Computing Project

**Analysis**

**Problem**

Outline

My final proposal is a database program that will contain a list of products and information about Laura Ashley products (for example different chairs, sofas and tables). This information includes dates of purchase, length it’s been in storage for, quantity, price and it’s SKU. Another database may be created to contain item returns and other error information. It can be expanded to interact with customer orders and a customer database that displays additional information such as address, items purchased and names etc. This will all be displayed in a GUI to allow ease of access to users. The GUI should be clear and easy to use.

Database would show purchase date so that my stakeholder knows which stock to take. Ideally when a customer purchases and item, they would want to sell them the same item that has been in storage the longest. This way old stock doesn’t stay in the warehouse forever. This will also be linked to a customer orders database so that stock is sent to the correct purchaser and purchase aren’t mixed up. It’s primarily a program to hold information about what products have recently been returned and information about the return such as the fault and who/where it came from. Additional information such as what products are in stock and which have been in stock for the longest amount of time are to make sure that it is replaced with the right item.

Justification

My solution will contain different databases that will help the stakeholders. It will allow them to quickly recognise the item that has a fault, where the item went and check others of the same item for the same fault. It will also help them know what stock they have and possibly need more of.

Decomposition

I will use decomposition to help create my program by splitting the problem up into different aspects. The whole program will consist of at least 3 different databases. I will start by creating the databases. One for product information, one for returns and one for customer information. These databases will be linked by key identifiers. More databases may be needed in order to link the main 3 databases. Then a GUI will be made to allow the user to interact with these databases.

Abstraction

I will use abstraction to help create my program by reduce the contents of the databases to just information that is required. This would mean that all relevant information will be sorted amongst separate databases and all irrelevant information like variation of tender, bank they use or internet browser they used to access website will be removed from this database.

Divide and conquer

I will split the information I need to store in databases into different categories and sets and make multiple different databases.

**Stakeholder analysis**

Overview

My primary stakeholder is my dad, he helps manage the quality of the company's products an. He has asked for a program that will help them assess errors with products, identify patterns or trends, view stock available and view other information. They require this because a system is already in place, but this system doesn’t show all the information they require. This program will put all the information they require into one place.

My stakeholder has asked for a program that shows product information and links it with warehouse stock and information, purchase date and links any customer returns with orders and the product. This can also be expanded to hold less important information such as if they brought the product online or in store and which staff served them. Product information would include unique product codes, price, short descriptions and how long it’s been in storage for. This is important because older stock is sold first. Return information would include who brought and returned stock, when they returned it and a short description of the problem.

My secondary stakeholder will be other employees who are able to access this database. They will use this mainly to keep track of what stock they have and what stock is leaving. They will still require most of the information available to the primary stakeholder, but permissions could be added in to adjust who can view what.

Questionnaire

|  |  |
| --- | --- |
| Question | Type |
| How do you currently keep stock? | Short answer text |
| What information would you like to be able to access? | Short answer text |
| What information do you store about stick before and after it has been sold? | Short answer text |
| Would you want every employee to be able to access the database? | Yes or no |
| If so, would you want restrictions so only certain people can view certain parts of the databases? | Yes or no |
| What kind of restrictions would you want? | Long answer text |
| Who would you want to be able to edit the databases? | Short answer text |
| How important is user interface? | 1-5, 1 being not at all, 5 being very |
| Would you be put off by a basic GUI? | Yes or no |
| How important is integration for returns? | 1-5, 1 being not at all, 5 being very |

Questionnaire answers

What system do you currently use to find errors about certain products?

“We need to go and look at spread sheets that team has put together to find that information. We currently use a central CRM (customer relationship management system) called synergy which the customer service team use to record details of issues on products.

There are a variety of other systems, not one central system, one of these automatically extracts information from systems that various teams manage and compiles it into spread sheets. This allows us to analyse the data and look for global trends and find problems.”

What info would you like to be able to see?

“I’d like to be able to see all the information about products and their faults or returns but most important would probably be the failure per centage per month. Maybe the failure per centage of certain batches as well as where they come from. This would help us consult the factory and find the cause of the errors or failures.”

What information do you store about stock before and after it has been sold?

“Stock is brought in by order from factory so we don’t hold much additional stock. After, we know everything. We know where it went, who got it, age of product, etc.”

Would you want every employee to be able to access the database?

“Yes, the information would be useful for the whole team”

If so, would you want restrictions so only certain people can view certain parts of the databases? (yes or no)

“No restrictions. Other than myself and my team, amongst other people, I would want the customer service team and store staff to be able to access it or use it.”

Who would you want to be able to edit the databases?

“When new products are introduced once information is compiled, different teams set up different parts of the information about the products, details such as SKU, size and weight. We use a range sheet, a big spread sheet where each line is a product, different teams have different responsibility i.e weight and dimensions, cost and supplier details and so on. A team in wales put all the information together on the range sheet.

We also use WMS (warehouse management system) to see what additional stock we have in the warehouse. In the warehouse we use a FIFO system, first in first out. This way we don’t leave old stock in the warehouse getting older and older.”

How important is user interface?

