How to Test	Expected Result
1	Press add user button	Add user screen should pop up
2	Enter user details and confirm	New user should be created and displayed
3	Select a user and press update button	Update user screen should pop up
4	Enter details and confirm	Updated user details should be displayed
5	Select a user and press delete button	Product should be deleted

3.5.8.5 Evidence Plan

- Screen Record: pressing add user button and screen popping up to allow user to add a user.
- Screen Record: entering inputs and adding the user.
- Screenshot: displayed users.
- Screen Record: selecting a user and pressing update button.
- Screen Record: entering inputs and updating user.
- Screenshot: updated users.
- Screen Record: selecting a user and deleting them.
- Screenshot: table to show user is deleted.

3.5.8.6 Class Diagram



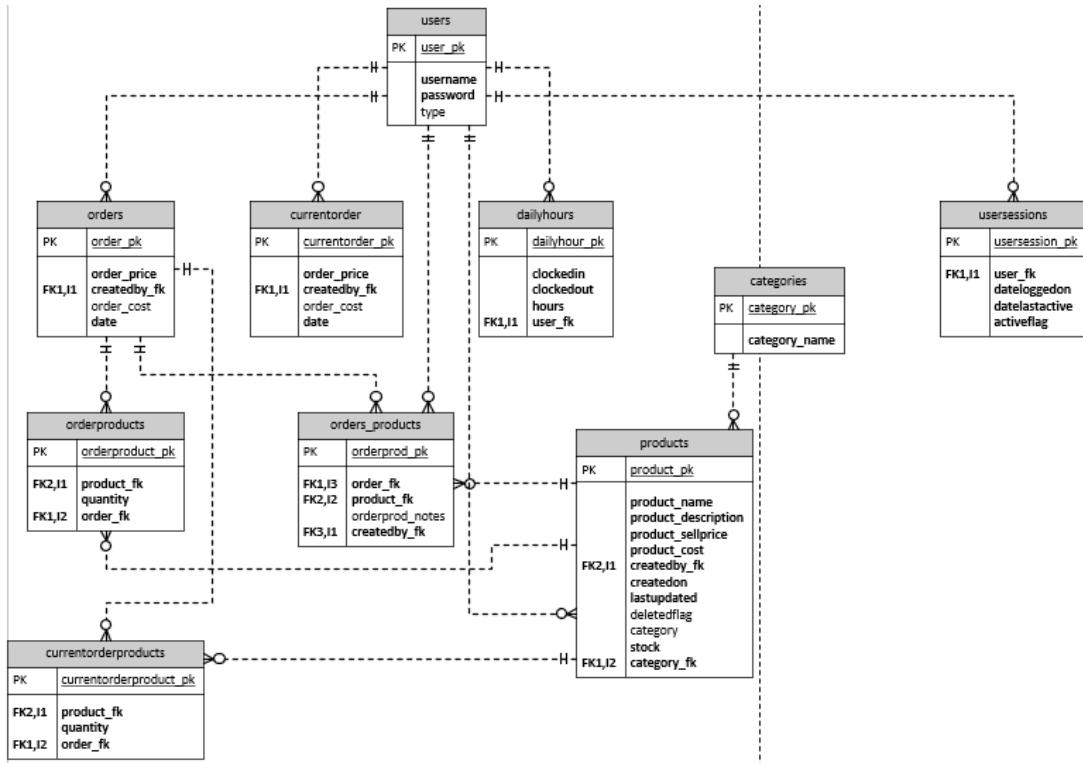
3.6 Database

3.6.1 Table List

Table	Function
users	This table contains the user information for each user such as their username, password, user type and their primary key (userID). It would be used for logging in, displaying correct menu
userssessions	This table contains the date and time a user first logs in and when they are next active. This would be used for timing out. This table would be linked to the "users" table
orders	This table contains all the orders and contains fields such as; price, date/time, products sold etc. This would be used for the orders section
products	This table contains all the products and information about them such as name, cost, sell price, stock etc. This would be used for the products section
salesreports	This table contains all the sales reports information created using the orders table such as avgprice, numorders etc. This would be used for the sale reports section
dailyhours	This table contains the time the user clocked in and out and calculates their hours. The users hours would be inserted into the "userhours" table which would be updated after each shift. This would be used in the clock in/out function
userhours	This table contains the users hours worked over a select time period. This would be used to calculate the users monthly hours. This table would be linked to the "users" table
currentorder	This table will contain the date and price etc of the current order which once confirmed will be sent to the orders table and the current order table will be wiped
currentOrderProducts	This table will contain the products and quantity in the current order and once confirmed the records will be inserted into the orderProducts table and this table will be wiped.
categories	This table will contain the different categories the user creates
orderProducts	This table will contain the products and quantity in an order and will use the order primary key as a foreign key to show that these products belong to this order.

Design

3.6.2 Entity Relationship Diagram



3.6.3 Table design

3.6.3.1 users

userPK	username	password	type
1	TheoPlank	HelloWorld	Admin

3.6.3.2 usersessions

usersessionPK	userFK	dateloggedon	datelastactive	activeflag
10	1	2022-09-01 7:38:14	2022-09-01 7:40:37	0

3.6.3.3 products

productPK	name	description	price	cost	createdbyFK	createdon	lastupdated	deletedflag	category	stock	categoryFK
17	milk	Cow milk	4.50	2.10	1	2022-09-01 7:38:14	2022-09-01 7:40:37	0	liquid	5	2

3.6.3.4 categories

categoryPK	name
2	dairy

3.6.3.5 Orders/currentOrder

orderPK	price	cost	date	userFK
---------	-------	------	------	--------

Design

1	9.00	4.20	2022-09-22 14:27:13	1
---	------	------	------------------------	---

3.6.3.6 *orderProducts/currentOrderProducts*

orderProductPK	productPK	quantity	orderPK
1	17	2	1

3.6.3.7 *dailyHours*

dailyHourPK	clockedIN	clockedOUT	hours	userFK
1	2022-08-03 17:00:00	2022-08-03 22:47:31	5.792	1

3.6.3.8 *salesReports*

reportPK	Date	profit	numOrders	avgPrice
1	2022-07-14 9:37:12	3267.83	312	54.62

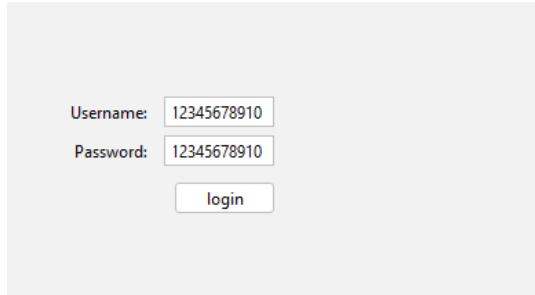
3.6.3.9 *userHours*

userHourPK	month	hours	userFK
1	july	55.38	1

4 Development and Testing

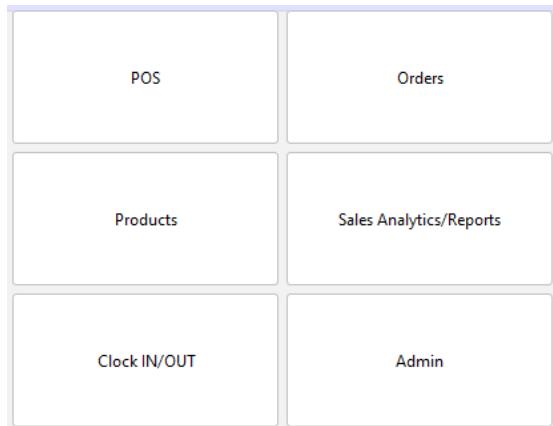
4.1 Stage 0/GUI development

4.1.1 Login



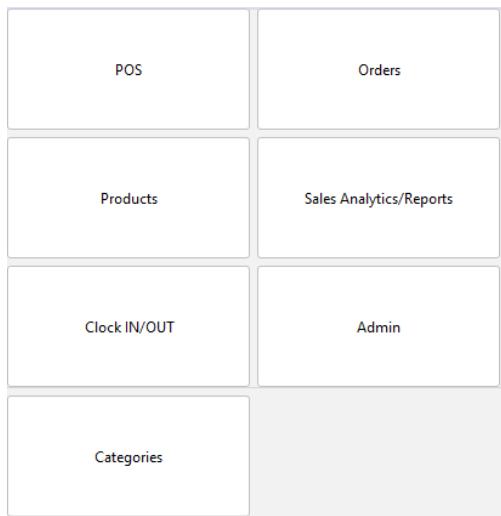
The login screen features a light gray background. It contains three input fields: 'Username' with the value '12345678910', 'Password' with the value '12345678910', and a 'login' button.

4.1.2 Menu



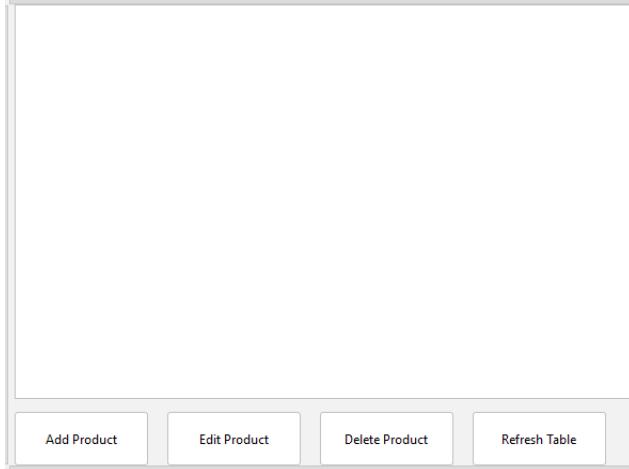
4.1.2.1 *Updates*

I updated the menu screen when coding the products section as I decided to add a categories section to the menu instead of putting it in the products section. The new menu GUI is below.

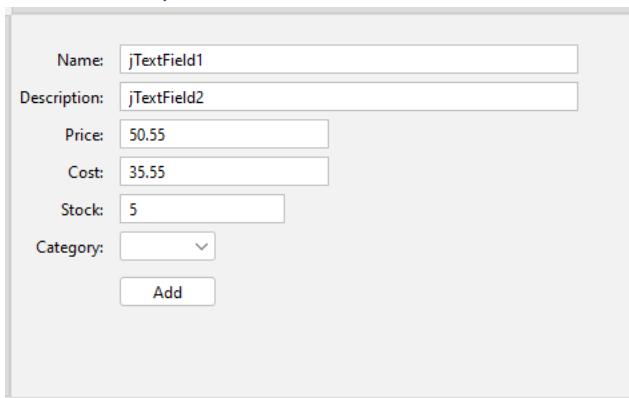


4.1.3 Products

4.1.3.1 Base Screen

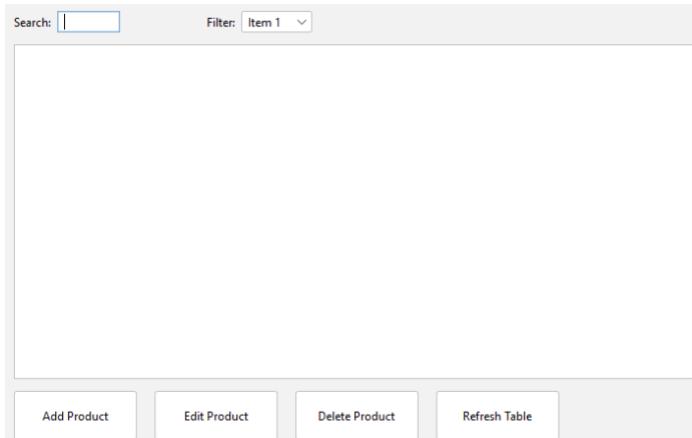


4.1.3.2 Add/Edit Product Screen



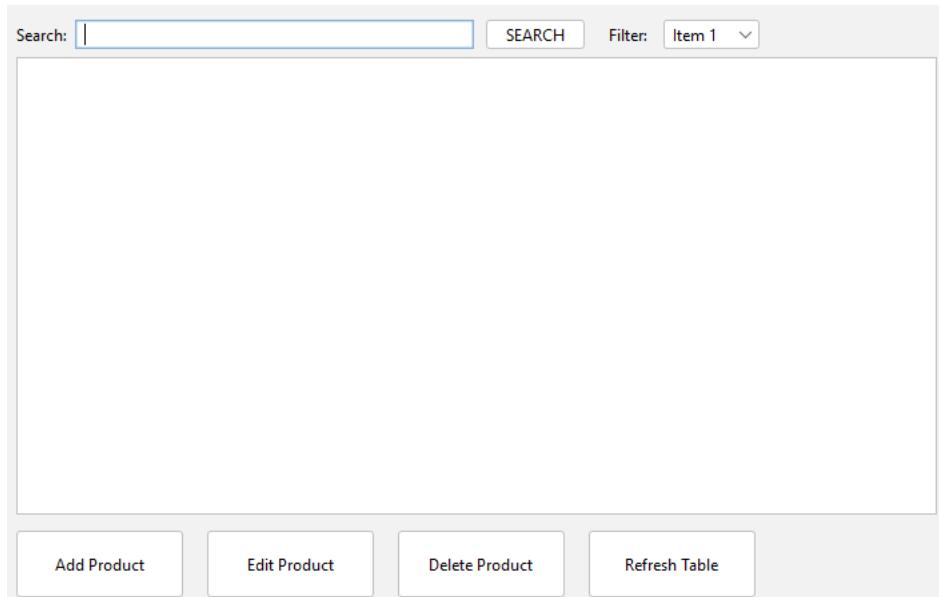
4.1.3.3 Updates

I updated the products screen as I decided to add a search function and a filter function as shown below.



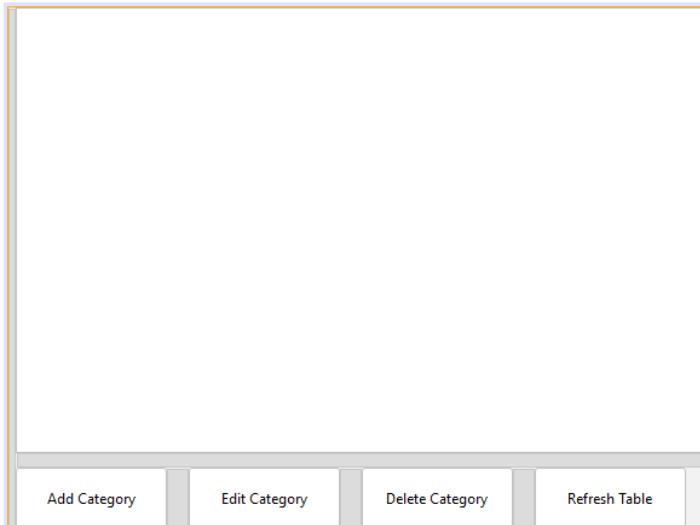
4.1.3.4 Updates

I added a search button to trigger the search as shown below.

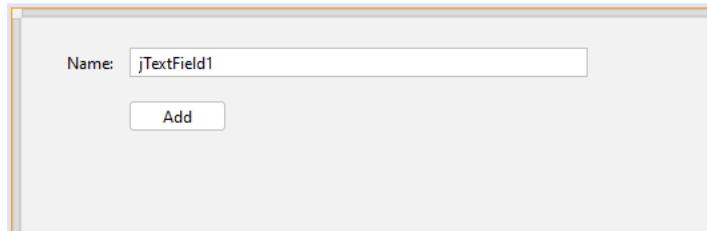


4.1.4 Categories

4.1.4.1 Base Screen



4.1.4.2 Add/Edit Category Screen



Development and Testing

4.1.5 POS

The screenshot shows a POS application window. On the left, a large panel titled "PRODUCTS" contains a blank list area. On the right, a panel titled "ORDER" also contains a blank list area. Below the "PRODUCTS" panel is a small button labeled "+". Under the "ORDER" panel is a button labeled "X". To the right of the "X" button are four text fields: "Total:", "Discount:", and "Final Total:", each with a corresponding input field. Below these fields is a "Confirm" button.

4.1.6 Orders

The screenshot shows an "Orders" interface. At the top, there is a search bar with placeholder text "Search:" and a dropdown filter menu set to "Item 1". Below this is a large empty table area. At the bottom, there are two buttons: "Delete Order" and "Refresh Table".

4.1.6.1 *Updates*

I added a search button to trigger the search as shown below.

The screenshot shows the same "Orders" interface as before, but with a key addition: a "SEARCH" button has been placed next to the search bar. The rest of the interface remains the same, including the table area and the "Delete Order" and "Refresh Table" buttons at the bottom.

4.1.7 Sales

Month:

Total Revenue:

Profit:

Average Sale:

Number of Sales:

4.1.8 Clock In

4.1.8.1 Base Screen

Username:

Password:

4.1.8.2 Clock in/out Choice Screen

4.1.9 Admin

4.1.9.1 Users Screen

Search:

Add User	Edit User	Delete User	Refresh Table
----------	-----------	-------------	---------------

4.1.9.2 Add/Edit User Screen

The screenshot shows a user interface for adding or editing a user. It consists of three text input fields and one dropdown field, all contained within a light gray rectangular box.

- Name:** A text input field containing the placeholder text "jTextField1".
- Password:** A text input field containing the placeholder text "jTextField1".
- Type:** A dropdown menu currently set to "Admin".

Below these fields is a single "Add" button.

4.2 Stage 1/Database

4.2.1 Table SQL Create Statements

4.2.1.1 users Table

```

1 • CREATE TABLE `users` (
2     `user_pk` int NOT NULL AUTO_INCREMENT,
3     `username` varchar(50) NOT NULL,
4     `password` varchar(50) NOT NULL,
5     PRIMARY KEY (`user_pk`)
6 )

```

4.2.2 userssessions Table

```

1 • CREATE TABLE `userssessions` (
2     `usersession_pk` int NOT NULL AUTO_INCREMENT,
3     `user_fk` int NOT NULL,
4     `dateloggedon` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
5     `datelastactive` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
6     `activeflag` tinyint(1) NOT NULL DEFAULT '1',
7     PRIMARY KEY (`usersession_pk`),
8     KEY `usersessions_fk_1` (`user_fk`),
9     CONSTRAINT `userssessions_fk_1` FOREIGN KEY (`user_fk`) REFERENCES `users` (`user_pk`)
10 )

```

4.2.3 products Table

```

CREATE TABLE `products` (
    `product_pk` int NOT NULL AUTO_INCREMENT,
    `product_name` varchar(100) NOT NULL,
    `product_description` varchar(255) NOT NULL DEFAULT 'description',
    `product_sellprice` decimal(13,2) NOT NULL DEFAULT '0.00',
    `product_cost` decimal(13,2) NOT NULL DEFAULT '0.00',
    `createdby_fk` int NOT NULL,
    `createdon` datetime NOT NULL,
    `lastupdated` datetime NOT NULL,
    `deletedflag` tinyint(1) DEFAULT NULL,
    `category` varchar(100) DEFAULT NULL,
    `stock` int NOT NULL DEFAULT '0',
    PRIMARY KEY (`product_pk`),
    KEY `products_fk_1` (`createdby_fk`),
    CONSTRAINT `products_fk_1` FOREIGN KEY (`createdby_fk`) REFERENCES `users` (`user_pk`),
)

```

4.2.4 orders Table

```

1 • CREATE TABLE `orders` (
2     `order_pk` int NOT NULL AUTO_INCREMENT,
3     `order_price` decimal(13,2) NOT NULL DEFAULT '0.00',
4     `createdby_fk` int NOT NULL,
5     `order_cost` float DEFAULT NULL,
6     `date` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
7     PRIMARY KEY (`order_pk`),
8     KEY `orders_fk_1` (`createdby_fk`),
9     CONSTRAINT `orders_fk_1` FOREIGN KEY (`createdby_fk`) REFERENCES `users` (`user_pk`)
10 )

```

4.2.5 currentorder Table

```

1 • Ⓜ CREATE TABLE currentorder (
2     `currentorder_pk` int NOT NULL AUTO_INCREMENT,
3     `order_price` decimal(13,2) NOT NULL DEFAULT '0.00',
4     `createdby_fk` int NOT NULL,
5     `order_cost` float DEFAULT NULL,
6     `date` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
7     PRIMARY KEY (`currentorder_pk`),
8     KEY `currentorder_fk_1` (`createdby_fk`),
9     CONSTRAINT `currentcurrentordercurrentorderorder_fk_1` FOREIGN KEY (`createdby_fk`) REFERENCES `users` (`user_pk`)
10    )

```

4.2.6 orderproducts Table

```

1 • Ⓜ CREATE table orderproducts (
2     orderproduct_pk int not null auto_increment,
3     product_fk int not null,
4     quantity int not null,
5     order_fk int not null,
6     primary key(orderproduct_pk),
7     CONSTRAINT orderproducts_fk_1 FOREIGN KEY (product_fk) REFERENCES products (product_pk),
8     CONSTRAINT orderproducts_fk_2 FOREIGN KEY (order_fk) REFERENCES orders (order_pk)
9 )

```

4.2.7 currentorderproducts Table

```

1 • Ⓜ CREATE table currentorderproducts (
2     currentorderproduct_pk int not null auto_increment,
3     product_fk int not null,
4     quantity int not null,
5     order_fk int not null,
6     primary key(currentorderproduct_pk),
7     CONSTRAINT currentorderproducts_fk_1 FOREIGN KEY (product_fk) REFERENCES products (product_pk),
8     CONSTRAINT currentorderproducts_fk_2 FOREIGN KEY (order_fk) REFERENCES orders (order_pk)
9 )
10

```

4.2.8 dailyhours Table

```

1 • Ⓜ create table dailyhours (
2     dailyhour_pk int not null auto_increment,
3     clockedin datetime not null default current_timestamp,
4     clockedout datetime not null default current_timestamp on update current_timestamp,
5     hours decimal(2,1) not null default 0.00,
6     user_fk int not null,
7     primary key(dailyhour_pk),
8     constraint dailyhours_fk_1 foreign key (user_fk) references users (user_pk)
9 )

```

4.2.9 salesreports Table

```
1 • Ⓜ create table salesreports(
2     report_pk int not null auto_increment,
3     date datetime not null default current_timestamp,
4     profit decimal(13,2) not null default 0.00,
5     numorders int not null default 0,
6     avgprice decimal(13,2) not null default 0.00,
7     primary key(report_pk)
8 )
```

4.2.10 userhours Table

```
⌚ create table userhours (
    userhour_pk int not null auto_increment,
    month datetime not null default current_timestamp,
    hours decimal (13,2) not null default 0.0,
    user_fk int not null,
    primary key(userhour_pk),
    constraint userhours_fk_1 foreign key (user_fk) references users (user_pk)
)
```

4.3 Stage 2/Login

The code below initialises the username and password variables, retrieves the user input and calculates the length of the username password as this will be used for validation.

```

String userName = usernameTextField.getText();
String password = passwordTextField.getText();
int uLen = userName.length();
int pLen = password.length();
PreparedStatement ps;
ResultSet rs;
int userPK;

try {
    if (uLen < 6 || pLen < 6) {
        JOptionPane.showMessageDialog(null, "invalid, username and password must be 6 or more cha
    }
}

```

This code validates the username and password length.

Video Reference: loginStageLengthCheck

4.3.1 Problem1

4.3.1.1 Broken Code

When communicating with the database in this project I used two main imports: PreparedStatement, ResultSet. When I first tried to send a query to the database I used the code below.

```

ps = prepareStatement("SELECT `username`, `password` , `user_pk` | B|
ps.setString(1, usernameTextField.getText());
ps.setString(2, passwordTextField.getText());
rs = ps.executeQuery();

```

However, I got this error. This is because I haven't specified what database the query should be sent to.

```

Exception in thread "AWT-EventQueue-0" java.lang.RuntimeException: Uncompilable code - cannot find symbol
symbol:   method prepareStatement(java.lang.String)
location: class loginScreen

```

I fixed this by adding the two lines of code below which specify the database name and password and the port

```
Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/posplex", "root", "Forgetit");
```

Development and Testing

This code will have to be used whenever I communicate with the database, so I made a class called database which holds all the database information necessary to access the database and execute a query such as the database name and the username and password used to log in to the database.

```
private static String databaseName = "posplex";
private static String username = "root";
private static String password = "Forgetit";

static Connection connection=null;

public static Connection getConnection()
{
    if (connection != null){
        return connection;
    }
    return getConnection(databaseName, username, password);
}
```

The code above contains key variables for the database class which store the database name, username and password. The method getConnection tries to make a connection with the database if there is not already a connection it calls the getConnection method and takes the database name, username and password as parameters and uses these to attempt to make a connection to the database as shown in the code below. If there is already a connection made it will return the connection.

```
private static Connection getConnection(String databaseName, String userName, String password) //make sure to add imports at top of file
{
    try
    {
        Class.forName("com.mysql.cj.jdbc.Driver");
        connection=DriverManager.getConnection("jdbc:mysql://localhost:3306/" +databaseName+"?useSSL=false&serverTimezone=UTC");
        System.out.println("connected");
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }

    return connection;
}

->?user="+userName+"&password="+password);
```

This getConnection method takes the database name, username and password as parameters and makes a connection to the database using these parameters and the MySQL database URL and port, this method is called in the other getConnection method. This method also has a catch statement to catch any errors and returns the connection.

I used this method when writing the new preparedStatement code as shown below:

4.3.1.2 Fixed Code

```

else {
    ps = classes.database.getConnection().prepareStatement("SELECT `username`, `password`, `user_pk` PK FROM `users` WHERE `username`"
    ps.setString(1, usernameTextField.getText());
    ps.setString(2, passwordTextField.getText());
    rs = ps.executeQuery();
}

WHERE `username` = ? AND `password` = ?");

```

I have used “?” as placeholders in the SQL query and set these placeholders to the username input and password input in the setString lines. Now when ps.executeQuery is done it makes a connection to the database and then executes the query statement and retrieves the results and stores them in the result set “rs”.

Video Reference: loginStageValidateCorrect

4.3.2 Problem2

Initially when coding this project once the login was working, I had the login button create a new frame and I had problems referencing different components. Firstly I couldn’t use this.dispose() to dispose of the login frame once the menu had opened and secondly I couldn’t access certain panels because they were private. To fix these problems firstly I made jforms without a jframe and had the class extend jframe using the code below, this allowed me to dispose of the login frame and this also fixed the panel access problem because I didn’t have to set the jframe content to a panel.

4.3.2.1 Broken Code

```

public class TestApp {

    JFrame menuFrame = new JFrame("Menu");
    menuFrame.setContentPane(new Menu().panel1);
    menuFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    menuFrame.pack();
    menuFrame.setVisible(true);

}
connection.close();
}

```

C:\Users\garmin\IdeaProjects\Test App\src\CustomPackage\TestApp.java:
java: panel1 has private access in CustomPackage.Menu

4.3.2.2 Fixed Code

```
public class loginScreen extends javax.swing.JFrame {
```

```

menuScreen menu = new menuScreen();
//menu.sessionID = currentSession;
menu.setDefaultCloseOperation(menuScreen.EXIT_ON_CLOSE);
menu.pack();
menu.setVisible(true);

this.dispose();

```

Video Reference: loginStageMenuCheck

4.3.3 Developing Session Creation Method code

The code below is the insertSession method, in the userSessions class, which inserts a record into the usersessions table which contains the current user primary key, the login time and the last active time, this method would be called whenever a user logs in.

```

public static int insertUserSession(int userPK){
    //creates a new userSession and returns the latest session primary key

    Connection connection = database.getConnection();
    PreparedStatement ps;
    ResultSet rs;
    int thisSessionPK = 0;

    try{
        ps = connection.prepareStatement("INSERT INTO usersessions(user_fk) VALUES (?)");
        ps.setString(1, Integer.toString(userPK));
        ps.executeUpdate();
        ps = connection.prepareStatement("SELECT MAX(usersession_pk) sessionPK FROM usersessions WHERE user_fk = ?");
        ps.setString(1, Integer.toString(userPK));
        rs = ps.executeQuery();
        rs.next();
        thisSessionPK = rs.getInt("sessionPK");
    }
    catch(SQLException ex){
        Logger.getLogger(userSessions.class.getName()).log(Level.SEVERE, null, ex);
    }
    return thisSessionPK;
}

```

```

if (rs.next()){

    userPK = rs.getInt("PK");
    menuScreen menu = new menuScreen();
    //menu.sessionID = currentSession;
    menu.setDefaultCloseOperation(menuScreen.EXIT_ON_CLOSE);
    menu.pack();
    menu.setVisible(true);
    int currentSession = classes.userSessions.insertUserSession(userPK);
    System.out.println(Integer.toString(currentSession));
    menu.sessionID = currentSession;
    menu.userPrimaryKey = userPK;
    this.dispose();

}

else{
    JOptionPane.showMessageDialog(null, "incorrect username or password");
}

}

catch(SQLException ex){
    Logger.getLogger(loginScreen.class.getName()).log(Level.SEVERE, null, ex);
}

```

The if statement checks if there is anything in the result set, if there is then the query has returned a record with a username and password matching the user inputted username and password.

The primary key for the user that has logged in is then retrieved from the result set.

A new user session is then created taking the primary key of the user who logged in as a parameter to use as a foreign key in the usersessions table for the session record.

The user primary key is then passed into the menu form and the login screen is disposed.

There is also a catch statement to catch any errors on the database side.

4.3.4 Additional Notes

I also have getter and setter methods for all key variables that relate to fields in the database in the users table for example I have getter and setter methods for username and password. Also all of this code is in the loginButton method which has an action listener which listens for when the button is pressed then executes the code.

4.3.5 Test Plan

Test Number	How to Test	Expected Result	Actual Result
1	Enter username and/or password under 6 characters	Error message	Error message
2	Enter incorrect username and/or password	Error message	Error message
3	Enter correct username and password	Menu screen should open	Menu Screen opened
*	Login and check if usersessions table in	Table should have updated with a login	Login table updated

	the database has updated	datetime and a last active datetime	
--	--------------------------	-------------------------------------	--

4.3.6 Evidence

4.3.6.1 Test 1, Test 2, Test 3

Video Reference: loginStageFinalVid

4.3.6.2 Test *

Video Reference: loginStageSessionCheck

4.3.7 Success Criteria

Success Criteria Number	Success Criteria	Achieved	Notes
12	Login system	Yes	None

4.3.8 Review/Stage Evaluation

In this stage I developed various classes for use in the login system so that only people who are users of the system can log in and use the system. Lots of the code and methods that I implemented in this section will be re-used or similar versions will be written later on in different stages for example; the database class methods will be re-used many times throughout the course of this project. I also developed some functionality behind an additional feature I decided to add. This feature logs out the user if they haven't been active on the main menu within 10 minutes, I added this feature to ensure the security of the system so that a non-employee/non-user cannot use the system if an employee has left their system out in the open still logged on. So far, I have simply created the user sessions class and a few key methods. The methods I have developed create a new record in the usersessions table in the database to record what time the user logged on at. In the next stage I will have to create the method to update the last active time in the usersessions table whenever the user clicks a main menu button, this method will also calculate the time in minutes between when the user was last active and the current time and log out the user and display a message if it has been over 10 minutes.

In this stage I was successful at achieving all the intended success criteria which is a good sign although it was a relatively broad success criterion.

4.4 Stage 3/Menu

I had already created the menu GUI before coding for the menu so firstly I added action listeners to all the buttons on the menu.

4.4.1 Developing Session Validation Method Code

I also had to create a method that checks if a session is valid which means that it checks if it has been over 10 minutes since the user was last active. If it has been over 10 minutes the user is logged out and must log in again to use the system and the activeflag for the record is set to false. if it hasn't been over 10 minutes the lastactive field is updated to the current date and time. This method would be called whenever the user presses a button on the menu. The code is below:

```
public static boolean isSessionValid(int thisSessionPK){
    //checks how long it has been since the user was last active
    //if it has been over 10 minutes then the session is invalid so user is logged out and has to log back in
    //if it hasn't been over 10 minutes then the last active field in the userSessions table in the database is updated with the current time

    Connection connection = database.getConnection();
    PreparedStatement ps;
    ResultSet rs;
    int timeDiff = 0;
    boolean valid = false;

    try{
        ps = connection.prepareStatement("SELECT TIMESTAMPDIFF(minute, datelastactive, sysdate()) diff FROM usersessions WHERE usersession_pk = ?");
        ps.setString(1, Integer.toString(thisSessionPK));
        rs = ps.executeQuery();
        rs.next();
        timeDiff = rs.getInt("diff");
        if (timeDiff > 10){
            valid = false;
            ps = connection.prepareStatement("UPDATE posplex.usersessions SET activeflag = false WHERE usersession_pk = ?");
            ps.setString(1, Integer.toString(thisSessionPK));
            ps.executeUpdate();
        }
        else{
            valid = true;
            ps = connection.prepareStatement("UPDATE posplex.usersessions SET activeflag = true, datelastactive = sysdate() WHERE usersession_pk = ?");
            ps.setString(1, Integer.toString(thisSessionPK));
            ps.executeUpdate();
        }
    }

    catch(SQLException ex){
        Logger.getLogger(userSessions.class.getName()).log(Level.SEVERE, null, ex);
    }
    return valid;
}
```

Firstly this method retrieves the time it has been in minutes since the user was last active using the TIMESTAMPDIFF method set to minutes and set to find the difference between the datelastactive field and the current date which is sysdate(), I set the integer variable timeDiff to diff which is what I named the retrieved time. I then use an if statement to check if it has been over 10 minutes and if it has the valid is set to false and this will be returned and the activeflag field is set to false, however if it hasn't been over 10 minutes then valid is set to true and returned and the activeflag is set to true and the datelastactive is set to sysdate().

```

private void posButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //validates the session
    //disposes of menuScreen and creates new loginScreen if the session has timed out
    //validates the user type

    boolean valid = classes.userSessions.isSessionValid(sessionID);
    System.out.println(sessionID);
    if (valid == false){
        JOptionPane.showMessageDialog(null, "You have been timed out, please log back in.");
        loginScreen login = new loginScreen();
        login.setDefaultCloseOperation(login.EXIT_ON_CLOSE);
        login.pack();
        login.setVisible(true);
        this.dispose();
    }
}

```

Above is the code I used to test the isSessionValid method, I initialised a boolean variable “valid” and set it to the value which will be returned from the isSessionValid method. I also passed in the current sessionID into the method to ensure the validation is performed on the correct record. I used an if statement to check whether valid is true or false, if it was false a message was displayed to the user telling them they have been timed out, at which point a new login screen is initialised and the menu screen is disposed of using the dispose() method.

4.4.2 Developing User Access Code

At this point in the development, I looked at the developing the “User login screens” as recorded in the success criteria which would be the code to give each user different menu screens to ensure they can’t access features the admin hasn’t given them access to. However, I decided a better way to do this would be to have one menu with all the features on it and create code to check the user type and check whether they have access to the feature. If they don’t have access to the feature they are trying to access then the a new screen won’t open. I decided to add this feature with preset user types and preset access for each type as this way is much quicker than allowing the admin to create their own types and access however, I will add this feature if I have time.

In order to do this I had to add a new field to the users table called “type” which held the user type for the user. The SQL to add the new field as well as the new user table create statement code is below.

```

1 • ALTER TABLE users
2   ADD type varchar(20) default null

1 • CREATE TABLE `users` (
2     `user_pk` int NOT NULL AUTO_INCREMENT,
3     `username` varchar(50) NOT NULL,
4     `password` varchar(50) NOT NULL,
5     `type` varchar(20) DEFAULT NULL,
6     PRIMARY KEY (`user_pk`)
7 )

```

Development and Testing

In order to validate the user type first I had to retrieve the user type for the appropriate user being the user for the current user session from the database. To do this I first had to retrieve the primary key for the user in the current session. The initial code to do this is below.

```
public static String getCurrentUserType() {
    //gets the primary key of the user who has the latest usersession
    //gets the user type of the user with the same primary key

    Connection connection = database.getConnection();
    PreparedStatement ps = null;
    ResultSet rs;
    int userpk = 0;
    String userType = null;

    try{
        String query = "SELECT user_fk FK FROM usersessions WHERE usersession_pk = ?";
        ps = connection.prepareStatement(query);
        ps.setInt(1, sessionID);
        rs = ps.executeQuery();
        while (rs.next()){
            userpk = rs.getInt("FK");
        }
    }
}
```

Initially I used the sessionID variable which in other classes holds the primary key for for the current and latest session. I queried the database to select the user foreign key which is the field name from the usersessions table where the usersession primary key is equal to the sessionID. This would return the user primary key for the latest user session however I hadn't passed the sessionID variable into this class so it won't work.

The code to fix this is below.

```
public static String getCurrentUserType() {
    //gets the primary key of the user who has the latest usersession
    //gets the user type of the user with the same primary key

    Connection connection = database.getConnection();
    PreparedStatement ps = null;
    ResultSet rs;
    int userpk = 0;
    String userType = null;

    try{
        String query = "SELECT user_fk FK FROM usersessions WHERE usersession_pk = ( SELECT MAX(usersession_pk) FROM usersessions );";
        ps = connection.prepareStatement(query);
        rs = ps.executeQuery();
        while (rs.next()){
            userpk = rs.getInt("FK");
        }
    }
}
```

The new code selects the user foreign key where the usersession primary key is equal to the biggest usersession primary key in the usersessions table which is the latest and therefore current session. I named the user foreign key FK in the query statement and set the userpk variable, which will hold the user primary key, to the results in the result set named FK.

Next I had to add the code to retrieve the type for the user in the current user session. The code for that is below.

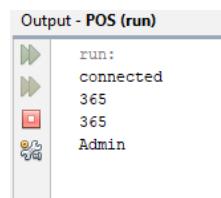
Development and Testing

```
try{
    ps = connection.prepareStatement("SELECT type FROM users WHERE user_pk = ?");
    ps.setInt(1, userpk);
    rs = ps.executeQuery();
    while (rs.next()){
        userType = rs.getString("type");
    }
}
catch(SQLException ex){
    Logger.getLogger(admin.class.getName()).log(Level.SEVERE, null, ex);
}

return userType;
}
```

In this code I query the database to select the type field from the users table where the user primary key is equal to '?', the '?' is a place holder which I set to userpk which is the primary key of the current user which I retrieved in the earlier code. Then I set the String userType to the type retrieved in the result set and return the userType.

I have tested this code by adding a two print lines to print the userType and the sessionID on button press the results are below.



The userType and sessionID are both correct so the method works.

The overall button press method code is below.

```
private void posButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //validates the session
    //disposes of menuScreen and creates new loginScreen if the session has timed out
    //validates the user type

    boolean valid = classes.userSessions.isSessionValid(sessionID);
    System.out.println(sessionID);
    if (valid == false){
        JOptionPane.showMessageDialog(null, "You have been timed out, please log back in.");
        loginScreen login = new loginScreen();
        login.setDefaultCloseOperation(loginScreen.EXIT_ON_CLOSE);
        login.pack();
        login.setVisible(true);
        this.dispose();
    }

    String userType;
    userType = classes.admin.getCurrentUserType();
    if (userType.equals("Admin") || userType.equals("Waiter")){
        System.out.println(userType);
        System.out.println("access granted");
    }
    else{
        System.out.println(userType);
        JOptionPane.showMessageDialog(null, "You don't have access to this feature");
    }
}
```

Now all I have to add is the code to initialise a new screen if valid is true and the user has access, the finalised code is below.

4.4.2.1 POS Button

```

private void posButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //validates the session
    //disposes of menuScreen and creates new loginScreen if the session has timed out
    //validates the user type

    boolean valid = classes.userSessions.isSessionValid(sessionID);
    System.out.println(sessionID);
    if (valid == false){
        JOptionPane.showMessageDialog(null, "You have been timed out, please log back in.");
        loginScreen login = new loginScreen();
        login.setDefaultCloseOperation(loginScreen.EXIT_ON_CLOSE);
        login.pack();
        login.setVisible(true);
        this.dispose();
    }

    String userType;
    userType = classes.admin.getCurrentUserType();
    if (userType.equals("Admin") || userType.equals("Waiter")){
        posScreen screen = new posScreen();
        screen.setDefaultCloseOperation(posScreen.EXIT_ON_CLOSE);
        screen.pack();
        screen.setVisible(true);
    }
    else{
        System.out.println(userType);
        JOptionPane.showMessageDialog(null, "You don't have access to this feature");
    }
}

```

Most of this code will be the same for all the buttons except the code to initialise a new screen will be different so each button initialises the appropriate screen, and the user types that have access to the screen will also be different.

4.4.3 Problems

After testing this feature I found two problems. The first problem was that when I closed a JFrame/screen it would close the menu screen and any other screens open as well. The second problem was that after I had timed out and the message was displayed the new screen would still open and then all the screens would close and the login screen would open.

4.4.3.1 Problem 1

Reference: menuProblem1

As you can see when I closed a window, in this case the POS screen, the whole application would stop running.

After some testing I realised this was because I had set the default close operation to EXIT_ON_CLOSE as seen in the code below.

```

posScreen screen = new posScreen();
screen.setDefaultCloseOperation(posScreen.EXIT_ON_CLOSE);

```

Because the default close operation is set to exit on close when I close a screen I also exit the application causing it to stop running but what I want it to do is to close the screen but keep the

application running. To do this I changed the EXIT_ON_CLOSE to DISPOSE_ON_CLOSE this way the screen is disposed of but the application isn't exited. The new code is below.

```
posScreen screen = new posScreen();
screen.setDefaultCloseOperation(posScreen.DISPOSE_ON_CLOSE);
```

This fixed the problem

Video Reference: menuProblem1Fixed

4.4.3.2 Problem 2

By fixing the first problem part of the second problem is fixed. The screen still opens after timing out however it doesn't close the application afterwards.

Video Reference: menuProblem2

In attempt to fix the problem I added an else clause to the if statement for the time out feature as seen below.

```
boolean valid = classes.userSessions.isSessionValid(sessionID);
if (valid == false){
    JOptionPane.showMessageDialog(null, "You have been timed out, please log back in.");
    loginScreen login = new loginScreen();
    login.setDefaultCloseOperation(loginScreen.DISPOSE_ON_CLOSE);
    login.pack();
    login.setVisible(true);
    this.dispose();
}
else if (valid == true){
    String userType;
    userType = classes.admin.getCurrentUserType();
    if (userType.equals("Admin") || userType.equals("Waiter")){
        posScreen screen = new posScreen();
        screen.setDefaultCloseOperation(posScreen.DISPOSE_ON_CLOSE);
        screen.pack();
        screen.setVisible(true);
    }
    else{
        JOptionPane.showMessageDialog(null, "You don't have access to this feature");
    }
}
```

This fixed the problem

Video Reference: menuProblem2Fixed

After this I decided to change the login screen default close operation to EXIT_ON_CLOSE as when the login screen is closed the application should stop running. The code is below

Development and Testing

```
private void posButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //validates the session
    //disposes of menuScreen and creates new loginScreen if the session has timed out
    //validates the user type

    boolean valid = classes.userSessions.isSessionValid(sessionID);
    if (valid == false){
        JOptionPane.showMessageDialog(null, "You have been timed out, please log back in.");
        loginScreen login = new loginScreen();
        login.setDefaultCloseOperation(login.EXIT_ON_CLOSE);
        login.pack();
        login.setVisible(true);
        this.dispose();
    }
    else if (valid == true){
        String userType;
        userType = classes.admin.getCurrentUserType();
        if (userType.equals("Admin") || userType.equals("Waiter")){
            posScreen screen = new posScreen();
            screen.setDefaultCloseOperation(posScreen.DISPOSE_ON_CLOSE);
            screen.pack();
            screen.setVisible(true);
        }
        else{
            JOptionPane.showMessageDialog(null, "You don't have access to this feature");
        }
    }
}
```

Once this was done I tested the code and it all worked so I recreated this code under all the other button methods for all the different jframes.

4.4.3.3 Feature Access Table

This table shows which user types have access to each feature.

Feature	User Type Allowed
Login	Admin, Waiter,
Menu	Admin, Waiter,
POS	Admin, Waiter
Products	Admin, Waiter, Kitchen, Warehouse
Orders	Admin, Waiter, Kitchen
Sales	Admin
Categories	Admin, Warehouse
Admin	Admin
Clock In	Admin, Waiter, Kitchen, Warhouse

4.4.4 Test Plan

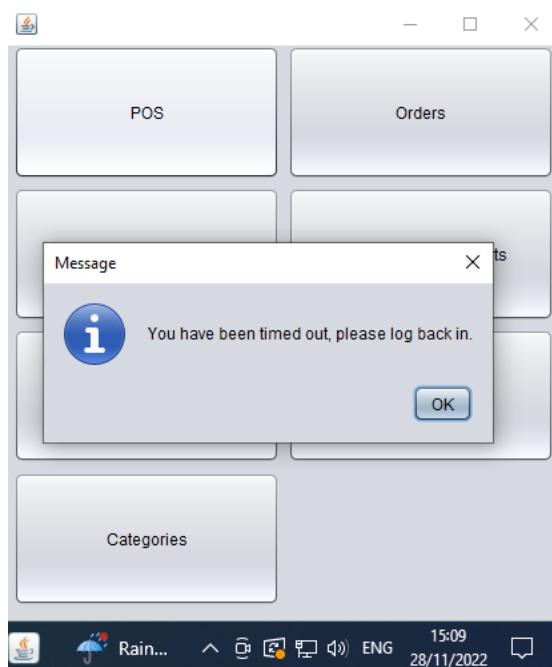
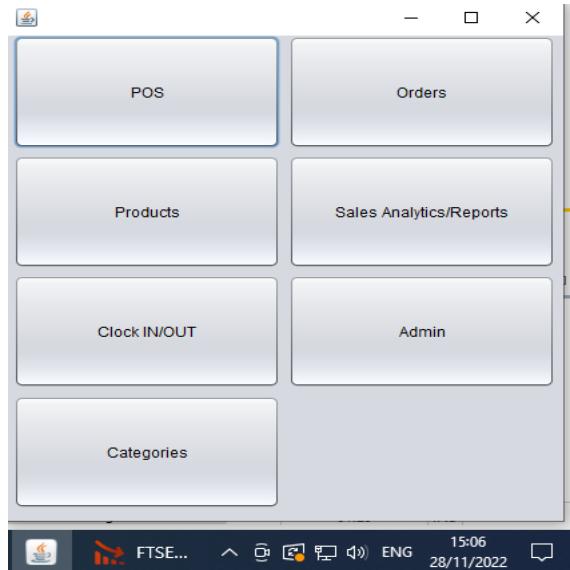
Test Number	How To Test	Expected Result	Actual Result
1	Login as waiter and try to access the admin function	Access denied message	Access denied message
2	Login and don't press anything for 10 minutes	Logged out	Logged out

4.4.5 Evidence

4.4.5.1 Test 1

Video Reference: menuAccessCheck

4.4.5.2 Test 2



Note: for this test I reduced the time out time from 10 minutes to 1 minute.

4.4.6 Success Criteria

Success Criteria Number	Success Criteria	Achieved	Notes
11	User Specific Menu Screens (users will have different user types which determine what functions they have access to)	Yes	Rather than having multiple different menu screen I implemented a feature that checks a user's user type when they try to access specific features and if they are don't have access, a message is displayed and the feature won't open.

4.4.7 Review/Stage Evaluation

In this stage I created the functionality behind the menu screen. This meant developing the methods for timing the user out, previously stated in the login stage evaluation. I also had to develop the method to check whether the logged in user is allowed to access the feature they are trying to access.

The first method I created was the session validation method. This method validates the latest/current session, it checks whether it has been over 10 minutes since the last active datetime, if it has it returns false to show the session isn't valid. If it hasn't been over 10 minutes then the last active datetime field is updated with the current datetime and true is returned. This method was fairly simple and mainly consisted of querying the database for specific data, something which I will be doing a lot in this project so it's good to see it works well at this stage.

The next key change I made in this stage was implementing the user access feature. This meant creating the admin class and developing a method to retrieve the user type for the current user. To do this, I first retrieve the user primary key for the latest session record. Then I query the database to select the user type for the user with that primary key. This type is returned to the user. I have done the selection for allowing a user depending on their type under the action listener for the appropriate button, I have used an if statement to check if the current user is the correct user type and if they aren't an error is displayed, if they are, the feature they are trying to access opens. In this stage I had some minor problems with swing functionality, but they were simple to fix and its better I come across them and learn to fix them now than later in the development of this project.

In this stage I achieved all the intended success criteria however I did so in a slightly different way than I initially planned. Instead of having multiple menu screens and displaying the correct one for the corresponding user, I decided it would be quicker, easier and simpler to just have one menu screen and to validate the users access on attempt to access each feature individually.

4.5 Stage 4/Categories

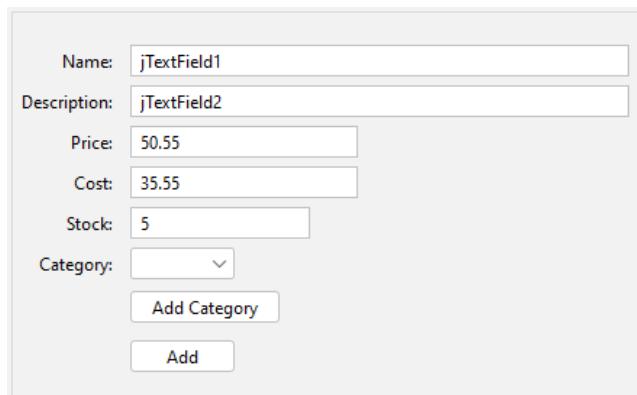
4.5.1 Design Changes

Originally in my design I planned to have the category section as a sort of sub section of the products section so that when a user adds a product, they have the option to add, update and delete categories. However, when I started coding, I decided that this implementation of the categories feature was clunky and inefficient as whenever a user wanted to add a category, they would have to open a add product screen as seen below. Therefore, I decided to make categories into its own section and added a button for the categories section on the menu screen as seen below.

4.5.1.1 Initial Code

```
private void AddCategoryButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    categoryScreen screen = new categoryScreen();
    screen.setDefaultCloseOperation(productScreen.DISPOSE_ON_CLOSE);
    screen.pack();
    screen.setVisible(true);
}
```

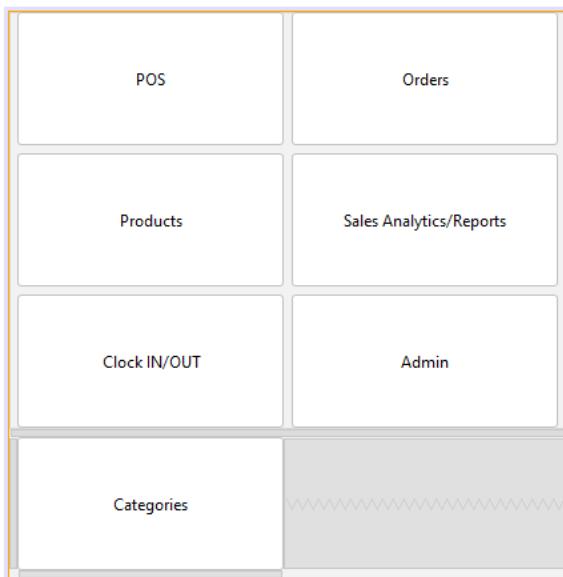
This code was in the addProduct class on the add category button press as shown below.



4.5.1.2 New Code

```
System.out.println(userType);
System.out.println("access granted");
categoryScreen screen = new categoryScreen();
screen.setDefaultCloseOperation(productScreen.DISPOSE_ON_CLOSE);
screen.pack();
screen.setVisible(true);
```

This code is in the menu class under the categories button set to open on button press as shown below. I had to add the categories button to the menu GUI after deciding to separate the category function from the products function.



4.5.2 Database Design Changes

By making the change to have categories be its own separate button and feature I decided to make a categories table to store the categories.

```

1 • CREATE TABLE `categories` (
2   `category_pk` int NOT NULL AUTO_INCREMENT,
3   `category_name` varchar(100) NOT NULL,
4   PRIMARY KEY (`category_pk`)
5 )

```

I also had to update the products table using the alter query below.

```

1 • ALTER TABLE products
2   ADD CONSTRAINT products_fk_2
3   FOREIGN KEY (category_fk) REFERENCES categories(category_pk)

```

4.5.3 Developing Table Model Code

The categories section is the first class I coded that uses a jitable. So I created a table class which handles implementation for most of the methods in the TableModel interface and also deals with the listeners for the table. I did this using the abstract class “AbstractTableModel”. The code is below.

```

import javax.swing.table.AbstractTableModel;
/*
 *
 * @author garmin
 */
public class table extends AbstractTableModel {

    private String[] columns;
    private Object[][] rows;

    public table() {}

    public table(Object[][] data, String[] columnName) {
        this.columns = columnName;
        this.rows = data;
    }

    @Override
    public int getRowCount() {
        return this.rows.length;
    }

    @Override
    public int getColumnCount() {
        return this.columns.length;
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
        return this.rows[rowIndex][columnIndex];
    }

    public String getColName(int column) {
        return this.columns[column];
    }
}

```

The `getRow`, `getColumn` and `getValue` methods are all implementations of methods from the `TableModel`. The `getColName` method is one that I implemented myself and simply returns the column name for the specific name. The `public table` method creates the table taking a 2d array for the data in each row and a 1d array for the column names. The `Object[][]` data will be initialised as a new object 2d array taking the size of a `categories` array, which will be created later and will hold all the categories from the `categories` table in an array, as a value and will also take the number of columns as the second value so that when the table is created it is created with the correct number of rows and columns.

4.5.4 Developing Category Instance Code

I developed a class constructor for `categories` which takes a primary key and a name as parameters. This meant that when the `category` class was called an object of `category` with a given name would be instantiated with a primary key and a category name.

```
public categories(int pk, String name)
{
    this.category_pk = pk;
    this.category_name = name;
}
```

This constructor takes a primary key variable and a name variable as parameters and sets the private variables: category_pk and category_name to these parameters.

4.5.5 Developing Add Category Method

I also had to develop code to allow the user to create a new category, and insert this category into the category table in the database. The code is below.

```
public static void addCategory(categories category){
    //add a new category to the category table in the database using user inputs

    Connection connect = database.getConnection();
    PreparedStatement ps = null;
    try{
        ps = connect.prepareStatement("INSERT INTO categories(category_name) VALUES (?)");
        ps.setString(1, category.getName());

        if(ps.executeUpdate() != 0){
            JOptionPane.showMessageDialog(null, "New Category Added");
        }
        else{
            JOptionPane.showMessageDialog(null, "Error");
        }
    }
    catch (SQLException ex) {
        Logger.getLogger(categories.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

This method takes an instance of the categories class called category as a parameter as this will be added to the category array list.

First this method makes the database connection, then it inserts a new record into the categories table and sets the category_name field for this record to the placeholder ?.

Then it sets the placeholder ? to the value of the getter method for the name for the object category.

4.5.6 Add Category Button Code

Next I had to add the code to create the add category screen on button press for the add category button.

```
private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //creates new newCategoryScreen

    newCategoryScreen categoryAdd = new newCategoryScreen();
    categoryAdd.setDefaultCloseOperation(newCategoryScreen.DISPOSE_ON_CLOSE);
    categoryAdd.pack();
    categoryAdd.setVisible(true);
```

This code calls the jform newCategoryScreen which is the screen that allow the user to input the category name and then add the category.

Then I had to develop the code to retrieve the user inputs from the screen and sets the parameters of the object Category to the user inputs.

```
private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //creates new category and names it with the text in the text field

    classes.categories category;

    String category_name = nameTextField.getText();

    category = new classes.categories(1, category_name);

    classes.categories.addCategory(category);
    this.dispose();
}
```

This code sets the string variable category_name to the value of the getText() method called on the nameTextField which is where the user enters the name for the new category.

Then it sets the object category to a new instance of the categories class taking 1 and category_name (user inputted value) as parameters and add the new object with the user inputted name to the category table by calling the addCategory method and taking the object as a parameter.

The user doesn't input a category primary key as this is done automatically by the database when the new category record is created.

4.5.7 Developing Categories Table Data Retrieval Code

```
public ArrayList<categories> categoryList (){
    //creates an arraylist
    //creates a new instance of "categories" (as shown in method earlier) with data from the categories table in the database
    //adds the new category to the arraylist

    ArrayList<categories> catList = new ArrayList<>();

    connection = database.getConnection();
    PreparedStatement ps;
    ResultSet rs;

    String query = "SELECT * FROM categories";

    try{
        ps = connection.prepareStatement(query);
        rs = ps.executeQuery();
        categories category;
        while (rs.next()){
            category = new categories(rs.getInt("category_pk"), rs.getString("category_name"));

            catList.add(category);
        }
    }
    catch (SQLException ex){
        Logger.getLogger(categories.class.getName()).log(Level.SEVERE, null, ex);
    }
    return catList;
}
```

The above code is the code I use to retrieve the categories from the database and create an ArrayList that holds objects of the categories class and fill this array list with the data from the database.

The first line creates a new ArrayList that holds objects of the categories class called catList.

The query statement selects all the data from the categories table.

The while loop loops through the result set until there is no more data.

An object called category is created as the categories class is called taking the retrieved category_pk and category_name as values.

The object category is added to catList.

NOTE: this code will be re-used for all the sections that use a jTable in my code however it will differ slightly between classes.

I tested this using a print line to print catList and the result is below:

```
[classes.categories@7e9a790a, classes.categories@556c5713, classes.categories@4b8ab27e, classes.categories@2c3a112f]
```

This means the code works as it was able to print out catList as seen which shows catList was filled with data from the database.

4.5.8 Developing Populate Table Code

This code is used to fill the jTable with the objects from the arrayList.

```
public void fillTable() {
    //fill the jTable with contents of the ArrayList

    ArrayList <classes.categories> catList = category.categoryList();
    String [] columnNames = {"ID", "Name"};
    Object [][] rows = new Object[catList.size()][2];

    for(int i = 0; i<catList.size(); i++){
        rows[i][0] = catList.get(i).getCatPK();
        rows[i][1] = catList.get(i).getName();
    }

    classes.table model = new classes.table(rows, columnNames);
    categoryTable.setModel(model);
}
```

NOTE: This code will be re-used for all the sections that use a jTable however it will differ slightly between classes.

The first line calls the categoryList method of the category class, this method returns the category array list, catList, filled with the categories from the database.

The 1d string array columnNames is then created as seen and holds the names for the two columns, "ID" and "Name".

The 2d Object array rows is then created using the size of the catList ArrayList and 2 as the size values for the rows, this ensures that there are enough rows for every category and there are two columns for each row. This array will hold the category data.

Then I used a for loop to loop through from 0 until the end of catList setting the space in position (i,0) to the primary key in catList in position i and then setting the space in position (i,1) to the name in catlist in position i. Now the row in position i in the array holds the primary key and name for a category.

Next I create model an object of the table class by calling the table class constructor and having it take the values rows and columnNames. This creates the table with all the data and the column names.

Then I set the category Jtable model to the object model.

This code worked, as seen below:

ID	Name
19	JTextField1
20	test
22	searchCategoryTest
23	JTextField1

4.5.9 Developing Update Category Code

Next I had to develop the code to allow the user to update categories. This code will be very similar to the add category code. The code is below:

```

public static void updateCategory(categories category) {
    //updates selected category using inputs from the user

    Connection connection = database.getConnection();
    PreparedStatement ps;

    try{
        ps = connection.prepareStatement("UPDATE categories SET category_name = ? WHERE category_pk = ?");
        ps.setString(1, category.getName());
        ps.setInt(2, category.getCatPK());

        if(ps.executeUpdate() != 0){
            JOptionPane.showMessageDialog(null, "Category updated.");
        }
        else{
            JOptionPane.showMessageDialog(null, "There was an error, please try again.");
        }
    }
    catch(SQLException ex){
        Logger.getLogger(categories.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

This method takes an object category of the categories class as a parameter so that it updates the correct category.

The prepare statement line, once executed, sends a update query to the database and sets the category name to the placeholder ? where the category_pk is equal to the placeholder ?.

The first placeholder is set to getter method for the name for the category so that the updated category takes the user inputted name.

The second placeholder is set to the getter method for the primary key so that the correct category is updated.

Then the ps.executeUpdate() method is called which executes the query and returns the number of rows affected by the query, if it returns anything but 0 then at least one row was affected so a successful message is displayed as at least one category was updated. If it returns 0 then no rows were affected and so an error message is displayed.

4.5.10 Update Button Code

I also had to call the update category method under the update button code once pressed.

```
private void editButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //creates new updateCategoryScreen
    //sets records to updated values from the updateCategoryScreen

    try{
        updateCategoryScreen categoryUpdate = new updateCategoryScreen();
        int rowIndex = categoryTable.getSelectedRow();

        categoryUpdate.category_pk = Integer.valueOf(categoryTable.getValueAt(rowIndex, 0).toString());
        categoryUpdate.nameTextField.setText(categoryTable.getValueAt(rowIndex, 1).toString());

        categoryUpdate.setDefaultCloseOperation(updateProductScreen.DISPOSE_ON_CLOSE);
        categoryUpdate.pack();
        categoryUpdate.setVisible(true);
    }
    catch (Exception ex){
        JOptionPane.showMessageDialog(null, "There was an error, please select a row and try again.");
    }
}
```

I added an action listener same as all the other button methods which listens for any activity such as a mouse click on the button and calls the method when that occurs.

First this button creates a new updateCategoryScreen, which is a jframe, called categoryUpdate.

A integer variable rowIndex is set to the returned value of the getSelectedRow() method in the categoryTable. This sets the variable to the index of the selected row.

Next I set the category_pk variable, which is in the updateCateogryScreen class, to the category primary key of the selected row. This is done by using the getValueAt method taking the rowIndex and 0 as parameters so the value returned is the value at the correct row index in the column at position 0 which is the primary key column. This is then converted to a string and then set to an integer.

The next line sets the text in the nameTextField, which is the box the user uses to change the name of the category, to the current name of the selected category. Again using the getValueAt method and the rowIndex variable.

4.5.11 Update Category Screen Code

This code sets the category object to the new user inputted category name.

```
private void updateButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //updates record with text field input from user

    classes.categories category;

    String category_name = nameTextField.getText();
    category = new classes.categories(category_pk, category_name);
    classes.categories.updateCategory(category);

}
```

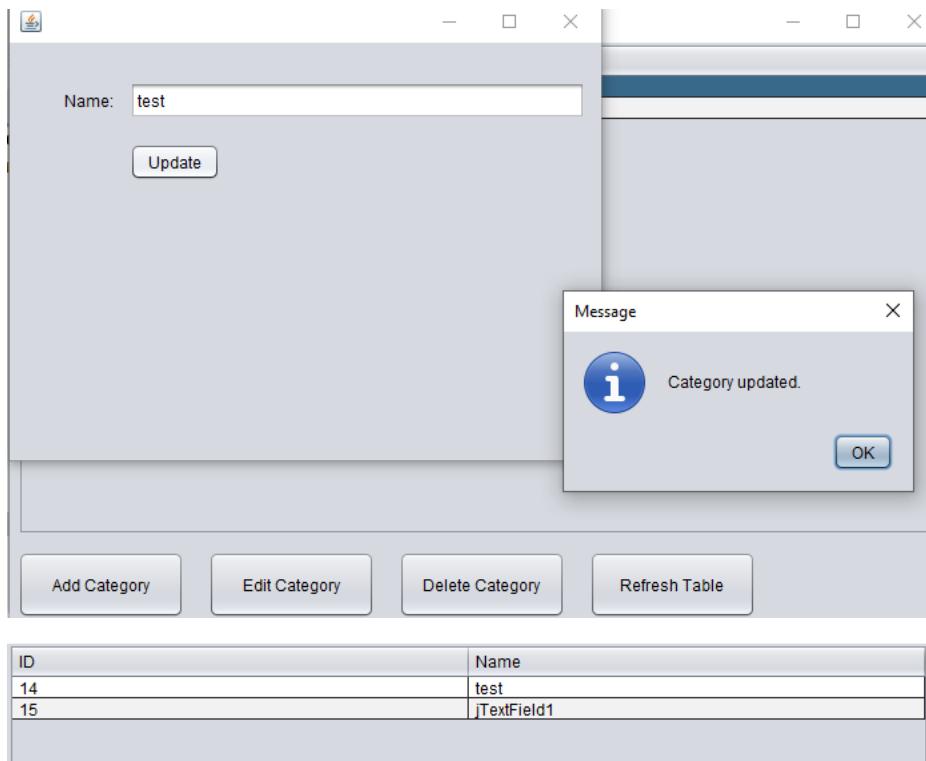
Development and Testing

The string variable category_name is set to the returned value of the getText() method for the nameTextField.

Next the object category is set to a new instance of the categories class by calling the constructor and passing in the values of the category primary key and the category name variable.

Then the updateCategory() method is called and the new object, category, is passed into this method.

This code worked as seen below.



4.5.12 Developing Delete Category Code

Next I had to develop the code to allow the user to delete categories.

```
public static void deleteCategory(int catID){  
    //deletes selected category  
  
    Connection connection = database.getConnection();  
    PreparedStatement ps;  
  
    try{  
        ps = connection.prepareStatement("DELETE FROM categories WHERE category_pk = ?;"); //AND ? = 0  
        ps.setInt(1, catID);  
        //ps.setInt(2, deletedflag);  
        if (ps.executeUpdate() != 0){  
            JOptionPane.showMessageDialog(null, "Category has been deleted.");  
        }  
        else{  
            JOptionPane.showMessageDialog(null, "There was an error, please try again.");  
        }  
    }  
    catch(SQLException ex){  
        Logger.getLogger(categories.class.getName()).log(Level.SEVERE, null, ex);  
    }  
}
```

Development and Testing

This code is quite simple, it takes the category id to identify the correct category to delete then issues a delete query.

The query is a simple delete statement where the category primary key is equal to the placeholder ?.

The placeholder is set to the category id which is passed into the method as a parameter.

Then the query is executed.

4.5.13 Delete Button Code

Next I had to add the code to the button so that when it was pressed, the delete category method was called.

```
private void deleteButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //deletes selected record

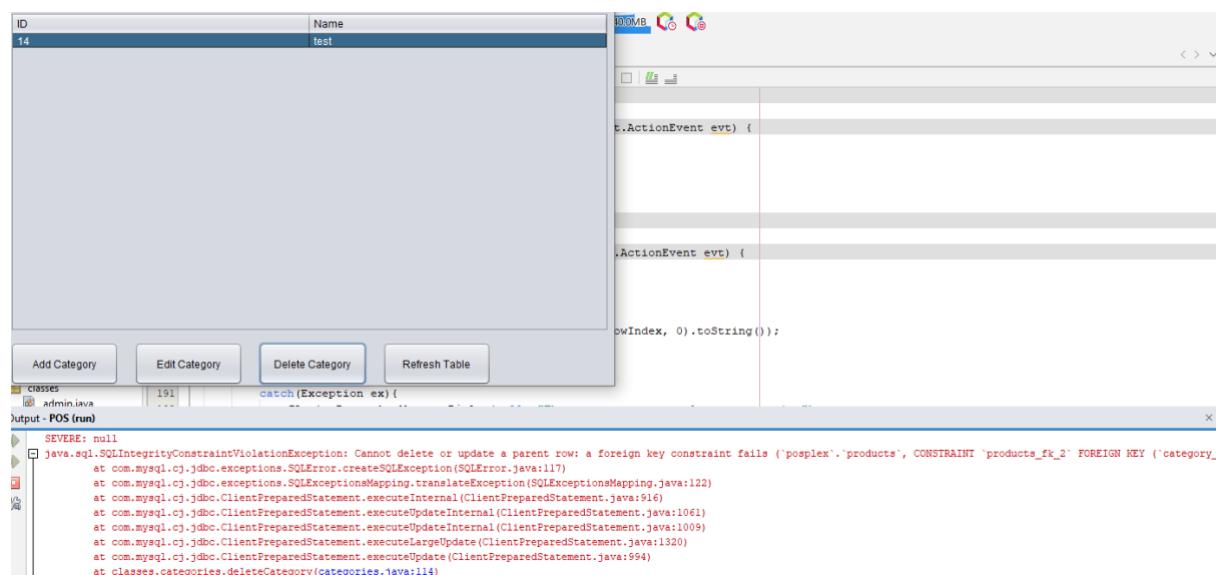
    int rowIndex = categoryTable.getSelectedRow();
    int id = Integer.valueOf(categoryTable.getValueAt(rowIndex, 0).toString());
    try{
        classes.categories.deleteCategory(id);
    }
    catch(Exception ex){
        JOptionPane.showMessageDialog(null, "There was an error, please try again.");
    }
}
```

Again I retrieve the row index by using the getSelectedRow() method.

Then using the rowIndex variable I retrieve the id for the selected category.

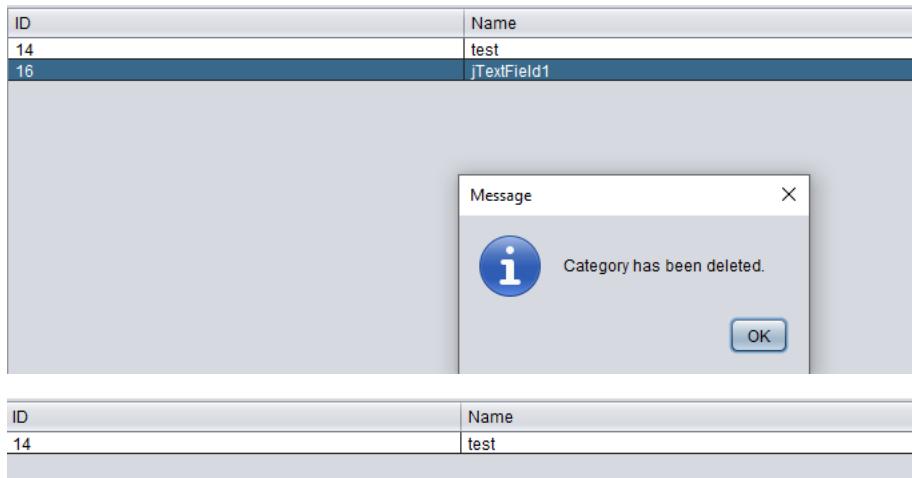
Then I call the deleteCategory method and pass in the retrieved id.

This is what happened when I used the method.



I got an error because of the foreign key constraint as I can't delete a category that has products that belong to it due to the relationship in the database and because this would cause data inconsistency between the linked tables in the database.

However, when I tried to delete a category that had no foreign key constraints the method worked as seen below:



The foreign key error is a good thing so I will bypass it by using a soft delete instead of a hard delete. This means that when the user tries to delete a category that has products that belong to it I will set the specific category record to not visible in the jTable so the user can no longer see it however the category will still exist and therefore there will be no foreign key constraint error.

There are two ways that I could implement a soft delete. The first way is to individually hide each record when it is deleted by the user, the second way to do it is to use a flag in the table to show when a record has been deleted and to only fill the jTable with records that have the deleted flag set to false.

I plan to implement the soft delete using the second method. I plan to do this by first adding the deleted flag field to the categories table, then updating the categories code so that when the user presses the delete button it sends an update query that sets the deleted flag to true and then lastly I will have to update the code that retrieves all the data from the table to only retrieve records with a deleted flag set to false.

4.5.13.1 Adding Deleted Flag To Table

I decided to use an integer datatype field instead of Boolean as it is simpler.

```
1 • ALTER TABLE categories
2   ADD deletedflag int
```

This is the query I used to update the categories table.

I then used the query below to set a default value for the deletedflag column so that when a new category is created the deletedflag field is set to 0.

```
1 • ALTER TABLE categories
2   ALTER deletedflag SET DEFAULT 0
```

This is the result.

category_pk	category_name	deletedflag
14	test	0

4.5.13.2 Updating Categories Code

Next I updated the categories code as seen below.

```
public static void deleteCategory(int catID){
    //deletes selected category

    Connection connection = database.getConnection();
    PreparedStatement ps;

    try{
        ps = connection.prepareStatement("UPDATE categories SET deletedflag = 1 WHERE category_pk = ?");
        ps.setInt(1, catID);
        //ps.setInt(2, deletedflag);
        if (ps.executeUpdate() != 0){
            JOptionPane.showMessageDialog(null, "Category has been deleted.");
        }
        else{
            JOptionPane.showMessageDialog(null, "There was an error, please try again.");
        }
    }
    catch(SQLException ex){
        Logger.getLogger(categories.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

I changed the query from a delete statement to an update statement that sets the selected records deletedflag field to 1.

I tested this and it works as seen below.

The screenshot shows a Java application interface. On the left, there is a table with two columns: 'ID' and 'Name'. The rows contain the following data:

ID	Name
14	test
17	jTextField
18	j

In the center, a message dialog box titled 'Message' is displayed, showing an info icon and the text 'Category has been deleted.' with an 'OK' button.

On the right, there is another table showing the same data, with the 'deletedflag' column explicitly labeled. The data is as follows:

	category_pk	category_name	deletedflag
▶	14	test	0
	17	jTextField	0
	18	j	1

4.5.13.3 Updating Data Retrieval Code

Then I updated the code that retrieves all the data from the categories table as seen below.

Development and Testing

```
public ArrayList<categories> categoryList () {
    //creates an arraylist
    //creates a new instance of "categories" (as shown in method earlier) with data from the categor
    //adds the new category to the arrayList

    ArrayList<categories> catList = new ArrayList<>();

    connection = database.getConnection();
    PreparedStatement ps;
    ResultSet rs;

    String query = "SELECT * FROM categories WHERE deletedflag = 0;";

    try{
        ps = connection.prepareStatement(query);
        rs = ps.executeQuery();
        categories category;
        while (rs.next()){
            category = new categories(rs.getInt("category_pk"), rs.getString("category_name"));

            catList.add(category);
        }
    } catch (SQLException ex){
        Logger.getLogger(categories.class.getName()).log(Level.SEVERE, null, ex);
    }
    return catList;
}
```

I changed the query by adding the where clause so that only records with a deletedflag set to 0 will be retrieved and added to the ArrayList.

This worked as seen below.

ID	Name
14	test
17	jTextFie

Now the delete method works as seen below:

ID	Name
14	test
17	jTextFie

Message

Category has been deleted.

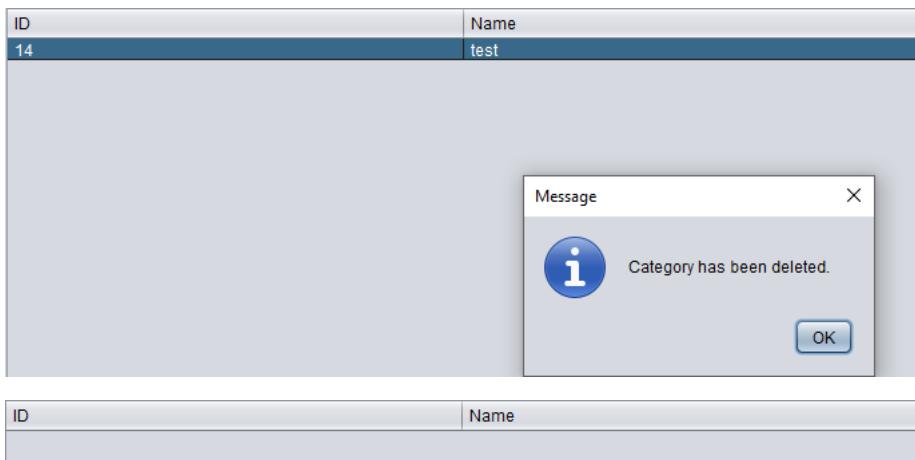
OK

	category_pk	category_name	deletedflag
▶	14	test	0
	17	jTextFie	1
*	18	j	1
*	NULL	NULL	NULL

ID	Name
14	test

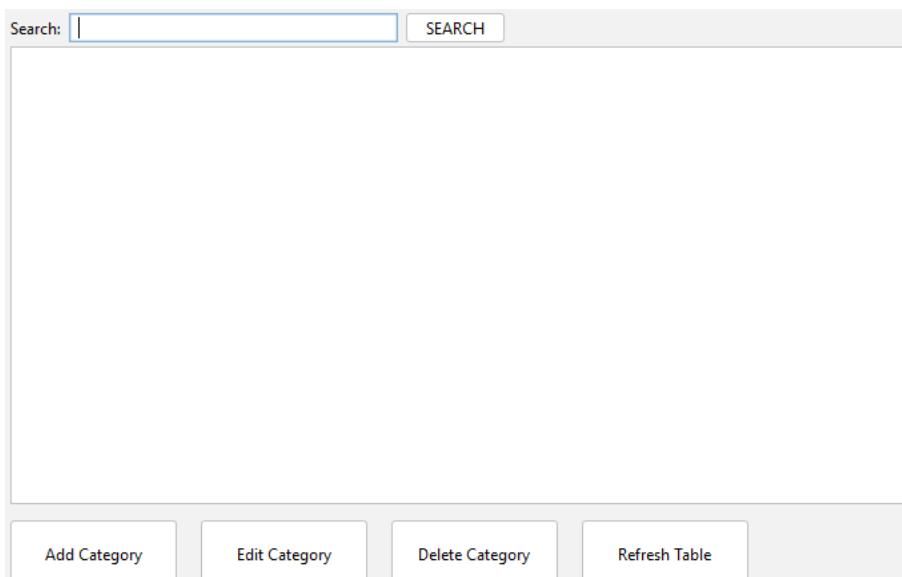
This worked for the category that had foreign key constraints to products as seen below:

Development and Testing



4.5.14 Adding Search Function

Next I plan to add the search function to the categories screen so that the user can search for specific categories.



First I had to add a search bar and button to the categories jframe as seen above.

Then I had to edit the categoryList method in the categories class as seen below.

Development and Testing

```
public ArrayList<categories> categoryList (String searchVal){  
    //creates an arraylist  
    //creates a new instance of "categories" (as shown in method earlier) with data from the catego  
    //adds the new category to the arrayList  
  
    ArrayList<categories> catList = new ArrayList<>();  
  
    connection = database.getConnection();  
    PreparedStatement ps;  
    ResultSet rs;  
  
    String query = "SELECT * FROM categories WHERE deletedflag = 0 AND category_name LIKE ?;";  
  
    try{  
        ps = connection.prepareStatement(query);  
        ps.setString(1, searchVal);  
        rs = ps.executeQuery();  
        categories category;  
        while (rs.next()) {  
            category = new categories(rs.getInt("category_pk"), rs.getString("category_name"));  
  
            catList.add(category);  
        }  
    }  
    catch (SQLException ex){  
        Logger.getLogger(categories.class.getName()).log(Level.SEVERE, null, ex);  
    }  
    return catList;  
}
```

I added the String parameter searchVal.

I edited the query by adding the LIKE factor at the end paired with the placeholder ?.

I used a set string line to set the placeholder to the searchVal parameter.

This means that the method will only retrieve categories that have names that match the searchVal or have similarities.

Then I had to edit the fillTable method in the categoryScreen class as seen below.

```
public void fillTable(String searchVal){  
    //fill the jTable with contents of the ArrayList  
  
    ArrayList <classes.categories> catList = category.categoryList(searchVal);  
    String [] columnNames = {"ID","Name"};  
    Object [][] rows = new Object[catList.size()][2];  
  
    for(int i = 0; i<catList.size(); i++){  
        rows[i][0] = catList.get(i).getCatPK();  
        rows[i][1] = catList.get(i).getName();  
    }  
  
    classes.table model = new classes.table(rows, columnNames);  
    categoryTable.setModel(model);  
}  
  
public void setColumnNames(){  
    //sets jTable column names  
  
    categoryTable.getColumnModel().getColumn(0).setHeaderValue("ID");  
    categoryTable.getColumnModel().getColumn(1).setHeaderValue("Name");  
}
```

Again I added the searchVal parameter to the method.

Then I passed the searchVal parameter into the categoryList method.

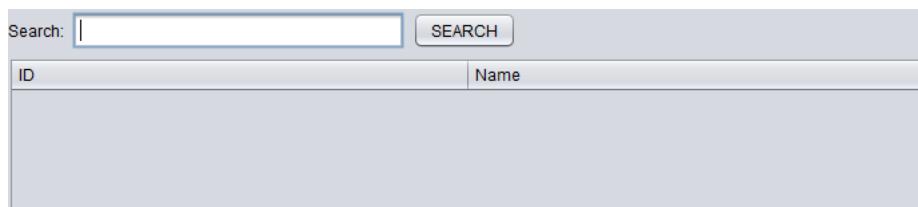
Development and Testing

```
private void searchButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    fillTable(searchTextField.getText());
    setColumnNames();
}
```

This edit calls the fillTable method when the searchButton is pressed and the method takes the value in the search text field as a parameter.

This means that the user inputter search value is passed into the fillTable method as a parameter, then the categoryList method is called and the search value is passed into this method. The search value is then used when retrieving the categories so that only categories matching the search value are retrieved.

Initially when testing this it didn't work and the jTable wasn't filled with any data as seen below.

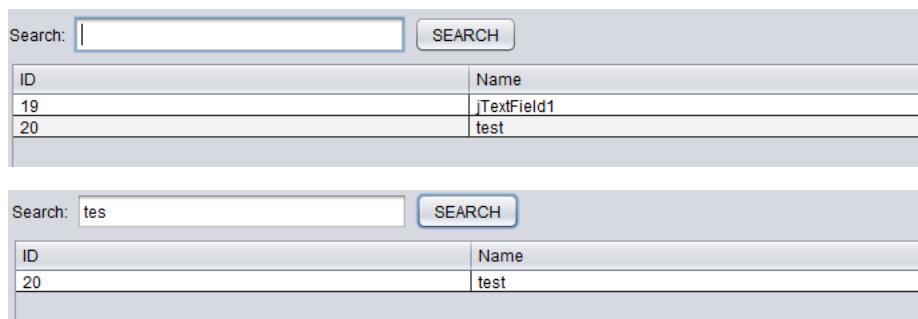


I realised this was because I hadn't specified how the LIKE function should be used so I edited the categoryList method as seen below:

```
ps.setString(1, "%" + searchVal + "%");
```

By adding the two percentage signs I am specifying so that the LIKE operator will find any records in that table that have the search value in any position in the category name.

After testing now it worked as seen below:



4.5.15 Adding Categories To Combo Box

Next I had to develop the method that adds categories to a hashmap which is then used to populate the combo box in the products section.

Development and Testing

A HashMap is similar to an arrayList except instead of using an index to access corresponding items it uses key/value pairs. This means that you can search for a value by using the key similar to how you can search for an item by using the index in an arrayList.

The code I developed is below:

```
public HashMap<String, Integer> addToCombo() {
    //creates a new hashmap that can store strings and integers
    //adds category from the categories table in the database to the product combo box

    HashMap<String, Integer> map = new HashMap<>();
    connection = database.getConnection();
    Statement st;
    ResultSet rs;
    try{
        st = connection.createStatement();
        rs = st.executeQuery("SELECT category_pk, category_name FROM categories");
        categories category;
        while(rs.next()){
            category = new categories(rs.getInt(1), rs.getString(2));
            map.put(category.getName(), category.getCatPK());
        }
    }
    catch(SQLException ex){
        Logger.getLogger(categories.class.getName()).log(Level.SEVERE, null, ex);
    }
    return map;
}
```

First I create a new HashMap called map.

Then I query the database to select all the categories.

Next I create new objects of the categories class with the data from the database.

Then I add the category name and primary key to the map using the put() method and using the getter methods.

By adding the name and primary key in the order seen above, it means the name will be the key and the primary key will be the value for the HashMap.

Next I added this line to the method to test whether it worked or not

```
System.out.println(map);
```

I then called the method and this was the result.

```
{searchCategoryTest=22, test=20, updateCategoryTest=21, jTextField1=17, j=18, jTextField1=19}
```

Although it worked in the sense that it retrieved the categories it retrieved categories that had been deleted.

I realised this was because I didn't add a where clause to the query so it was retrieving all the categories rather than just undeleted categories. I edited the code as seen below:

```
rs = st.executeQuery("SELECT category_pk, category_name FROM categories WHERE deletedflag = 0");
```

This was the result.

```
{searchCategoryTest=22, test=20, jTextField1=19}
```

This is a success as only undeleted categories were retrieved.

Lastly during testing I found that the update category jframe was still open after the category had been updated so I had to add a dispose method as seen below to the code under the update category button in the update category screen.