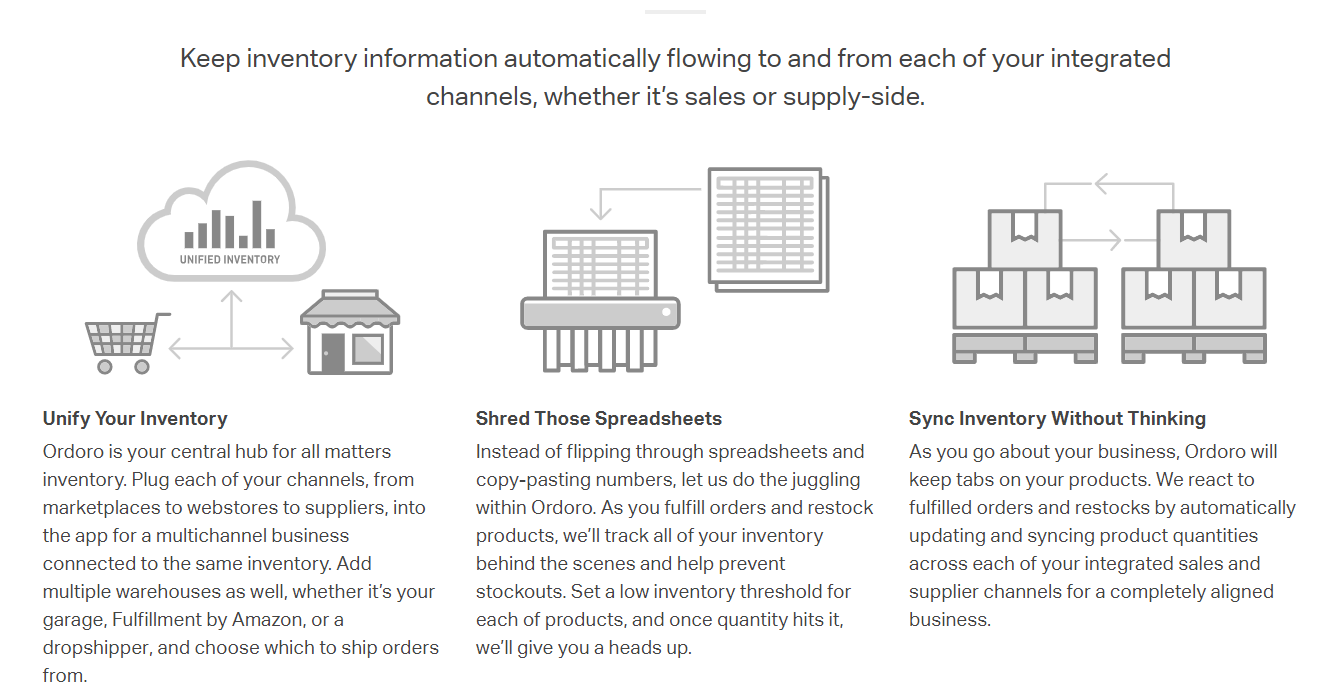
“Very, the current one is very basic and isn’t very good.”

How important is integration for returns?

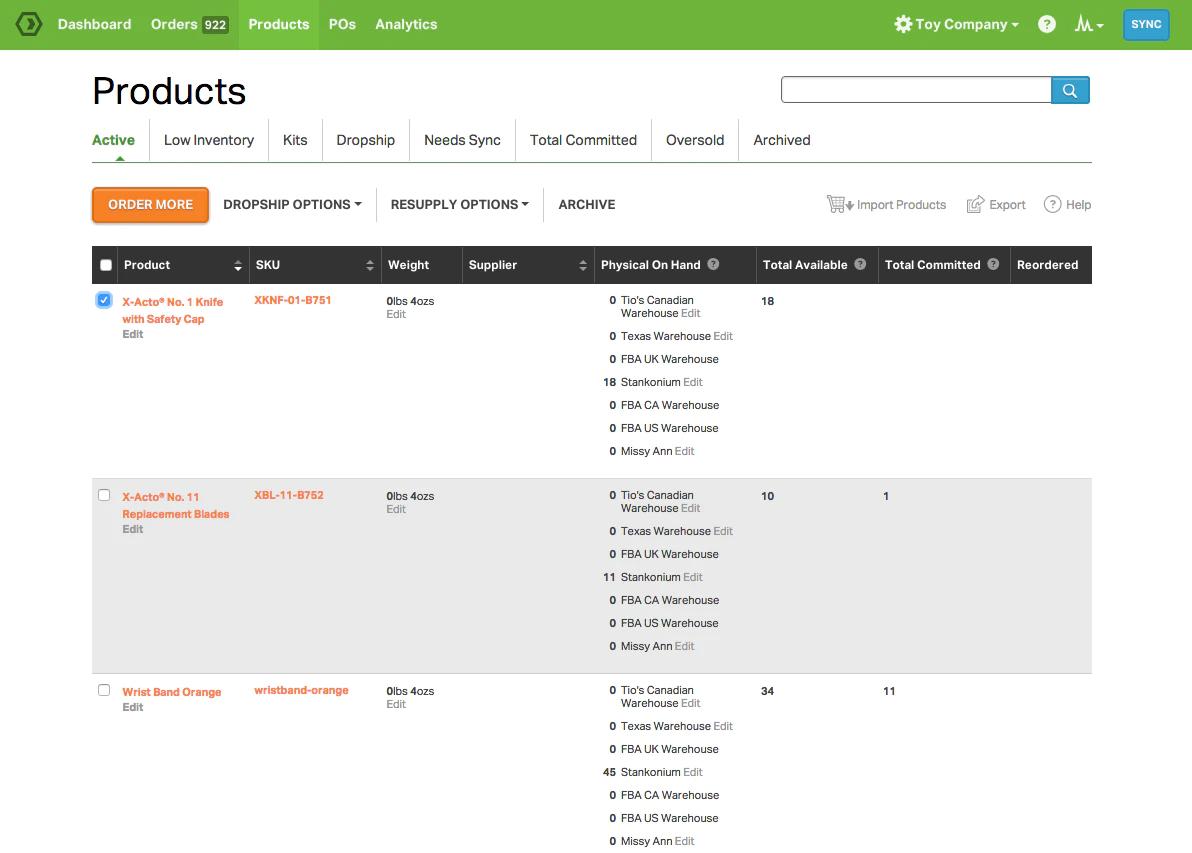
“Very, almost every item that is returned will have a fault. I need to be able to see what has been returned and why, if it’s all in one place then it would make everything much easier.”