```
private void updateButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //updates record with text field input from user

    classes.categories category;

    String category_name = nameTextField.getText();
    category = new classes.categories(category_pk, category_name);
    classes.categories.updateCategory(category);
    this.dispose();
}
```

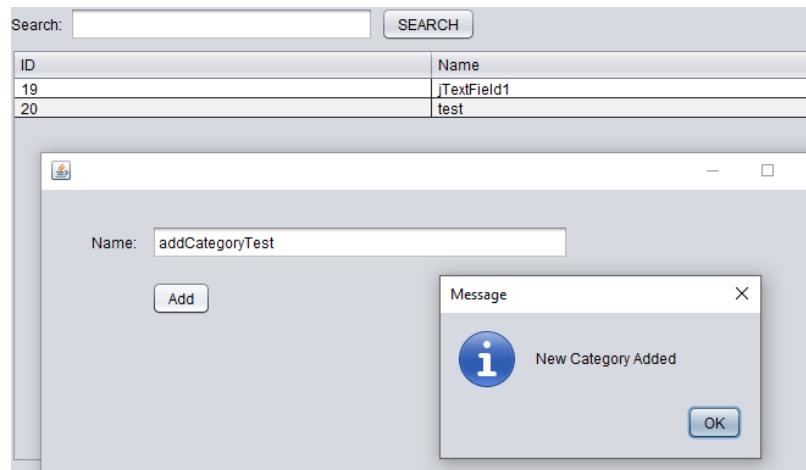
After re-testing it worked and the updateCategoryScreen was closed immediately after the successful update message was closed by the user.

4.5.16 Test Plan

Test Number	How To Test	Expected Result	Actual Result
1	Attempt to add a category	Category should be added to table.	Category was added to table.
2	Attempt to update a category	Category should be updated with new values.	Category was updated.
3	Attempt to delete a category	Category should be deleted from jTable.	Category was deleted from jTable.
4	Type into search bar	Only categories with names similar to the search input should be shown.	Only categories with similar or the same name were shown.

4.5.17 Evidence

4.5.17.1 Test 1



Development and Testing

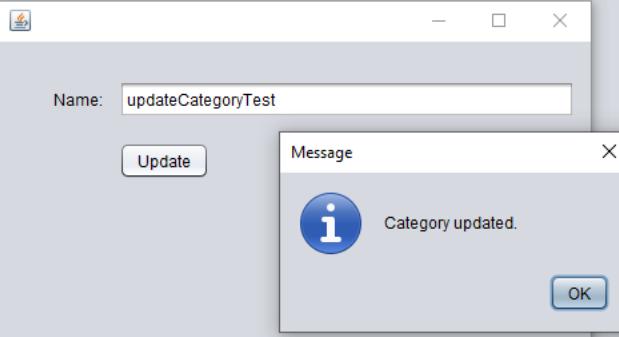
Search:

ID	Name
19	jTextField1
20	test
21	addCategoryTest

4.5.17.2 Test 2

Search:

ID	Name
19	jTextField1
20	test
21	addCategoryTest


An update dialog is open over the main window. It contains a text field labeled "Name:" with the value "updateCategoryTest" and a button labeled "Update". A message dialog box is overlaid on the update dialog, showing an information icon and the text "Category updated." with an "OK" button.

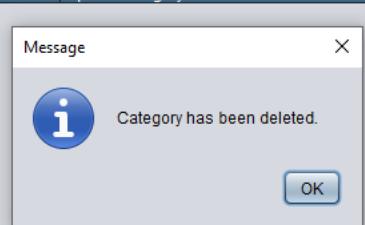
Search:

ID	Name
19	jTextField1
20	test
21	updateCategoryTest

4.5.17.3 Test 3

Search:

ID	Name
19	jTextField1
20	test
21	updateCategoryTest


A message dialog box is displayed, showing an information icon and the text "Category has been deleted." with an "OK" button.

Search:

ID	Name
19	jTextField1
20	test

4.5.17.4 Test 4

Search: <input type="text"/>	<input type="button" value="SEARCH"/>
ID	Name
19	jTextField1
20	test
22	searchCategoryTest

Search: <input type="text"/>	<input type="button" value="SEARCH"/>
ID	Name
22	searchCategoryTest

4.5.18 Success Criteria

Success Criteria Number	Success Criteria	Achieved	Notes
5	Allow user to add items to categories	Yes	This criteria is half achieved as the user can create categories and the categories will be added to the HashMap however the products section is not finished so they cant add products to categories yet.

4.5.19 Review/Stage Evaluation

In this stage I developed the functionality behind the categories section; this meant developing methods to allow the user to add, update and delete categories.

The first method I created was the add category method, this method simply takes the user inputted category name and inserts it into the categories table in the database, a new record is created with that name.

Next, I created the category retrieval method and the fill table method. These methods retrieve all the categories in the category table in the database and then store them in an array as objects of the category class and create a table model for the jtable by filling the a 2d array with all the categories data and passing this array into the constructor method to create a new object of the table class with the values from the array and then finally setting the jtable to this model.

Next, I created the method to update a category. This method is similar to the add category method in that it takes the new category name and instead of using an insert query it uses an update query and sets the category name field for the selected category to the new category name.

Lastly, I created the delete categories method. Initially when creating this method, It performed a hard delete meaning the category would be completely deleted from the database. However, through testing I found that due to the foreign key constraint from a product to a category, I would get an error when deleting categories that have products that belong to them. To overcome this problem, I instead implemented a soft delete. This meant adding a deleted flag field to the

categories table and when a product is deleted, this flag is set to true. Then I updated the category retrieval code, so it only retrieves categories that have a deleted flag that is set to false hence when the user deletes a category, it isn't actually deleted but rather it is hidden from the user.

The last two pieces of functionality that I added to this section were: a method that adds the categories to a HashMap so that later on this HashMap can be used to fill a combo box and I added a search bar and updated the category retrieval code to only retrieve categories that had a category name that included the search value.

This section was okay, I was able to achieve the first half of the category success criteria and the other half will be achieved when I have developed the functionality to allow the user to add products to categories. It is good that I have already come across the foreign key constraint error and found a way to bypass it by implementing a soft delete and I might need to do this in other section of the code as well.

4.6 Stage 5/Products

NOTE: Many of the methods used in this class will be similar to the categories section.

4.6.1 Developing Product Instance Code

The first method I made in this class, similar to the categories class, is the products class constructor below which takes parameters for all the fields of a product in the products table and sets the private variables for all of these fields to the parameter variables. This constructor creates an instance of the product class in the format it would be in as a record in the products table. The code is below.

```
public products(Integer pk, String name, String description, float price, float cost, int fk, Date createdOn, Date updated, boolean deletedFlag
{
    this.product_pk = pk;
    this.product_name = name;
    this.product_description = description;
    this.product_sellprice = price;
    this.product_cost = cost;
    this.createdby_fk = fk;
    this.createdon = createdOn;
    this.lastupdated = updated;
    this.deletedflag = deletedFlag;
    this.category = categoryName;
    this.stock = stock;
    this.catFK = categoryfk;
}
```

This constructor is called in the code below which is the productList method which is a method of type ArrayList, an array list is a resizable array. The ArrayList holds products hence the earlier code and this method takes a parameter of val which will be used later on for the search function.

4.6.2 Developing Product Creation Code

One of the methods I created in the products class was the add product method. This method creates a new product using user inputted information. The code is below.

```
public static void addProduct(products product){
    //creates a new product using user inputs and adds it to the products table in the database

    Connection connect = database.getConnection();
    PreparedStatement ps = null;
    try{
        ps = connect.prepareStatement("INSERT INTO products(product_name, product_description, product_sellprice, product_cost, category, stock,
        ps.setString(1, product.getName());
        ps.setString(2, product.getDescription());
        ps.setFloat(3, product.getPrice());
        ps.setFloat(4, product.getCost());
        ps.setString(5, product.getCat());
        ps.setInt(6, product.getStock());
        ps.setInt(7, product.getFK());
        ps.setInt(8, product.getCatFK());

        if(ps.executeUpdate() != 0){
            JOptionPane.showMessageDialog(null, "New Product Added");
        }
        else{
            JOptionPane.showMessageDialog(null, "Error");
        }
    }
    catch (SQLException ex) {
        Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
    }
}

stock, createdby_fk, createdon, lastupdated, deletedflag, category_fk) VALUES(?,?,?,?,?, ?, ?, sysdate(), sysdate(), 0, ?)"';


```

The insert statement inserts a record into the products table setting most of the fields to a ? placeholder but setting certain fields to specific values, createdon is the date the product was created and is therefore set to sysdate() which is the current date, lastupdated is the date the product was last updated and as the product is being created here it is also set to sysdate() and deletedflag shows whether or not the product is deleted and is therefore set to 0.

The setString lines set each placeholder to a value for example the first placeholder which is for the product name is set to the value of the getName method for the product. Note: I have getter and setter methods for each field.

4.6.3 Add Product Button Code

Once the product creation code was created I had to add the code to call the new product screen/jframe under the add product button. The code is below.

```
private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    newProductScreen productadd = new newProductScreen();
    productadd.setDefaultCloseOperation(newProductScreen.DISPOSE_ON_CLOSE);
    productadd.pack();
    productadd.setVisible(true);
    productadd.userPrimaryKey = userPrimaryKey;
}
```

The code above calls the jform to allow the user to add a new product and passed in the userPrimaryKey.

In the new product screen class I had to write the code to add the categories to the combo box so the users can select categories from the combo box.

```
public void comboBoxAttach() {
    //adds categories to the combo box

    classes.categories category = new classes.categories();
    HashMap<String, Integer> map = category.addToCombo();
    for (String set: map.keySet()) {
        categoryComboBox.addItem(set);
    }
}
```

Previously in the categories section I created the addToCombo() method which adds the categories to the HashMap: map. The method above adds the keys from map to the combo box.

The first line creates a new objects of the categories class called category.

The next line creates the HashMap map so that the keys are strings and the values are integers and sets it equal to the returned value of the addToCombo() method on the object category. This method will return the map with the categories in it so now map is filled with the categories.

The next line adds the keys to the comboBox using a for loop. I use the String set to step through map from the start of map to the end. I use the keySet() method so set will be equal to the key in the map and not the value. Then I use the addlItem() method taking set as a parameter to add the keys to the category combo box.

Next I had to develop the code in the new product screen which will take the users inputs and pass them into the products method to create a new object products with the users inputs taken as parameters. The code is below.

```
private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //creates new products using inputs from user

    classes.products product;
    classes.categories category = new classes.categories();
    HashMap<String, Integer> map = category.addToCombo();

    String product_name = nameTextField.getText();
    String product_description = descriptionTextField.getText();
    float product_sellprice;
    float product_cost;
    int stock;
    int categoryfk;
    String categoryName = null;

    stock = Integer.valueOf(stockTextField.getText());

    categoryfk = map.get(categoryComboBox.getSelectedItem().toString());
    categoryName = categoryComboBox.getSelectedItem().toString();
    product_sellprice = Float.parseFloat(priceTextField.getText());
    product_cost = Float.parseFloat(costTextField.getText());

    product = new classes.products(1, product_name, product_description, product_sellprice, product_cost, userPrimaryKey, null, null, false, ca

    classes.products.addProduct(product);
    this.dispose();
```

First I call the products constructor.

Then I instantiate a new object of the categories class called category.

Again, I set map equal to the returned value of the addToCombo method which returns map filled with categories.

Then I declare variables for all of the parameters in the products constructor method for example I create a product_name variable of type String and assign it the value of the text in the nameTextField using the getText() method.

For certain variables such as categoryfk I use the getSelectedItem() method to return the selected item from the combo box(which will be a key, so a category name, in the map), then I use the toString() method to convert the item to a string, then I use the get() method as the category primary keys are the values in the map so they aren't added to the combo box so they must be retrieved form the map using the corresponding key which will be a category name.

Then I instantiate products which is an object of the products class and pass in the earlier declared variables as parameters.

Then I use the addProduct() method and pass in product as a parameter so a new product is created in the database with the users inputs.

4.6.4 Developing Product Table Data Retrieval Code

```
public ArrayList<products> productList(String searchVal) {
    String query = "SELECT products.product_pk, product_name, product_description, "
        + "product_sellprice, product_cost, createdby_fk, createdon, lastupdated, "
        + "deletedflag, category, stock FROM products WHERE CONVERT(products.product_pk, "
        + "char(10)) AND product_name AND product_description AND CONVERT(product_sellprice, "
        + "char(10)) AND CONVERT(product_cost, char(10)) AND CONVERT(createdby_fk, char(10)) "
        + "AND CONVERT(createdon, char(10)) AND CONVERT(lastupdated, char(10)) AND "
        + "CONVERT(deletedflag, char(10)) AND category AND CONVERT(stock, char(10)) LIKE ?";
```

Initially I used the code above to retrieve the data from the table however this code didn't work as when I tried printing the ArrayList it came up empty as seen below.

[]

So I instead used a much simpler select statement below

```
String query = "SELECT * FROM products WHERE deletedflag = 0 AND product_name LIKE ?";
```

The query selects all the products from the products table where the deletedflag is 0.

This worked as seen below.

```
[classes.products@7c7ecf89, classes.products@2ea4d728, classes.products@714a16a, classes.products@622f5ace]
```

The ArrayList was filled with products so this query works.

Development and Testing

```
try{
    ps = connection.prepareStatement(query);
    rs = ps.executeQuery();
    products product;
    while (rs.next()){
        product = new products(rs.getInt("product_pk"), rs.getString("product_name"), rs.getString("product_descrip' +
            'tions"));
        prodList.add(product);
    }
}
catch (SQLException ex){
    Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
}
return prodList;
```

The code above creates new objects of the products class called product and the object takes the values from the database and is added to the array list.

```
ArrayList<products> prodList = new ArrayList<>();
```

The first line connects to the database and creates the PreparedStatement object taking the query as a parameter of the prepareStatement method.

The next line gets the results from the table by setting result set to the value of the executeQuery() method on the PreparedStatement object.

Next I use a while loop to loop through the result set, creating an instance of the products class taking the data from the result set as parameters which creates an object product which has all the values of the record in the database. This product is then added to prodList. Then this will be repeated for each record retrieved in the result set.

Whole method screenshot:

```
public ArrayList<products> productList(String searchVal){
    //creates an arraylist
    //creates a new instance of "products" (as shown in method earlier) with data from the products table in the database
    //adds the new product to the arrayList

    ArrayList<products> prodList = new ArrayList<>();

    connection = database.getConnection();
    PreparedStatement ps;
    ResultSet rs;

    String query = "SELECT * FROM products WHERE deletedflag = 0;";

    try{
        ps = connection.prepareStatement(query);
        rs = ps.executeQuery();
        products product;
        while (rs.next()){
            product = new products(rs.getInt("product_pk"), rs.getString("product_name"), rs.getString("product_descrip' +
                'tions"));
            prodList.add(product);
        }
    }
    catch (SQLException ex){
        Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
    }
    System.out.println(prodList);
    return prodList;
}
```

This method worked as seen below.

```
[classes.products@7c7ecf89, classes.products@2ea4d728, classes.products@714a16a, classes.products@622f5ace]
```

Development and Testing

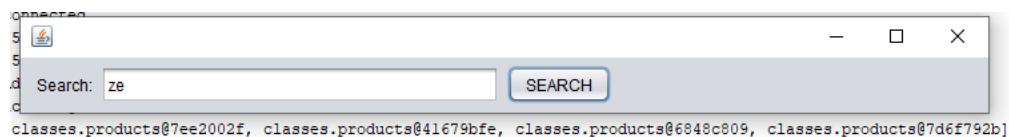
Next I need to update the method so that the search function works, the method already takes searchVal as a parameter so all I have to do is update the query and add a setString() line as shown below.

```
String query = "SELECT * FROM products WHERE deletedflag = 0 AND product_name LIKE ?";  
  
try{  
    ps = connection.prepareStatement(query);  
    ps.setString(1, "%" + searchVal + "%");
```

The query is now updated so that only products that are like the placeholder ? are retrieved.

So I set the placeholder as seen in the setString line, this means that any product that contain searchVal in their name at any place are retrieved.

I tested this method using a print line, the results are below.



All the products were retrieved even though I searched for a specific product.

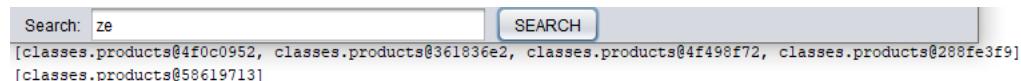
I realised this was because I hadn't coded the functionality behind the text field or the search button.

First I had to add an action listener to the search button.

Then under the button method I added the code below.

```
private void searchButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    fillTable(searchTextField.getText());  
    setColumnNames();  
}
```

I tested the search function again with a print statement and the results are below.



The first line was before the "ze" search and the second line was after. As you can see the search method worked as there was only one product in the array list after the search.

4.6.5 Developing Populate Table Code

I also had to develop the code to fill the jTable with the data retrieved from the products table in the database. The code is below.

```
public void fillTable(String searchVal){
    //fills the jTable with contents of arraylist

    classes.products prod = new classes.products();
    ArrayList <classes.products> prodList = prod.productList(searchVal);
    String [] columnNames = {"ID", "Name", "Description", "Price", "Cost", "Category", "Stock"};
    Object [][] rows = new Object[prodList.size()][7];

    for(int i = 0; i<prodList.size(); i++){
        rows[i][0] = prodList.get(i).getProductPK();
        rows[i][1] = prodList.get(i).getName();
        rows[i][2] = prodList.get(i).getDescription();
        rows[i][3] = prodList.get(i).getPrice();
        rows[i][4] = prodList.get(i).getCost();
        rows[i][5] = prodList.get(i).getCat();
        rows[i][6] = prodList.get(i).getStock();
    }

    classes.table model = new classes.table(rows, columnNames);
    productTable.setModel(model);
}
```

This method takes the parameter searchVal which is passed in to the productList method when called.

In the first line I call the products constructor creating a new instance of products called prod.

The second line sets prodList which is an ArrayList that can contain objects of the products class to the returned value of the productList() method

The third line of code fills the 1d array columnNames with the correct columnNames and the fourth line of code creates the 2d array rows taking the product array list “prodList”’s size as a value so that it has the right amount of rows and it also takes 7 as the second value so it has the right amount of columns.

Next, I use a simple for loop to loop through the product array list. It takes the row in position i, which is a counter, and sets the first column of this row to the product primary key held in position i in the product array list. This is repeated 7 times for each column as seen above with the second value in the 2d array incrementing by 1 each time so that each piece of data is in the correct corresponding column.

Lastly, I instantiate the class ‘table’ creating an instance of table called model with the parameters rows and columnNames so a table object is created with the right amount of rows, columns and column names.

Then I use the setModel() method for the productTable to set the productTable model to the object model.

This worked as seen below:

Development and Testing

Search:

ID	Name	Description	Price	Cost	Category	Stock
4	jTextField1	jTextField2	50.55	35.55	jTextField1	5
7	jTextField1	jTextField2	50.55	35.55	jTextField1	5
9	jTextField1	jTextField2	50.55	35.55	jTextField1	5
10	zebra	jTextField2	50.55	35.55	searchCatego...	5

The search method also worked as seen below:

Search: ze

ID	Name	Description	Price	Cost	Category	Stock
10	zebra	jTextField2	50.55	35.55	searchCatego...	5

4.6.6 Developing Delete Product Code

Next, I developed the code to allow the user to delete a product. I plan to develop a soft delete rather than a hard delete which means that instead of actually deleting the product it just hides it from the user. The reason I will do this is because later on in the project I will code the orders section, order records will relate to and contain products via a foreign key constraint , this means that if I was to hard delete a product there would be an error caused by the foreign key constraint.

This method is quite simple, all I have to do is send an update query to the database to update the selected products and change the deletedflag field to 1 so that it is not retrieved by the productList method. The code is below.

Development and Testing

```
public static void deleteProduct(int id){
    //deletes selected product

    Connection connection = database.getConnection();
    PreparedStatement ps;

    try{
        ps = connection.prepareStatement("UPDATE products SET deletedflag = 1 WHERE product_pk = ?;");
        ps.setInt(1, id);
        //ps.setInt(2, deletedflag);
        if (ps.executeUpdate() != 0){
            JOptionPane.showMessageDialog(null, "Product has been deleted.");
        }
        else{
            JOptionPane.showMessageDialog(null, "There was an error, please try again.");
        }
    }
    catch(SQLException ex){
        Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

I pass the integer variable id into this method as a way to identify the selected product.

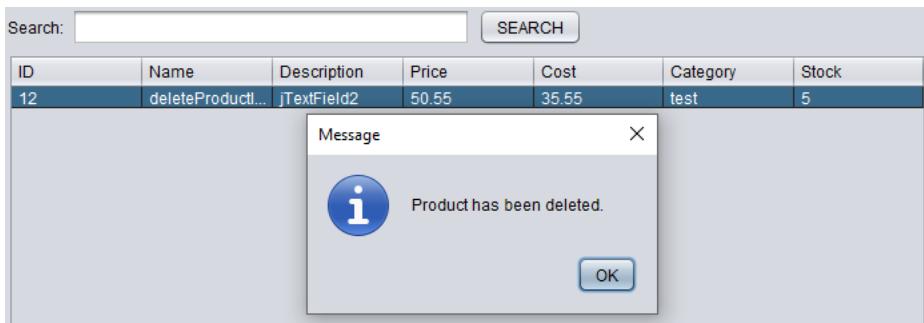
The query is very simple, it simply updates the products table and sets the deletedflag field to 1 for all the products that have a product_pk field equal to the placeholder ?

I then use the setInt() method to set the placeholder to the variable id.

Then I use the executeUpdate() method which returns the number of rows affected by the PreparedStatement and if this is not equal to 0 then it was a success so a successful message is displayed. Else an unsuccessful message is displayed.

I tested this by setting id to 12, which is the primary key for a test product, under the action listener on the delete button. The results are below.

```
int id = 12;
try{
    classes.products.deleteProduct(id);
}
catch(Exception ex){
    JOptionPane.showMessageDialog(null, "There was an error, please try again.");
}
```



4.6.7 Delete Product Button Code

Now that I know the method works I have to add the proper code so that the method deletes the product selected by the user. The code is below:

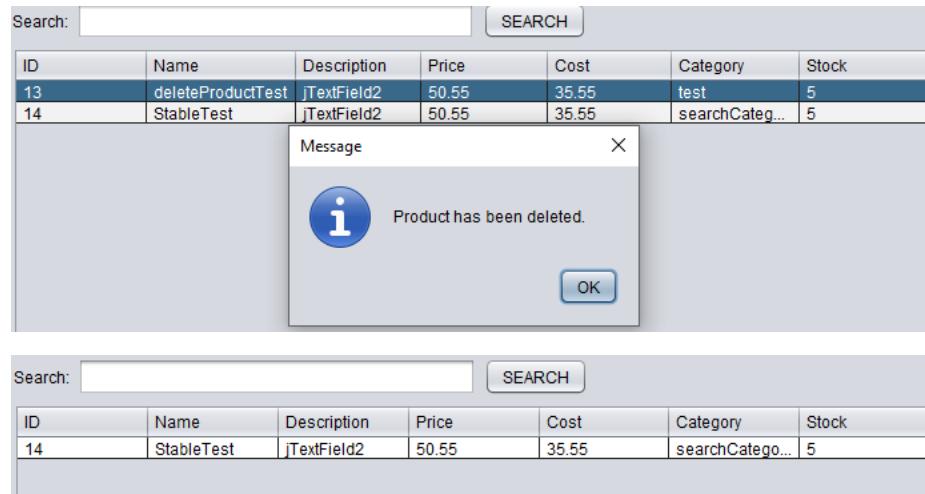
```
private void deleteButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int rowIndex = productTable.getSelectedRow();
    int id = Integer.valueOf(productTable.getValueAt(rowIndex, 0).toString());
    try{
        classes.products.deleteProduct(id);
    }
    catch(Exception ex){
        JOptionPane.showMessageDialog(null, "There was an error, please try again.");
    }
}
```

I use the `getSelectedRow()` method to retrieve the row index for the selected row in the `productTable`.

Then I retrieve the product id/primary key by using the `getValueAt()` method taking the `rowIndex` and `0` as parameters so it retrieves the value at the correct selected row index in the column at position `0` which is the primary key/id column. Then I convert this value to a string. Then I convert it to an integer.

Lastly I pass the id into the `deleteProduct()` method as seen so the correct product is deleted.

This method works as seen below:



4.6.8 Developing Update Product Code

Next I developed the code to allow the user to update products which is below.

```
public static void updateProduct(products product) {
    //updates selected product with user inputs

    Connection connection = database.getConnection();
    PreparedStatement ps;

    try{
        ps = connection.prepareStatement("UPDATE products SET product_name = ?, product_description = ?, product_sellprice = ?, product_cost = ?,
                                         category = ?, stock = ?, createdby_fk = ?, createdon = createdon, lastupdated = sysdate(), deletedflag = 0, category_fk = ? WHERE product_pk = ?");
        ps.setString(1, product.getName());
        ps.setString(2, product.getDescription());
        ps.setFloat(3, product.getPrice());
        ps.setFloat(4, product.getCost());
        ps.setString(5, product.getCat());
        ps.setInt(6, product.getStock());
        ps.setInt(7, product.getFK());
        ps.setInt(8, product.getProductPK());
        ps.setInt(9, product.getCatFK());

        //ps.setDate(8, product.getDateCreated());
        //ps.setDate(9, product.getLastUpdated());

        if(ps.executeUpdate() != 0){
            JOptionPane.showMessageDialog(null, "Product updated.");
        }
        else{
            JOptionPane.showMessageDialog(null, "There was an error, please try again.");
        }
    }
    catch(SQLException ex){
        Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

category = ?, stock = ?, createdby_fk = ?, createdon = createdon, lastupdated = sysdate(), deletedflag = 0, category_fk = ? WHERE product_pk = ?")

This code is very similar to the add product code except it uses an update query instead and sets each user customizable field to the placeholder ? and updates the product where the product primary key is equal to the selected product primary key. I then set each placeholder to the user inputted value.

4.6.9 Update Product Button Code

Next I had to add the code under the edit product button to call the updateProduct() method and pass in the user inputted value for each field.

```
private void editButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    try{
        updateProductScreen productUpdate = new updateProductScreen();
        int rowIndex = productTable.getSelectedRow();
        //updateProductScreen.userPrimaryKey = userPrimaryKey;

        productUpdate.product_pk = Integer.valueOf(productTable.getValueAt(rowIndex, 0).toString());
        productUpdate.nameTextField.setText(productTable.getValueAt(rowIndex, 1).toString());
        productUpdate.descriptionTextField.setText(productTable.getValueAt(rowIndex, 2).toString());
        productUpdate.priceTextField.setText(productTable.getValueAt(rowIndex, 3).toString());
        productUpdate.costTextField.setText(productTable.getValueAt(rowIndex, 4).toString());
        productUpdate.categoryComboBox.setSelectedItem(productTable.getValueAt(rowIndex, 5));
        productUpdate.stockTextField.setText(productTable.getValueAt(rowIndex, 6).toString());

        productUpdate.setDefaultCloseOperation(updateProductScreen.DISPOSE_ON_CLOSE);
        productUpdate.pack();
        productUpdate.setVisible(true);
    }
    catch (Exception ex){
        JOptionPane.showMessageDialog(null, "There was an error, please select a row and try again.");
    }
}
```

First I call the update product screen which is the jframe the user will use to update the products.

Next I get the row index of the product the user wants to update using the getSelectedRow() method.

Then I update each field of the selected row individually taking the users input from the update product screen and setting the correct column of the selected row to the user inputted value.

This is the code I used under the update product button on the updateProductScreen.

Development and Testing

```
private void updateButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //updates products with user inputs

    classes.products product;
    classes.categories category = new classes.categories();
    HashMap<String, Integer> map = category.addToCombo();

    String product_name = nameTextField.getText();
    String product_description = descriptionTextField.getText();
    float product_sellprice;
    float product_cost;
    int stock;
    String categoryName = null;
    int categoryfk;

    stock = Integer.valueOf(stockTextField.getText());

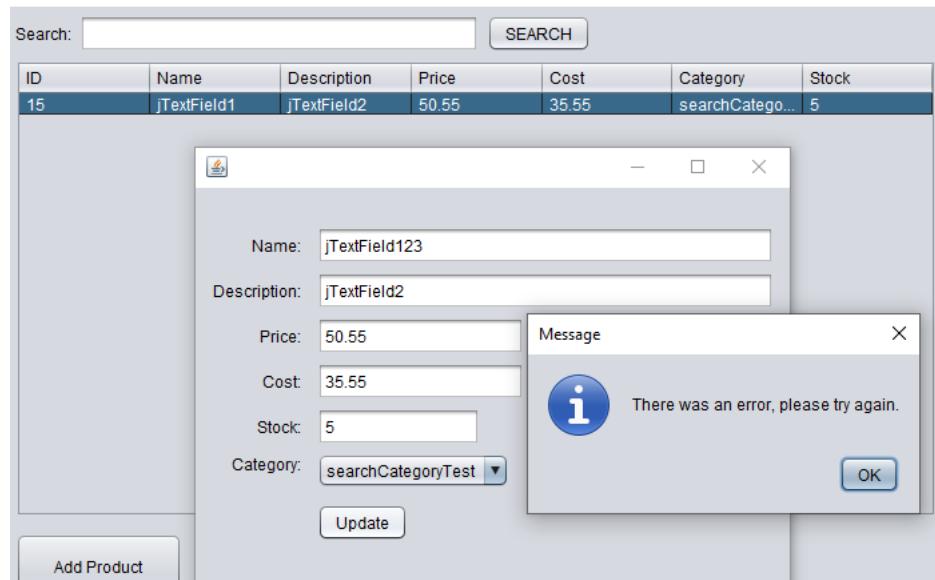
    categoryfk = map.get(categoryComboBox.getSelectedItem().toString());
    categoryName = categoryComboBox.getSelectedItem().toString();
    product_sellprice = Float.parseFloat(priceTextField.getText());
    product_cost = Float.parseFloat(costTextField.getText());

    product = new classes.products(product_pk, product_name, product_description, product_sellprice, product_cost, userP);
    classes.products.updateProduct(product);

}
```

This code is the same as the code under the add product button except instead of calling the `addProduct()` method I call the `updateProduct()` method.

I tested this method and the results are below:



I also printed out the query with the user inputs below:

```
com.mysql.cj.jdbc.ClientPreparedStatement: UPDATE products SET product_name = 'jTextField123', product_description = 'jTextField2', product_sellprice = 50.55, product_cost = 35.55, category_fk = 'searchCategoryTest', stock = 5, createdby_fk = 0, createdon = createdon, lastupdated = sysdate(), deletedflag = 0, category_fk = 15 WHERE product_pk = 22
```

As you can see the query has set `category_fk` to 15 and is updating a product that has a `product_pk` of 22. After looking at the table I could see that it had set the `category_fk` to the selected product primary key.

This is because in the query the `category_fk` is set to the 8th ? placeholder and the `product_pk` is set to the 9th ? placeholder as you can see below.

Development and Testing

```
category_fk = ? WHERE product_pk = ?");
```

However when setting the placeholders to the corresponding getter methods I got them the wrong way round as you can see below.

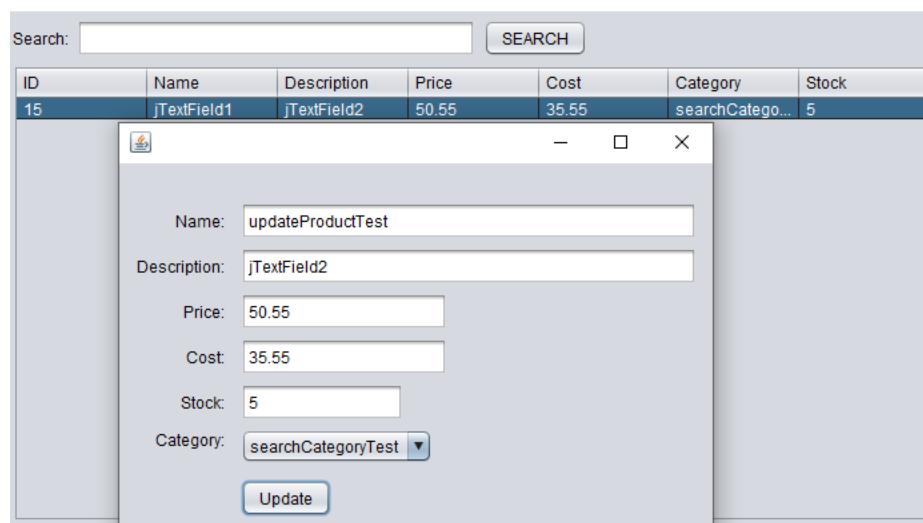
```
ps.setInt(8, product.getProductPK());  
ps.setInt(9, product.getCatFK());
```

This caused the query to attempt to change the category foreign key for the product which cannot be done as it is linked to a specific category and it caused the query to search for a product with a primary key equal to 22 which doesn't exist in my database.

After swapping these around as seen below:

```
ps.setInt(8, product.getCatFK());  
ps.setInt(9, product.getProductPK());
```

I tested the method again and got this error:



```
java.sql.SQLIntegrityConstraintViolationException: Cannot add or update a child row: a foreign key constraint fails ('posplex`.`products', CONSTRAINT `products_fk_1` FOREIGN KEY (`createdby_fk`) REFERENCES `users` (`user_pk`))
```

This is because in the query I had set createdby_fk to a placeholder and set the placeholder to the getter method for the foreign key as seen below:

```
createdby_fk = ?,  
  
ps.setInt(7, product.getFK());
```

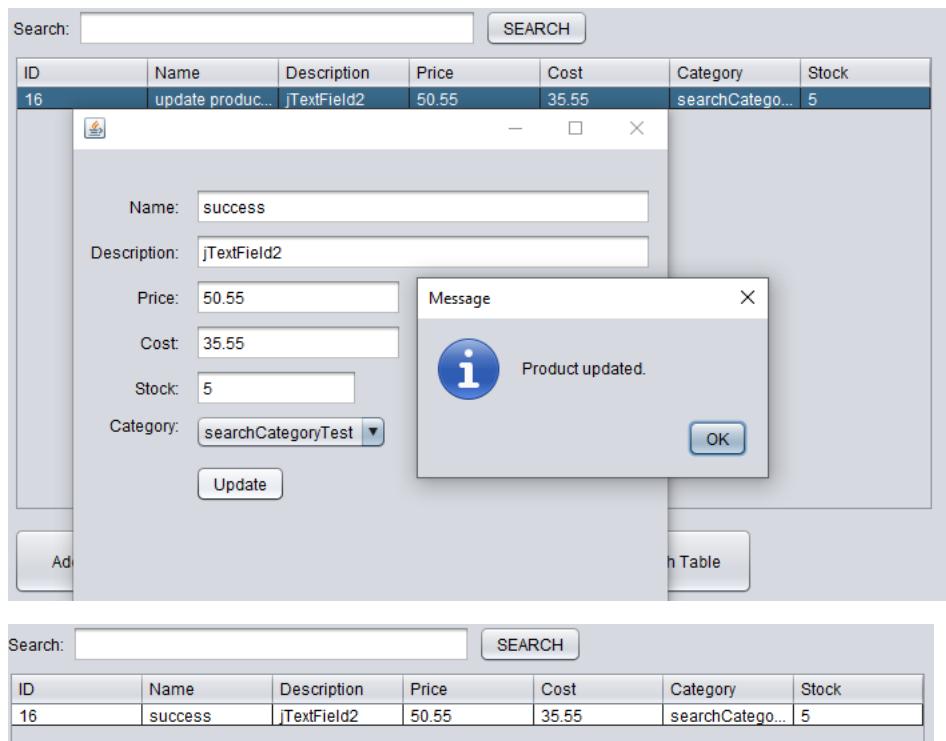
This doesn't work because the createdby_fk field has a foreign key constraint to the users table so it cannot be manually updated.

After changing the query as seen below:

```
createdby_fk = createdby_fk,
```

And removing and updating the prepared statement set lines appropriately. The method worked as shown below:

Development and Testing



The final overall method code is below.

```

public static void updateProduct(products product){
    //updates selected product with user inputs

    Connection connection = database.getConnection();
    PreparedStatement ps;

    try{
        ps = connection.prepareStatement("UPDATE products SET product_name = ?, product_description = ?, product_sellpri");
        ps.setString(1, product.getName());
        ps.setString(2, product.getDescription());
        ps.setFloat(3, product.getPrice());
        ps.setFloat(4, product.getCost());
        ps.setString(5, product.getCat());
        ps.setInt(6, product.getStock());
        ps.setInt(7, product.getCatFK());
        ps.setInt(8, product.getProductPK());

        if(ps.executeUpdate() != 0){
            JOptionPane.showMessageDialog(null, "Product updated.");
        }
        else{
            JOptionPane.showMessageDialog(null, "There was an error, please try again.");
        }
    }
    catch(SQLException ex){
        Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
    }
}

sellprice = ?, product_cost = ?, category = ?, stock = ?, createdby_fk = createdby_fk, createdon = createdon, lastupdated =
lastupdated = sysdate(), deletedflag = 0, category_fk = ? WHERE product_pk = ?");
}

```

Development and Testing

Lastly I had to add the dispose method to the update product screen under the update product button so the screen was closed after the product was updated as seen below.

```
private void updateButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    //updates products with user inputs  
  
    classes.products product;  
    classes.categories category = new classes.categories();  
    HashMap<String, Integer> map = category.addToCombo();  
  
    String product_name = nameTextField.getText();  
    String product_description = descriptionTextField.getText();  
    float product_sellprice;  
    float product_cost;  
    int stock;  
    String categoryName = null;  
    int categoryfk;  
  
    stock = Integer.valueOf(stockTextField.getText());  
  
    categoryfk = map.get(categoryComboBox.getSelectedItem().toString());  
    categoryName = categoryComboBox.getSelectedItem().toString();  
    product_sellprice = Float.parseFloat(priceTextField.getText());  
    product_cost = Float.parseFloat(costTextField.getText());  
  
    product = new classes.products(product_pk, product_name, product_description, product_sellprice, product_cost, u:  
  
    classes.products.updateProduct(product);  
    this.dispose();  
}  
}
```

This worked and the update product screen was closed after the product successfully updated message was closed.

4.6.10 Test Plan

Test Number	How To Test	Expected Result	Actual Result
1	Attempt to add a product	Product should be added and displayed	Product was added to the table and displayed
2	Attempt to update a product	Product should be updated	Product was updated successfully
3	Attempt to delete a product	Product should be soft deleted	Product was deleted from jTable
4	Type into search bar	Only products that contain or are similar to the search should be displayed	Only products that contained the search were displayed

4.6.11 Evidence

Development and Testing

4.6.11.1 Test 1

Screenshot of a Java Swing application window titled "Development and Testing". The window contains a form for adding a product:

Name:	addProductTest
Description:	jTextField2
Price:	50.55
Cost:	35.55
Stock:	5
Category:	searchCategoryTest ▾

Below the form is an "Add" button. A message dialog box titled "Message" is displayed, showing "New Product Added" with an information icon. At the bottom are buttons for "Add Product", "Edit Product", "Delete Product", and "Refresh Table".

Below the window is a table showing the added product:

ID	Name	Description	Price	Cost	Category	Stock
18	addProductTest	jTextField2	50.55	35.55	searchCatego...	5

4.6.11.2 Test 2

Screenshot of the same Java Swing application window. The form for updating a product is shown:

Name:	updateProductTest
Description:	jTextField2
Price:	50.55
Cost:	35.55
Stock:	5
Category:	searchCategoryTest ▾

Below the form is an "Update" button. A message dialog box titled "Message" is displayed, showing "Product updated." with an information icon. At the bottom are buttons for "Add Product", "Edit Product", "Delete Product", and "Refresh Table".

Below the window is a table showing the updated product:

ID	Name	Description	Price	Cost	Category	Stock
18	updateProdu...	jTextField2	50.55	35.55	searchCatego...	5

4.6.11.3 Test 3

Development and Testing

Search:

ID	Name	Description	Price	Cost	Category	Stock
18	updateProduc...	jTextField2	50.55	35.55	searchCatego...	5

Message



Product has been deleted.

Search:

ID	Name	Description	Price	Cost	Category	Stock

4.6.11.4 Test 4

Search:

ID	Name	Description	Price	Cost	Category	Stock
19	searchProduc...	jTextField2	50.55	35.55	searchCatego...	5
20	testProduct	jTextField2	50.55	35.55	searchCatego...	5

Search:

ID	Name	Description	Price	Cost	Category	Stock
19	searchProduc...	jTextField2	50.55	35.55	searchCatego...	5

4.6.12 Success Criteria

Success Criteria Number	Success Criteria	Achieved	Notes
3	Allow user to add an item with custom name, picture, price, multiple products	Partially	I haven't yet added the feature that allows the user to add multiple products to one item, however the user can successfully add a product with a name, price and other fields.
5	Allow user to add items to categories	Yes	None

4.6.13 Review/Stage Evaluation

In this stage I created the methods to allow the user to add, update and delete products. The methods I used to develop this functionality are very similar to the categories methods as they are practically doing the same thing.

For the first method which was the product creation method, same as the categories section I create a new record in the categories table using the user inputted values.

For the next method which is the data retrieval method, same as the categories section I fill an array list with instances of the products class that have the values of a product record from the database and then in the fill table method I again use the 2d array and the table model to create a model with all the data and then set the table to this model.

For the delete product method I use a deleted flag and implement a soft delete as was done in the categories section.

Lastly, in the update products method same as the categories section I simply use an update query to set each field to the new user inputted field values.

I came across a few errors when developing the update products section, but they were all very minor and quick to fix.

This stage went very well as I was able to re-use a lot of the code from the categories section and I just had to make the appropriate changes to match the products sections requirements. Because of the code re-use this section was very quick to code which should help give me more time for the more complicated features later on such as the validation and sales reports.

I am satisfied with the level at which I achieved the success criteria, one was completely achieved and the other one was partially achieved as I didn't add the functionality to allow the user to add multiple products to an item however I no longer plan on adding this as I want to focus on completing the rest of the code, if I have time later, I will add it.

4.7 Stage 6/Data Validation universal checker

I found when developing my system so far that the validation is needed now. If I enter bad data, it can either crash the Java Swing code or it crashes the database operation with the bad data.

For my data validation I need to always run some code when a user enters data that needs to be validated.

Then I need to know where the data came from in the system as I can't validate it unless I know where it was input and what the data is meant for.

Also, I need to know what the data actually is so I can check if it is valid and fits the data type I am checking.

Then I can decide how to validate it.

I did this bit by bit as it seems a bit complicated to make the code run, then get what the data is, then get how to validate it, then validate it.

I have broken it up into these modules.

1. Find out when the user inputs some data so understand when to validate the data.
2. Work out where the data is coming from and what data type it should be validated against.
3. Validate the data and display suitable error messages.

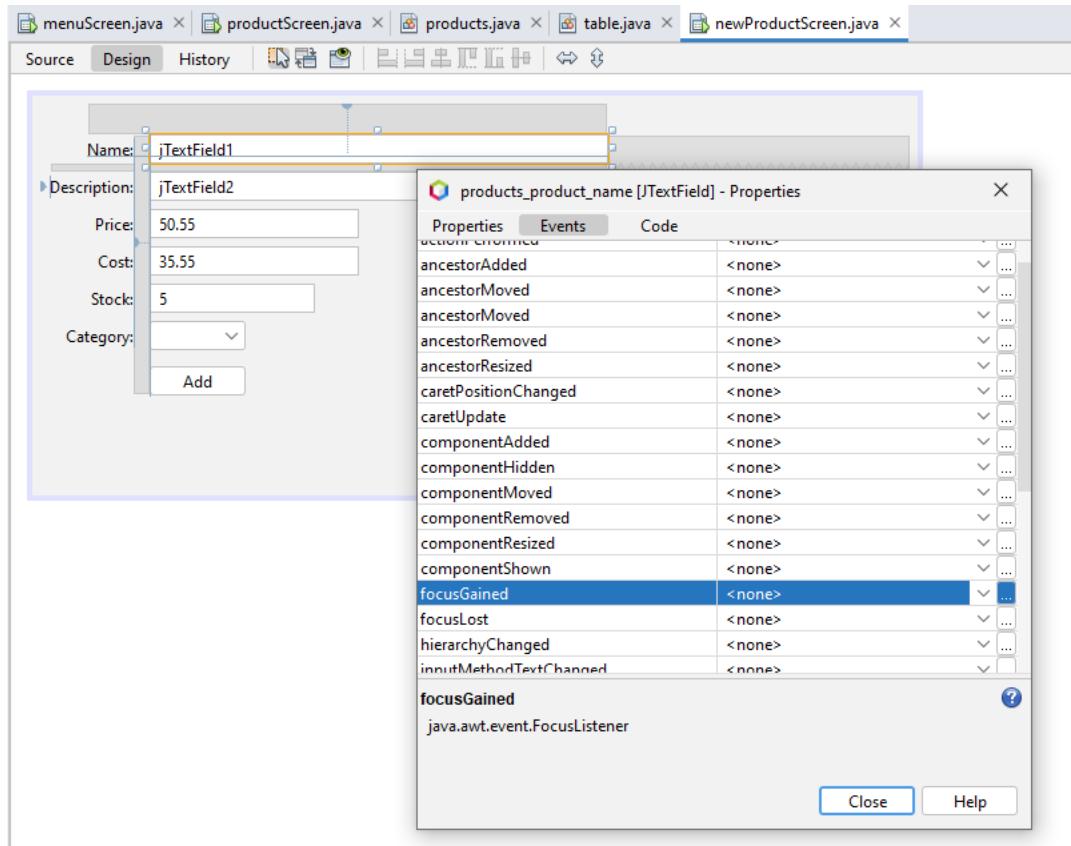
4.7.1 Problems

1. I ran into some problems with my plan to use the JField variable name to determine how to validate the data inputted. I learned that the JField variable name is not actually the name of the object that holds the data that was inputted. It is just a reference to the object. I had to name the objects explicitly in a method so that when I make the error message I can tell the user what data field is incorrect and why.
2. It got too complicated with my design plan to use the name of the JField to work out what the field was and how to validate it. I gave up on trying to get the data type from the database although I did write the MySQL query to get the data types and I could probably have got it working fine.
3. I used lots of if statements in my validation method for each datatype, but I learned that it is more efficient to use a switch and case type syntax to evaluate the data type.
4. Also, I couldn't get the Java string split function to work on my decimal validation until I worked out from the docs how it handles a decimal point. This told me what a regular expression is and later I used this again on the date validation.
5. Finally, I couldn't work out how best to validate a date in the format DD-MM-YYYY so I learned how to do this with a Java String matches (String regular expression) which is really complicated but I worked out how the regular expression works for it and got it working.

I documented these problems and how I fixed them below.

4.7.2 Developing Method To Find When User Inputs Data

I need to work out when to validate some data inputted. I can use an event handler like focusLost in the Swing JTextField or other JComponent to decide when to run my validation code. So when the user moves the focus away from the data input field then my validation would run.

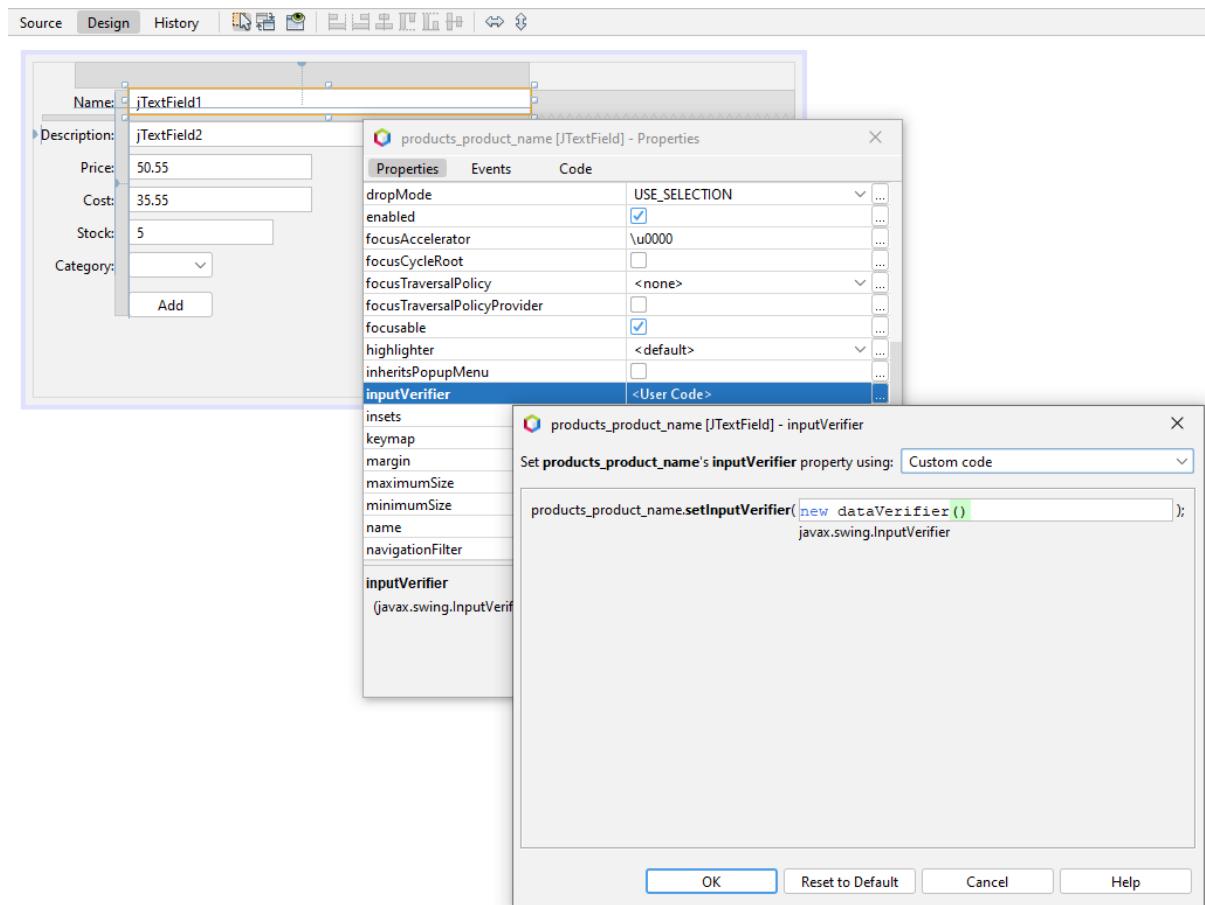


But Swing has a class called javax.swing.InputVerifier designed for validation so I will use that. This class is used to make sure data is the correct format before the user can move to another Swing component like another field.

I have to make a subclass of InputVerifier that's my own validation class and I have to attach it to the JTextField that I want to validate using the UI editor.

So I made a subclass called dataVerifier.

Development and Testing



I named the JTextField like the database tablename_columnname so I should know where the data is headed for so then I know how to validate it. This is the Swing code that links the InputVerifier to my subclass dataVerifier.

```
83
84     products_product_name.setColumns(30);
85     products_product_name.setText("jTextField1");
86     products_product_name.setInputVerifier(new dataVerifier());
87
88     products_product_description.setColumns(30);
89     products_product_description.setText("jTextField2");
90     products_product_description.setInputVerifier(new dataVerifier());
91
92     priceTextField.setColumns(13);
93     priceTextField.setText("50.55");
94
95     costTextField.setColumns(13);
96     costTextField.setText("35.55");
97
98     stockTextField.setColumns(13);
99     stockTextField.setText("5");
100
101    categoryComboBox.setModel(new javax.swing.DefaultComboBoxModel(new String[] {"", "Add"}));
102
103    jScrollPane1.setViewportView(jTable1);
104
105    // End of generated code
106
107    products_product_name.setInputVerifier(new dataVerifier());
```

4.7.3 Developing Method To Get What Field The Data Came From And Data Type

Before I do any validation I needed to find out where the data was typed in and what the data type is so I can develop a method to check if the value entered matches the data type.

I made my new subclass dataVerifier() like this.

```

5  package classes;
6
7  import javax.swing.InputVerifier;
8  import javax.swing.JComponent;
9  import javax.swing.JOptionPane;
10 import javax.swing.JTextField;
11
12 /**
13 *
14 * @author garmin
15 */
16 public class dataVerifier extends InputVerifier
17 {
18     @Override
19     public boolean verify(JComponent input) {
20
21         //the verify method gets the JComponent passed in and this is the text field from the screen that I am validating
22         //this will allow me to see where the data came from and what the data is
23         JTextField componentToValidate=(JTextField)input;
24         String dataIsFrom=componentToValidate.getName();
25         String dataToValidate=componentToValidate.getText();
26
27         //output where the data came from and what it is so I can see it works
28         System.out.println(dataIsFrom);
29         System.out.println(dataToValidate);
30
31         //return true so the validation is always true
32         return true;
33     }
34 }
35

```

So this is a subclass of InputVerifier like they say to do in Swing docs.

I got the value of the text field to output so this is what to validate. And the name of the field where the value is coming from.

The return is Boolean and is from my class dataValidation dataisValid method I pass the name of the field and the data value into it.

```

/*
public class dataVerifier extends InputVerifier
{
    @Override
    public boolean verify(JComponent input) {

        //the verify method gets the JComponent passed in and this is the text field from the screen that I am validating
        //this will allow me to see where the data came from and what the data is
        JTextField componentToValidate=(JTextField)input;
        String dataIsFrom=componentToValidate.getName();
        String dataToValidate=componentToValidate.getText();

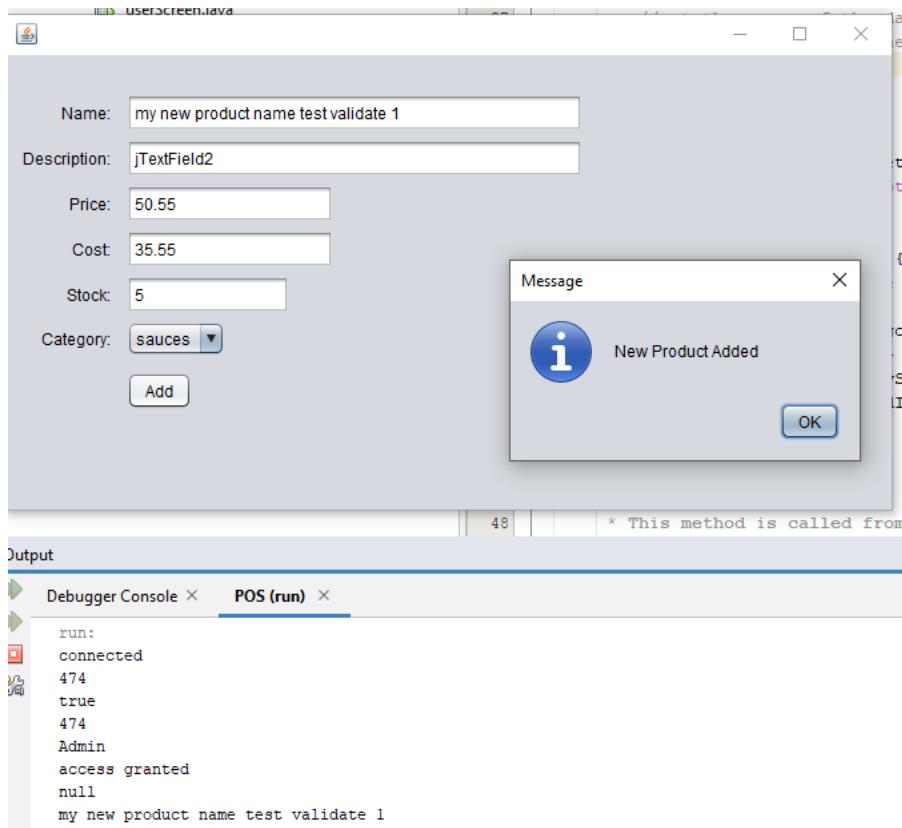
        //output where the data came from and what it is so I can see it works
        System.out.println(dataIsFrom);
        System.out.println(dataToValidate);

        //return true or false depends on my validation of the data in my dataValidation class
        //return dataValidation.dataisValid(dataIsFrom,dataToValidate);
        return classes.dataValidation.dataisValid2(dataIsFrom,dataToValidate);
    }
}

```

4.7.3.1 Problem 1 – no field name

The name of the field where the data was input isn't there when I run it. It just says null. The inputVerifier ran and wrote out the value of the data to be validated but the name of the JField where it came from is null.



So I read up that the name of a variable in Swing is just a reference to an object. It isn't the name of the object so I can't get the name of the text field that was passed as the input into my verifier subclass. I need the name of the object.

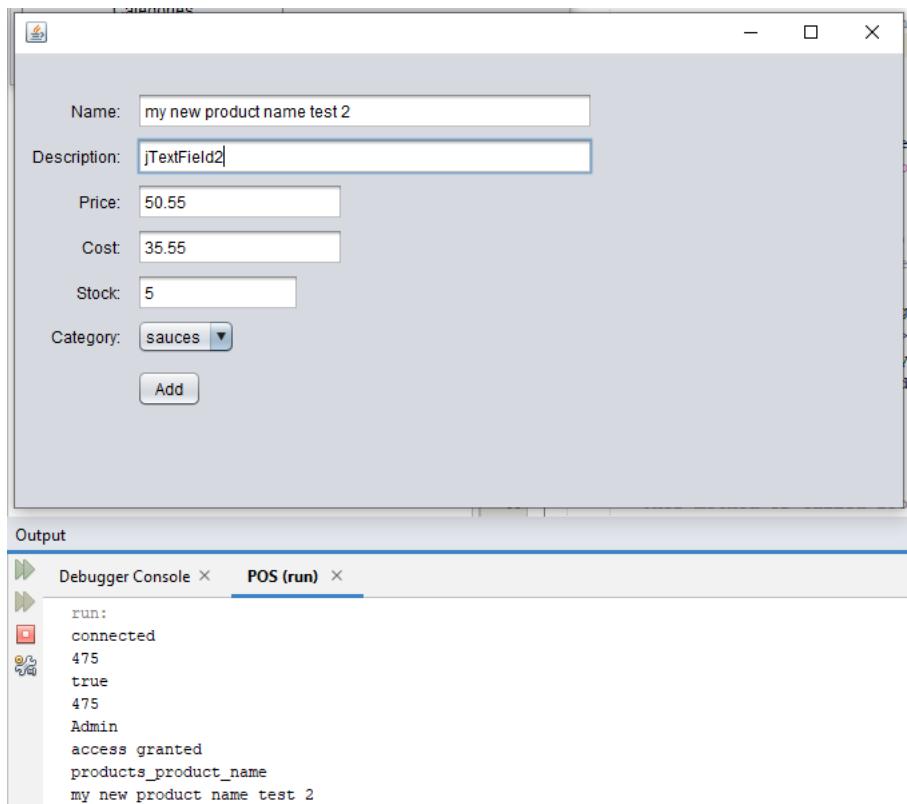
So I need to set the name of the object before it gets passed into my validation method. I need a name for the input so I can tell how to validate the data inside the input which is a JComponent JField.

So when the form does the init() in Swing when I make the newProductScreen() { then I call a new method public void fieldTypeSet(){ which I made. This names the objects that are the variables where the user inputs the data. Even though I named the variables like tablename_columnname I can't get the variable name. But if I set the object name for the object that holds that variable I can get that name in my verifier class and then I know what the data is and how I will validate it.

Development and Testing

```
20
21
22
23     /**
24      * Creates new form newProductScreen
25     */
26
27     public newProductScreen() {
28         initComponents();
29         comboBoxAttach();
30
31         //set the names of the data entry fields so I know how to validate them
32         //because I found out the setName and getName don't work unless you actually set the name of JComponents
33         fieldTypeSet();
34
35     }
36
37     public void fieldTypeSet() {
38         products_product_name.setName("products_product_name");
39         products_product_description.setName("products_product_description");
40
41     public void comboBoxAttach() {
42         //adds categories to the combo box
43
44         classes.categories category = new classes.categories();
45         HashMap<String, Integer> map = category.addToCombo();
46         ...
47     }
48 }
```

Now when I test it then I can see the object names as well as the data input into that object by the user.



So I am going to name the objects in a way like I can use the name to decide how to validate the data like this.

```
//this is my method to set the name of the object holding a data to be validated
//so when my inputVerifier receives the object I can use the name of the object to
//decide how to validate the value in the object
public void fieldTypeSet(){
    products_product_name.setName("st@tic_n0tnull_char_100");
    products_product_description.setName("d@ta_products_product_description");
```

So if I get a name starting st@tic_ I know to validate this not dynamically so not to go and get the data type from the database for that field. In this example the object name tells me it is static, it is not null, it is a string, it has a max length of 100 characters.

Development and Testing

```
//method to validate the data called by the dataVerifier
//I check first if its a database or a static validation using the name of the object
//I named the object like static_ or data_ and then the data type or the database column details after
//like this products_product_name.setName("st@tic_n0tnull_char_100");
public static Boolean dataIsValid(String dataIsFrom, String dataToValidate)
{
    //if the data validate type is static call the static checker st@tic_
    //else call the databasechecker d@ta_
    if (dataIsFrom.contains("st@tic_")) {
        return dataIsValidStatic(dataIsFrom,dataToValidate);
    } else if (dataIsFrom.contains("d@ta_")){
        return dataIsValidDb(dataIsFrom,dataToValidate);
    }

    //if the object name isn't static or database then just return true without validating
    return dataIsValid;
}

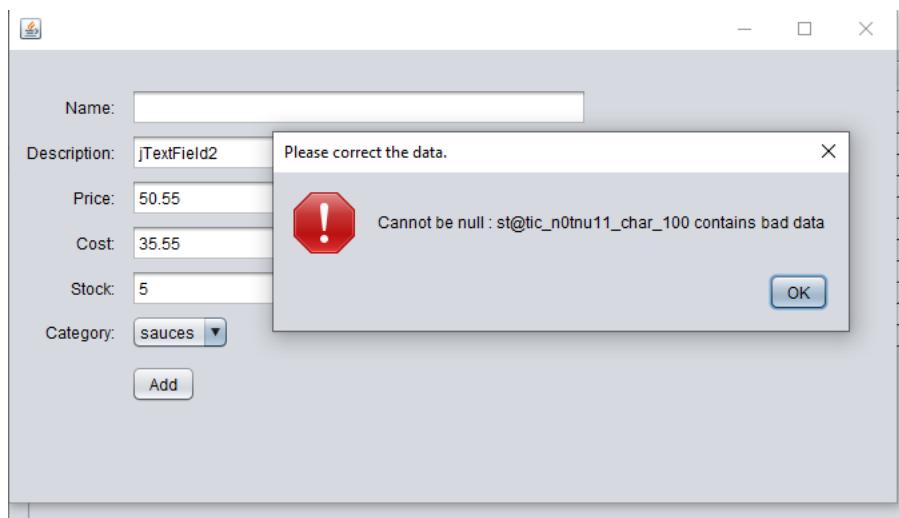
//validate the data in the static way without using database data types
private static Boolean dataIsValidStatic(String dataIsFrom, String dataToValidate)
{
    System.out.println(dataIsFrom);
    System.out.println(dataToValidate);

    //I check if it is allowed to be null first
    //_n0tnull_
    if (dataIsFrom.contains("_n0tnull_")) {
        if (dataToValidate.isBlank()) {
            return dataIsInvalid;
        }
    }

    return true;
}
```

4.7.3.2 Problem 2 – working out the data type from the name I set

It's kind of complicated to work out all the parts of the validation inside the string name of the object. Also if the name starts d@ta_ then I was going to get the data types from the database and it would work but it is complicated. I was going to parse the object name and decide which kind of validation like do it manually from the way the object name is or get the data type from MySQL and validate it based on what the database has.



Development and Testing

```
//validate the data by using database data types
//I didnt finish this
private static Boolean dataIsValidDb(String dataIsFrom, String dataToValidate)
{
    System.out.println(dataIsFrom);
    System.out.println(dataToValidate);
    //I was going to use a query like this and work out the table and column from the dataIsFrom
    //to get the data type and then use that to validate the field
    //but its quite complicated however I think it would work OK if I finished it
    //SELECT
    //table_name,
    //column_name,
    //data_type,
    //character_maximum_length,
    //numeric_precision,
    //numeric_scale,
    //is_nullable
    //FROM information_schema.columns
    //WHERE table_schema="posplex"
    //AND
    //table_name="products"
    //ORDER BY table_name, ordinal_position asc;
    //products      product_pk      int          10      0      NO
    //products      product_name     varchar(100)   NO
    //products      product_description  varchar(255) NO
    //products      product_sellprice decimal(13,2)  13      2      NO
    //products      product_cost      decimal(13,2)  13      2      NO
    //products      createdby_fk     int          10      0      NO
    //products      createdon        datetime     NO
    //products      lastupdated      datetime     NO
    //products      deletedflag      tinyint     3       0      YES
    //products      category         varchar(100) YES
    //products      stock            int          10      0      NO
    //products      category_fk     int          10      0      NO

    return dataIsValid;
}
```

So I decided to do the validation using a switch case statement to check what the data type is and then to validate the data. I will name the JField objects with the same name as on the screen and use this name in the invalid error message so the user can see where the bad data is inputted.

```
//this is my method to set the name of the object holding a data to be validated
//so when my inputVerifier receives the object I can use the name of the object to
//decide how to validate the value in the object
public void fieldTypeSet()
{
    products_product_name.setName("Product Name:");
    products_product_description.setName("Product Description:");
    priceTextField.setName("Price:");
    costTextField.setName("Cost:");
    stockTextField.setName("Stock:");
}
```

I set some variables up in my class to hold the data type details then set these when the JField in the screen gets the focus.

Development and Testing

```
//this is my variables that I set from the JField focus so each time
//the system knows what datatype to validate for the field
public static String myDataType = "notset";
public static int myPrecision = 0;
public static int myScale = 0;
public static int myLength = 0;
public static boolean myNull=false;

//methods to be called when a JTextField gets focus so the inputVerifier can use it to validate the JTextField.value
//these set the data type variables
public static void typeIsDecimal(int inPrecision, int inScale)
{
    myDataType="Decimal";
    myPrecision=inPrecision;
    myScale=inScale;
}
public static void typeIsInteger(int inLength)
{
    myDataType="Integer";
    myLength=inLength;
}
public static void typeIsString(int inLength)
{
    myDataType="String";
    myLength=inLength;
}
public static void typeIsDate()
{
    myDataType="Date";
}
public static void typeIsNull(boolean inNull)
{
    myNull=inNull;
}
```

This sets the data types when the fields get the focus.

```
private void stockTextFieldFocusGained(java.awt.event.FocusEvent evt) {
    classes.dataValidation.typeIsInteger(10);
    classes.dataValidation.typeIsNull(false);
}

private void priceTextFieldFocusGained(java.awt.event.FocusEvent evt) {
    classes.dataValidation.typeIsDecimal(10,2);
    classes.dataValidation.typeIsNull(false);
}

private void products_product_nameFocusGained(java.awt.event.FocusEvent evt) {
    classes.dataValidation.typeIsString(100);
    //classes.dataValidation.typeIsDate();
    classes.dataValidation.typeIsNull(false);
}

private void costTextFieldFocusGained(java.awt.event.FocusEvent evt) {
    classes.dataValidation.typeIsDecimal(10,2);
    classes.dataValidation.typeIsNull(false);
}

private void products_product_descriptionFocusGained(java.awt.event.FocusEvent evt) {
    classes.dataValidation.typeIsString(255);
    classes.dataValidation.typeIsNull(false);
}
```

4.7.4 Developing Method To Validate Data

So when the input verifier calls my dataIsValid2 method then the method uses the data types to check the data that was input like this. There is a dataIsValid method but it was from when I was trying to use the object name I set to decide how to validate the data so I made a new one dataIsValid2.

The focusgained event for the JField to be validated has set the data type variables in the dataValidation class so I know what I am validating and if it can be null or a character or a number.

```
//this is my variables that I set from the JField focus so each time
//the system knows what datatype to validate for the field
public static String myDataType = "notset";
public static int myPrecision = 0;
public static int myScale = 0;
public static int myLength = 0;
public static boolean myNull=false;
```

```
private void stockTextFieldFocusGained(java.awt.event.FocusEvent evt) {
    classes.dataValidation.typeIsInteger(6);
    classes.dataValidation.typeIsNull(false);
}
```

4.7.4.1 Problem 3 – switch not if

Originally I wrote the dataIsValid method using 4 if statements for Character, Integer, Decimal and Date and with an if statement for if null is allowed. But I read that all the if statements will be evaluated when actually myDataType will only be true for one of the if statements. So it is more efficient to use a switch which only evaluates the code which is true and then stops with the break; command rather than test the data type for all types when it will only ever be one type.

So now I use a switch statement.

```
public static Boolean dataIsValid2(String dataIsFrom, String dataToValidate)
{
    boolean validStatus=false;
    //first I check if the data can be null if its null I send an error to stop
    if (dataToValidate.isBlank() & myNull==false) {
        validStatus=false;
        validationErrorMessage( dataIsFrom, dataToValidate, "Cannot be null");
    }
    else //if it passed the null test now I test the data type
    {
        //case to check what the datatype is set by the Jfield event focus
        switch (myDataType)
        {
            case "Decimal":
                String[] arrayDigits = dataToValidate.split("\\.", 2);
                System.out.println(arrayDigits[0]);
                System.out.println(arrayDigits[1]);
                if (String.valueOf(arrayDigits[0]).length() > myPrecision||String.valueOf(arrayDigits[1]).length() > myScale) {
                    validStatus=false;
                }
        }
    }
}
```

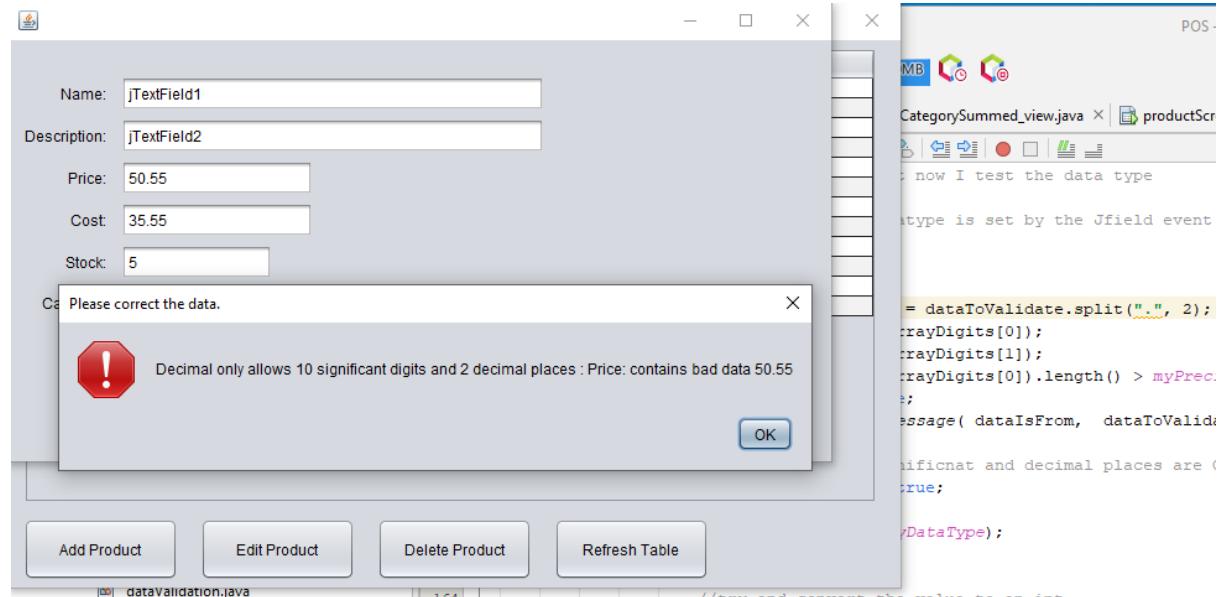
4.7.4.2 Problem 4 – trying to split the decimal number

I used the split Java function to divide the decimal data which is in a string from the JField object so I can see what the significant digits are before the decimal point and the decimal digits after the decimal place. So I split the decimal data to get the significant digits and the decimal digits to check the length of each.

```

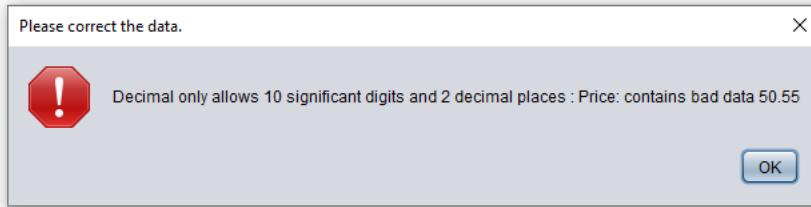
public static Boolean dataIsValid2(String dataIsFrom, String dataToValidate)
{
    boolean validStatus=false;
    //first I check if the data can be null if its null I send an error to stop
    if (dataToValidate.isBlank() & myNull==false) {
        validStatus=false;
        validationErrorMessage( dataIsFrom, dataToValidate, "Cannot be null");
    }
    else //if it passed the null test now I test the data type
    {
        //case to check what the datatype is set by the Jfield event focus
        switch (myDataType)
        {
            case "Decimal":
                String[] arrayDigits = dataToValidate.split(".", 2);
                System.out.println(arrayDigits[0]);
                System.out.println(arrayDigits[1]);
                if (String.valueOf(arrayDigits[0]).length() > myPrecision||String.valueOf(arrayDigits[1]).length() > myScale) {
                    validStatus=false;
                    validationErrorMessage( dataIsFrom, dataToValidate, "Decimal only allows "+myPrecision+" significant digits and "+myScale+" decimal places");
                } else {
                    //if the significant and decimal places are OK then return true for valid
                    validStatus=true;
                }
                System.out.println(myDataType);
                break;
            case "Integer":
                //try and convert the value to an int
                //if it fails send an error and return false to say its invalid
                try {
                    int i = Integer.parseInt(dataToValidate);
                    //if it is an int then see how long and if longer than the length set for that field send an error
                    if (String.valueOf(dataToValidate).length() > myLength){
                        validStatus=false;
                        validationErrorMessage( dataIsFrom, dataToValidate, "Integer is too long");
                    } else {
                        //if the int and the length are ok then return true for valid
                        validStatus=true;
                    }
                } catch (NumberFormatException ex) {
                    validStatus=false;
                    validationErrorMessage( dataIsFrom, dataToValidate, "Is not a valid integer");
                }
                System.out.println(myDataType);
                break;
            case "String":
        }
    }
}

```



Development and Testing

```
run:  
connected  
512  
true  
512  
Admin  
access granted  
Product Name:  
jTextField1  
String  
Price:  
50.55  
0.55
```

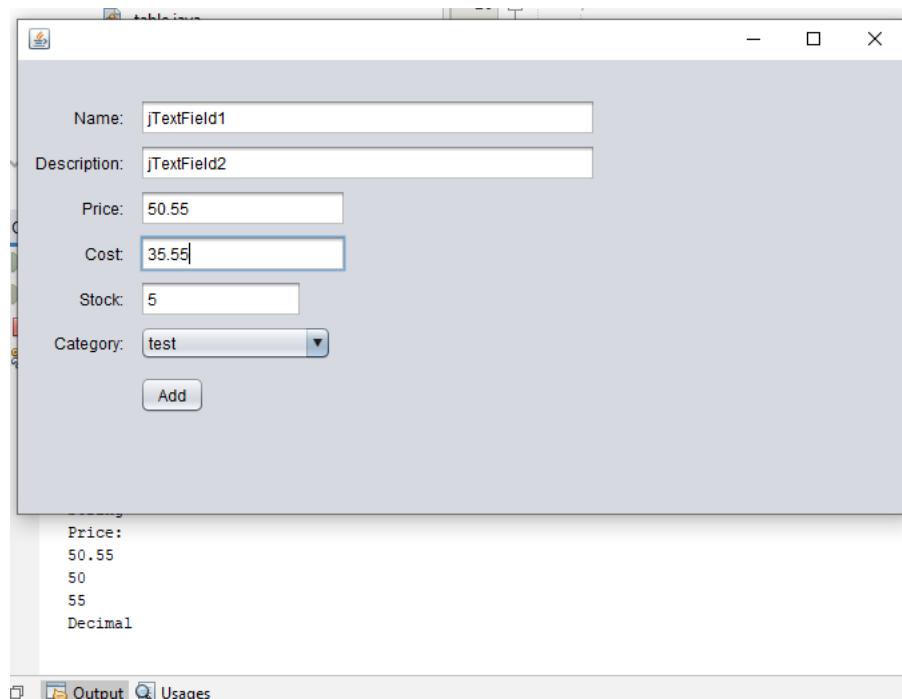


50.55 is a valid decimal data value. The `println` should have shown 50 and then 55 so I can get the length of each so it is valid. But it outputs a null empty for the significant digits and a 0.55 for the decimal part.

I read that the “.” In the `split` function is a regular expression and the `.` character is special so I have to escape it. This means put two `\\"` in front of the `.` so the `split` method uses the `.` to decide where to split the string.

```
case "Decimal":  
    String[] arrayDigits = dataToValidate.split("\\.", 2);  
    System.out.println(arrayDigits[0]);  
    System.out.println(arrayDigits[1]);  
    if (String.valueOf(arrayDigits[0]).length() > myPrecision || String.valueOf(arrayDigits[1]).length() > myScale) {  
        validStatus=false;  
        validationErrorMessage( dataIsFrom, dataToValidate, "Decimal only allows "+myPrecision+" significant digits and "+myScale+" decimal places");  
    } else {  
        //if the significant and decimal places are OK then return true for valid  
        validStatus=true;  
    }  
    System.out.println(myDataType);  
    break;
```

Now it works. I can see the significant digits so I can check the length as well as the decimal digits.



4.7.4.3 Problem 5 – how to validate the date data

I needed a Date validation too but haven't tried it yet so will try it on a test screen. It uses a regular expression for the date format of DD-MM-YYYY so the Date inputted has to match that format.

Regular expressions are really complicated. A regular expression is a pattern to describe a piece of text so how that text is formatted what characters are allowed and in what place in the string of text. Java has a regular expression method called matches(String regex) on the String object so I used this to match my regular expression against the dataToValidate string.

```

case "Date":
    //if its a date use a regular expression to see if its DD-MM-YYYY format else invalid
    String regex="(0[1-9]|1[0-9]|3[01])[- /.](0[1-9]|1[012])[- /.](19|20)\d\d";
    if (!(dataToValidate.matches(regex))) {
        validStatus=false;
        validationErrorMessage( dataIsFrom,  dataToValidate, "Not a valid date DD-MM-YYYY");
    } else {
        //if the date is ok then return true for valid
        validStatus=true;
    }
    System.out.println(myDataType);
    validStatus=true;
    break;
default:
    validStatus=true;

```

If the date data inputted matches the regular expression then the data is a valid date.

The first part of the regex string consists of three options. The first option (0[1-9] |) matches the numbers 01 up to 09, the second option [12][0-9] | 10 up 29, and the third option 3[01]) matches either 30 or 31. So using the first or the second or the third options separated by | has to match the days DD part of the text inputted else the validStatus is set to false.

The separator [- /.] has to match a – or a / or a . then the second part of the regex string is for the MM part of the date. It can be 01 up to 09 or 10 or 11 or 12 so it is (0[1-9]|1[012]). The another separator like this [- /.]

Finally the third part of the regex string has to start with either 19 or 20 so 1900 or 2000 and then any single digit 0 to 9 after 19 or 20 so 199 and another single digit after that so 1999 or 2023 so it looks like this (19|20)\d\d

The 19 and the 20 are in brackets and with an | Or in the middle so it forces it to be either 19 or 20 to start the century. This is called alternation in regular expression checking.

public boolean matches(String regex)

From the docs - this method tells whether or not this string matches the given regular expression. An invocation of this method of the form str.matches(regex) yields exactly the same result as the expression Pattern.matches(regex, str).

4.7.5 Test Plan

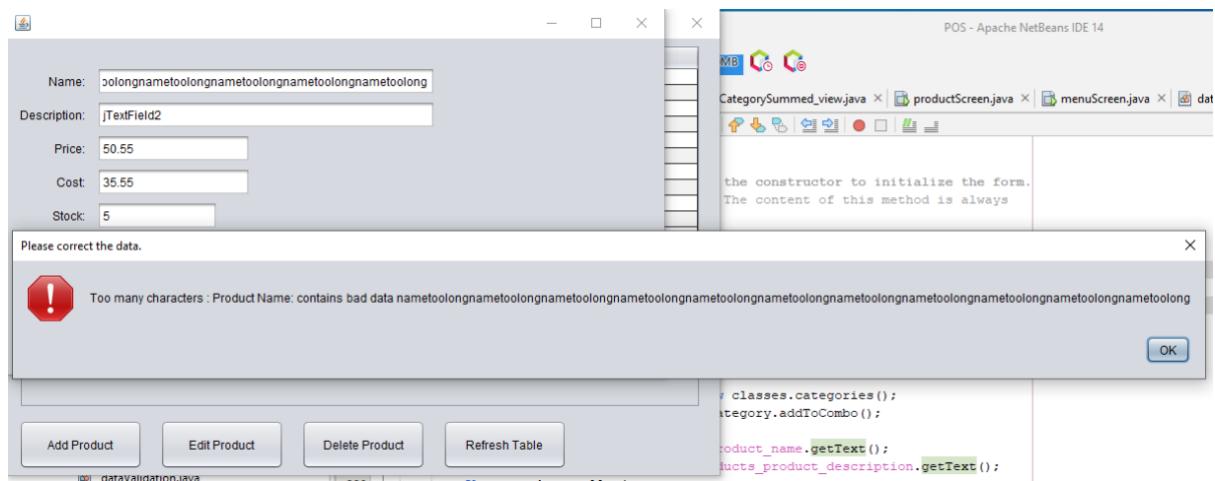
I tested the validation routine on my Product screen using the AddProduct button.

Test Number	How To Test	Expected Result	Actual Result
1	Attempt to add a product with a name > 100 characters	Error too many characters	The error was shown
2	Attempt to input a Price that is longer than 10 numbers and 2 decimal places	Error should show the Price field has too many digits	The error was shown
3	Attempt to input a stock that is too long for the integer length	Error should show the Stock field has too many digits	The error was shown
4	Put a character in a numeric field like Stock	Error should show the Stock field is not a number	The error was shown
5	Attempt to add a product with a name that is empty	Error should show the Name field is cannot be null	The error was shown
6	Try a date validation make a text Jfield date format and test	Error should show the Name field is not a date just for a test	The error was shown
7	Try to add a product with all the fields with valid values	No error and add product	The product added was shown

4.7.6 Evidence

4.7.6.1 Test 1 String length too long

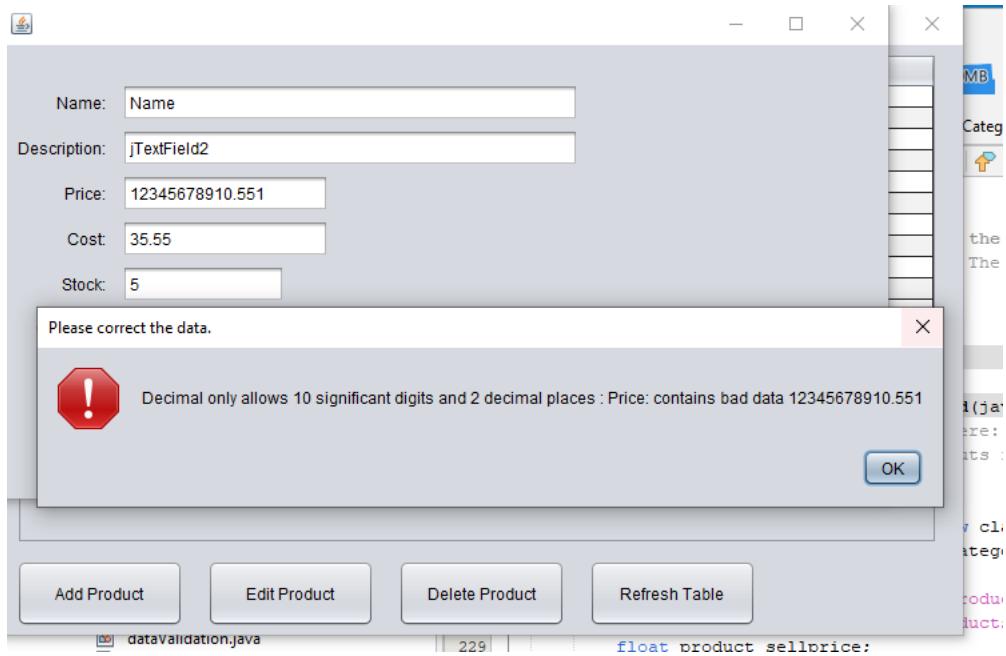
Name field has more than 100 characters.



Development and Testing

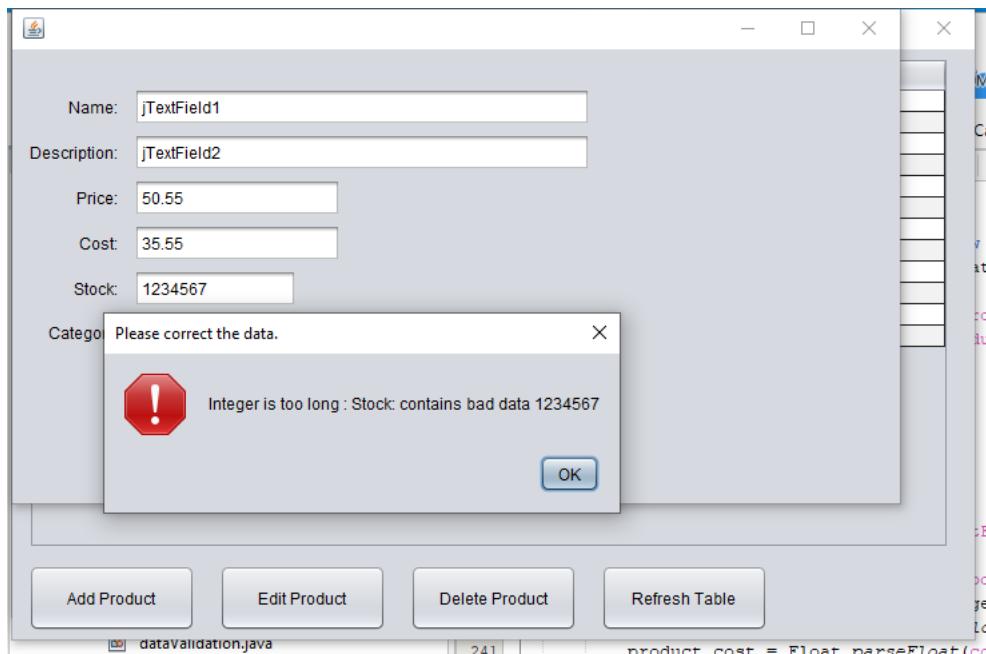
4.7.6.2 Test 2 Decimal length not 10 significant 2 decimal

Decimal has 3 decimal digits.



4.7.6.3 Test 3 Integer length > 6 digits

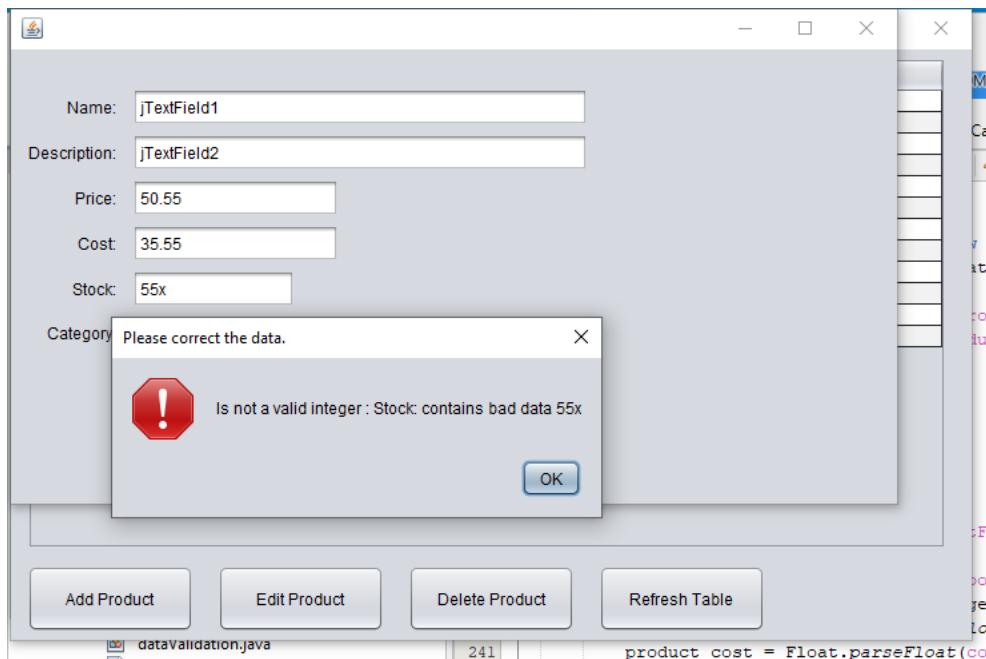
7 digits in a 6 digit integer field.



Development and Testing

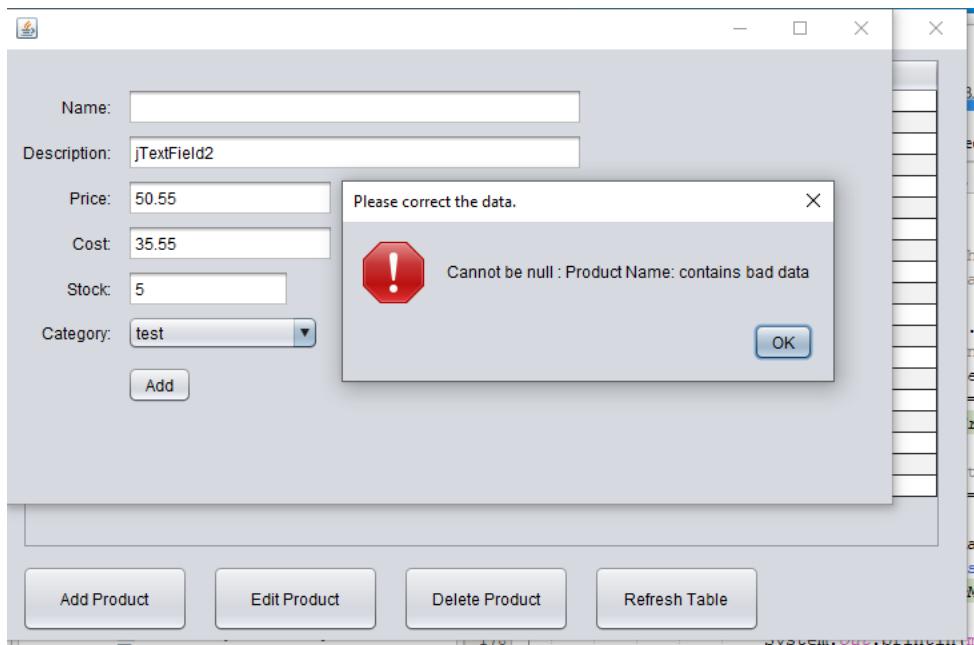
4.7.6.4 Test 4 Integer contains not number characters

An "x" in the Stock integer field.



4.7.6.5 Test 5 Not null field Name: is empty

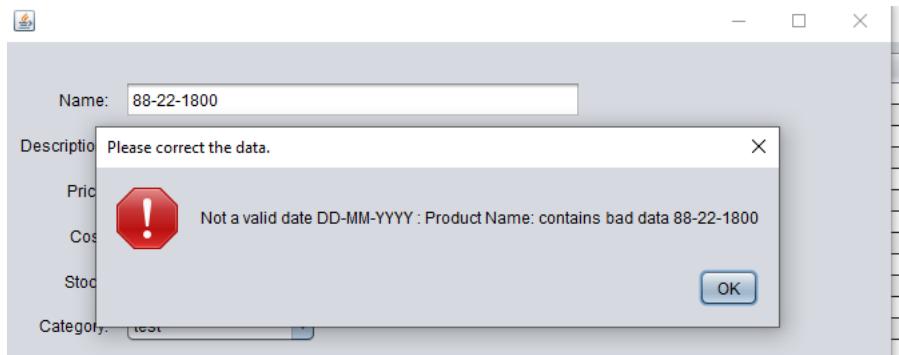
Empty Name field.



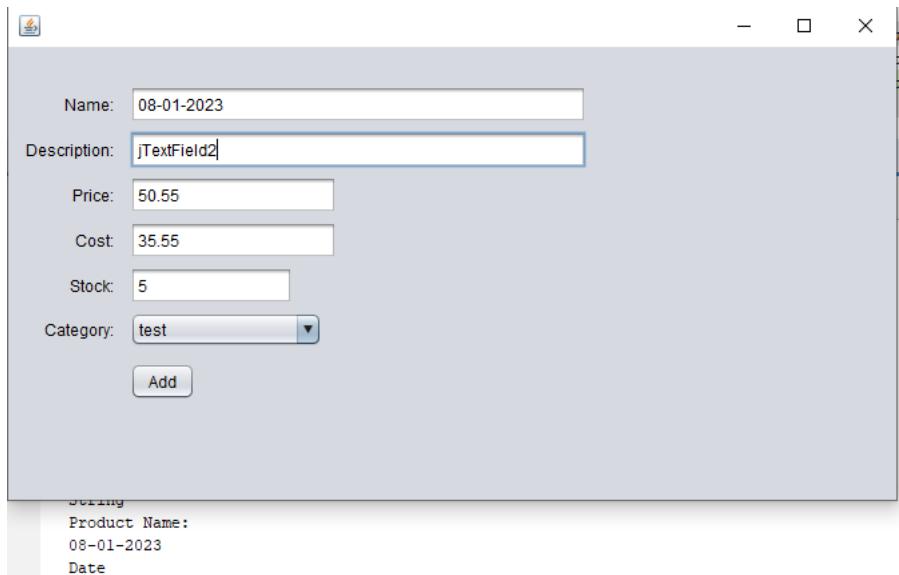
4.7.6.6 Test 6 Date field Name: is not a date

For the test I made the Name field work as a date type.

```
private void products_product_nameFocusGained(java.awt.event.FocusEvent evt) {  
    //classes.dataValidation.typeIsString(100);  
    classes.dataValidation.typeIsDate();  
    classes.dataValidation.typeIsNull(false);  
}
```

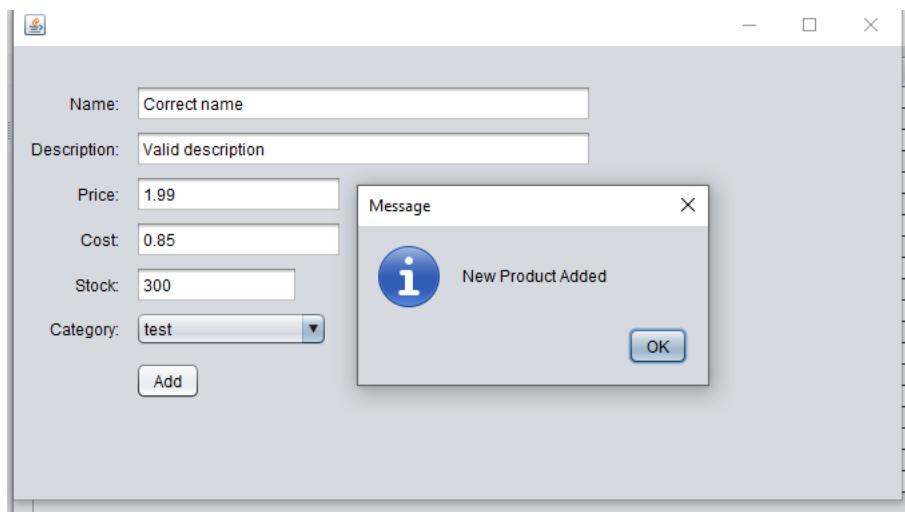


I changed it to 08-01-2023 and it passed the check.

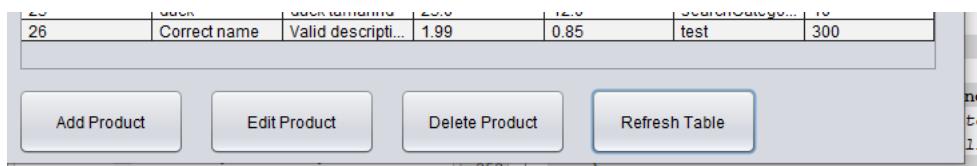


4.7.6.7 Test 7 All product data is valid, add the product

With correct data.



Product was added.



4.7.7 Success Criteria for universal requirements like good data

Success Criteria Number	Success Criteria	Achieved	Notes
UR1	Notify the user if they try to enter an invalid integer	Yes	The user can't enter a character in a integer or a integer that is too long or that is empty if it is a not null field
UR2	Notify the user if they try to enter an invalid decimal	Yes	The user can't enter a character in a decimal or a decimal that is too long or too many decimal places or that is empty if it is a not null field
UR3	Notify the user if they try to enter an invalid character field	Yes	The user can't enter too long a character value like 250 characters in a character (100) field or no characters in a not null field
UR4	Notify the user if they try to enter an invalid date field	Yes	The user can't enter a date that is not DD-MM-YYYY

4.7.8 Review/Stage Evaluation

In this stage I developed the verification functionality behind the products section. Initially my plan was to use this section to validate the categories section and the admin section so the user couldn't add bad data however in those sections each input field can be anything as they are names or passwords so the user can use any characters. The only way to break the code in those sections with user inputs is to add a category or user with too many characters, however since I don't have much time to adapt the validation classes to work for those sections I will later add simple validation such as an if statement to check if the inputted data is too long and return an error if it is.

This stage took quite a long time due to many problems however it does work as intended. First, I made a subclass of the InputVerifier class and had my class called whenever the user inputs data into a field. The method I created called verify takes the data that has been inputted and the name of the component where the data is coming from and passes those values into the dataisValid method from my data validation class, verify() returns the value of dataisValid which will be true or false depending on whether the data is valid or not. When testing this method, I came across many errors most of which took some time to solve however after fixing the errors my validation section works fine.

This stage was okay, it took a lot of time which means I will have less time to finish of the rest of the code however it successfully accomplished all the universal requirement success criteria so the user cannot input invalid data for the products section.

4.8 Stage 7/POS

4.8.1 Updating The Products Section

4.8.1.1 Updating Data Retrieval Code

There are two key parts to this section, one is the products part and the other is the order part. First, I will code the products part in which I will be displaying all products with stock over 0 in a products table for the user to select and add to the order.

When developing this section, I will first develop the code for the products table then I will update the code later on so that it displays only products that are from the user selected category. I plan to use a combo box to allow the user to select the category.

Rather than re-writing the product code in the pos class to retrieve all the products with stock over 0 I am going to add another parameter to the method called stock so in the products section it will retrieve all the product with stock over or equal to 0 however in the pos section it will retrieve only the product with stock over 0.

```
public ArrayList<products> productList(String searchVal, int stock){
    //creates an arraylist
    //creates a new instance of "products" (as shown in method earlier) with data from the products table in the database
    //adds the new product to the arraylist

    ArrayList<products> prodList = new ArrayList<>();

    connection = database.getConnection();
    PreparedStatement ps;
    ResultSet rs;

    String query = "SELECT * FROM products WHERE deletedflag = 0 AND product_name LIKE ? AND stock > ?;";

    try{
        ps = connection.prepareStatement(query);
        ps.setString(1, "%" + searchVal + "%");
        ps.setInt(2, stock);
        rs = ps.executeQuery();
        products product;
        while (rs.next()){
            product = new products(rs.getInt("product_pk"), rs.getString("product_name"), rs.getString("product_descript
                prodList.add(product);
            }
        }
        catch (SQLException ex){
            Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
        }
        return prodList;
    }
```

These are the first changes I made to the productList method.

I added the stock parameter of type integer.

I updated the query so it only retrieves product with a stock > ?.

I set the placeholder to the stock variable using the setInt() method.

By using the greater than sign I can update the products section so that it passes in -1 for the value of stock so all product with a stock of 0 and over will be retrieved and for the pos section I can pass in a value of 0 so all products with a value of 0 are retrieved.

4.8.1.2 Updating Fill jTable Code

Next I have to update the fillTable() method as this method also needs to take stock as a parameter.

The code is below.

```
public void fillTable(String searchVal, int stock) {
    //fills the jTable with contents of arrayList

    classes.products prod = new classes.products();
    ArrayList <classes.products> prodList = prod.productList(searchVal, -1);
    String [] columnNames = {"ID", "Name", "Description", "Price", "Cost", "Category", "Stock"};
    Object [][] rows = new Object[prodList.size()][7];

    for(int i = 0; i<prodList.size(); i++){
        rows[i][0] = prodList.get(i).getProductPK();
        rows[i][1] = prodList.get(i).getName();
        rows[i][2] = prodList.get(i).getDescription();
        rows[i][3] = prodList.get(i).getPrice();
        rows[i][4] = prodList.get(i).getCost();
        rows[i][5] = prodList.get(i).getCat();
        rows[i][6] = prodList.get(i).getStock();
    }

    classes.table model = new classes.table(rows, columnNames);
    productTable.setModel(model);
}
```

Firstly, I added the integer stock variable as a parameter.

Then I had to update the line that calls the productList method and add -1 to the correct parameter position so it takes -1 for the value of stock in the products screen.

Next I had to also update all the locations where the fillTable() method was called as seen below:

```
public productScreen() {
    initComponents();
    fillTable("", -1);
    setColumnNames();
}

private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    fillTable("", -1);
    setColumnNames();
}

private void searchButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    fillTable(searchTextField.getText(), -1);
    setColumnNames();
}
```

Development and Testing

I tested the products section and the results are below:

Search: <input type="text"/>		SEARCH				
ID	Name	Description	Price	Cost	Category	Stock
19	searchProduc...	jTextField2	50.55	35.55	searchCatego...	0
20	testProduct	jTextField2	50.55	35.55	searchCatego...	5
21	jTextField1	jTextField2	50.55	35.55	searchCatego...	5

As you can see it retrieved product with a stock of 0 so the method works for the product section.

4.8.2 Developing POS Screen Fill Product Table Code

This code will be very similar to the products section fillTable() method however the parameter values will be different.

Below is the fillTable() method in the posScreen class:

```
public void fillProductsTable(String searchVal, int stock){
//fills the jTable with contents of arrayList

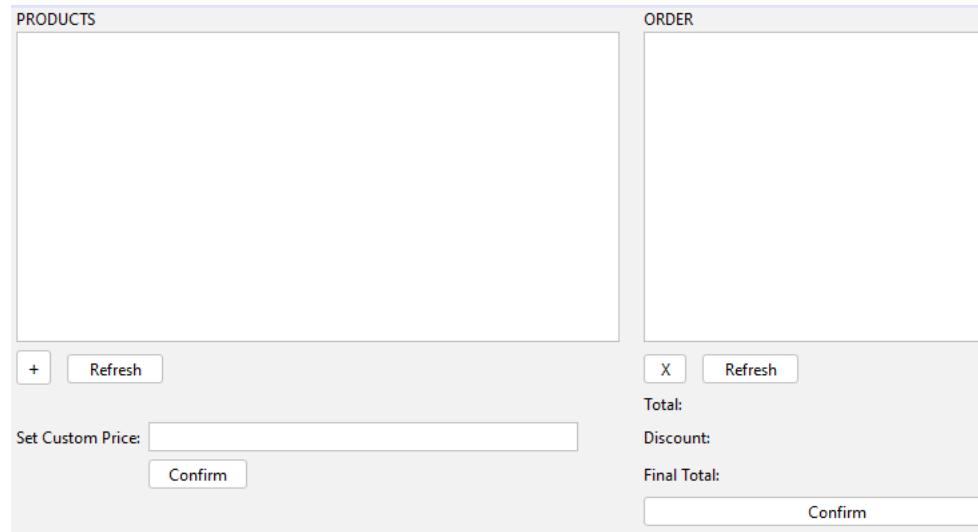
classes.products prod = new classes.products();
ArrayList <classes.products> prodList = prod.productList("", 0);
String [] columnNames = {"ID","Name","Description","Price","Cost","Category","Stock"};
Object [][] rows = new Object[prodList.size()][7];

for(int i = 0; i<prodList.size(); i++){
    rows[i][0] = prodList.get(i).getProductPK();
    rows[i][1] = prodList.get(i).getName();
    rows[i][2] = prodList.get(i).getDescription();
    rows[i][3] = prodList.get(i).getPrice();
    rows[i][4] = prodList.get(i).getCost();
    rows[i][5] = prodList.get(i).getCat();
    rows[i][6] = prodList.get(i).getStock();
}

classes.table model = new classes.table(rows, columnNames);
productTable.setModel(model);
}
```

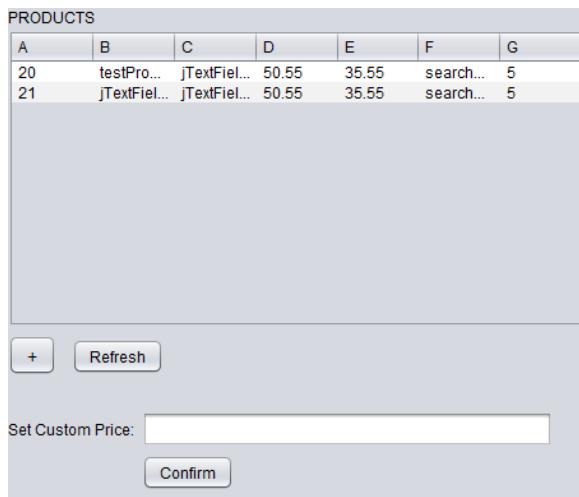
I set the value of searchVal to "" in the productList method calling and the value of stock to 0 so that the productList method retrieves all the products irrelevant of the name, that have a stock over 0.

I also realised I would need refresh buttons for each table so the new posScreen is below.



Development and Testing

This method works, as seen below:



4.8.3 Developing Column Name Method

Next, I had to develop the code to set the column Names, this method will be used for both tables however here I will only document the code relevant to the products table.

```
public void setColumnNames() {
    //sets the jTable column names

    productTable.getColumnModel().getColumn(0).setHeaderValue("ID");
    productTable.getColumnModel().getColumn(1).setHeaderValue("Name");
    productTable.getColumnModel().getColumn(2).setHeaderValue("Description");
    productTable.getColumnModel().getColumn(3).setHeaderValue("Price");
    productTable.getColumnModel().getColumn(4).setHeaderValue("Cost");
    productTable.getColumnModel().getColumn(5).setHeaderValue("Category");
    productTable.getColumnModel().getColumn(6).setHeaderValue("Stock");
}
```

Next I had to call the method in the class constructor as seen below:

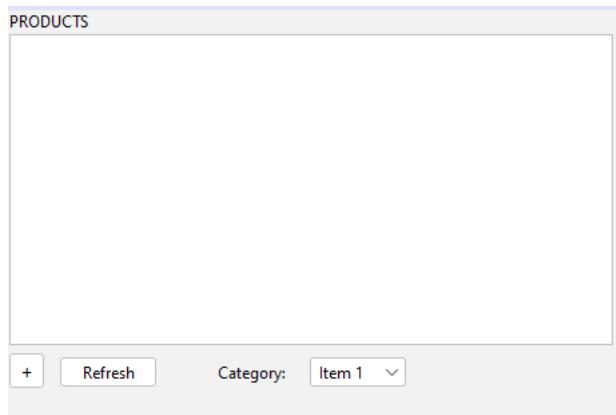
```
public posScreen() {
    initComponents();
    fillProductsTable("", 0);
    setColumnNames();
}
```

This method works as only products that have stock over 0 were retrieved.



4.8.4 Adding Category Selection Code

First I need to add a jComboBox to the jframe as seen below:



Next I had to add the comboBoxAttach() method from the products section as seen below:

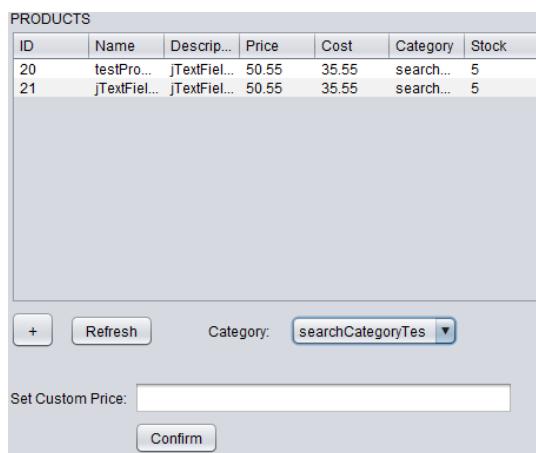
```
public void comboBoxAttach() {
    //adds categories to the combo box

    classes.categories category = new classes.categories();
    HashMap<String, Integer> map = category.addToCombo();
    for (String set: map.keySet()){
        categoryComboBox.addItem(set);
    }
}
```

Then I had to call this method in the constructor for the pos class:

```
public posScreen() {
    initComponents();
    fillProductsTable("",0);
    setColumnNames();
    comboBoxAttach();
}
```

This allows the user to select a category from the combo box as seen below:



Development and Testing

The next thing I plan to do is to add another parameter to the product retrieval code, similar to what I did with the stock, I will add a category parameter so only the user selected category will be retrieved rather than products from all the categories.

The code is below:

```
public ArrayList<products> productList(String searchVal, int stock, int catfk){  
    //creates an arraylist  
    //creates a new instance of "products" (as shown in method earlier) with data from the products table in the database  
    //adds the new product to the arrayList  
  
    ArrayList<products> prodList = new ArrayList<>();  
  
    connection = database.getConnection();  
    PreparedStatement ps;  
    ResultSet rs;  
  
    String query = "SELECT * FROM products WHERE deletedflag = 0 AND product_name LIKE ? AND stock > ? AND category_fk > ?";  
  
    try{  
        ps = connection.prepareStatement(query);  
        ps.setString(1, "%" + searchVal + "%");  
        ps.setInt(2, stock);  
        ps.setInt(3, catfk);  
        rs = ps.executeQuery();  
        products product;  
        while (rs.next()) {  
            product = new products(rs.getInt("product_pk"), rs.getString("product_name"), rs.getString("product_description"));  
            prodList.add(product);  
        }  
    } catch (SQLException ex){  
        Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);  
    }  
    return prodList;
```

I have added the integer parameter catfk which will hold the foreign key of the user selected category.

I also updated the query and added “category_fk > ?”.

And then set the placeholder to catfk.

By doing it this way, in the products section I can set catfk to 0 so it will retrieve all the products where their category_fk field is over 0 which is all the products, and in the pos products table I can set it to the retrieved category_fk from the user selected category in the combo box so it only retrieves the products from that category.

Next I updated the fill table method in the productScreen class as seen below.

```
public void fillTable(String searchVal, int stock, int catfk){  
    //fills the jTable with contents of arrayList  
  
    classes.products prod = new classes.products();  
    ArrayList <classes.products> prodList = prod.productList(searchVal, -1, 0);  
    String [] columnNames = {"ID", "Name", "Description", "Price", "Cost", "Category", "Stock"};  
    Object [][] rows = new Object[prodList.size()][7];  
  
    for(int i = 0; i<prodList.size(); i++){  
        rows[i][0] = prodList.get(i).getProductPK();  
        rows[i][1] = prodList.get(i).getName();  
        rows[i][2] = prodList.get(i).getDescription();  
        rows[i][3] = prodList.get(i).getPrice();  
        rows[i][4] = prodList.get(i).getCost();  
        rows[i][5] = prodList.get(i).getCat();  
        rows[i][6] = prodList.get(i).getStock();  
    }  
  
    classes.table model = new classes.table(rows, columnNames);  
    productTable.setModel(model);  
}
```

Development and Testing

Here I have added the parameter catfk again.

Also I have set the catfk parameter for the productList method to 0 so all products with a category foreign key over 0 are retrieved which is all products.

I also have to update all the places where the fill table method is called:

```
public productScreen() {
    initComponents();
    fillTable("", -1, 0);
    setColumnNames();
}

private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    fillTable("", -1, 0);
    setColumnNames();
}

private void searchButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    fillTable(searchTextField.getText(), -1, 0);
    setColumnNames();
}
```

Next I have to update the posScreen class.

Firstly I set the catfk position parameter to 0 when I first call the fillProductTable() method in the constructor as seen below:

```
public posScreen() {
    initComponents();
    fillProductsTable("", 0, 0);
    setColumnNames();
    comboBoxAttach();
}
```

I do this so that when the user first opens the pos screen all the products are displayed.

Next I have to update the fillProductsTable method so it takes the users input and retrieves the category foreign key and passes that into the productList() method.

```
public void fillProductsTable(String searchVal, int stock, int catfk){
    //fills the jTable with contents of arrayList

    classes.categories category = new classes.categories();
    HashMap<String, Integer> map = category.addToCombo();
    catfk = map.get(categoryComboBox.getSelectedItem().toString());

    classes.products prod = new classes.products();
    ArrayList <classes.products> prodList = prod.productList("", 0, catfk);

    String [] columnNames = {"ID","Name","Description","Price","Cost","Category","Stock"};
    Object [][] rows = new Object[prodList.size()][7];

    for(int i = 0; i<prodList.size(); i++){
        rows[i][0] = prodList.get(i).getProductPK();
        rows[i][1] = prodList.get(i).getName();
        rows[i][2] = prodList.get(i).getDescription();
        rows[i][3] = prodList.get(i).getPrice();
        rows[i][4] = prodList.get(i).getCost();
        rows[i][5] = prodList.get(i).getCat();
        rows[i][6] = prodList.get(i).getStock();
    }

    classes.table model = new classes.table(rows, columnNames);
    productTable.setModel(model);
}
```

Again I added catfk as a parameter.

Then I call the categories class and create a new instance of categories called category.

I set the HashMap map to the returned value of the addToCombo() method returns map with all the categories in it.

Then I used the getSelectedItem() and toString() method to retrieve the selected item as a string from the categoryComboBox and used the get method to return the value mapped by the key, the key is the selected category.

Then I updated where I call productList and added catfk to the corresponding postion.

After testing this method I got this error when I tried to open the pos section:

"AWT-EventQueue-0" java.lang.NullPointerException: Cannot invoke "Object.toString()" because the return value of "javax.swing.JComboBox.getSelectedItem()" is null

I wrote a print line to print the map and I got the result below:

```
{test=20, searchCategoryTest=22, jTextField1=23, disposeMethodTest=24}
```

I realised the reason I had gotten the error message is because the fillProductsTable() method was called in the constructor as soon as the pos screen was opened however at this time the user wouldn't have selected any category in the combo box so the choice was null, therefore the return value of the getSelectedItem() method would be null .

To fix this I added a button so that the user could select a category then press the button to confirm the search, I also had to update the fillProductsTable() method as seen below:

```
public void fillProductsTable(String searchVal, int stock, int catfk){
    //fills the jTable with contents of arrayList
    classes.products prod = new classes.products();
    ArrayList <classes.products> prodList = prod.productList("", 0, catfk);

    String [] columnNames = {"ID","Name","Description","Price","Cost","Category","Stock"};
    Object [][] rows = new Object[prodList.size()][7];

    for(int i = 0; i<prodList.size(); i++){
        rows[i][0] = prodList.get(i).getProductPK();
        rows[i][1] = prodList.get(i).getName();
        rows[i][2] = prodList.get(i).getDescription();
        rows[i][3] = prodList.get(i).getPrice();
        rows[i][4] = prodList.get(i).getCost();
        rows[i][5] = prodList.get(i).getCat();
        rows[i][6] = prodList.get(i).getStock();
    }

    classes.table model = new classes.table(rows, columnNames);
    productTable.setModel(model);
}

private void confirmCategoryButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    classes.categories category = new classes.categories();
    HashMap<String, Integer> map = category.addToCombo();
    System.out.println(map);
    int catfk = map.get(categoryComboBox.getSelectedItem().toString());
    fillProductsTable("", 0, catfk);
    setColumnNames();
}
```

The first two lines of code have been previously tested and successfully work to fill map with the categories from the database.

The print line prints out the contents of map for testing purposes.

The catfk retrieves the corresponding value in the HashMap map for the key retrieved from the combo box.

Then I pass catfk into the fillProductsTable() method in the correct position so the correct products are retrieved.

Lastly I set the column names again using the setColumnNames method.

After testing I received the results below:

ID	Name	Descrip...	Price	Cost	Category	Stock
20	testPro...	jTextFiel...	50.55	35.55	search...	5
21	jTextFiel...	jTextFiel...	50.55	35.55	search...	5

When I selected the “test” category, products that were in the “searchCategoryTes” category were shown.

I added a print line to print catfk after it had been retrieved:

```
private void confirmCategoryButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    classes.categories category = new classes.categories();
    HashMap<String, Integer> map = category.addToCombo();
    System.out.println(map);
    int catfk = map.get(categoryComboBox.getSelectedItem().toString());
    System.out.println(catfk);
    fillProductsTable("", 0, catfk);
    setColumnNames();
}
```

This is the result I got:

20

This printed out the correct category primary key for the test category. This means that the problem was not in the posScreen class code but was instead in the productList method code.

Development and Testing

After reviewing the productList method I found this:

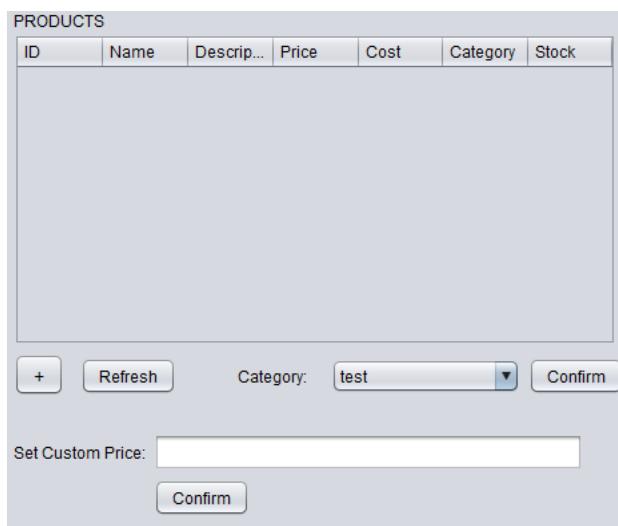
```
String query = "SELECT * FROM products WHERE deletedflag = 0 AND product_name LIKE ? AND stock > ? AND category_fk > ?";
```

This means that only products that have a category_fk over 20 were retrieved which is products that are in the “searchTes” category because the foreign key for that category is 22.

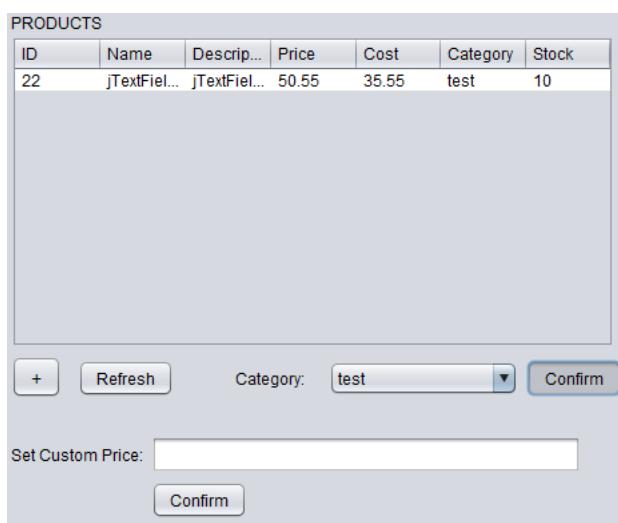
I updated the query by changing the greater than method to an equal to method as seen below:

```
String query = "SELECT * FROM products WHERE deletedflag = 0 AND product_name LIKE ? AND stock > ? AND category_fk = ?";
```

Now when I open the pos section there are no products displayed:



But when I search for a specific category the products from that category are displayed successfully:



I realised this was because I used an “equal to” sign so initially when the pos screen is opened the constructor is called which calls the fillProductsTable method but passes in 0 for the catfk so the method was displaying all the products with a category foreign key of 0, which is none. However the

Development and Testing

category sorting part was working because the method was displaying all the products that have a category foreign key that matches the selected category primary key.

My next attempt at fixing this problem was to add another placeholder to the productList() and fillTable() methods which defined the value of the sign used in the query as seen below:

```
public ArrayList<products> productList(String searchVal, int stock, int catfk, String sign){
    //creates an arraylist
    //creates a new instance of "products" (as shown in method earlier) with data from the products table in the database
    //adds the new product to the arrayList

    ArrayList<products> prodList = new ArrayList<>();

    connection = database.getConnection();
    PreparedStatement ps;
    ResultSet rs;

    String query = "SELECT * FROM products WHERE deletedflag = 0 AND product_name LIKE ? AND stock > ? AND category_fk ? ?";

    try{
        ps = connection.prepareStatement(query);
        ps.setString(1, "%" + searchVal + "%");
        ps.setInt(2, stock);
        ps.setString(3, sign);
        ps.setInt(4, catfk);
        rs = ps.executeQuery();
        products product;
        while (rs.next()){
            product = new products(rs.getInt("product_pk"), rs.getString("product_name"), rs.getString("product_description"));

            prodList.add(product);
        }
    }
    catch (SQLException ex){
        Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

I added the parameter “sign” of type String and updated the query as seen.

Then I used a setString line to set the value of the 3rd ? placeholder to the variable sign.

Next I had to update the fillProductsTable() method below:

```
public void fillProductsTable(String searchVal, int stock, int catfk, String sign){
    //fills the jTable with contents of arrayList
    classes.categories category = new classes.categories();
    HashMap<String, Integer> map = category.addToCombo();
    //catfk = map.get(categoryComboBox.getSelectedItem().toString());

    if (categoryComboBox.getSelectedItem() == null){
        sign = ">=";
    }
    else{
        sign = "=";
    }

    classes.products prod = new classes.products();
    ArrayList <classes.products> prodList = prod.productList("", 0, catfk, sign);
    //System.out.println(prodList);

    String [] columnNames = {"ID","Name","Description","Price","Cost","Category","Stock"};
    Object [][] rows = new Object[prodList.size()][7];

    for(int i = 0; i<prodList.size(); i++){
        rows[i][0] = prodList.get(i).getProductPK();
        rows[i][1] = prodList.get(i).getName();
        rows[i][2] = prodList.get(i).getDescription();
        rows[i][3] = prodList.get(i).getPrice();
        rows[i][4] = prodList.get(i).getCost();
        rows[i][5] = prodList.get(i).getCat();
        rows[i][6] = prodList.get(i).getStock();
    }
}
```

First I added the parameter sign of type String to the method.

Then I added the if statement so that if the value of getSelectedItem() method is null, which it will be when the screen is first opened, then sign is set to “>=” so that when the constructor is called, in the query it writes:

“WHERE category_fk >= 0” which will be all the products as all the categories have a primary key over 0. However when the user has selected an item the value of getSelectedItem() won’t be null so sign will be set to “=” so the query will write:

“WHERE category_fk = ” and it will be equals the selected category foreign key.

However I have now realised I won’t need the else statement as the fillProductsTable() method is called on button press for the confirm category button so in that method I can just pass in the retrieved category foreign key and the correct sign which will be “=”.

However after testing I got this error:

You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near ''>=' 0' at line 1

This means that although the methods worked to pass in the correct symbol, which means the if statement worked, they passed it in surrounded by single quotes which causes an error.

So I decided to back to just using an equals sign in the query and try to solve the problem from there.

My plan to fix this is to edit the productList() method and add an if statement and to use two different queries so when the screen is first opened the query is set to select everything from the products table irrelevant of the category_fk field and when the user selects a category the category foreign key will be retrieved and a query will be sent to retrieve everything from the products table where the category_fk field matches the retrieved category foreign key.

The code is below:

```

if (catfk != 0){
    query = "SELECT * FROM products WHERE deletedflag = 0 AND product_name LIKE ? AND stock > ? AND category_fk = ?;";
    try{
        ps = connection.prepareStatement(query);
        ps.setString(1, "%" + searchVal + "%");
        ps.setInt(2, stock);
        ps.setInt(3, catfk);
        rs = ps.executeQuery();
        products product;
        while (rs.next()){
            product = new products(rs.getInt("product_pk"), rs.getString("product_name"), rs.getString("product_description"));

            prodList.add(product);
        }
    }
    catch(SQLException ex){
        Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
    }
}
else{
    query = "SELECT * FROM products WHERE deletedflag = 0 AND product_name LIKE ? AND stock > ?;";
    try{
        ps = connection.prepareStatement(query);
        ps.setString(1, "%" + searchVal + "%");
        ps.setInt(2, stock);
        rs = ps.executeQuery();
        products product;
        while (rs.next()){
            product = new products(rs.getInt("product_pk"), rs.getString("product_name"), rs.getString("product_description"));
    }
}

```

Development and Testing

I also now have to update where I call the methods in both the productsScreen and the posScreen. In the products screen I want to retrieve all the products regardless of category foreign key so I will pass in 0 for catfk when I call the method but in the posScreen when I first call the method in the constructor I will pass in 0 but when I call the method under the confirmCategory button I will pass in the retrieved category as seen below:

```
public posScreen() {
    initComponents();
    fillProductsTable("", 0, 0);
    comboBoxAttach();
    setColumnNames();
}

private void confirmCategoryButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    classes.categories category = new classes.categories();
    HashMap<String, Integer> map = category.addToCombo();
    System.out.println(map);
    int catfk = map.get(categoryComboBox.getSelectedItem().toString());
    System.out.println(catfk);
    fillProductsTable("", 0, catfk);
    setColumnNames();
}
```

After testing I got these results:

Search: <input type="text"/> <input type="button" value="SEARCH"/>						
ID	Name	Description	Price	Cost	Category	Stock
19	searchProdu...	JTextField2	50.55	35.55	searchCatego...	0
20	testProduct	JTextField2	50.55	35.55	searchCatego...	5
21	JTextField1	JTextField2	50.55	35.55	searchCatego...	5
22	JTextField1	JTextField2	50.55	35.55	test	10

All of the correct products were retrieved and displayed in the products section.

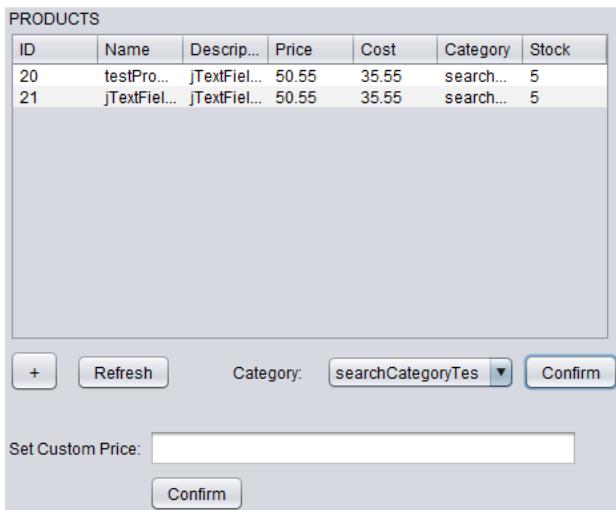
PRODUCTS

ID	Name	Description	Price	Cost	Category	Stock
20	testPro...	JTextFiel...	50.55	35.55	search...	5
21	JTextFiel...	JTextFiel...	50.55	35.55	search...	5
22	JTextFiel...	JTextFiel...	50.55	35.55	test	10

+ Refresh Category: test Confirm

Set Custom Price:
Confirm

Initially when I opened the POS screen all the correct products were shown.



However, after selecting a category only products from that category were shown.

NOTE: By only retrieving products that have a stock over 0 there is no need to add the function that notifies the user when they add a product with 0 stock to the order.

NOTE: if I decide to add a sort by category feature to the products section I will re-use the code in this section but for now I want to focus on completing the rest of the project.

4.8.5 Adding Refresh Button Code

Next I have to add the code to refresh the table when the refresh button is clicked.

```
private void refreshProductsTableActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    fillProductsTable("", 0, 0);
    setColumnNames();
}
```

First I call the `fillProductsTable()` method and pass in the values seen above so that all the products with any name, stock over 0 and any category foreign key are shown.

Then I call the `setColumnNames()` to reset the column names as they will revert to A, B, C etc when the `fillProductsTable()` method is called.

This method works as seen below:

Development and Testing

ID	Name	Description	Price	Cost	Category	Stock
22	jTextFiel...	jTextFiel...	50.55	35.55	test	10

ID	Name	Description	Price	Cost	Category	Stock
20	testPro...	jTextFiel...	50.55	35.55	search...	5
21	jTextFiel...	jTextFiel...	50.55	35.55	search...	5
22	jTextFiel...	jTextFiel...	50.55	35.55	test	10

4.8.6 Updates in Database and Classes Due To Change In Plan

In this section I will write the code to add products to the necessary tables in the database when they are added to an order.

I have decided to do this differently than how I initially planned. Instead of having the currentorder, currentorderproducts and orderproducts table I will just have an orderproducts table which will hold information for an order such as order_fk to show which order the orderproducts record relates to and it would also have a product_fk field to show which products was added to the order. In addition it would also have a quantity field.

4.8.6.1 Creating New Orderproducts Table And Deleting Old Tables

Below is the sql I used to drop the old tables:

```
1 •  drop table currentorder  
1 •  drop table currentorderproducts  
1 •  drop table orderproducts  
1 •  drop table orders_products
```

Next, I have to create the new orderproducts table as seen below:

```

1 • CREATE TABLE `orderproducts` (
2     `orderproducts_pk` int NOT NULL AUTO_INCREMENT,
3     `order_fk` int NOT NULL,
4     `product_fk` int NOT NULL,
5     `quantity` int NOT NULL,
6     PRIMARY KEY (`orderproducts_pk`),
7     KEY `orderproducts_fk_1` (`order_fk`),
8     KEY `orderproducts_fk_3` (`product_fk`),
9     CONSTRAINT `orderproducts_fk_1` FOREIGN KEY (`order_fk`) REFERENCES `orders` (`order_pk`),
10    CONSTRAINT `orderproducts_fk_2` FOREIGN KEY (`product_fk`) REFERENCES `products` (`product_pk`)

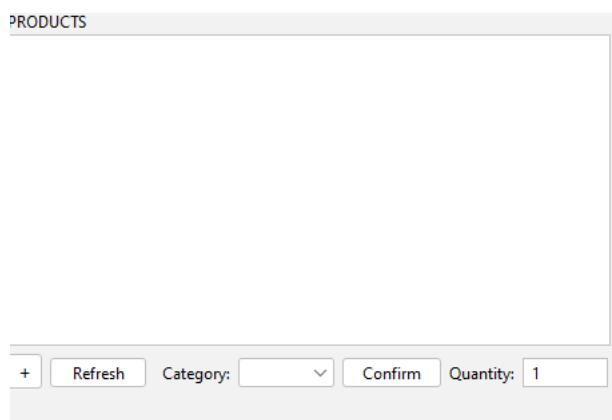
```

A new orderproducts record will be created for every different product added to an order.

I have set the default value to 0 so when an order is created initially it is incomplete, but when the user presses the confirm order button I will update this and set the value to 1 to show the order is complete.

4.8.6.2 Jframe Changes

Next I need to add a text field so the user can enter the quantity for the product they want to add.



Now I can start developing the code that will create an order when a product is added.

I am going to create an arrayList that holds the products that have been added to the order.

Since I will be using the orderproducts table I won't need any fields for product related information in the order table, product description, except information relevant to the order such as price and cost etc.

I am going to display the product name and the product price and primary key in the order jTable so I will create a method that creates objects of the pos class that have values for the name and the price and primary key and the arrayList will hold these objects.

The first methods I am going to create are getter and setter methods for all the necessary variables:

```

public Integer getProductPK(){
    return product_pk;
}

public Integer getOrderPK(){
    return order_pk;
}

public float getPrice(){
    return product_sellprice;
}

public float getCost(){
    return product_cost;
}

public Integer getFK(){
    return createdby_fk;
}

public Date getDateCreated(){
    return createdon;
}

public Integer getQuantity(){
    return quantity;
}

public void setProductPK(int productPK){
    this.product_pk = productPK;
}

public void setOrderPK(int orderPK){
    this.order_pk = orderPK;
}

public void setPrice(float price){
    this.product_sellprice = price;
}

public void setCost(float cost){
    this.product_cost = cost;
}

public void setQuantity(int quantity){
    this.quantity = quantity;
}

```

4.8.7 Developing pos Class constructor

Next I need to create a method that when called will create objects of the pos class that hold the correct information to display in the order table. The code is below:

```

public pos(Integer pk, String name, int quantity, float price){

    this.product_pk = pk;
    this.product_name = name;
    this.quantity = quantity;
    this.product_sellprice = price;

}

```

4.8.8 Developing Add Product To Order Array List Method

Now I need to create a method that takes the selected products primary key and creates objects of the pos class which take values from the corresponding fields in the products table from the record with a matching primary key and inserts these objects into an arrayList.

```

public ArrayList<pos> productInOrderList(int quantity, int id) {
    ArrayList<pos> productOrderList = new ArrayList<>();

    connection = database.getConnection();
    PreparedStatement ps;
    ResultSet rs;
    String query;

    query = "SELECT product_pk, product_name, product_sellprice FROM products WHERE product_pk = ?";
    try{
        ps = connection.prepareStatement(query);
        ps.setInt(1, id);
        rs = ps.executeQuery();
        pos orderProduct;

        while(rs.next()){
            orderProduct = new pos(rs.getInt("product_pk"), rs.getString("product_name"), quantity, rs.getFloat("product_sellprice"));
            productOrderList.add(orderProduct);
        }
    } catch(SQLException ex){
        Logger.getLogger(pos.class.getName()).log(Level.SEVERE, null, ex);
    }
    return productOrderList;
}

```

I have passed in the integer variables quantity and id as these are both dependent on the user as the user must enter the quantity manually and select the product.

First I create an ArrayList that holds objects of the pos class called productOrderList.

Then I define all the necessary variables.

Then I create the query variable which retrieves the primary key, name, and price of the selected product from the products table.

I set the placeholder for the where clause to id as this will be retrieved in the posScreen class code.

Then I create orderProduct which is an instance of the pos class, taking the values from the database as parameters and I add this object to productOrderList.

Then I catch any errors to make sure the user doesn't get a complicated error displayed to them.

4.8.9 Developing Fill Order jTable Method

Now I need to develop the method to fill the table with the orderProduct objects in the productOrderList. This method will take the same parameters as the method above so I can pass the user inputted quantity into the productInOrderList method and so the correct product data is retrieved.

```

public void fillOrderTable(int quantity, int id){
    //fills the jTable with contents of arrayList

    int rowIndex = orderTable.getSelectedRow();
    quantity = Integer.valueOf(quantityTextField.getText());
    id = Integer.valueOf(orderTable.getValueAt(rowIndex, 0).toString());

    classes.pos pos = new classes.pos();
    ArrayList <classes.pos> posList = pos.productInOrderList(quantity, id);
    String [] columnNames = {"ID", "Name", "Quantity", "Price"};
    Object [][] rows = new Object[posList.size()][4];

    for(int i = 0; i<posList.size(); i++){
        rows[i][0] = posList.get(i).getProductPK();
        rows[i][1] = posList.get(i).getProductName();
        rows[i][2] = posList.get(i).getQuantity();
        rows[i][3] = posList.get(i).getPrice();
    }

    classes.table model = new classes.table(rows, columnNames);
    orderTable.setModel(model);
}

```

Here I just edited the fillProductTable code. I created integer parameters quantity and id.

I passed quantity and id into the productInOrderList() method as seen.

I created a 1d array called columnNames the same as for the other jTable sand set the names as seen.

Then I created a 2d object array called rows that takes the returned value of the size() method on posList and 4 as values so it has the right amount of rows and columns.

Then I use a for loop to loop through posList setting the row in position i and column 0 to the primary key for the selected product and repeated this for the other field as seen.

Then I created an instance of the table class called model and passed in rows[] and columnNames[].

Then I use the setModel() method and pass in the object model to set the order jTable to this model.

4.8.10 Developing Add Product To Order Button Code

In this code I will need to call the fillTable() method when the button is pressed. The code is below:

```

private void addProductToOrderButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int rowIndex = orderTable.getSelectedRow();
    int quantity = Integer.valueOf(quantityTextField.getText());
    int id = Integer.valueOf(orderTable.getValueAt(rowIndex, 0).toString());
    fillOrderTable(quantity, id);
}

```

I retrieved the rowIndex using the getSelectedRow() method.

Then I set quantity to be equal to the user inputted value in the quantity text field and used the getText() method to retrieve this value and then used the Integer.valueOf() method to conver this value to an integer.

Then I retrieved the primary key for the selected product by using the getValueAt() method and passing in rowIndex and 0 so it gets the value at the correct selected products row index and converts this value to a string and then to an integer.

Then I passed these value into the fillOrderTable() method.

Development and Testing

After testing this method I got this error:

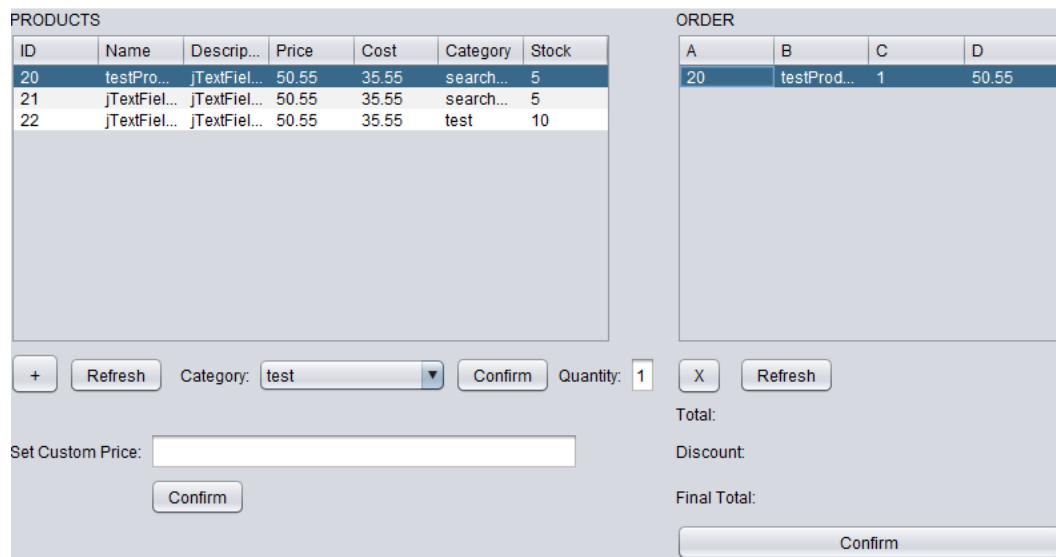
```
Exception in thread "AWT-EventQueue-0" java.lang.ArrayIndexOutOfBoundsException: 0 >= 0
```

This is because i referenced the orderTable and not the productTable in the code above.

After fixing the method as seen below:

```
private void addProductToOrderButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int rowIndex = productTable.getSelectedRow();
    int quantity = Integer.valueOf(quantityTextField.getText());
    int id = Integer.valueOf(productTable.getValueAt(rowIndex, 0).toString());
    fillOrderTable(quantity, id);
}
```

I tested again and these were the results:



As you can see the method worked and the selected product was displayed in the order table.

Now I need to create the setColumnNames() method for this table.

4.8.11 Developing setOrderColumnNames Method

Now I need to develop the method to set the column names for the order table. This method is simple so I can just copy it from the products table and change the names and the table it references as seen below:

```
public void setOrderColumnNames() {
    orderTable.getColumnModel().getColumn(0).setHeaderValue("ID");
    orderTable.getColumnModel().getColumn(1).setHeaderValue("Name");
    orderTable.getColumnModel().getColumn(2).setHeaderValue("Quantity");
    orderTable.getColumnModel().getColumn(3).setHeaderValue("Price");
}
```

After testing this method it worked as seen below:

ORDER			
ID	Name	Quantity	Price
20	testProd...	1	50.55
<input type="button" value="X"/>		<input type="button" value="Refresh"/>	

The quantity field also works:

ORDER			
ID	Name	Quantity	Price
23	rice	10	5.0
<input type="button" value="X"/>		<input type="button" value="Refresh"/>	

However I found that I couldn't add more than one product to the order, every time I tried to add another product it replaced the original one as seen below:

ORDER			
ID	Name	Quantity	Price
24	chicken	1	50.0
<input type="button" value="X"/>		<input type="button" value="Refresh"/>	

ORDER			
ID	Name	Quantity	Price
25	duck	5	25.0
<input type="button" value="X"/>		<input type="button" value="Refresh"/>	

4.8.11.1 Adding Multiple Products Problem

To test where the problem is I added a print line at the bottom of the fillOrderTable() method. The print line prints out posList which is the ArrayList that contains the products being added to the order. These were the results I got when I added a new product to the order:

```
[classes.pos@1a4e8af0]
[classes.pos@3bcda406]
[classes.pos@1fbc7c5c]
```

This is because every time I added a product to the order I call the productInOrderList() method so a new ArrayList is being created so only one product is being displayed in the order section as there is only ever one product in the ArrayList.

I can fix this by developing the methods to add records to the order table and the orderproducts table and calling these methods when the user adds a product to the order.

4.8.12 Developing Add Order Method

First, I need to develop the method to create an order in the orders table in the database. This is because the orderproducts table has a field order_fk which holds the order primary key so the order must be created first so there is a primary key to reference.

The order table has a field createdby_fk which holds the primary key value for the user currently logged in to show they were the user who made the order. I already have a global variable called userPrimaryKey in the menu screen class which holds the current user primary key so I need to pass this primary key into the posScreen class. I can do this by creating a global variable in the posScreen class called userPrimaryKey and setting this variable to the value of userPrimaryKey in the menuScreen class. I would do this when the pos button is pressed so the code will be under the action listener for the pos button.

```
posScreen screen = new posScreen();
screen.setDefaultCloseOperation(posScreen.DISPOSE_ON_CLOSE);
screen.pack();
screen.setVisible(true);
screen.userPrimaryKey = userPrimaryKey;
```

Because I have the order products table which has a foreign key field for the orders table and the products table I don't need the price or cost fields in the orders table so I have dropped these columns using the sql below:

```
1 • alter table orders
2   drop column order_cost

1 • alter table orders
2   drop column order_price
```

Now I need to create the add order method.

This method will just be an insert statement to insert a new record in the orders table. There will be an integer parameter for this method which will hold the primary key for the user.

```
public static void addOrder(int userPK){

    Connection connection = database.getConnection();
    PreparedStatement ps;
    String query;

    query = "INSERT INTO orders(createdby_fk, date) VALUES(?,sysdate())";

    try{
        ps = connection.prepareStatement(query);
        ps.setInt(1, userPK);

        if(ps.executeUpdate() != 0){
            //JOptionPane.showMessageDialog(null, "Order Added");
        }
        else{
            JOptionPane.showMessageDialog(null, "Error");
        }
    }
    catch(SQLException ex){
        Logger.getLogger(orders.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

The query inserts into the createdby_fk field and date field in the orders table. It inserts the values ?, and sysdate(). The sysdate() function returns the current datetime so that value is inserted into the date field.

I have used a `setInt()` method to set the ? placeholder to `userPK` so the `createdby_fk` field will hold the value of the user primary key passed into the method when it is called.

Again I use the `executeUpdate()` method which returns the number of records affected by the executed query and if this is equal to 0 then no records were affected so the query has failed so an error message is displayed else if it affects a record then the method was a success so no message is displayed.

Next I have to call this method under the add product to order button as seen below:

```
if (orderCreated == false){  
    classes.orders.addOrder(userPrimaryKey);  
    orderCreated = true;
```

I have created a Boolean global variable called `orderCreated` which is true if the order has been created and false if it hasn't. When the constructor is first called this variable is declared as false so when the user presses the `addProductToOrderButton` the `addOrder()` method is called and the global variable `userPrimaryKey` is passed in. Then `orderCreated` is set to true, `orderCreated` will be set to false again when the confirm order button is pressed so the user can start a new order.

```
private void confirmOrderButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    orderCreated = false;  
}
```

After testing the `addOrder()` method worked as this record was inserted into the order table:

33 10 2023-01-06 09:13:50

4.8.13 Developing Get Latest Order Primary Key method

I also am going to need a method that inserts into the `orderproducts` table however first I need to have the corresponding order primary key so I have the correct values to insert into the `orderproducts` table.

This method is quite simple it will just select the biggest primary key of the record in the `orders` table for the correct user. This method will need a parameter to pass in the current user primary key variable.

```

public static int getLatestOrder(int userPK) {
    Connection connection = database.getConnection();
    PreparedStatement ps;
    ResultSet rs;
    String query;
    int orderPK = 0;

    query = "select max(order_pk)order_pk from orders where createdby_fk = ?";

    try{
        ps = connection.prepareStatement(query);
        ps.setInt(1, userPK);

        rs = ps.executeQuery();
        rs.next();
        orderPK = rs.getInt("order_pk");
    }
    catch(SQLException ex){
        Logger.getLogger(orders.class.getName()).log(Level.SEVERE, null, ex);
    }

    return orderPK;
}

```

The userPK parameter will be used to pass in the userPrimaryKey variable when this method is called.

I have initialised all the necessary variables and objects such as Connection.

The query uses the max function to select the biggest order_pk from the order table where the value in the createdby_fk field is equal to ?.

Then I use the setInt() method to set ? to userPK.

Next I execute the query and call the next() method to iterate through the result set.

Then I set orderPK to the returned value of the getInt() method on the order_pk field which returns the primary key for the order.

Then I catch any database errors.

Lastly I return orderPK to the user.

Next I need to call this method under the addProductToOrderButton action listener and declare a variable to hold the order primary key:

```
int orderPK = classes.orders.getLatestOrder(userPrimaryKey);
```

Here I have called the method and passed in userPrimaryKey so the current users primary key is passed into the method so the correct order primary key is retrieved.

4.8.14 Developing Add oderProduct method

NOTE: I have renamed the pos class to orderproducts as it contains only methods that affect the orderproducts table.

Next, I need to develop the method to insert into the orderproducts table. This method will be similar to the add order method and will use an insert statement to create a record in the orderproducts table and I will pass in the values for the quantity, order primary key and product primary key.

```

public static void addOrderProduct(int quantity, int productPK, int orderPK){

    Connection connection = database.getConnection();
    PreparedStatement ps;
    String query;

    query = "INSERT INTO orderproducts(order_fk, product_fk, quantity) VALUES(?, ?, ?)";

    try{
        ps = connection.prepareStatement(query);
        ps.setInt(1, orderPK);
        ps.setInt(2, productPK);
        ps.setInt(3, quantity);

        if(ps.executeUpdate() != 0){
            //JOptionPane.showMessageDialog(null, "Order Added");
        }
        else{
            JOptionPane.showMessageDialog(null, "error");
        }
    }
    catch(SQLException ex){
        Logger.getLogger(orders.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

I have added three integer parameters one each for quantity, order primary key and product primary key.

I use an insert statement to insert into the order_fk, product_fk and quantity fields in the orderproducts table and pass in the values ?, ?, ?.

Then I call the prepareStatement() method and pass in the query and use the setInt() method to set the values to the appropriate parameters as seen. This creates the pre-compiled query with all the passed in values.

Then I call the executeUpdate() method to execute the query.

This method works as this record was added to the orderproducts table when I added a product to the order:

24 34 23 1

4.8.15 Updating Product In Order ArrayList Method

Next I need to update the method that retrieves the necessary data from the orderproducts and products table in the database. In order to do this I will have to join the orderproducts and products table as I need to select the product_pk, product_name, and product_sellprice fields from the products table and the quantity from the orderproducts table.

```

public ArrayList<orderProducts> productInOrderList(int orderPK){

    ArrayList<orderProducts> productOrderList = new ArrayList<>();

    connection = database.getConnection();
    PreparedStatement ps;
    ResultSet rs;
    String query;

    query = "select product_pk, product_name, product_sellprice, quantity from products,orderproducts where product_pk
try{
    ps = connection.prepareStatement(query);
    ps.setInt(1, orderPK);
    rs = ps.executeQuery();
    orderProducts orderProduct;

    while(rs.next()){
        orderProduct = new orderProducts(rs.getInt("product_pk"), rs.getString("product_name"), rs.getInt("quantit
        productOrderList.add(orderProduct);
    }
}
catch(SQLException ex){
    Logger.getLogger(orderProducts.class.getName()).log(Level.SEVERE, null, ex);
}
return productOrderList;
}

where product_pk = orderproducts.product_fk and order_fk = ?";

    .getInt("quantity"), (rs.getFloat("product_sellprice")*rs.getInt("quantity")));

```

For this method I have an integer parameter orderPK which will be used to pass in the primary key for the order.

I create a new ArrayList that holds objects of the orderProduct class.

The query selects the product_pk, product_name, product_sellprice and quantity fields. I have used a comma, which can be used to replace the join keyword when joining tables, to join the two tables and used a where clause so the data is retrieved from the product with a primary key matching the product foreign key field in the orderproducts table and where the order foreign key field in the orderproducts table is equal to ? which will hold the order primary key so the correct product information is being retrieved.

I have used the setInt() method to set ? to orderPK.

I have created a new instance of the orderProducts class called orderProduct and passed in the values retrieved from the database as seen.

I have multiplied the retrieved price for the product by the retrieved quantity so the orderProduct object has the correct price.

4.8.16 Updating Fill Order jTable Method

Next I need to update the fill order method. I no longer need to pass in the quantity or product primary key for this method so don't need those parameters however I do need an integer parameter to pass in the order primary key value.

Development and Testing

```
public void fillOrderTable(int orderPK){  
    //fills the jTable with contents of arrayList  
  
    classes.orderProducts orderProduct = new classes.orderProducts();  
    ArrayList <classes.orderProducts> productsList = orderProduct.productInOrderList(orderPK);  
    String [] columnNames = {"ID","Name","Quantity","Price"};  
    Object [][] rows = new Object[productsList.size()][4];  
  
    for(int i = 0; i<productsList.size(); i++){  
        rows[i][0] = productsList.get(i).getProductPK();  
        rows[i][1] = productsList.get(i).getProductName();  
        rows[i][2] = productsList.get(i).getQuantity();  
        rows[i][3] = productsList.get(i).getPrice();  
    }  
  
    classes.table model = new classes.table(rows, columnNames);  
    orderTable.setModel(model);  
    System.out.println(productsList);  
}
```

This method hasn't changed much, all that's different is I am only passing in the order primary key value and the name of the class is different now so I have updated the method accordingly.

Lastly, I need to call this method under the addProductToOrderButton action listener as seen below:

```
fillOrderTable(orderPK);  
setOrderColumnNames();
```

I have passed in the orderPK variable as this holds the primary key for the order and I have also called the setOrderColumnNames() method so the header values are set correctly for the order jTable.

This method worked as seen below:

ORDER			
ID	Name	Quantity	Price
23	rice	1	5.0
24	chicken	1	50.0

4.8.17 Developing Delete Order Product Method

Next, I need to develop the method to allow the user to delete products from the order table. I will use the getSelectedItem() method to get the primary key for the product the user has selected then I will pass this primary key into the delete order product method and execute a delete query to delete the orderproduct record with a product_fk field that matches the primary key of the selected record.

Development and Testing

```
public static void deleteOrderProduct(int productPK) {  
  
    Connection connection = database.getConnection();  
    PreparedStatement ps;  
  
    try{  
        ps = connection.prepareStatement("DELETE FROM orderproducts WHERE product_fk = ?");  
        ps.setInt(1, productPK);  
  
        if (ps.executeUpdate() != 0){  
            JOptionPane.showMessageDialog(null, "Product deleted from order");  
        }  
        else{  
            JOptionPane.showMessageDialog(null, "Error");  
        }  
    }  
    catch(SQLException ex){  
        Logger.getLogger(orderProducts.class.getName()).log(Level.SEVERE, null, ex);  
    }  
}
```

This method has an integer parameter productPK used to pass in the primary key of the selected product.

The query is simply deletes all the products from orderproducts where the product_fk field value is equal to ?.

I have used the setInt() method to set ? to productPK.

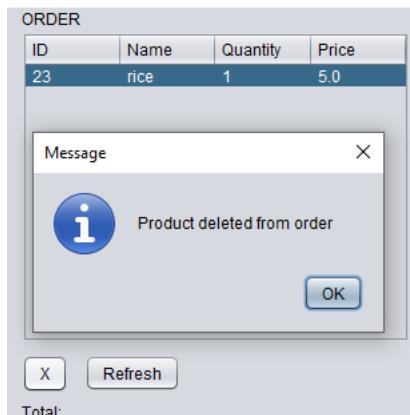
Then I execute the query and display an appropriate message for success or failure.

4.8.18 Delete Product From Order Button Code

Now I need to call the deleteOrderProduct() method under the action listener method for the delete product button. I will have to retrieve the selected product primary key in this method aswell.

```
private void deleteProductFromOrderButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    int rowIndex = orderTable.getSelectedRow();  
    int productPK = Integer.valueOf(orderTable.getValueAt(rowIndex, 0).toString());  
    classes.orderProducts.deleteOrderProduct(productPK);  
}
```

When testing this method I got these results:



However the product wasn't removed from the order jTable, this is because I need to call the fillOrderTable() method again so I will call this method under the refresh button so the user can click that button to refresh the table.

4.8.19 Refresh Button Code

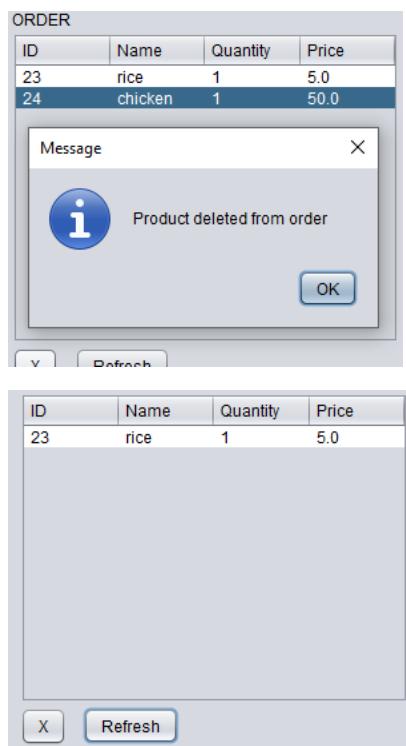
In order to refresh the table I need to pass in the order primary key into the fillOrderTable() method so I will have to retrieve the order primary key using the getLatestOrder() method.

```
private void refreshOrderTableActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int orderPK = classes.orders.getLatestOrder(userPrimaryKey);
    fillOrderTable(orderPK);
    setOrderColumnNames();
}
```

I have retrieved the primary key by passing the userPrimaryKey global variable into the getLatestOrder() method.

Then I have called the fillOrderTable() method and passed in orderPK which holds the order primary key value.

This method worked as seen below:



4.8.20 Developing Calculate Total Price Method

This method will sum up the price of the products in the order and display it to the user. This method will also allow the user to enter a flat amount to deduct from the total price as a discount for the order. This method will also allow the user to enter their own custom price and will set the total price to the user inputted price.

```

public void calculateTotalPrice() {
    float total = 0;
    float finalPrice = 0;
    String totalPriceString;
    String finalPriceString;

    for (int rowIndex = 0; rowIndex < orderTable.getRowCount(); rowIndex++) {
        total = total + Integer.valueOf(orderTable.getValueAt(rowIndex, 3).toString());
    }
    totalPriceString = Float.toString(total);

    totalPriceLabel.setText(totalPriceString);

    if (customPriceTextField.getText() == null) {
        if (discountTextField.getText() != null) {
            float discount = Float.parseFloat(discountTextField.getText());
            finalPrice = total - discount;
        } else {
            finalPrice = total;
        }
    } else {
        float customPrice = Float.parseFloat(customPriceTextField.getText());
        finalPrice = customPrice;
    }

    finalPriceString = Float.toString(finalPrice);
    finalPriceLabel.setText(finalPriceString);
}

```

I have declared four variables: total and finalPrice are floats and will be used for calculations, totalPriceString and finalPriceString are String variables that will be used to display the total and final price to the user.

I have used a for loop to iterate through each row in the order jTable setting total to the value of itself + the value in the price field in the table.

I have used the Float.toString() method and passed in total to convert total from a float to a String so it can be displayed to the user. Then I have used the setText() method and passed in totalPriceString so the totalPriceLabel is set to the value of totalPriceString.

Then I have used an if statement to check if the customPriceTextField has text in it from the user or whether it is null. If it is null then another if statement checks if the discountTextField is null. If it isn't null then the discount amount is retrieved using the Float.parseFloat() method to convert the string value in the text field to a float. Then finalPrice is set to the value of total minus the discount value. Else if the discount text field is null then the finalPrice variable is set to the value of the total variable. If the customPricetextField isn't null then the user inputted custom price is retrieved as a float using the Float.parseFloat() method and finalPrice is set to customPrice.

Lastly, I need to call this method whenever the add product button is pressed.

After testing this method I had this problem:

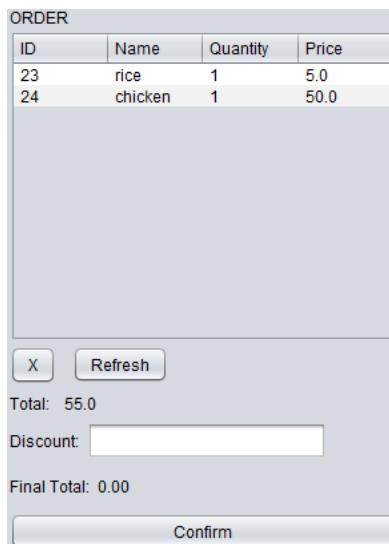
`Exception in thread "AWT-EventQueue-0" java.lang.NumberFormatException: For input string: "5.0"`

This is because I used the `Integer.valueOf` method to convert the String to an integer and I set total which is a float variable to this value, and a float variable cannot be set to an integer value so the error occurred.

After re-testing I got this error:

```
Exception in thread "AWT-EventQueue-0" java.lang.NumberFormatException: empty String
```

However, the total price jLabel was successfully set to the sum of the price field as seen below:



This means the error was referring to the `finalPrice` variables.

In order to fix this error I used the `isEmpty()` method instead of checking if the field was null as seen below:

```
if (customPriceTextField.getText().isEmpty() == true){

    if(discountTextField.getText().isEmpty() == true){
        finalPrice = total;
    }
    else{
        float discount = Float.parseFloat(discountTextField.getText());
        finalPrice = total - discount;
    }

}
else{
    float customPrice = Float.parseFloat(customPriceTextField.getText());
    finalPrice = customPrice;
}
```

Development and Testing

After testing the error was fixed:

ORDER			
ID	Name	Quantity	Price
23	rice	1	5.0
24	chicken	1	50.0

X Refresh

Total: 55.0

Discount:

Final Total: 55.0

Confirm

However, I found that when I deleted a product from the order the total fields didn't update:

ORDER			
ID	Name	Quantity	Price
23	rice	1	5.0
24	chicken	1	50.0
25	duck	1	25.0

X Message Refresh

Product deleted from order

OK

Total: 80.0

Discount:

Final Total: 80.0

Confirm

ORDER			
ID	Name	Quantity	Price
23	rice	1	5.0
24	chicken	1	50.0

X Refresh

Total: 80.0

Discount:

Final Total: 80.0

This is because I haven't created a method to decrease the price when a product is removed.

4.8.21 Developing Method to calculate price after a product is deleted from the order

I need to develop a method that re-calculates the total and final price when a product is deleted from the order. This method will be called under the delete product button. This method will retrieve the current total and final price text fields and subtract the value in the price field for the selected product when the delete product button is pressed.

```
public void calculateNewTotalPrice() {
    float total = 0;
    float finalPrice = 0;
    String totalPriceString;
    String finalPriceString;

    float deletedProductPrice = 0;

    int rowIndex = orderTable.getSelectedRow();
    deletedProductPrice = Float.parseFloat(orderTable.getValueAt(rowIndex, 3).toString());

    total = Float.parseFloat(totalPriceLabel.getText());
    finalPrice = Float.parseFloat(finalPriceLabel.getText());

    total = total - deletedProductPrice;
    finalPrice = finalPrice - deletedProductPrice;

    totalPriceString = Float.toString(total);
    finalPriceString = Float.toString(finalPrice);

    totalPriceLabel.setText(totalPriceString);
    finalPriceLabel.setText(finalPriceString);

}
}
```

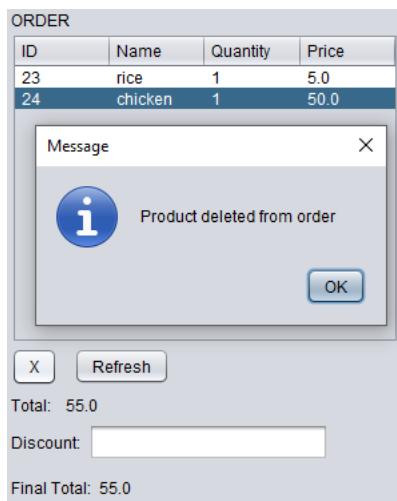
I retrieve the selected row index using the `getSelectedRow()` method. Then I set a variable to hold the price of the deleted product using the `getValueAt()` method and converting it to a string and then to a float.

Then I retrieve the current value for the total and final price labels as floats.

Then I set the new total and final price by subtracting the `deletedProductPrice` variable from them.

Then I convert the total and final price variables to strings and use the `setText()` method to set the value of the total and final price labels.

This method works as seen below:



ORDER			
ID	Name	Quantity	Price
23	rice	1	5.0

Total: 5.0

Discount:

Final Total: 5.0

4.8.22 Developing set custom price and set discounted price methods

During testing for this method I found that the discount and custom price text fields don't set the text. Because of this I have decided to remove the if statements from the calculateTotalPrice() method so that method will solely calculate the price of the products in the table and then I will create two separate methods that retrieve the values from the discount and custom price text fields and set the final total to these values when the confirm button is pressed for the corresponding text field.

First I need to add a confirm button for the discount text field.

Discount:

Final Total: 0.00

The method to set the final price to the custom user inputted price will be very easy I will simply retrieve the value in the customPriceTextField and set the finalPriceLabel to this value.

The method to discount the final price will be slightly more complicated. I will have to retrieve the total as a float and retrieve the user inputted value in the discount text field as a float and then subtract the discount amount from the total and convert it back to a string and then set the finalPriceLabel to this value.

```

public void setFinalTotalToCustom() {
    String customPrice;
    customPrice = customPriceTextField.getText();
    finalPriceLabel.setText(customPrice);
}

public void DiscountFinalTotal() {
    float total = 0;
    float discount = 0;
    float finalPrice = 0;

    String finalPriceString;

    total = Float.parseFloat(totalPriceLabel.getText());
    discount = Float.parseFloat(discountTextField.getText());
    finalPrice = total - discount;

    finalPriceString = Float.toString(finalPrice);

    finalPriceLabel.setText(finalPriceString);
}

```

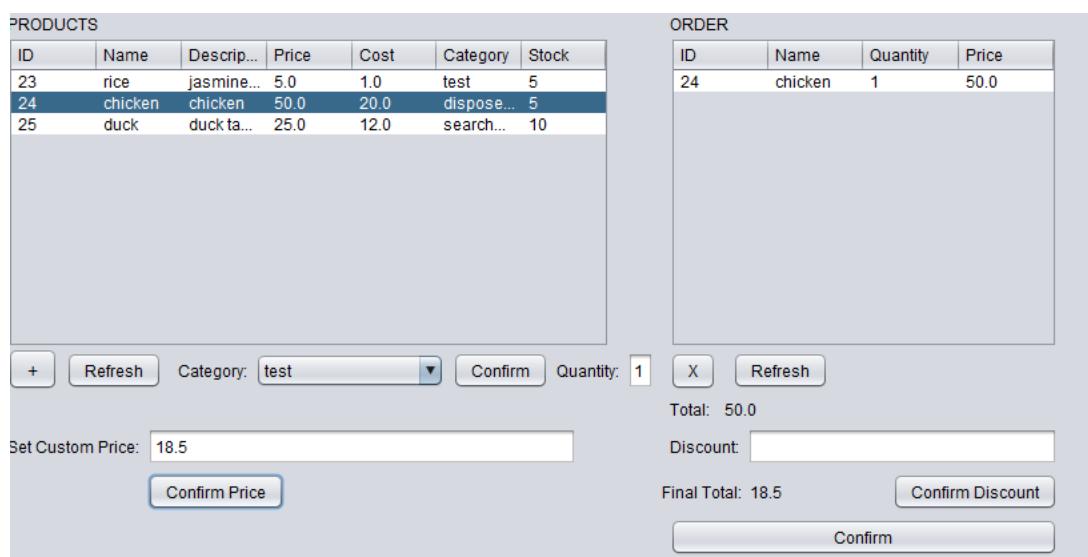
Development and Testing

Next, I need to call these methods under the corresponding confirm buttons as seen below:

```
private void confirmPriceButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    setFinalTotalToCustom();
}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    discountFinalTotal();
}
```

These methods worked as seen below:



4.8.23 Developing Update Stock Method

Now that the pos section is almost finished I need to develop the method to decrease the stock for the products in the order. This method will be fairly simple, it will take two parameters the product primary key and the quantity. The method will issue an update query to decrease the stock of all the products with a matching primary key by the quantity amount.

```

public static void updateStock(int productPK, int quantity){

    Connection connection = database.getConnection();
    PreparedStatement ps;
    try{
        ps = connection.prepareStatement("UPDATE products SET stock = stock - ? WHERE product_pk = ?;");
        ps.setInt(1, quantity);
        ps.setInt(2, productPK);
        ps.executeUpdate();
    }

    catch(SQLException ex){
        Logger.getLogger(orders.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

This method has two parameters productPK and quantity to pass in the product primary keys and the quantity of these products.

The update query sets the value of the stock field to itself minus the quantity of the product.

4.8.24 Confirm Order Button Code

Next I need to call this method under the confirm order button action listener method. I also need to call the fillOrderTable method so that the order table is reset to being empty when the user confirms the order and I need to set the text fields for prices to 0. I also need to set the orderCreated flag to false.