**Research**

There are many different types of inventory and stock management software available for companies. Different programs allow the user to do different things, but each one has the same structure, they are all cantered around what stock is available.



Ordoro

Ordoro is software that describes itself as a “control center for inventory management”. Ordoro can automatically update information sucah as product quantities. An automatic system that updates information on other databases is very usueful and could be also be used to manually changed the information of many items at once. Ordor has a lot of information on its display that is well organised but is cluttered.

Product and SKU are already included in my table design.

This table has weight and supplier columns.

In my table, weight and supplier would be useful to include.

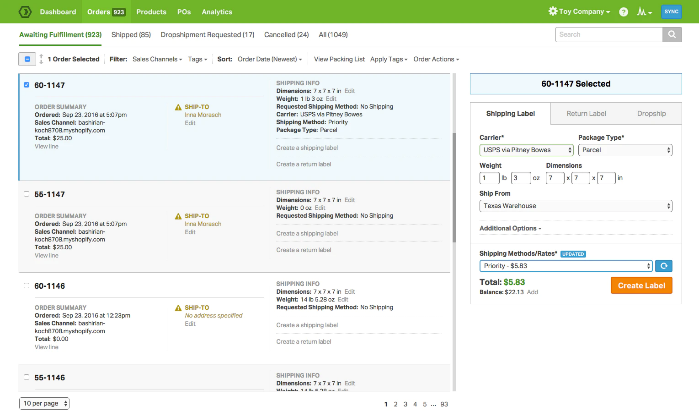
They will be stored in their own separate column instead of being included in products description.

This column shows total amount of stock available across all storage locations.

This column shows how much stock there is left and where the stock is.

This could be a useful feature to implement but

This table has a filter and sort function for the table. These will be useful to implement but might not be doable because of time constraints.



When an order is clicked on, extra information is displayed that is also not necessary.

Even though it isn’t useful information, it is an interesting function to incorporate to show more information about specific items.

This table contains information about the orders such as its total cost, date of the order and shipping info.

All this information could be included but won’t be necessary because the primary function of my program is for assistance with quality control and error checking

Ordoro is useful because of the amount information available through the program but this comes at a cost. That cost being how cluttered, claustrophobic and busy the user interface is. It has some good side features that I can think about implementing but time constraints might hinder this.

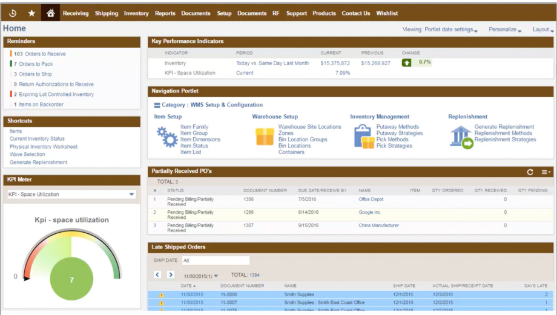
Netsuite ERP

Netsuite ERP is the most popular inventory management system. It is separated into financial management, financial planning, order management, production management, supply chain management, warehouse and fulfilment and procurement. The sections that are most useful are warehouse fulfilment and supply chain management.

 The supply chain management sections

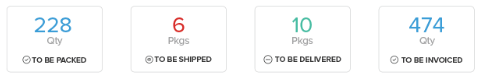
The warehouse and fulfilment section displays information such as orders received, orders to pack, orders to ship, authorised returns, expired items in inventory and items in back order. The visual display for Netsuite ERP is very crowded, like Ordoro, and that is something I want to avoid in my program.

This software has a navigation bar at the top of the window. This is a useful feature that allows easy transfer between windows.