```

private void confirmOrderButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    for (int rowIndex = 0; rowIndex < orderTable.getRowCount(); rowIndex++){
        int productPK = Integer.valueOf(orderTable.getValueAt(rowIndex, 0).toString());
        int quantity = Integer.valueOf(orderTable.getValueAt(rowIndex, 2).toString());
        classes.orders.updateStock(productPK, quantity);
    }

    orderCreated = false;
    fillOrderTable(0);
    setOrderColumnNames();
    totalPriceLabel.setText("0.00");
    finalPriceLabel.setText("0.00");
}

```

I have used a for loop to loop through each row in the order table retrieving the product primary key and quantity for that product on each iteration.

The product primary key and quantity variables are then passed into the updateStock() method.

After testing you can see that this method works:

ID	Name	Description	Price	Cost	Category	Stock
27	Rice	Jasmine Rice	2.5	0.5	Food	19

ID	Name	Quantity	Price
27	Rice	1	2.5

Development and Testing

ID	Name	Description	Price	Cost	Category	Stock
27	Rice	Jasmine	2.5	0.5	Food	18

Category: Quantity:

As you can see after confirming the order with 1 rice and refreshing the products table the stock decreased from 19 to 18.

4.8.25 Test Plan

Test Number	How To Test	Expected Result	Actual Result
1	Attempt to add a product to the order	Product should be added to the order and displayed	Products was added to order and displayed
2	Attempt to remove a product from an order	Product should be removed from order	Product was removed from order
3	Add and remove products from order	Both fields should update when adding and removing products to and from the order	Both fields updated
4	Enter custom price	Final total field should be updated with user inputted price	Field was updated accordingly
5	Enter discount amount	Final total should be decremented by entered amount	Field was updated accordingly
6	Add a product to an order and confirm the order	The stock for that product should decrease by the quantity added to the order	Stock correctly decreased

Development and Testing

4.8.26 Evidence

4.8.26.1 Test 1

PRODUCTS						
ID	Name	Descrip...	Price	Cost	Category	Stock
27	Rice	Jasmin...	2.5	0.5	Food	20

ORDER			
ID	Name	Quantity	Price
27	Rice	1	2.5

+ Refresh Category: Food Confirm Quantity: 1 X Refresh

4.8.26.2 Test 2

ORDER			
ID	Name	Quantity	Price
27	Rice	1	2.5

Message

i Product deleted from order

OK

X Refresh X Refresh

4.8.26.3 Test 3

ORDER			
ID	Name	Quantity	Price
27	Rice	1	2.5
27	Rice	1	2.5

ORDER			
ID	Name	Quantity	Price
27	Rice	1	2.5
27	Rice	1	2.5

ORDER			
ID	Name	Quantity	Price

Total: 5.0 Total: 5.0 Total: 2.5

Discount: Discount: Discount:

Final Total: 5.0 Final Total: 5.0 Final Total: 2.5

Confirm Discount Confirm Discount Confirm Discount

Confirm Confirm Confirm

Development and Testing

Total: 2.5
Discount: <input type="text"/>
Final Total: 2.5
<input type="button" value="Confirm Discount"/>

As you can see the price updates accordingly when removing products from the order however if I have multiple of one product added to the order separately, then when I remove one of the products both are removed from the order. This is a problem that I don't have time to fix. However I know how I would fix it, when a product that already exists in the order is added to the order, instead of adding it separately I would retrieve the quantity of the product in the order and the user entered quantity for the product they want to add to the order and add them together. Then I would update the quantity field for the product in the order with the new quantity. However, I won't have time to implement this fix.

4.8.26.4 Test 4

PRODUCTS						
ID	Name	Descrip...	Price	Cost	Category	Stock
27	Rice	Jasmin...	2.5	0.5	Food	20

ORDER			
ID	Name	Quantity	Price
27	Rice	1	2.5

<input style="border: none; background-color: inherit; color: inherit; font-size: inherit; font-weight: inherit; margin-right: 5px;" type="button" value="+"/>	<input style="border: none; background-color: inherit; color: inherit; font-size: inherit; font-weight: inherit; margin-right: 5px;" type="button" value="Refresh"/>	Category: <input style="border: none; background-color: inherit; color: inherit; font-size: inherit; font-weight: inherit; margin-right: 5px;" type="button" value="Food"/>	<input style="border: none; background-color: inherit; color: inherit; font-size: inherit; font-weight: inherit; margin-right: 5px;" type="button" value="Confirm"/>	Quantity: <input type="text" value="1"/>
Set Custom Price: <input type="text" value="1.50"/>				
<input style="border: none; background-color: inherit; color: inherit; font-size: inherit; font-weight: inherit; margin-right: 5px;" type="button" value="Confirm Price"/>				
<input style="border: none; background-color: inherit; color: inherit; font-size: inherit; font-weight: inherit; margin-right: 5px;" type="button" value="X"/> <input style="border: none; background-color: inherit; color: inherit; font-size: inherit; font-weight: inherit; margin-right: 5px;" type="button" value="Refresh"/>				
Total: 2.5 Discount: <input type="text"/> Final Total: 1.50 <input type="button" value="Confirm Discount"/> <input type="button" value="Confirm"/>				

4.8.26.5 Test 5

ORDER			
ID	Name	Quantity	Price
27	Rice	1	2.5

<input style="border: none; background-color: inherit; color: inherit; font-size: inherit; font-weight: inherit; margin-right: 5px;" type="button" value="X"/>	<input style="border: none; background-color: inherit; color: inherit; font-size: inherit; font-weight: inherit; margin-right: 5px;" type="button" value="Refresh"/>
Total: 2.5	
Discount: <input type="text" value="0.5"/>	
Final Total: 2.0	<input type="button" value="Confirm Discount"/>
<input type="button" value="Confirm"/>	

4.8.26.6 Test 6

<table border="1"> <thead> <tr> <th>ID</th><th>Name</th><th>Description</th><th>Price</th><th>Cost</th><th>Category</th><th>Stock</th></tr> </thead> <tbody> <tr> <td>27</td><td>Rice</td><td>Jasmine</td><td>2.5</td><td>0.5</td><td>Food</td><td>16</td></tr> </tbody> </table>	ID	Name	Description	Price	Cost	Category	Stock	27	Rice	Jasmine	2.5	0.5	Food	16	<table border="1"> <thead> <tr> <th>ID</th><th>Name</th><th>Quantity</th><th>Price</th></tr> </thead> <tbody> <tr> <td>27</td><td>Rice</td><td>5</td><td>12.5</td></tr> <tr> <td>27</td><td>Rice</td><td>1</td><td>2.5</td></tr> </tbody> </table>	ID	Name	Quantity	Price	27	Rice	5	12.5	27	Rice	1	2.5
ID	Name	Description	Price	Cost	Category	Stock																					
27	Rice	Jasmine	2.5	0.5	Food	16																					
ID	Name	Quantity	Price																								
27	Rice	5	12.5																								
27	Rice	1	2.5																								
PRODUCTS																											
<table border="1"> <thead> <tr> <th>ID</th><th>Name</th><th>Description</th><th>Price</th><th>Cost</th><th>Category</th><th>Stock</th></tr> </thead> <tbody> <tr> <td>27</td><td>Rice</td><td>Jasmine</td><td>2.5</td><td>0.5</td><td>Food</td><td>10</td></tr> </tbody> </table>	ID	Name	Description	Price	Cost	Category	Stock	27	Rice	Jasmine	2.5	0.5	Food	10													
ID	Name	Description	Price	Cost	Category	Stock																					
27	Rice	Jasmine	2.5	0.5	Food	10																					

4.8.27 Success Criteria

Success Criteria Number	Success Criteria	Achieved	Notes
2	Form orders	Yes	None
4	Have a keypad	No	Although my POS screen doesn't have a keypad it does have a custom price field that allows the user to enter their own price which serves the same purpose as the keypad. However, I didn't have time to code the method to create the order with the value in the custom price field so it serves no real purpose.
1	Track Stock – automatically decrease stock per purchase	Yes	The stock decreased appropriately when the quantity was one, more than one, and when there were multiple instances of one product in the order.

4.8.28 Review/Stage Evaluation

In this stage I developed the POS section of the code. This is the area that allows the user to select products and add them to orders. The different features in this section include: filtering products by category, adding products to order, deleting products from order, custom price, discount price and stock tracking.

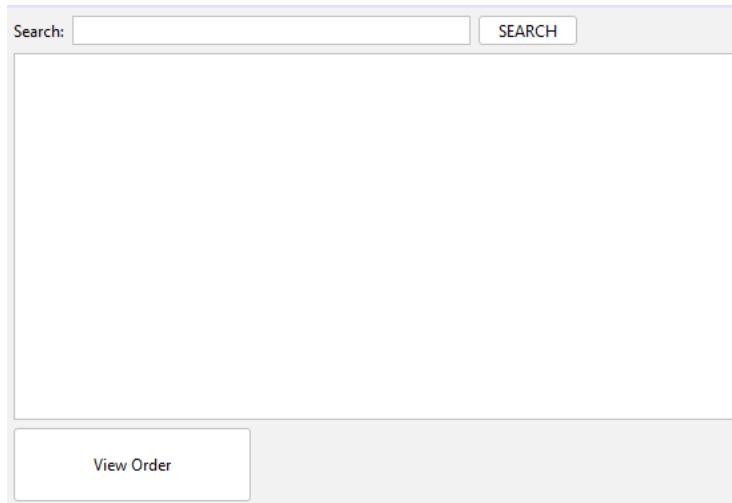
In this section I was also able to re-use the fill jtable methods by altering them slightly according to which table they are filling. I had to create new methods for most of these features such as for

creating a new order when a product is added, creating a new order product record for each product added to the order etc.

This section is the main feature of a POS system and therefore took a lot of time however I am happy with how it turned out, there are a few areas that could be improved such as validating the custom price and discount price fields, but the section works well and accomplishes all three of the success criteria. For success criteria 4 which says the section should have a keypad, instead of a keypad I have the custom price field which serves the same purpose as it allows the user to set the price for the order.

4.9 Stage 8/Orders

I have removed the filter function from the orders screen as I plan to focus on finishing the rest of the sections in the project. This section is also going to require an additional screen as I am going to have one JFrame with a JTable that displays limited information about the order and I will add a button that creates a new JFrame with a JTable that displays much more in depth information about the order such as what products were in it and how much profit the order made. I have also removed the delete order and refresh table buttons.



4.9.1 Getter and Setter methods

First I need to create private global variables for the information I want to display about the orders.

```
private int order_pk;
private int createdby_fk;
private Date date;
private float profit;
```

Then I need to create getter and setter methods

Development and Testing

```
public Integer getOrderPK(){
    return order_pk;
}

public Integer getFK(){
    return createdby_fk;
}

public Float getProfit(){
    return profit;
}

public Date getDate(){
    return date;
}

public void setOrderPK(int orderPK){
    this.order_pk = orderPK;
}

public void setProfit(float profit){
    this.profit = profit;
}
```

4.9.2 Developing order retrieval method

Next, I need to develop the ArrayList method to retrieve all the orders from the database and create objects of the orders class that take the values from the orders table as parameters and then populate an orders ArrayList with these objects.

```
public ArrayList<orders> ordersList(String orderPK) {
    ArrayList<orders> orderList = new ArrayList<>();

    Connection connection = database.getConnection();
    PreparedStatement ps;
    ResultSet rs;
    String query;

    query = "select * from orders where order_pk LIKE ?";

    try{
        ps = connection.prepareStatement(query);
        ps.setString(1,"%"+orderPK+"%");
        rs = ps.executeQuery();
        orders order;

        while(rs.next()){
            order = new orders(rs.getInt("order_pk"), rs.getInt("ordercreatedby_fk"), rs.getDate("date"));

            orderList.add(order);
        }
    }
    catch(SQLException ex){
        Logger.getLogger(orderProducts.class.getName()).log(Level.SEVERE, null, ex);
    }
    return orderList;
}
```

First, I create a new ArrayList called orderList that can hold objects of the orders class.

Then I query the database to select all the data from the orders table that have an order_pk similar to the value of ?

Then I use setString() to set ? to orderPK which is the parameter and will be used to pass in the order primary key value that the user searches for.

Then I create a new object of the orders class called order and pass in the values from the database using the getInt() and getDate() methods.

Then I add order to orderList and return orderList.

4.9.3 Developing Fill Order Table Method

Next I need to create the fillOrderTable() method to fill the table with the order objects from the orderList ArrayList.

```
public void fillOrderTable(String orderPK) {
    classes.orders order = new classes.orders();
    ArrayList <classes.orders> orderList = order.ordersList(orderPK);
    String [] columnNames = {"ID", "UserID", "Date"};
    Object [][] rows = new Object[orderList.size()][3];

    for(int i = 0; i<orderList.size(); i++){
        rows[i][0] = orderList.get(i).getOrderPK();
        rows[i][1] = orderList.get(i).getFK();
        rows[i][2] = orderList.get(i).getDate();
    }

    classes.table model = new classes.table(rows, columnNames);
    orderTable.setModel(model);
}
```

This method has a String parameter orderPK which will hold the order primary key the user searches for.

The ArrayList line fills orderList with orders by calling the ordersList() method which retrieves the data from the database and creates the instances of the orders class.

Then I set the column names for the 1d columnNames array and set the size values for the 2d rows array.

I use a for loop to iterate through orderList filling rows[] with the data in the appropriate place.

Lastly I create model which is an instance of the table class and pass in rows and columnNames and then I use the setModel() method and pass in model to set the model for the orderTable.

Then I call the method in the order screen class constructor and under the search button as seen:

```
private void searchButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String orderPK = searchTextField.getText();
    fillOrderTable(orderPK);
}

public orderScreen() {
    initComponents();
    fillOrderTable("");
}
```

This method worked as seen below:

The screenshot shows a Java Swing application window. At the top left is a label "Search:" followed by a text input field containing a single character. To the right of the input field is a "SEARCH" button. Below this is a JTable with three columns labeled A, B, and C. Column A contains integers from 4 to 22. Column B contains the value 8 for all rows. Column C contains dates all set to "2023-01-05". The table has a vertical scrollbar on the right side.

A	B	C
4	8	2023-01-05
5	8	2023-01-05
6	8	2023-01-05
7	8	2023-01-05
8	8	2023-01-05
9	8	2023-01-05
10	8	2023-01-05
11	8	2023-01-05
12	8	2023-01-05
13	8	2023-01-05
14	8	2023-01-05
15	8	2023-01-05
16	8	2023-01-05
17	8	2023-01-05
18	8	2023-01-05
19	8	2023-01-05
20	8	2023-01-05
21	8	2023-01-05
22	8	2023-01-05

4.9.4 Developing Set Column Names method

```
public void setColumnNames() {
    orderTable.getColumnModel().getColumn(0).setHeaderValue("ID");
    orderTable.getColumnModel().getColumn(1).setHeaderValue("User-ID");
    orderTable.getColumnModel().getColumn(2).setHeaderValue("Date");
}
```

```
public orderScreen() {
    initComponents();
    fillOrderTable("");
    setColumnNames();
}
```

Calling method in constructor

This method worked:

ID	User-ID	Date
4	8	2023-01-05
5	8	2023-01-05
6	8	2023-01-05
7	8	2023-01-05

4.9.5 Creating Products In Order jFrame



The only component on this jframe is a jTable as this table simply needs to display information about products.

4.9.6 Getter and setter methods + constructor for productsInOrder class

```
private int product_pk;
private int order_pk;
private String product_name;
private float product_sellprice;
private float product_cost;
private int quantity;
private float profit;
```

```

public Integer getProductPK() {
    return product_pk;
}

public Integer getOrderPK() {
    return order_pk;
}

public String getName() {
    return product_name;
}

public float getPrice() {
    return product_sellprice;
}

public float getCost() {
    return product_cost;
}

public Integer getQuantity() {
    return quantity;
}

public float getProfit() {
    return profit;
}

public void setProductPK(int productPK) {
    this.product_pk = productPK;
}

public void setOrderPK(int orderPK) {
    this.order_pk = orderPK;
}

public void setName(String name) {
    this.product_name = name;
}

public void setPrice(float price) {
    this.product_sellprice = price;
}

public void setCost(float cost) {
    this.product_cost = cost;
}

public void setQuantity(int quantity) {
    this.quantity = quantity;
}

public void setProfit(float profit) {
    this.profit = profit;
}

public productsInOrder(Integer pk, String name, float price, float cost, int quantity, float profit) {
    this.product_pk = pk;
    this.product_name = name;
    this.product_sellprice = price;
    this.product_cost = cost;
    this.quantity = quantity;
    this.profit = profit;
}

```

4.9.7 Developing Order Product Detail Retrieval method

The code for the productsInOrderSceen class above will all be in a separate class called productsInOrder.

I need to develop the method to retrieve all the data about the products in the order that I need to display such as products primary key, name, price, cost and quantity.

Development and Testing

Once the data is retrieved, I will create objects of the productsInOrder class and create an arrayList to hold these objects.

```
public ArrayList<productsInOrder> productsInOrderList(int orderPK) {  
  
    ArrayList<productsInOrder> productList = new ArrayList<>();  
  
    connection = database.getConnection();  
    PreparedStatement ps;  
    ResultSet rs;  
    String query;  
  
    query = "select product_pk, product_name, product_sellprice, product_cost, quantity from products,orderproducts where product_pk = orderproducts.product_pk and orderproducts.order_fk = ?";  
  
    try{  
        ps = connection.prepareStatement(query);  
        ps.setInt(1, orderPK);  
        rs = ps.executeQuery();  
        productsInOrder product;  
  
        while (rs.next()) {  
            product = new productsInOrder(rs.getInt("product_pk"), rs.getString("product_name"), rs.getFloat("product_sellprice"),  
                rs.getFloat("product_cost"), rs.getInt("quantity"));  
            productList.add(product);  
        }  
    }  
    catch(SQLException ex){  
        Logger.getLogger(orderProducts.class.getName()).log(Level.SEVERE, null, ex);  
    }  
    return productList;  
}  
  
product_pk = orderproducts.product_fk and orderproducts.order_fk = ?";  
  
float("product_cost"), rs.getInt("quantity"), ((rs.getFloat("product_sellprice")-rs.getFloat("product_cost"))*rs.getInt("quantity")));
```

This method has an integer parameter orderPK this will hold the primary key of the selected order.

I have used a select query and joined all the products and orderproducts table using a comma instead of the key word JOIN. This query retrieves the primary key, name, price, and cost from the products table and the quantity from the orderproducts table where the primary key for the product matches the value in the product_fk field and where the primary key for the order held in the order_fk field matches the selected order primary key.

I have created a new instance of the productsInOrder class called product and passed in the retrieved values as seen. The last parameter for the productsInOrder constructor is the profit parameter which should hold the profit so to make this work I have subtracted the cost from the sell price and multiplied it by the quantity sold for the products.

Then I add the product to the arrayList productList.

I have tested the query works as seen below:

```
1 •  select product_pk, product_name, product_sellprice, product_cost, quantity from products,orderproducts where product_pk = orderproducts.product_fk and orderproducts.order_fk = ?";  
float("product_cost"), rs.getInt("quantity"), ((rs.getFloat("product_sellprice")-rs.getFloat("product_cost"))*rs.getInt("quantity")));
```

product_pk	product_name	product_sellprice	product_cost	quantity
23	rice	5.00	1.00	1

4.9.8 Developing fill products in order table method

Next I need to develop the fillTable method to fill the jTable with the objects in the arrayList.

```
private void fillProductsInOrderTable() {
    System.out.println("method print:" + orderPK);

    classes.productsInOrder product = new classes.productsInOrder();
    ArrayList <classes.productsInOrder> productList = product.productsInOrderList(orderPK);
    String [] columnNames = {"ID", "Name", "Price", "Cost", "Quantity", "Profit"};
    Object [][] rows = new Object [productList.size()][6];

    for(int i = 0; i<productList.size(); i++){
        rows[i][0] = productList.get(i).getProductPK();
        rows[i][1] = productList.get(i).getProductName();
        rows[i][2] = productList.get(i).getPrice();
        rows[i][3] = productList.get(i).getCost();
        rows[i][4] = productList.get(i).getQuantity();
        rows[i][5] = productList.get(i).getProfit();
    }

    classes.table model = new classes.table(rows, columnNames);
    productsInOrderTable.setModel(model);
}
```

This method also have an integer parameter to pass in the order primary key so that I can pass it into the productsInOrderList() method when I have called it.

I have used a 1d array columnNames to set the names for the columns and a 2d array rows to hold the data.

I have used a for loop to iterate through productList filling rows[][] with data from the productList as seen.

Then I create an object of the table class called model and pass in rows and columnNames and lastly, I use the setModel() method nad pass in model so set the model for the productsInOrderTable.

4.9.9 Adding code to pass the order primary key to the new screen

Now I need to pass the order primary key from the orders screen and into the productsInOrderScreen. I will do this by creating a global integer variable called orderPK in the productsInOrderScreen. I will retrieve the order primary key using the getValueAt() method for the orders table and then I will pass it into the global orderPK variable in the productsInOrderScreen.

First, I need to add an action listener to the view order button and retrieve the selected order primary key.

```
private void viewOrderButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int rowIndex = orderTable.getSelectedRow();
    int orderPK = Integer.valueOf(orderTable.getValueAt(rowIndex, 0).toString());

    productsInOrderScreen screen = new productsInOrderScreen();
    screen.setDefaultCloseOperation(productsInOrderScreen.DISPOSE_ON_CLOSE);
    screen.pack();
    screen.setVisible(true);
    screen.orderPK = orderPK;
}
```

Development and Testing

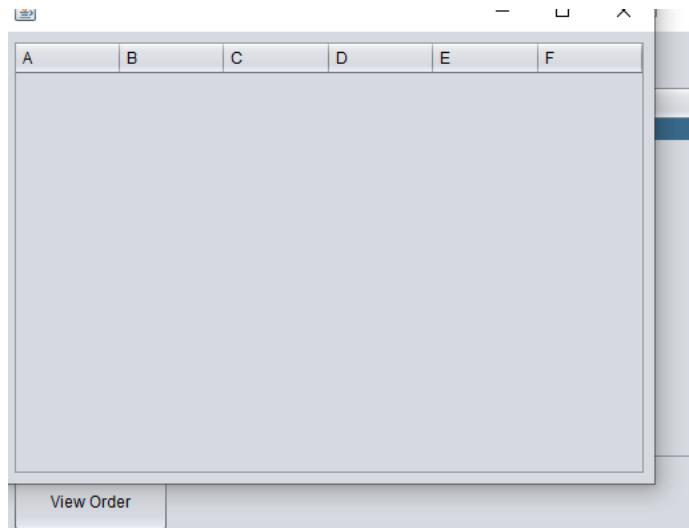
Here I have retrieved the orderPK as seen and then called the new jframe and passed the orderPK value through to the new screen by referencing the global orderPK variable in the new jframe.

Now I need to call the fillProductsInOrderTable() method in the constructor and pass in orderPK.

I have called the method as seen below:

```
public productsInOrderScreen() {
    initComponents();
    fillProductsInOrderTable(orderPK);
}
```

However, the jTable is empty:



I added some print lines to test whether the order primary key value was being passed into the method successfully.

```
constructor print:0
method print:0
66
```

The order screen printed out the order primary key successfully when I pressed the view order button however in the productInOrderScreen the value of orderPK was 0 both in the constructor and in the fill table method.

This is because I was passing the value through to the new screen after I had called the new screen. This meant that the fillTable method in the new screen was being called before the value had been passed through.

```
private void viewOrderButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int rowIndex = orderTable.getSelectedRow();
    int orderPK = Integer.valueOf(orderTable.getValueAt(rowIndex, 0).toString());

    productsInOrderScreen.orderPK = orderPK;
    productsInOrderScreen screen = new productsInOrderScreen();
    screen.setDefaultCloseOperation(productsInOrderScreen.DISPOSE_ON_CLOSE);
    screen.pack();
    screen.setVisible(true);
}
```

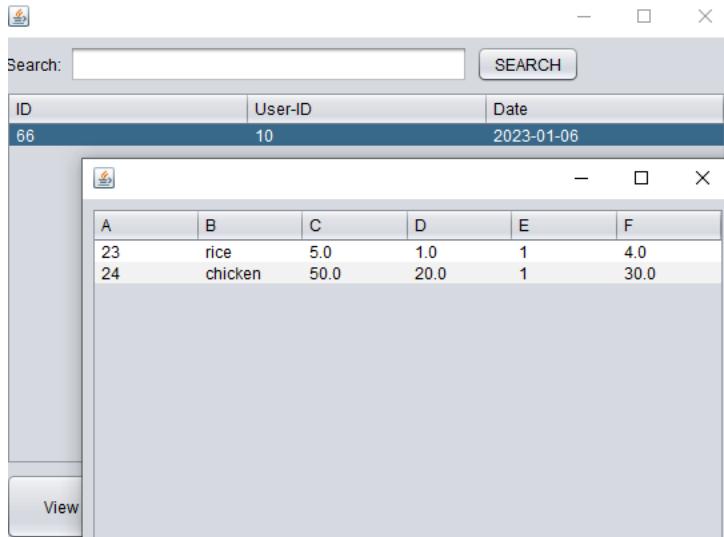
This is the new viewOrderButton code which passes the value of orderPK through to the global variable in productsInOrderScreen before I have called the productsInOrderScreen constructor.

Now the print lines successfully print the value of the selected order primary key:

Development and Testing

```
constructor print:66  
method print:66
```

And now the method works:

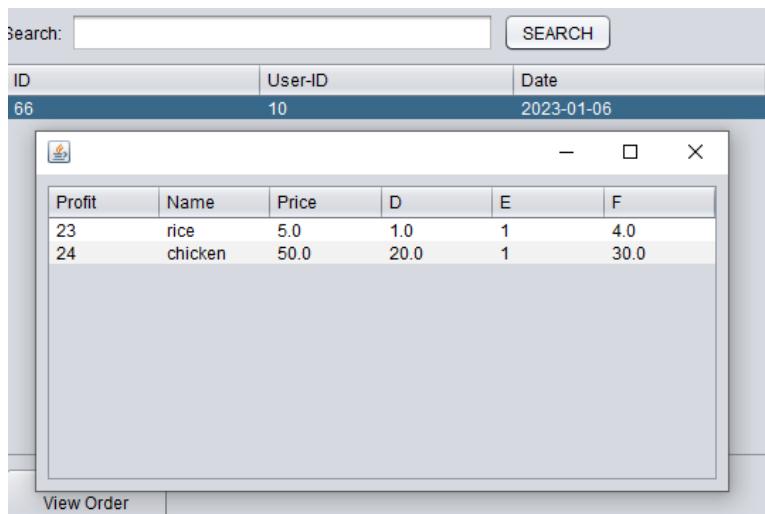


4.9.10 Developing product in order table column names method

Now I just need to develop the method to set the column names for this table same as for the other jTables.

```
public void setColumnNames() {  
  
    productsInOrderTable.getColumnModel().getColumn(0).setHeaderValue("ID");  
    productsInOrderTable.getColumnModel().getColumn(1).setHeaderValue("Name");  
    productsInOrderTable.getColumnModel().getColumn(2).setHeaderValue("Price");  
    productsInOrderTable.getColumnModel().getColumn(0).setHeaderValue("Cost");  
    productsInOrderTable.getColumnModel().getColumn(0).setHeaderValue("Quantity");  
    productsInOrderTable.getColumnModel().getColumn(0).setHeaderValue("Profit");  
  
}
```

This method worked as seen below:



4.9.11 Test Plan

Test Number	How To Test	Expected Result	How To Test
1	Create and Confirm orders	Order should be displayed with the correct price and information in the order section	Orders were displayed correctly

4.9.12 Evidence

4.9.12.1 Test 1

ORDER

ID	Name	Quantity	Price
28	rice	1	2.5

ID	User-ID	Date
68	1	2023-01-07
69	1	2023-01-13
70	1	2023-01-18
71	1	2023-01-18
72	1	2023-01-18
73	1	2023-01-18
74	1	2023-01-18
75	1	2023-01-19
76	1	2023-01-22
77	1	2023-01-22
78	1	2023-01-22
79	1	2023-01-22
80	1	2023-01-22
81	1	2023-01-22
82	1	2023-01-22

View Order

ID	Name	Price	Cost	Quantity	Profit
28	rice	2.5	0.75	1	1.75

4.9.13 Success Criteria

Success Criteria Number	Success Criteria	Achieved	Notes
6	Track Orders	Yes	I have displayed the order primary key, order date and the primary key of the user who created the order and then I have a view order button which allows the user to view more details about the order such

			as name, price and quantity and more.
--	--	--	---------------------------------------

4.9.14 Review/Stage Evaluation

This section allows the user to view details about specific orders. This section was fairly quick to code and was relatively error free, this is because this section is basically just two jTables so I could re-use the fill table methods with some slight adjustments.

First, I developed the method to fill the order table which went well and then I developed the method to retrieve all the products in the selected order from the database. In this stage I was able to fully achieve the success criteria.

4.10 Stage 9/Sales Reports/Analytics

4.10.1 Design Changes

4.10.1.1 *I added charts*

I decided to use a free Java chart library to add some charts to my reports as well as the data I was intending to report.

This is what I chose after researching. I did look at JCharts but it wasn't very easy to understand.

JFreeChart is a free chart library for Java that can generate charts for use in swing.

This is the link I used to help myself understand freechart:

<http://www.jfree.org/jfreechart/index.html>

There are lots of charts that can be created in JFreeChart but I only use the barchart and the stacked barchart types.

4.10.1.2 *I used MySQL Views instead of table*

Originally in my design I was going to use a salesReports database table and put the reports in the table and display that to the user.

I found out that MySQL has a feature called a view which is a query of tables saved in the database with its own name. I can select from the view and it will get the results of the query and return them to my Swing code. So I used the MySQL functions like SUM (sums up the values in a number column) or COUNT (counts the number of occurrences of some data). Also I can make a summary by using the GROUP feature which can group

4.10.2 Development Plan

I have found during the project that if I break up the code into small parts and do it bit by bit then it's easier to see what the code does, and I can reuse code. Also that for something like screens they are all pretty similar so I can take the same approach for more than one part of the system. So I am going to make the reports all using similar approaches as they all have common parts like Enter the Report parameters screen, Make the dataset, Make the chart from the dataset, Make the data table from the dataset to show the report data, and put the chart and the data table together in a JFrame to show to the user. Also, I will develop the SQL for the reports as views so each report has its own

view. I can use MySQL functions to calculate the report totals and the view can join up all the tables required so this is simpler than making the queries inside the Java.

Also, each view is like a table but read only so I have made a class for each view object just like for each table object in the POS system screens. The Java forms use the table classes to add or delete data from the tables and the table classes have the methods to delete or update or insert for each table. For the reports each view has a class but it just has methods for the report so they are just to get data from the view and make the charts and put it altogether in a frame and then make this visible.

4.10.3 Developing Report Views in SQL

There are 3 reports to develop, profit by month, most sold product by month, and total number of orders by month.

4.10.3.1 Profit by month SQL query

The database tables I will join are the orders table and the products table and the orderproducts table.



```
1 •  SELECT
2   MONTH(o.date), YEAR(o.date),
3   order_pk,
4   product_pk,
5   product_cost,
6   product_sellprice,
7   product_sellprice-product_cost
8   FROM
9   orders o, products p, orderproducts op
10  WHERE
11  o.order_pk=op.order_fk
12  AND
13  op.product_fk=p.product_pk
14  ORDER BY
15  order_pk,
16  product_pk
17
```

This gives me every order for every month with all the products per order and the product details like cost and price.

Development and Testing

	MONTH(o.date)	YEAR(o.date)	order_pk	product_pk	product_cost	product_sellprice	product_sellprice-product_cost
▶	1	2023	15	23	1.00	5.00	4.00
	1	2023	16	23	1.00	5.00	4.00
	1	2023	17	23	1.00	5.00	4.00
	1	2023	18	24	20.00	50.00	30.00
	1	2023	19	25	12.00	25.00	13.00
	1	2023	20	23	1.00	5.00	4.00
	1	2023	21	23	1.00	5.00	4.00
	1	2023	22	25	12.00	25.00	13.00
	1	2023	23	23	1.00	5.00	4.00
	1	2023	24	23	1.00	5.00	4.00
	1	2023	25	23	1.00	5.00	4.00
	1	2023	26	24	20.00	50.00	30.00
	1	2023	28	23	1.00	5.00	4.00
	1	2023	29	24	20.00	50.00	30.00
	1	2023	30	25	12.00	25.00	13.00
	2	2023	31	23	1.00	5.00	4.00
	2	2023	31	23	1.00	5.00	4.00
	2	2023	31	24	20.00	50.00	30.00
	2	2023	31	25	12.00	25.00	13.00
	3	2023	32	23	1.00	5.00	4.00
	3	2023	32	24	20.00	50.00	30.00

The profit is calculated in the query by subtracting cost from price.

I used the SQL concat() function to put the YYYY and the MM part together so the user can enter the month to report on like 2023-1 for January 2023.

```

19  SELECT
20    CONCAT(YEAR(o.date),"-", MONTH(o.date)) mymonth,
21    order_pk,
22    product_pk,
23    product_cost,
24    product_sellprice,
25    product_sellprice-product_cost
26  FROM
27    orders o, products p, orderproducts op
28  WHERE
29    o.order_pk=op.order_fk
30  AND
31    op.product_fk=p.product_pk
32  ORDER BY
33    order_pk,
34    product_pk
35

```

	mymonth	order_pk	product_pk	product_cost	product_sellprice	product_sellprice-product_cost
▶	2023-1	15	23	1.00	5.00	4.00
	2023-1	16	23	1.00	5.00	4.00
	2023-1	17	23	1.00	5.00	4.00
	2023-1	18	24	20.00	50.00	30.00

Next I take out the order primary key and the product primary key in the SELECT clause because I want to add up all the cost and price and profit for the entire month and not have it split out for each order and orderproduct. I GROUP BY the YYYY-MM so I get a total for each month.

Development and Testing

```
37  SELECT
38    CONCAT(YEAR(o.date),"-", MONTH(o.date)) mymonth,
39    -- order_pk,
40    -- product_pk,
41    SUM(product_cost) total_cost,
42    SUM(product_sellprice) total_price,
43    SUM(product_sellprice-product_cost) total_profit
44  FROM
45  orders o, products p, orderproducts op
46  WHERE
47  o.order_pk=op.order_fk
48  AND
49  op.product_fk=p.product_pk
50  GROUP BY
51  CONCAT(YEAR(o.date),"-", MONTH(o.date))
52  ORDER BY
53  CONCAT(YEAR(o.date),"-", MONTH(o.date))
```

result Grid			
mymonth	total_cost	total_price	total_profit
2023-1	105.00	270.00	165.00
2023-2	34.00	85.00	51.00
2023-3	21.00	55.00	34.00

So, this is the query for the Profit By Month report it shows the totals for the 3 months that I have orders for.

Development and Testing

I can make this into a view so it is saved in the database and I can use this view to retrieve the data in the Java code to make the report by selecting from the view instead of having the full SQL

```
16    CREATE VIEW profitbymonth_view AS
17    SELECT
18        CONCAT(YEAR(o.date), "-", MONTH(o.date)) mymonth,
19        -- order_pk,
20        -- product_pk,
21        SUM(product_cost) total_cost,
22        SUM(product_sellprice) total_price,
23        SUM(product_sellprice - product_cost) total_profit
24    FROM
25        orders o, products p, orderproducts op
26    WHERE
27        o.order_pk=op.order_fk
28        AND
29        op.product_fk=p.product_pk
30    GROUP BY
31        CONCAT(YEAR(o.date), "-", MONTH(o.date))
32    ORDER BY
33        CONCAT(YEAR(o.date), "-", MONTH(o.date))
34
35    select * from profitbymonth_view
36
```

Result Grid | Filter Rows: [] | Export: [] | Wrap Cell Content: []

mymonth	total_cost	total_price	total_profit
2023-1	105.00	270.00	165.00
2023-2	34.00	85.00	51.00
2023-3	21.00	55.00	34.00

And for the report screen I will build it so the user can enter in a month to filter the results to a month.

```
54
55    select * from profitbymonth_view where mymonth = "2023-3"
56
```

Result Grid | Filter Rows: [] | Export: [] | Wrap Cell Content: []

mymonth	total_cost	total_price	total_profit
2023-3	21.00	55.00	34.00

Or I will make it so the user can filter between a start month and an end month.

```
55    select * from profitbymonth_view where mymonth = "2023-3"
56    select * from profitbymonth_view where mymonth between "2023-1" and "2023-2"
57
58
59    Result Grid | Filter Rows: [ ] | Export: [ ] | Wrap Cell Content: [ ]
60
61    mymonth total_cost total_price total_profit
62    2023-1 105.00 270.00 165.00
63    2023-2 34.00 85.00 51.00
64
65    select * from profitbymonth_view where mymonth = "2023-3"
66    select * from profitbymonth_view where mymonth between "2023-1" and "2023-2"
67
68
69    Result Grid | Filter Rows: [ ] | Export: [ ] | Wrap Cell Content: [ ]
70
71    mymonth total_cost total_price total_profit
72    2023-1 105.00 270.00 165.00
73    2023-2 34.00 85.00 51.00
```

4.10.3.2 Most sold product by month SQL query

This query is pretty similar to my profit by month SQL. The database tables I will join are the orders table and the products table and the orderproducts table.

First I get the quantity ordered for every product in the order and then I will add up all the quantities for each product on the order.

```

SELECT
CONCAT(YEAR(o.date),"-", MONTH(o.date)) mymonth,
order_pk,
product_fk,
CONCAT(product_pk,"-",product_name) product,
op.quantity
FROM
orders o, products p, orderproducts op
WHERE
o.order_pk=op.order_fk
AND
op.product_fk=p.product_pk
ORDER BY
CONCAT(YEAR(o.date),"-", MONTH(o.date))

```

I used the CONCAT function again but this time on product key and product name because I found my test data allows duplicate product names. I put this in the Problems section below it can be fixed with a unique index on product name but for now I will join product key and name together so the report shows product keys unique and doesn't add duplicate product names together.

	mymonth	order_pk	product_fk	product	quantity
▶	2023-1	15	23	23-rice	1
	2023-1	16	23	23-rice	1
	2023-1	17	23	23-rice	1
	2023-1	18	24	24-chicken	1
	2023-1	19	25	25-duck	1
	2023-1	20	23	23-rice	1
	2023-1	21	23	23-rice	1
	2023-1	22	25	25-duck	1
	2023-1	23	23	23-rice	1
	2023-1	24	23	23-rice	1
	2023-1	25	23	23-rice	1
	2023-1	26	24	24-chicken	1
	2023-1	28	23	23-rice	1
	2023-1	29	24	24-chicken	1
	2023-1	30	25	25-duck	1
	2023-2	31	23	23-rice	1
	2023-2	31	24	24-chicken	1
	2023-2	31	25	25-duck	1
	2023-2	31	23	23-rice	5
	2023-3	32	23	23-rice	1
	2023-3	32	24	24-chicken	1

Also there might be a good reason to have product names the same but with different categories so for now the product unique key makes the report accurate as it is supposed to show the most sold products by month and the product key makes each product a different record so it should be added up using the unique key and not grouped by the product name.

Development and Testing

```
SELECT
    CONCAT(YEAR(o.date), "-", MONTH(o.date)) mymonth,
    -- order_pk,
    -- product_fk,
    CONCAT(product_pk, "-", product_name) product,
    SUM(op.quantity) total_quantity_sold
FROM
    orders o, products p, orderproducts op
WHERE
    o.order_pk=op.order_fk
    AND
    op.product_fk=p.product_pk
GROUP BY
    CONCAT(YEAR(o.date), "-", MONTH(o.date)), product_pk
ORDER BY
    CONCAT(YEAR(o.date), "-", MONTH(o.date)),
    SUM(op.quantity) DESC
```

I use the SUM function in MySQL to add up the quantity of each product in each order and use the GROUP function to group the results by product on each order so no matter how many times a product is on an order in the order_products table I add all the quantities together for all the order products record for the order to get the total sold.

	mymonth	product	total_quantity_sold
▶	2023-1	23-rice	9
	2023-1	24-chicken	3
	2023-1	25-duck	3
	2023-2	23-rice	6
	2023-2	24-chicken	1
	2023-2	25-duck	1
	2023-3	23-rice	1
	2023-3	24-chicken	1

I ordered the results by the total quantity so the most sold product comes first for each of the months and made it into a view.

```
CREATE VIEW mostsoldbymonth_view AS
SELECT
    CONCAT(YEAR(o.date), "-", MONTH(o.date)) mymonth,
    -- order_pk,
    -- product_fk,
    CONCAT(product_pk, "-", product_name) product,
    SUM(op.quantity) total_quantity_sold
FROM
    orders o, products p, orderproducts op
WHERE
    o.order_pk=op.order_fk
    AND
    op.product_fk=p.product_pk
GROUP BY
    CONCAT(YEAR(o.date), "-", MONTH(o.date)), product_pk
ORDER BY
    CONCAT(YEAR(o.date), "-", MONTH(o.date)),
    SUM(op.quantity) DESC
```

Development and Testing

Now I can select for a month or for months between a month range.

```
115 •   select * from mostsoldbymonth_view;
116 •   select * from mostsoldbymonth_view where mymonth = "2023-3";
117 •   select * from mostsoldbymonth_view where mymonth between "2023-1" and "2023-2";
```

	mymonth	product	total_quantity_sold
▶	2023-1	24-chicken	3
	2023-1	25-duck	3
	2023-1	23-rice	9
	2023-2	24-chicken	1
	2023-2	25-duck	1
	2023-2	23-rice	6
	2023-3	23-rice	1
	2023-3	24-chicken	1

Result Grid | Filter Rows: | Export:

	mymonth	product	total_quantity_sold
▶	2023-3	23-rice	1
	2023-3	24-chicken	1

Result Grid | Filter Rows: | Export:

	mymonth	product	total_quantity_sold
▶	2023-1	24-chicken	3
	2023-1	25-duck	3
	2023-1	23-rice	9
	2023-2	24-chicken	1
	2023-2	25-duck	1
	2023-2	23-rice	6

So the user can order it by total_quantity_sold and the first product for each month will be the most sold product for that month.

```
•   select * from mostsoldbymonth_view order by mymonth ASC, total_quantity_sold DESC;
```

Result Grid | Filter Rows: |

	mymonth	product	total_quantity_sold
▶	2023-1	23-rice	9
	2023-1	24-chicken	3
	2023-1	25-duck	3
	2023-2	23-rice	6
	2023-2	24-chicken	1
	2023-2	25-duck	1
	2023-3	23-rice	1
	2023-3	24-chicken	1

4.10.3.3 Total number of orders by month SQL query

This query is simpler than my profit by month SQL. The database table is just the orders table.

First I get all the orders along with the year and month.

- **SELECT**

```
CONCAT(YEAR(o.date),"-", MONTH(o.date)) mymonth, order_pk
FROM
orders o
ORDER BY
CONCAT(YEAR(o.date),"-", MONTH(o.date));
```

Result Grid	
mymonth	order_pk
2023-1	16
2023-1	17
2023-1	18
2023-1	19
2023-1	20
2023-1	21
2023-1	22
2023-1	23
2023-1	24
2023-1	25
2023-1	26
2023-1	27
2023-1	28
2023-1	29
2023-1	30
2023-2	31
2023-3	32

That's all the orders with their year and month next to them. Now I can use the COUNT() function in MySQL to count all the orders for each month by GROUP BY the mymonth column.

```
SELECT
CONCAT(YEAR(o.date),"-", MONTH(o.date)) mymonth,
COUNT(order_pk) total_orders
FROM
orders o
GROUP BY
CONCAT(YEAR(o.date),"-", MONTH(o.date))
ORDER BY
CONCAT(YEAR(o.date),"-", MONTH(o.date));
```

That gives me the months with a total orders column.

Result Grid	
mymonth	total_orders
2023-1	27
2023-2	1
2023-3	1

Development and Testing

I made the SQL into report view so I can select from it in the Java code.

```

• CREATE VIEW totalordersbymonth_view AS
SELECT
    CONCAT(YEAR(o.date), "-", MONTH(o.date)) mymonth,
    COUNT(order_pk) total_orders
FROM
    orders o
GROUP BY
    CONCAT(YEAR(o.date), "-", MONTH(o.date))
ORDER BY
    CONCAT(YEAR(o.date), "-", MONTH(o.date));

```

```

-- 
139 •   select * from totalordersbymonth_view
140

```

	mymonth	total_orders
▶	2023-1	27
	2023-2	1
	2023-3	1

4.10.4 Problems

4.10.4.1 Products can have same name

My productScreen lets users add products with the same name. I need to add a unique constraint on product_name so each product is unique in the mostSoldProduct by month report.

```
CREATE INDEX idx1 ON t1 ((col1 + col2));
```

```
0      107      12:18:20      CREATE UNIQUE INDEX products_name_1 ON products
(product_name)      Error Code: 1062. Duplicate entry 'jTextField1' for key
'products.products_name_1'  0.031 sec
```

I tried but my test data that I set up has got a lot of duplicates.

product_pk	product_name	product_description	product_sellprice	product_cost	createdby_fk	createdon	lastupdated	deletedflag	category	stock	category_fk
14	StableTest	jTextField2	50.55	35.55	10	2022-12-28 18:53:43	2022-12-28 18:53:43	1	searchCategoryTest	5	22
15	jTextField1	jTextField2	50.55	35.55	10	2022-12-28 19:02:38	2022-12-28 19:02:38	1	searchCategoryTest	5	22
16	success	jTextField2	50.55	35.55	10	2022-12-28 19:24:25	2022-12-28 19:29:23	1	searchCategoryTest	5	22
17	addProductTest	jTextField2	50.55	35.55	10	2022-12-29 21:36:47	2022-12-29 21:36:47	1	searchCategoryTest	5	22
18	updateProductTest	jTextField2	50.55	35.55	10	2022-12-29 21:37:10	2022-12-29 21:38:06	1	searchCategoryTest	5	22
19	searchProductTest	jTextField2	50.55	35.55	10	2022-12-29 21:40:11	2022-12-30 13:38:44	1	searchCategoryTest	0	22
20	testProduct	jTextField2	50.55	35.55	10	2022-12-29 21:40:39	2022-12-30 13:45:38	1	searchCategoryTest	5	22
21	jTextField1	jTextField2	50.55	35.55	10	2022-12-30 13:40:21	2022-12-30 13:40:21	1	searchCategoryTest	5	22
22	jTextField1	jTextField2	50.55	35.55	10	2022-12-30 18:19:25	2022-12-30 18:19:25	1	test	10	20
23	rice	jasmine rice	5.00	1.00	10	2023-01-03 16:55:46	2023-01-03 16:55:46	0	test	5	20
24	chicken	chicken	50.00	20.00	10	2023-01-03 16:56:09	2023-01-03 16:56:09	0	disposeMethodTest	5	24
25	duck	duck tamarind	25.00	12.00	10	2023-01-03 16:56:58	2023-01-03 16:56:58	0	searchCategoryTest	10	22
26	Correct name	Valid description	1.99	0.85	10	2023-01-08 14:23:16	2023-01-08 14:23:16	0	test	300	20

ducts 1 x

put :::::

Action	Output
#	Time Action
107	12:18:20 CREATE UNIQUE INDEX products_name_1 ON products (product_name)
108	12:25:05 SELECT * FROM posplex.products LIMIT 0, 1000

Message
Error Code: 1062. Duplicate entry 'jTextField1' for key 'products.products_name_1'
20 row(s) returned

Having products with the same name is unnecessary and confusing and breaks my query from working so I will join the product key to the product name so it becomes unique and then my query will work ok.

Development and Testing

```

SELECT
    CONCAT(YEAR(o.date),"-", MONTH(o.date)) mymonth,
    order_pk,
    product_fk,
    CONCAT(product_pk,"-",product_name) product,
    op.quantity
FROM
    orders o, products p, orderproducts op
WHERE
    o.order_pk=op.order_fk
    AND
    op.product_fk=p.product_pk
ORDER BY
    CONCAT(YEAR(o.date),"-", MONTH(o.date))

```

Result Grid | Filter Rows: | Export:

mymonth	order_pk	product_fk	product	quantity
2023-1	21	23	23-rice	1
2023-1	22	25	25-duck	1
2023-1	23	23	23-rice	1
2023-1	24	23	23-rice	1
2023-1	25	23	23-rice	1
2023-1	26	24	24-chicken	1
2023-1	28	23	23-rice	1
2023-1	29	24	24-chicken	1
2023-1	30	25	25-duck	1
2023-2	31	23	23-rice	1
2023-2	31	24	24-chicken	1
2023-2	31	25	25-duck	1
2023-2	31	23	23-rice	5
2023-3	32	23	23-rice	1
2023-3	32	24	24-chicken	1

Result 43 ×

4.10.4.2 Prepare statement can't bind the SQL ORDER BY column

I wanted to let the user change the SQL dynamically depending on what they chose in the parameter form for what to order by. So in my method to make the chart I pass in the orderByVal that is the column the user chose to order the report.

This method takes the orderByVal to make the SQL that will order the results by the order by column.

```

//make the chart
ChartPanel chartpanel = classes.ProfitByMonth_view.ProfitByMonthPanel(
    startDate, endDate, metricIn, orderbyVal);

```

Development and Testing

When I prepare the statement as a SQL string I bind the third variable ? in the ORDER BY to the orderbyVal column the user chose.

```
//create the chart but return it in panel so it can be assembled into the final report
public static ChartPanel ProfitByMonthPanel(
    String startDate, String endDate, String metricIn, String orderbyVal)
{
    Connection connect = database.getConnection();

    PreparedStatement ps;
    ResultSet rs;

    String mymonth="";
    float metric=0;

    //query to retrieve data from the db view where the category is like the report param passed in val
    String query = "SELECT * FROM ProfitByMonth_view WHERE mymonth BETWEEN ? AND ? ORDER BY ?";

    DefaultCategoryDataset dataset = new DefaultCategoryDataset();

    try{
        ps = connect.prepareStatement(query);

        ps.setString(1, startDate);
        ps.setString(2, endDate);
        ps.setString(3, orderbyVal);

        System.out.println(ps);

        rs = ps.executeQuery();

        while (rs.next()){

    
```

But in the prepared statement if I output it to debug it then it looks like this.

```
com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM ProfitByMonth_view WHERE mymonth BETWEEN '2023-1' AND '2023-3' ORDER BY 'total_cost'
sessionID: 665
userPrimaryKey: 10
userType: Admin
665
Admin
access granted
2023-1
2023-3
total_profit
total_profit
com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM ProfitByMonth_view WHERE mymonth BETWEEN '2023-1' AND '2023-3' ORDER BY 'total_profit'
```

This won't work as in SQL it will just order by the string 'total_profit' and I need it to order by the column name total_profit not as a string 'total_profit'.

So I made the string SQL query just to set the 2 variables startDate and endDate in the SQL before I prepare the statement. And before I set the variables in the SQL string I have made a method to add the ORDER BY column to the SQL string as part of the string, not by binding it to a variable using the ? and setString() method.

```
//query to retrieve data from the db view where the category is like the report param passed in val
String query = ProfitByMonth_viewSQL(orderbyVal);

DefaultCategoryDataset dataset = new DefaultCategoryDataset();

try{
    ps = connect.prepareStatement(query);

    ps.setString(1, startDate);
    ps.setString(2, endDate);

    System.out.println(ps);
```

And I made a method that makes the SQL string with the order by column added to the string.

Development and Testing

```
//private method just to make the SQL for my report ProfitByMonth
private static String ProfitByMonth_viewSQL(String orderbyVal)
{
    //query to retrieve data from the db view where the category is like the report param passed in val
    String query = "SELECT * FROM ProfitByMonth_view WHERE mymonth BETWEEN ? AND ? ORDER BY ?";

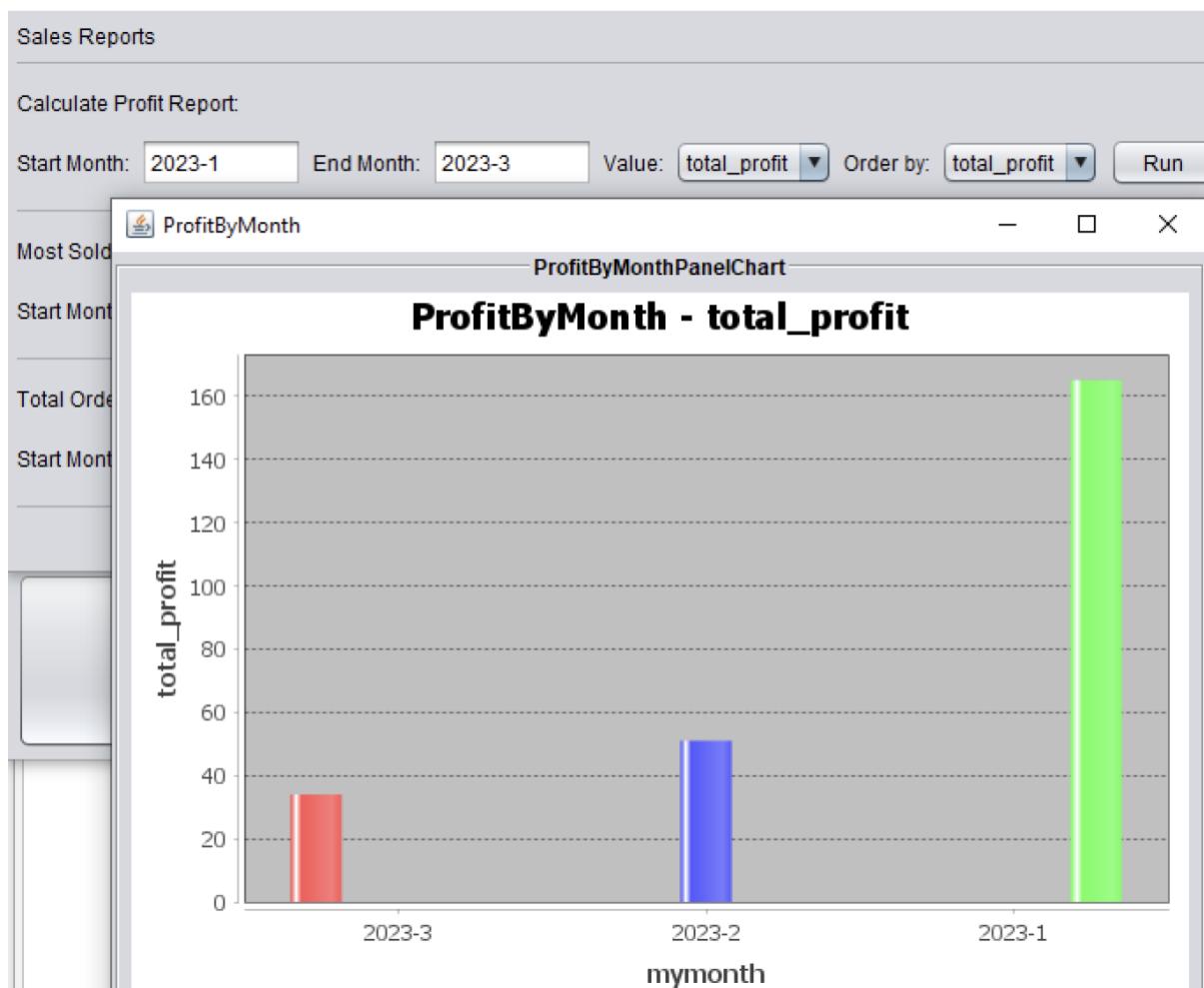
    //append the correct ORDER BY column for the query based on the orderbyVal param
    //note this can in some cases be a risk for SQL injection but PreparedStatement doesn't allow ORDER BY ?
    if (orderbyVal=="mymonth") {
        query=query+"mymonth;";
    } else if (orderbyVal=="total_cost") {
        query=query+"total_cost;";
    } else if (orderbyVal=="total_price") {
        query=query+"total_price;";
    } else if (orderbyVal=="total_profit") {
        query=query+"total_profit;";
    } else {
        query=query+"mymonth;"; //remove SQL injection risk by examining the orderbyVal and only allowing mymonth column name
    }

    return query;
}
```

Now when I run it I have added the right column name to the end of the ORDER BY clause.

```
com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM ProfitByMonth_view WHERE mymonth BETWEEN '2023-1' AND '2023-3' ORDER BY mymonth;
sessionId: 666
userPrimaryKey: 10
userType: Admin
666
Admin
access granted
2023-1
2023-3
total_profit
total_profit
com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM ProfitByMonth_view WHERE mymonth BETWEEN '2023-1' AND '2023-3' ORDER BY total_profit;
```

So now the ORDER BY is by the column name not the column name in “”.



And I can see the chart is ordered by the total_profit starting lowest to highest months. When I looked into how to develop this to work I saw people talking about SQL injection and the problem that if my system builds SQL statements from strings that the UI sends to the database class then there is a hack called “SQL injection” that hackers use to send bad SQL to the system to run against the database. So I made some changes to how I make the SQL string and documented these below.

4.10.4.3 Stopping SQL injection

During my development of the reports section I learned that if I make SQL statements from data sent by the user then there is a risk called “SQL injection” where a malicious user or hacker might send SQL to the system that is intended to damage the database.

I have coded it like the recommended way. This means to code it so the system only accepts valid strings that will be used to make the SQL and would never accept anything outside of the allowed strings like if the hacker sent the ORDER BY column not just as “mymonth” but as “mymonth; DELETE database posplex;” this would delete the entire database after the SELECT statement has run.

My method only allows the orderByVal to be mymonth or total_cost or total_price or total_profit anything else and it will be set to mymonth.

```

String query = "SELECT * FROM ProfitByMonth_view WHERE mymonth BETWEEN ? AND ? ORDER BY ";
//append the correct ORDER BY column for the query based on the orderbyVal param
//note this can in some cases be a risk for SQL injection but PreparedStatement doesn't allow ORDER BY ?
if (orderbyVal=="mymonth") {
    query=query+"mymonth;";
} else if (orderbyVal=="total_cost") {
    query=query+"total_cost;";
} else if (orderbyVal=="total_price") {
    query=query+"total_price;";
} else if (orderbyVal=="total_profit") {
    query=query+"total_profit;";
} else {
    query=query+"mymonth;"; //remove SQL injection risk by examining the orderbyVal and only allowing mymonth column name
}

```

The hacker would have to make a Swing report calling form that let them change the ORDER BY combo box so they could pass in the ORDER BY column_name and also a ; and another SQL command but that's not impossible. Probably it is more of a risk for a web based application where a hacker could code a web page that posted the hacked SQL to the Swing class.

4.10.4.4 Laying out the report

For my reports they are not like my forms which are built using the Netbeans UI design tool so the JComponents that I am using to show the report are all generated in the program as it runs.

So I had to fix the size of the charts in the Java as I couldn't get it to resize the frame to fit whatever screen size for the user. I think this might not work so well if different hardware is used to run the system and with different screen sizes.

```

//make the chart
ChartPanel chartpanel = classes.ProfitByMonth_view.ProfitByMonthPanel(
    startDate, endDate, metricIn, orderbyVal);
//set the size of the chart
chartpanel.setPreferredSize(new Dimension(chartWidth, chartHeight));

```

Also I had to do this with the JTable that shows the report data under the chart to set the size.

```

//make the data table
JTable jTableData = classes.ProfitByMonth_view.fillProfitByMonth_viewTable(startDate, endDate, orderbyVal);
//set the size of the table
jTableData.setPreferredSize(new Dimension(tableWidth, tableHeight));

```

Next I had a problem with adding components to the frame to show the report because it isn't like the Netbeans UI editor that takes care of the way the JComponents are laid out. So it was just laying out the chart and the data on top of each other so hiding the first panel I added to the frame with the second panel I added. So I looked into the GridBagLayout manager and I used this in the end to make the report appear OK.

The GridBagLayout manager uses a grid with columns and rows dividing the frame area and lets me select where to place components and also how many rows and columns.

To start with the Swing example for GridBagLayout explains how to add JComponents to a JPanel which is the container pane in a JFrame so I read up on it and developed it to work for my report frame.

I looked up the Oracle Swing docs for how to lay out components on a JFrame and there are a few options.

- BorderLayout
 - BoxLayout
 - CardLayout
 - FlowLayout
 - GridBagLayout
 - GridLayout
 - GroupLayout
 - SpringLayout

In the notes it says “Note: This lesson covers writing layout code by hand, which can be challenging. If you are not interested in learning all the details of layout management, you might prefer to use the GroupLayout layout manager combined with a builder tool to lay out your GUI. One such builder tool is the NetBeans IDE. Otherwise, if you want to code by hand and do not want to use GroupLayout, then GridBagLayout is recommended as the next most flexible and powerful layout manager.”

Development and Testing

So I decided to use GridLayout layout manager.

In my assemble method `fillProfitByMonth_viewAssemble` I made the chart and the data table and put them in their own JPanels.

```
//assembler for the report - this gets the chart and the data table and lays them out using GridLayout layout manager
public static JFrame fillProfitByMonth_viewAssemble(String startDate, String endDate, String metricIn, String orderbyVal,
| | | int chartWidth, int chartHeight, int tableWidth, int tableHeight, String frameTitle, String chartTitle, String tableTitle)
{
    //make the chart
```

I added the chart to a new JPanel.

```
//make the chart
ChartPanel chartpanel = classes.ProfitByMonth_view.ProfitByMonthPanel(
| | | startDate, endDate, metricIn, orderbyVal);
//set the size of the chart
chartpanel.setPreferredSize(new Dimension(chartWidth, chartHeight));
JPanel jPanelChart = new JPanel();
jPanelChart.add(chartpanel);
jPanelChart.setBorder(BorderFactory.createTitledBorder( BorderFactory.createEtchedBorder(), chartTitle, TitledBorder.CENTER, TitledBorder.TOP));
```

Then I added the JTable to a new JPanel.

```
//make the data table
JTable jTableData = classes.ProfitByMonth_view.fillProfitByMonth_viewTable(startDate, endDate, orderbyVal);
//set the size of the table
jTableData.setPreferredSize(new Dimension(tableWidth, tableHeight));
JPanel jPanelData = new JPanel();
jPanelData.add(jTableData);
jPanelData.setBorder(BorderFactory.createTitledBorder( BorderFactory.createEtchedBorder(), tableTitle, TitledBorder.CENTER, TitledBorder.TOP));
```

I made a method `addComponentsToPane`(Container pane, JPanel chartpanel, JPanel datapanel) based on how I read up on how to use GridLayout manager. First I made the new JFrame where I will use the GridBag layout manager to layout the chart panel and the data panel.

```
//new frame to show the GridLayoutDemo for the POS reports
JFrame frame = new JFrame(frameTitle);

//Set up the content pane by adding the chart panel and the data panel
//GridLayout manager time!
classes.ProfitByMonth_view.addComponentsToPane(frame.getContentPane(), jPanelChart, jPanelData);
```

Then I called my `addComponentsToPane`(Container pane, JPanel chartpanel, JPanel datapanel) method and passed in the content pane of my new JFrame using the `getContentPane()` method, also I passed in the chart in the JPanel and the data in the other JPanel that I made.

The first thing I do in my `addComponentsToPane` method is set the flow of the components direction – I set it based on my constant `RIGHT_TO_LEFT` which I already set to false so the pane will be laid out LEFT TO RIGHT which is the default language setting anyway although some languages go RIGHT TO LEFT.

```
//I used GridLayout manager as I need to dynamically lay out the chart and the data table
//if I didn't use this layout manager then its really hard to make the JComponents go where I need
//they just overlap themselves
public static void addComponentsToPane(Container pane, JPanel chartpanel, JPanel datapanel) {
    if (RIGHT_TO_LEFT) {
        pane.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);
    }
}
```

Then I use the `setLayout` method to set the layout of my container which is the content pane of my JFrame to tell it I will be using the GridLayout so when I add my panels to the pane I can lay them out using GridBag constraints.

```
pane.setLayout(new GridBagLayout());
GridBagConstraints c = new GridBagConstraints();
```

Now I have a GridBagConstraints object that is set by the constructor GridBagConstraints() to the default constraints for a GridBagLayout. So now I can set the settings for the layout by setting the constraints each time before I add a component to the container pane.

So here I tell GridBagLayout to fill the cell in the grid horizontally so use the height of the chart panel as it is and fill horizontally as required.

I set gridwidth to 1 as this is the number of columns and I need 1 column. If I had 3 columns I could set this to 3.

I set gridx to 0 as there is only one column, if I had 3 columns I could set gridx to 2 and my chart panel would appear in the middle column.

I set gridy to 0 which is the first row in my grid layout.

```
//layout the chartpanel passed in <<<<<<<<<<<<<<<<<<
c.fill = GridBagConstraints.HORIZONTAL;
c.weightx = 0.0;
c.gridx = 1;
c.gridy = 0;
pane.add(chartpanel, c);
```

Finally I use the Container.add() method to add my chartpanel to my pane and also pass in the GridBag constraints I have just set.

Next I want to make the datapanel which is a JTable with my report data lay out below the chart.

I change gridy from 0 to 1. This tells the layout manager to put the datapanel on the row below the chartpanel before.

```
//layout the data panel passed in <<<<<<<<<<<<<<<<<<
c.fill = GridBagConstraints.HORIZONTAL;
c.weightx = 0.0;
c.gridx = 1;
c.gridy = 1;
pane.add(datapanel, c);
```

Finally I use the Container.add() method to add my datapanel to my pane and also pass in the GridBag constraints I have just set. So each time before adding a component to a container using GridBag I set the constraints for that component which tells the layout manager how to lay out that component.

Development and Testing

4.10.4.4.1 My original code without a layout manager

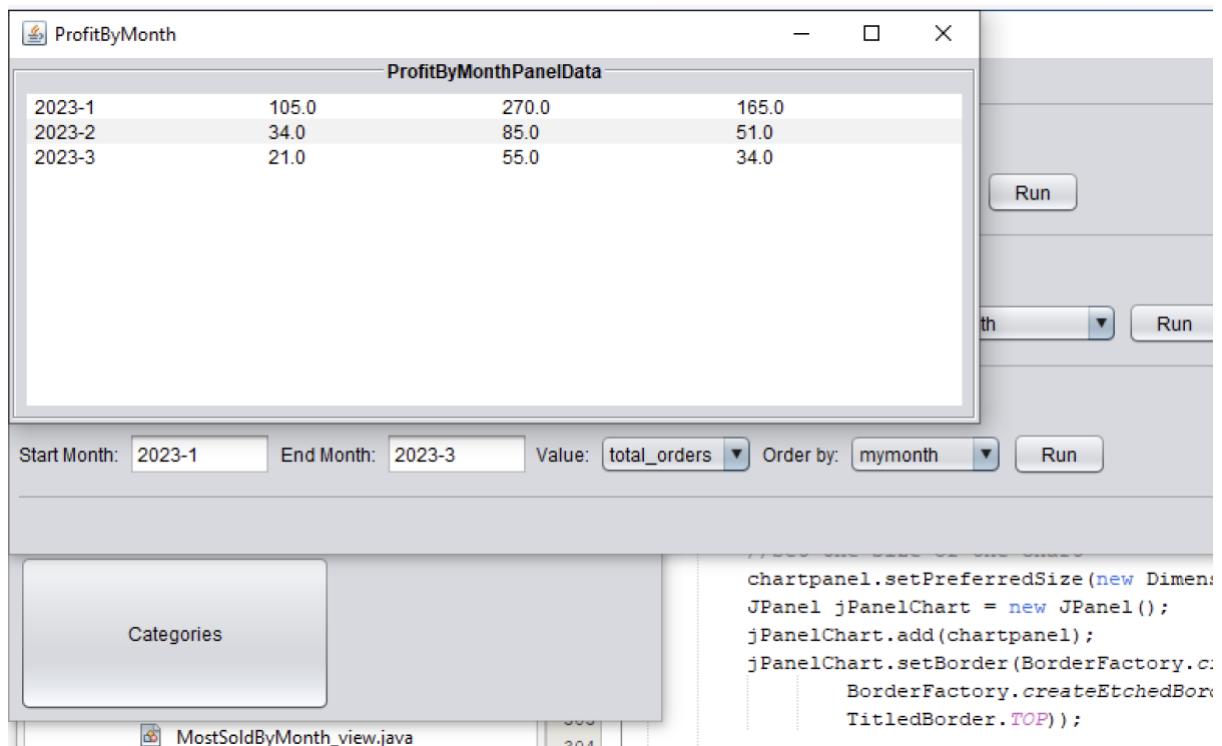
This just adds the JPanels to the new JFrame and I thought it would just add them one after another so they were both visible but it doesn't.

```
//add the chart and data to the frame to return
JFrame frame = new JFrame(frameTitle);
frame.add(jPanelChart);
frame.add(jPanelData);

//return the assembled report
return frame;

}
```

It puts the panels on top of each other or actually I think it completely replaces the first pane; with the second panel so when the .add() method runs it makes a new content pane.



So this is when I read up on the Layout Managers available in Swing.

Development and Testing

4.10.4.4.2 My code using the GridBagLayout manager approach

This code uses the GridBagLayout manager approach.

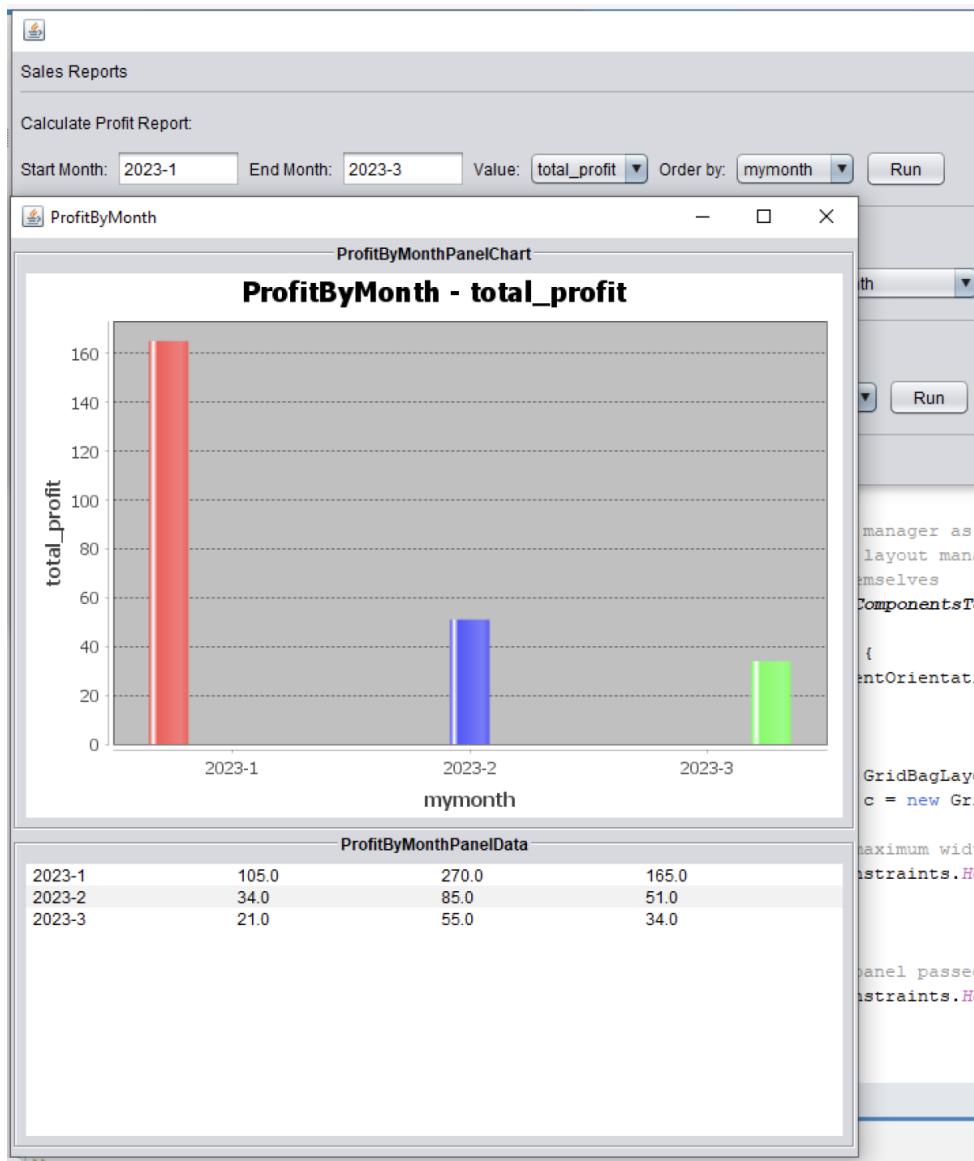
```
//add the chart and data to the frame to return
JFrame frame = new JFrame(frameTitle);

//Set up the content pane by adding the chart panel and the data panel
//GridBagLayout manager time!
classes.ProfitByMonth_view.addComponentstoPane(frame.getContentPane(), jPanelChart, jPanelData);

//return the assembled report
return frame;

}
```

So now I have solved my problem because I can see my JFrame has a grid with one column and two rows and the chart is in row one and the data is in row two.



4.10.5 Developing Report Parameter Form

This JForm will run the reports based on the parameters the user enters on the form. This screen is called from the menuScreen form.

There are 3 reports all of them report data by month so I can make the reports very similar.

I designed the screen like this to be the same number of parameters for each report although the actual parameters for each report are different because they use different views with different columns being produced.

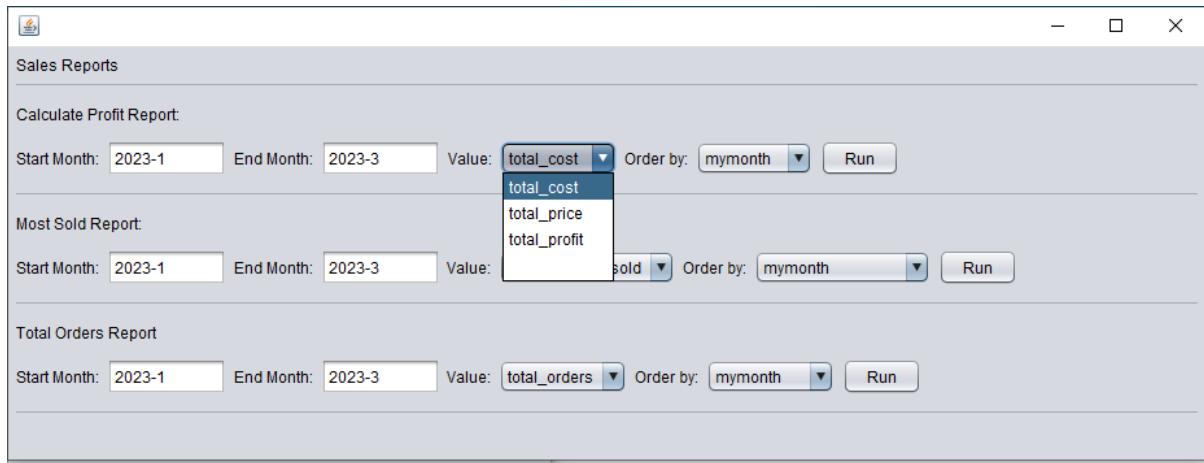
The screenshot shows a JForm window titled "Sales Reports". The window has a toolbar at the top with buttons for Source, Design, History, and various file operations. The main area contains three report sections:

- Calculate Profit Report:**
Start Month: 2023-1 | End Month: 2023-3 | Value: total_cost | Order by: mymonth | Run
- Most Sold Report:**
Start Month: 2023-1 | End Month: 2023-3 | Value: total_quantity_sold | Order by: mymonth | Run
- Total Orders Report:**
Start Month: 2023-1 | End Month: 2023-3 | Value: total_orders | Order by: mymonth | Run

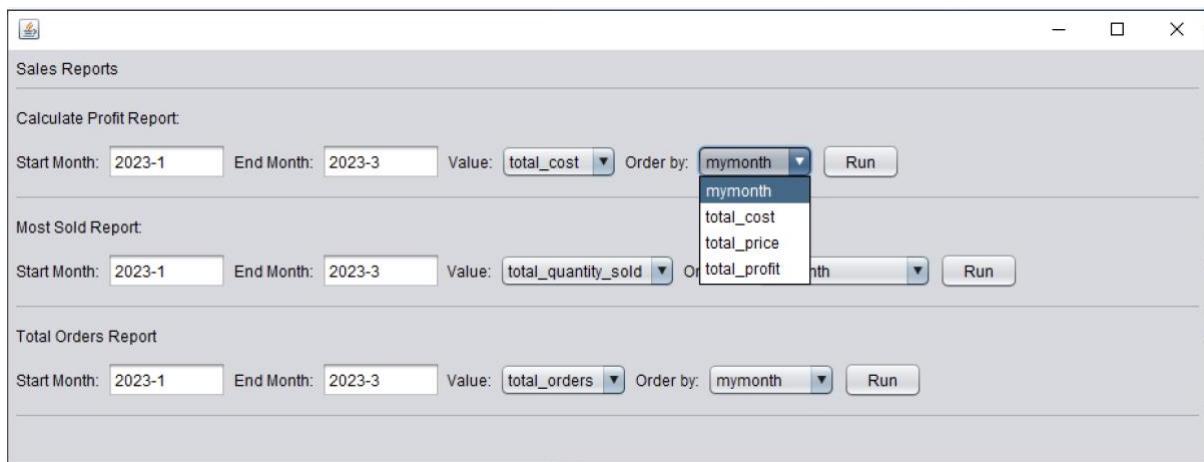
Development and Testing

4.10.5.1 Profit by month report parameter

For the profit report the view I developed has to have total cost and total price so I can calculate the total profit so I can let the user choose what to show, cost or price or profit.



Because I will have cost and price and profit to show from my view then I can also order the report by these as well as by month so that can help the manager see the report ordered by cheapest month as well as the most expensive or the most profitable.



Once the user has typed in the date range in YYYY-MM format for start and end months they can select the Value to show in the report and the Order by which can be any of the columns retrieved in the Profit by month SQL query which is in the view profitbymonth_view.

Every view has its own class so I can call classes.viewname.method for the method to build the report from that view now the user presses Run.

Development and Testing

```
//set some titles for the report components
String frameTitle="ProfitByMonth";
String chartTitle="ProfitByMonthPanelChart";
String tableTitle="ProfitByMonthPanelData";

//new frame to display the report - initialised by the ProductsByCategorySummed_view
//method which assembles the chart and the data table into a report
JFrame frame = classes.ProfitByMonth_view.fillProfitByMonth_viewAssemble(
    jTextFieldCPR_startmonthVal, jTextFieldCPR_endmonthVal, jComboBox1_CPR_ValueToShow,
    jComboBox1_CPRorderValue, 600, 400, 600, 200, frameTitle, chartTitle, tableTitle);

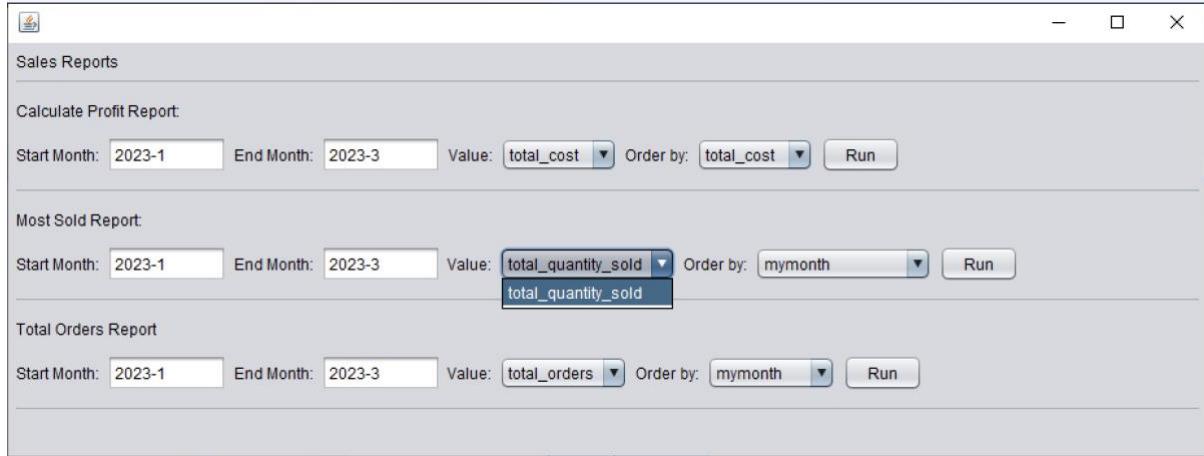
//Display the frame
frame.pack();
frame.setVisible(true);
```

This method will get the user parameters and then it gets the size of the chart and the size of the data table and also the titles for the frame and chart and table. It returns a JFrame to the report screen and then I just make that frame visible and it shows the report.

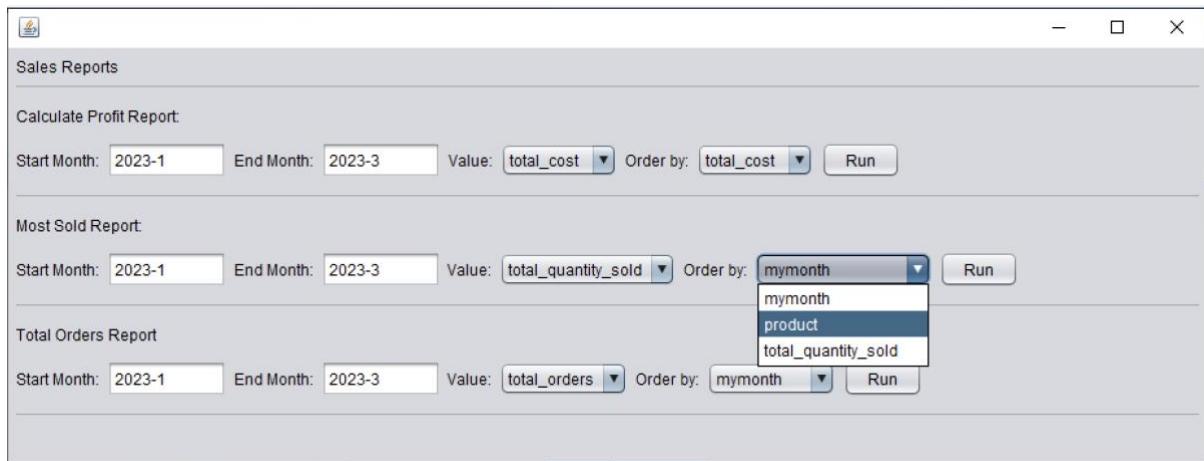
Development and Testing

4.10.5.2 Most sold by month report parameter

For the most sold by month report the view I developed has to have total quantity sold for each product in a month and show the product with the most sold so I let the user choose just the total_quantity_sold data from the view.



Because I will have the product name and the total_quantity_sold to show from my view then I can also order the report by these as well as by month so that can help the manager see the report ordered by product or by the most sold product in the report.



Once the user has typed in the date range in YYYY-MM format for start and end months they can select the Value to show in the report and the Order by which can be any of the columns retrieved in the Most sold product by month SQL query which is in the view `mostsoldbymonth_view`.

Every view has its own class so I can call `classes.viewname.method` for the method to build the report from that view now the user presses Run.

Development and Testing

```
//set some titles for the report components
String frameTitle="MostSoldByMonth";
String chartTitle="MostSoldByMonthPanelChart";
String tableTitle="MostSoldByMonthPanelData";

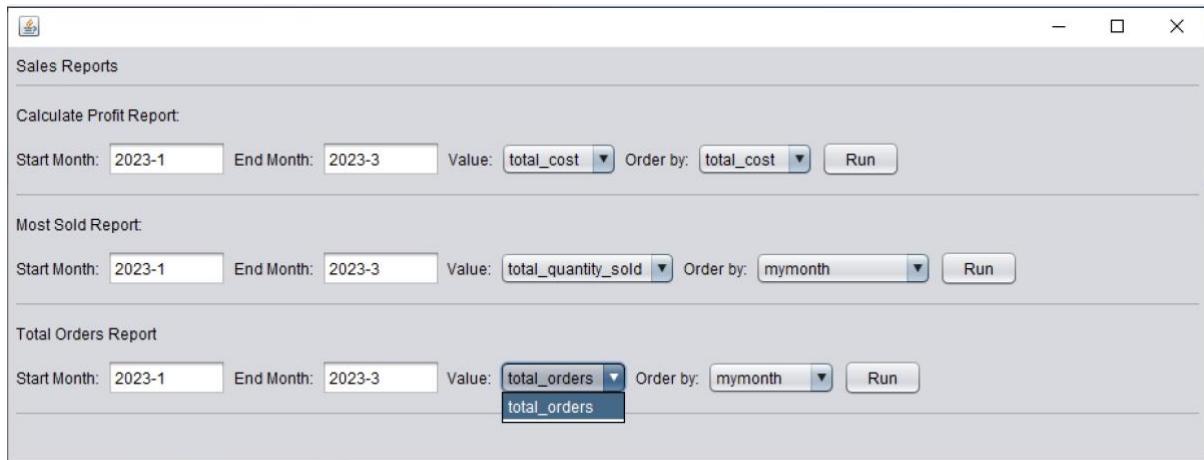
//new frame to display the report - initialised by the ProductsByCategorySummed_view
//method which assembles the chart and the data table into a report
JFrame frame = classes.MostSoldByMonth_view.fillMostSoldByMonth_viewAssemble(
    jTextFieldMSR_startmonthVal, jTextFieldMSR_endmonthVal, jComboBox1_MSR_ValueToShow,
    jComboBox1_MSRRorderValue, 600, 400, 600, 200, frameTitle, chartTitle, tableTitle);

//Display the frame
frame.pack();
frame.setVisible(true);
```

This method will get the user parameters and then it gets the size of the chart and the size of the data table and also the titles for the frame and chart and table. It returns a JFrame to the report screen and then I just make that frame visible and it shows the report.

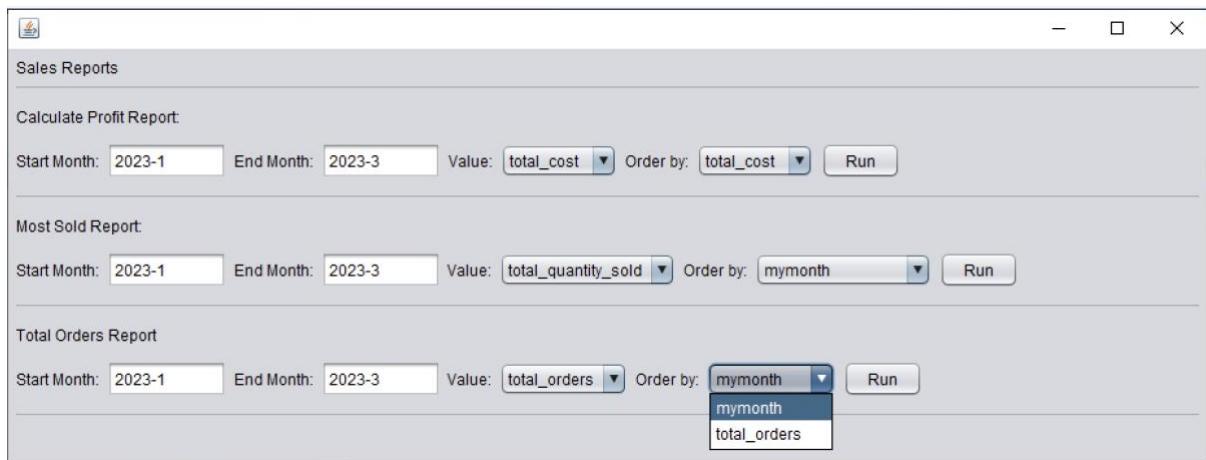
4.10.5.3 Total number of orders by month report parameter

For the Total number of orders by month report the view I developed has to have just the total number of orders in a month so I let the user choose just the total_orders data from the view.



Because I will have the total_orders to show from my view then I can also order the report by this as well as by month so that can help the manager see the report ordered by total_orders so if it had several months in the report it could show the best month first. Or it can just be ordered by month.

Development and Testing



Once the user has typed in the date range in YYYY-MM format for start and end months they can select the Value to show in the report and the Order by which can be any of the columns retrieved in the Total number of orders by month SQL query which is in the view totalordersbymonth_view.

Every view has its own class so I can call classes.viewname.method for the method to build the report from that view now the user presses Run.