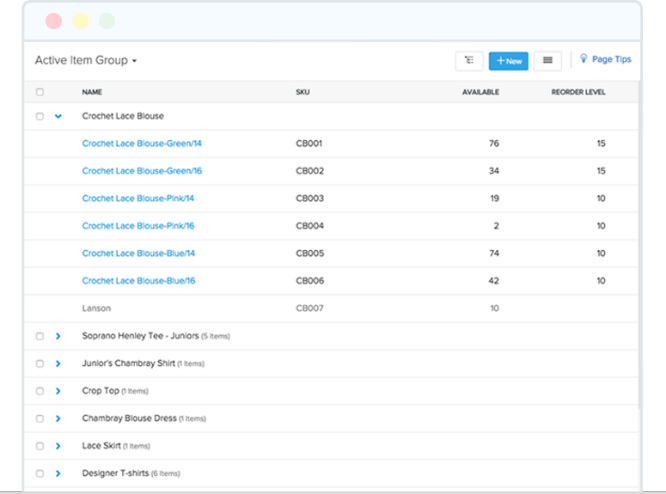


Zoho

Zoho is another inventory management software. This one allows you view stock levels for individual warehouses. The visual display is very simple, simpler than Netsuite ERP and Ordoro, and makes it easy to read and understand. It uses different colours to represent the values for different quantities. This would help split up and separate the values and make it easier to read.



It has a main visual display that shows stock that needs to be packaged, to be shipped, to be delivered, to be invoiced.



This shows stock that is currently available. It doesn’t show where the stock is which.

The additional information shown in this table is the SKU code.

All the programs that I researched include a lot of information that I won't be using such as items in transit, items currently in dispatch etc. My program will have less information so the GUI will not become as cluttered, but it will still need to be organised and clean.

**Solution details**

Essential features

The systems will have to have a database containing product information otherwise, one of the main purposes of the program is to hold product information to help with correcting errors. The primary user will be able to edit the contents of the database and it will have other interactive features to help sort and search through the information for useful data. They need to be able to use the program to view all errors that certain products have had so other databases which hold errors information and link it to product codes will help. Another essential feature is a GUI. A visual display for the databases that can allows the contents to be sorted by certain characteristics.

Possible limitations

Most companies try to integrate smart phones into their employees work so that they can continue working during journeys. Making a program like this as a mobile app as well as a computer program will be unachievable. Users won’t be able to use the program or access the database from their phones, it will be a computer only program.

I don’t expect that the final program will be used afterwards because the program won’t be secure enough for proper use by a company. All information would have to be input manually which would slow down productivity of the department. Time restraints would also prevent me from adding additional features to allow easier use of the program such as the ability to edit the databases while in the program

Hardware/software requirements

Some companies have a wide range of hardware devices so the program should be able to run easily on low specification computers. It will be a program that can only run on computers, so mobile support won’t be needed.

The company only uses windows on the computers in their main office. This means that the it only needs to be coded for windows machines and no other operating systems. The program won't require internet access to run.

Pythons basic run requirements are an Intel Atom or i3 with at least 1GB free disk space and an operating system that is either windows 7 or above, macOS or Linux. All the computers at the office run a version of windows that is 7 or higher and have more than 1GB. These are just the minimum system requirements. The recommended system requirements are an i5 4300M at 2.59GHz, 2 to 3GB of disk space and windows 10.

**Success criteria**

Main menu screen

* Initial menu screen loads when program is initiated
* Menu screen displays options, options presented as buttons
  + Login
  + Signup
  + Databases
  + Exit
* When sign up in button is pressed, user is taken to sign in page
* When login button is pressed, user is taken to login page
* Have login status displayed
* When database button is pressed, user is taken to database page if login status is true
* Have exit button linked to exit function
* When exit button is pressed, program is closed

Signup screen

* Loads when sign up button is pressed on main menu
* Input for username, password and repeat of password
* Could display password as asterisks
* Open database that holds login information for the program
* Check database for username
  + If username is in there, then display message saying that username is already taken
* Compare password and repeat of password
  + If they match, hash the password and store salt and hashed password in table then exit back to main menu, display message saying successfully signed up
  + If they don’t match, get user to re-enter passwords
* When back button is pressed, send to main menu screen

Login screen

* Loads when login button is pressed
* Input for username and password
* Store username and password
* Open user database and search for username
  + If username is in database, hash password and compare them
  + If username isn’t in database, prompt user to re-enter username and password
* If the passwords match take user back to main menu screen and set login status to logged in
* If the passwords don’t match, give the user a prompt
* When back button is pressed, take user back to main menu screen

Select database screen

* When database is pressed on main menu, display database screen
* When database options are pressed, display database in the window
* Create report button, when pressed opens drop down menu with a lot of options
* When options are pressed, construct sql statement
* Use statement to gather data and place into word document or spreadsheet
* When back button is pressed, take user back to main menu

**Design**

Main Menu

Customer info

Product info

Error info

Select database

Log in

Sign up

This is a diagram showing how the user can move from screen to screen. The arrowless lines going from the select database to the names of the databases themselves means that the screen doesn’t change, the databases are just displayed on the select database screen. The option to change between databases is.

Database design

Custom info

The table will hold information about the customers who have purchased products from the company. Each customer will have an ID and their unique identifier, each ID will be unique. It will also contain their first name, last name, address, post code and return ID. Return ID is used to link the customer to the order they returned which is held in error info.

Error info

The error info table will hold information about returned orders. Return ID is used to link order to customer. This table also includes the SKU of the item, customer ID, type of error and date of return. Return ID is unique for each error.

Product info

Product info contains the products SKU, which is unique for each product, the name of the product, its quantity, its price and a short description about the product.

User info

This table won’t be able to be viewed by the user at the database selection screen. It will hold all the user details to log in with. It will hold username, hashed version of the password and the password salt.

The password will be stored as a hash so that if someone gains access to the database that shouldn’t, they won’t be able to access users accounts with the info they gather.