```
//set some titles for the report components
String frameTitle="TotalOrdersByMonth";
String chartTitle="TotalOrdersByMonthPanelChart";
String tableTitle="TotalOrdersByMonthPanelData";

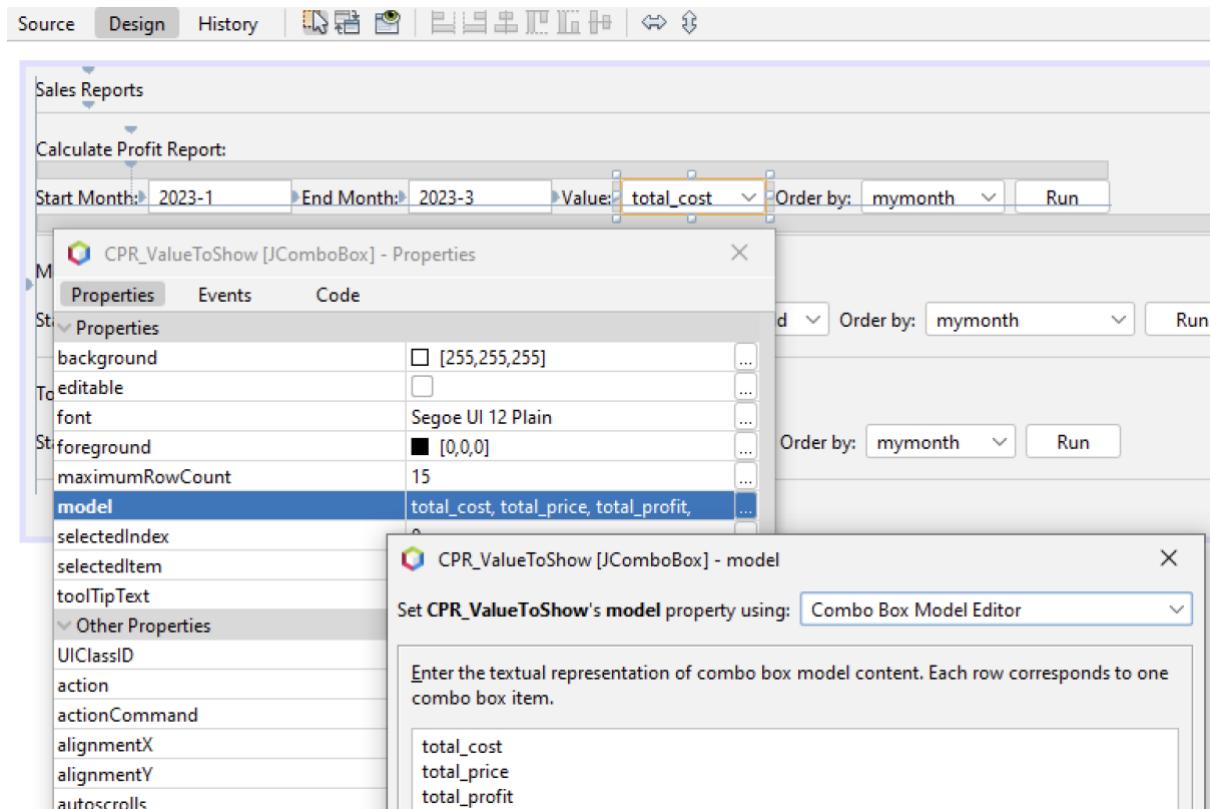
//new frame to display the report - initialised by the ProductsByCategorySummed_view
//method which assembles the chart and the data table into a report
JFrame frame = classes.TotalOrdersByMonth_view.fillTotalOrdersByMonth_viewAssemble(
    jTextFieldTOR_startmonthVal, jTextFieldTOR_endmonthVal, jComboBox1_TOR_ValueToShow,
    jComboBox1_TORordervalue, 600, 400, 600, 200, frameTitle, chartTitle, tableTitle);

//Display the frame
frame.pack();
frame.setVisible(true);
```

This method will get the user parameters and then it gets the size of the chart and the size of the data table and also the titles for the frame and chart and table. It returns a JFrame to the report screen and then I just make that frame visible and it shows the report.

4.10.5.4 Things I could have done better in the reports form

I could have populated the jComboBox for the Value: and the Order by: from the database schema like how I was going to do my form JField validation. Because the values in the combo boxes are set in the Java Swing UI builder statically like this:



And I could have populated them like I did in the newProductScreen like the code below.

```
public void comboBoxAttach() {
    //adds categories to the combo box

    classes.categories category = new classes.categories();
    HashMap<String, Integer> map = category.addToCombo();
    for (String set: map.keySet()){
        categoryComboBox.addItem(set);
    }
}
```

This uses the `addToCombo` method in my `categories` class.

Development and Testing

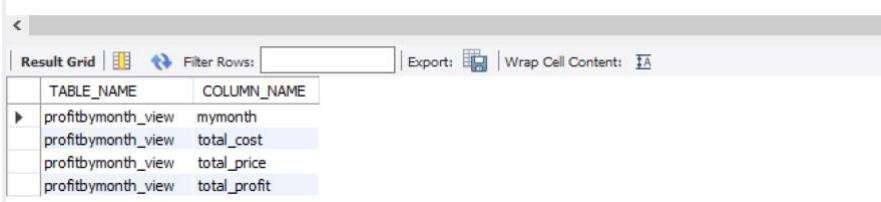
```
public HashMap<String, Integer> addToCombo(){
    //creates a new hashmap that can store strings and integers
    //adds category from the categories table in the database to the product combo box

    HashMap<String, Integer> map = new HashMap<>();
    connection = database.getConnection();
    Statement st;
    ResultSet rs;
    try{
        st = connection.createStatement();
        rs = st.executeQuery("SELECT category_pk, category_name FROM categories");
        categories category;
        while(rs.next()){
            category = new categories(rs.getInt(1), rs.getString(2));
            map.put(category.getName(), category.getCatPK());
        }
    } catch(SQLException ex){
        Logger.getLogger(categories.class.getName()).log(Level.SEVERE, null, ex);
    }
    return map;
}

}
```

Using some SQL code like this below I could get the values for the report combo box choices.

```
27
28 •  SELECT
29   table_name,
30   column_name
31   FROM information_schema.columns
32   WHERE table_schema="posplex"
33   AND
34   table_name="profitbymonth_view"
35   ORDER BY table_name, ordinal_position asc;
```



TABLE_NAME	COLUMN_NAME
profitbymonth_view	mymonth
profitbymonth_view	total_cost
profitbymonth_view	total_price
profitbymonth_view	total_profit

However I need to focus on finishing the project completely and ensuring all the sections are done before touching up or improving areas as I don't have long left to finish the coding.

4.10.6 Developing Profit By Month Report

Every report has its own class like how I did the database tables for the user screens to add orders. So there are 3 report classes named after the report view. The report class is based on the SQL view whereas the table classes are based on an actual database table.

So these classes are pretty similar but as I don't change data in the reports classes I don't need setter methods to set data all I need are the getter methods for each variable which is each field in the database.

The SQL view ProfitByMonth_view has these fields in it.

```

28 •   SELECT
29     table_name,
30     column_name
31   FROM information_schema.columns
32   WHERE table_schema="posplex"
33   AND
34     table_name="profitbymonth_view"
35   ORDER BY table_name, ordinal_position asc;

```

TABLE_NAME	COLUMN_NAME
profitbymonth_view	mymonth
profitbymonth_view	total_cost
profitbymonth_view	total_price
profitbymonth_view	total_profit

So I made a new class named the same as the SQL view called ProfitByMonth_view with the variables and the getter methods for each field in the view.

```

public class ProfitByMonth_view {

    Connection connection;

    //I added these as they are used by addComponentsToPane which is the GridBagLayout layout manager
    final static boolean shouldFill = false;
    final static boolean shouldWeightX = true;
    final static boolean RIGHT_TO_LEFT = false;

    //ProfitByMonth_view attributes
    private String mymonth;
    private float total_cost;
    private float total_price;
    private float total_profit;

    //methods to return the values of each attribute
    public String getmymonth(){
        return mymonth;
    }

    public float gettotal_cost(){
        return total_cost;
    }

    public float gettotal_profit(){
        return total_profit;
    }

    public float gettotal_price(){
        return total_price;
    }
}

```

I defined a new Connection called connection later I will use this with my classes.database.GetConnection() method to either return the active connection to MySQL or if there is no connection it will make a new Connection. A Connection is a session with a database that

I defined in my classes.database class. My SQL statements are executed and results are returned using this connection.

Then I defined some constants that the GridBagLayout manager uses.

```
//I added these as they are used by addComponentsToPane which is the GridBagLayout layout manager
final static boolean shouldFill = false;
final static boolean shouldWeightX = true;
final static boolean RIGHT_TO_LEFT = false;
```

These are used later to set the GridBagConstraints class which is the constraints for the components that are being laid out using the GridBagLayout class. I used the constants in the method addComponentsToPane class that I adapted to layout the report in a grid of 1 column and 2 rows because the chart is in row 1 and the data table JTable is in row 2.

shouldFill is set to false this will tell the layout manager not to try and fill the display area.

shouldWeightX I didn't use in my method but it was to set the weightx constraint otherwise all the components appear in the middle of the JFrame but for my report it only has 1 column and 2 rows so I do want everything in the middle of the panel.

RIGHT_TO_LEFT is used to set if I want the components to layout the other way from normal but I set to false so it runs left to right.

I made an object to hold a row of data from the report object so later I can make an arraylist of all the rows of data in the report.

```
//ProfitByMonth_view row of data object - used to create an array of rows of data - used to hold the res
public ProfitByMonth_view(String mymonth, float total_cost, float total_price, float total_profit)
{
    this.mymonth = mymonth;
    this.total_cost = total_cost;
    this.total_price = total_price;
    this.total_profit = total_profit;
}
```

So the first method called from the report screen is the method to get all the parts of the report together and assemble them into the final report.

It is called fillProfitByMonth_viewAssemble and it is the main method for the public class ProfitByMonth_view.

4.10.6.1 *fillProfitByMonth_viewAssemble part 1 – chart panel*

This method will make the chart and get the report data and assemble them into a JFrame for the report screen to make visible.

```
//assembler for the report - this gets the chart and the data table and lays them out using GridBagLayout layout manager
public static JFrame fillProfitByMonth_viewAssemble(String startDate, String endDate, String metricIn, String orderbyVal,
    int chartWidth, int chartHeight, int tableWidth, int tableHeight, String frameTitle, String chartTitle, String tableTitle)
{
    //make the chart
    ChartPanel chartpanel = classes.ProfitByMonth_view.ProfitByMonthPanel(startDate, endDate, metricIn, orderbyVal);
    //set the size of the chart
    chartpanel.setPreferredSize(new Dimension(chartWidth, chartHeight));
    JPanel jPanelChart = new JPanel();
    jPanelChart.add(chartpanel);
    jPanelChart.setBorder(BorderFactory.createTitledBorder(BorderFactory.createEtchedBorder(), chartTitle, TitledBorder.CENTER, TitledBorder.TOP));

    //make the data table
    JTable jTableData = classes.ProfitByMonth_view.fillProfitByMonth_viewTable(startDate, endDate, orderbyVal);
    //set the size of the table
    jTableData.setPreferredSize(new Dimension(tableWidth, tableHeight));
    JPanel jPanelData = new JPanel();
    jPanelData.add(jTableData);
    jPanelData.setBorder(BorderFactory.createTitledBorder(BorderFactory.createEtchedBorder(), tableTitle, TitledBorder.CENTER, TitledBorder.TOP));

    //new frame to show the GridLayoutDemo for the POS reports
    JFrame frame = new JFrame(frameTitle);

    //Set up the content pane by adding the chart panel and the data panel
    //GridBagLayout manager time!
    classes.ProfitByMonth_view.addComponentstoPane(frame.getContentPane(), jPanelChart, jPanelData);

    //return the assembled report
    return frame;
}
```

This method returns a JFrame to the report screen. The frame will contain the ChartPanel and the JTable for the chart and the report data so all the report screen has to do is set the frame visible.

First I make a new chart panel by calling a method I wrote called ProfitByMonthPanel in my view class that gets the data and creates the chart.

```
//make the chart
ChartPanel chartpanel = classes.ProfitByMonth_view.ProfitByMonthPanel(
    startDate, endDate, metricIn, orderbyVal);
```

4.10.6.2 classes.ProfitByMonth_view.ProfitByMonthPanel

My ProfitByMonthPanel method takes the variables for the start and end YYYY-MM date range from the report screen plus the view column to use when making the chart like mymonth or total_cost or total_price or total_profit, these are from the combo box in my report screen and they are named the same as the columns in the SQL view so that when I reference the columns in the recordset returned from the database it will use the column in the recordset that the user selected.

Also I pass in the ORDER BY selected by the user in the report screen.

```
//create the chart but return it in panel so it can be assembled into the final report
public static ChartPanel ProfitByMonthPanel(
    String startDate, String endDate, String metricIn, String orderbyVal)
{
    Connection connect = database.getConnection();

    PreparedStatement ps;
    ResultSet rs;

    String mymonth="";
    float metric=0;

    //query to retrieve data from the db view where the category is like the report param passed in val
    String query = ProfitByMonth_viewSQL(orderbyVal);

    DefaultCategoryDataset dataset = new DefaultCategoryDataset();

    try{
        ps = connect.prepareStatement(query);

        ps.setString(1, startDate);
        ps.setString(2, endDate);

        System.out.println(ps);

        rs = ps.executeQuery();

        while (rs.next()){

            mymonth = rs.getString("mymonth");
            metric = rs.getFloat(metricIn);
            dataset.setValue(metric, mymonth, mymonth);
        }
    }
    catch (SQLException ex){
        Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
    }

    //make the chart, set the attributes, return the panel to the assembler
    JFreeChart chart = ChartFactory.createBarChart("ProfitByMonth - "+metricIn,"mymonth",metricIn,dataset,
    CategoryPlot p = chart.getCategoryPlot();
    p.setRangeGridlinePaint(Color.black);

    ChartPanel panel = new ChartPanel(chart);

    return panel;
}
```

This method will get the report data from my view and it will put just the columns chosen by the user in my report screen into a dataset to be passed to the JFreeChart ChartFactory class to make a barchart.

```
//create the chart but return it in panel so it can be assembled into the final report
public static ChartPanel ProfitByMonthPanel(
    String startDate, String endDate, String metricIn, String orderbyVal)
{
    Connection connect = database.getConnection();

    PreparedStatement ps;
    ResultSet rs;
```

First I get a connection using my classes.database.GetConnection method.

Then I make a PreparedStatement ps that I will use to make the SQL that retrieves the data from the report view and I make a ResultSet rs that will hold all the rows of report data from the view so I can loop through the ResultSet and add the results to the dataset that I will use to build the barchart.

```
//query to retrieve data from the db view where the category is like the report param passed in val
String query = "SELECT * FROM ProfitByMonth_view WHERE mymonth BETWEEN ? AND ? ORDER BY ?";

DefaultCategoryDataset dataset = new DefaultCategoryDataset();

try{
    ps = connect.prepareStatement(query);

    ps.setString(1, startDate);
    ps.setString(2, endDate);
    ps.setString(3, orderbyVal);

    System.out.println(ps);

    rs = ps.executeQuery();
```

Now I have made a String to hold the SQL query that will get the data from the report view. I use the PreparedStatement.setString method to set the first ? in the SQL string to the startDate variable that was passed in, and the second ? to the endDate and the third ? is set to the orderbyVal that the user chose to order the report results.

Also I made a dataset of type DefaultCategoryDataset this is a datatype belonging to JFree from the data class that is from the JCommon classes that are used along with JFreeCharts classes to make the charts.

```
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartFrame;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;
```

DefaultCategoryDataset implements an interface for a JFree dataset with one or more series, and values associated with categories. I use the DefaultCategoryDataset() constructor this creates a new empty dataset that I will use to hold the data that I choose from the ResultSet to go into the chart.

Now I can prepare the statement and loop through the results and put the report data into the dataset to make my chart.

```

try{
    ps = connect.prepareStatement(query);

    ps.setString(1, startDate);
    ps.setString(2, endDate);
    ps.setString(3, endDate);

    //debug code to see the prepared statement
    System.out.println(ps);

    rs = ps.executeQuery();

    //populate the DefaultCategoryDataset
    while (rs.next()){

        mymonth = rs.getString("mymonth");
        metric = rs.getFloat(metricIn);
        dataset.setValue(metric, mymonth, mymonth);
    }
}

catch (SQLException ex){
    Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
}

```

I wrap my result set code in a try so I can catch any SQL exception that might happen when it tries to access the database.

First I prepare the statement from the variables passed in to my method. Then I execute the query and while my resultset.next is true that means there are results in the record set left to process so then for every result in the set I get the query value for the resultset column called mymonth and the query value for the resultset column named the same as the value the user selected in the report screen.

```

        mymonth = rs.getString("mymonth");
        metric = rs.getFloat(metricIn);
        dataset.setValue(metric, mymonth, mymonth);

```

For each result in the resultset I use the DefaultCategoryDataset.setValue method to add the current mymonth which is DDDD-MM and the metric which might be total profit or total cost or total price to add a new value in the chart dataset. I add the metric which is the value for the dataset row then the rowkey this is the key to what row of the dataset so it is always the YYYY-MM for the dataset value being added and then the columnkey which is the YYYY-MM for this metric in this value.

setValue

```
public void setValue(double value,
                    Comparable rowKey,
                    Comparable columnKey)

Adds or updates a value in the table and sends a DatasetChangeEvent

Parameters:
value - the value.
rowKey - the row key (null not permitted).
columnKey - the column key (null not permitted).
```

This is the format for the dataset to make a barchart so when the results are all added to the dataset then I can make the chart.

```
//make the chart, set the attributes, return the panel to the assembler
JFreeChart chart = ChartFactory.createBarChart("ProfitByMonth - "+metricIn,
                                              "mymonth", metricIn, dataset, PlotOrientation.VERTICAL, false, true, false);
CategoryPlot p = chart.getCategoryPlot();
p.setRangeGridlinePaint(Color.black);

ChartPanel panel = new ChartPanel(chart);

return panel;
```

I make the chart with type JFreeChart and use the constructor ChartFactory.createBarChart to make the barchart with the title "ProfitByMonth - "+metricIn so the title will tell the user what report it is and for what metric that the user chose in the report screen either profit or cost or sell price. Also I pass in the x axis which is the months and the y axis which will be the profit or cost or sell price. Then I pass in the dataset I just built in the resultset loop. I set some other JFreeChart parameters too like plot orientation so the bars are vertical instead of horizontal.

Finally I make a new variable of type CategoryPlot which is the y axis and use the chart constructor GetCategoryPlot to get the plot for the chart I just made so I can set the grid line on the chart to Color.black or red or whatever colour I want.

JFreeChart has a JPanel type called ChartPanel that extends JPanel and is for containing a JFreeChart so I make a new frame "panel" of type ChartPanel and initialize it with the "chart" of type JFreeChart that I made just before when I used the ChartFactory and I return it to my assemble method to be part of the report.

```
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartFrame;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;
```

```

    ChartPanel panel = new ChartPanel(chart);

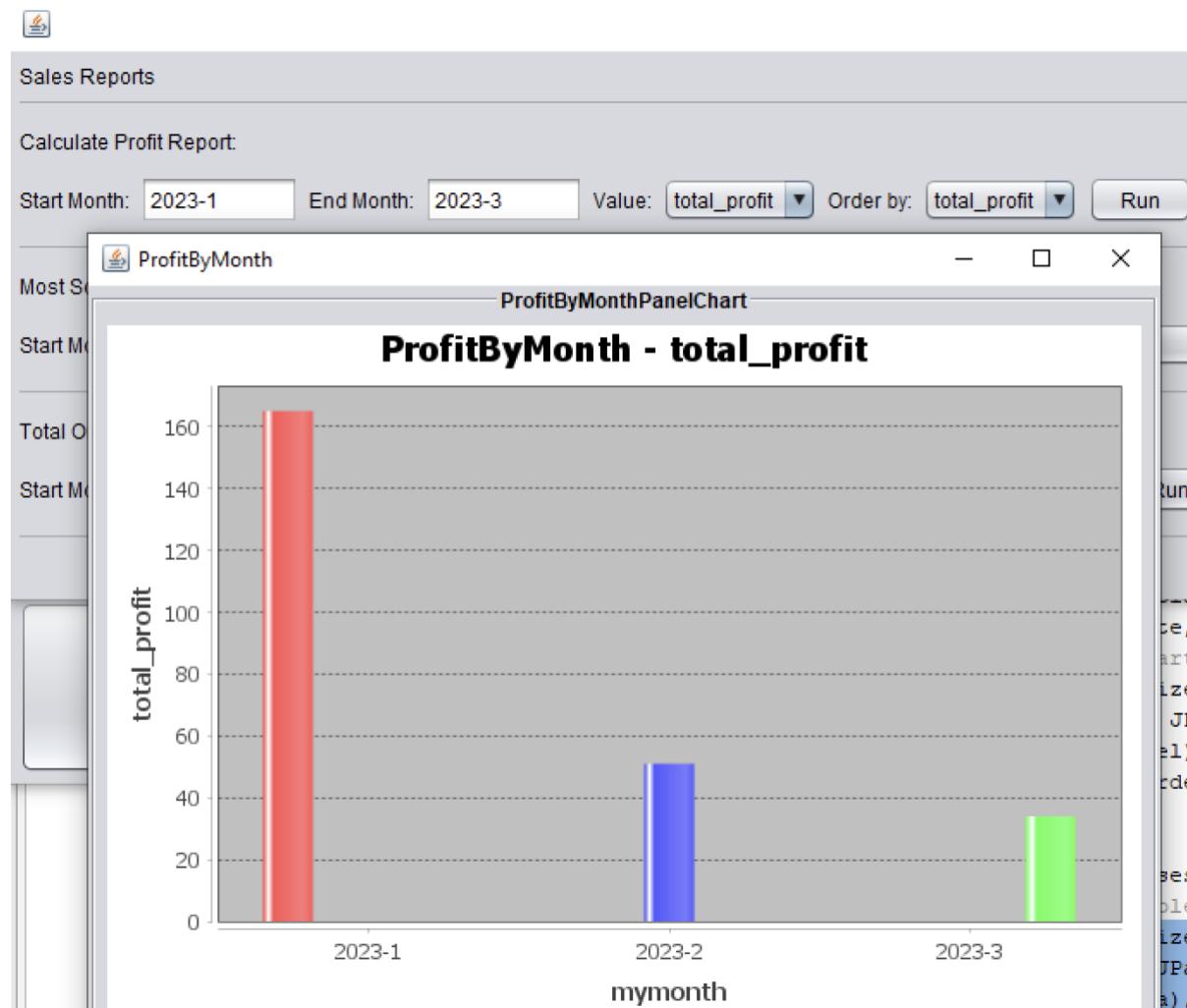
    return panel;

}

```

This returns the chart frame to my assemble method for the first part of the report.

When I run the report there is a problem. The chart works but the order by does not work.



I can see in my debug that the SQL statement in the PreparedStatement won't order the data properly because the ORDER BY clause is ordering by the string 'total_profit'.

```

com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM ProfitByMonth_view WHERE mymonth BETWEEN '2023-1' AND '2023-3' ORDER BY 'total_cost'
sessionId: 665
userPrimaryKey: 10
userType: Admin
665
Admin
access granted
2023-1
2023-3
total_profit
total_profit
com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM ProfitByMonth_view WHERE mymonth BETWEEN '2023-1' AND '2023-3' ORDER BY 'total_profit'

```

I need it to order by total_profit not 'total_profit'.

Development and Testing

I fixed this and wrote it up in my Problem section in this development section. Basically I changed it so I join the SQL string and the orderbyVal string together before I bind the variables and prepare the statement.

```
//query to retrieve data from the db view where the category is like
String query = ProfitByMonth_viewSQL(orderbyVal);

DefaultCategoryDataset dataset = new DefaultCategoryDataset();

try{
    ps = connect.prepareStatement(query);

    ps.setString(1, startDate);
    ps.setString(2, endDate);

    System.out.println(ps);
}
```

Then I made a method to make the SQL string before I prepare the statement.

```
//private method just to make the SQL for my report ProfitByMonth
private static String ProfitByMonth_viewSQL(String orderbyVal)
{
    //query to retrieve data from the db view where the category is like the report param passed
    String query = "SELECT * FROM ProfitByMonth_view WHERE mymonth BETWEEN ? AND ? ORDER BY ";

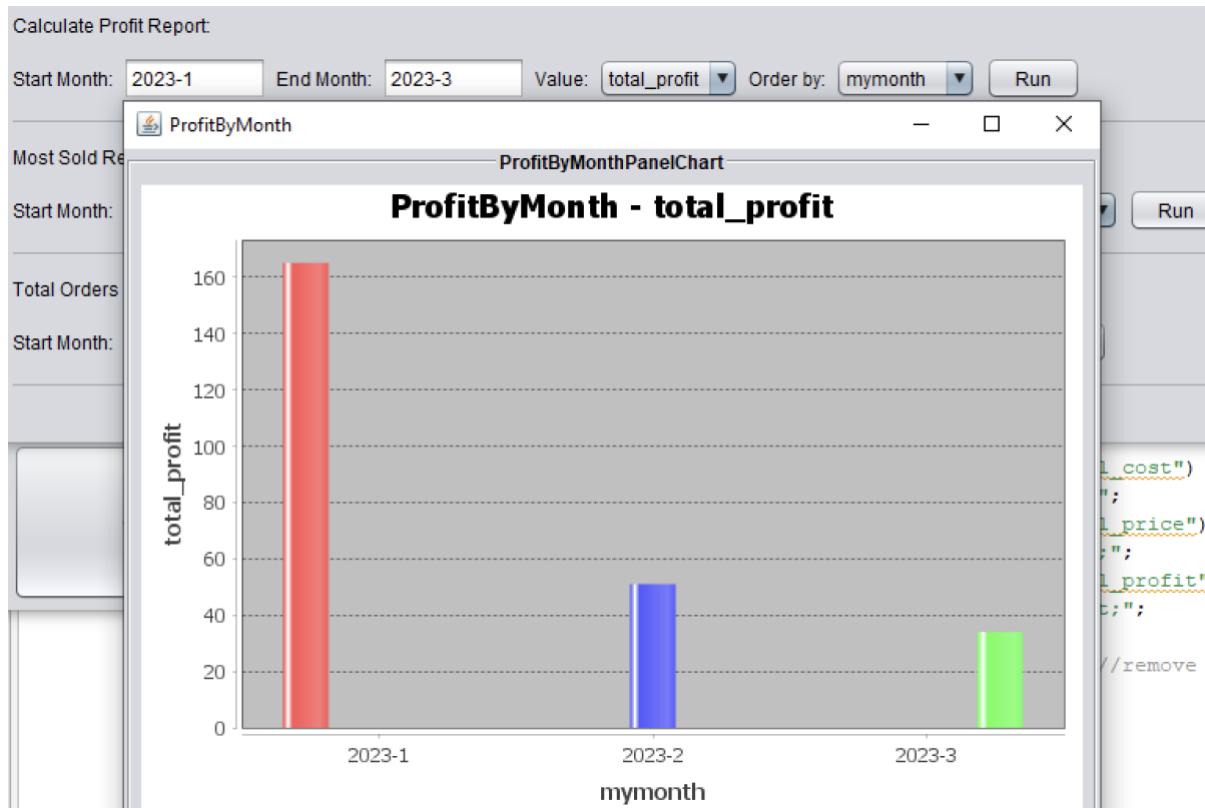
    //append the correct ORDER BY column for the query based on the orderbyVal param
    //note this can in some cases be a risk for SQL injection but PreparedStatement doesn't allow
    if (orderbyVal=="mymonth") {
        query=query+"mymonth;";
    } else if (orderbyVal=="total_cost") {
        query=query+"total_cost;";
    } else if (orderbyVal=="total_price") {
        query=query+"total_price;";
    } else if (orderbyVal=="total_profit") {
        query=query+"total_profit;";
    } else {
        query=query+"mymonth;"; //remove SQL injection risk by examining the orderbyVal and only
    }

    return query;
}
```

Now it works OK and has the column name without the " making the column just a string in the ORDER BY part.

```
2023-1
2023-3
total_profit
mymonth
com.mysql.cj.jdbc.ClientPreparedStatement: SELECT * FROM ProfitByMonth_view WHERE mymonth BETWEEN '2023-1' AND '2023-3' ORDER BY mymonth;
```

Development and Testing



And the chart is ordered by month ascending so it works OK now.

So at this stage I have returned my panel of type ChartPanel back to my assemble method.

```
//assembler for the report - this gets the chart and the data table and lays them out using GridBagLayout layout
public static JFrame fillProfitByMonth_viewAssemble(String startDate,
    String endDate, String metricIn, String orderbyVal,
    int chartWidth, int chartHeight, int tableWidth, int tableHeight,
    String frameTitle, String chartTitle, String tableTitle)
{
    //make the chart
    ChartPanel chartpanel = classes.ProfitByMonth_view.ProfitByMonthPanel(
        startDate, endDate, metricIn, orderbyVal);
    //set the size of the chart
    chartpanel.setPreferredSize(new Dimension(chartWidth, chartHeight));
    JPanel jPanelChart = new JPanel();
    jPanelChart.add(chartpanel);
    jPanelChart.setBorder(BorderFactory.createTitledBorder(
        BorderFactory.createEtchedBorder(), chartTitle, TitledBorder.CENTER,
        TitledBorder.TOP));

    //make the data table
    JTable jTableData = classes.ProfitByMonth_view.fillProfitByMonth_viewTable(startDate, endDate, orderbyVal);
    //set the size of the table
    jTableData.setPreferredSize(new Dimension(tableWidth, tableHeight));
    JPanel jPanelData = new JPanel();
    jPanelData.add(jTableData);
    jPanelData.setBorder(BorderFactory.createTitledBorder( BorderFactory.createEtchedBorder(),
        tableTitle, TitledBorder.CENTER, TitledBorder.TOP));

    //new frame to show the GridBagLayoutDemo for the POS reports
    JFrame frame = new JFrame(frameTitle);

    //Set up the content pane by adding the chart panel and the data panel
    //GridBagLayout manager time!
    classes.ProfitByMonth_view.addComponentstoPane(frame.getContentPane(), jPanelChart, jPanelData);

    //return the assembled report
    return frame;
}
```

4.10.6.3 fillProfitByMonth_viewAssemble part 2 – data table panel

Back in the fillProfitByMonth_viewAssemble method I have returned the chart panel to this method.

```
//make the chart
ChartPanel chartpanel = classes.ProfitByMonth_view.ProfitByMonthPanel(
    startDate, endDate, metricIn, orderbyVal);
//set the size of the chart
chartpanel.setPreferredSize(new Dimension(chartWidth, chartHeight));
JPanel jPanelChart = new JPanel();
jPanelChart.add(chartpanel);
jPanelChart.setBorder(BorderFactory.createTitledBorder(
    BorderFactory.createEtchedBorder(), chartTitle, TitledBorder.CENTER,
    TitledBorder.TOP));
```

After I have got the chartpanel then I set the size of the panel to 600 x 400 to make it fit the screen.

4.10.6.3.1 Something I could have done better

Now I have set the size of the chart panel probably I should figure out how to let Swing lay out the chart but I read up that JFreeChart just supplies a standard size 680 x 420 and I tried to use getPreferredSize() as I could set the size dynamically. If I could get this working I could set the size of the chart panel based on the size of the container available like if it was a laptop or a table or a phone.

So all I did here is set the preferred size using the parameters I passed in from the report screen as I could change the report screen so the user could actually select what size they wanted.

```
//set the size of the chart
chartpanel.setPreferredSize(new Dimension(chartWidth, chartHeight));
```

Finally I make a new JPanel and I add my chartpanel to the JPanel and I set the type of border to be a titled border with an etched border appearance and I pass in the title of the chart and centre the title and put the title at the top.

```
JPanel jPanelChart = new JPanel();
jPanelChart.add(chartpanel);
jPanelChart.setBorder(BorderFactory.createTitledBorder(
    BorderFactory.createEtchedBorder(), chartTitle, TitledBorder.CENTER,
    TitledBorder.TOP));
```

That is the end of the chart panel part of the assemble method.

Now I need to make the data table to have the report data in a JTable so I can add this to the report layout.

```
//make the data table
JTable jTableData = classes.ProfitByMonth_view.fillProfitByMonth_viewTable(startDate, endDate, orderbyVal);
//set the size of the table
jTableData.setPreferredSize(new Dimension(tableWidth, tableHeight));
JPanel jPanelData = new JPanel();
jPanelData.add(jTableData);
jPanelData.setBorder(BorderFactory.createTitledBorder( BorderFactory.createEtchedBorder(), tableTitle, TitledBorder.CENTER, TitledBorder.TOP));
```

I create a new JTable called jTableData and I make a method fillProfitByMonth_viewTable to fill that JTable.

```
//make the data table
JTable jTableData = classes.ProfitByMonth_view.fillProfitByMonth_viewTable(startDate, endDate, orderbyVal);
```

4.10.6.4 classes.ProfitByMonth_view.fillProfitByMonth_viewTable

My fillProfitByMonth_viewTable is a lot like the fill table methods I do for my NetBeans UI forms when I have to fill the JTable on a user screen from the database. I reused a lot of that code for these read only report view classes.

```
//make the data table
JTable jTableData = classes.ProfitByMonth_view.fillProfitByMonth_viewTable(startDate, endDate, orderbyVal);
```

This method takes the variables for the start and end YYYY-MM date range from the report screen but I don't take the metric variable like I did for the chart method because the data table shows every column in the report view ProfitByMonth_view which is the month YYYY-MM and the total_cost, total_sellprice, total_profit. I only needed the metric variable for the chart as that shows just the metric by the months.

This method just fills the JTable with data from the array list that I build from a MySQL query on the report view.

```
//fills and returns the jTable with contents of arrayList so it can be added to the data panel
public static JTable fillProfitByMonth_viewTable(String startDate, String endDate, String orderbyVal){

    classes.ProfitByMonth_view ProfitByMonth_view = new classes.ProfitByMonth_view();
    ArrayList <classes.ProfitByMonth_view> ProfitByMonth_viewList =
        ProfitByMonth_view.ProfitByMonth_viewList(startDate, endDate, orderbyVal);

    String [] columnNames = {"mymonth","total_cost","total_price","total_profit"};
    Object [][] rows = new Object[ProfitByMonth_viewList.size()][4];

    for(int i = 0; i<ProfitByMonth_viewList.size(); i++){
        rows[i][0] = ProfitByMonth_viewList.get(i).getmymonth();
        rows[i][1] = ProfitByMonth_viewList.get(i).gettotal_cost();
        rows[i][2] = ProfitByMonth_viewList.get(i).gettotal_price();
        rows[i][3] = ProfitByMonth_viewList.get(i).gettotal_profit();
    }

    classes.table model = new classes.table(rows, columnNames);
    JTable ProfitByMonth_viewTable = new JTable();
    ProfitByMonth_viewTable.setModel(model);

    //sets the jTable column names
    //bug not working right now, table has no column headers when displayed
    ProfitByMonth_viewTable.getColumnModel().getColumn(0).setHeaderValue("mymonth");
    ProfitByMonth_viewTable.getColumnModel().getColumn(1).setHeaderValue("total_cost");
    ProfitByMonth_viewTable.getColumnModel().getColumn(2).setHeaderValue("total_price");
    ProfitByMonth_viewTable.getColumnModel().getColumn(3).setHeaderValue("total_profit");

    //return the table of data so it can be laid out in the assembler
    return ProfitByMonth_viewTable;
}
```

Firstly I instantiated a new ProfitByMonth_view class so I can use the classes and methods in it to build the JTable for the report data table. I use the blank constructor to make the new class in memory. Then I created an ArrayList containing the elements of the collection defined in the ProfitByMonth_view class which are defined like:

```
//ProfitByMonth_view attributes
private String mymonth;
private float total_cost;
private float total_price;
private float total_profit;
```

So now I have a resizable array called ProfitByMonth_viewList that can hold the columns of data that I will get from the profitbymonth_view SQL view so I initialize this array list to contain the data in the

array list that is returned from my ProfitByMonth_viewList (String startDate, String endDate, String orderbyVal) method. This ProfitByMonth_viewList is where I prepare the statement and get the data from MySQL. I will explain it later. I have to pass into this method the parameters from the report screen like the start and end YYYY-MM months and the ORDER BY column name.

```
public static JTable fillProfitByMonth_viewTable(String startDate, String endDate, String
    classes.ProfitByMonth_view ProfitByMonth_view = new classes.ProfitByMonth_view();
    ArrayList <classes.ProfitByMonth_view> ProfitByMonth_viewList =
        ProfitByMonth_view.ProfitByMonth_viewList(startDate, endDate, orderbyVal);
```

Now I make an array called columnNames to hold an array of strings which are the column names and initialized it with my column names from the report view so the JTable table model headings will use these later. Then I make a 2-dimensional array called rows using the Object superclass and initialized it to be the same number of rows in my array list ProfitByMonth_viewList that I filled with data above, and to be 4 columns.

Then for every row [i] in the array I use the getter methods in my ProfitByMonth_viewList class to set each column in the current row with the values of the columns in the corresponding row get(i) in the array list that I made earlier and filled with the MySQL data.

```
String [] columnNames = {"mymonth","total_cost","total_price","total_profit"};
Object [][] rows = new Object[ProfitByMonth_viewList.size()][4];

for(int i = 0; i<ProfitByMonth_viewList.size(); i++){
    rows[i][0] = ProfitByMonth_viewList.get(i).getmymonth();
    rows[i][1] = ProfitByMonth_viewList.get(i).gettotal_cost();
    rows[i][2] = ProfitByMonth_viewList.get(i).gettotal_price();
    rows[i][3] = ProfitByMonth_viewList.get(i).gettotal_profit();
}

classes.table model = new classes.table(rows, columnNames);
JTable ProfitByMonth_viewTable = new JTable();
ProfitByMonth_viewTable.setModel(model);
```

Finally I used the 2-dimensional array of data (rows) and the 1-dimensional array of column names in columnNames to initialize a new instance called “model” of my tables class where the methods for the TableModel interface are already defined because I used them for the NetBeans UI builder and the JTables in the forms. This gives me the “model” with my data and column names using the AbstractTableModel classes that are in Swing so now I can make a new JTable and then use the JTable.setModel(TableModel dataModel) method with my “model” as the datasource for the JTable.

Development and Testing

```
//sets the jTable column names
//bug not working right now, table has no column headers when displayed
ProfitByMonth_viewTable.getColumnModel().getColumn(0).setHeaderValue("mymonth");
ProfitByMonth_viewTable.getColumnModel().getColumn(1).setHeaderValue("total_cost");
ProfitByMonth_viewTable.getColumnModel().getColumn(2).setHeaderValue("total_price");
ProfitByMonth_viewTable.getColumnModel().getColumn(3).setHeaderValue("total_profit");

//return the table of data so it can be laid out in the assembler
return ProfitByMonth_viewTable;

}
```

Finally I set the heading values in my JTable of data to the column_names from the view. And then I return the JTable filled with the data to the fillProfitByMonth_viewAssemble method ready to be added to a JFrame to be returned to the report screen so that it can be made visible.

```
//assembler for the report - this gets the chart and the data table and lays them out using GridBagLayout layout
public static JFrame fillProfitByMonth_viewAssemble(String startDate,
    String endDate, String metricIn, String orderbyVal,
    int chartWidth, int chartHeight, int tableWidth, int tableHeight,
    String frameTitle, String chartTitle, String tableTitle)
{
    //make the chart
    ChartPanel chartpanel = classes.ProfitByMonth_view.ProfitByMonthPanel(
        startDate, endDate, metricIn, orderbyVal);
    //set the size of the chart
    chartpanel.setPreferredSize(new Dimension(chartWidth, chartHeight));
    JPanel jPanelChart = new JPanel();
    jPanelChart.add(chartpanel);
    jPanelChart.setBorder(BorderFactory.createTitledBorder(
        BorderFactory.createEtchedBorder(), chartTitle, TitledBorder.CENTER,
        TitledBorder.TOP));

    //make the data table
    JTable jTableData = classes.ProfitByMonth_view.fillProfitByMonth_viewTable(startDate, endDate, orderbyVal);
    //set the size of the table
    jTableData.setPreferredSize(new Dimension(tableWidth, tableHeight));
    JPanel jPanelData = new JPanel();
    jPanelData.add(jTableData);
    jPanelData.setBorder(BorderFactory.createTitledBorder( BorderFactory.createEtchedBorder(),
        tableTitle, TitledBorder.CENTER, TitledBorder.TOP));

    //new frame to show the GridBagLayoutDemo for the POS reports
    JFrame frame = new JFrame(frameTitle);

    //Set up the content pane by adding the chart panel and the data panel
    //GridBagLayout manager time!
    classes.ProfitByMonth_view.addComponentToPane(frame.getContentPane(), jPanelChart, jPanelData);

    //return the assembled report
    return frame;
}
```

4.10.6.5 classes.ProfitByMonth_view.ProfitByMonth_viewList

In the previous section for the classes.ProfitByMonth_view.fillProfitByMonth_viewTable method I am just filling my JTable with a data model that contains the results of the MySQL query for the report from the view. I get the data by making a method that returns an array list of data created from a prepared SQL statement that I made. This is the method where I get my report data from the view using a database connection.

```
//this ArrayList class is a resizable array, which is present in the java. util package from the docs
// built-in arrays have a fixed size, ArrayLists can change their size dynamically so
//I used it to retrieve the data from the database view so it can be shown as the data table in the report
public ArrayList<ProfitByMonth_view> ProfitByMonth_viewList
    (String startDate, String endDate, String orderbyVal){

    //create arraylist of ProductsByCategorySummed_view rows
    ArrayList<ProfitByMonth_view> ProfitByMonth_viewList = new ArrayList<>();

    //get a db connection and make the resultset to hold the db data
    connection = database.getConnection();
    ResultSet rs;

    //query to retrieve data from the db view where the category is like the report param passed in val
    String query = ProfitByMonth_viewSQL(orderbyVal);

    try{
        connection = database.getConnection();

        //PreparedStatement ps from the query with the categoryID like %val%
        PreparedStatement ps = connection.prepareStatement(query);
        ps.setString(1, startDate);
        ps.setString(2, endDate);

        rs = ps.executeQuery();

        //create the view data row and loop through the data in the SQL and add each row to the arraylist
        ProfitByMonth_view ProfitByMonth_viewRow;
        while (rs.next()){
            ProfitByMonth_viewRow = new ProfitByMonth_view(rs.getString("mymonth"),
                rs.getFloat("total_cost"), rs.getFloat("total_price"), rs.getFloat("total_profit") );

            ProfitByMonth_viewList.add(ProfitByMonth_viewRow);
        }
    }
    catch (SQLException ex){
        Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
    }

    //return the dataset in the arraylist
    return ProfitByMonth_viewList;
}
```

I pass into the method the start and end dates and the column name to order the SQL query from the report view.

Development and Testing

Firstly I made an array list with the attributes of my ProfitByMonth_view class so this has the columns like this and initialize it as an empty array list because it has no data yet.

I make a connection that I will use to run my database query on the report view and a result set to hold the results of my query.

```
//ProfitByMonth_view attributes
private String mymonth;
private float total_cost;
private float total_price;
private float total_profit;

//create arraylist of ProductsByCategorySummed_view rows
ArrayList<ProfitByMonth_view> ProfitByMonth_viewList = new ArrayList<>();

//get a db connection and make the resultset to hold the db data
connection = database.getConnection();
ResultSet rs;

//query to retrieve data from the db view where the category is like the report param passed
String query = ProfitByMonth_viewSQL(orderbyVal);
```

I made the query string reusing the same method I used to make the query for the chart pane before in my method ProfitByMonthPanel. I documented this before in the ProfitByMonthPanel section.

```
//private method just to make the SQL for my report ProfitByMonth
private static String ProfitByMonth_viewSQL(String orderbyVal)
{
    //query to retrieve data from the db view where the category is like the report param passed
    String query = "SELECT * FROM ProfitByMonth_view WHERE mymonth BETWEEN ? AND ? ORDER BY ";

    //append the correct ORDER BY column for the query based on the orderbyVal param
    //note this can in some cases be a risk for SQL injection but PreparedStatement doesn't allow this
    if (orderbyVal=="mymonth") {
        query=query+"mymonth;";
    } else if (orderbyVal=="total_cost") {
        query=query+"total_cost;";
    } else if (orderbyVal=="total_price") {
        query=query+"total_price;";
    } else if (orderbyVal=="total_profit") {
        query=query+"total_profit;";
    } else {
        query=query+"mymonth;"; //remove SQL injection risk by examining the orderbyVal and on
    }

    return query;
}
```

Now I can prepare the statement with the SQL string and get the results and then I loop through the results set and for each row I use a row of data of type ProfitByMonth_view so it has the attributes like this below and I set the viewRow attributes using the ProfitByMonth_view method.

```
//ProfitByMonth_view attributes
private String mymonth;
private float total_cost;
private float total_price;
private float total_profit;
```

Development and Testing

```
try{
    connection = database.getConnection();

    //PreparedStatement ps from the query with the categoryID like %val%
    PreparedStatement ps = connection.prepareStatement(query);
    ps.setString(1, startDate);
    ps.setString(2, endDate);

    rs = ps.executeQuery();

    //create the view data row and loop through the data in the SQL and add each row to the arraylist
    ProfitByMonth_view ProfitByMonth_viewRow;
    while (rs.next()){
        ProfitByMonth_viewRow = new ProfitByMonth_view(rs.getString("mymonth"),
            rs.getFloat("total_cost"), rs.getFloat("total_price"), rs.getFloat("total_profit") );

        ProfitByMonth_viewList.add(ProfitByMonth_viewRow);
    }
}
catch (SQLException ex){
    Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
}

//return the dataset in the arraylist
return ProfitByMonth_viewList;
}
```

Firstly I put it all in a try statement with an exception to catch if any database operation goes wrong.

Then I make the connection and use the prepareStatement method to make a prepared statement from my SQL query and I bind the variables for the start and end dates. And then I can make the resultset using the executeQuery() method of the prepared statement.

After this I made an object of type ProfitByMonth_view with the attributes for each column returned by the query into the result set. Then I loop and while resultset.true() returns true meaning there is data in the result set to process then I can add that row in the result set to the array list because the array list is not like a normal array which is created with a set size instead I can change the size of my array list dynamically using the .add(row) method for an array list.

```
/create the view data row and loop through the data in the SQL and add each row to the arraylist
ProfitByMonth_view ProfitByMonth_viewRow;
while (rs.next()){
    ProfitByMonth_viewRow = new ProfitByMonth_view(rs.getString("mymonth"),
        rs.getFloat("total_cost"), rs.getFloat("total_price"), rs.getFloat("total_profit") );

    ProfitByMonth_viewList.add(ProfitByMonth_viewRow);
}
```

I use my ProfitByMonth_view method as below to set the attributes in ProfitByMonth_viewrow object so for each row in the result set I call this method below and initialize the ProfitByMonth_viewrow then I have a row with the data.

```
//ProfitByMonth_view row of data object - used to create an array of rows of data - used to hold the
public ProfitByMonth_view(String mymonth, float total_cost, float total_price, float total_profit)
{
    this.mymonth = mymonth;
    this.total_cost = total_cost;
    this.total_price = total_price;
    this.total_profit = total_profit;
}
```

Finally I add the row in ProfitByMonth_viewrow to the array list in ProfitByMonth_viewList.

```

        ProfitByMonth_viewList.add(ProfitByMonth_viewRow);
    }
}

catch (SQLException ex){
    Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
}

//return the dataset in the arraylist
return ProfitByMonth_viewList;
}

```

And I return the ProfitByMonth_viewList which is an array list of the report view data from my SQL to my method fillProfitByMonth_viewTable which makes the JTable from the array list and returns the JTable to the final part of my assemble method fillProfitByMonth_viewAssemble where I lay out the JPanel “jPanelChart” and my JTable “jTableData” using GridBag to make a frame I can return to my report screen to make visible to the user.

```

//assembler for the report - this gets the chart and the data table and lays them out using GridBagLayout layout
public static JFrame fillProfitByMonth_viewAssemble(String startDate,
    String endDate, String metricIn, String orderbyVal,
    int chartWidth, int chartHeight, int tableWidth, int tableHeight,
    String frameTitle, String chartTitle, String tableTitle)
{
    //make the chart
    ChartPanel chartpanel = classes.ProfitByMonth_view.ProfitByMonthPanel(
        startDate, endDate, metricIn, orderbyVal);
    //set the size of the chart
    chartpanel.setPreferredSize(new Dimension(chartWidth, chartHeight));
    JPanel jPanelChart = new JPanel();
    jPanelChart.add(chartpanel);
    jPanelChart.setBorder(BorderFactory.createTitledBorder(
        BorderFactory.createEtchedBorder(), chartTitle, TitledBorder.CENTER,
        TitledBorder.TOP));

    //make the data table
    JTable jTableData = classes.ProfitByMonth_view.fillProfitByMonth_viewTable(startDate, endDate, orderbyVal);
    //set the size of the table
    jTableData.setPreferredSize(new Dimension(tableWidth, tableHeight));
    JPanel jPanelData = new JPanel();
    jPanelData.add(jTableData);
    jPanelData.setBorder(BorderFactory.createTitledBorder( BorderFactory.createEtchedBorder(),
        tableTitle, TitledBorder.CENTER, TitledBorder.TOP));

    //new frame to show the GridBagLayoutDemo for the POS reports
    JFrame frame = new JFrame(frameTitle);

    //Set up the content pane by adding the chart panel and the data panel
    //GridBagLayout manager time!
    classes.ProfitByMonth_view.addComponentstoPane(frame.getContentPane(), jPanelChart, jPanelData);

    //return the assembled report
    return frame;
}

```

4.10.6.6 *fillProfitByMonth_viewAssemble part 3 – layout manager*

Finally in the fillProfitByMonth_viewAssemble method I have returned the chart panel and the data table as a JPanel and a JTable to this method ready to be assembled into the final report which is a JFrame returned to the report screen.

```
//assembler for the report - this gets the chart and the data table and lays them out using GridBagLayout layout
public static JFrame fillProfitByMonth_viewAssemble(String startDate,
    String endDate, String metricIn, String orderbyVal,
    int chartWidth, int chartHeight, int tableWidth, int tableHeight,
    String frameTitle, String chartTitle, String tableTitle)
{
    //make the chart
    ChartPanel chartpanel = classes.ProfitByMonth_view.ProfitByMonthPanel(
        startDate, endDate, metricIn, orderbyVal);
    //set the size of the chart
    chartpanel.setPreferredSize(new Dimension(chartWidth, chartHeight));
    JPanel jPanelChart = new JPanel();
    jPanelChart.add(chartpanel);
    jPanelChart.setBorder(BorderFactory.createTitledBorder(
        BorderFactory.createEtchedBorder(), chartTitle, TitledBorder.CENTER,
        TitledBorder.TOP));

    //make the data table
    JTable jTableData = classes.ProfitByMonth_view.fillProfitByMonth_viewTable(startDate, endDate, orderbyVal);
    //set the size of the table
    jTableData.setPreferredSize(new Dimension(tableWidth, tableHeight));
    JPanel jPanelData = new JPanel();
    jPanelData.add(jTableData);
    jPanelData.setBorder(BorderFactory.createTitledBorder( BorderFactory.createEtchedBorder(),
        tableTitle, TitledBorder.CENTER, TitledBorder.TOP));

    //new frame to show the GridBagLayoutDemo for the POS reports
    JFrame frame = new JFrame(frameTitle);

    //Set up the content pane by adding the chart panel and the data panel
    //GridBagLayout manager time!
    classes.ProfitByMonth_view.addComponentstoPane(frame.getContentPane(), jPanelChart, jPanelData);

    //return the assembled report
    return frame;
}
```

Like the chartpanel I then set the size of the JTable to 600 x 200 to make it fit the screen.

4.10.6.1 Something I could have done better

Now I have set the size of the JTable probably I should figure out how to let Swing lay out the table dynamically. If I could get this working I could set the size of the table based on the size of the container available like if it was a laptop or a table or a phone.

Also I might try and put it in a JScrollPane so the user could scroll the report data if there was too much to show on a screen.

So all I did here is set the preferred size using the parameters I passed in from the report screen as I could change the report screen so the user could actually select what size they wanted.

```
//set the size of the table
jTableData.setPreferredSize(new Dimension(tableWidth, tableHeight));
JPanel jPanelData = new JPanel();
jPanelData.add(jTableData);
```

Finally I make a new JPanel “jPanelData” and I add my “jTableData” to the JPanel and I set the type of border to be a titled border with an etched border appearance and I pass in the title of the data and centre the title and put the title at the top.

Development and Testing

```
JPanel jPanelData = new JPanel();
jPanelData.add(jTableData);
jPanelData.setBorder(BorderFactory.createTitledBorder( BorderFactory.createEtchedBorder(),
    tableTitle, TitledBorder.CENTER, TitledBorder.TOP));
```

That is the end of the data panel part of the assemble method. Now I have a JPanel containing my chart panel with the barchart and I have a JTable containing my array list of report data.

Now I need to add the chart and the data to the JFrame to return to the report screen.

First I tried to just add the panels to a frame like this.

```
//assembler for the report - this gets the chart and the data table and lays them out using GridBagLayout
public static JFrame fillProfitByMonth_viewAssemble(String startDate,
    String endDate, String metricIn, String orderbyVal,
    int chartWidth, int chartHeight, int tableWidth, int tableHeight,
    String frameTitle, String chartTitle, String tableTitle)
{
    //make the chart
    ChartPanel chartpanel = classes.ProfitByMonth_view.ProfitByMonthPanel(
        startDate, endDate, metricIn, orderbyVal);
    //set the size of the chart
    chartpanel.setPreferredSize(new Dimension(chartWidth, chartHeight));
    JPanel jPanelChart = new JPanel();
    jPanelChart.add(chartpanel);
    jPanelChart.setBorder(BorderFactory.createTitledBorder(
        BorderFactory.createEtchedBorder(), chartTitle, TitledBorder.CENTER,
        TitledBorder.TOP));

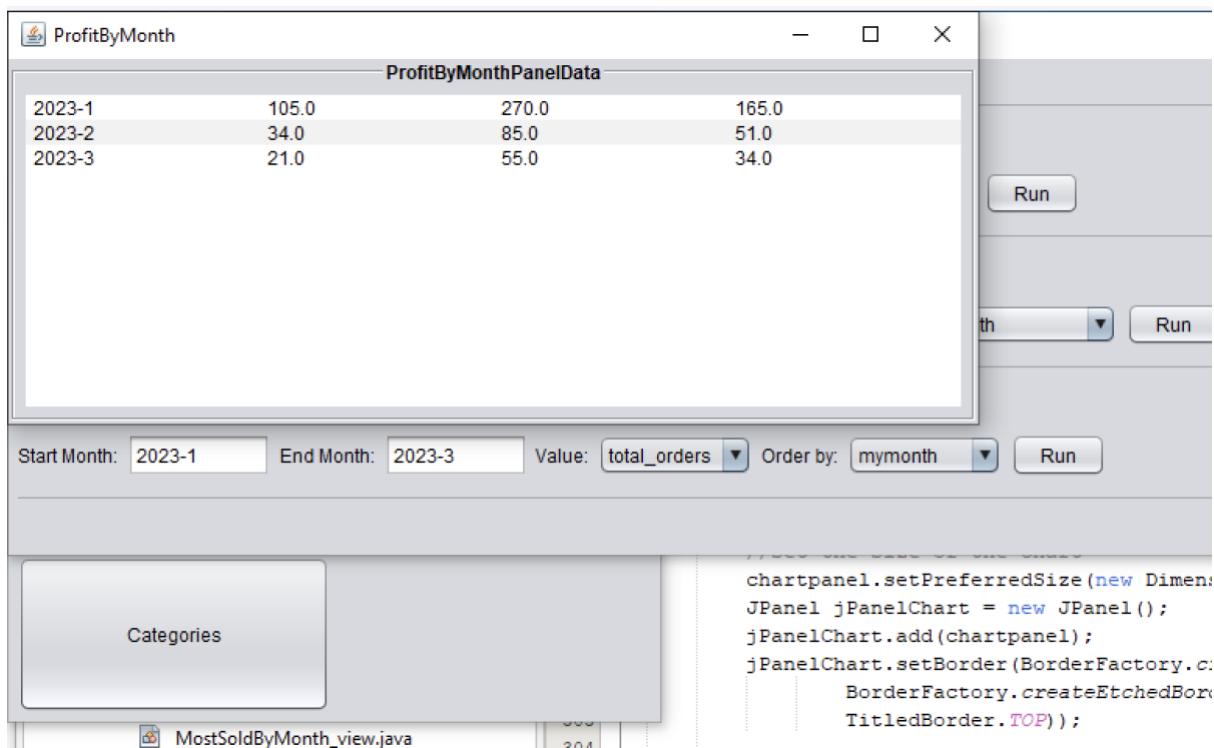
    //make the data table
    JTable jTableData = classes.ProfitByMonth_view.fillProfitByMonth_viewTable(startDate, endDate);
    //set the size of the table
    jTableData.setPreferredSize(new Dimension(tableWidth, tableHeight));
    JPanel jPanelData = new JPanel();
    jPanelData.add(jTableData);
    jPanelData.setBorder(BorderFactory.createTitledBorder( BorderFactory.createEtchedBorder(),
        tableTitle, TitledBorder.CENTER, TitledBorder.TOP));

    //add the chart and data to the frame to return
    JFrame frame = new JFrame(frameTitle);
    frame.add(jPanelChart);
    frame.add(jPanelData);

    //return the assembled report
    return frame;
}
```

But it doesn't work well. It looked like this. I think when I add the jPanelChart to the JFrame it works but then when I pass the jPanelData it just replaces the JPanel Data.

Development and Testing



So I researched how to do layouts if you are not using a UI builder tool like NetBeans which I am not because I am creating these reports dynamically.

I read up on the Oracle Swing docs on the layout managers that solve this problem of controlling where my components appear on a JFrame if I am making a JFrame dynamically and I want to make items like JFields and JPanels appear where I want them on the screen. I chose the GridBagLayout manager which is provided by Swing and is what they recommend for laying out items if I am not using the NetBeans UI designer tool.

I wrote up this in the Problems section and I changed the code to use the GridBagLayout example but I changed it to a simple frame with 1 column and 2 rows as the chart is on row 1 and the data is on row 2.

I changed my method from just adding the panels to the frame over to using GridBagLayout and the addComponentsToPane method.

```
//add the chart and data to the frame to return
JFrame frame = new JFrame(frameTitle);

//Set up the content pane by adding the chart panel and the data panel
//GridBagLayout manager time!
classes.ProfitByMonth_view.addComponentsToPane(frame.getContentPane(), jPanelChart, jPanelData);

//return the assembled report
return frame;

}
```

Development and Testing

I have documented the `addComponentsToPane` method and how `GridBagLayout` manager works in the Problems section where I solved this problem.

Basically I get the pane of the JFrame I am laying out and I get the chart panel and the data panel into my method. The pane for the frame is the container where I will display my chart and data.

Then I tell the pane that I am using the `GridBagLayout` manager.

Then I make a new `GridBagConstraints` object which is just an object containing all the attributes that control how the layout will be managed.

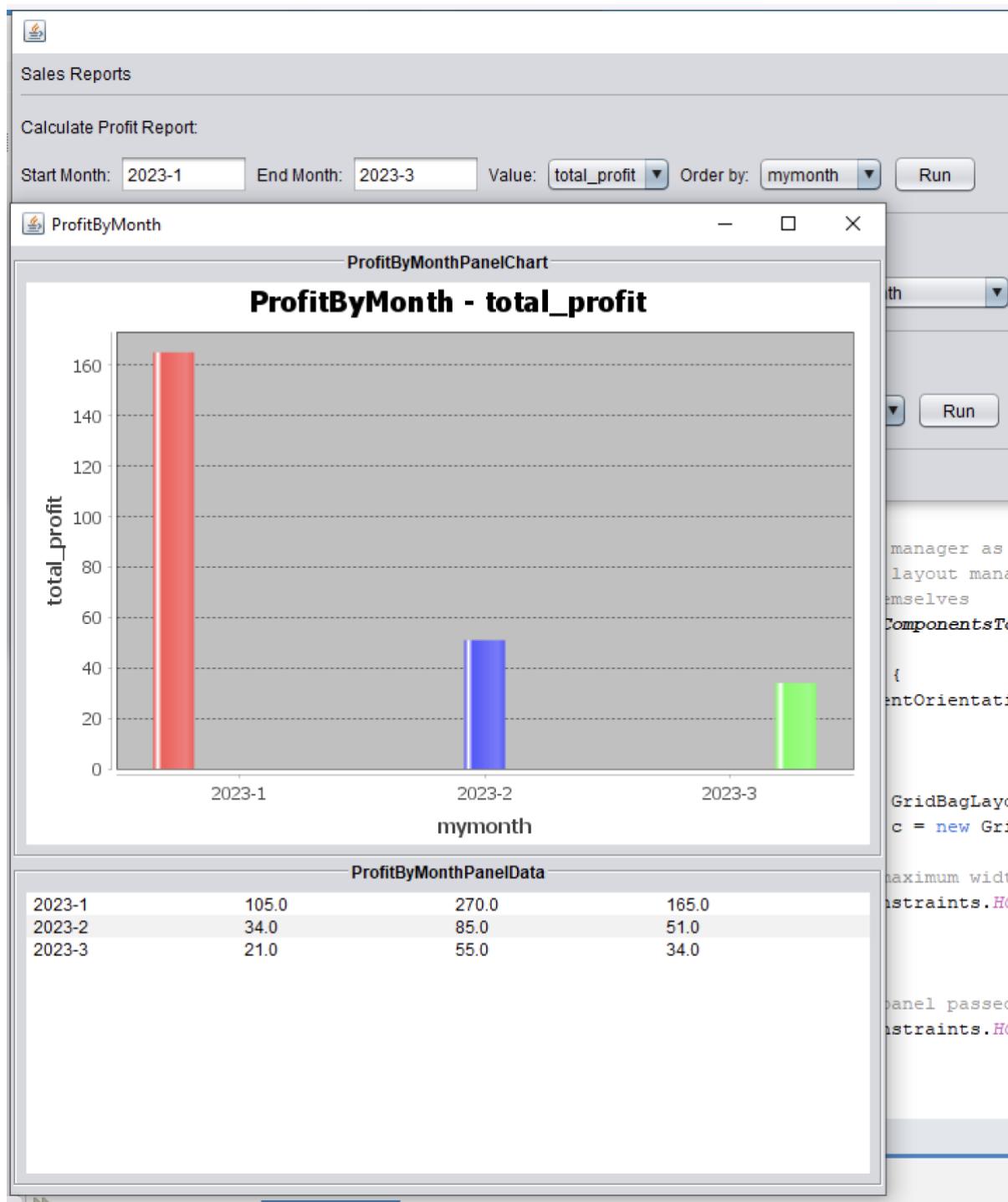
Then I set the constraints for the component like the chart panel and I add the panel to the pane also passing in the constraints I just set.

Then I repeat for the data panel but put it on the second row of the grid with gridy = 1.

Basically it is a grid layout with rows and columns and cells and for each cell I can set how it appears and if it is across multiple columns or in the first column or the second row and so on.

Development and Testing

So now when I use the GridBagLayout manager and probably I could have used the other layout managers too but not if I just add the panels to the frame then I can get what I want to appear.



4.10.7 Developing Most Sold Product Report

This report uses a class that is pretty much the same as the Profit By Month report. So for this report I made a class for the `MostSoldByMonth_view`.

```
public class MostSoldByMonth_view {

    Connection connection;

    //I added these as they are used by addComponentsToPane which is the GridBagL
    final static boolean shouldFill = false;
    final static boolean shouldWeightX = true;
    final static boolean RIGHT_TO_LEFT = false;

    //MostSoldByMonth_view attributes like the columns in the view same as those
    private String mymonth;
    private String product;
    private int total_quantity_sold;

    //methods to return the values of each attribute
    public String getmymonth(){
        return mymonth;
    }
    public String getproduct(){
        return product;
    }

    public int gettotal_quantity_sold(){
        return total_quantity_sold;
    }

    public MostSoldByMonth_view(){}
}

//MostSoldByMonth_view row of data object - used to create an array of rows o
public MostSoldByMonth_view(String mymonth, String product, int total_quantit
{
    this.mymonth = mymonth;
    this.product = product;
    this.total_quantity_sold = total_quantity_sold;
}
```

It's the same as the `ProfitByMonth_view` but it uses the `MostSoldByMonth_view` in MySQL.

```
141 ✘ SELECT * FROM MostSoldByMonth_view WHERE mymonth BETWEEN '2023-1' AND '2023-3' ORDER BY mymonth
142
```

Result Grid | Filter Rows: Export: Wrap Cell Content:

mymonth	product	total_quantity_sold
2023-1	24-chicken	3
2023-1	25-duck	3
2023-1	23-rice	9
2023-2	24-chicken	1
2023-2	25-duck	1
2023-2	23-rice	6
2023-3	23-rice	1
2023-3	24-chicken	1

Development and Testing

The assemble method to make up the report is the same as the ProfitByMonth_view class.

```
//assembler for the report - this gets the chart and the data table and lays them out using GridBagLayout layout
public static JFrame fillMostSoldByMonth_viewAssemble(String startDate, String endDate, String metricIn, String frameTitle)
{
    //make the chart
    ChartPanel chartpanel = classes.MostSoldByMonth_view.MostSoldByMonthPanel(startDate, endDate, metricIn, chartTitle);
    //set the size of the chart
    chartpanel.setPreferredSize(new Dimension(chartWidth, chartHeight));
    JPanel jPanelChart = new JPanel();
    jPanelChart.add(chartpanel);
    jPanelChart.setBorder(BorderFactory.createTitledBorder( BorderFactory.createEtchedBorder(), chartTitle, TitledBorder.CENTER, TitledBorder.TOP ));

    //make the data table
    JTable jTableData = classes.MostSoldByMonth_view.fillMostSoldByMonth_viewTable(startDate, endDate, orderbyVal);
    //set the size of the table
    jTableData.setPreferredSize(new Dimension(tableWidth, tableHeight));
    JPanel jPanelData = new JPanel();
    jPanelData.add(jTableData);
    jPanelData.setBorder(BorderFactory.createTitledBorder( BorderFactory.createEtchedBorder(), tableTitle, TitledBorder.CENTER, TitledBorder.TOP ));

    //new frame to show the GridBagLayoutDemo for the POS reports
    JFrame frame = new JFrame(frameTitle);

    //Set up the content pane by adding the chart panel and the data panel
    //GridBagLayout manager time!
    classes.MostSoldByMonth_view.addComponentstoPane(frame.getContentPane(), jPanelChart, jPanelData);

    //return the assembled report
    return frame;
}
```

The difference is that the methods refer to the MostSoldByMonth_view attributes which are the columns in the MySQL view MostSoldByMonth_view. For example the method to make the SQL query relates to the MostSoldByMonth_view.

```
//private method just to make the SQL for my report MostSoldByMonth_view
//I did this because I can't bind the order by field name in a prepare statement
//so I had to build the order clause onto the SQL like this
private static String MostSoldByMonth_viewSQL(String orderbyVal)
{
    //query to retrieve data from the db view where the category is like the report param passed in val
    String query = "SELECT * FROM MostSoldByMonth_view WHERE mymonth BETWEEN ? AND ? ORDER BY ";

    //append the correct ORDER BY column for the query based on the orderbyVal param
    //note this can in some cases be a risk for SQL injection but PreparedStatement doesn't allow ORDER BY ?
    if (orderbyVal=="mymonth") {
        query=query+"mymonth;";
    } else if (orderbyVal=="product") {
        query=query+"product;";
    } else if (orderbyVal=="total_quantity_sold") {
        query=query+"total_quantity_sold;";
    } else {
        query=query+"mymonth;"; //remove SQL injection risk by examining the orderbyVal and only allowing mymonth
    }

    return query;
}
```

Development and Testing

One other difference is that this report creates a stacked barchart. The total sold for each product is ordered and shown for each month. So the user can see how each product is performing and the product that sold the most for every month. This needed an extra category for the x axis so I did it like this.

```
DefaultCategoryDataset dataset = new DefaultCategoryDataset();

try{
    ps = connect.prepareStatement(query);

    ps.setString(1, startDate);
    ps.setString(2, endDate);

    System.out.println(ps);

    rs = ps.executeQuery();

    while (rs.next()){

        mymonth = rs.getString("mymonth");
        product = rs.getString("product");
        metric = rs.getFloat(metricIn);
        dataset.setValue(metric, product, mymonth);
    }
}
```

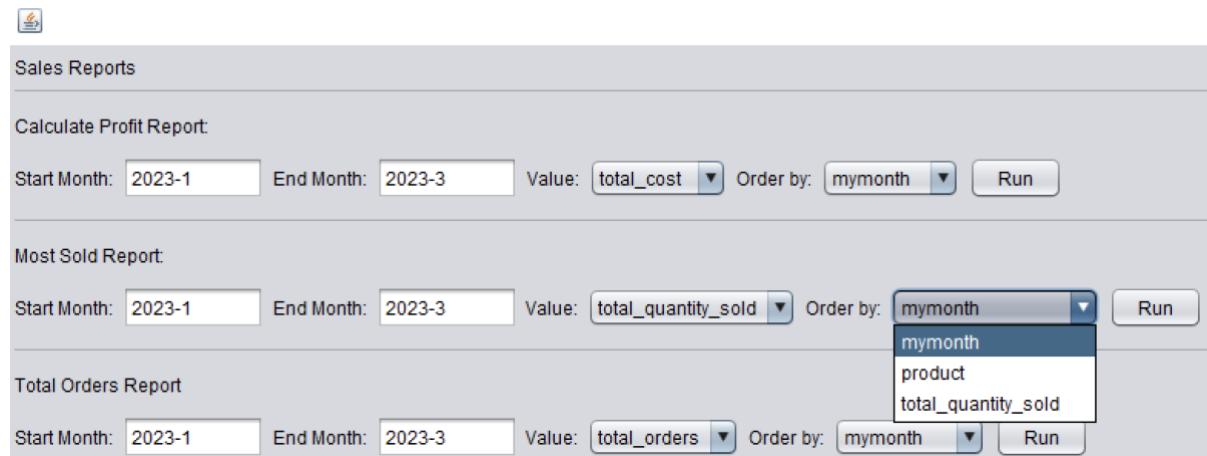
So for the months mymonth then I added the products and for each product the metricIn which is the report column chosen by the user in the report screen.

The screenshot shows a Java application window titled "Sales Reports". It contains three report configuration sections:

- Calculate Profit Report:** Fields: Start Month: 2023-1, End Month: 2023-3, Value: total_cost, Order by: mymonth, Run.
- Most Sold Report:** Fields: Start Month: 2023-1, End Month: 2023-3, Value: total_quantity_sold (highlighted in blue), Order by: mymonth, Run.
- Total Orders Report:** Fields: Start Month: 2023-1, End Month: 2023-3, Value: total_orders, Order by: mymonth, Run.

Development and Testing

And the user can choose the order by column like this so they can order the report by month or by product or by total_quantity sold.



The ChartFactory method is different from the ProfitByMonth method because I use a stacked barchart method provided by JFreeChart but it's not very different.

```
//make the chart, set the attributes, return the panel to the assembler
JFreeChart chart = ChartFactory.createStackedBarChart(
    "MostSoldByMonth - "+metricIn,"mymonth",metricIn,dataset,
    PlotOrientation.VERTICAL,false,true,false);
CategoryPlot p = chart.getCategoryPlot();
p.setRangeGridlinePaint(Color.black);

ChartPanel panel = new ChartPanel(chart);

return panel;

}
```

All the other methods are the same as ProfitByMonth except they refer to the MostSoldByMonth view attributes. For example.

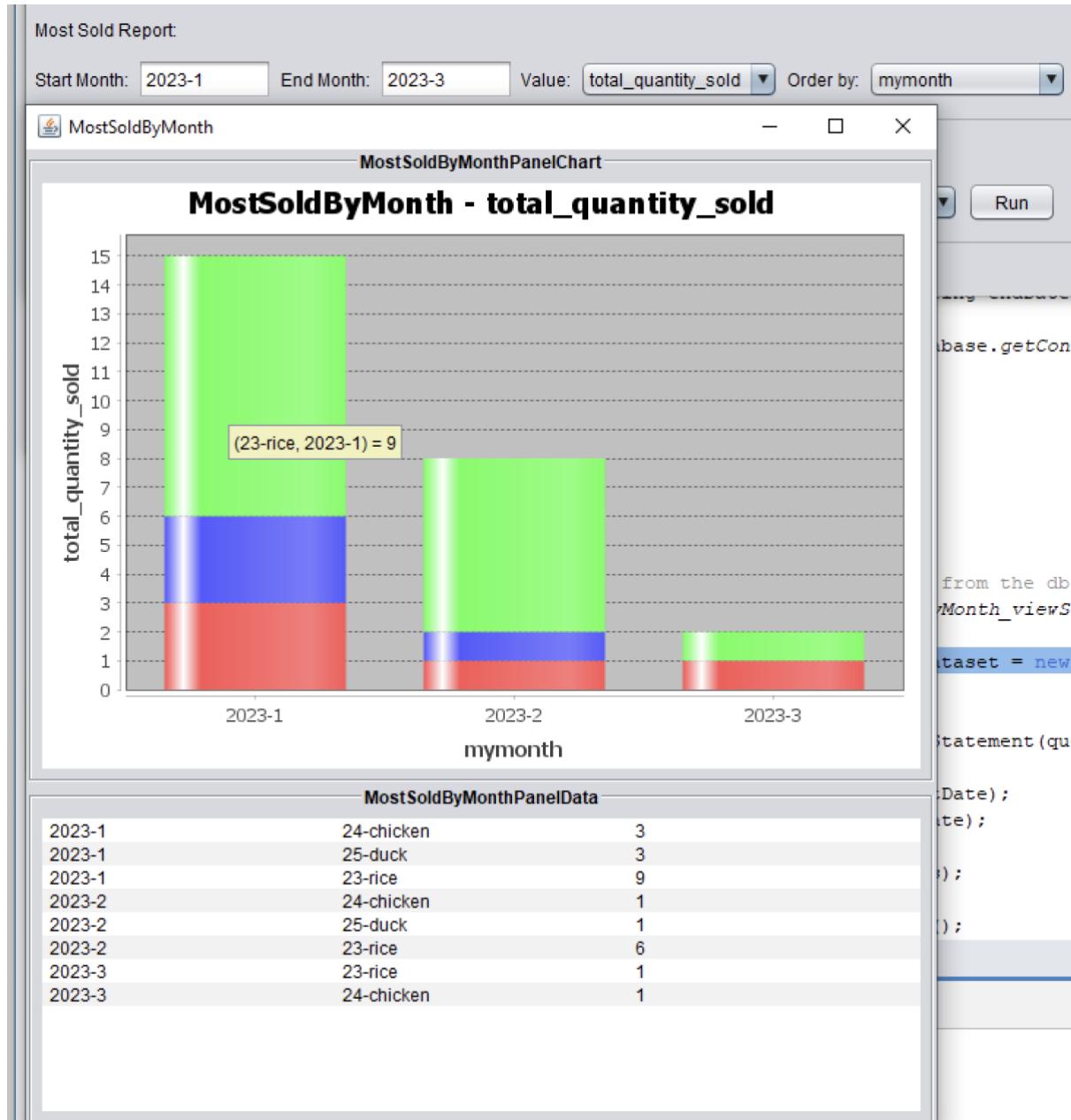
```
//fills and returns the jTable with contents of arrayList so it can be added to the data panel
public static JTable fillMostSoldByMonth_viewTable(String startDate, String endDate, String orderbyVal){

    classes.MostSoldByMonth_view MostSoldByMonth_view = new classes.MostSoldByMonth_view();
    ArrayList <classes.MostSoldByMonth_view> MostSoldByMonth_viewList = MostSoldByMonth_view.MostSoldByM

    String [] columnNames = {"mymonth","product","total_quantity_sold"};
    Object [][] rows = new Object[MostSoldByMonth_viewList.size()][3];
    
```

Development and Testing

Finally the report looks a bit different as it uses the stacked barchart. It shows clearly the most sold product as the biggest stack in the chart.



4.10.8 Developing Number Of Orders Report

The TotalOrdersByMonth report is the same as the other two reports but simpler. It just shows the total orders for each month in the date range.

```
public class TotalOrdersByMonth_view {

    Connection connection;

    //I added these as they are used by addComponentsToPane which is the GridBagLayout layout manager to
    final static boolean shouldFill = false;
    final static boolean shouldWeightX = true;
    final static boolean RIGHT_TO_LEFT = false;

    //TotalOrdersByMonth_view attributes
    private String mymonth;
    private int total_orders;

    //methods to return the values of each attribute
    public String getmymonth(){
        return mymonth;
    }

    public int gettotal_orders(){
        return total_orders;
    }

    public TotalOrdersByMonth_view(){}

    //TotalOrdersByMonth_view row of data object - used to create an array of rows of data - used to hold
    public TotalOrdersByMonth_view(String mymonth, int total_orders)
    {
        this.mymonth = mymonth;
        this.total_orders = total_orders;
    }

    //private method just to make the SQL for my report TotalOrdersByMonth
    private static String TotalOrdersByMonth_viewSQL(String orderbyVal)
    {
        //query to retrieve data from the db view where the category is like the report param passed in
        String query = "SELECT * FROM TotalOrdersByMonth_view WHERE mymonth BETWEEN ? AND ? ORDER BY ";

        //append the correct ORDER BY column for the query based on the orderbyVal param
        //note this can in some cases be a risk for SQL injection but PreparedStatement doesn't allow OR
        if (orderbyVal=="mymonth") {
            query=query+"mymonth;";
        } else if (orderbyVal=="total_orders") {
            query=query+"total_orders;";
        } else {
            query=query+"mymonth;"; //remove SQL injection risk by examining the orderbyVal and only all
        }

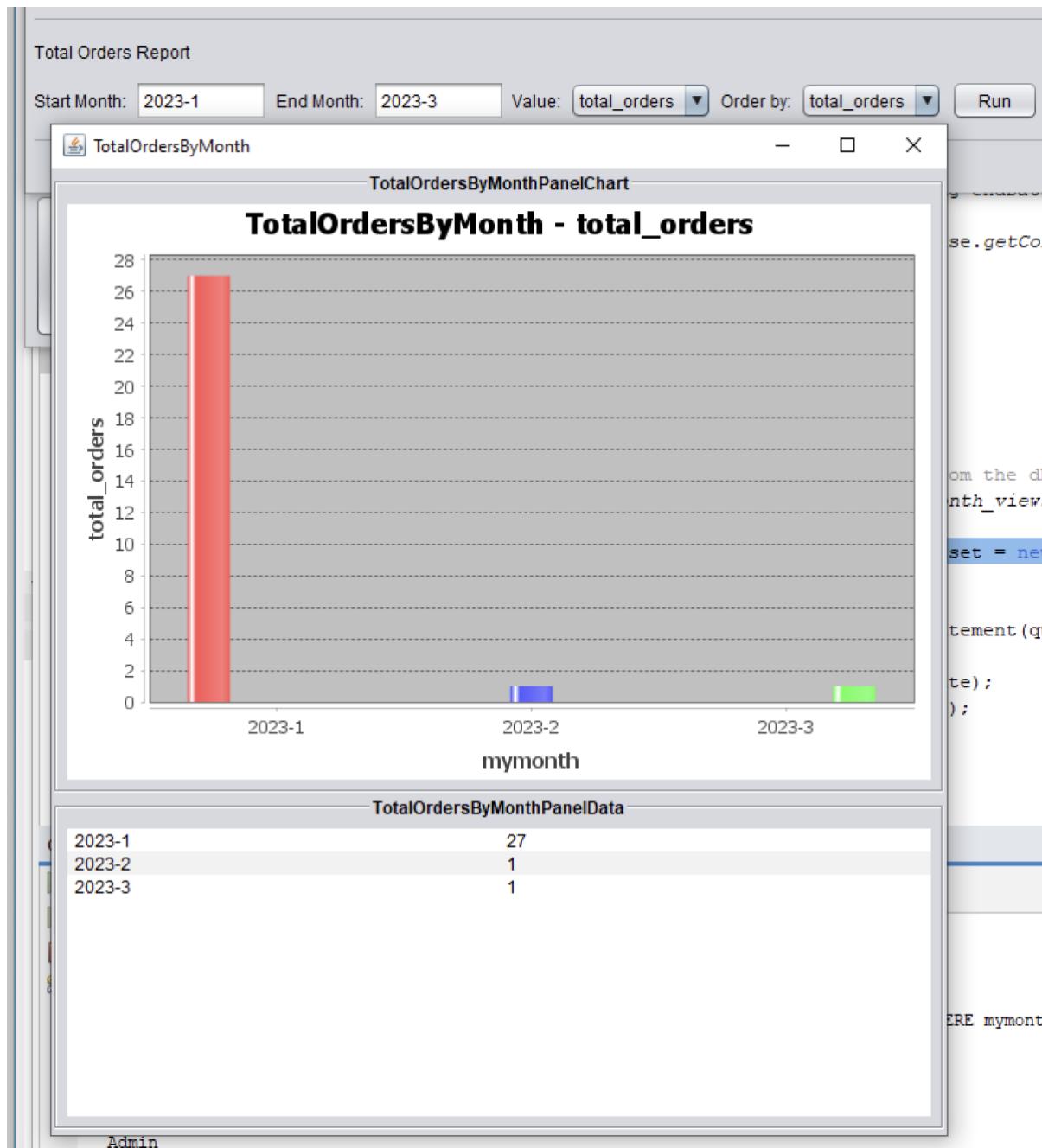
        return query;
    }
}
```

I just changed the view attributes to mymonth and total_orders. The methods are changed just to use these two attributes which are the columns in the report view. For example.