Main Menu

Log in

Sign up

Databases

Exit

Not logged in

Main Menu

Main menu screen. Button presses link to the login, signup or database selection windows. The last option is an exit button, when clicked it will close the program. A logged in notification system can be placed in to indicate whether the user is successfully logged in or not logged in at all.

Not logged in

Says that user isn’t logged in. When no one is logged in then the option to change to the database menu is blocked.

Logged in as Bob McBob

Says that use Bob McBob is logged in. When a user is logged in then databases option is open.

At this stage in the design I am debating whether or not to have the log in notification bar/system on all of the pages or just the main menu.

Pseudocode

mainmenu

if loginstatus is false

display "not logged in"

else

display "logged in as 'username'"

when login button is pressed

load login page

close mainmenu

when signup button is pressed

load signup page

close mainmenu

when database button is pressed

if loginstatus is false

print "not logged in..."

else

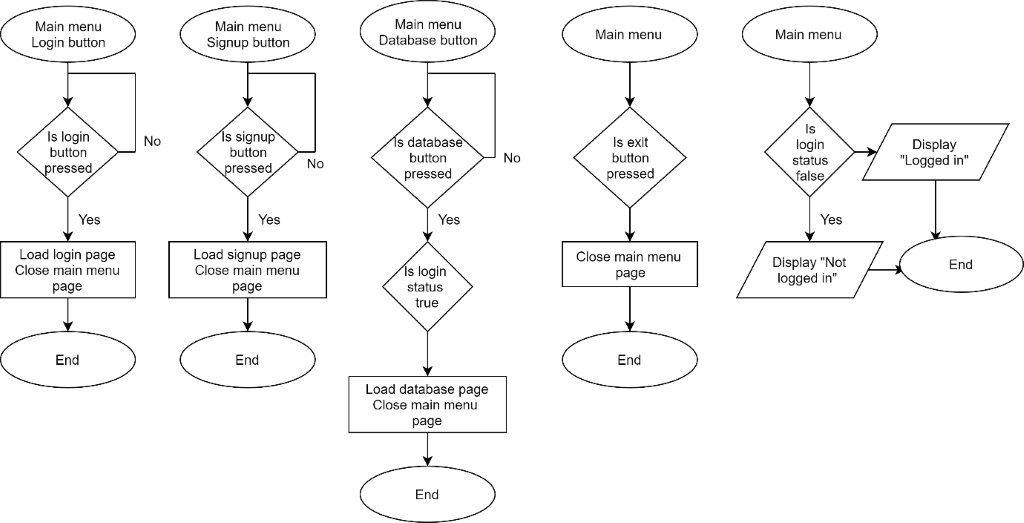
load database page

close mainmenu

when exit button is pressed

close mainmenu

Flowchart



Sign up

Enter username:

Enter password:

Re-enter password:

Sign Up

Back to main menu

This is the signup page. It allows people to create accounts with unique usernames and passwords. The username is checked against the database to make sure that the username being entered isn't already taken. It has two password entry boxes so that the user has to repeat their password on signup. There is a final signup button which triggers the username check. Once the signup button has finished, user is directed back to main menu.

There is another main menu button to allow access to the main menu.

Pseudocode

signup page

is signup button pressed

are user input boxes populated

if yes

store text from username in enteredusername

store text from password in enteredpassword

check username and password meet regex

check password == reenteredpassword

if password == reenteredpassword

connect to userinfo database

search database for enteredusername

if enteredpassword is in database

usernamefound == true

output "username already in database, would you like to login?"

choice1 = userinput

if choice1 == yes or y

load login page

close signup page

else

usernamefound == false

write enteredusername to username field in database

hash enteredpassword

store in hashedpassword

write hashedpassword to password field in database

output "signed up"

loginstatus == true

else

output "password and password that you reentered do not match, please retry"