```
//create the view data row and loop through the data in the SQL and add each row to the arraylist
TotalOrdersByMonth_view TotalOrdersByMonth_viewRow;
while (rs.next()){
    TotalOrdersByMonth_viewRow = new TotalOrdersByMonth_view(rs.getString("mymonth"), rs.getInt("total_orders"));
    TotalOrdersByMonth_viewList.add(TotalOrdersByMonth_viewRow);
}
```

Development and Testing

Now the developed report looks like this showing the total orders for each month in the date range entered in the reports screen.



4.10.9 Test Plan

Test Number	How to Test	Expected Result
1	Create and confirm multiple orders	<p>Correct sales analytics generated.</p> <p>The profit for each month is calculated correctly.</p> <p>The product that was sold most shows in the report.</p> <p>The total number of orders for each month is correct.</p>

4.10.10 Evidence

- Screenshot: generated sales analytics.
- Screen Record: new sales analytics being generated as a new order is confirmed.
- Screenshot: sales analytics displayed.

4.10.10.1 Profit by month report evidence

I made a SQL which will show the database data for all the orders and order products for every month like this. I can use the ROW_NUMBER() OVER (ORDER BY column_name) MySQL function to give me the number of records for the test data.

The test SQL will get the total profit using the algorithm:

Profit = total (order product sell price – order product cost price) * order product quantity

```

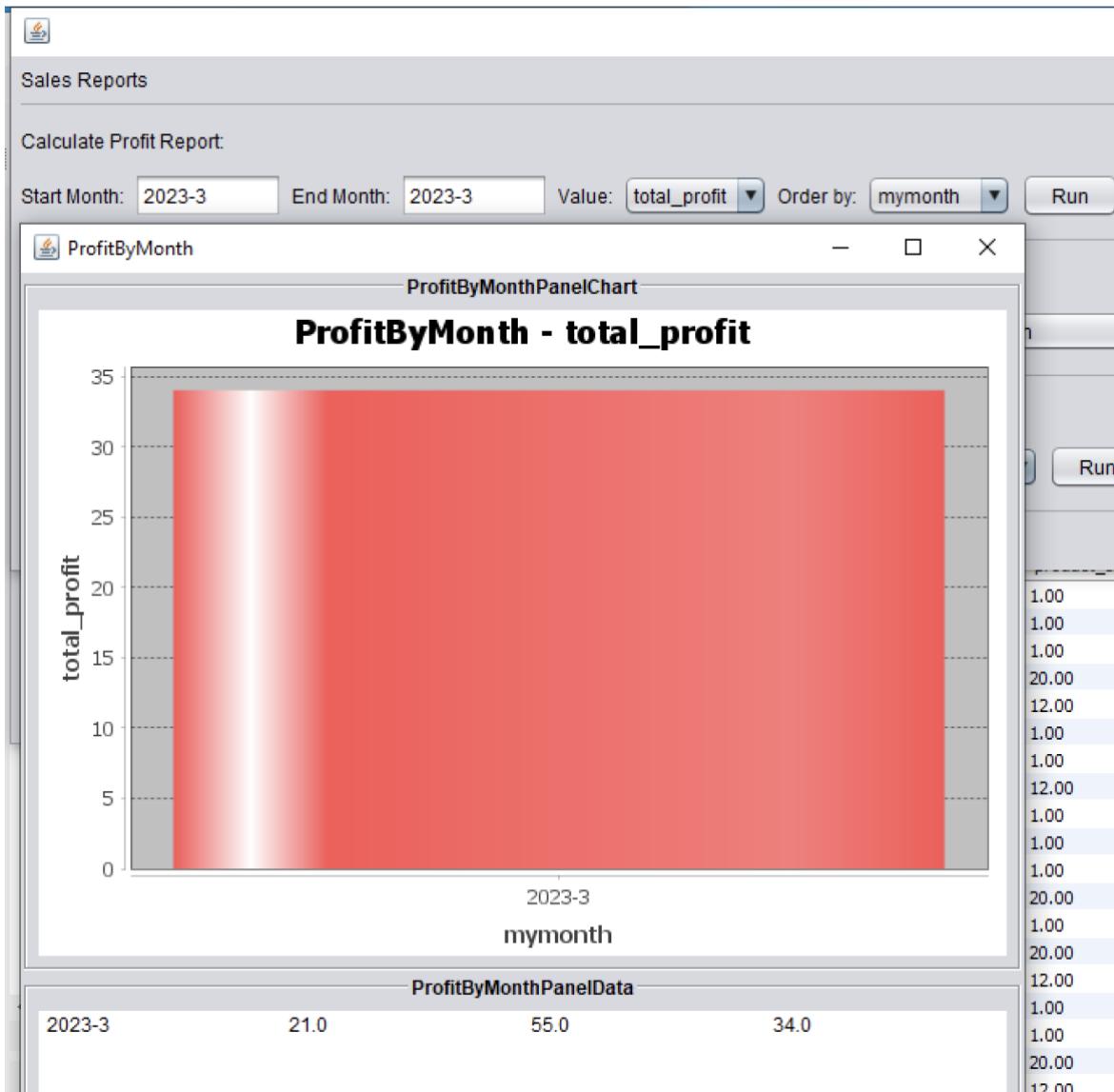
156      -- all the profit on the products * quantity ordered on every order test data SQL
157 •  SELECT
158    row_number() over (order by order_pk) as row_count,
159    CONCAT(YEAR(o.date),"-", MONTH(o.date)) mymonth,
160    order_pk,
161    product_pk,
162    product_name,
163    product_cost,
164    product_sellprice,
165    op.quantity as order_product_quantity,
166    op.quantity * product_cost as total_product_cost,
167    op.quantity * product_sellprice as total_product_sellprice,
168    (op.quantity * product_sellprice) - (op.quantity * product_cost) as total_product_profit
169    FROM
170    orders o, products p, orderproducts op
171    WHERE
172    o.order_pk=op.order_fk
173    AND
174    op.product_fk=p.product_pk
175    ORDER BY
176    order_pk,
177    product_pk;
```

Development and Testing

This produces the following results showing the total profit for every order product line on every order.

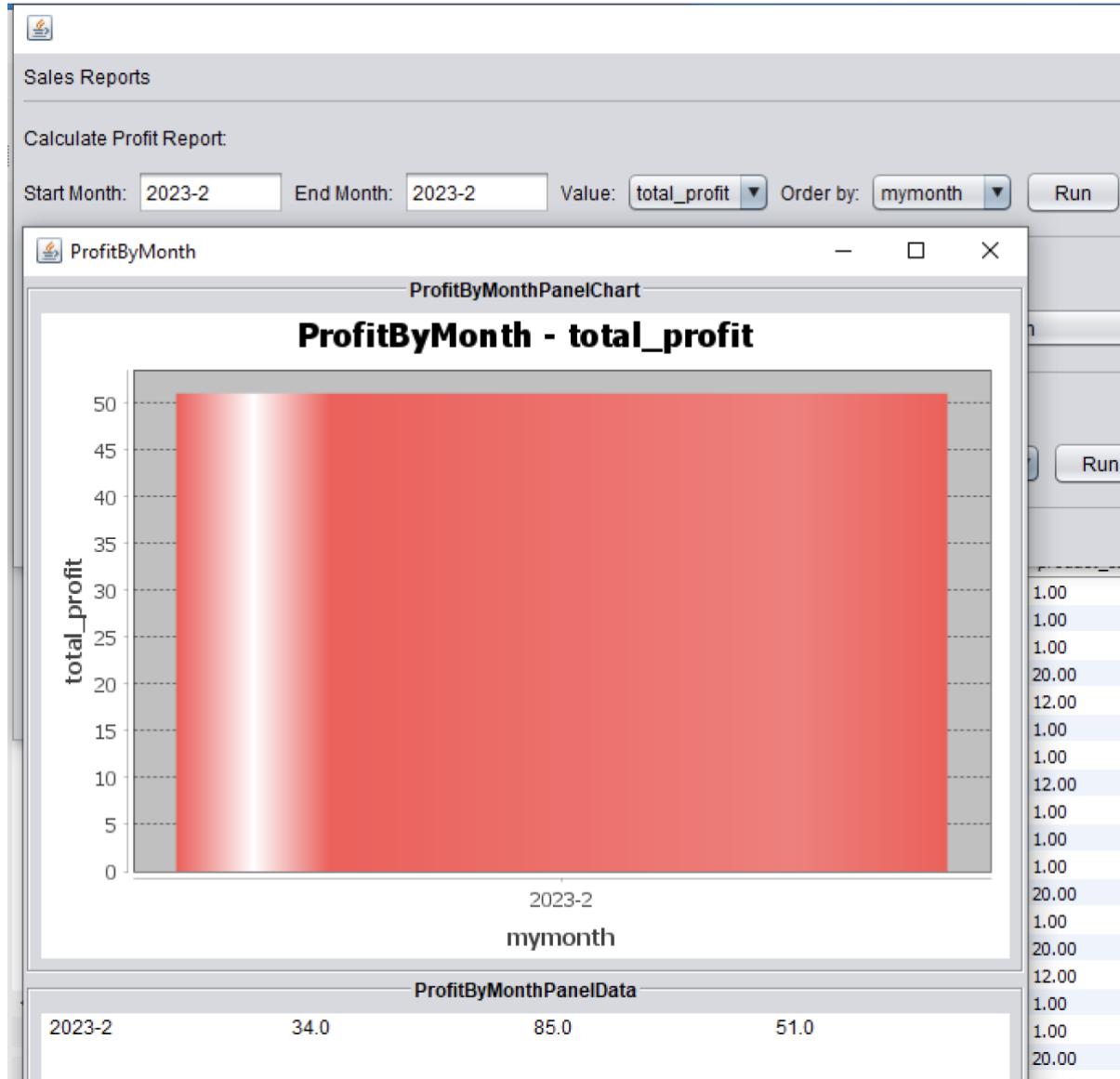
row_count	mymonth	order_pk	product_pk	product_name	product_cost	product_sellprice	order_product_quantity	total_product_cost	total_product_sellprice	total_product_profit
1	2023-1	15	23	rice	1.00	5.00	1	1.00	5.00	4.00
2	2023-1	16	23	rice	1.00	5.00	1	1.00	5.00	4.00
3	2023-1	17	23	rice	1.00	5.00	1	1.00	5.00	4.00
4	2023-1	18	24	chicken	20.00	50.00	1	20.00	50.00	30.00
5	2023-1	19	25	duck	12.00	25.00	1	12.00	25.00	13.00
6	2023-1	20	23	rice	1.00	5.00	1	1.00	5.00	4.00
7	2023-1	21	23	rice	1.00	5.00	1	1.00	5.00	4.00
8	2023-1	22	25	duck	12.00	25.00	1	12.00	25.00	13.00
9	2023-1	23	23	rice	1.00	5.00	1	1.00	5.00	4.00
10	2023-1	24	23	rice	1.00	5.00	1	1.00	5.00	4.00
11	2023-1	25	23	rice	1.00	5.00	1	1.00	5.00	4.00
12	2023-1	26	24	chicken	20.00	50.00	1	20.00	50.00	30.00
13	2023-1	28	23	rice	1.00	5.00	1	1.00	5.00	4.00
14	2023-1	29	24	chicken	20.00	50.00	1	20.00	50.00	30.00
15	2023-1	30	25	duck	12.00	25.00	1	12.00	25.00	13.00
16	2023-2	31	23	rice	1.00	5.00	1	1.00	5.00	4.00
19	2023-2	31	23	rice	1.00	5.00	5	5.00	25.00	20.00
17	2023-2	31	24	chicken	20.00	50.00	1	20.00	50.00	30.00
18	2023-2	31	25	duck	12.00	25.00	1	12.00	25.00	13.00
20	2023-3	32	23	rice	1.00	5.00	1	1.00	5.00	4.00
21	2023-3	32	24	chicken	20.00	50.00	1	20.00	50.00	30.00

So if I run the report for 2023-3 my data has 1 order with 2 products with a total profit of 34.



This is correct.

I ran the report for 2023-2 so my data has 1 order with 4 products with a total profit of 67.



So I found a bug in my profit report as the total profit of 51 is not correct. I went and checked my ProfitByMonth_view SQL by getting MySQL to generate the script to create my view.

Development and Testing

```
-- found a bug in the profit
-- so there is no quantity in my SQL code for the order products
CREATE ALGORITHM=UNDEFINED DEFINER='root'@'localhost' SQL SECURITY DEFINER VIEW `profitbymonth_view` AS
select concat(year(`o`.`date`),'-',month(`o`.`date`)) AS `mymonth`,
sum(`p`.`product_cost`) AS `total_cost`,
sum(`p`.`product_sellprice`) AS `total_price`,
sum(`p`.`product_sellprice` - `p`.`product_cost`) AS `total_profit`
from ((`orders` `o` join `products` `p`) join `orderproducts` `op`)
where ((`o`.`order_pk` = `op`.`order_fk`)
and (`op`.`product_fk` = `p`.`product_pk`))
group by concat(year(`o`.`date`),'-',month(`o`.`date`))
order by concat(year(`o`.`date`),'-',month(`o`.`date`));
```

So one of the benefits of using a MySQL SQL view is that I can hopefully just fix the SQL for my report view and then it should fix everything.

I didn't put the quantity in my SQL for the report view for the order_products.

```
192 -- all the profit on the products * quantity ordered on every order test data SQL
193 -- for the buggy year 2023-2
194 • SELECT
195 row_number() over (order by order_pk) as row_count,
196 CONCAT(YEAR(o.date),"-", MONTH(o.date)) mymonth,
197 order_pk,
198 product_pk,
199 product_name,
200 product_cost,
201 product_sellprice,
202 op.quantity as order_product_quantity,
203 op.quantity * product_cost as total_product_cost,
204 op.quantity * product_sellprice as total_product_sellprice,
205 (op.quantity * product_sellprice) - (op.quantity * product_cost) as total_product_profit
206 FROM
207 orders o, products p, orderproducts op
208 WHERE
209 o.order_pk=op.order_fk
210 AND
211 op.product_fk=p.product_pk
212 AND
213 CONCAT(YEAR(o.date),"-", MONTH(o.date)) = "2023-2"
214 ORDER BY
215 order_pk,
216 product_pk;
217
```

Result Grid										
row_count	mymonth	order_pk	product_pk	product_name	product_cost	product_sellprice	order_product_quantity	total_product_cost	total_product_sellprice	total_product_profit
1	2023-2	31	23	rice	1.00	5.00	1	1.00	5.00	4.00
4	2023-2	31	23	rice	1.00	5.00	5	5.00	25.00	20.00
2	2023-2	31	24	chicken	20.00	50.00	1	20.00	50.00	30.00
3	2023-2	31	25	duck	12.00	25.00	1	12.00	25.00	13.00

So if I change the report view SQL it should be OK.

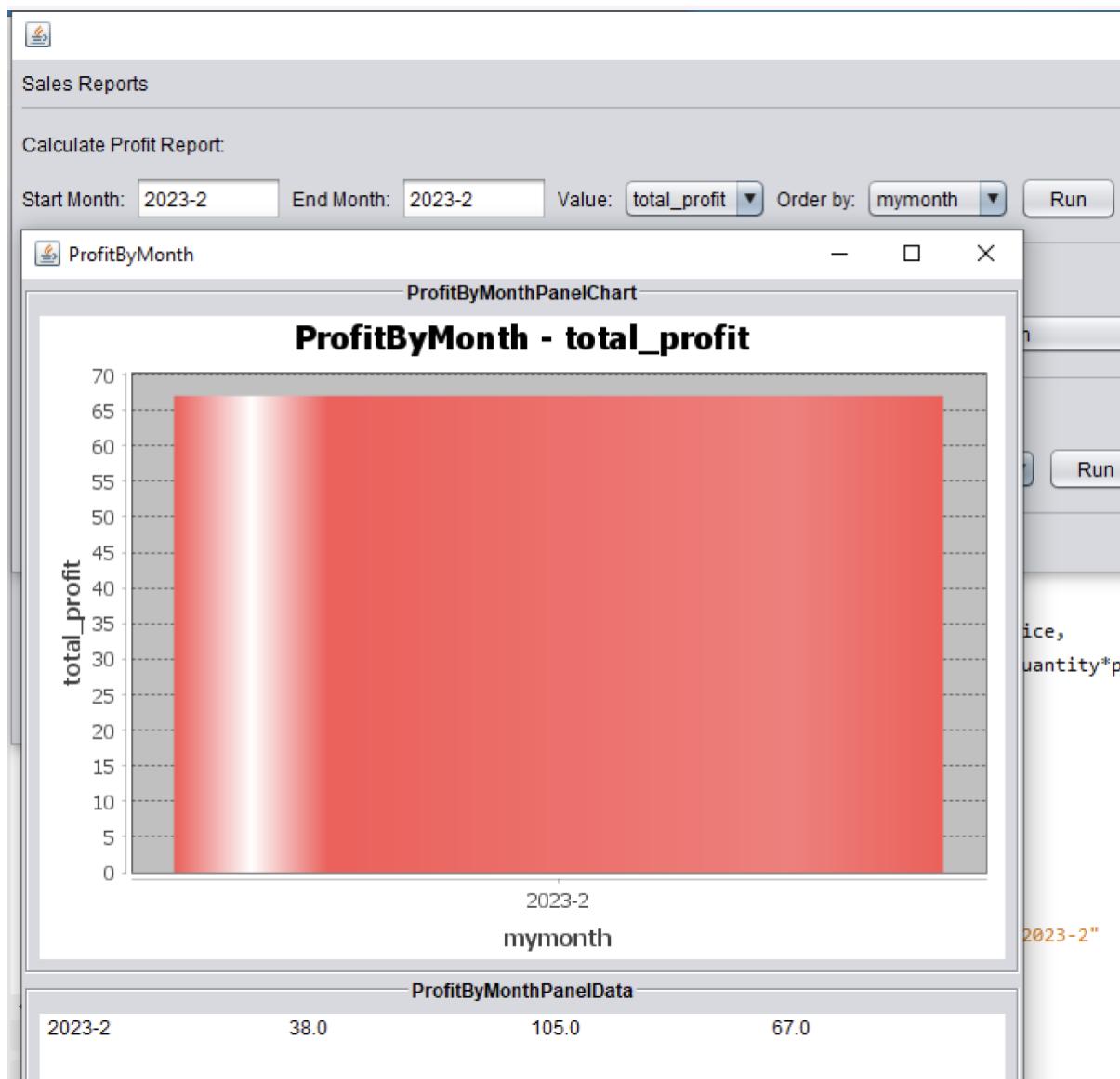
Development and Testing

```
--  
218    -- fix for the profit by month view  
219 • ALTER VIEW profitbymonth_view AS  
220     SELECT  
221        CONCAT(YEAR(o.date),"-", MONTH(o.date)) mymonth,  
222        -- order_pk,  
223        -- product_pk,  
224        SUM(op.quantity*product_cost) total_cost,  
225        SUM(op.quantity*product_sellprice) total_price,  
226        SUM(op.quantity*product_sellprice)-SUM(op.quantity*product_cost) total_profit  
227     FROM  
228     orders o, products p, orderproducts op  
229     WHERE  
230     o.order_pk=op.order_fk  
231     AND  
232     op.product_fk=p.product_pk  
233     AND  
234     CONCAT(YEAR(o.date),"-", MONTH(o.date)) = "2023-2"  
235     GROUP BY  
236     CONCAT(YEAR(o.date),"-", MONTH(o.date))  
237     ORDER BY  
238     CONCAT(YEAR(o.date),"-", MONTH(o.date));  
239  
240 •   select * from profitbymonth_view where mymonth = "2023-2";  
241
```

The screenshot shows a database query results grid. At the top, there are navigation buttons for back, forward, and search, followed by tabs for 'Result Grid' and 'Export'. Below the tabs is a search bar labeled 'Filter Rows:' with a placeholder 'Search' and an 'Export' button. The main area displays a single row of data in a table:

	mymonth	total_cost	total_price	total_profit
▶	2023-2	38.00	105.00	67.00

So now my report view is changed to use the quantity so the report should be OK in the system.



And this has fixed my report so the total profit for the month 2023-2 is now 67 which is correct.

4.10.10.2 Most sold product by month report evidence

I made a SQL which will show the database data for all the orders and order products for every month like this. I can use the ROW_NUMBER() OVER (ORDER BY column_name) MySQL function to give me the number of records for the test data.

I can see the most sold product is rice for 2023-1 and the same rice for 2023-2 and 2023-3 is equal for rice and chicken.

Development and Testing

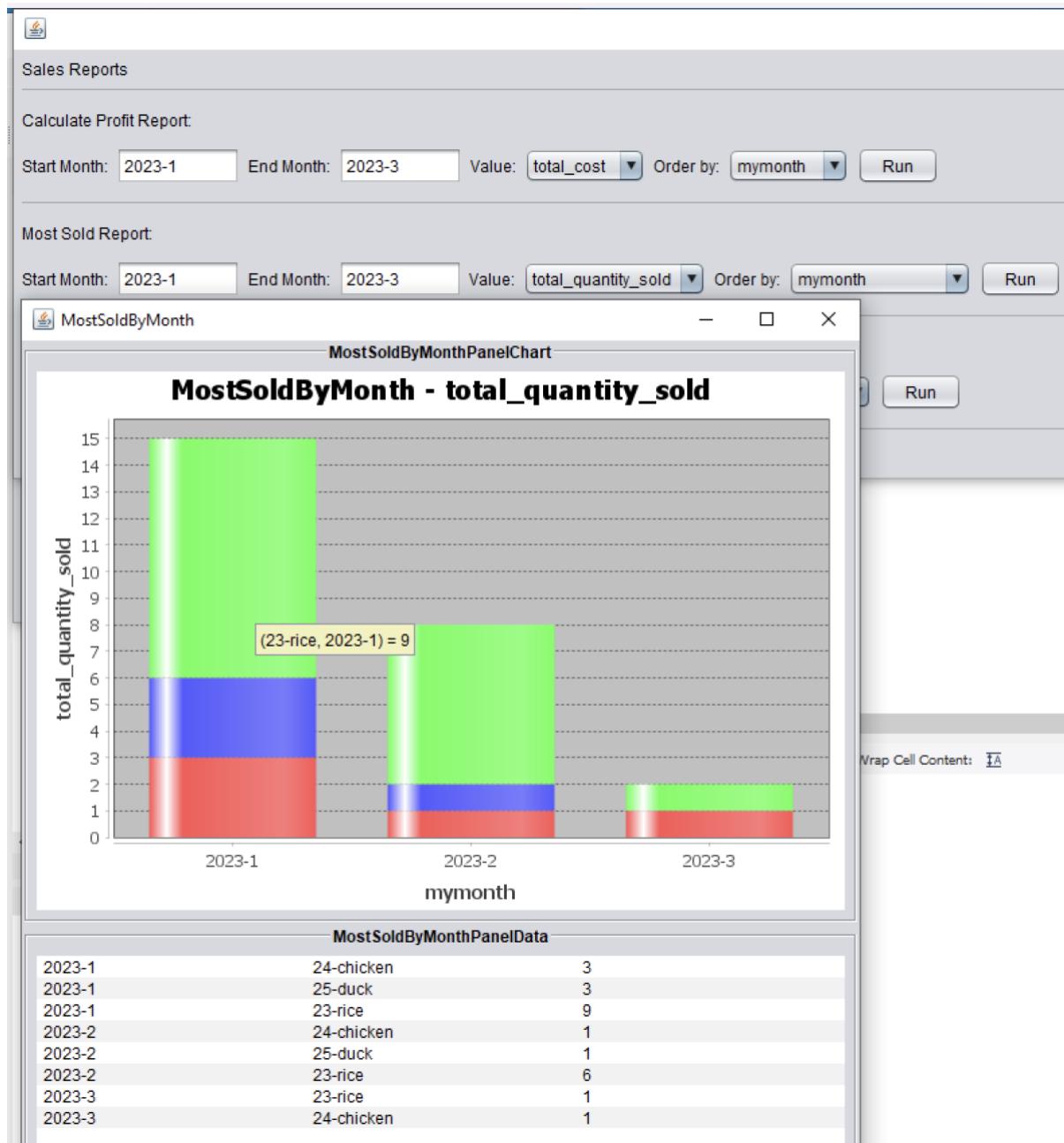
```
245      -- most sold products by month test data SQL
246  •  SELECT
247      CONCAT(YEAR(o.date),"-", MONTH(o.date)) mymonth,
248      order_pk,
249      product_fk,
250      CONCAT(product_pk,"-",product_name) product,
251      op.quantity
252  FROM
253      orders o, products p, orderproducts op
254  WHERE
255      o.order_pk=op.order_fk
256  AND
257      op.product_fk=p.product_pk
258  ORDER BY
259      CONCAT(YEAR(o.date),"-", MONTH(o.date)),
260      product_fk
```

The screenshot shows a database query results grid. The grid has columns: mymonth, order_pk, product_fk, product, and quantity. The data is as follows:

	mymonth	order_pk	product_fk	product	quantity
▶	2023-1	15	23	23-rice	1
	2023-1	16	23	23-rice	1
	2023-1	17	23	23-rice	1
	2023-1	20	23	23-rice	1
	2023-1	21	23	23-rice	1
	2023-1	23	23	23-rice	1
	2023-1	24	23	23-rice	1
	2023-1	25	23	23-rice	1
	2023-1	28	23	23-rice	1
	2023-1	18	24	24-chicken	1
	2023-1	26	24	24-chicken	1
	2023-1	29	24	24-chicken	1
	2023-1	19	25	25-duck	1
	2023-1	22	25	25-duck	1
	2023-1	30	25	25-duck	1
	2023-2	31	23	23-rice	1
	2023-2	31	23	23-rice	5
	2023-2	31	24	24-chicken	1
	2023-2	31	25	25-duck	1
	2023-3	32	23	23-rice	1
	2023-3	32	24	24-chicken	1

Development and Testing

I ran the most sold by month report for all the months.



So my report is correct and the stacked barchart shows the right most sold products for each month.

4.10.10.3 Total orders by month report evidence

I made a SQL which will show the database data for all the orders for every month like this. I can use the ROW_NUMBER() OVER (ORDER BY column_name) MySQL function to give me the number of records for the test data.

```

144      -- total orders by month test data SQL
145 •  SELECT
146      row_number() over (order by order_pk) as row_count,
147      CONCAT(YEAR(o.date),"-", MONTH(o.date)) mymonth,
148      order_pk order_id
149  FROM
150  orders o
151  WHERE CONCAT(YEAR(o.date),"-", MONTH(o.date)) BETWEEN '2023-1' AND '2023-3'
152  ORDER BY
153      CONCAT(YEAR(o.date),"-", MONTH(o.date));
154

```

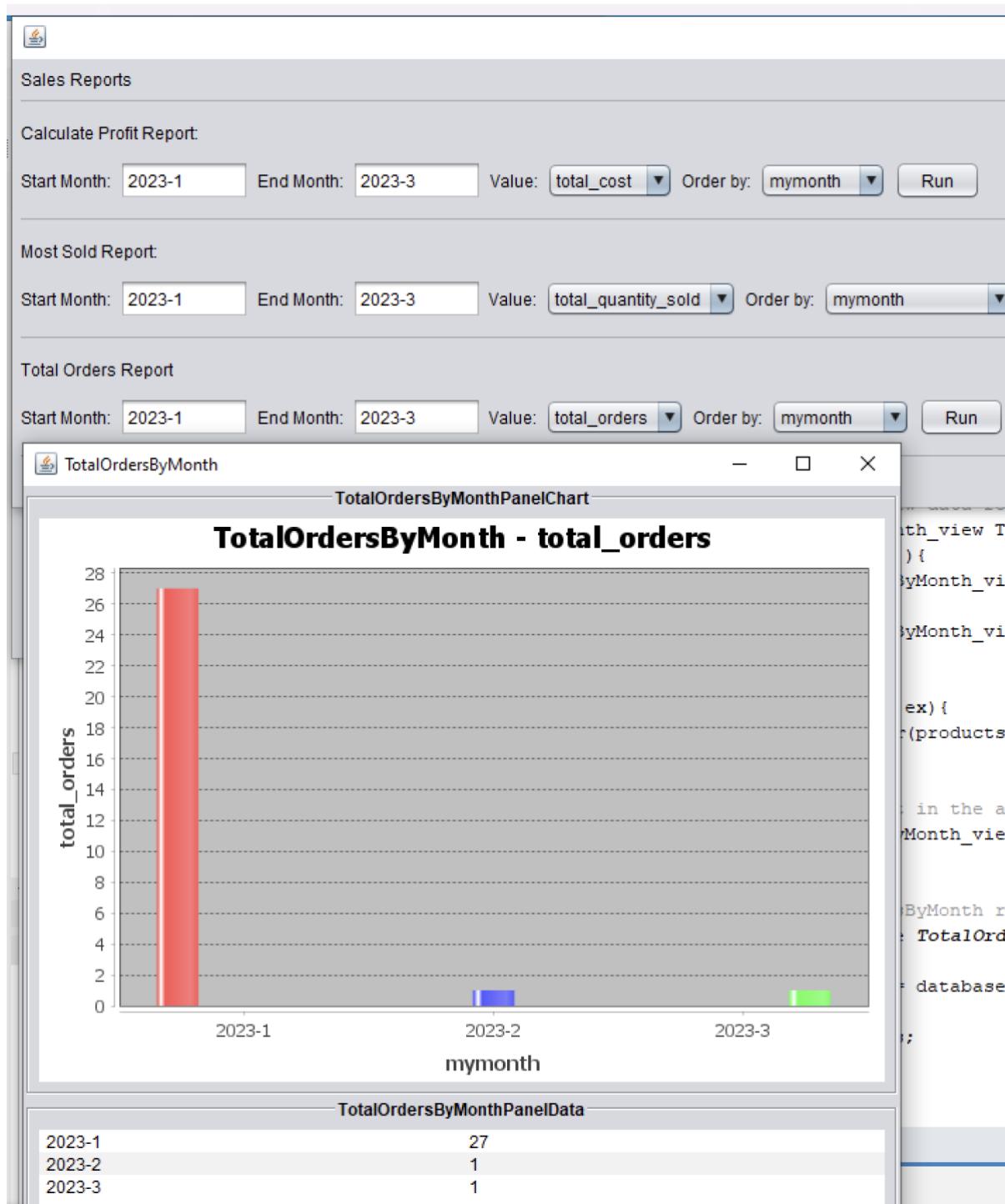
	row_count	mymonth	order_id
▶	1	2023-1	4
	2	2023-1	5
	3	2023-1	6
	4	2023-1	7
	5	2023-1	8
	6	2023-1	9
	7	2023-1	10
	8	2023-1	11
	9	2023-1	12
	10	2023-1	13
	11	2023-1	14
	12	2023-1	15
	13	2023-1	16
	14	2023-1	17
	15	2023-1	18
	16	2023-1	19
	17	2023-1	20
	18	2023-1	21
	19	2023-1	22
	20	2023-1	23
	21	2023-1	24
	22	2023-1	25
	23	2023-1	26
	24	2023-1	27
	25	2023-1	28
	26	2023-1	29
	27	2023-1	30
	28	2023-2	31
	29	2023-3	32

Result 74 ×

So I can see I have 27 orders for 2023-01 and 1 order for 2023-2 and 1 order for 2023-3.

Development and Testing

Now I can run the report in my POS system to see what the total orders for the months is in my report I developed.



This is correct, 27 orders for 2023-1 and 1 order for 2023-2 and 1 order for 2023-3.

4.10.11 Success Criteria for Sales Reports

Success Criteria Number	Success Criteria	Achieved	Notes
7	Create and display Sales Reports/analytics	Yes but I found a bug in the profit report so I fixed it.	The sales reports work. The orders in the database show in the sales reports. The profit and the most sold products and the total orders by month are all calculated correctly now I have fixed the bug in the profit by month view.

4.10.12 Review/Stage Evaluation

In this stage I developed the functionality behind the sales report section. Originally in my plan I intended to use a table in the database to store some relatively simple data about sales such as total sales for a specific month etc. However, after I found views, I changed my design and development plan to instead use views. Views are virtual tables based on the results of a query; the query is saved in the database.

Views allow you to join data from multiple tables and represent it as a single table. When the view is referenced, the query it is based on will run so you can see any changes made to the database tables the view is based on. For example, in a simple view that sums up the price of every order, when a new order is created and the view is next referenced, the query will be re-run so the view will include the price of the new order. Using a view is much better than using a table when generating sale statistics because the view will automatically update whereas with a table new sale statistic would have to be calculated each time there is a new order and inserted into the table.

Initially I had to research jFreeChart to understand how to use the library to generate bar charts and stacked bar charts using the data from views in the database. I used various websites to do this such as the oracle website and the jfree website.

This stage went well as I was able to fully achieve the success criteria however it did take quite a long time leaving me with less time to finish off the rest of the project however, that shouldn't be a problem as the next two stages are relatively straightforward.

4.11 Stage 10 Clock In

I have decided to code this section differently to how I designed it. I am going to delete the userhours table and only use the dailyhours table. When a user clocks in a record will be created in the dailyhours table and when they clock out their record will be updated with the clockout time. Their hours will also be calculated for that record. When I am displaying the user hours to the user I will sum up the hours field for the selected month for each user and display that amount.

4.11.1 Username And Password Validation + user primary key retrieval

In the clock in screen the user will enter their username and password and press enter. I need to develop the code to validate that their username and password is the correct length and is correct. Also I need to retrieve their primary key so I can use it when inserting records into the dailyhours table when they clock in.

I have already developed the code to validate the username and password in the login section so I will simply copy and past that code into this class under the action listener method on the enter button:

```
private void enterButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String userName = usernameTextField.getText();
    String password = passwordTextField.getText();
    int uLen = userName.length();
    int pLen = password.length();
    PreparedStatement ps;
    ResultSet rs;
    int userPK;

    try {
        if (uLen < 6 || pLen < 6) {
            JOptionPane.showMessageDialog(null, "invalid, username and password must be 6 or more characters long");
        }
        else {
            ps = classes.database.getConnection().prepareStatement("SELECT `username`, `password`, `user_pk` PK FROM `u
            ps.setString(1, usernameTextField.getText());
            ps.setString(2, passwordTextField.getText());
            rs = ps.executeQuery();

            if (rs.next()){

                userPK = rs.getInt("PK");
                menuScreen menu = new menuScreen();
                //menu.sessionID = currentSession;
                menu.setDefaultCloseOperation(menuScreen.EXIT_ON_CLOSE);
                menu.pack();
                menu.setVisible(true);

                int currentSession = classes.userSessions.insertUserSession(userPK);
                System.out.println(Integer.toString(currentSession));
                menu.sessionID = currentSession;
                menu.userPrimaryKey = userPK;
                this.dispose();

            }
            else{
                JOptionPane.showMessageDialog(null, "incorrect username or password");
            }
        }
    }
    catch(SQLException ex){
        Logger.getLogger(loginScreen.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

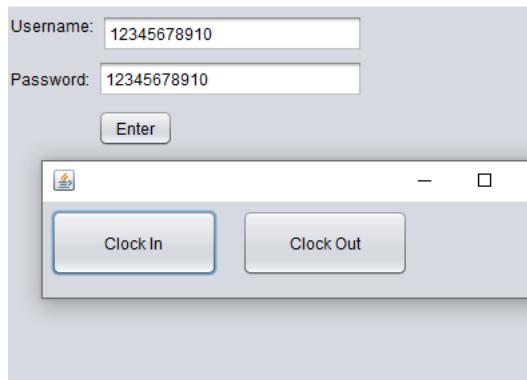
Now all I need to do is edit the code so it calls the choiceClockInScreen jframe instead of the menu jframe.

The code is below:

```
if (rs.next()) {

    userPK = rs.getInt("PK");
    choiceClockinScreen choiceScreen = new choiceClockinScreen();
    //menu.sessionID = currentSession;
    choiceScreen.setDefaultCloseOperation(choiceClockinScreen.EXIT_ON_CLOSE);
    choiceScreen.pack();
    choiceScreen.setVisible(true);
}
```

After testing this code it worked as seen below:



Lastly I need to pass the retrieved user primary key into the choice clock in screen. First I create an integer global variable called userPrimaryKey in the choice clock in screen class then I set this variable equal to userPK which has the value of the retrieved user primary key.

```
choiceScreen.userPrimaryKey = userPK;
```

4.11.2 Developing Clock In Method

Next I have created a clockin class that will hold the methods for clocking in and out. First I will develop the clock in method.

This method will have an integer parameter that will be used to pass in the correct user primary key so the user_fk record in the dailhours table is correct. This method will simply insert sysdate() and the variable holding the primary key into the dailyhours table so a new record is created showing this user has clocked in. The code is below:

Development and Testing

```
public static void clockUserIn(int userPK) {
    Connection connection = database.getConnection();
    PreparedStatement ps;
    ResultSet rs;

    try{
        ps = connection.prepareStatement("INSERT INTO dailyhours (clockin, user_fk) VALUES (sysdate(),?)");
        ps.setInt(1, userPK);

        if (ps.executeUpdate() != 0){
            JOptionPane.showMessageDialog(null, "You are clocked in");
        }
        else{
            JOptionPane.showMessageDialog(null, "Error");
        }
    }
    catch(SQLException ex){
        Logger.getLogger(clockin.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

userPK is the integer parameter used to pass in the user primary key.

The insert query simply inserts sysdate(), which is the current date, into the clockin field and inserts ? into user_fk.

Then I used the setInt() method to set the ? placeholder to userPK.

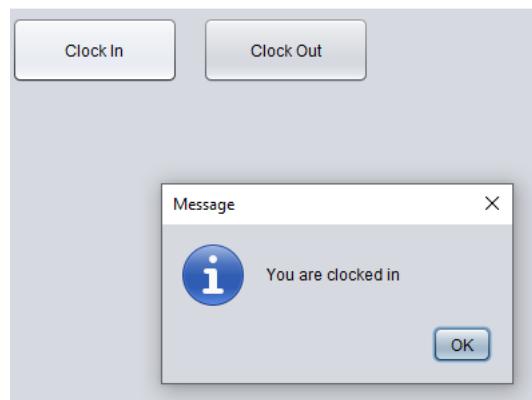
If ps.executeUpdate() returns 0 then 0 records were affected so the method has failed but if it return anything other than 0 then atleast 1 record was affected so the method was a success.

4.11.3 Clock In Button Code

Next I need to call the clock in method under the clock in button and pass in the primary key variable. The code is below:

```
private void clockinButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    classes.clockin.clockUserIn(userPrimaryKey);
}
```

After testing this method these are the results I got:



	dailyhour_pk	clockin	clockout	hours	user_fk
▶	2	2023-01-04 16:56:36	NULL	0.0	8

As you can see the method worked.

4.11.4 Developing Clock Out User Method

Now I need to develop a method to update the users record in the dailyhours table and add the clockout time when the user presses the clock out button. I will do this using an update query and use sysdate() to set the clockout field value to the current datetime. The code is below:

```
public static void clockUserOut(int userPK) {
    Connection connection = database.getConnection();
    PreparedStatement ps;

    try{
        ps = connection.prepareStatement("UPDATE dailyhours SET clockout = sysdate() WHERE user_fk = ?");
        ps.setInt(1, userPK);

        if (ps.executeUpdate() != 0){
            JOptionPane.showMessageDialog(null, "You have been clocked out.");
        }
        else{
            JOptionPane.showMessageDialog(null, "There was an error.");
        }
    }
    catch(SQLException ex){
        Logger.getLogger(clockin.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

I have passed in the integer parameter userPK and use dthe setInt() method to set the ? placeholder in the where clause to the value of userPK.

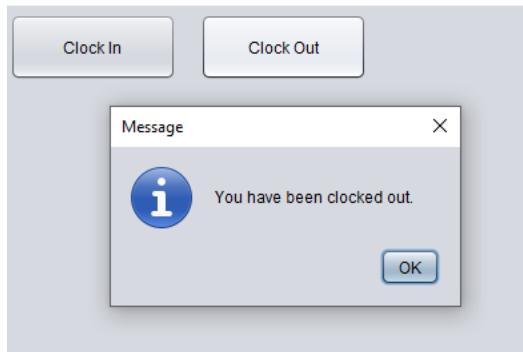
userPK will hold the value of the users primary key so the update statement will only update their record.

4.11.5 Adding Clock Out Button Code

Now I need to add the code to call the clock out method under the clock out button. Here is where I will pass in the users primary key using the userPrimaryKey variable as I have in other methods in this class.

```
private void clockoutButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    classes.clockin.clockUserOut(userPrimaryKey);
}
```

This method worked as seen below:



	dailyhour_pk	clockin	clockout	hours	user_fk
▶	6	2023-01-04 18:01:21	2023-01-04 20:00:37	0.0	8

Lastly I need to add the dispose() method to dispose of the clockinChoiceScreen once the clock out button has been pressed.

```
private void clockoutButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    classes.clockin.clockUserOut(userPrimaryKey);
    this.dispose();
}
```

4.11.6 Developing Calculate Hours Method

Now I need to develop the method that will calculate the difference between the clock in and out times in hours and insert this value into the hours field in the dailyhours table. I will do this using the timestampdiff() function in MYSQL. This function takes three parameters, one is the unit which in my case will be "HOURS" and the other two are the values that the function will be calculating the difference between, in this case this will be the clockin and clockout field. I will user an update query.

```
public static void calculateUserHours(int userPK) {
    Connection connection = database.getConnection();
    PreparedStatement ps;

    try{
        ps = connection.prepareStatement("UPDATE dailyhours SET hours = TIMESTAMPDIFF(HOURS, clockin, clockout) WHERE user_fk = ?");
        ps.setInt(1, userPK);

    } catch(SQLException ex){
        Logger.getLogger(clockin.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

I pass in userPK again which holds the user primary key.

I set ? in the where clause to userPK using a setInt() method so the correct record is updated.

I am going to test the query by executing the query in MYSQL as seen below:

```
1 • UPDATE dailyhours SET hours = TIMESTAMPDIFF(HOURS, clockin, clockout) WHERE user_fk = 8
```

However, I got this error:

So, I changed the query to:

```
1 • update dailyhours set hours = round(TIMESTAMPDIFF(SECOND, clockin, clockout)/3600, 2) where user_fk = 8
```

Here I am retrieving the time in seconds between the clockin and clockout datetimes and then dividing it by 3600 to convert it to hours and then using the round() function to round it to two decimal places. The method worked:

dailyhour_pk	clockin	clockout	hours	user_fk
10	2023-01-05 18:24:11	2023-01-07 11:07:34	40.72	8

Now I need to update the calculate hours method as seen below:

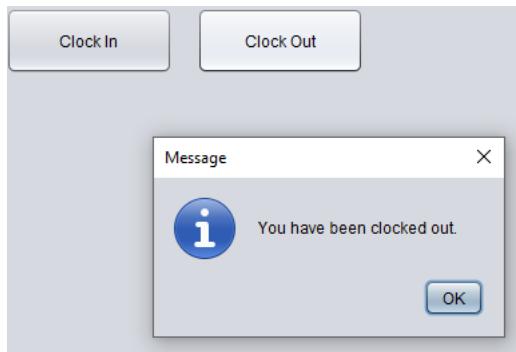
Development and Testing

```
public static void calculateUserHours(int userPK){  
  
    Connection connection = database.getConnection();  
    PreparedStatement ps;  
  
    try{  
  
        ps = connection.prepareStatement("update dailyhours set hours = round(TIMESTAMPDIFF(SECOND, clockin, clockout)/3600, 2) where  
        ps.setInt(1, userPK);  
  
    }  
    catch(SQLException ex){  
        Logger.getLogger(clockin.class.getName()).log(Level.SEVERE, null, ex);  
    }  
}  
  
user_fk = ?");
```

Now I need to call the method under the clockout button action listener

```
private void clockoutButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    classes.clockin.clockUserOut(userPrimaryKey);  
    classes.clockin.calculateUserHours(userPrimaryKey);  
    this.dispose();  
}
```

After testing these are the results:



dailyhour_pk	clockin	clockout	hours	user_fk
18	2023-01-07 12:49:24	2023-01-07 13:09:06	0.33	8

This method worked.

4.11.7 Test Plan

Test Number	How To Test	Expected Result	Actual Result
1	Attempt to clock in	New clock in record should be created with the users primary key and the date time they clocked in	Clock in record was created with correct details
2	Attempt to clock out	The clockout field for the users clockin record should be updated with the current datetime	Clockout field was updated appropriately

3	Clock in and wait a few minutes then clock out	The difference in time in hours between the clockin and clockout field should be calculated and inserted into the hours field	Time in hours was calculated and inserted into the hours field
4	Clock in and out again	A new clockin record should be created for that user with the correct clockin, clockout and hours fields, the users total hours for that month should be updated with the new hours added	New record was created correctly and the total hours were displayed to the user correctly.

4.11.8 Evidence

4.11.8.1 Test 1

	dailyhour_pk	clockin	clockout	hours	user_fk
▶	30	2023-01-24 09:29:59	NULL	0.00	1

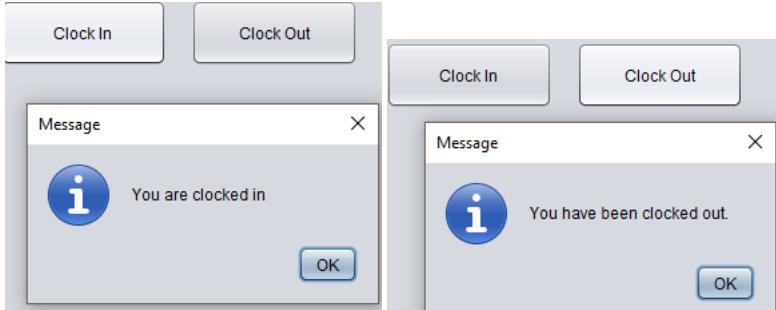
4.11.8.2 Test 2

	dailyhour_pk	clockin	clockout
30	2023-01-24 09:29:59	2023-01-24 09:33:06	

4.11.8.3 Test 3

ID	Name	Hours	Year-Month
1	theoplank	0.05	2023-January

4.11.8.4 Test 4



ID	Name	Hours	Year-Month
1	theoplank	0.24	2023-January

4.11.9 Success Criteria

Success Criteria Number	Success Criteria	Achieved	Notes
9	Clock in/out function (calculate hours between clock in and out)	Yes	None

4.11.10 Review/Stage Evaluation

In this stage I developed the functionality behind the clock in section. This means allowing the user to enter their username and password and validating their inputs, giving them the choice of clocking in or out and updating the database appropriately, and lastly calculating the new user hours when they clock out. This stage was quick to develop as I didn't have too many major errors or problems with the code. Also, I was able to fully achieve the success criteria for this stage.

First, I was able to re-use the method from the login section for validating the inputted username and password which saved time. Once that was done and tested, I developed the clock in method.

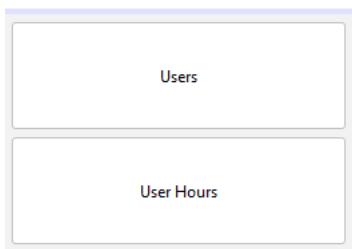
This method takes the primary key of the user that entered the username and password and inserts it along with the current date time into the dailyhours table so a new record is created.

Next, I created the clock out method which again takes the user primary key and updates the record with the user foreign key field matching the user primary key with the current date time.

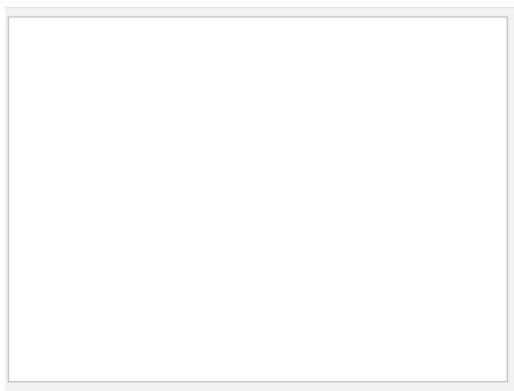
Lastly, I developed the calculate hours method which when called inserts into the hours field of the dailyhorus table the difference in hours to two decimal places between the clockin and clockout datetimes and called this method after I called the clockout method under the clockout action listener method.

4.12 Stage 11/Admin

This section will allow the admin user to view, add, delete and update user information and will also allow the admin to view the number of hours the users have worked per month.



This is the updated adminScreen jframe it has two buttons with action listeners, one opens the users table and one opens the user hours table.



This is the userhours jframe, it only has one component which is a jTable.

4.12.1 Getter/Setter methods for userHours class + global variables

```
public class userHours {  
  
    private int user_pk;  
    private String user_name;  
    private float hours;  
    private String yearMonth;
```

```

public Integer getUserPK(){
    return user_pk;
}

public String getUserName(){
    return user_name;
}

public float getHours(){
    return hours;
}

public String getYearMonth(){
    return yearMonth;
}

public void setUserPK(int userPK){
    this.user_pk = userPK;
}

public void setUserName(String name){
    this.user_name = name;
}

public void setHours(float hours){
    this.hours = hours;
}

public void setYearMonth(String yearMonth){
    this.yearMonth = yearMonth;
}

```

4.12.2 userHours Constructor

```

public userHours(int userPK, String name, float hours, String yearMonth){

    this.user_pk = userPK;
    this.user_name = name;
    this.hours = hours;
    this.yearMonth = yearMonth;

}

```

This constructor method will be used to create objects of the userHours class with values from the database. The objects will be used to fill an ArrayList which will be used to fill a jTable with data.

4.12.3 Developing user hours retrieval Method

This method will create objects of the userHours class with values from the database and fill an arrayList with these objects.

First, I developed this query in sql and tested it in the database:

```

1 • select user_pk, username, date_format(date_add(clockin,interval -DAY(clockin)+1 DAY),"%%Y-%%M") month, SUM(hours) hours from users,dailyhours WHERE users.
WHERE users.user_pk = dailyhours.user_fk GROUP BY date_format(date_add(clockin,interval -DAY(clockin)+1 DAY),"%%Y-%%M") ORDER BY date_format(date_add(clo
ORDER BY date_format(date_add(clockin,interval -DAY(clockin)+1 DAY),"%%Y-%%M"))

```

Which retrieved this data when I executed it:

	user_pk	username	month	hours
▶	8	theoplank	2022-February	0.00
	8	theoplank	2023-January	4.00

I then copy and pasted that query into the String query variable in the method

Development and Testing

```
public ArrayList<userHours> userHoursList() {
    ArrayList<userHours> userHourList = new ArrayList<>();
    Connection connection = database.getConnection();
    PreparedStatement ps;
    ResultSet rs;

    String query = "SELECT user_pk, username, date_format(date_add(clockin, interval -DAY(clockin)+1 DAY), \"%Y-%M\") month, SUM(hours) hours FROM user_hours GROUP BY month";
    try{
        ps = connection.prepareStatement(query);
        rs = ps.executeQuery();
        userHours user;
        while (rs.next()){
            user = new userHours(rs.getInt("user_pk"), rs.getString("username"), rs.getFloat("hours"), rs.getString("month"));
            userHourList.add(user);
        }
    } catch (SQLException ex){
        Logger.getLogger(admin.class.getName()).log(Level.SEVERE, null, ex);
    }
    return userHourList;
}
```

This method creates a new ArrayList that can hold objects of the userHours class.

Then I query the database for the user primary key, user name, date as a year and month and the hours summed up for each month.

Then I use a while loop and the next() method for the result set to create objects of the userHours class that take the values from the database as parameters.

Lastly, I return userHourList.

4.12.4 Developing Fill user hours table method

Now that I have retrieved the data from the database I need to create the method to fill the jTable with the objects in the arrayList.

```
public void fillOrderTable(){
    classes.userHours user = new classes.userHours();
    ArrayList <classes.userHours> userList = user.userHoursList();
    String [] columnNames = {"ID", "Name", "Hours", "Year-Month"};
    Object [][] rows = new Object[userList.size()][4];

    for(int i = 0; i<userList.size(); i++){
        rows[i][0] = userList.get(i).getUserPK();
        rows[i][1] = userList.get(i).getUserName();
        rows[i][2] = userList.get(i).getHours();
        rows[i][3] = userList.get(i).getYearMonth();
    }

    classes.table model = new classes.table(rows, columnNames);
    userHoursTable.setModel(model);
}
```

4.12.5 Developing set column names method

```
public void setColumnNames(){
    userHoursTable.getColumnModel().getColumn(0).setHeaderValue("ID");
    userHoursTable.getColumnModel().getColumn(1).setHeaderValue("Name");
    userHoursTable.getColumnModel().getColumn(2).setHeaderValue("Hours");
    userHoursTable.getColumnModel().getColumn(3).setHeaderValue("Year-Month");
}
```

These methods worked as you can see below:

ID	Name	Hours	Year-Month
8	theoplank	0.0	2022-February
8	theoplank	4.0	2023-January

4.12.6 Getter/Setter Method + Global Variable Declaration + Constructor for Admin class

```
public class admin {
    Connection connection;

    private int user_pk;
    private String username;
    private String password;
    public String type;

    public String getUserName(){
        return username;
    }

    public String getPassword(){
        return password;
    }

    public String getUserType(){
        return type;
    }

    public void setUserPK(int user_PK){
        this.user_pk = user_PK;
    }

    public void setName(String name){
        this.username = name;
    }

    public void setPassword(String Password){
        this.password = Password;
    }

    public void setType(String Type){
        this.type = Type;
    }

    public admin(int user_PK, String name, String Password, String Type)
    {
        this.user_pk = user_PK;
        this.username = name;
        this.password = Password;
        this.type = Type;
    }
}
```

4.12.7 Developing Create User Method

Now I will create the method to allow the user to create new users with a username and a password. This method will be similar to the other create methods I have developed.

Development and Testing

```
public static void addUser(admin user){
//adds a new user to the users table in the database using inputs from the user

    Connection connect = database.getConnection();
    PreparedStatement ps = null;
    try{
        ps = connect.prepareStatement("INSERT INTO users(username, password, type) VALUES(?, ?, ?)");
        ps.setString(1, user.getUserName());
        ps.setString(2, user.getPassword());
        ps.setString(3, user.getUserType());

        if(ps.executeUpdate() != 0){
            JOptionPane.showMessageDialog(null, "New User Added");
        }
        else{
            JOptionPane.showMessageDialog(null, "Error");
        }
    }
    catch (SQLException ex) {
        Logger.getLogger(admin.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

This method simply uses an insert statement to insert into the username, password and type fields of the users table.

I have used ? placeholders to set the values of each field to the returned value of the corresponding getter methods.

A success or error message is displayed depending on if 0 records or more were affected.

4.12.8 Add User Button Code

This code is under the action listener method for the add button which is a component on the newUserScreen jform which is called from pressing the add user button on the user jform.

```
private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //creates a new user using user inputs from text fields and combo box

    classes.admin user;

    String username = nameTextField.getText();
    String password = passwordTextField.getText();
    String type = typeComboBox.getSelectedItem().toString();

    user = new classes.admin(1, username, password, type);

    classes.admin.addUser(user);
    this.dispose();
}
```

This method is simple

I set variables to the values of each text field and combo box

Then create a new instance of admin with these values and call the addUser() method and pass in the new object.

4.12.9 Developing fill user jTable Method

This method will be very similar to other fill jTable methods, it will simply retrieve all the data from the database that is similar to the search value and create objects of the admin class with this data and fill an array list with these objects.

```
public ArrayList<admin> userList (String searchVal) {
    //creates an arraylist
    //creates a new instance of "admin" (as shown in method earlier) with data from the users table in the database
    //adds the new user to the arrayList

    ArrayList<admin> userList = new ArrayList<>();

    connection = database.getConnection();
    PreparedStatement ps;
    ResultSet rs;

    String query = "SELECT * FROM users WHERE username LIKE ?";

    try{
        ps = connection.prepareStatement(query);
        ps.setString(1, "%" + searchVal + "%");
        rs = ps.executeQuery();
        admin user;
        while (rs.next()){
            user = new admin(rs.getInt("user_pk"), rs.getString("username"), rs.getString("password"), rs.getString("name"), rs.getString("type"));
            userList.add(user);
        }
    }
    catch (SQLException ex){
        Logger.getLogger(admin.class.getName()).log(Level.SEVERE, null, ex);
    }
    return userList;
}
```

This method takes a string parameter which is the users search value.

I create a new array list of type admin so it can hold instances of the admin class where each instance will be a user.

The query selects all the data from the users table where the username is similar to the user inputted search value.

I have used a while loop to iterate through each record in the result set and create new instances of the admin class taking the values in each record as parameters. Then each object is added to the array list.

After testing these methods worked as seen below:

Search: <input type="text"/>				<input type="button" value="SEARCH"/>
ID	Name	Password	Type	
1	theoplank	theoplank	Admin	

I have created a user and it has been displayed so both methods work.

4.12.10 Developing Delete User Method

This method will simply use a delete query to delete the selected user.

```

public static void deleteUser(int userID){
    //deletes the selected user

    Connection connection = database.getConnection();
    PreparedStatement ps;

    try{
        ps = connection.prepareStatement("DELETE FROM users WHERE user_pk = ?"); //AND ?
        ps.setInt(1, userID);
        //ps.setInt(2, deletedflag);
        if (ps.executeUpdate() != 0){
            JOptionPane.showMessageDialog(null, "User has been deleted.");
        }
        else{
            JOptionPane.showMessageDialog(null, "There was an error, please try again.");
        }
    }
    catch(SQLException ex){
        Logger.getLogger(admin.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

I have passed in userID which will hold the value of the primary key for the selected user.

Then I query the database to delete all the records in the users table where the user_pk field value is equal to the value of userID passed in.

Then I use an if statement to output appropriate success or error messages.

This method issues a hard delete meaning that it completely deletes the selected record from the database. This will be a problem as users will be related to other tables in the database such as the orders table to show which user created the order, this means that there will be a foreign key constraint error when I try to delete a user that is referenced in another table in the database.

After testing I got this error:

```
Cannot delete or update a parent row: a foreign key constraint fails
```

I could and have fixed this in the past by implementing a soft delete using a deleted flag and by updating the data retrieval method to only retrieve records with the deleted flag set to false and when the delete method is called instead of deleting the record it instead updates the deleted flag field to true. However, I don't have time to do this as I need to finish the rest of the project quickly so I will leave it as is and know that I can and will fix it if I have time.

4.12.11 Developing Update User Method

This method again will be very similar to the other update methods and will simply set the field values for the selected user to the new user inputted values.

```

public static void updateUser(admin user){
//updates a user using user inputs

    Connection connection = database.getConnection();
    PreparedStatement ps;

    try{
        ps = connection.prepareStatement("UPDATE users SET username = ?, password = ?, type = ? WHERE user_pk = ?");
        ps.setString(1, user.getUserName());
        ps.setString(2, user.getPassword());
        ps.setString(3, user.getUserType());
        ps.setInt(4, user.getUserPK());

        if(ps.executeUpdate() != 0){
            JOptionPane.showMessageDialog(null, "User updated.");
        }
        else{
            JOptionPane.showMessageDialog(null, "There was an error, please try again.");
        }
    }
    catch(SQLException ex){
        Logger.getLogger(admin.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

I have passed in the instance user of the admin class.

The update statement sets each field to the placeholder ? where the user_pk field is equal to ?

I have set each placeholder to the correct value using the getter methods on the object passed in

I have used an if statement to output success or error messages.

4.12.12 Update User Button Code

This code is under the action listener method for the update button on the updateUserScreen jform which is called from pressing the edit user button on the user jform.

```

private void updateButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //updates selected user with user inputs

    classes.admin user;

    String username = nameTextField.getText();
    String password = passwordTextField.getText();
    String type = typeComboBox.getSelectedItem().toString();

    user = new classes.admin(l, username, password, type);

    classes.admin.updateUser(user);
    this.dispose();
}

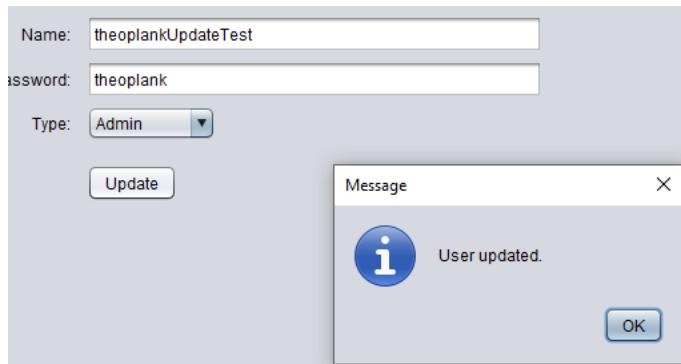
```

I set variables to the values of each text field and combo box

Then create a new instance of admin with these values and call the addUser() method and pass in the new object.

After testing this method worked as seen below:

Development and Testing



A screenshot of a Java Swing application window titled "User Update". It contains three text fields: "Name" (theoplankUpdateTest), "Password" (theoplank), and "Type" (Admin). Below the fields is an "Update" button. To the right, a modal dialog box titled "Message" displays an information icon and the text "User updated." with an "OK" button.

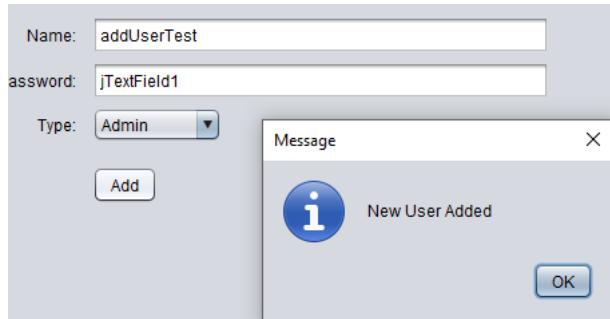
ID	Name	Password	Type
1	theoplankUpdateTest	theoplank	Admin

4.12.13 Test Plan

Test Number	How To Test	Expected Result	Actual Result
1	Attempt to add a new user	New user should be created	New User was created
2	Attempt to update an old user	User should be updated	User was updated
3	Attempt to delete a user	User should be deleted from table (however It won't work as I haven't implemented the soft delete)	Foreign key constraint error as expected

4.12.14 Evidence

4.12.14.1 Test 1



A screenshot of a Java Swing application window titled "User Add". It contains three text fields: "Name" (addUserTest), "Password" (jTextField1), and "Type" (Admin). Below the fields is an "Add" button. To the right, a modal dialog box titled "Message" displays an information icon and the text "New User Added" with an "OK" button.

ID	Name	Password	Type
1	theoplank	theoplank	Admin
11	addUserTest	jTextField1	Admin

4.12.14.2 Test 2

ID	Name	Password	Type
1	updateUserTest	jTextField1	Admin

4.12.14.3 Test 3

Cannot delete or update a parent row: a foreign key constraint fails

4.12.15 Success Criteria

Success Criteria Number	Success Criteria	Achieved	Notes
*	No specific success criteria	Yes	There was no specific success critieria for this feature however in order to achieve success criteria 11 which states that access should be restricted depending on user type, users must be able to create new users of different types.
15	Calculate Employee Wages	Partly	This section displays the users hours but doesn't calculate their wages, the manager would have to do that themselves.

4.12.16 Review/Stage Evaluation

In this stage I created functionality behind two jtables.

One shows all the users and allows the user to create and update users. It would also allow the user to delete users however I didn't have time to implement the soft delete as previously stated.

Development and Testing

The other shows the user's hours for each month of each year that they have worked. This stage was very simple and quick as it required a lot of cut and pasting of methods with minor adjustments. Although this stage didn't have any specific success criteria, it should have as it is necessary for user of other features in the system.

5 Evaluation

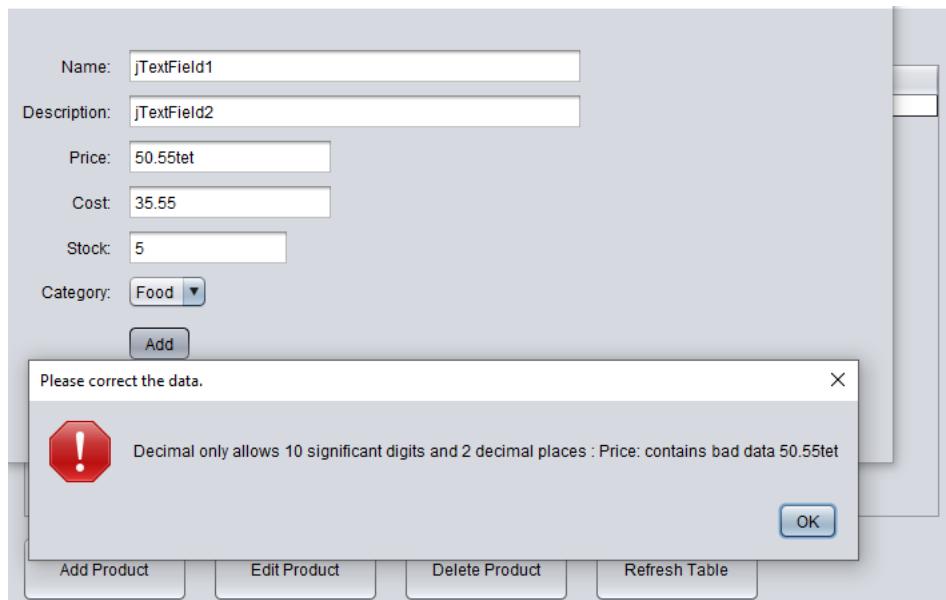
5.1 Testing For Evaluation

5.1.1 Robustness Testing

For my robustness testing I will be entering invalid/bad data for all the sections that take a text user input. These sections will be: the products section, categories section, admin section, pos section, login, reports section and the clock in section. I won't be robustness testing the search bars as in my system the query uses the LIKE function to retrieve data matching the search value, so any type of data input is fine.

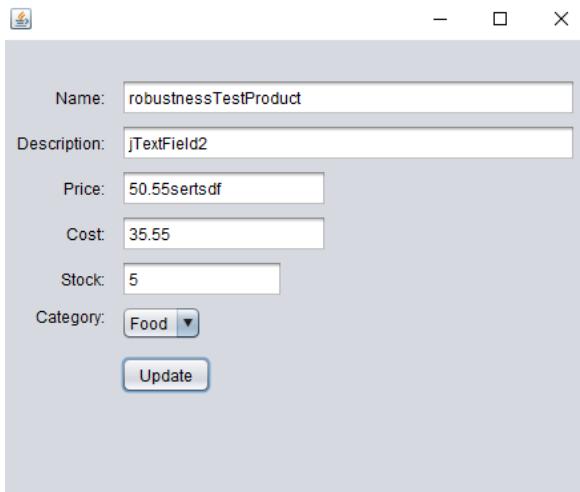
I didn't have time to finish the input data validation for all the sections so currently the product section is validated properly but the other sections aren't. This is bad but the other sections such as the categories section and admin section allow the user to input the name for categories and username, password and type for the admin section; only the length validation is a problem for these sections as name, username and password can include letters, numbers and symbols because the fields are of data type varchar, also for the user type it is a combo box with predefined options so the user cannot input bad data there. However, in places like the discount field for the pos section or the date fields for the reports section there is no validation as I didn't have time so the user can enter anything. This isn't a problem for the reports section as the data inputted won't be stored in the database so there will be no error it just won't work, but for the discount field there will be an error.

5.1.1.1 Product Section



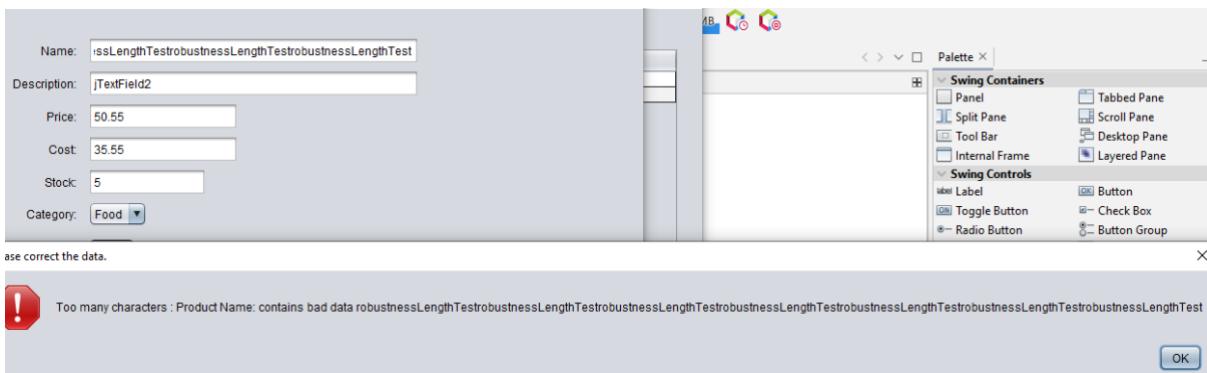
Here you can see that when I try to enter bad data for the price field, an error message pops up and the product is not created.

Evaluation



Exception in thread "AWT-EventQueue-0" java.lang.NumberFormatException: For input string: "50.55sertsdf"

Here you can see that although no message pops up when I try to update the product with bad data, an error is returned in the output box and the product is not updated and the system still works.



Here I try to create a product with too many characters and an error message pops up and the product is not created

5.1.1.2 Categories Section

For the categories section I will only do a length test as the user can only input the category name which can include letters, numbers and symbols as the name is stored as varchar in the database.

com.mysql.cj.jdbc.exceptions.MysqlDataTruncation: Data truncation: Data too long for column 'category name' at row 1

When I try to create a category with a name that has too many characters for the field, this error is output however the input isn't caught as didn't have time to validate this section.

5.1.1.3 Admin Section

For the admin section, specifically the users part of the admin section where new users can be created; I will perform the robustness testing only by inputting data that is too long for the field as there is no need to input bad data with the wrong datatype as the field it is stored in is of datatype varchar.

Evaluation

com.mysql.cj.jdbc.exceptions.MySQLDataTruncation: Data truncation: Data too long for column 'username' at row 1

This error was output when I tried to enter a username with too many characters, this is because I didn't have time to validate this section.

5.1.1.4 POS Section

In this section I will test the discount field, custom price field and quantity field. For the custom price field and the discount field all they do is change the value of the text for the final total, when the order is confirmed the price field will be the value of the price for each product regardless of the custom price field input or the discount input as I didn't have time to code the methods to set the order price to the value of these inputs. This means that for the custom price field the user could enter anything and there will be no error except if they enter something too long the label may push the components to its right off the window. However, for the discount field if the user enters letters for example, then the system will try to take the final total value and subtract letters from it, this will cause an error.

ORDER			
ID	Name	Quantity	Price
28	rice	1	2.5

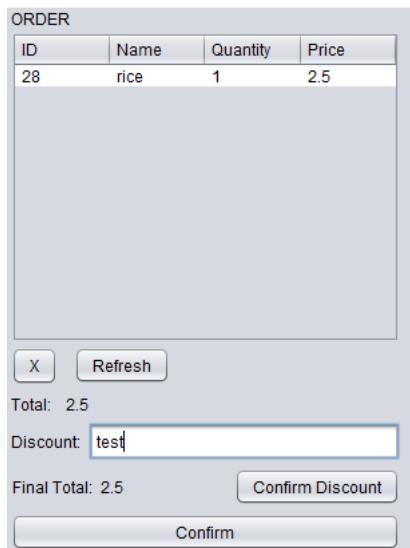
X Refresh

Total: 2.5

Discount:

Final Total: 2.5 Confirm Discount

Confirm



Exception in thread "AWT-EventQueue-0" java.lang.NumberFormatException: For input string: "test"

Here an error is caused as the program tries to subtract "test" from 2.5 which doesn't work because you cannot subtract a string from a float.

Total: 2.5

Set Custom Price:

Confirm Price

Discount:

Final Total: test Confirm Discount

Confirm



Here I have entered test into the custom price field and the value of the final total label has been set to test. When I confirm the order there is no error so this doesn't break the system but it is wrong.

Evaluation

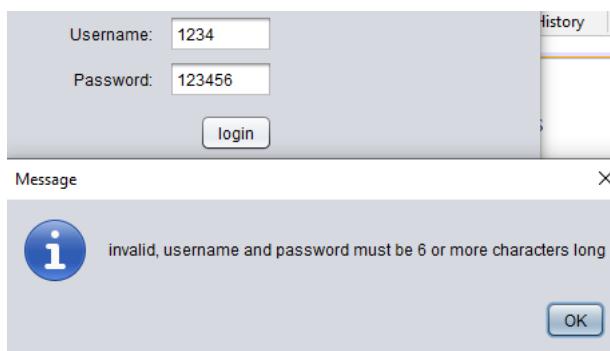
+ Refresh Category: Food Confirm Quantity: test

Exception in thread "AWT-EventQueue-0" java.lang.NumberFormatException: For input string: "test"

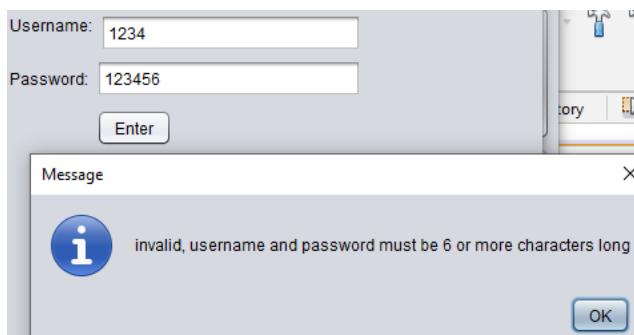
When I attempt to add a product to the order with quantity "test", the error above is output. No error message is shown to the user so this is bad as they might be unaware as to why the product isn't being added to the order.

5.1.1.5 Login And Clock In Section

For the login section it is simply taking the user input and comparing it to the data in the users table in the database to check if it matches any of the users username and password. This means that any symbol, letter or number can be entered however I have a minimum of 6 characters set for the username so it should output an error message if I try to enter a username or password under 6 characters. For the clock in section I used the same code so the result will be the same.



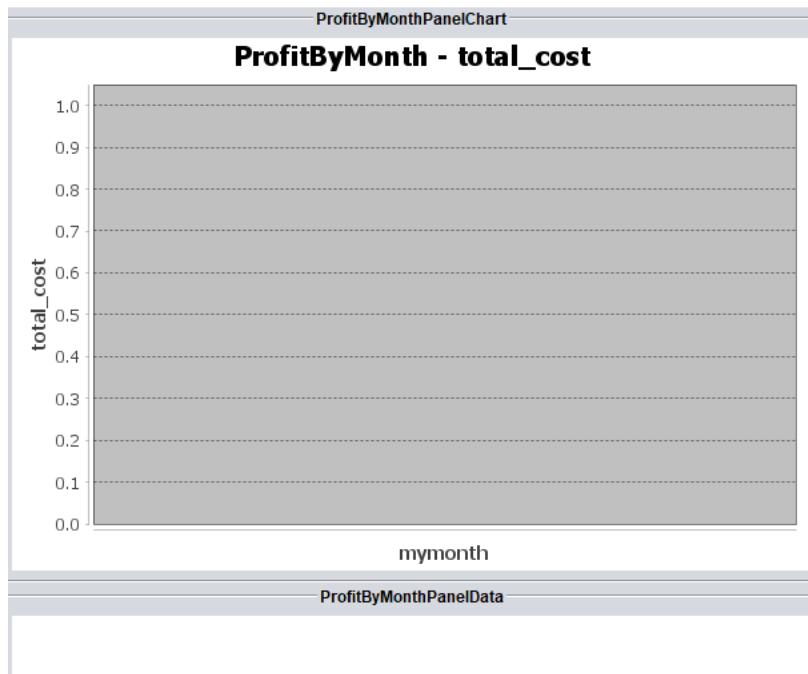
Here, I tried to enter 4 characters for the username which isn't allowed and an error message was properly output.



Here in the clock in section I enter a username that is too short, and the appropriate message is output.

5.1.1.6 Reports Section

In this section I will try entering bad data for the month fields however it shouldn't break the system it will just produce an empty report.



Here I tried to enter “test” for the month fields when the data input should be a date in the form YYYY-MM and an empty report was built.

5.1.2 User Testing

For this section I will carry out black box testing on potential users to gather feedback on the functionality and usability of my system. I will run the software and let them take control to navigate the POS system as they wish. I have created a questionnaire for the users to fill out after they have tested the system.

5.1.2.1 *The questions I will ask are:*

- How clear was it to navigate between screens?
- Are the user interfaces clear and intuitive?
- Is all the text readable?
- Are all the buttons of a suitable size and colour?
- Can you suggest any improvements with any of the screens?
- Did you come across any bugs or issues with the system?
- Are there any additional features that could be added to improve the system?
- Any other comments on the system?

5.1.2.2 *Answers:*

User 1:

1-10/10

2-Yes

3-Yes

4-Yes

5-Explain what POS means

6- Discount doesn't work

7-Summing price of orders in orders section, hide password when logging in

8-Overall good, make cost and price meaning clearer, there should always be an admin user (the admin shouldn't be able to remove themselves as admin)

User 2:

1-10/10

2-Yes

3-Yes

4-Yes

5- In the order section , ABC

6- POS section, discount value doesn't save

7-Nope

8- All in all, the program was smooth . I feel like everything is straight forward and very clear to understand . The one thing I would change is actually saving the discount price for the future .

User 3:

1-10/10 it was easy to navigate to different screens

2-The sales report section could make it clearer which month you are selecting.

3-Yes

4-Yes

5-Change POS to sell to make it more clear

6-Deletes all of the same product if there are multiple instances of it in the order

7-No

8-Bar chart should have same colour bars as it's the same value being compared

5.1.2.3 Review

The users all found it very easy to use my system so the overall usability of the system is good due to various usability features. Multiple users said that they didn't know what POS meant and it was suggested that I change the name of this section to "sell". Two bugs were found, the discount feature doesn't work, and if multiple instances of a product are added to an order and one instance is deleted all the instances are deleted. It was also suggested that I sum up the price and cost of orders and display this to the user, and that I hide the password when logging in. Also, one user found it difficult to understand the charts due to having different colour bars and another user

Evaluation

pointed out that there always needs to be an admin user otherwise no one will be able to access the admin section to create new users, so the admin shouldn't be able to change their own user type unless there is another admin user.

5.2 Overall Evaluation

5.2.1 Success Criteria

Number	Criteria	Achieved	Notes
	Must		
1	Track stock (automatically decrease stock as per purchase)	Yes	None
2	Form orders (calculate price, add meals or products to an order)	Yes	None
3	Allow user to add an item with custom name, picture, price, multiple products	Yes / Partially	The user can create products with a name, description, price, cost, and category. However, they can't have an item that includes multiple products or a picture as I deemed these weren't a high priority so instead focused on finishing the system.
4	Have a Keypad	No	My system doesn't have a keypad, but it does have a custom price field that can be used to set the price for an order. However I didn't have time to code the method to set the order price to the value in the

Evaluation

			custom price field so it is useless
5	Allow user to add items to categories	Yes	None
6	Track orders	Yes	None
7	Create and display Sales Reports/analytics	Yes	None
8	Notify the user when there is low stock for a product or multiple products (with a customizable low stock level which the user can change from product to product) and have a negative stock alert	No	Decided not to do this feature as I was running out of time, and this is a somewhat minor feature of the system
9	Clock in/out function (calculate hours between clock in and out)	Yes	None
10	Card/cash payment	No	Ran out of time – would have been good to have a pop up from the pos screen to allow the user to enter in the amount payed although I never intended to implement card reader usability or store customer details so not completely relevant
11	User Specific Menu Screens (users will have different user types which determine what functions they have access to)	Yes / Partially	Instead of having different menu screens I have one screen that validates the access of the logged in user when they try to access a feature
12	Login System	Yes	None
	Could		
13	Barcode Usability (so the user can make a product with a specific barcode, then they can scan this barcode with their camera and will be given options either to add the product	No	None

Evaluation

	(to the order or edit/alter the product)		
14	AI that predicts various things such as how busy you will be, what your most sold product will be on a certain day based off of weather, if it's a national holiday, what day of the week it is etc	No	None
15	Calculate employees' wages based off of hours worked in the week/month and hourly wage	No	None
16	Have a filter/sort selection function for the product list/order list	No	I don't have a filter feature for the order or product section but I do have a filter by category box for the POS section
17	Have a search bar for the orders list/products list, this will allow the user to search by order name, item name, order cost, item cost.	Yes / Partially	My search bar feature allows the user to search by product name or order id
18	Customizable UI	No	None
19	Automated discount system	No	None
20	Receipt system	No	None
21	Customizable colour theme	No	None
22	Discount creation	No	None
23	Time out function	Yes	None

Evaluation

5.2.2 Success Criteria Evidence

5.2.2.1 1 – Track Stock

PRODUCTS							ORDER			
ID	Name	Descrip...	Price	Cost	Category	Stock	ID	Name	Quantity	Price
28	rice	jasmine...	2.5	0.75	Food	10	28	rice	5	12.5

+ Refresh Category: Food Confirm Quantity: 5

X Refresh Total: 12.5

Set Custom Price: Discount:

Confirm Price Final Total: 12.5 Confirm Discount

Confirm

PRODUCTS						
ID	Name	Descrip...	Price	Cost	Category	Stock
28	rice	jasmine...	2.5	0.75	Food	5

I added 5 rice to the order and originally rice had a stock level of 10, after confirming the order the stock level was at 5. This shows that my system decrements stock according to the quantity of the product in the order, so this success criteria is achieved.

5.2.2.2 2-Form Orders

PRODUCTS							ORDER			
ID	Name	Descrip...	Price	Cost	Category	Stock	ID	Name	Quantity	Price
28	rice	jasmine...	2.5	0.75	Food	5	28	rice	1	2.5
33	chicken	sweet a...	15.99	3.5	Food	5	33	chicken	1	15.99

+ Refresh Category: Food Confirm Quantity: 1

X Refresh Total: 18.49

Set Custom Price: Discount:

Confirm Price Final Total: 18.49 Confirm Discount

Confirm

This shows that my system can form orders correctly as when I add multiple products to the order they are both shown in the order and the correct total and final total are calculated. Therefore this success criteria is achieved.

Evaluation

5.2.2.3 3-Allow User To Add an Item With Custom Name, Picture, Price, Multiple Products + 5-Allow User To Add Items To Categories

The screenshot shows a product addition form and a message dialog. The form fields are: Name: product1, Description: first product, Price: 12.50, Cost: 3.50, Stock: 5, Category: Food. An 'Add' button is present. A message dialog titled 'Message' displays 'New Product Added' with an information icon. Below is a table of products:

ID	Name	Description	Price	Cost	Category	Stock
28	rice	jasmine rice	2.5	0.75	Food	5
33	chicken	sweet and sour	15.99	3.5	Food	5
34	product1	first product	12.5	3.5	Food	5

This shows that I can create a new product with its own name, description, price, cost, stock and category. Success criteria 5 has been fully achieved and 3 has been partially achieved as the user can add products to categories, however although the user can't add a product picture or add multiple products to an item as it didn't seem necessary and I didn't have enough time.

5.2.2.4 6-Track Orders

The screenshot shows an order tracking interface. At the top is a search bar and a 'SEARCH' button. Below is a table of orders:

ID	User-ID	Date
79	1	2023-01-22
80	1	2023-01-22
81	1	2023-01-22
82	1	2023-01-22
83	1	2023-02-15
84	1	2023-02-19
85	1	2023-02-19
86	1	2023-02-19
87	1	2023-02-19
88	1	2023-02-22
89	1	2023-02-23
90	1	2023-02-23
91	1	2023-02-23
92	1	2023-02-23
93	1	2023-02-26
94	1	2023-02-26

A 'View Order' button is on the left. A detailed view window is open at the bottom right, showing the contents of order ID 94:

ID	Name	Price	Cost	Quantity	Profit
28	rice	2.5	0.75	1	1.75
33	chicken	15.99	3.5	1	12.49

I have selected the latest order and pressed the view order button which called the frame in the bottom right of the screenshot. This window shows that my system tracks the orders and allows the user to select an order to view its contents. Therefore this success criteria has been achieved.

Evaluation

5.2.2.5 7-Create And Display Sales Reports/Analytics

Sales Reports

Calculate Profit Report:

Start Month: End Month: Value: Order by:

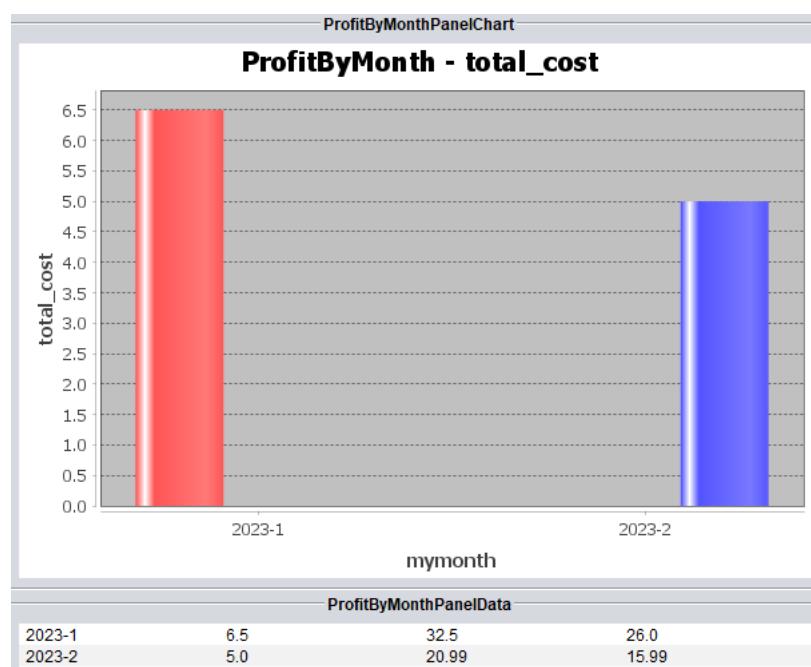
Most Sold Report:

Start Month: End Month: Value: Order by:

Total Orders Report

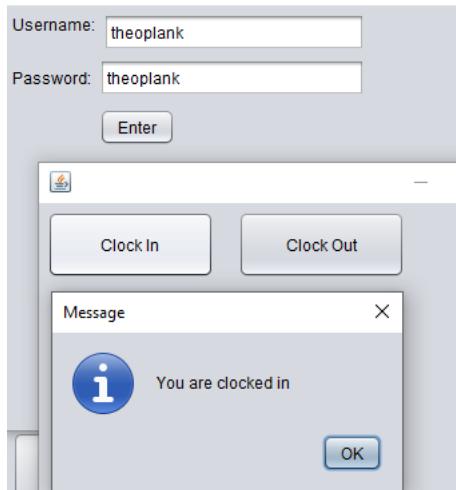
Start Month: End Month: Value: Order by:

This screen allows the user to enter the months for the report and select what they want the report to show and how it should be ordered.

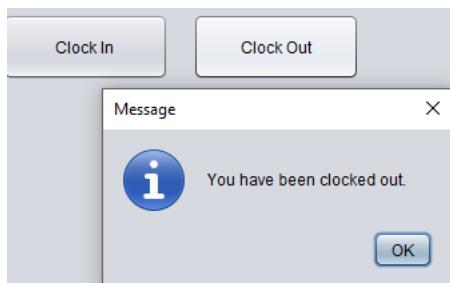


This report is built and displayed when I press the run button for the calculate profit report . This shows that this success criteria has been achieved.

5.2.2.6 9-Clock In and Out Function



This shows that the system allows you to clock in



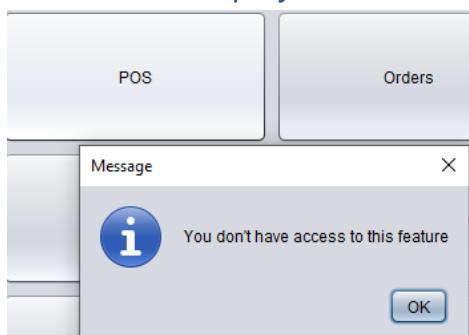
This shows that you can clock out

ID	Name	Hours	Year-Month
1	theoplank	0.0	2023-February
1	theoplank	0.24	2023-January

This shows that the hours for each user for each month is calculated and displayed to the user.

Therefore, this success criteria has been achieved.

5.2.2.7 11-User Specific Menu Screens

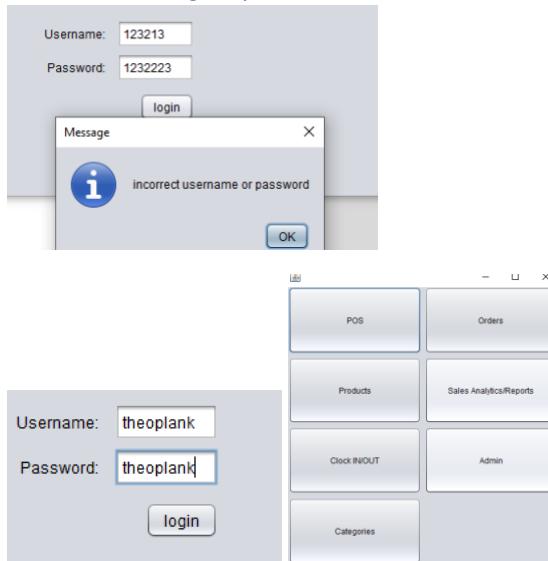


This shows that when I try to access the pos section of my system when I am logged in as a kitchen user type, it doesn't let me access it. Although I don't have user specific menu screens this system means I only need one menu screen that denies access to features the same way having those features hidden from the menu screen would. Therefore functionally wise this success criteria is fully

Evaluation

achieved however I have only partially achieved it by name as my system doesn't have different menu screens.

5.2.2.8 12-Login System



This shows that when I enter the wrong username and/or password my login system doesn't allow me into the menu however when I enter a valid and correct username and password for an existing user the menu screen opens. Therefore this success criteria is fully achieved.

5.2.2.9 17-Have a search bar for the orders/products list, this will allow the user to search by order/product name and order/product cost

ID	Name	Description	Price	Cost	Category	Stock
28	rice	jasmine rice	2.5	0.75	Food	5
33	chicken	sweet and sour	15.99	3.5	Food	5
34	product1	first product	12.5	3.5	Food	5

Search: rice						
ID	Name	Description	Price	Cost	Category	Stock
28	rice	jasmine rice	2.5	0.75	Food	5

This shows that my system allows you to search by product name however it doesn't let you search by product cost as I didn't think that would be very useful.

Evaluation

The screenshot shows two separate search results tables. The top table has columns 'ID', 'User-ID', and 'Date'. The bottom table has columns 'A', 'B', and 'C'.

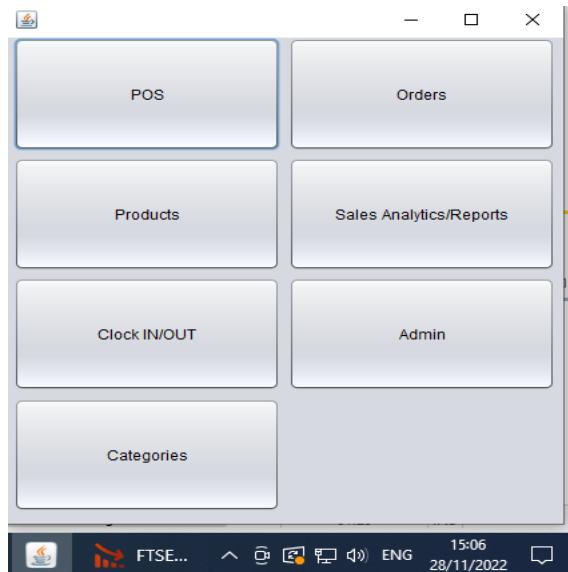
ID	User-ID	Date
68	1	2023-01-07
69	1	2023-01-13
70	1	2023-01-18
71	1	2023-01-18
72	1	2023-01-18
73	1	2023-01-18
74	1	2023-01-18
75	1	2023-01-19
76	1	2023-01-22
77	1	2023-01-22
78	1	2023-01-22
79	1	2023-01-22
80	1	2023-01-22
81	1	2023-01-22
82	1	2023-01-22
83	1	2023-02-15

A	B	C
70	1	2023-01-18

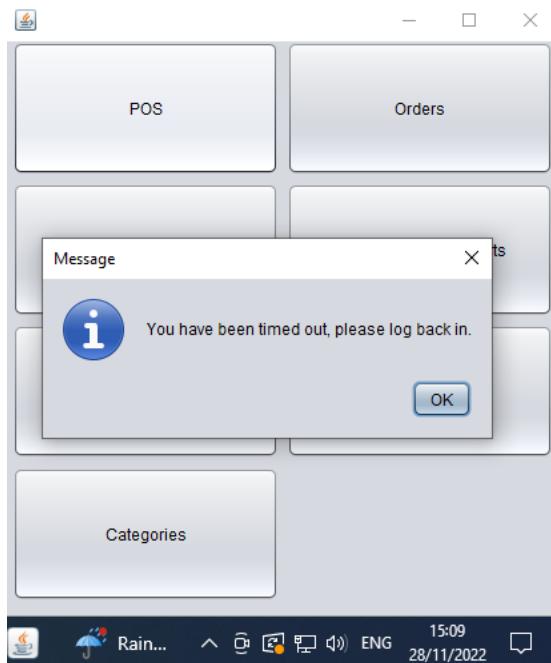
This shows that you can search by order ID which is essentially the order name however you can't search by order cost as I didn't think it was necessary.

Therefore, this success criteria is partially achieved.

5.2.2.10 23-Time Out Function



Evaluation

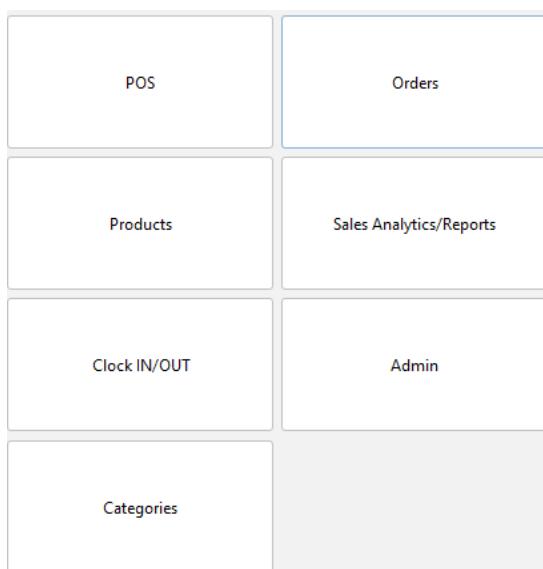


During this example I lowered the time out limit from 10 minutes to 1 minute. This shows that when I tried to access the system after being idle for over 1 minute I was logged out and had to log back in. Therefore, this success criteria was fully achieved.

5.2.3 Usability Features

5.2.3.1 Large and Well Labelled Buttons

In my system all the buttons are large and well labelled to make it easier to navigate and make use of the system.



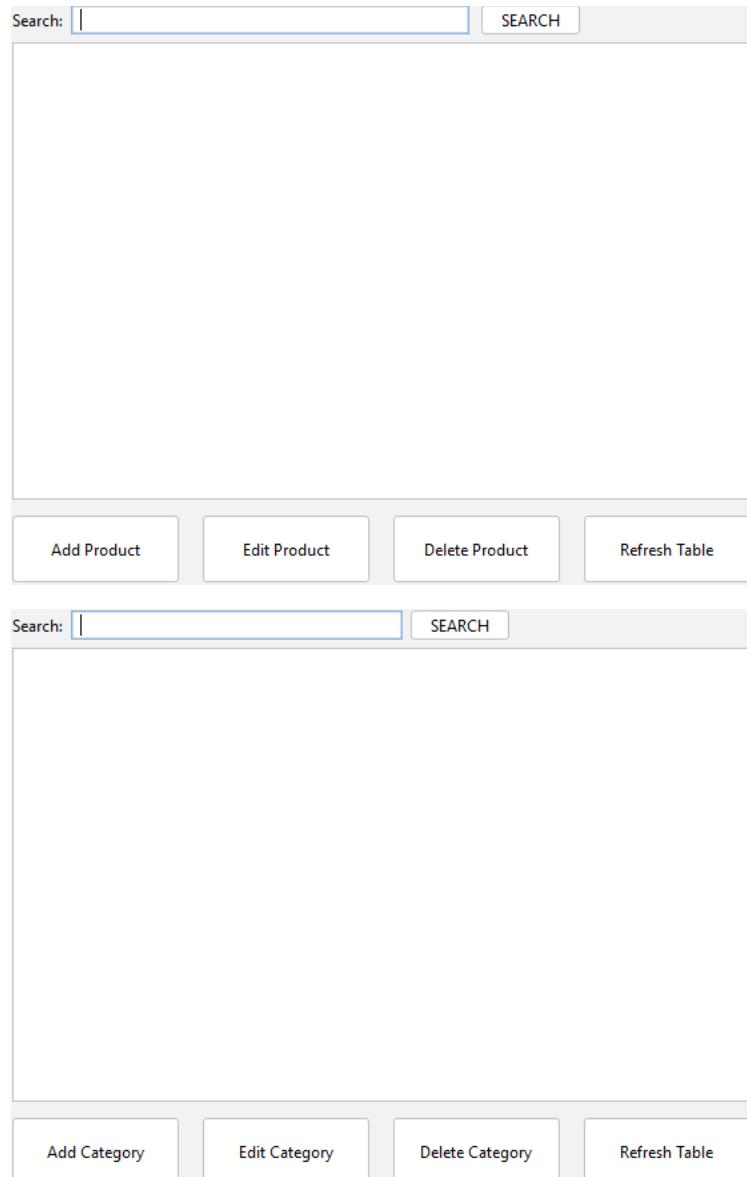
As you can see each of the 7 buttons are very large and labelled clearly as to their respective functions. In addition the buttons turn to a lighter colour when the mouse hovers over it and when a button is clicked its border turns blue to clearly show which button has been pressed. This helps the

Evaluation

user navigate the system with ease and tell where their mouse pointer is on the screen and whether the system registered the click.

5.2.3.2 *Consistently Named Buttons*

Buttons that perform the same function are named the same to make the system less confusing and clearer and easier to use.

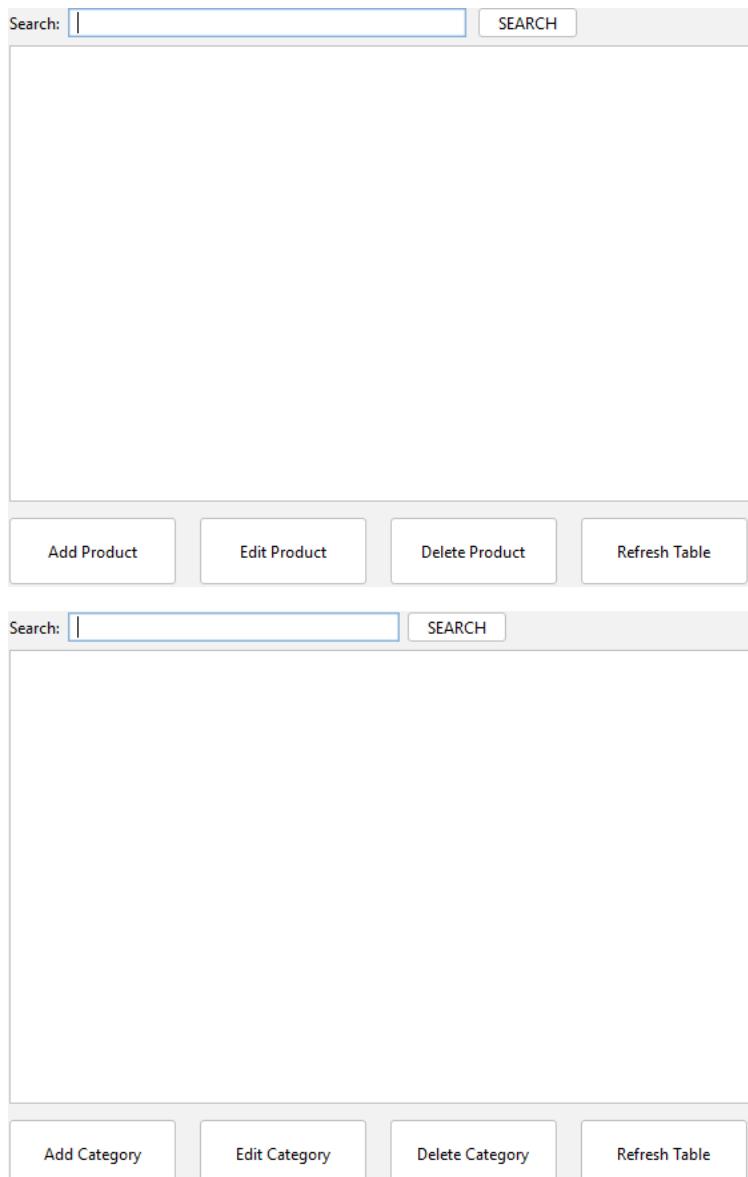


Here you can see that the buttons for creating, updating and deleting products and categories all have the same respective names and so does the button for refreshing the table. By having the buttons named in the same format of function-type e.g add category, update product, it makes the system as a whole easier to use as once the user learns what one button does they know and understand what that button will do for the other sections with the same button.

5.2.3.3 *Consistent Design Scheme and Layout*

I have designed each frame to be similar in layout to make the system easier to use.

Evaluation

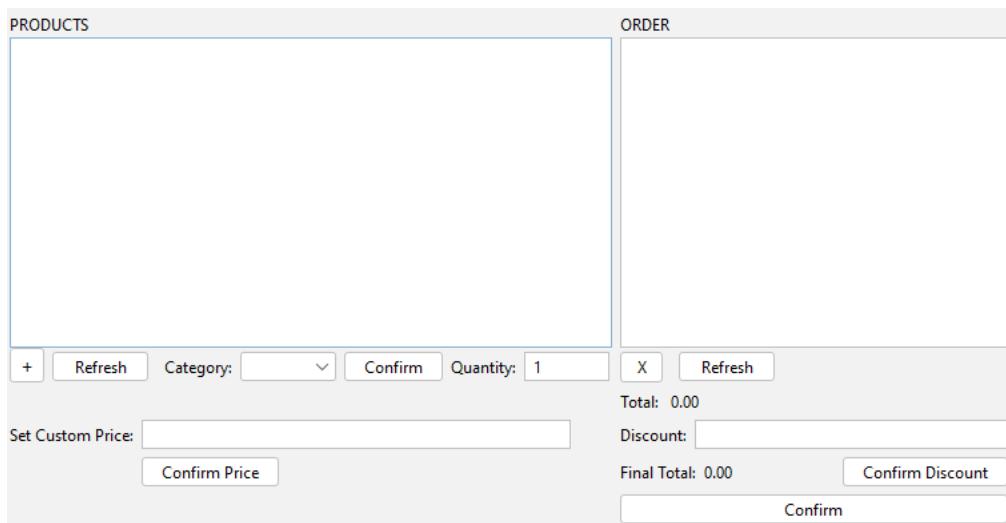


As you can see here, the buttons are in the same order: add, edit, delete, refresh. And also the search bar in every frame that has one is always at the top of the screen above the jTable with a search button next to it. This helps the user navigate and use the system with ease and quickly as they will be able to learn the format/layout very quickly.

5.2.3.4 Simple POS screen

The pos screen I have developed is simple as it is only one frame that allows the user to select products and quantity and view the current order.

Evaluation



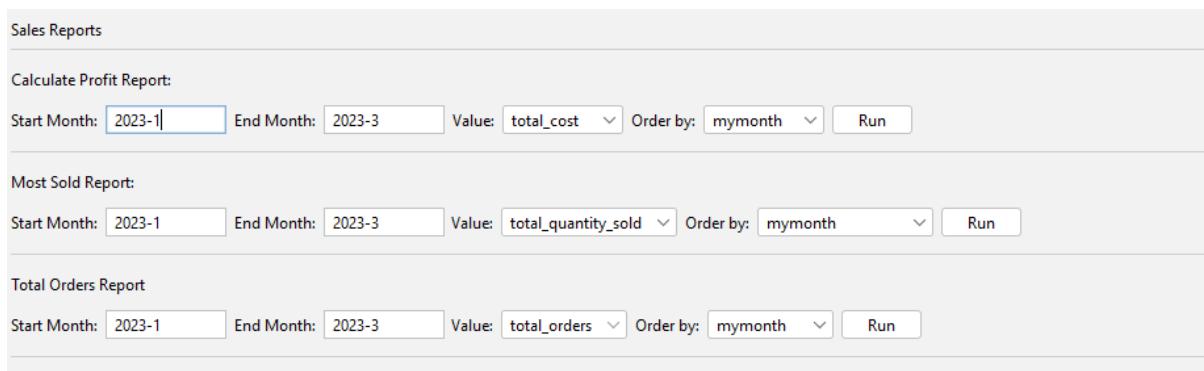
This is much easier to use as the user can view the products and add them to the order on one side and view the order on the other as opposed to using multiple screens to show the product and the order.

5.2.3.5 Cross-Platform Portability

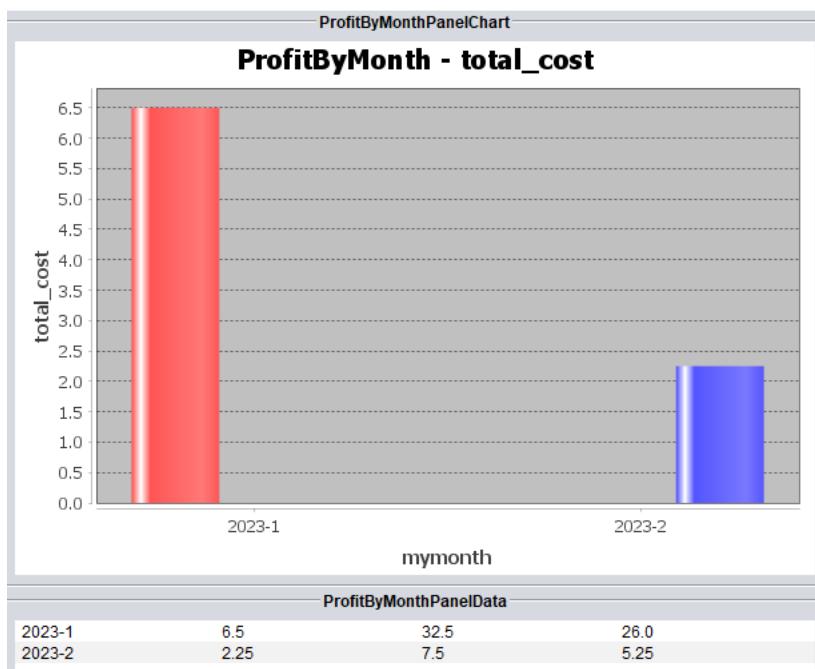
My system was developed using java swing which is a GUI widget toolkit for java and is part of Oracle's foundation java classes. Swing is completely written in java therefore it is cross platform so my system can be used on multiple different platforms. In addition to cross platform usability swing provides platform specific Look and Feel and also pluggable Look and Feel, allowing my application to have Look and Feel independent of the underlying platform.

5.2.3.6 Simple Reports Screen

The reports screen I have developed is very simple as it is just one frame that the user uses to enter the months, the type of report and how it should be ordered. Then from that screen all the user has to do is press run and the selected report is built on a new jframe which is displayed to the user.



As you can see this frame is simple and easy to use.



This is the calculate profit report. To make it easier to understand for the user there is a jtable below the chart so the user can clearly see exact values. This screen has simple well named inputs and the actual report is well labelled to help the user understand the chart.

5.2.4 Limitations

One limitation of my system is that it doesn't notify the user if a product has low stock. This could cause the user to run out of stock for products because they didn't know how low the stock was for those products which could lead to less sales so less profit. Also, it doesn't let the user add a low stock level for each product so that they are notified when the stock for a specific product reaches that low stock level. I don't need to alert the user if a product has 0 stock as when developing the POS section, I created the methods purposely so that they only retrieve products that have stock over 0 so product with 0 stock aren't displayed so the user cannot select them to add them to the order.

Another limitation of my system is that there is no option to log out so a new user can log in to take an order. This means that the new user would have to wait until the old user is timed out before they can log in so they can take an order.

5.2.5 Bits I Added That Could Have Been Done Better + How?

The custom price field and discount price field in the pos section could have been done better. If they actually changed the price of the order rather than just changing the value of the total. In my system I don't store the price of the order, I only store the order and the products that are in the order, this means in order to have the custom price and discount field change the price I would have to add another field to the orders table that displays the price and to make it easier for the user I would probably also add fields to show: the original price, the discount and if the price was set using the custom text field. This way the user can set a custom price that would be stored as the price of that order and they could add a discount that changes the price of the order. If I was to do this I would also have to update the way I do my reports. Instead of building the profitbymonth view using the

sum of the products cost and product price I would build it using the final order price and cost so the report is accurate to the order and not to the original order price or cost.

In the orders section when I search for an order the headers are reset to A B C, this is because I forgot to call the `setColumnHeaders()` method under the search button action listener after I called the `fillTable()` method. I have other sections with a search function however in those sections I have called the `setColumnHeader()` method under the search button action listener so they work as expected.

The POS section was unclear to multiple users in the user testing so that could be named better so it is clearer. This can be done very easily I simply change the text displayed by the button to something like “sell” or “point-of-sale”.

In the POS section if the user adds multiple of the same product to an order separately, the products are represented in the order separately. During my user testing a user found that when they tried to delete one of the products in the order, both are deleted. This is because in my delete method the query deletes all the order products with the selected product primary key. In order to fix this, I would create a method that checks whether the product the user is adding to the order already exists in the order, if it does then I would simply update the quantity of the product in the order rather than creating another instance of that product in the order. This method would be called every time the user tries to add a product to an order. Once this is done then there can only be one instance of each product in the order which would fix the deleting product bug so now that product would be completely deleted from the order, and they can re-add it to the order.

A user from my user testing suggested that I should sum up the cost and price of an order and display it to the user, if I was going to do this, I would also calculate the order profit and display that to the user as well. This would be very easy to do, I would add three more fields to the order jtable, one for cost, one for price, one for profit. Then when retrieving the order data from the database I would also retrieve the sum of the cost field and the sum of the price field and use those value to calculate the total profit then I would display these values in the new fields in the order jtable.

Another feature that I could have done better is the login feature. A user from my user testing suggested that I should have the password hidden using Asterix’s to replace each character when they enter their password so that someone can’t watch as they enter their password. This is very easy to do. I would first change the text field to a JPasswordField and use `JPasswordField.setEchoChar('*')` to mask the password with *. When I need to retrieve the password entered for validation I can use `JPasswordField.getPassword()`. I could even add a check box that allows the user to hide or unhide their password. I would use an if statement to check If the box is checked, if it is I would call `JPasswordField.setEchoChar('*')` to hide the password and if it is unchecked I would call `JPasswordField.setEchoChar((char)0)` so the password is visible.

In the admin section a user from my user testing pointed out that there always needs to be an admin user in the database. If there is only one admin user they shouldn't be allowed to change their user type from admin unless another admin user is created. In order to implement this, I would create a method to return true if there is more than one admin user and false if there is only one. This would be a simple query that sums the number of records in the users table that have a usertype field value of "admin", if the sum is 1 then false would be returned, if the sum is more than 1 then true would be returned. Then I would use an if statement on the delete and update user functions to check whether the user can delete or update the admin user's type. If the method returns true then they can delete the admin user or change their user type, if it returns false then an appropriate message would be output, and they would be stopped from deleting or changing the user type for the admin user.

5.2.6 Bits I Could Have Added That Would Improve the Solution / Would Solve the Limitations

In order to solve the stock notification limitation and achieve the stock notification solution I would add another field to the products table that stores an integer which will be called lowstocklevel. When the user is creating a new product, I would add another text field so they can set the low stock level for the product. Then I would develop a Boolean method to compare the products stock to the products low stock level, if the stock is equal to or less than the low stock level when it is added to the order then the method would return true, and I would use an if statement to check if the method returns true or false. If it returns true, I would have a window pop up that says something like "this product has low stock".

In the products section I could have added a sort by category feature so the user can select a specific category from a drop-down combo box and only products from that category are displayed. This would be very easy to do as I have already coded this exact feature into the POS section so I could just re-use all the methods from that section. This is a feature that I would have added once I had finished the project, but I didn't have time.

In order to achieve success criteria number 10 which is the card/cash payment I would create a new frame that opens when the user presses the confirm order button. This frame would have an option for cash which would just be a text box that the user can enter in the cash amount paid and an option for card where the user can enter in the card details. The reason I didn't need to add this feature but other POS systems I reviewed did is because in those POS systems they recorded profit or money made as the total amount of money they have been paid from customers, whereas in my system it is automatically calculated from how many of each product has been sold. This means that in the other systems they allow an order to be partially paid for whereas my system does not allow for that.

I could have added a feature that allows the admin user to create specific access allowances for each user so they can create a new user and pick each section of the system that the user has access to. Currently in my system there are pre-set user types that decide which sections the user has access to. However, if the admin could create their own user types of just select individual sections of the system for each user, then the system would be more versatile for application in different restaurants and establishments.

5.2.7 Maintenance

My system can be easily changed and have new features added as it is heavily modular and also the software has been designed in two parts. One part is the JFrame classes which make up the user interface and the other part is the Java classes I have created for the main functionality of the system. This means the design can be easily updated and changed without affecting the functionality of the program and the how the program works can be updated and changed without changing the user interface.

The program is heavily commented with all of the methods and classes being commented to explain what they do and how they work so a new developer could easily understand and further develop the program. Also, all the Swing components such as the JTables, buttons etc are named appropriately and clearly so a new developer could easily see where they are referenced in the code.

One limitation of this system is that the user can open multiple of the same window as they can still navigate the menu screen after opening a window. Although in itself this doesn't cause any problems, if lots of the same window are open and are being used for example if multiple of the products screen are open then in one screen a product could be deleted but it won't have updated in the other screen unless the table is refreshed.

This could be fixed by making the JFrames opened from the menu modal so that when they are opened the user cannot navigate back to the menu to open more windows without closing the original one they opened.

6 Final Code

6.1 adminScreen.java

```
public class adminScreen extends javax.swing.JFrame {

    /**
     * Creates new form adminScreen
     */
    public adminScreen() {
        initComponents();
    }

    private void usersButtonActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        // creates new userScreen
        userScreen screen = new userScreen();
    }
}
```

Final Code

```
screen.setDefaultCloseOperation(userScreen.DISPOSE_ON_CLOSE);
screen.pack();
screen.setVisible(true);
}

private void userHoursButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    userHoursScreen screen = new userHoursScreen();
    screen.setDefaultCloseOperation(userHoursScreen.DISPOSE_ON_CLOSE);
    screen.pack();
    screen.setVisible(true);
}
```

6.2 categoryScreen.java

```
import java.util.ArrayList;  
import javax.swing.JOptionPane;  
  
public class categoryScreen extends javax.swing.JFrame {  
  
    /** * Creates new form categoryScreen */  
    public categoryScreen() {  
        initComponents();  
        fillTable("");  
        setColumnNames();  
    }  
  
    public void fillTable(String searchVal){  
        //fill the jTable with contents of the ArrayList  
  
        ArrayList <classes.categories> catList = category.categoryList(searchVal);  
        String [] columnNames = {"ID","Name"};  
        Object [][] rows = new Object[catList.size()][2];  
  
        for(int i = 0; i<catList.size(); i++){  
            rows[i][0] = catList.get(i).getCatPK();  
            rows[i][1] = catList.get(i).getName();  
        }  
  
        classes.table model = new classes.table(rows, columnNames);
```

Final Code

```
categoryTable.setModel(model);
}

public void setColumnNames(){
    //sets jTable column names

    categoryTable.getColumnModel().getColumn(0).setHeaderValue("ID");
    categoryTable.getColumnModel().getColumn(1).setHeaderValue("Name");
}

private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //creates new newCategoryScreen

    newCategoryScreen categoryAdd = new newCategoryScreen();
    categoryAdd.setDefaultCloseOperation(newCategoryScreen.DISPOSE_ON_CLOSE);
    categoryAdd.pack();
    categoryAdd.setVisible(true);

}

private void editButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //creates new updateCategoryScreen
    //sets records to updated values from the updateCategoryScreen

    try{
        updateCategoryScreen categoryUpdate = new updateCategoryScreen();
        int rowIndex = categoryTable.getSelectedRow();
```

Final Code

```
categoryUpdate.category_pk = Integer.valueOf(categoryTable.getValueAt(rowIndex, 0).toString());  
categoryUpdate.nameTextField.setText(categoryTable.getValueAt(rowIndex, 1).toString());  
  
categoryUpdate.setDefaultCloseOperation(updateProductScreen.DISPOSE_ON_CLOSE);  
categoryUpdate.pack();  
categoryUpdate.setVisible(true);  
}  
  
catch (Exception ex){  
    JOptionPane.showMessageDialog(null, "There was an error, please select a row and try again.");  
}  
}  
  
private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    //refresh table  
  
    fillTable("");  
    setColumnNames();  
}  
  
private void deleteButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    //deletes selected record  
  
    int rowIndex = categoryTable.getSelectedRow();  
    int id = Integer.valueOf(categoryTable.getValueAt(rowIndex, 0).toString());  
    try{  
        classes.categories.deleteCategory(id);  
    }  
    catch(Exception ex){
```

Final Code

```
JOptionPane.showMessageDialog(null, "There was an error, please try again.");  
}  
}  
  
private void searchButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    fillTable(searchTextField.getText());  
    setColumnNames();  
}  
}
```

6.3 choiceClockinScreen.java

```
public class choiceClockinScreen extends javax.swing.JFrame {  
    public static int userPrimaryKey;  
  
    /**  
     * Creates new form choiceClockinScreen  
     */  
    public choiceClockinScreen() {  
        initComponents();  
    }  
  
    private void clockinButtonActionPerformed(java.awt.event.ActionEvent evt) {  
        // TODO add your handling code here:  
        classes.clockin.clockUserIn(userPrimaryKey);  
        this.dispose();  
    }  
  
    private void clockoutButtonActionPerformed(java.awt.event.ActionEvent evt) {  
        // TODO add your handling code here:  
        classes.clockin.clockUserOut(userPrimaryKey);  
        classes.clockin.calculateUserHours(userPrimaryKey);  
        this.dispose();  
    }  
}
```

6.4 clockinScreen.java

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

public class clockinScreen extends javax.swing.JFrame {

    /**
     * Creates new form clockinScreen
     */
    public clockinScreen() {
        initComponents();
    }

    private void enterButtonActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String userName = usernameTextField.getText();
        String password = passwordTextField.getText();
        int uLen = userName.length();
        int pLen = password.length();
        PreparedStatement ps;
        ResultSet rs;
        int userPK;

        try {
            if (uLen < 6 || pLen < 6){
```

Final Code

```
JOptionPane.showMessageDialog(null, "invalid, username and password must be 6 or more  
characters long");  
}  
  
else {  
  
    ps = classes.database.getConnection().prepareStatement("SELECT `username`, `password` ,  
`user_pk` PK FROM `users` WHERE `username` = ? AND `password` = ?");  
  
    ps.setString(1, usernameTextField.getText());  
  
    ps.setString(2, passwordTextField.getText());  
  
    rs = ps.executeQuery();  
  
  
  
if (rs.next()){  
  
    userPK = rs.getInt("PK");  
  
    choiceClockinScreen.userPrimaryKey = userPK;  
  
    choiceClockinScreen choiceScreen = new choiceClockinScreen();  
  
    choiceScreen.setDefaultCloseOperation(choiceClockinScreen.DISPOSE_ON_CLOSE);  
  
    choiceScreen.pack();  
  
    choiceScreen.setVisible(true);  
  
}  
  
else{  
  
    JOptionPane.showMessageDialog(null, "incorrect username or password");  
  
}  
  
}  
  
}  
  
}  
  
}  
  
catch(SQLException ex){  
  
    Logger.getLogger(loginScreen.class.getName()).log(Level.SEVERE, null, ex);  
  
}
```

6.5 loginScreen.java

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

public class loginScreen extends javax.swing.JFrame {

    /**
     * Creates new form login
     */
    public loginScreen() {
        initComponents();
    }

    private void loginButtonActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //validates length of username and password
        //validates that the username and password are correct and match a user in the users table in
        //the database
        //creates new menuScreen
        //creates new session record in userSessions table in the database
        //disposes of login form

        String userName = usernameTextField.getText();
        String password = passwordTextField.getText();
        int uLen = userName.length();
        int pLen = password.length();
        PreparedStatement ps;
```

Final Code

```
ResultSet rs;
int userPK;

try {
    if (uLen < 6 || pLen < 6){
        JOptionPane.showMessageDialog(null, "invalid, username and password must be 6 or more
characters long");
    }
    else {
        ps = classes.database.getConnection().prepareStatement("SELECT `username`, `password` ,
`user_pk` PK FROM `users` WHERE `username` = ? AND `password` = ?");
        ps.setString(1, usernameTextField.getText());
        ps.setString(2, passwordTextField.getText());
        rs = ps.executeQuery();

        if (rs.next()){
            userPK = rs.getInt("PK");
            menuScreen menu = new menuScreen();
            menu.setDefaultCloseOperation(menuScreen.EXIT_ON_CLOSE);
            menu.pack();
            menu.setVisible(true);
            int currentSession = classes.userSessions.insertUserSession(userPK);
            menu.sessionID = currentSession;
            menu.userPrimaryKey = userPK;
            this.dispose();
        }
        else{
            JOptionPane.showMessageDialog(null, "incorrect username or password");
        }
    }
}
```

Final Code

```
    }  
}  
  
}  
catch(SQLException ex){  
    Logger.getLogger(loginScreen.class.getName()).log(Level.SEVERE, null, ex);  
}
```

6.6 menuScreen.java

```
import javax.swing.JOptionPane;

public class menuScreen extends javax.swing.JFrame {

    public int sessionID;
    public int userPrimaryKey;

    /**
     * Creates new form menu
     */
    public menuScreen() {
        initComponents();
    }

    private void posButtonActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //validates the session
        //disposes of menuScreen and creates new loginScreen if the session has timed out
        //validates the user type

        boolean valid = classes.userSessions.isSessionValid(sessionID);
        if (valid == false){
            JOptionPane.showMessageDialog(null, "You have been timed out, please log back in.");
            loginScreen login = new loginScreen();
            login.setDefaultCloseOperation(login.EXIT_ON_CLOSE);
            login.pack();
            login.setVisible(true);
            this.dispose();
        }
    }
}
```

Final Code

```
}

else if (valid == true){

    String userType;

    userType = classes.admin.getCurrentUserType();

    if (userType.equals("Admin") || userType.equals("Waiter")){

        posScreen screen = new posScreen();

        screen.setDefaultCloseOperation(posScreen.DISPOSE_ON_CLOSE);

        screen.pack();

        screen.setVisible(true);

        screen.userPrimaryKey = userPrimaryKey;

    }

    else{

        JOptionPane.showMessageDialog(null, "You don't have access to this feature");

    }

}

}

}
```

```
private void adminButtonActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

    //validates the session

    //disposes of menuScreen and creates new loginScreen if the session has timed out

    //validates the user type


    boolean valid = classes.userSessions.isSessionValid(sessionID);

    if (valid == false){

        JOptionPane.showMessageDialog(null, "You have been timed out, please log back in.");

        loginScreen login = new loginScreen();

        login.setDefaultCloseOperation(login.EXIT_ON_CLOSE);

        login.pack();

        login.setVisible(true);

        this.dispose();

    }

}
```

Final Code

```
}
```



```
else if (valid == true){
```



```
    String userType;
```



```
    userType = classes.admin.getCurrentUserType();
```



```
    if (userType.equals("Admin")){
```



```
        adminScreen screen = new adminScreen();
```



```
        screen.setDefaultCloseOperation(adminScreen.DISPOSE_ON_CLOSE);
```



```
        screen.pack();
```



```
        screen.setVisible(true);
```



```
    }
```



```
    else{
```



```
        JOptionPane.showMessageDialog(null, "You don't have access to this feature");
```



```
    }
```



```
}
```



```
}
```

```
private void ordersButtonActionPerformed(java.awt.event.ActionEvent evt) {
```



```
    // TODO add your handling code here:
```



```
    //validates the session
```



```
    //disposes of menuScreen and creates new loginScreen if the session has timed out
```



```
    //validates the user type
```



```
    boolean valid = classes.userSessions.isSessionValid(sessionID);
```



```
    if (valid == false){
```



```
        JOptionPane.showMessageDialog(null, "You have been timed out, please log back in.");
```



```
        loginScreen login = new loginScreen();
```



```
        login.setDefaultCloseOperation(login.EXIT_ON_CLOSE);
```



```
        login.pack();
```



```
        login.setVisible(true);
```



```
        this.dispose();
```

Final Code

```
}

else if (valid == true){

    String userType;

    userType = classes.admin.getCurrentUserType();

    if (userType.equals("Admin")||userType.equals("Waiter")||userType.equals("Kitchen")){
        orderScreen screen = new orderScreen();

        screen.setDefaultCloseOperation(orderScreen.DISPOSE_ON_CLOSE);

        screen.pack();

        screen.setVisible(true);

    }

    else{

        JOptionPane.showMessageDialog(null, "You don't have access to this feature");

    }

}

}

private void productsButtonActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

    //validates the session

    //disposes of menuScreen and creates new loginScreen if the session has timed out

    //validates the user type


    boolean valid = classes.userSessions.isSessionValid(sessionID);

    if (valid == false){

        JOptionPane.showMessageDialog(null, "You have been timed out, please log back in.");

        loginScreen login = new loginScreen();

        login.setDefaultCloseOperation(loginScreen.EXIT_ON_CLOSE);

        login.pack();

        login.setVisible(true);

        this.dispose();

    }

}
```

Final Code

```
}

else if (valid == true){

    String userType;

    userType = classes.admin.getCurrentUserType();

    if

(userType.equals("Admin")| |userType.equals("Waiter")| |userType.equals("Kitchen")| |userType.equals("Warehouse")){

        productScreen screen = new productScreen();

        screen.setDefaultCloseOperation(productScreen.DISPOSE_ON_CLOSE);

        screen.pack();

        screen.setVisible(true);

        screen.userPrimaryKey = userPrimaryKey;

    }

    else{

        JOptionPane.showMessageDialog(null, "You don't have access to this feature");

    }

}

}

private void salesButtonActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

    //validates the session

    //disposes of menuScreen and creates new loginScreen if the session has timed out

    //validates the user type

    boolean valid = classes.userSessions.isSessionValid(sessionID);

    if (valid == false){

        JOptionPane.showMessageDialog(null, "You have been timed out, please log back in.");

        loginScreen login = new loginScreen();

        login.setDefaultCloseOperation(loginScreen.EXIT_ON_CLOSE);

        login.pack();

    }

}
```

Final Code

```
login.setVisible(true);
this.dispose();
}

else if (valid == true){

    String userType;
    userType = classes.admin.getCurrentUserType();
    if (userType.equals("Admin")){

        reportScreen screen = new reportScreen();
        screen.setDefaultCloseOperation(reportScreen.DISPOSE_ON_CLOSE);
        screen.pack();
        screen.setVisible(true);
        screen.userPrimaryKey = userPrimaryKey;
        screen.sessionID = sessionID;
        screen.userType=userType;

    }
    else{
        JOptionPane.showMessageDialog(null, "You don't have access to this feature");
    }
}

private void clockButtonActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:
    //validates the session
    //disposes of menuScreen and creates new loginScreen if the session has timed out
    //validates the user type

    boolean valid = classes.userSessions.isSessionValid(sessionID);
```

Final Code

```
if (valid == false){

    JOptionPane.showMessageDialog(null, "You have been timed out, please log back in.");

    loginScreen login = new loginScreen();

    login.setDefaultCloseOperation(loginScreen.EXIT_ON_CLOSE);

    login.pack();

    login.setVisible(true);

    this.dispose();

}

else if (valid == true){

    String userType;

    userType = classes.admin.getCurrentUserType();

    if

(userType.equals("Admin")||userType.equals("Waiter")||userType.equals("Kitchen")||userType.equals("Warehouse")){

        clockinScreen screen = new clockinScreen();

        screen.setDefaultCloseOperation(clockinScreen.DISPOSE_ON_CLOSE);

        screen.pack();

        screen.setVisible(true);

    }

    else{

        JOptionPane.showMessageDialog(null, "You don't have access to this feature");

    }

}

}

private void categoryButtonActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:

    //validates the session

    //disposes of menuScreen and creates new loginScreen if the session has timed out

    //validates the user type
```

Final Code

```
boolean valid = classes.userSessions.isSessionValid(sessionID);

if (valid == false){

    JOptionPane.showMessageDialog(null, "You have been timed out, please log back in.");

    loginScreen login = new loginScreen();

    login.setDefaultCloseOperation(loginScreen.EXIT_ON_CLOSE);

    login.pack();

    login.setVisible(true);

    this.dispose();

}

else if (valid == true){

    String userType;

    userType = classes.admin.getCurrentUserType();

    if (userType.equals("Admin") || userType.equals("Warehouse")){

        categoryScreen screen = new categoryScreen();

        screen.setDefaultCloseOperation(categoryScreen.DISPOSE_ON_CLOSE);

        screen.pack();

        screen.setVisible(true);

    }

    else{

        JOptionPane.showMessageDialog(null, "You don't have access to this feature");

    }

}

}
```

6.7 newCategoryScreen.java

```
public class newCategoryScreen extends javax.swing.JFrame {  
    public int userPrimaryKey;  
  
    /**  
     * Creates new form newProductScreen  
     */  
    public newCategoryScreen() {  
        initComponents();  
    }  
  
    private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {  
        // TODO add your handling code here:  
        //creates new category and names it with the text in the text field  
  
        classes.categories category;  
  
        String category_name = nameTextField.getText();  
  
        category = new classes.categories(1, category_name);  
  
        classes.categories.addCategory(category);  
        this.dispose();  
    }  
}
```

6.8 newProductScreen.java

```
import classes.dataVerifier;  
import java.util.HashMap;  
  
public class newProductScreen extends javax.swing.JFrame {  
    public int userPrimaryKey;  
  
    /**  
     * Creates new form newProductScreen  
     */  
    public newProductScreen() {  
        initComponents();  
        comboBoxAttach();  
  
        //set the names of the data entry fields so I know how to validate them  
        //because I found out the setName and getName don't work unless you actually set the name of  
        JComponents  
        fieldTypeSet();  
    }  
  
    //this is my method to set the name of the object holding a data to be validated  
    //so when my inputVerifier receives the object I can use the name of the object to  
    //decide how to validate the value in the object  
    public void fieldTypeSet(){  
        products_product_name.setName("Product Name:");  
        products_product_description.setName("Product Description:");  
        priceTextField.setName("Price:");  
        costTextField.setName("Cost:");  
        stockTextField.setName("Stock:");  
    }  
}
```

Final Code

```
}
```

```
public void comboBoxAttach(){  
    //adds categories to the combo box  
  
    classes.categories category = new classes.categories();  
    HashMap<String, Integer> map = category.addToCombo();  
    for (String set: map.keySet()){  
        categoryComboBox.addItem(set);  
    }  
}  
  
private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    //creates new products using inputs from user  
  
    classes.products product;  
    classes.categories category = new classes.categories();  
    HashMap<String, Integer> map = category.addToCombo();  
  
    String product_name = products_product_name.getText();  
    String product_description = products_product_description.getText();  
    float product_sellprice;  
    float product_cost;  
    int stock;  
    int categoryfk = 0;  
    String categoryName = null;  
  
    stock = Integer.valueOf(stockTextField.getText());
```

Final Code

```
categoryfk = map.get(categoryComboBox.getSelectedItem().toString());  
categoryName = categoryComboBox.getSelectedItem().toString();  
product_sellprice = Float.parseFloat(priceTextField.getText());  
product_cost = Float.parseFloat(costTextField.getText());  
  
product = new classes.products(1, product_name, product_description, product_sellprice,  
product_cost, userPrimaryKey, null, null, false, categoryName, stock, categoryfk);  
  
classes.products.addProduct(product);  
this.dispose();  
}  
  
private void products_product_nameFocusGained(java.awt.event.FocusEvent evt) {  
    // TODO add your handling code here:  
    classes.dataValidation.typeIsString(100);  
    //classes.dataValidation.typeIsDate();  
    classes.dataValidation.typeIsNull(false);  
}  
  
private void products_product_descriptionFocusGained(java.awt.event.FocusEvent evt) {  
    // TODO add your handling code here:  
    classes.dataValidation.typeIsString(255);  
    classes.dataValidation.typeIsNull(false);  
}  
  
private void priceTextFieldFocusGained(java.awt.event.FocusEvent evt) {  
    // TODO add your handling code here:  
    classes.dataValidation.typeIsDecimal(10,2);  
    classes.dataValidation.typeIsNull(false);  
}
```

Final Code

```
private void costTextFieldFocusGained(java.awt.event.FocusEvent evt) {  
    // TODO add your handling code here:  
    classes.dataValidation.typeIsDecimal(10,2);  
    classes.dataValidation.typeIsNull(false);  
}
```

```
private void stockTextFieldFocusGained(java.awt.event.FocusEvent evt) {  
    // TODO add your handling code here:  
    classes.dataValidation.typeIsInteger(6);  
    classes.dataValidation.typeIsNull(false);  
}
```

6.9 newUserScreen.java

```
public class newUserScreen extends javax.swing.JFrame {  
    public int userPrimaryKey;  
  
    /**  
     * Creates new form newUserScreen  
     */  
    public newUserScreen() {  
        initComponents();  
    }  
  
    private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {  
        // TODO add your handling code here:  
        //creates a new user using user inputs from text fields and combo box  
  
        classes.admin user;  
  
        String username = nameTextField.getText();  
        String password = passwordTextField.getText();  
        String type = typeComboBox.getSelectedItem().toString();  
  
        user = new classes.admin(1, username, password, type);  
  
        classes.admin.addUser(user);  
        this.dispose();  
    }  
}
```

6.10 orderScreen.java

```
import java.util.ArrayList;

public class orderScreen extends javax.swing.JFrame {

    /**
     * Creates new form orderScreen
     */
    public orderScreen() {
        initComponents();
        fillOrderTable("");
        setColumnNames();
    }

    public void fillOrderTable(String orderPK){

        classes.orders order = new classes.orders();
        ArrayList <classes.orders> orderList = order.ordersList(orderPK);
        String [] columnNames = {"ID", "User-ID", "Date"};
        Object [][] rows = new Object[orderList.size()][3];

        for(int i = 0; i<orderList.size(); i++){
            rows[i][0] = orderList.get(i).getOrderPK();
            rows[i][1] = orderList.get(i).getFK();
            rows[i][2] = orderList.get(i).getDate();
        }

        classes.table model = new classes.table(rows, columnNames);
        orderTable.setModel(model);
    }
}
```

Final Code

```
}

public void setColumnNames(){

    orderTable.getColumnModel().getColumn(0).setHeaderValue("ID");
    orderTable.getColumnModel().getColumn(1).setHeaderValue("User-ID");
    orderTable.getColumnModel().getColumn(2).setHeaderValue("Date");

}

private void searchButtonActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:
    String orderPK = searchTextField.getText();
    fillOrderTable(orderPK);
}

private void viewOrderButtonActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:
    int rowIndex = orderTable.getSelectedRow();
    int orderPK = Integer.valueOf(orderTable.getValueAt(rowIndex, 0).toString());

    productsInOrderScreen.orderPK = orderPK;
    productsInOrderScreen screen = new productsInOrderScreen();
    screen.setDefaultCloseOperation(productsInOrderScreen.DISPOSE_ON_CLOSE);
    screen.pack();
    screen.setVisible(true);

}
```

6.11 posScreen.java

```
import java.util.ArrayList;
import java.util.HashMap;

public class posScreen extends javax.swing.JFrame {

    boolean orderCreated = false;
    public int userPrimaryKey;

    /**
     * Creates new form posScreen
     */
    public posScreen() {
        initComponents();
        fillProductsTable("",0,0);
        comboBoxAttach();
        setProductColumnNames();
    }

    public void fillProductsTable(String searchVal, int stock, int catfk){
        //fills the jTable with contents of arrayList

        classes.products prod = new classes.products();
        ArrayList <classes.products> prodList = prod.productList("", 0, catfk);

        String [] columnNames = {"ID","Name","Description","Price","Cost","Category","Stock"};
        Object [][] rows = new Object[prodList.size()][7];

        for(int i = 0; i<prodList.size(); i++){
    }
```

Final Code

```
rows[i][0] = prodList.get(i).getProductPK();
rows[i][1] = prodList.get(i).getName();
rows[i][2] = prodList.get(i).getDescription();
rows[i][3] = prodList.get(i).getPrice();
rows[i][4] = prodList.get(i).getCost();
rows[i][5] = prodList.get(i).getCat();
rows[i][6] = prodList.get(i).getStock();
}

classes.table model = new classes.table(rows, columnNames);
productTable.setModel(model);
}

public void setProductColumnNames(){
//sets the jTable column names

productTable.getColumnModel().getColumn(0).setHeaderValue("ID");
productTable.getColumnModel().getColumn(1).setHeaderValue("Name");
productTable.getColumnModel().getColumn(2).setHeaderValue("Description");
productTable.getColumnModel().getColumn(3).setHeaderValue("Price");
productTable.getColumnModel().getColumn(4).setHeaderValue("Cost");
productTable.getColumnModel().getColumn(5).setHeaderValue("Category");
productTable.getColumnModel().getColumn(6).setHeaderValue("Stock");
}

public void comboBoxAttach(){
//adds categories to the combo box

classes.categories category = new classes.categories();
HashMap<String, Integer> map = category.addToCombo();
for (String set: map.keySet()){


```

Final Code

```
categoryComboBox.addItem(set);
}

}

public void fillOrderTable(int orderPK){
//fills the jTable with contents of arrayList

classes.orderProducts orderProduct = new classes.orderProducts();
ArrayList <classes.orderProducts> productsList = orderProduct.productInOrderList(orderPK);
String [] columnNames = {"ID","Name","Quantity","Price"};
Object [][] rows = new Object[productsList.size()][4];

for(int i = 0; i<productsList.size(); i++){
    rows[i][0] = productsList.get(i).getProductPK();
    rows[i][1] = productsList.get(i).getProductName();
    rows[i][2] = productsList.get(i).getQuantity();
    rows[i][3] = productsList.get(i).getPrice();
}

classes.table model = new classes.table(rows, columnNames);
orderTable.setModel(model);
}

public void setOrderColumnNames(){

orderTable.getColumnModel().getColumn(0).setHeaderValue("ID");
orderTable.getColumnModel().getColumn(1).setHeaderValue("Name");
orderTable.getColumnModel().getColumn(2).setHeaderValue("Quantity");
orderTable.getColumnModel().getColumn(3).setHeaderValue("Price");
}
```

Final Code

```
public void calculateTotalPrice(){

    float total = 0;
    float finalPrice = 0;
    String totalPriceString;
    String finalPriceString;

    for (int rowIndex = 0; rowIndex<orderTable.getRowCount(); rowIndex++){
        total = total + Float.parseFloat(orderTable.getValueAt(rowIndex, 3).toString());
    }

    totalPriceString = Float.toString(total);

    totalPriceLabel.setText(totalPriceString);

    finalPrice = total;
    finalPriceString = Float.toString(finalPrice);

    finalPriceLabel.setText(finalPriceString);
}

public void calculateNewTotalPrice(){

    float total = 0;
    float finalPrice = 0;
    String totalPriceString;
    String finalPriceString;

    float deletedProductPrice = 0;
```

Final Code

```
int rowIndex = orderTable.getSelectedRow();
deletedProductPrice = Float.parseFloat(orderTable.getValueAt(rowIndex, 3).toString());

total = Float.parseFloat(totalPriceLabel.getText());
finalPrice = Float.parseFloat(finalPriceLabel.getText());

total = total - deletedProductPrice;
finalPrice = finalPrice - deletedProductPrice;

totalPriceString = Float.toString(total);
finalPriceString = Float.toString(finalPrice);

totalPriceLabel.setText(totalPriceString);
finalPriceLabel.setText(finalPriceString);

}

public void setFinalTotalToCustom(){

    String customPrice;
    customPrice = customPriceTextField.getText();
    finalPriceLabel.setText(customPrice);

}

public void discountFinalTotal(){

    float total = 0;
    float discount = 0;
    float finalPrice = 0;
```

Final Code

```
String finalPriceString;

total = Float.parseFloat(totalPriceLabel.getText());
discount = Float.parseFloat(discountTextField.getText());
finalPrice = total - discount;

finalPriceString = Float.toString(finalPrice);

finalPriceLabel.setText(finalPriceString);

}

/** 
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jLabel11 = new javax.swing.JLabel();
    jScrollPane1 = new javax.swing.JScrollPane();
    orderTable = new javax.swing.JTable();
    confirmOrderButton = new javax.swing.JButton();
    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    jScrollPane2 = new javax.swing.JScrollPane();
```

Final Code

```
productTable = new javax.swing.JTable();  
jLabel5 = new javax.swing.JLabel();  
deleteProductFromOrderButton = new javax.swing.JButton();  
addProductToOrderButton = new javax.swing.JButton();  
jLabel6 = new javax.swing.JLabel();  
customPriceTextField = new javax.swing.JTextField();  
confirmPriceButton = new javax.swing.JButton();  
refreshProductsTable = new javax.swing.JButton();  
refreshOrderTable = new javax.swing.JButton();  
categoryComboBox = new javax.swing.JComboBox<>();  
jLabel7 = new javax.swing.JLabel();  
confirmCategoryButton = new javax.swing.JToggleButton();  
jLabel8 = new javax.swing.JLabel();  
quantityTextField = new javax.swing.JTextField();  
totalPriceLabel = new javax.swing.JLabel();  
finalPriceLabel = new javax.swing.JLabel();  
discountTextField = new javax.swing.JTextField();  
jButton1 = new javax.swing.JButton();  
  
jLabel11.setText("jLabel11");  
  
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);  
  
orderTable.setModel(new javax.swing.table.DefaultTableModel(  
    new Object [][] {  
          
    },  
    new String [] {  
          
    }  
));
```

Final Code

```
jScrollPane1.setViewportView(orderTable);

confirmOrderButton.setText("Confirm");
confirmOrderButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        confirmOrderButtonActionPerformed(evt);
    }
});

jLabel1.setText("Total: ");

jLabel2.setText("Discount:");

jLabel3.setText("Final Total:");

jLabel4.setText("ORDER");

productTable.setModel(new javax.swing.table.DefaultTableModel(
    new Object [][] {
        {},
        {},
        {},
        {},
        {}
    },
    new String [] {
        ""
    }
));
jScrollPane2.setViewportView(productTable);

jLabel5.setText("PRODUCTS");
```

Final Code

```
deleteProductFromOrderButton.setText("X");
deleteProductFromOrderButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        deleteProductFromOrderButtonActionPerformed(evt);
    }
});

addProductToOrderButton.setFont(new java.awt.Font("Segoe UI", 0, 14)); // NOI18N
addProductToOrderButton.setText("+");
addProductToOrderButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        addProductToOrderButtonActionPerformed(evt);
    }
});

jLabel6.setText("Set Custom Price:");

customPriceTextField.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        customPriceTextFieldActionPerformed(evt);
    }
});

confirmPriceButton.setText("Confirm Price");
confirmPriceButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        confirmPriceButtonActionPerformed(evt);
    }
});
```

Final Code

```
refreshProductsTable.setText("Refresh");
refreshProductsTable.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        refreshProductsTableActionPerformed(evt);
    }
});

refreshOrderTable.setText("Refresh");
refreshOrderTable.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        refreshOrderTableActionPerformed(evt);
    }
});

jLabel7.setText("Category:");
confirmCategoryButton.setText("Confirm");
confirmCategoryButton.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        confirmCategoryButtonActionPerformed(evt);
    }
});

jLabel8.setText("Quantity:");
quantityTextField.setText("1");

totalPriceLabel.setText("0.00");

finalPriceLabel.setText("0.00");
```

Final Code

```
jButton1.setText("Confirm Discount");

jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGap(10, 10, 10)
        .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(10, 10, 10)
        .addComponent(jLabel5)
        .addGap(10, 10, 10)
        .addGroup(layout.createSequentialGroup()
            .addComponent(addProductToOrderButton)
            .addGap(10, 10, 10)
            .addComponent(refreshProductsTable)
            .addGap(10, 10, 10)
            .addComponent(jLabel7)
            .addGap(10, 10, 10)
            .addComponent(categoryComboBox, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(10, 10, 10)
            .addComponent(confirmCategoryButton)
            .addGap(10, 10, 10)
            .addComponent(jLabel8)
        ).addGap(10, 10, 10)
    ).addGap(10, 10, 10)
).addGap(10, 10, 10)
.addContainerGap()
);
```

Final Code

```
.addComponent(quantityTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel6)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(confirmPriceButton)

.addComponent(customPriceTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
322, javax.swing.GroupLayout.PREFERRED_SIZE)))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(finalPriceLabel)

.addGap(83, 83, 83)

.addComponent(jButton1)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)

.addComponent(jLabel4)

.addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 291, Short.MAX_VALUE)

.addComponent(confirmOrderButton, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)

.addComponent(deleteProductFromOrderButton,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

.addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
```

Final Code

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGap(12, 12, 12)
        .addComponent(refreshOrderTable))
    .addGroup(layout.createSequentialGroup()
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(totalPriceLabel)))
    .addGroup(layout.createSequentialGroup()
        .addComponent(jLabel2)
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(discountTextField)))
    .addContainerGap(32, Short.MAX_VALUE))
);

layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jLabel4)
            .addComponent(jLabel5))
        .addGap(2, 2, 2)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
            .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT_SIZE, 233,
Short.MAX_VALUE)
            .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 0,
Short.MAX_VALUE)))
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(addProductToOrderButton)
                .addComponent(refreshProductsTable)
                .addComponent(refreshOrderTable))
```

Final Code

```
.addComponent(categoryComboBox, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(jLabel7)

.addComponent(confirmCategoryButton)

.addComponent(jLabel8)

.addComponent(quantityTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

.addGroup(layout.createSequentialGroup()

.addGap(3, 3, 3)

.addComponent(deleteProductFromOrderButton,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel1)

.addComponent(totalPriceLabel)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel2)

.addComponent(jLabel6)

.addComponent(customPriceTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

.addComponent(discountTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(confirmPriceButton)

.addComponent(jLabel3)

.addComponent(finalPriceLabel)

.addComponent(jButton1))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(confirmOrderButton)

.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
```

Final Code

```
);

pack();
}// </editor-fold>

private void confirmOrderButtonActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:
    for (int rowIndex = 0; rowIndex < orderTable.getRowCount(); rowIndex++){
        int productPK = Integer.valueOf(orderTable.getValueAt(rowIndex, 0).toString());
        int quantity = Integer.valueOf(orderTable.getValueAt(rowIndex, 2).toString());
        classes.orders.updateStock(productPK, quantity);
    }

    orderCreated = false;
    fillOrderTable(0);
    setOrderColumnNames();
    totalPriceLabel.setText("0.00");
    finalPriceLabel.setText("0.00");
}

private void deleteProductFromOrderButtonActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:
    int rowIndex = orderTable.getSelectedRow();
    int productPK = Integer.valueOf(orderTable.getValueAt(rowIndex, 0).toString());
    classes.orderProducts.deleteOrderProduct(productPK);
    calculateNewTotalPrice();
}

private void addProductToOrderButtonActionPerformed(java.awt.event.ActionEvent evt) {

    // TODO add your handling code here:
    int rowIndex = productTable.getSelectedRow();
```

Final Code

```
int quantity = Integer.valueOf(quantityTextField.getText());  
int id = Integer.valueOf(productTable.getValueAt(rowIndex, 0).toString());  
  
if (orderCreated == false){  
    classes.orders.addOrder(userPrimaryKey);  
    orderCreated = true;  
  
}  
int orderPK = classes.orders.getLatestOrder(userPrimaryKey);  
classes.orderProducts.addOrderProduct(quantity, id, orderPK);  
fillOrderTable(orderPK);  
setOrderColumnNames();  
calculateTotalPrice();  
}  
  
private void customPriceTextFieldActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
}  
  
private void refreshProductsTableActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    fillProductsTable("",0,0);  
    setProductColumnNames();  
}  
  
private void confirmCategoryButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    classes.categories category = new classes.categories();  
    HashMap<String, Integer> map = category.addToCombo();  
    int catfk = map.get(categoryComboBox.getSelectedItem().toString());  
    fillProductsTable("", 0, catfk);
```

Final Code

```
setProductColumnNames();  
}  
  
private void refreshOrderTableActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    int orderPK = classes.orders.getLatestOrder(userPrimaryKey);  
    fillOrderTable(orderPK);  
    setOrderColumnNames();  
}  
  
private void confirmPriceButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    setFinalTotalToCustom();  
}  
  
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    discountFinalTotal();  
}
```

6.12 productScreen.java

```
import java.util.ArrayList;
import javax.swing.JOptionPane;

public class productScreen extends javax.swing.JFrame {
    public int userPrimaryKey;
    classes.products product;

    /**
     * Creates new form productScreen
     */
    public productScreen() {
        initComponents();
        fillTable("", -1, 0);
        setColumnNames();
    }

    public void fillTable(String searchVal, int stock, int catfk){
        //fills the jTable with contents of arrayList

        classes.products prod = new classes.products();
        ArrayList <classes.products> prodList = prod.productList(searchVal, -1, 0);
        String [] columnNames = {"ID","Name","Description","Price","Cost","Category","Stock"};
        Object [][] rows = new Object[prodList.size()][7];

        for(int i = 0; i<prodList.size(); i++){
            rows[i][0] = prodList.get(i).getProductPK();
```

Final Code

```
rows[i][1] = prodList.get(i).getName();
rows[i][2] = prodList.get(i).getDescription();
rows[i][3] = prodList.get(i).getPrice();
rows[i][4] = prodList.get(i).getCost();
rows[i][5] = prodList.get(i).getCat();
rows[i][6] = prodList.get(i).getStock();

}

classes.table model = new classes.table(rows, columnNames);
productTable.setModel(model);

}

public void setColumnNames(){
//sets the jTable column names

productTable.getColumnModel().getColumn(0).setHeaderValue("ID");
productTable.getColumnModel().getColumn(1).setHeaderValue("Name");
productTable.getColumnModel().getColumn(2).setHeaderValue("Description");
productTable.getColumnModel().getColumn(3).setHeaderValue("Price");
productTable.getColumnModel().getColumn(4).setHeaderValue("Cost");
productTable.getColumnModel().getColumn(5).setHeaderValue("Category");
productTable.getColumnModel().getColumn(6).setHeaderValue("Stock");

}

private void editButtonActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
try{
    updateProductScreen productUpdate = new updateProductScreen();
    int rowIndex = productTable.getSelectedRow();
    //updateProductScreen.userPrimaryKey = userPrimaryKey;
}
```

Final Code

```
productUpdate.product_pk = Integer.valueOf(productTable.getValueAt(rowIndex, 0).toString());  
productUpdate.nameTextField.setText(productTable.getValueAt(rowIndex, 1).toString());  
productUpdate.descriptionTextField.setText(productTable.getValueAt(rowIndex, 2).toString());  
productUpdate.priceTextField.setText(productTable.getValueAt(rowIndex, 3).toString());  
productUpdate.costTextField.setText(productTable.getValueAt(rowIndex, 4).toString());  
productUpdate.categoryComboBox.setSelectedItem(productTable.getValueAt(rowIndex, 5));  
productUpdate.stockTextField.setText(productTable.getValueAt(rowIndex, 6).toString());  
  
productUpdate.setDefaultCloseOperation(updateProductScreen.DISPOSE_ON_CLOSE);  
productUpdate.pack();  
productUpdate.setVisible(true);  
  
}  
catch (Exception ex){  
    JOptionPane.showMessageDialog(null, "There was an error, please select a row and try again.");  
}  
  
}  
  
private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    newProductScreen productadd = new newProductScreen();  
    productadd.setDefaultCloseOperation(newProductScreen.DISPOSE_ON_CLOSE);  
    productadd.pack();  
    productadd.setVisible(true);  
    productadd.userPrimaryKey = userPrimaryKey;  
}
```

Final Code

```
private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    fillTable("", -1, 0);  
    setColumnNames();  
}  
  
private void deleteButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    int rowIndex = productTable.getSelectedRow();  
    int id = Integer.valueOf(productTable.getValueAt(rowIndex, 0).toString());  
    try{  
        classes.products.deleteProduct(id);  
    }  
    catch(Exception ex){  
        JOptionPane.showMessageDialog(null, "There was an error, please try again.");  
    }  
}  
  
private void searchButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    fillTable(searchTextField.getText(), -1, 0);  
    setColumnNames();  
}
```

6.13 productsInOrderScreen.java

```

import java.util.ArrayList;

public class productsInOrderScreen extends javax.swing.JFrame {

    public static int orderPK;

    /**
     * Creates new form orderProductScreen
     */
    public productsInOrderScreen() {

        initComponents();
        fillProductsInOrderTable();
        setColumnNames();
    }

    public void fillProductsInOrderTable(){

        classes.productsInOrder product = new classes.productsInOrder();
        ArrayList <classes.productsInOrder> productList = product.productsInOrderList(orderPK);
        String [] columnNames = {"ID","Name","Price","Cost","Quantity","Profit"};
        Object [][] rows = new Object[productList.size()][6];

        for(int i = 0; i<productList.size(); i++){
            rows[i][0] = productList.get(i).getProductPK();
            rows[i][1] = productList.get(i).getProductName();
            rows[i][2] = productList.get(i).getPrice();
            rows[i][3] = productList.get(i).getCost();
        }
    }
}

```

Final Code

```
rows[i][4] = productList.get(i).getQuantity();
rows[i][5] = productList.get(i).getProfit();

}

classes.table model = new classes.table(rows, columnNames);
productsInOrderTable.setModel(model);

}

public void setColumnNames(){

productsInOrderTable.getColumnModel().getColumn(0).setHeaderValue("ID");
productsInOrderTable.getColumnModel().getColumn(1).setHeaderValue("Name");
productsInOrderTable.getColumnModel().getColumn(2).setHeaderValue("Price");
productsInOrderTable.getColumnModel().getColumn(3).setHeaderValue("Cost");
productsInOrderTable.getColumnModel().getColumn(4).setHeaderValue("Quantity");
productsInOrderTable.getColumnModel().getColumn(5).setHeaderValue("Profit");

}

}
```

6.14 reportScreen.java

```
import javax.swing.JFrame;
import javax.swing.JOptionPane;

public class reportScreen extends javax.swing.JFrame {

    public int sessionID;
    public int userPrimaryKey;
    public String userType;

    /**
     * Creates new form reportmenuScreen
     */
    public reportScreen() {
        initComponents();
    }

    private void printVars(){
        System.out.println("sessionID: "+sessionID);
        System.out.println("userPrimaryKey: "+userPrimaryKey);
        System.out.println("userType: "+userType);
    }

    //this assembles the chart and the data table components onto panels and uses the
    java.awt.GridBagLayout layout manager to create the report

    private void RunProfitReportActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:

        printVars();

        boolean valid = classes.userSessions.isSessionValid(sessionID);
    }
}
```

Final Code

```
System.out.println(sessionID);

if (valid == false){

    JOptionPane.showMessageDialog(null, "You have been timed out, please log back in.");

    loginScreen login = new loginScreen();

    login.setDefaultCloseOperation(loginScreen.EXIT_ON_CLOSE);

    login.pack();

    login.setVisible(true);

    this.dispose();

}

String userType;

userType = classes.admin.getCurrentUserType();

if (userType.equals("Admin") || userType.equals("Waiter")){

    System.out.println(userType);

    System.out.println("access granted");

//have a quick look at the report parameters

String jTextFieldCPR_startmonthVal = CPR_startmonthVal.getText();

System.out.println(jTextFieldCPR_startmonthVal);

String jTextFieldCPR_endmonthVal = CPR_endmonthVal.getText();

System.out.println(jTextFieldCPR_endmonthVal);

String jComboBox1_CPR_ValueToShow = CPR_ValueToShow.getSelectedItem().toString();

System.out.println(jComboBox1_CPR_ValueToShow);

String jComboBox1_CPRorderValue = jComboBox1_CPRorder.getSelectedItem().toString();

System.out.println(jComboBox1_CPRorderValue);

//set some titles for the report components
```

Final Code

```
String frameTitle="ProfitByMonth";
String chartTitle="ProfitByMonthPanelChart";
String tableTitle="ProfitByMonthPanelData";

//new frame to display the report - initialised by the ProductsByCategorySummed_view
method which assembles the chart and the data table into a report

JFrame frame = classes.ProfitByMonth_view.fillProfitByMonth_viewAssemble(
jTextFieldCPR_startmonthVal, jTextFieldCPR_endmonthVal, jComboBox1_CPR_ValueToShow,
jComboBox1_CPROrderValue, 600, 400, 600, 200, frameTitle, chartTitle, tableTitle);

//Display the frame
frame.pack();
frame.setVisible(true);

}

else{
    System.out.println(userType);
    JOptionPane.showMessageDialog(null, "You don't have access to this feature");
}

}

private void RunMostSoldReportActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here: total_quantity_sold
    printVars();

    boolean valid = classes.userSessions.isSessionValid(sessionID);
    System.out.println(sessionID);
    if (valid == false){
        JOptionPane.showMessageDialog(null, "You have been timed out, please log back in.");
        loginScreen login = new loginScreen();
        login.setDefaultCloseOperation(login.EXIT_ON_CLOSE);
```

Final Code

```
login.pack();
login.setVisible(true);
this.dispose();
}

String userType;
userType = classes.admin.getCurrentUserType();

if (userType.equals("Admin") || userType.equals("Waiter")){
    System.out.println(userType);
    System.out.println("access granted");

    //have a quick look at the report parameters
    String jTextFieldMSR_startmonthVal = MSR_startmonthVal.getText();
    System.out.println(jTextFieldMSR_startmonthVal);

    String jTextFieldMSR_endmonthVal = MSR_endmonthVal.getText();
    System.out.println(jTextFieldMSR_endmonthVal);

    String jComboBox1_MSR_ValueToShow = MSR_ValueToShow.getSelectedItem().toString();
    System.out.println(jComboBox1_MSR_ValueToShow);

    String jComboBox1_MSRRorderValue = jComboBox1_MSRRorder.getSelectedItem().toString();
    System.out.println(jComboBox1_MSRRorderValue);

    //set some titles for the report components
    String frameTitle="MostSoldByMonth";
    String chartTitle="MostSoldByMonthPanelChart";
    String tableTitle="MostSoldByMonthPanelData";
```

Final Code

```
//new frame to display the report - initialised by the ProductsByCategorySummed_view  
method which assembles the chart and the data table into a report  
  
JFrame frame = classes.MostSoldByMonth_view.fillMostSoldByMonth_viewAssemble(  
jTextFieldMSR_startmonthVal, jTextFieldMSR_endmonthVal, jComboBox1_MSR_ValueToShow,  
jComboBox1_MSRRordervalue, 600, 400, 600, 200, frameTitle, chartTitle, tableTitle);  
  
  
//Display the frame  
frame.pack();  
frame.setVisible(true);  
  
}  
else{  
    System.out.println(userType);  
    JOptionPane.showMessageDialog(null, "You don't have access to this feature");  
}  
  
}  
  
  
private void CPR_startmonthValActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
}  
  
  
private void MSR_startmonthValActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
}  
  
  
private void TOR_startmonthValActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
}  
  
  
private void RunTotalOrdersReportActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
}
```

Final Code

```
printVars();

boolean valid = classes.userSessions.isSessionValid(sessionID);
System.out.println(sessionID);
if (valid == false){
    JOptionPane.showMessageDialog(null, "You have been timed out, please log back in.");
    loginScreen login = new loginScreen();
    login.setDefaultCloseOperation(loginScreen.EXIT_ON_CLOSE);
    login.pack();
    login.setVisible(true);
    this.dispose();
}

String userType;
userType = classes.admin.getCurrentUserType();

if (userType.equals("Admin") || userType.equals("Waiter")){
    System.out.println(userType);
    System.out.println("access granted");

    //have a quick look at the report parameters
    String jTextFieldTOR_startmonthVal = TOR_startmonthVal.getText();
    System.out.println(jTextFieldTOR_startmonthVal);

    String jTextFieldTOR_endmonthVal = TOR_endmonthVal.getText();
    System.out.println(jTextFieldTOR_endmonthVal);

    String jComboBox1_TOR_ValueToShow = TOR_ValueToShow.getSelectedItem().toString();
    System.out.println(jComboBox1_TOR_ValueToShow);
```

Final Code

```
String jComboBox1_TORordervalue = jComboBox1_TORorder.getSelectedItem().toString();
System.out.println(jComboBox1_TORordervalue);

//set some titles for the report components
String frameTitle="TotalOrdersByMonth";
String chartTitle="TotalOrdersByMonthPanelChart";
String tableTitle="TotalOrdersByMonthPanelData";

//new frame to display the report - initialised by the ProductsByCategorySummed_view
method which assembles the chart and the data table into a report

JFrame frame = classes.TotalOrdersByMonth_view.fillTotalOrdersByMonth_viewAssemble(
jTextFieldTOR_startmonthVal, jTextFieldTOR_endmonthVal, jComboBox1_TOR_ValueToShow,
jComboBox1_TORordervalue, 600, 400, 600, 200, frameTitle, chartTitle, tableTitle);

//Display the frame
frame.pack();
frame.setVisible(true);

}

else{
    System.out.println(userType);
    JOptionPane.showMessageDialog(null, "You don't have access to this feature");
}

}
```

6.15 updateCategoryScreen.java

```
public class updateCategoryScreen extends javax.swing.JFrame {

    public int userPrimaryKey;
    public int category_pk;

    /**
     * Creates new form updateProductScreen
     */
    public updateCategoryScreen() {
        initComponents();
    }

    private void updateButtonActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //updates record with text field input from user

        classes.categories category;

        String category_name = nameTextField.getText();
        category = new classes.categories(category_pk, category_name);
        classes.categories.updateCategory(category);
        this.dispose();
    }
}
```

6.16 updateProductScreen.java

```
import java.util.HashMap;

public class updateProductScreen extends javax.swing.JFrame {

    public int userPrimaryKey;
    public int product_pk;

    /**
     * Creates new form updateProductScreen
     */
    public updateProductScreen() {
        initComponents();
        comboBoxAttach();
    }

    public void comboBoxAttach(){
        //adds categories to combo box

        classes.categories category = new classes.categories();
        HashMap<String, Integer> map = category.addToCombo();
        for (String set: map.keySet()){
            categoryComboBox.addItem(set);
        }
    }

    private void updateButtonActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //updates products with user inputs
    }
}
```

Final Code

```
classes.products product;
classes.categories category = new classes.categories();
HashMap<String, Integer> map = category.addToCombo();

String product_name = nameTextField.getText();
String product_description = descriptionTextField.getText();
float product_sellprice;
float product_cost;
int stock;
String categoryName = null;
int categoryfk;

stock = Integer.valueOf(stockTextField.getText());

categoryfk = map.get(categoryComboBox.getSelectedItem().toString());
categoryName = categoryComboBox.getSelectedItem().toString();
product_sellprice = Float.parseFloat(priceTextField.getText());
product_cost = Float.parseFloat(costTextField.getText());

product = new classes.products(product_pk, product_name,
product_description, product_sellprice, product_cost, userPrimaryKey, null, null, false,
categoryName, stock, categoryfk);

classes.products.updateProduct(product);
this.dispose();

}
```

6.17 updateUserScreen.java

```
public class updateUserScreen extends javax.swing.JFrame {  
    public int userPrimaryKey;  
  
    /**  
     * Creates new form newProductScreen  
     */  
    public updateUserScreen() {  
        initComponents();  
    }  
  
    private void updateButtonActionPerformed(java.awt.event.ActionEvent evt) {  
        // TODO add your handling code here:  
        //updates selected user with user inputs  
  
        classes.admin user;  
  
        String username = nameTextField.getText();  
        String password = passwordTextField.getText();  
        String type = typeComboBox.getSelectedItem().toString();  
  
        user = new classes.admin(1, username, password, type);  
  
        classes.admin.updateUser(user);  
        this.dispose();  
    }  
}
```

6.18 userHoursScreen.java

```
import java.util.ArrayList;

public class userHoursScreen extends javax.swing.JFrame {

    /**
     * Creates new form userHoursScreen
     */
    public userHoursScreen() {
        initComponents();
        fillOrderTable();
        setColumnNames();
    }

    public void fillOrderTable(){
        classes.userHours user = new classes.userHours();
        ArrayList <classes.userHours> userList = user.userHoursList();
        String [] columnNames = {"ID", "Name", "Hours", "Year-Month"};
        Object [][] rows = new Object[userList.size()][4];

        for(int i = 0; i<userList.size(); i++){
            rows[i][0] = userList.get(i).getUserPK();
            rows[i][1] = userList.get(i).getUserName();
            rows[i][2] = userList.get(i).getHours();
            rows[i][3] = userList.get(i).getYearMonth();
        }

        classes.table model = new classes.table(rows, columnNames);
    }
}
```

Final Code

```
userHoursTable.setModel(model);  
}  
  
public void setColumnNames(){  
  
    userHoursTable.getColumnModel().getColumn(0).setHeaderValue("ID");  
    userHoursTable.getColumnModel().getColumn(1).setHeaderValue("Name");  
    userHoursTable.getColumnModel().getColumn(2).setHeaderValue("Hours");  
    userHoursTable.getColumnModel().getColumn(3).setHeaderValue("Year-Month");  
  
}
```

6.19 userScreen.java

```
import java.util.ArrayList;
import javax.swing.JOptionPane;

public class userScreen extends javax.swing.JFrame {

    /**
     * Creates new form categoryScreen
     */
    //classes.admin user = new classes.admin();
    public userScreen() {
        initComponents();
        fillTable("");
        setColumnNames();
    }

    public void fillTable(String searchVal){
        //fills table with contents from array list

        classes.admin user = new classes.admin();
        ArrayList <classes.admin> userList = user.userList(searchVal);
        String [] columnNames = {"ID", "Name", "Password", "Type"};
        Object [][] rows = new Object[userList.size()][4];

        for(int i = 0; i<userList.size(); i++){
            rows[i][0] = userList.get(i).getUserPK();
            rows[i][1] = userList.get(i).getUserName();
            rows[i][2] = userList.get(i).getPassword();
            rows[i][3] = userList.get(i).getUserType();
        }
    }
}
```

Final Code

```
}

classes.table model = new classes.table(rows, columnNames);
userTable.setModel(model);

}

public void setColumnNames(){
//sets table column names

userTable.getColumnModel().getColumn(0).setHeaderValue("ID");
userTable.getColumnModel().getColumn(1).setHeaderValue("Name");
userTable.getColumnModel().getColumn(2).setHeaderValue("Password");
userTable.getColumnModel().getColumn(3).setHeaderValue("Type");

}

private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
//creates new newUserScreen

newUserScreen userAdd = new newUserScreen();
userAdd.setDefaultCloseOperation(newUserScreen.DISPOSE_ON_CLOSE);
userAdd.pack();
userAdd.setVisible(true);

}

private void editButtonActionPerformed(java.awt.event.ActionEvent evt) {
// TODO add your handling code here:
//creates new updateUserScreen
```

Final Code