else

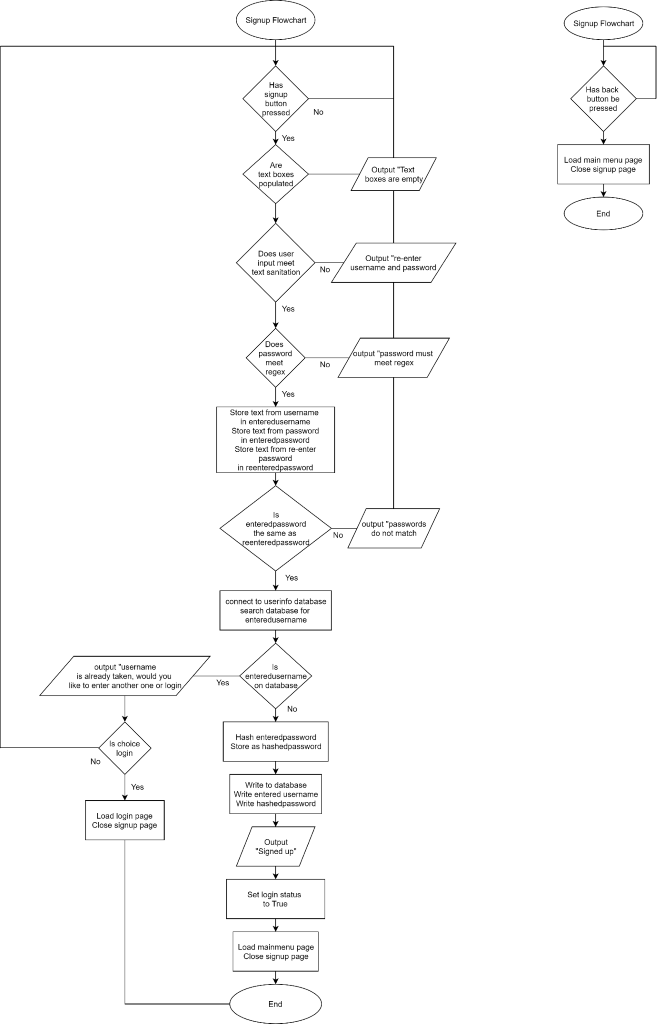
start again

is back to mainmenu button pressed

load mainmenu page

close signup page

Flowchart



Log in

Username:

Password:

Back to main menu

Log in

This is the login page. It will have a section for the user to input their username and password. The database is then searched for the entered username. If the username entered matches one on the database then the passwords are compared. If passwords are the same then you are taken back to the main menu and the logged in notification systems displays you as logged in. If they don’t match then a small window appears next to input area saying that password doesn’t match and that you need to re-enter it.

I haven’t thought about a forgotten password section.

Username:

Password:

Username or password was incorrect

Please re-enter your log in details

Pseudocode

login page

is login button pressed

are user input boxes populated

if yes check input against input sanitisation

if input contains 'unsafe' text

output "reenter username and password"

else

connect to userinfo database

store username in enteredusername

store password in enteredpassword

search database enteredusername

if enteredusername is found

hash enteredpassword

store in hashedpassword

compare hashedpassword to dbpassword

if hashedpassword == dbpassword

output "successfully logged in"

loginstatus == true

load mainmenu page

close login page

else

output "username or password were incorrect, please try again"

empty input fields

start again

else

output "entered username is not found, please try again"

empty input fields

start again

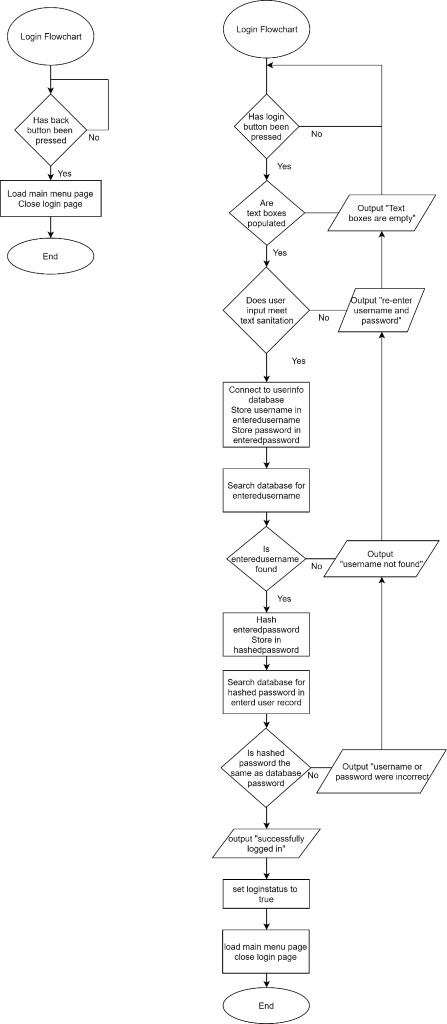
else

start again

is back to mainmenu button pressed

load mainmenu page

close login page

Flowchart

Select database

Back to main menu

Customer info

Product info

Error info

Create Report

This page allows the user to select which database they want to view. When clicked it will display the database in the window. I intend for the displayed database to be interactive; the user can scroll through the database and sort the table by specific criteria.

Generate report

Allows user to set criteria to create report. Takes selected information from database and puts into external text document (word). I was thinking of also implementing a simple search feature, where the user enters what they are looking for and the code generate an SQL statement and puts the result onto an external text document.

There will also be an ability to generate reports by specific pre-sets. I will ask my stakeholder what information he is going to be using the most to create pre-sets that are tailored to his needs.

Report pre-sets:

* Returns in the past month
* List of returns from specific customers
* Returns for specific products

Once a report has been generated there is an option to export to an external document appears. If the button is pressed it will put the data selected into a spreadsheet.

Pseudocode

database page

is customerinfo button pressed

if yes

display customerinfo database on page

else

do nothing

is productinfo button pressed

if yes

display productinfo database on page

else

do nothing

is errorinfo button pressed

if yes

display errorinfo database on page

else

do nothing

is createreport button pressed

if yes

list avaiable presets "returns in past month, list of returns from specific customers, returns for specific products"