```
//updates user values using inputs from the user

try{
    updateUserScreen userUpdate = new updateUserScreen();
    int rowIndex = userTable.getSelectedRow();
    //updateProductScreen.userPrimaryKey = userPrimaryKey;

    userUpdate.userPrimaryKey = Integer.valueOf(userTable.getValueAt(rowIndex, 0).toString());
    userUpdate.nameTextField.setText(userTable.getValueAt(rowIndex, 1).toString());
    userUpdate.passwordTextField.setText(userTable.getValueAt(rowIndex, 2).toString());
    userUpdate.typeComboBox.setSelectedItem(userTable.getValueAt(rowIndex, 3).toString());

    userUpdate.setDefaultCloseOperation(updateProductScreen.DISPOSE_ON_CLOSE);
    userUpdate.pack();
    userUpdate.setVisible(true);
    //updateProductScreen.userPrimaryKey = userPrimaryKey;
}

catch (Exception ex){
    JOptionPane.showMessageDialog(null, "There was an error, please select a row and try again.");
}

private void refreshButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    //refreshes user table

    fillTable("");
    setColumnNames();
}
```

Final Code

```
private void deleteButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    //deletes selected user record  
  
    int rowIndex = userTable.getSelectedRow();  
    int id = Integer.valueOf(userTable.getValueAt(rowIndex, 0).toString());  
    try{  
        classes.admin.deleteUser(id);  
    }  
    catch(Exception ex){  
        JOptionPane.showMessageDialog(null, "There was an error, please try again.");  
    }  
}  
  
private void searchValueTextFieldActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
  
}  
  
private void searchButtonActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    fillTable(searchValueTextField.getText());  
    setColumnNames();  
  
}
```

6.20 admin.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
 * license
 *
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package classes;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

/**
 *
 * @author garmin
 */
public class admin {

    Connection connection;

    private int user_pk;
    private String username;
    private String password;
    public String type;

    public Integer getUserPK(){
        return user_pk;
    }
}
```

Final Code

```
}

public String getUserName(){
    return username;
}

public String getPassword(){
    return password;
}

public String getUserType(){
    return type;
}

public void setUserPK(int user_PK){
    this.user_pk = user_PK;
}

public void setName(String name){
    this.username = name;
}

public void setPassword(String Password){
    this.password = Password;
}

public void setUserType(String Type){
    this.type = Type;
}

public admin(){}
```

```

public admin(int user_PK, String name, String Password, String Type)

{
    this.user_pk = user_PK;
    this.username = name;
    this.password = Password;
    this.type = Type;
}

public static void addUser(admin user){

    //adds a new user to the users table in the database using inputs from the user

    Connection connect = database.getConnection(); //creates connection to the database
    PreparedStatement ps = null; //declares the pre compiled query
    try{
        //creates the query that is sent to the database
        //? used to label parameters which are set using the getter methods
        ps = connect.prepareStatement("INSERT INTO users(username, password, type)
VALUES(?, ?, ?);");
        ps.setString(1, user.getUserName());
        ps.setString(2, user.getPassword());
        ps.setString(3, user.getUserType());

        //executeUpdate() returns the amount of records affected by the sql statement
        //if anything but 0 records are affected then atleast 1 record was affected so the method was
        succesfull
        //if 0 records are affected then there was an error
        //if method succesfull a window pops up with the message "new user added"
        //if there was an error a window pops up with the message "error"
        if(ps.executeUpdate() != 0){
            JOptionPane.showMessageDialog(null, "New User Added");
        }
    }
}

```

Final Code

```
else{
    JOptionPane.showMessageDialog(null, "Error");
}

}

//catches any sql exceptions thrown

catch (SQLException ex) {
    Logger.getLogger(admin.class.getName()).log(Level.SEVERE, null, ex);
}

}

public ArrayList<admin> userList (String searchVal){
    //string parameter used to pass in the search value in the search box

    //creates an ArrayList of type admin, so it can hold objects of the admin class, ArrayList is
    named userList

    ArrayList<admin> userList = new ArrayList<>();

    //creates connection to the database
    connection = database.getConnection();
    PreparedStatement ps;
    ResultSet rs;

    //creates query that will be sent to the database
    String query = "SELECT * FROM users WHERE username LIKE ?;";

    try{
        //sends the query to the database
        ps = connection.prepareStatement(query);
        //sets the ? parameter to the search value passed in
        ps.setString(1, "%" + searchVal + "%");
    }
}
```

Final Code

```
//initializes result set to the value returned from executing the prepared statement query
rs = ps.executeQuery();

admin user;

//loops through the data in the result set
//creates a new instance of the admin class called user
//user takes the values of the result set
//adds user to the ArrayList userList
while (rs.next()){

    user = new admin(rs.getInt("user_pk"), rs.getString("username"), rs.getString("password"),
rs.getString("type"));

    userList.add(user);

}

}

//catches any sql exceptions thrown
catch (SQLException ex){

    Logger.getLogger(admin.class.getName()).log(Level.SEVERE, null, ex);
}

return userList;
}

public static void deleteUser(int userID){

    //integer parameter passes in the primary key of the selected user

    //creates connection to the database
    Connection connection = database.getConnection();
    PreparedStatement ps;

    try{
        //creates prepared statement with the query and send it to the database

```

Final Code

```
ps = connection.prepareStatement("DELETE FROM users WHERE user_pk = ?;");

//sets ? parameter to userID

ps.setInt(1, userID);

//executeUpdate() returns the number of records affected by the query

//if 0 is returned there was an error so a window pops up with an error message

//if anything but 0 is returned then its a success so a window pops up with an appropriate
message

if (ps.executeUpdate() != 0){

    JOptionPane.showMessageDialog(null, "User has been deleted.");

}

else{

    JOptionPane.showMessageDialog(null, "There was an error, please try again.");

}

//cathes any sql exceptions thrown

catch(SQLException ex){

    Logger.getLogger(admin.class.getName()).log(Level.SEVERE, null, ex);

}

}

public static String getCurrentUserType(){

    //creates connection to the database

    Connection connection = database.getConnection();

    PreparedStatement ps = null;

    ResultSet rs;

    int userpk = 0;

    String userType = null;

    //intializes query variable to the query written
```

Final Code

```
//the query returns the primary key of the user for the latest usersession record
//initializes the prepared statement to the value of the query
try{
    String query = "SELECT user_fk FK FROM usersessions WHERE usersession_pk = ( SELECT
MAX(usersession_pk) FROM usersessions);";
    ps = connection.prepareStatement(query);
    rs = ps.executeQuery();
    //loops through the result set
    //sets userpk to the value returned from the query
    while (rs.next()){
        userpk = rs.getInt("FK");
    }
}
//catches any sql exceptions thrown
catch(SQLException ex){
    Logger.getLogger(admin.class.getName()).log(Level.SEVERE, null, ex);
}

//initializes prepared statement as the query written in the brackets
//sets the ? parameter to the user primary key for the latest session
//initializes result set to the value returned from executing the prepared statement query
try{
    ps = connection.prepareStatement("SELECT type FROM users WHERE user_pk = ?");
    ps.setInt(1, userpk);
    rs = ps.executeQuery();
    while (rs.next()){
        userType = rs.getString("type");
    }
}
//catches any sql exceptions thrown
catch(SQLException ex){
```

Final Code

```
Logger.getLogger(admin.class.getName()).log(Level.SEVERE, null, ex);
}

//returns the retrieved userType
return userType;
}

public static void updateUser(admin user){

    //creates connection to the database
    Connection connection = database.getConnection();
    PreparedStatement ps;

    //initializes prepared statement to the query written in the brackets
    //sets the ? parameters to the value of the appropriate getter method
    try{
        ps = connection.prepareStatement("UPDATE users SET username = ?, password = ?, type = ?
WHERE user_pk = ?");

        ps.setString(1, user.getUserName());
        ps.setString(2, user.getPassword());
        ps.setString(3, user.getUserType());
        ps.setInt(4, user.getUserPK());

        //executeUpdate() returns the number of records affected by the query
        //if 0 records affected an error message window pops up
        //if anything but 0 records were affected a success message window pops up
        if(ps.executeUpdate() != 0){
            JOptionPane.showMessageDialog(null, "User updated.");
        }
        else{
            JOptionPane.showMessageDialog(null, "There was an error, please try again.");
        }
    }
}
```

Final Code

```
    }  
}  
//catches any exceptions thrown  
catch(SQLException ex){  
    Logger.getLogger(admin.class.getName()).log(Level.SEVERE, null, ex);  
}  
}  
  
}
```

6.21 categories.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
 * license
 *
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package classes;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import java.sql.Statement;

/**
 *
 *
 * @author garmin
 */
public class categories {

    Connection connection;

    private int category_pk;
    private String category_name;

    public Integer getCatPK(){


```

Final Code

```
return category_pk;
}

public String getName(){
    return category_name;
}

public void setCatPK(int catPK){
    this.category_pk = catPK;
}

public void setName(String name){
    this.category_name = name;
}

public categories(){}
public categories(int pk, String name)
{
    this.category_pk = pk;
    this.category_name = name;
}

public static void addCategory(categories category){

    //creates connection to the database
    Connection connect = database.getConnection();
    PreparedStatement ps = null;

    //initializes prepraed statement to the query in the brackets
}
```

Final Code

```
//sets ? parameter in the query to the value return from the getName() method on the category passed in
```

```
try{  
    ps = connect.prepareStatement("INSERT INTO categories(category_name) VALUES(?)");  
    ps.setString(1, category.getName());  
  
    //executeUpdate() returns the number of records affected  
    //if 0 is returned an error message window pops up  
    //if anything but 0 is returned a success message window pops up  
    if(ps.executeUpdate() != 0){  
        JOptionPane.showMessageDialog(null, "New Category Added");  
    }  
    else{  
        JOptionPane.showMessageDialog(null, "Error");  
    }  
}  
//catches exceptions thrown  
catch (SQLException ex) {  
    Logger.getLogger(categories.class.getName()).log(Level.SEVERE, null, ex);  
}  
}
```

```
public ArrayList<categories> categoryList (String searchVal){
```

```
    //String parameter used to pass in the search value
```

```
    //creates new ArrayList of type categories called catList
```

```
    ArrayList<categories> catList = new ArrayList<>();
```

```
    //creates connection to the database
```

```
    connection = database.getConnection();
```

Final Code

```
PreparedStatement ps;
ResultSet rs;

//initializes query
String query = "SELECT * FROM categories WHERE deletedflag = 0 AND category_name LIKE ?;";

//initializes prepared statement to the query and sends it to the database
//sets ? parameter in the query to the search value passed in
//initializes result set to the values returned from executing the query
try{
    ps = connection.prepareStatement(query);
    ps.setString(1, "%" + searchVal + "%");
    rs = ps.executeQuery();
    categories category;

    //loops through the result set
    //instantiates new objects of the categories class with the values in the result set
    //adds each object to the ArrayList catList
    while (rs.next()){
        category = new categories(rs.getInt("category_pk"), rs.getString("category_name"));

        catList.add(category);
    }
}

//catches exceptions thrown
catch (SQLException ex){
    Logger.getLogger(categories.class.getName()).log(Level.SEVERE, null, ex);
}

return catList;
}
```

Final Code

```
public static void deleteCategory(int catID){
    //parameter passes in the selected category primary key

    //creates connection to the database
    Connection connection = database.getConnection();
    PreparedStatement ps;

    //initializes prepared statement to the query in the brackets and sends it to the database
    //sets ? parameters to category primary key passed in
    try{
        ps = connection.prepareStatement("UPDATE categories SET deletedflag = 1 WHERE
category_pk = ?;");
        ps.setInt(1, catID);
        //executeUpdate() returns number of records affected by the query
        //if 0 is returned, an error message window pops up
        //if anything but 0 is returned, a success message window pops up
        if (ps.executeUpdate() != 0){
            JOptionPane.showMessageDialog(null, "Category has been deleted.");
        }
        else{
            JOptionPane.showMessageDialog(null, "There was an error, please try again.");
        }
    }
    //catches exceptions thrown
    catch(SQLException ex){
        Logger.getLogger(categories.class.getName()).log(Level.SEVERE, null, ex);
    }
}

public static void updateCategory(categories category){
```

Final Code

```
//creates connection to the database
Connection connection = database.getConnection();

PreparedStatement ps;

//initializes prepared statement to the query in the brackets and sends it to the database
//sets each ? parameter in the query to the appropriate getter method
try{
    ps = connection.prepareStatement("UPDATE categories SET category_name = ? WHERE
category_pk = ?");
    ps.setString(1, category.getName());
    ps.setInt(2, category.getCatPK());

    //executeUpdate() returns the number of records affected by the query
    //if 0 is returned, an error message window pops up
    //if anything but 0 is returned, a success message window pops up
    if(ps.executeUpdate() != 0){
        JOptionPane.showMessageDialog(null, "Category updated.");
    }
    else{
        JOptionPane.showMessageDialog(null, "There was an error, please try again.");
    }
}

//catches exceptions thrown
catch(SQLException ex){
    Logger.getLogger(categories.class.getName()).log(Level.SEVERE, null, ex);
}

}

public HashMap<String, Integer> addToCombo(){

//creates new HashMap called map that stores string keys and integer values
```

Final Code

```
//creates connection to the database
HashMap<String, Integer> map = new HashMap<>();
connection = database.getConnection();

Statement st;
ResultSet rs;

//creates a statement object used to send sql statement to the database
//initializes result set to the values returned from executing the query in the brackets
try{
    st = connection.createStatement();

    rs = st.executeQuery("SELECT category_pk, category_name FROM categories WHERE
deletedflag = 0");

    categories category;

    //loops through the result set
    //instantiates new objects of the categories class with the values from the result set
    //adds the new instances to the hashmap map using getter methods
    while(rs.next()){

        category = new categories(rs.getInt(1), rs.getString(2));

        map.put(category.getName(), category.getCatPK());
    }

    //catches exceptions thrown
    catch(SQLException ex){

        Logger.getLogger(categories.class.getName()).log(Level.SEVERE, null, ex);
    }

    //returns the map with all the categories in it
    return map;
}
```

Final Code

}

6.22 clockin.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
 * license
 *
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package classes;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.sql.ResultSet;
import javax.swing.JOptionPane;

/***
 *
 * @author garmin
 */
public class clockin {

    public static void clockUserIn(int userPK){
        //passes in the primary key of the user

        //creates connection to the database
        Connection connection = database.getConnection();
        PreparedStatement ps;
        ResultSet rs;

        //initializes prepared statement to the query in the brackets and sends it to the database
        try{
```

Final Code

```
ps = connection.prepareStatement("INSERT INTO dailyhours (clockin, user_fk) VALUES  
(sysdate(),?);");  
  
ps.setInt(1, userPK);  
  
//executeUpdate() returns the number of records affected by the query  
//if 0 is returned an error message window pops up  
//if anything but 0 is returned a success message window pops up  
if (ps.executeUpdate() != 0){  
    JOptionPane.showMessageDialog(null, "You are clocked in");  
}  
else{  
    JOptionPane.showMessageDialog(null, "Error");  
}  
}  
  
//catches sql exceptions thrown  
catch(SQLException ex){  
    Logger.getLogger(clockin.class.getName()).log(Level.SEVERE, null, ex);  
}  
  
}  
  
}  
  
public static void clockUserOut(int userPK){  
    //passes in the primary key of the user  
  
    //creates connection to the database  
    Connection connection = database.getConnection();  
    PreparedStatement ps;  
  
    //initializes prepared statement to the query in the brackets and sends it to the database  
    //sets ? parameter to the user primary key passed in  
    try{
```

Final Code

```
ps = connection.prepareStatement("UPDATE dailyhours SET clockout = sysdate() WHERE user_fk = ?");  
ps.setInt(1, userPK);  
  
//executeUpdate() returns the number of records affected by the prepared statement query  
//if 0 is returned an error message window pops up  
//if anything but 0 is returned a success message window pops up  
if (ps.executeUpdate() != 0){  
    JOptionPane.showMessageDialog(null, "You have been clocked out.");  
}  
else{  
    JOptionPane.showMessageDialog(null, "There was an error.");  
}  
}  
  
//catches sql exceptions thrown  
catch(SQLException ex){  
    Logger.getLogger(clockin.class.getName()).log(Level.SEVERE, null, ex);  
}  
  
}  
  
}  
  
public static void calculateUserHours(int userPK){  
    //passes in primary key of user  
  
    //creates connection to the database  
    Connection connection = database.getConnection();  
    PreparedStatement ps;  
  
    //initializes prepared statement to the query in the brackets and sends it to the database  
    //sets ? parameter to the user primary key passed in
```

Final Code

```
//executes the query
try{
    ps = connection.prepareStatement("update dailyhours set hours =
round(TIMESTAMPDIFF(SECOND, clockin, clockout)/3600, 2) where user_fk = ?");
    ps.setInt(1, userPK);
    ps.executeUpdate();
}

//catches sql exceptions thrown
catch(SQLException ex){
    Logger.getLogger(clockin.class.getName()).log(Level.SEVERE, null, ex);
}

}

}
```

6.23 dataValidation.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
 * license
 *
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package classes;

import javax.swing.JOptionPane;

/**
 *
 * @author garmin
 */

//dataValidation.typeIsDecimal()

public class dataValidation {

    //I put these so my return values from the validation code is really obvious
    final static boolean dataIsInvalid = false;
    final static boolean dataIsValid = true;

    //this is my variables that I set from the JField focus so each time
    //the system knows what datatype to validate for the field
    public static String myDataType = "notset";
    public static int myPrecision = 0;
    public static int myScale = 0;
    public static int myLength = 0;
    public static boolean myNull=false;
}
```

Final Code

```
//methods to be called when a JField gets focus so the inputVerifier can use it to validate the
JField.value

//these set the data type variables

public static void typeIsDecimal(int inPrecision, int inScale)

{
    myDataType="Decimal";
    myPrecision=inPrecision;
    myScale=inScale;
}

public static void typeIsInteger(int inLength)

{
    myDataType="Integer";
    myLength=inLength;
}

public static void typeIsString(int inLength)

{
    myDataType="String";
    myLength=inLength;
}

public static void typeIsDate()

{
    myDataType="Date";
}

public static void typeIsNull(boolean inNull)

{
    myNull=inNull;
}

//method to validate the data called by the dataVerifier
//I check first if its a database or a static validation using the name of the object
```

Final Code

```
//I named the object like static_ or data_ and then the data type or the database column details  
after  
  
//like this products_product_name.setName("st@tic_n0tnu11_char_100");  
  
public static Boolean dataIsValid(String dataIsFrom, String dataToValidate)  
{  
  
    //if the data validate type is static call the static checker st@tic_  
    //else call the databasechecker d@ta_  
  
    if (dataIsFrom.contains("st@tic_")) {  
  
        return dataIsValidStatic(dataIsFrom,dataToValidate);  
    } else if (dataIsFrom.contains("d@ta_")){  
  
        return dataIsValidDb(dataIsFrom,dataToValidate);  
    }  
  
    //if the object name isn't static or database then just return true without validating  
    return dataIsValid;  
}  
  
//validate the data in the static way without using database data types  
  
private static Boolean dataIsValidStatic(String dataIsFrom, String dataToValidate)  
{  
  
    System.out.println(dataIsFrom);  
    System.out.println(dataToValidate);  
  
    //I check if it is allowed to be null first  
    //_n0tnu11_  
  
    if (dataIsFrom.contains("_n0tnu11_")) {  
  
        if (dataToValidate.isBlank()) {  
  
            return dataIsInvalid;  
        }  
    }  
}
```

Final Code

```
return true;  
}  
  
//validate the data by using database data types  
//I didnt finish this  
private static Boolean dataIsValidDb(String dataIsFrom, String dataToValidate)  
{  
    System.out.println(dataIsFrom);  
    System.out.println(dataToValidate);  
    //I was going to use a query like this and work out the table and column from the dataIsFrom  
    //to get the data type and then use that to validate the field  
    //but its quite complicated however I think it would work OK if I finished it  
    //SELECT  
    //table_name,  
    //column_name,  
    //data_type,  
    //character_maximum_length,  
    //numeric_precision,  
    //numeric_scale,  
    //is_nullable  
    //FROM information_schema.columns  
    //WHERE table_schema="posplex"  
    //AND  
    //table_name="products"  
    //ORDER BY table_name, ordinal_position asc;  
    //products product_pk      int          10      0      NO  
    //products product_name  varchar 100                  NO  
    //products product_description  varchar 255          NO  
    //products product_sellprice   decimal        13      2      NO
```

Final Code

```
//products product_cost    decimal          13   2    NO
//products createdby_fk    int              10   0    NO
//products createdon        datetime
//products lastupdated      datetime
//products deletedflag      tinyint         3    0    YES
//products category         varchar(100)
//products stock             int              10   0    NO
//products category_fk      int              10   0    NO

return dataIsValid;
}

//method to validate the data called by the dataVerifier
//I check first if it is a database or a static validation using the name of the object
public static Boolean dataIsValid2(String dataIsFrom, String dataToValidate)
{
    boolean validStatus=false;
    //first I check if the data can be null if its null I send an error to stop
    if (dataToValidate.isBlank() & myNull==false) {
        validStatus=false;
        validationErrorMessage( dataIsFrom, dataToValidate, "Cannot be null");
    }
    else //if it passed the null test now I test the data type
    {
        //case to check what the datatype is set by the Jfield event focus
        switch (myDataType)
        {
            case "Decimal":
                String[] arrayDigits = dataToValidate.split("\\.", 2);
                System.out.println(arrayDigits[0]);
        }
    }
}
```

Final Code

```
System.out.println(arrayDigits[1]);

    if (String.valueOf(arrayDigits[0]).length() >
myPrecision || String.valueOf(arrayDigits[1]).length() > myScale) {

        validStatus=false;

        validationErrorMessage( datasFrom, dataToValidate, "Decimal only allows
"+myPrecision+" significant digits and "+myScale+" decimal places");

    } else {

        //if the significnat and decimal places are OK then return true for valid

        validStatus=true;

    }

    System.out.println(myDataType);

    break;

case "Integer":

    //try and convert the value to an int

    //if it fails send an error and return false to say its invalid

    try {

        int i = Integer.parseInt(dataToValidate);

        //if it is an int then see how long and if longer than the length set for that field send an
error

        if (String.valueOf(dataToValidate).length() > myLength){

            validStatus=false;

            validationErrorMessage( datasFrom, dataToValidate, "Integer is too long");

        } else {

            //if the int and the length are ok then return true for valid

            validStatus=true;

        }

    } catch (NumberFormatException ex) {

        validStatus=false;

        validationErrorMessage( datasFrom, dataToValidate, "Is not a valid integer");

    }

    System.out.println(myDataType);

    break;
```

Final Code

```
case "String":  
    //its a string so check the length of the characters see if its too big for the data  
    if (String.valueOf(dataToValidate).length() > myLength){  
        validStatus=false;  
        validationErrorMessage( datalsFrom, dataToValidate, "Too many characters");  
    } else {  
        //if the length is ok then return true for valid  
        validStatus=true;  
    }  
    System.out.println(myDataType);  
    validStatus=true;  
    break;  
  
case "Date":  
    //if its a date use a regular expression to see if its DD-MM-YYYY format else invalid  
    String regex="(0[1-9] | [12][0-9]|3[01])[- /.](0[1-9]|1[012])[- /.](19|20)\\d\\d";  
    if (!!(dataToValidate.matches(regex))) {  
        validStatus=false;  
        validationErrorMessage( datalsFrom, dataToValidate, "Not a valid date DD-MM-  
        YYYY");  
    } else {  
        //if the date is ok then return true for valid  
        validStatus=true;  
    }  
    System.out.println(myDataType);  
    validStatus=true;  
    break;  
  
default:  
    validStatus=true;  
}
```

Final Code

```
//now I have done the validation for this field I reset the defaults back for the next one
resetDefaults();

//and return the validStatus that is true or false
return validStatus;

}

//this just resets the field type variables to the default
private static void resetDefaults(){

    myDataType = "notset";
    myPrecision = 0;
    myScale = 0;
    myLength = 0;
    myNull=false;
}

//this is my method to put an error message it shows the problem and where the data is from and
what the data is
private static void validationErrorMessage(String dataIsFrom, String dataToValidate, String
errMsg){

    String finalMsg=errMsg +" : "+dataIsFrom+" contains bad data "+dataToValidate;
    JOptionPane.showMessageDialog(null, finalMsg,"Please correct the data.",
    JOptionPane.ERROR_MESSAGE);
}

}
```

6.24 dataVerifier.java

```
package classes;

import javax.swing.InputVerifier;
import javax.swing.JComponent;
import javax.swing.JTextField;

public class dataVerifier extends InputVerifier {

    @Override
    public boolean verify(JComponent input) {

        //the verify method gets the JComponent passed in and this is the text field from the screen
        //that I am validating
        //this will allow me to see where the data came from and what the data is
        JTextField componentToValidate=(JTextField)input;
        String dataIsFrom=componentToValidate.getName();
        String dataToValidate=componentToValidate.getText();

        //output where the data came from and what it is so I can see it works
        System.out.println(dataIsFrom);
        System.out.println(dataToValidate);

        //return true or false depends on my validation of the data in my dataValidation class
        //return dataValidation.dataIsValid(dataIsFrom,dataToValidate);
        return classes.dataValidation.dataIsValid2(dataIsFrom,dataToValidate);

    }
}
```

6.25 database.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
 * license
 *
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package classes;

import java.sql.Connection;
import java.sql.DriverManager;

/**
 *
 * @author garmin
 */
public class database {

    //initializes variables for the database name, username and password
    private static String databaseName = "posplex";
    private static String username = "root";
    private static String password = "Forgetit";

    //initializes connection as null
    static Connection connection=null;

    //if connection is not null so there is a connection, it returns the connection
    //else if the connectino is null, it returns the value of the other getConnection method with
    //the database name, username and password values passed in
    public static Connection getConnection()

    {
        if (connection != null){


```

Final Code

```
    return connection;
}

return getConnection(databaseName, username, password);
}

private static Connection getConnection(String databaseName, String userName, String password)
    //passes in the database name, username and password
{
    try
    {
        //returns com.mysql.cj.jdbc.Driver.class
        Class.forName("com.mysql.cj.jdbc.Driver");
        //creates a connection to the database

        connection=DriverManager.getConnection("jdbc:mysql://localhost:3306/" +databaseName+"?user="
+userName+"&password="+password);

        System.out.println("connected");
    }
    //catches exceptions
    catch(Exception e)
    {
        e.printStackTrace();
    }

    return connection;
}

}
```

6.26 mostSoldByMonth_view.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
 * license
 *
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package classes;

import java.awt.Color;
import java.awt.ComponentOrientation;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTable;
import javax.swing.border.TitledBorder;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartFrame;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
```

Final Code

```
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;

/**
 *
 * @author garmin
 */
public class mostSoldByMonth_view {

    Connection connection;

    //I added these as they are used by addComponentsToPane which is the GridBagLayout layout
    //manager to dynamically create and position the chart and the data table in Swing
    final static boolean shouldFill = false;
    final static boolean shouldWeightX = true;
    final static boolean RIGHT_TO_LEFT = false;

    //MostSoldByMonth_view attributes like the columns in the view same as those
    private String mymonth;
    private String product;
    private int total_quantity_sold;

    //methods to return the values of each attribute
    public String getmymonth(){
        return mymonth;
    }
    public String getproduct(){
        return product;
    }
}
```

Final Code

```
public int gettotal_quantity_sold(){

    return total_quantity_sold;

}

public mostSoldByMonth_view(){}

//MostSoldByMonth_view row of data object - used to create an array of rows of data - used to
hold the results of a dataset queried from the view

public mostSoldByMonth_view(String mymonth, String product, int total_quantity_sold)

{

    this.mymonth = mymonth;
    this.product = product;
    this.total_quantity_sold = total_quantity_sold;
}

//private method just to make the SQL for my report mostSoldByMonth_view
//I did this because I can't bind the order by field name in a prepare statement
//so I had to build the order clause onto the SQL like this

private static String MostSoldByMonth_viewSQL(String orderbyVal)

{

    //query to retrieve data from the db view where the category is like the report param passed in
    val

    String query = "SELECT * FROM MostSoldByMonth_view WHERE mymonth BETWEEN ? AND ?
    ORDER BY ";

    //append the correct ORDER BY column for the query based on the orderbyVal param
    //note this can in some cases be a risk for SQL injection but PreparedStatement doesn't allow
    ORDER BY ?

    if (orderbyVal=="mymonth") {

        query=query+"mymonth;";

    } else if (orderbyVal=="product") {
```

Final Code

```
query=query+"product;";

} else if (orderbyVal=="total_quantity_sold") {

    query=query+"total_quantity_sold;";

} else {

    query=query+"mymonth;"; //remove SQL injection risk by examining the orderbyVal and only
allowing mymonth column name

}

return query;

}

//this ArrayList class is a resizable array, which is present in the java. util package from the docs
// built-in arrays have a fixed size, ArrayLists can change their size dynamically so
//I used it to retrieve the data from the database view so it can be shown as the data table in the
report

public ArrayList<mostSoldByMonth_view> MostSoldByMonth_viewList (String startDate, String
endDate, String orderbyVal){

    //create arraylist of ProductsByCategorySummed_view rows
    ArrayList<mostSoldByMonth_view> MostSoldByMonth_viewList = new ArrayList<>();

    //get a db connection and make the resultset to hold the db data
    connection = database.getConnection();
    ResultSet rs;

    //query to retrieve data from the db view where the category is like the report param passed in
val
    String query = MostSoldByMonth_viewSQL(orderbyVal);

    //its the same as when I get data for a Jtable in any of the screens
    //connect the database and run the query and loop through the records
    //and add them to the list
```

Final Code

```
try{
    connection = database.getConnection();

    //PreparedStatement ps from the query with the categoryId like %val%
    PreparedStatement ps = connection.prepareStatement(query);
    ps.setString(1, startDate);
    ps.setString(2, endDate);

    rs = ps.executeQuery();

    //create the view data row and loop through the data in the SQL and add each row to the
    arraylist
    mostSoldByMonth_view MostSoldByMonth_viewRow;
    while (rs.next()){

        MostSoldByMonth_viewRow = new mostSoldByMonth_view(rs.getString("mymonth"),
        rs.getString("product"), rs.getInt("total_quantity_sold"));

        MostSoldByMonth_viewList.add(MostSoldByMonth_viewRow);
    }
}

catch (SQLException ex){
    Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
}

//return the dataset in the arraylist
return MostSoldByMonth_viewList;
}

//the simple ProfitByMonth report which is just the chart returned in a frame to be popped up
public static ChartFrame MostSoldByMonth(String startDate, String endDate, String orderbyVal)
{
```

Final Code

```
Connection connect = database.getConnection();

PreparedStatement ps;
ResultSet rs;

String mymonth="";
String product="";
int total_quantity_sold=0;

//query to retrieve data from the db view where the category is like the report param passed in
val
String query = MostSoldByMonth_viewSQL(orderbyVal);

//JFreeChart dataset object to pass the data into the createBarChart method
DefaultCategoryDataset dataset = new DefaultCategoryDataset();

try{
    ps = connect.prepareStatement(query);

    ps.setString(1, startDate);
    ps.setString(2, endDate);

    //debug code to see the prepared statement
    System.out.println(ps);

    rs = ps.executeQuery();

    //populate the DefaultCategoryDataset
    while (rs.next()){

        mymonth = rs.getString("mymonth");
```

Final Code

```
product = rs.getString("product");

total_quantity_sold = rs.getInt("total_quantity_sold");

dataset.setValue(total_quantity_sold, product, mymonth);

}

}

catch (SQLException ex){

    Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);

}

//make the chart I used a stacked bar chart to show the total quantity sold for each product

JFreeChart chart = ChartFactory.createStackedBarChart("MostSoldByMonth -
"+total_quantity_sold","mymonth","total_quantity_sold",dataset,PlotOrientation.VERTICAL,false,true,false);

//set some chart attributes

CategoryPlot p = chart.getCategoryPlot();

p.setRangeGridlinePaint(Color.black);

//create a frame with the chart in it using JFree ChartFrame type

//public class ChartFrame extends the swing JFrame

ChartFrame frame = new ChartFrame("MostSoldByMonth - "+total_quantity_sold,chart);

return frame;

}

//create the chart but return it in panel so it can be assembled into the final report

public static ChartPanel MostSoldByMonthPanel(String startDate, String endDate, String metricIn,
String orderbyVal)

{

    Connection connect = database.getConnection();

    PreparedStatement ps;
```

Final Code

```
ResultSet rs;

String mymonth="";
String product="";
float metric=0;

//query to retrieve data from the db view where the category is like the report param passed in
val

String query = MostSoldByMonth_viewSQL(orderbyVal);

DefaultCategoryDataset dataset = new DefaultCategoryDataset();

try{
    ps = connect.prepareStatement(query);

    ps.setString(1, startDate);
    ps.setString(2, endDate);

    System.out.println(ps);

    rs = ps.executeQuery();

    while (rs.next()){

        mymonth = rs.getString("mymonth");
        product = rs.getString("product");
        metric = rs.getFloat(metricIn);
        dataset.setValue(metric, product, mymonth);
    }
}

catch (SQLException ex){
```

Final Code

```
Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);

}

//make the chart, set the attributes, return the panel to the assembler
JFreeChart chart = ChartFactory.createStackedBarChart("MostSoldByMonth -
"+metricIn,"mymonth",metricIn,dataset,PlotOrientation.VERTICAL,false,true,false);

CategoryPlot p = chart.getCategoryPlot();
p.setRangeGridlinePaint(Color.black);

ChartPanel panel = new ChartPanel(chart);

return panel;

}

//fills and returns the jTable with contents of arrayList so it can be added to the data panel
public static JTable fillMostSoldByMonth_viewTable(String startDate, String endDate, String
orderbyVal){

    classes.mostSoldByMonth_view MostSoldByMonth_view = new
    classes.mostSoldByMonth_view();

    ArrayList <classes.mostSoldByMonth_view> MostSoldByMonth_viewList =
    MostSoldByMonth_view.MostSoldByMonth_viewList(startDate, endDate, orderbyVal);

    String [] columnNames = {"mymonth","product","total_quantity_sold"};
    Object [][] rows = new Object[MostSoldByMonth_viewList.size()][3];

    for(int i = 0; i<MostSoldByMonth_viewList.size(); i++){
        rows[i][0] = MostSoldByMonth_viewList.get(i).getmymonth();
        rows[i][1] = MostSoldByMonth_viewList.get(i).getproduct();
        rows[i][2] = MostSoldByMonth_viewList.get(i).gettotal_quantity_sold();
    }
}
```

Final Code

```
}

classes.table model = new classes.table(rows, columnNames);

JTable MostSoldByMonth_viewTable = new JTable();

MostSoldByMonth_viewTable.setModel(model);

//sets the jTable column names

//bug not working right now, table has no column headers when displayed

MostSoldByMonth_viewTable.getColumnModel().getColumn(0).setHeaderValue("mymonth");

MostSoldByMonth_viewTable.getColumnModel().getColumn(1).setHeaderValue("product");

MostSoldByMonth_viewTable.getColumnModel().getColumn(2).setHeaderValue("total_quantity_sold");

//return the table of data so it can be laid out in the assembler

return MostSoldByMonth_viewTable;

}

//assembler for the report - this gets the chart and the data table and lays them out using
GridBagLayout layout manager

public static JFrame fillMostSoldByMonth_viewAssemble(String startDate, String endDate, String
metricIn, String orderbyVal, int chartWidth, int chartHeight, int tableWidth, int tableHeight, String
frameTitle, String chartTitle, String tableTitle)

{
    //make the chart

    ChartPanel chartpanel = classes.mostSoldByMonth_view.MostSoldByMonthPanel(startDate,
endDate, metricIn, orderbyVal);

    //set the size of the chart

    chartpanel.setPreferredSize(new Dimension(chartWidth, chartHeight));

    JPanel jPanelChart = new JPanel();

    jPanelChart.add(chartpanel);
```

Final Code

```
jPanelChart.setBorder(BorderFactory.createTitledBorder( BorderFactory.createEtchedBorder(),
chartTitle, TitledBorder.CENTER, TitledBorder.TOP));

//make the data table

JTable jTableData = classes.mostSoldByMonth_view.fillMostSoldByMonth_viewTable(startDate,
endDate, orderbyVal);

//set the size of the table

jTableData.setPreferredSize(new Dimension(tableWidth, tableHeight));

JPanel jPanelData = new JPanel();

jPanelData.add(jTableData);

jPanelData.setBorder(BorderFactory.createTitledBorder( BorderFactory.createEtchedBorder(),
tableTitle, TitledBorder.CENTER, TitledBorder.TOP));

//new frame to show the GridBagLayoutDemo for the POS reports

JFrame frame = new JFrame(frameTitle);

//Set up the content pane by adding the chart panel and the data panel

//GridBagLayout manager time!

classes.mostSoldByMonth_view.addComponentsToPane(frame.getContentPane(), jPanelChart,
jPanelData);

//return the assembled report

return frame;

}

//I used GridBagLayout manager as I need to dynamically lay out the chart and the data table

//if I didn't use this layout manager then its really hard to make the JComponents go where I need

//they just overlap themselves

public static void addComponentsToPane(Container pane, JPanel chartpanel, JPanel datapanel) {

    if (RIGHT_TO_LEFT) {
```

Final Code

Final Code

}

6.27 orderProducts.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
 * license
 *
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package classes;

import java.sql.Date;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

/**
 *
 * @author garmin
 */
public class orderProducts {

    Connection connection;

    private int product_pk;
    private int order_pk;
    private String product_name;
    private float product_sellprice;
    private float product_cost;
    private int createdby_fk;
```

Final Code

```
private Date createdon;  
private int quantity;  
  
public Integer getProductPK(){  
    return product_pk;  
}  
  
public Integer getOrderPK(){  
    return order_pk;  
}  
  
public String getProductName(){  
    return product_name;  
}  
  
public float getPrice(){  
    return product_sellprice;  
}  
  
public float getCost(){  
    return product_cost;  
}  
  
public Integer getFK(){  
    return createdby_fk;  
}  
  
public Date getDateCreated(){  
    return createdon;  
}
```

Final Code

```
public Integer getQuantity(){
    return quantity;
}

public void setProductPK(int productPK){
    this.product_pk = productPK;
}

public void setOrderPK(int orderPK){
    this.order_pk = orderPK;
}

public void setName(String name){
    this.product_name = name;
}

public void setPrice(float price){
    this.product_sellprice = price;
}

public void setCost(float cost){
    this.product_cost = cost;
}

public void setQuantity(int quantity){
    this.quantity = quantity;
}

public orderProducts(){}
```

Final Code

```
public orderProducts(Integer pk, String name, int quantity, float price){

    this.product_pk = pk;
    this.product_name = name;
    this.quantity = quantity;
    this.product_sellprice = price;

}

public ArrayList<orderProducts> productInOrderList(int orderPK){

    //passes in the order primary key

    //creates new ArrayList of type orderProducts called productOrderList
    ArrayList<orderProducts> productOrderList = new ArrayList<>();

    //creates a new connection
    connection = database.getConnection();
    PreparedStatement ps;
    ResultSet rs;
    String query;

    //initializes query
    query = "select product_pk, product_name, product_sellprice, quantity from
products,orderproducts where product_pk = orderproducts.product_fk and order_fk = ?";

    //initializes prepared statement to the query and sends it to the database
    //sets ? parameter to the order primary key
    //initializes result set to the data from the prepared statement query
    try{
        ps = connection.prepareStatement(query);
```

Final Code

```
ps.setInt(1, orderPK);
rs = ps.executeQuery();
orderProducts orderProduct;

//loops through the result set
//creates new instance of the orderProducts class with the values from the result set data
//add each object to the ArrayList productOrderList
while(rs.next()){
    orderProduct = new orderProducts(rs.getInt("product_pk"), rs.getString("product_name"),
rs.getInt("quantity"), (rs.getFloat("product_sellprice")*rs.getInt("quantity")));
    productOrderList.add(orderProduct);
}

//catches exceptions thrown
catch(SQLException ex){
    Logger.getLogger(orderProducts.class.getName()).log(Level.SEVERE, null, ex);
}

//returns the array list
return productOrderList;
}

public static void addOrderProduct(int quantity, int productPK, int orderPK){
    //pass in the product primary key, order primary key, quantity of the product

    //creates connection to the database
    Connection connection = database.getConnection();
    PreparedStatement ps;
    String query;

    //initializes query
    query = "INSERT INTO orderproducts(order_fk, product_fk, quantity) VALUES(?, ?, ?)";

    ps = connection.prepareStatement(query);
    ps.setInt(1, orderPK);
    ps.setInt(2, productPK);
    ps.setInt(3, quantity);
    ps.executeUpdate();
}
```

Final Code

```
//initializes prepared statement to the query and sends it to the database
//sets each ? parameter to the parameters passed in to the method
try{
    ps = connection.prepareStatement(query);
    ps.setInt(1, orderPK);
    ps.setInt(2, productPK);
    ps.setInt(3, quantity);

    //executeUpdate() returns the number of records affected by the prepared statement query
    //if 0 isn't returned the record has been created but the user doesn't need to know so nothing
    //it output
    //if 0 is returned then no record were affected so the record wasn't created so an error
    //message window pops up
    if(ps.executeUpdate() != 0){
        }
    else{
        JOptionPane.showMessageDialog(null, "error");
        }
    }

    //catches exceptions thrown
    catch(SQLException ex){
        Logger.getLogger(orders.class.getName()).log(Level.SEVERE, null, ex);
        }
}

public static void deleteOrderProduct(int productPK){
    //passes in the product primary key

    //creates connection to the database
    Connection connection = database.getConnection();
```

Final Code

```
PreparedStatement ps;

//initializes prepared statement to the query in the brackets and sends it to the database
//set ? parameter in the query to product primary key passed in
try{
    ps = connection.prepareStatement("DELETE FROM orderproducts WHERE product_fk = ?");
    ps.setInt(1, productPK);

    //executeUpdate() returns the number of records affected by the query
    //if 0 isn't returned then a record was affected so a success message window pops up
    //if 0 is returned then no record were affected so an error message window pops up
    if (ps.executeUpdate() != 0){
        JOptionPane.showMessageDialog(null, "Product deleted from order");
    }
    else{
        JOptionPane.showMessageDialog(null, "Error");
    }
}

//catches exceptions thrown
catch(SQLException ex){
    Logger.getLogger(orderProducts.class.getName()).log(Level.SEVERE, null, ex);
}

}

}
```

6.28 orders.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
 * license
 *
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package classes;

import java.sql.Date;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

/**
 *
 * @author garmin
 */
public class orders {

    private int order_pk;
    private int createdby_fk;
    private Date date;

    public Integer getOrderPK(){
        return order_pk;
    }
}
```

Final Code

```
}

public Integer getFK(){
    return createdby_fk;
}

public Date getDate(){
    return date;
}

public void setOrderPK(int orderPK){
    this.order_pk = orderPK;
}

public orders() {}

public orders(Integer pk, int fk, Date date){
    this.order_pk = pk;
    this.createdby_fk = fk;
    this.date = date;
}

public static void addOrder(int userPK){
    //passes in the user primary key

    //creates connection to the database
    Connection connection = database.getConnection();
    PreparedStatement ps;
    String query;
```

Final Code

```
//initializes query
query = "INSERT INTO orders(ordercreatedby_fk, date) VALUES(?,sysdate())";

//initializes prepared statement to the query and sends it to the database
//sets the ? parameter to the user primary key value
try{
    ps = connection.prepareStatement(query);
    ps.setInt(1, userPK);

    //executeUpdate() returns number of records affected by the query
    //if 0 isn't returned then a record was affected so it worked
    //if 0 is returned then no record was created so an error message window pops up
    if(ps.executeUpdate() != 0){
        }
    else{
        JOptionPane.showMessageDialog(null, "error");
        }
    }

//catches exception thrown
catch(SQLException ex){
    Logger.getLogger(orders.class.getName()).log(Level.SEVERE, null, ex);
}

}

public static int getLatestOrder(int userPK){
    //passes in user primary key

    //creates connection to the database
    Connection connection = database.getConnection();
    PreparedStatement ps;
```

Final Code

```
ResultSet rs;
String query;
int orderPK = 0;

//initializes query
query = "select max(order_pk)order_pk from orders where ordercreatedby_fk = ?";

//initializes prepared statement to the query and sends it to the database
//sets ? parameter to the user primary key value
//initializes result set to the data returned from executing the query
//sets orderPK to the value in the result set
try{
    ps = connection.prepareStatement(query);
    ps.setInt(1, userPK);
    rs = ps.executeQuery();
    rs.next();
    orderPK = rs.getInt("order_pk");
}
//catches exceptions thrown
catch(SQLException ex){
    Logger.getLogger(orders.class.getName()).log(Level.SEVERE, null, ex);
}

//returns the latest order primary key
return orderPK;
}

public ArrayList<orders> ordersList(String orderPK){
    //passes in order primary key
    //creates new ArrayList of type orders called orderList
}
```

Final Code

```
ArrayList<orders> orderList = new ArrayList<>();

//creates connection to the database
Connection connection = database.getConnection();
PreparedStatement ps;
ResultSet rs;
String query;

//initializes query
query = "select * from orders where order_pk LIKE ?";

//initializes prepared statement to the query and sends it to the database
//sets ? parameter in the query to the value of orderPK
try{
    ps = connection.prepareStatement(query);
    ps.setString(1,"%" + orderPK + "%");
    rs = ps.executeQuery();
    orders order;
    //loops through the data in the result set
    //creates new instance of orders with the values from the result set
    //adds the objects to the orderList ArrayList
    while(rs.next()){
        order = new orders(rs.getInt("order_pk"), rs.getInt("ordercreatedby_fk"),
rs.getDate("date"));
        orderList.add(order);
    }
}
//catches sql exceptions thrown
catch(SQLException ex){
    Logger.getLogger(orderProducts.class.getName()).log(Level.SEVERE, null, ex);
}
```

Final Code

```
//returns the ArrayList with all the orders in it
return orderList;
}

public static void updateStock(int productPK, int quantity){
    //passes in the product primary key and quantity

    //creates connection to the database
    Connection connection = database.getConnection();
    PreparedStatement ps;
    //initializes prepared statement to the query in the brackets and sends it to the database
    //sets ? parameters to the parameters passed in
    //executes the query
    try{
        ps = connection.prepareStatement("UPDATE products SET stock = stock - ? WHERE
product_pk = ?;");
        ps.setInt(1, quantity);
        ps.setInt(2, productPK);
        ps.executeUpdate();
    }

    //catches exceptions thrown
    catch(SQLException ex){
        Logger.getLogger(orders.class.getName()).log(Level.SEVERE, null, ex);
    }
}

}
```

6.29 products.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
 * license
 *
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package classes;

import java.sql.Date;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
/***
 *
 * @author garmin
 */
public class products {

    Connection connection;

    private int product_pk;
    private String product_name;
    private String product_description;
    private float product_sellprice;
    private float product_cost;
    private int createdby_fk;
    private Date createdon;
```

Final Code

```
private Date lastupdated;  
private boolean deletedflag;  
private String category;  
private int stock;  
private int catFK;  
  
public Integer getProductPK(){  
    return product_pk;  
}  
  
public String getName(){  
    return product_name;  
}  
  
public String getDescription(){  
    return product_description;  
}  
  
public float getPrice(){  
    return product_sellprice;  
}  
  
public float getCost(){  
    return product_cost;  
}  
  
public int getFK(){  
    return createdby_fk;  
}  
  
public Date getDateCreated(){
```

Final Code

```
return createdon;  
}  
  
public Date getLastUpdated(){  
    return lastupdated;  
}  
  
public boolean getFlag(){  
    return deletedflag;  
}  
  
public String getCat(){  
    return category;  
}  
  
public int getStock(){  
    return stock;  
}  
  
public int getCatFK(){  
    return catFK;  
}  
  
public void setpk(int productPK){  
    this.product_pk = productPK;  
}  
  
public void setName(String name){  
    this.product_name = name;  
}
```

Final Code

```
public void setDescription(String description){  
    this.product_description = description;  
}  
  
public void setPrice(float price){  
    this.product_sellprice = price;  
}  
  
public void setCost(float cost){  
    this.product_sellprice = cost;  
}  
  
public void setStock(int stock){  
    this.stock = stock;  
}  
  
public void setCat(String category){  
    this.category = category;  
}  
  
public void setCatFK (int catfk){  
    this.catFK = catfk;  
}  
  
public products(){}
public products(Integer pk, String name, String description, float price, float cost,int fk, Date createdOn, Date updated, boolean deletedFlag,String categoryName, Integer stock, Integer categoryfk)  
{  
    this.product_pk = pk;
```

Final Code

```
this.product_name = name;
this.product_description = description;
this.product_sellprice = price;
this.product_cost = cost;
this.createdby_fk = fk;
this.createdon = createdOn;
this.lastupdated = updated;
this.deletedflag = deletedFlag;
this.category = categoryName;
this.stock = stock;
this.catFK = categoryfk;
}

public static void addProduct(products product){

    //creates connection to the database
    Connection connect = database.getConnection();
    PreparedStatement ps = null;

    //initializes prepared statement to the query in the brackets and sends it to the database
    //sets ? parameters to the appropriate getter methods on the product passed in

    try{
        ps = connect.prepareStatement("INSERT INTO products(product_name, product_description,
product_sellprice, product_cost, category, stock, createdby_fk, createdon, lastupdated, deletedflag,
category_fk) VALUES(?,?,?,?,?, ?, ?, sysdate(), sysdate(), 0, ?)");
        ps.setString(1, product.getName());
        ps.setString(2, product.getDescription());
        ps.setFloat(3, product.getPrice());
        ps.setFloat(4, product.getCost());
        ps.setString(5, product.getCat());
        ps.setInt(6, product.getStock());
    }
}
```

Final Code

```
ps.setInt(7, product.getFK());
ps.setInt(8, product.getCatFK());

//executeUpdate() returns the number of records affected
//if 0 is returned, no records were created so an error message window pops up
//if 0 isn't returned then a record was affected/created so a success message window pops up
if(ps.executeUpdate() != 0){
    JOptionPane.showMessageDialog(null, "New Product Added");
}
else{
    JOptionPane.showMessageDialog(null, "Error");
}
}

//catches exceptions thrown
catch (SQLException ex) {
    Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
}

}

public ArrayList<products> productList(String searchVal, int stock, int catfk){
    //passes in search value, stock and category primary key

    //creates new ArrayList of type products called prodList
    ArrayList<products> prodList = new ArrayList<>();

    //creates connection to the database
    connection = database.getConnection();
    PreparedStatement ps;
    ResultSet rs;
    String query;
```

Final Code

```
//if a category has been selected
if (catfk != 0){
    //initializes query to only retrieve products from the selected category
    query = "SELECT * FROM products WHERE deletedflag = 0 AND product_name LIKE ? AND
stock > ? AND category_fk = ?;";

    //initializes prepared statement to the query and sends it to the database
    //sets ? parameters to the parameters passed in
    //initializes result set to the data from executing the query
    try{
        ps = connection.prepareStatement(query);
        ps.setString(1, "%" + searchVal + "%");
        ps.setInt(2, stock);
        ps.setInt(3, catfk);
        rs = ps.executeQuery();
        products product;

        //loops through the result set
        //creates new instance of the product class with values from the result set
        //add each instance to prodList
        while (rs.next()){

            product = new products(rs.getInt("product_pk"), rs.getString("product_name"),
rs.getString("product_description"), rs.getFloat("product_sellprice"), rs.getFloat("product_cost"),
rs.getInt("createdby_fk"), rs.getDate("createdon"), rs.getDate("lastupdated"),
rs.getBoolean("deletedflag"), rs.getString("category"), rs.getInt("stock"), rs.getInt("category_fk"));

            prodList.add(product);
        }
    }
    //catches exceptions thrown
    catch(SQLException ex){
```

Final Code

```
Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);

}

}

//if a category hasn't been selected

else{

    //initializes query

    query = "SELECT * FROM products WHERE deletedflag = 0 AND product_name LIKE ? AND
stock > ?;";




    //initializes prepared statement to query and sends it to the database

    //sets ? parameters to the values passed in

    //initializes result set to the data returned from the query

    try{

        ps = connection.prepareStatement(query);

        ps.setString(1, "%" + searchVal + "%");

        ps.setInt(2, stock);

        rs = ps.executeQuery();

        products product;

        //loops through the result set

        //instantiates new objects of the products class using the values from the result set

        //add each product to the ArrayList prodList

        while (rs.next()){

            product = new products(rs.getInt("product_pk"), rs.getString("product_name"),
rs.getString("product_description"), rs.getFloat("product_sellprice"), rs.getFloat("product_cost"),
rs.getInt("createdby_fk"), rs.getDate("createdon"), rs.getDate("lastupdated"),
rs.getBoolean("deletedflag"), rs.getString("category"), rs.getInt("stock"), rs.getInt("category_fk"));

            prodList.add(product);

        }

    }

    //catches exceptions thrown

    catch(SQLException ex){

        Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);

    }

}
```

Final Code

```
}

}

//returns the ArrayList prodList with all the products in it

return prodList;

}

public static void deleteProduct(int id){

    //passes in the selected product primary key

    //creates connection to the database

    Connection connection = database.getConnection();

    PreparedStatement ps;

    //initialises prepared statement to the query in the brackets and sends it to the database

    //sets ? parameter to the product primary key passed in

    try{

        ps = connection.prepareStatement("UPDATE products SET deletedflag = 1 WHERE product_pk
= ?;");

        ps.setInt(1, id);

        //executeUpdate() returns the number of records affected

        //if 0 isn't returned, a record has been updated so a success message window pops up

        //if 0 is returned, 0 records have been updated so an error message window pops up

        if (ps.executeUpdate() != 0){

            JOptionPane.showMessageDialog(null, "Product has been deleted.");

        }

        else{

            JOptionPane.showMessageDialog(null, "There was an error, please try again.");

        }

    }

    //catches exceptions thrown
```

Final Code

```
catch(SQLException ex){  
    Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);  
}  
}  
  
public static void updateProduct(products product){  
  
    //creates connection to the database  
    Connection connection = database.getConnection();  
    PreparedStatement ps;  
  
    //initializes prepared statement to the query in the brackets and sends it to the database  
    //sets ? parameters to the appropriate getter methods on the product passed in  
    try{  
        ps = connection.prepareStatement("UPDATE products SET product_name = ?,  
product_description = ?, product_sellprice = ?, product_cost = ?, category = ?, stock = ?,  
createdby_fk = createdby_fk, createdon = createdon, lastupdated = sysdate(), deletedflag = 0,  
category_fk = ? WHERE product_pk = ?");  
        ps.setString(1, product.getName());  
        ps.setString(2, product.getDescription());  
        ps.setFloat(3, product.getPrice());  
        ps.setFloat(4, product.getCost());  
        ps.setString(5, product.getCat());  
        ps.setInt(6, product.getStock());  
        ps.setInt(7, product.getCatFK());  
        ps.setInt(8, product.getProductPK());  
  
        //executeUpdate() returns the number of records affected  
        //if 0 isn't returned then a record was updated so a success message window pops up  
        //if 0 is returned then 0 records have been updated so an error message window pops up  
        if(ps.executeUpdate() != 0){  
            JOptionPane.showMessageDialog(null, "Product updated.");  
        }  
    }  
}
```

Final Code

```
}

else{
    JOptionPane.showMessageDialog(null, "There was an error, please try again.");
}

}

//catches exceptions thrown
catch(SQLException ex){
    Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
}

}

}
```

6.30 productsInOrder.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
 * license
 *
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package classes;

import java.sql.Date;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

/**
 *
 * @author garmin
 */
public class productsInOrder {

    Connection connection;

    private int product_pk;
    private int order_pk;
    private String product_name;
    private float product_sellprice;
    private float product_cost;
```

Final Code

```
private int quantity;  
private float profit;  
  
public Integer getProductPK(){  
    return product_pk;  
}  
  
public Integer getOrderPK(){  
    return order_pk;  
}  
  
public String getProductName(){  
    return product_name;  
}  
  
public float getPrice(){  
    return product_sellprice;  
}  
  
public float getCost(){  
    return product_cost;  
}  
  
public Integer getQuantity(){  
    return quantity;  
}  
  
public float getProfit(){  
    return profit;  
}
```

Final Code

```
public void setProductPK(int productPK){  
    this.product_pk = productPK;  
}  
  
public void setOrderPK(int orderPK){  
    this.order_pk = orderPK;  
}  
  
public void setName(String name){  
    this.product_name = name;  
}  
  
public void setPrice(float price){  
    this.product_sellprice = price;  
}  
  
public void setCost(float cost){  
    this.product_cost = cost;  
}  
  
public void setQuantity(int quantity){  
    this.quantity = quantity;  
}  
  
public void setProfit(int profit){  
    this.profit = profit;  
}  
  
public productsInOrder(){}  
}
```

Final Code

```
public productsInOrder(Integer pk, String name, float price, float cost, int quantity, float profit){

    this.product_pk = pk;
    this.product_name = name;
    this.product_sellprice = price;
    this.product_cost = cost;
    this.quantity = quantity;
    this.profit = profit;

}

public ArrayList<productsInOrder> productsInOrderList(int orderPK){

    //creates new arraylist of type productsInOrder
    //select query retrieves necessary fields from products and orderproducts tables where the
    product primary key matches the foreign key of the product in the order
    //while loop creates new instances of productsInOrder with the values from the database
    //then adds these instances to the arraylist

    ArrayList<productsInOrder> productList = new ArrayList<>();

    connection = database.getConnection();
    PreparedStatement ps;
    ResultSet rs;
    String query;

    query = "select product_pk, product_name, product_sellprice, product_cost, quantity from
    products,orderproducts where product_pk = orderproducts.product_fk and orderproducts.order_fk
    = ?";

    try{
        ps = connection.prepareStatement(query);
        ps.setInt(1, orderPK);
```

Final Code

```
rs = ps.executeQuery();
productsInOrder product;

while (rs.next()){

    product = new productsInOrder(rs.getInt("product_pk"), rs.getString("product_name"),
rs.getFloat("product_sellprice"), rs.getFloat("product_cost"), rs.getInt("quantity"),
((rs.getFloat("product_sellprice")-rs.getFloat("product_cost"))*rs.getInt("quantity")));

    productList.add(product);

}

}

catch(SQLException ex){

    Logger.getLogger(orderProducts.class.getName()).log(Level.SEVERE, null, ex);
}

return productList;

}

}
```

6.31 profitByMonth_view.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
 * license
 *
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package classes;

import java.awt.Color;
import java.awt.ComponentOrientation;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTable;
import javax.swing.border.TitledBorder;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartFrame;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
```

Final Code

```
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;

/**
 *
 * @author garmin
 */
public class profitByMonth_view {

    Connection connection;

    //I added these as they are used by addComponentsToPane which is the GridBagLayout layout
    //manager to dynamically create and position the chart and the data table in Swing
    final static boolean shouldFill = false;
    final static boolean shouldWeightX = true;
    final static boolean RIGHT_TO_LEFT = false;

    //ProfitByMonth_view attributes
    private String mymonth;
    private float total_cost;
    private float total_price;
    private float total_profit;

    //methods to return the values of each attribute
    public String getmymonth(){
        return mymonth;
    }

    public float gettotal_cost(){
        return total_cost;
    }
```

Final Code

```
}

public float gettotal_profit(){
    return total_profit;
}

public float gettotal_price(){
    return total_price;
}

public profitByMonth_view(){}

//ProfitByMonth_view row of data object - used to create an array of rows of data - used to hold
the results of a dataset queried from the view

public profitByMonth_view(String mymonth, float total_cost, float total_price, float total_profit)
{
    this.mymonth = mymonth;
    this.total_cost = total_cost;
    this.total_price = total_price;
    this.total_profit = total_profit;
}

//private method just to make the SQL for my report ProfitByMonth
private static String ProfitByMonth_viewSQL(String orderbyVal)
{
    //query to retrieve data from the db view where the category is like the report param passed in
    val

    String query = "SELECT * FROM ProfitByMonth_view WHERE mymonth BETWEEN ? AND ?
    ORDER BY ";

    //append the correct ORDER BY column for the query based on the orderbyVal param
```

Final Code

```
//note this can in some cases be a risk for SQL injection but PreparedStatement doesn't allow
ORDER BY ?

if (orderbyVal=="mymonth") {
    query=query+"mymonth;";
} else if (orderbyVal=="total_cost") {
    query=query+"total_cost;";
} else if (orderbyVal=="total_price") {
    query=query+"total_price;";
} else if (orderbyVal=="total_profit") {
    query=query+"total_profit;";
} else {
    query=query+"mymonth;"; //remove SQL injection risk by examining the orderbyVal and only
allowing mymonth column name
}

return query;
}

//this ArrayList class is a resizable array, which is present in the java. util package from the docs
// built-in arrays have a fixed size, ArrayLists can change their size dynamically so
//I used it to retrieve the data from the database view so it can be shown as the data table in the
report

public ArrayList<profitByMonth_view> ProfitByMonth_viewList (String startDate, String endDate,
String orderbyVal){

    //create arraylist of ProductsByCategorySummed_view rows
    ArrayList<profitByMonth_view> ProfitByMonth_viewList = new ArrayList<>();

    //get a db connection and make the resultset to hold the db data
    connection = database.getConnection();
    ResultSet rs;
```

Final Code

```
//query to retrieve data from the db view where the category is like the report param passed in  
val  
  
String query = ProfitByMonth_viewSQL(orderbyVal);  
  
try{  
    connection = database.getConnection();  
  
    //PreparedStatement ps from the query with the categoryID like %val%  
    PreparedStatement ps = connection.prepareStatement(query);  
    ps.setString(1, startDate);  
    ps.setString(2, endDate);  
  
    rs = ps.executeQuery();  
  
    //create the view data row and loop through the data in the SQL and add each row to the  
    arraylist  
    profitByMonth_view ProfitByMonth_viewRow;  
    while (rs.next()){  
        ProfitByMonth_viewRow = new profitByMonth_view(rs.getString("mymonth"),  
        rs.getFloat("total_cost"), rs.getFloat("total_price"), rs.getFloat("total_profit") );  
  
        ProfitByMonth_viewList.add(ProfitByMonth_viewRow);  
    }  
}  
catch (SQLException ex){  
    Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);  
}  
  
//return the dataset in the arraylist  
return ProfitByMonth_viewList;  
}
```

Final Code

```
//the simple ProfitByMonth report which is just the chart returned in a frame to be popped up
public static ChartFrame ProfitByMonth(String startDate, String endDate, String metricIn, String
orderbyVal)

{
    Connection connect = database.getConnection();

    PreparedStatement ps;
    ResultSet rs;

    String mymonth="";
    float metric=0;

    //query to retrieve data from the db view where the category is like the report param passed in
    val
    String query = ProfitByMonth_viewSQL(orderbyVal);

    //JFreeChart dataset object to pass the data into the createBarChart method
    DefaultCategoryDataset dataset = new DefaultCategoryDataset();

    try{
        ps = connect.prepareStatement(query);

        ps.setString(1, startDate);
        ps.setString(2, endDate);

        //debug code to see the prepared statement
        System.out.println(ps);

        rs = ps.executeQuery();

        //populate the DefaultCategoryDataset
```

Final Code

```
while (rs.next()){

    mymonth = rs.getString("mymonth");
    metric = rs.getFloat(metricIn);
    dataset.setValue(metric, mymonth, mymonth);
}

}

catch (SQLException ex){

    Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
}

//make the chart

JFreeChart chart = ChartFactory.createBarChart("ProfitByMonth -
"+metricIn,"mymonth",metricIn,dataset,PlotOrientation.VERTICAL,false,true,false);

//set some chart attributes

CategoryPlot p = chart.getCategoryPlot();
p.setRangeGridlinePaint(Color.black);

//create a frame with the chart in it using JFree ChartFrame type

//public class ChartFrame extends the swing JFrame

ChartFrame frame = new ChartFrame("ProfitByMonth - "+metricIn,chart);

return frame;

}

//create the chart but return it in panel so it can be assembled into the final report

public static ChartPanel ProfitByMonthPanel(String startDate, String endDate, String metricIn,
String orderbyVal)

{
    Connection connect = database.getConnection();
```

Final Code

```
PreparedStatement ps;
ResultSet rs;

String mymonth="";
float metric=0;

//query to retrieve data from the db view where the category is like the report param passed in
val
String query = ProfitByMonth_viewSQL(orderbyVal);

DefaultCategoryDataset dataset = new DefaultCategoryDataset();

try{
    ps = connect.prepareStatement(query);

    ps.setString(1, startDate);
    ps.setString(2, endDate);

    System.out.println(ps);

    rs = ps.executeQuery();

    while (rs.next()){

        mymonth = rs.getString("mymonth");
        metric = rs.getFloat(metricIn);
        dataset.setValue(metric, mymonth, mymonth);
    }
}

catch (SQLException ex){
    Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
}
```

Final Code

```
}

//make the chart, set the attributes, return the panel to the assembler
JFreeChart chart = ChartFactory.createBarChart("ProfitByMonth -
"+metricIn,"mymonth",metricIn,dataset,PlotOrientation.VERTICAL,false,true,false);

CategoryPlot p = chart.getCategoryPlot();
p.setRangeGridlinePaint(Color.black);

ChartPanel panel = new ChartPanel(chart);

return panel;

}

//fills and returns the jTable with contents of arrayList so it can be added to the data panel
public static JTable fillProfitByMonth_viewTable(String startDate, String endDate, String
orderbyVal){

    classes.profitByMonth_view ProfitByMonth_view = new classes.profitByMonth_view();

    ArrayList <classes.profitByMonth_view> ProfitByMonth_viewList =
ProfitByMonth_view.ProfitByMonth_viewList(startDate, endDate, orderbyVal);

    String [] columnNames = {"mymonth","total_cost","total_price","total_profit"};
    Object [][] rows = new Object[ProfitByMonth_viewList.size()][4];

    for(int i = 0; i<ProfitByMonth_viewList.size(); i++){
        rows[i][0] = ProfitByMonth_viewList.get(i).getmymonth();
        rows[i][1] = ProfitByMonth_viewList.get(i).gettotal_cost();
        rows[i][2] = ProfitByMonth_viewList.get(i).gettotal_price();
        rows[i][3] = ProfitByMonth_viewList.get(i).gettotal_profit();
    }
}
```

Final Code

```
classes.table model = new classes.table(rows, columnNames);

JTable ProfitByMonth_viewTable = new JTable();
ProfitByMonth_viewTable.setModel(model);

//sets the jTable column names
//bug not working right now, table has no column headers when displayed
ProfitByMonth_viewTable.getColumnModel().getColumn(0).setHeaderValue("mymonth");
ProfitByMonth_viewTable.getColumnModel().getColumn(1).setHeaderValue("total_cost");
ProfitByMonth_viewTable.getColumnModel().getColumn(2).setHeaderValue("total_price");
ProfitByMonth_viewTable.getColumnModel().getColumn(3).setHeaderValue("total_profit");

//return the table of data so it can be laid out in the assembler
return ProfitByMonth_viewTable;

}

//assembler for the report - this gets the chart and the data table and lays them out using
GridBagLayout layout manager

public static JFrame fillProfitByMonth_viewAssemble(String startDate, String endDate, String
metricIn, String orderbyVal, int chartWidth, int chartHeight, int tableWidth, int tableHeight, String
frameTitle, String chartTitle, String tableTitle)
{
    //make the chart
    ChartPanel chartpanel = classes.profitByMonth_view.ProfitByMonthPanel(startDate, endDate,
metricIn, orderbyVal);

    //set the size of the chart
    chartpanel.setPreferredSize(new Dimension(chartWidth, chartHeight));
    JPanel jPanelChart = new JPanel();
    jPanelChart.add(chartpanel);
    jPanelChart.setBorder(BorderFactory.createTitledBorder( BorderFactory.createEtchedBorder(),
chartTitle, TitledBorder.CENTER, TitledBorder.TOP));
}
```

Final Code

```
//make the data table

JTable jTableData = classes.profitByMonth_view.fillProfitByMonth_viewTable(startDate,
endDate, orderbyVal);

//set the size of the table

jTableData.setPreferredSize(new Dimension(tableWidth, tableHeight));

JPanel jPanelData = new JPanel();

jPanelData.add(jTableData);

jPanelData.setBorder(BorderFactory.createTitledBorder( BorderFactory.createEtchedBorder(),
tableTitle, TitledBorder.CENTER, TitledBorder.TOP));

//new frame to show the GridBagLayoutDemo for the POS reports

JFrame frame = new JFrame(frameTitle);

//Set up the content pane by adding the chart panel and the data panel

//GridBagLayout manager time!

classes.profitByMonth_view.addComponentstoPane(frame.getContentPane(), jPanelChart,
jPanelData);

//return the assembled report

return frame;

}

//I used GridBagLayout manager as I need to dynamically lay out the chart and the data table

//if I didn't use this layout manager then its really hard to make the JComponents go where I need

//they just overlap themselves

public static void addComponentsToPane(Container pane, JPanel chartpanel, JPanel datapanel) {

if (RIGHT_TO_LEFT) {

pane.setComponentOrientation(ComponentOrientation.RIGHT_TO_LEFT);

}
```

Final Code

6.32 table.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
license
 *
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package classes;

import javax.swing.table.AbstractTableModel;

/**
 *
 *
 * @author garmin
 */
public class table extends AbstractTableModel {

    private String[] columns;
    private Object[][] rows;

    public table(){}
    
    public table(Object[][] data, String[] columnsName){
        this.columns = columnsName;
        this.rows = data;
    }

    @Override
    public int getRowCount() {
        return this.rows.length;
```

Final Code

```
}
```



```
@Override
public int getColumnCount() {
```



```
    return this.columns.length;
```



```
}
```



```
@Override
public Object getValueAt(int rowIndex, int columnIndex) {
```



```
    return this.rows[rowIndex][columnIndex];
```



```
}
```



```
public String getColName(int columns){
```



```
    return this.columns[columns];
```



```
}
```



```
}
```

6.33 totalOrdersByMonth_view.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
 * license
 *
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package classes;

import java.awt.Color;
import java.awt.ComponentOrientation;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTable;
import javax.swing.border.TitledBorder;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartFrame;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
```

Final Code

```
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.DefaultCategoryDataset;

/**
 *
 * @author garmin
 */
public class totalOrdersByMonth_view {

    Connection connection;

    //I added these as they are used by addComponentsToPane which is the GridBagLayout layout
    manager to dynamically create and position the chart and the data table in Swing

    final static boolean shouldFill = false;
    final static boolean shouldWeightX = true;
    final static boolean RIGHT_TO_LEFT = false;

    //TotalOrdersByMonth_view attributes

    private String mymonth;
    private int total_orders;

    //methods to return the values of each attribute

    public String getmymonth(){
        return mymonth;
    }

    public int gettotal_orders(){
        return total_orders;
    }
}
```

Final Code

```
public totalOrdersByMonth_view(){}

//TotalOrdersByMonth_view row of data object - used to create an array of rows of data - used to
hold the results of a dataset queried from the view

public totalOrdersByMonth_view(String mymonth, int total_orders)
{
    this.mymonth = mymonth;
    this.total_orders = total_orders;
}

//private method just to make the SQL for my report TotalOrdersByMonth
private static String TotalOrdersByMonth_viewSQL(String orderbyVal)
{
    //query to retrieve data from the db view where the category is like the report param passed in
    val

    String query = "SELECT * FROM TotalOrdersByMonth_view WHERE mymonth BETWEEN ? AND ?
    ORDER BY ";

    //append the correct ORDER BY column for the query based on the orderbyVal param
    //note this can in some cases be a risk for SQL injection but PreparedStatement doesn't allow
    ORDER BY ?

    if (orderbyVal=="mymonth") {
        query=query+"mymonth;";
    } else if (orderbyVal=="total_orders") {
        query=query+"total_orders;";
    } else {
        query=query+"mymonth;"; //remove SQL injection risk by examining the orderbyVal and only
        allowing mymonth column name
    }

    return query;
}
```

Final Code

```
//this ArrayList class is a resizable array, which is present in the java. util package from the docs
// built-in arrays have a fixed size, ArrayLists can change their size dynamically so
// I used it to retrieve the data from the database view so it can be shown as the data table in the
report

public ArrayList<totalOrdersByMonth_view> TotalOrdersByMonth_viewList (String startDate,
String endDate, String orderbyVal){

    //create arraylist of ProductsByCategorySummed_view rows
    ArrayList<totalOrdersByMonth_view> TotalOrdersByMonth_viewList = new ArrayList<>();

    //get a db connection and make the resultset to hold the db data
    connection = database.getConnection();
    ResultSet rs;

    //query to retrieve data from the db view where the category is like the report param passed in
val
    String query = TotalOrdersByMonth_viewSQL(orderbyVal);

    try{
        connection = database.getConnection();

        //PreparedStatement ps from the query with the categoryID like %val%
        PreparedStatement ps = connection.prepareStatement(query);
        ps.setString(1, startDate);
        ps.setString(2, endDate);

        rs = ps.executeQuery();

        //create the view data row and loop through the data in the SQL and add each row to the
arraylist
        totalOrdersByMonth_view TotalOrdersByMonth_viewRow;
```

Final Code

```
while (rs.next()){

    TotalOrdersByMonth_viewRow = new totalOrdersByMonth_view(rs.getString("mymonth"),
rs.getInt("total_orders") );




    TotalOrdersByMonth_viewList.add(TotalOrdersByMonth_viewRow);

}

}

catch (SQLException ex){

    Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);

}

//return the dataset in the arraylist

return TotalOrdersByMonth_viewList;

}

//the simple TotalOrdersByMonth report which is just the chart returned in a frame to be popped up

public static ChartFrame TotalOrdersByMonth(String startDate, String endDate, String metricIn,
String orderbyVal)

{

    Connection connect = database.getConnection();

    PreparedStatement ps;

    ResultSet rs;

    String mymonth="";
    float metric=0;

    //query to retrieve data from the db view where the category is like the report param passed in val

    String query = TotalOrdersByMonth_viewSQL(orderbyVal);
```

Final Code

```
//JFreeChart dataset object to pass the data into the createBarChart method
DefaultCategoryDataset dataset = new DefaultCategoryDataset();

try{
    ps = connect.prepareStatement(query);

    ps.setString(1, startDate);
    ps.setString(2, endDate);

    //debug code to see the prepared statement
    System.out.println(ps);

    rs = ps.executeQuery();

    //populate the DefaultCategoryDataset
    while (rs.next()){

        mymonth = rs.getString("mymonth");
        metric = rs.getFloat(metricIn);
        dataset.setValue(metric, mymonth, mymonth);
    }
}

catch (SQLException ex){
    Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);
}

//make the chart
JFreeChart chart = ChartFactory.createBarChart("TotalOrdersByMonth - "
"+metricIn,"mymonth",metricIn,dataset,PlotOrientation.VERTICAL,false,true,false);

//set some chart attributes
CategoryPlot p = chart.getCategoryPlot();
```

Final Code

```
p.setRangeGridlinePaint(Color.black);

//create a frame with the chart in it using JFree ChartFrame type
//public class ChartFrame extends the swing JFrame
ChartFrame frame = new ChartFrame("TotalOrdersByMonth - "+metricIn,chart);

return frame;

}

//create the chart but return it in panel so it can be assembled into the final report
public static ChartPanel TotalOrdersByMonthPanel(String startDate, String endDate, String
metricIn, String orderbyVal)
{
    Connection connect = database.getConnection();

    PreparedStatement ps;
    ResultSet rs;

    String mymonth="";
    float metric=0;

    //query to retrieve data from the db view where the category is like the report param passed in
val
    String query = TotalOrdersByMonth_viewSQL(orderbyVal);

    DefaultCategoryDataset dataset = new DefaultCategoryDataset();

    try{
        ps = connect.prepareStatement(query);

        ps.setString(1, startDate);

        ps.setDouble(2, metric);
        ps.setString(3, mymonth);
        ps.setString(4, metricIn);
        ps.setString(5, orderbyVal);
        ps.setString(6, endDate);
    }
}
```

Final Code

```
ps.setString(2, endDate);

System.out.println(ps);

rs = ps.executeQuery();

while (rs.next()){

    mymonth = rs.getString("mymonth");

    metric = rs.getFloat(metricIn);

    dataset.setValue(metric, mymonth, mymonth);

}

}

catch (SQLException ex){

    Logger.getLogger(products.class.getName()).log(Level.SEVERE, null, ex);

}

//make the chart, set the attributes, return the panel to the assembler

JFreeChart chart = ChartFactory.createBarChart("TotalOrdersByMonth -
"+metricIn,"mymonth",metricIn,dataset,PlotOrientation.VERTICAL,false,true,false);

CategoryPlot p = chart.getCategoryPlot();

p.setRangeGridlinePaint(Color.black);

ChartPanel panel = new ChartPanel(chart);

return panel;

}

//fills and returns the jTable with contents of arrayList so it can be added to the data panel
```

Final Code

```
public static JTable fillTotalOrdersByMonth_viewTable(String startDate, String endDate, String orderbyVal){

    classes.totalOrdersByMonth_view TotalOrdersByMonth_view = new
    classes.totalOrdersByMonth_view();

    ArrayList <classes.totalOrdersByMonth_view> TotalOrdersByMonth_viewList =
    TotalOrdersByMonth_view.TotalOrdersByMonth_viewList(startDate, endDate, orderbyVal);

    String [] columnNames = {"mymonth","total_orders"};
    Object [][] rows = new Object[TotalOrdersByMonth_viewList.size()][2];

    for(int i = 0; i<TotalOrdersByMonth_viewList.size(); i++){
        rows[i][0] = TotalOrdersByMonth_viewList.get(i).getmymonth();
        rows[i][1] = TotalOrdersByMonth_viewList.get(i).gettotal_orders();
    }

    classes.table model = new classes.table(rows, columnNames);
    JTable TotalOrdersByMonth_viewTable = new JTable();
    TotalOrdersByMonth_viewTable.setModel(model);

    //sets the jTable column names
    //bug not working right now, table has no column headers when displayed

    TotalOrdersByMonth_viewTable.getColumnModel().getColumn(0).setHeaderValue("mymonth");

    TotalOrdersByMonth_viewTable.getColumnModel().getColumn(1).setHeaderValue("total_orders");

    //return the table of data so it can be laid out in the assembler
    return TotalOrdersByMonth_viewTable;

}
```

Final Code

```
//assembler for the report - this gets the chart and the data table and lays them out using
GridBagLayout layout manager

public static JFrame fillTotalOrdersByMonth_viewAssemble(String startDate, String endDate,
String metricIn, String orderbyVal, int chartWidth, int chartHeight, int tableWidth, int tableHeight,
String frameTitle, String chartTitle, String tableTitle)

{
    //make the chart

    ChartPanel chartpanel =
classes.totalOrdersByMonth_view.TotalOrdersByMonthPanel(startDate, endDate, metricIn,
orderbyVal);

    //set the size of the chart

    chartpanel.setPreferredSize(new Dimension(chartWidth, chartHeight));

    JPanel jPanelChart = new JPanel();

    jPanelChart.add(chartpanel);

    jPanelChart.setBorder(BorderFactory.createTitledBorder( BorderFactory.createEtchedBorder(),
chartTitle, TitledBorder.CENTER, TitledBorder.TOP));

    //make the data table

    JTable jTableData =
classes.totalOrdersByMonth_view.fillTotalOrdersByMonth_viewTable(startDate, endDate,
orderbyVal);

    //set the size of the table

    jTableData.setPreferredSize(new Dimension(tableWidth, tableHeight));

    JPanel jPanelData = new JPanel();

    jPanelData.add(jTableData);

    jPanelData.setBorder(BorderFactory.createTitledBorder( BorderFactory.createEtchedBorder(),
tableTitle, TitledBorder.CENTER, TitledBorder.TOP));

    //new frame to use the GridBagLayout manager for the POS reports

    JFrame frame = new JFrame(frameTitle);

    //Set up the content pane by adding the chart panel and the data panel

    //GridBagLayout manager to make the pane with the chart and the table
```

Final Code

Final Code

6.34 userHours.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
 * license
 *
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package classes;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;
import java.sql.Statement;

/**
 *
 * @author garmin
 */
public class userHours {

    private int user_pk;
    private String user_name;
    private float hours;
    private String yearMonth;

    public Integer getUserPK(){

```

Final Code

```
return user_pk;  
}  
  
public String getUserName(){  
    return user_name;  
}  
  
public float getHours(){  
    return hours;  
}  
  
public String getYearMonth(){  
    return yearMonth;  
}  
  
public void setUserPK(int userPK){  
    this.user_pk = userPK;  
}  
  
public void setUserName(String name){  
    this.user_name = name;  
}  
  
public void setHours(float hours){  
    this.hours = hours;  
}  
  
public void setYearMonth(String yearMonth){  
    this.yearMonth = yearMonth;  
}
```

Final Code

```
public userHours(){}

public userHours(int userPK, String name, float hours, String yearMonth){

    this.user_pk = userPK;
    this.user_name = name;
    this.hours = hours;
    this.yearMonth = yearMonth;

}

public ArrayList<userHours> userHoursList(){

    //creates new arraylist of type userHours

    //select query retrieves user primary key, user name and the month and the total hours worked
    //in that month from the users and dailyhours tables

    //groups and orders the data by month

    //while loop creates new instances of the userHours class with the values retrieved from
    database

    //then adds each instance to the arraylist

    ArrayList<userHours> userHourList = new ArrayList<>();

    Connection connection = database.getConnection();
    PreparedStatement ps;
    ResultSet rs;

    String query = "SELECT user_pk, username, date_format(date_add(clockin,interval -
    DAY(clockin)+1 DAY),\"%Y-%M\") month, SUM(hours) hours from users,dailyhours WHERE
    users.user_pk = dailyhours.user_fk GROUP BY date_format(date_add(clockin,interval -
    DAY(clockin)+1 DAY),\"%Y-%M\") ORDER BY date_format(date_add(clockin,interval -DAY(clockin)+1
    DAY),\"%Y-%M\");";
}
```

Final Code

```
try{
    ps = connection.prepareStatement(query);
    rs = ps.executeQuery();
    userHours user;
    while (rs.next()){
        user = new userHours(rs.getInt("user_pk"), rs.getString("username"), rs.getFloat("hours"),
        rs.getString("month"));

        userHourList.add(user);
    }
}

catch (SQLException ex){
    Logger.getLogger(admin.class.getName()).log(Level.SEVERE, null, ex);
}

return userHourList;

}
```

}

6.35 userSessions.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
 * license
 *
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package classes;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.sql.ResultSet;

/**
 *
 * @author garmin
 */
public class userSessions {

    public static int insertUserSession(int userPK){
        //creates a new userSession and returns the latest session primary key

        Connection connection = database.getConnection();
        PreparedStatement ps;
        ResultSet rs;
        int thisSessionPK = 0;

        try{
            ps = connection.prepareStatement("INSERT INTO usersessions(user_fk) VALUES (?)");

```

Final Code

```
ps.setString(1, Integer.toString(userPK));
ps.executeUpdate();

ps = connection.prepareStatement("SELECT MAX(usersession_pk) sessionPK FROM
usersessions WHERE user_fk = ?");

ps.setString(1, Integer.toString(userPK));

rs = ps.executeQuery();
rs.next();

thisSessionPK = rs.getInt("sessionPK");

}

catch(SQLException ex){

    Logger.getLogger(userSessions.class.getName()).log(Level.SEVERE, null, ex);
}

return thisSessionPK;
}

public static boolean isSessionValid(int thisSessionPK){

//checks how long it has been since the user was last active

//if it has been over 10 minutes then the session is invalid so user is logged out and has to log back
in

//if it hasn't been over 10 minutes then the last active field in the userSessions table in the
database is updated with the current time

Connection connection = database.getConnection();
PreparedStatement ps;
ResultSet rs;
int timeDiff = 0;
boolean valid = false;

try{
    ps = connection.prepareStatement("SELECT TIMESTAMPDIFF(minute, datelastactive,
sysdate()) diff FROM usersessions WHERE usersession_pk = ?");
    ps.setString(1, Integer.toString(thisSessionPK));
```

Final Code

```
rs = ps.executeQuery();
rs.next();
timeDiff = rs.getInt("diff");
if (timeDiff > 10){
    valid = false;
    ps = connection.prepareStatement("UPDATE posplex.usersessions SET activeflag = false WHERE usersession_pk = ?");
    ps.setString(1, Integer.toString(thisSessionPK));
    ps.executeUpdate();
}
else{
    valid = true;
    ps = connection.prepareStatement("UPDATE posplex.usersessions SET activeflag = true, datelastactive = sysdate() WHERE usersession_pk = ?");
    ps.setString(1, Integer.toString(thisSessionPK));
    ps.executeUpdate();
}
catch(SQLException ex){
    Logger.getLogger(userSessions.class.getName()).log(Level.SEVERE, null, ex);
}
return valid;
}
```