choice2 = preset chosen

generate sql statement using preset

output result on display

generate export button

if export button pressed

create spreadsheet with infomation from preset

else

do nothing

else

do nothing

is back to mainmenu button pressed

load mainmenu page

close database page

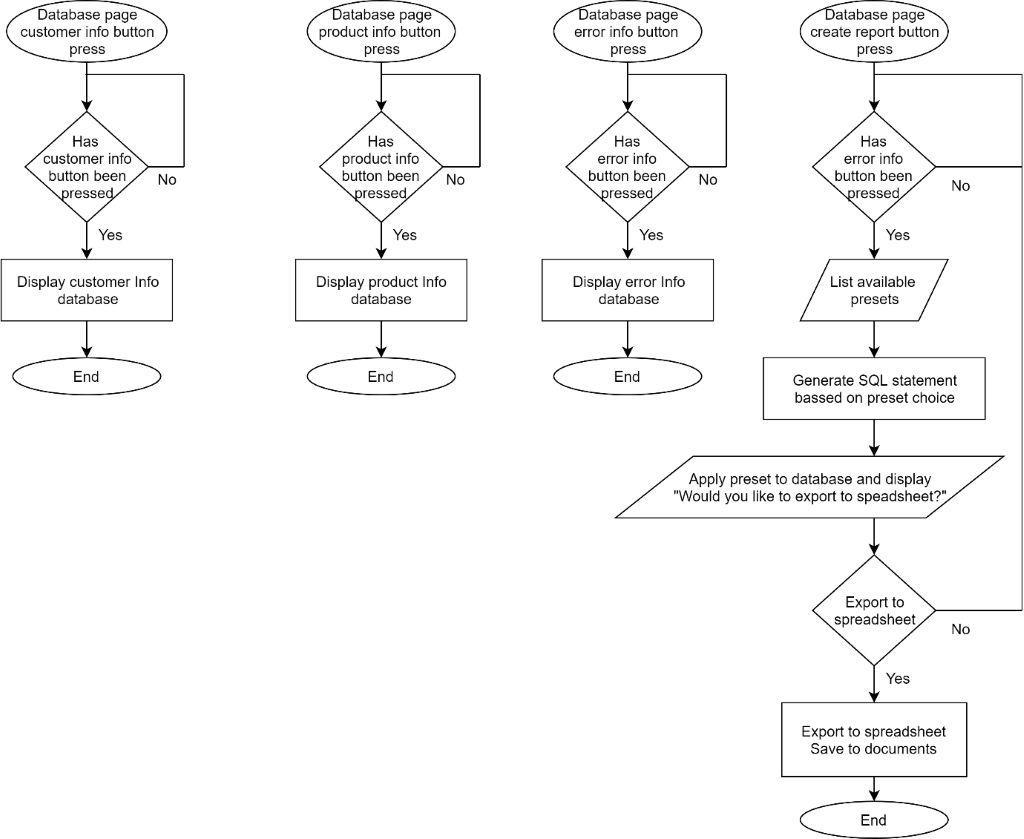
Flowchart

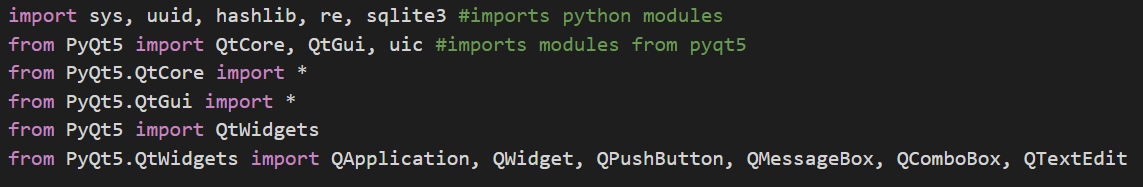
Table of variables

|  |  |  |
| --- | --- | --- |
| Name | Data type | Value |
| loginstatus | Boolean | True/false |
| username | String |  |
| password | String |  |
| dbusername | String |  |
| dbpassword | String |  |
| enteredusername | String |  |
| enteredpassword | String |  |
| reenteredpassword | String |  |
| hashedpassword | String |  |
| usernamefound | Boolean | True/false |
| choice1 | String |  |
| choice2 | String |  |
|  |  |  |
|  |  |  |
|  |  |  |

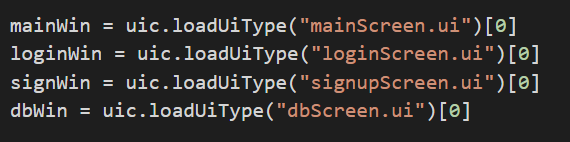
**Development**

Stage 1 – setup

The first step was importing all the modules I need for the password hashing, allowing the user interface file and code to interact and the database.



These lines load the UI files for the different screens into the code so they can used with specific functions. These statements store the UI files in variables. I did this so that instead of repeatedly stating the file name, I can just use the variable name instead.

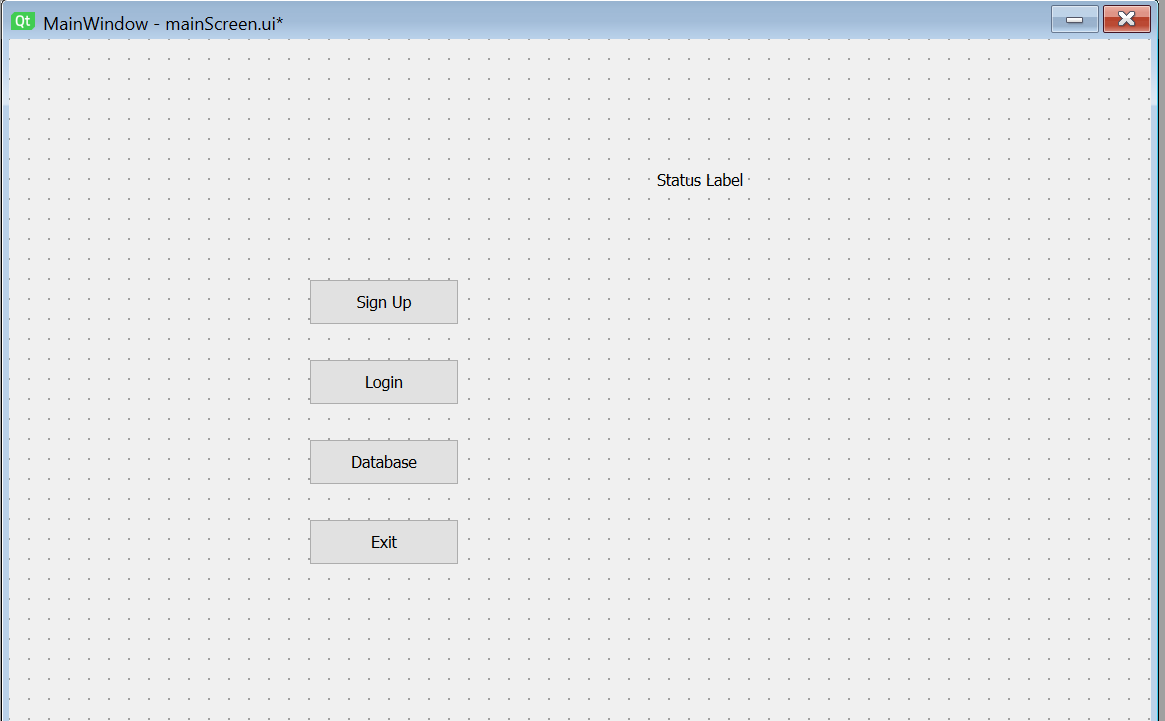


Stage 2 – creating the user interfaces

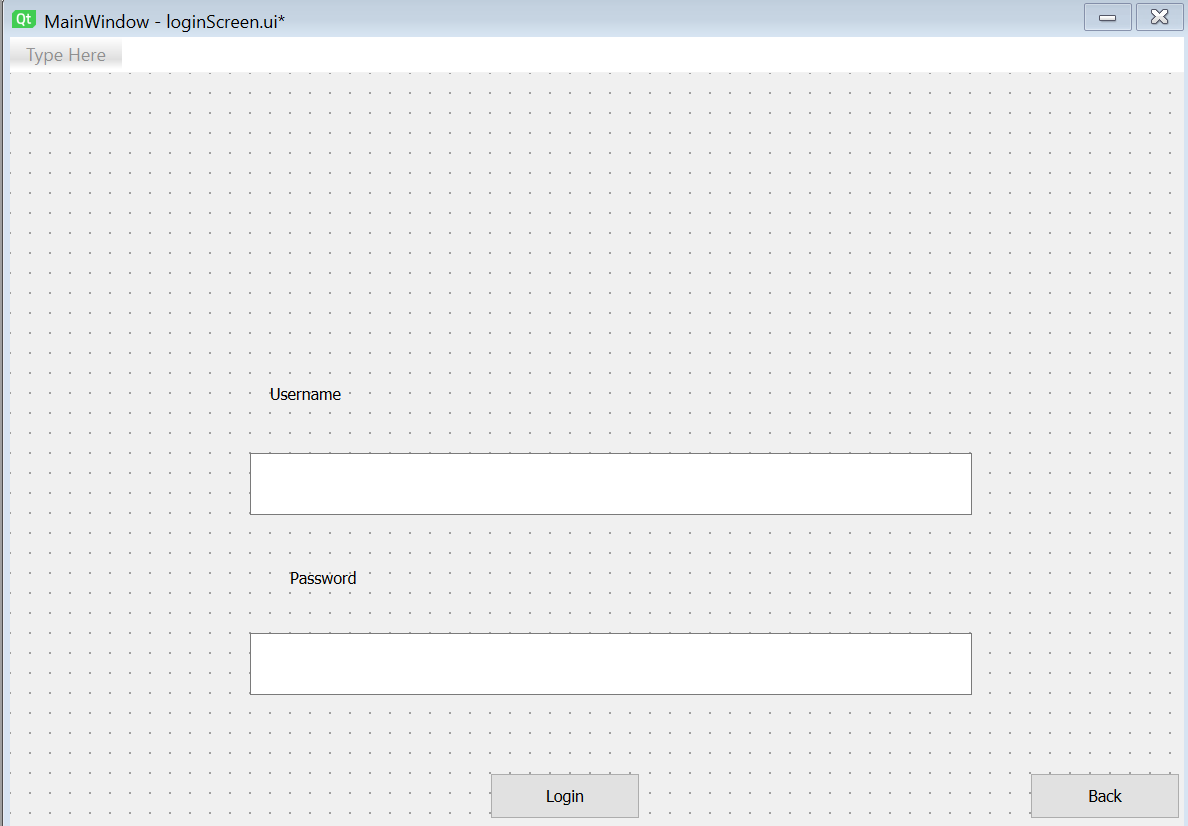
After I had imported all the modules I needed and loaded the UI files into the program, I needed to create the UI files for the program. I started by just making the screen for the main menu, login and signup page. The screens I made were direct copies of the screens I made in my design. I decided to get the login, signup and main menus parts of the code working before I started on the database screen and code.

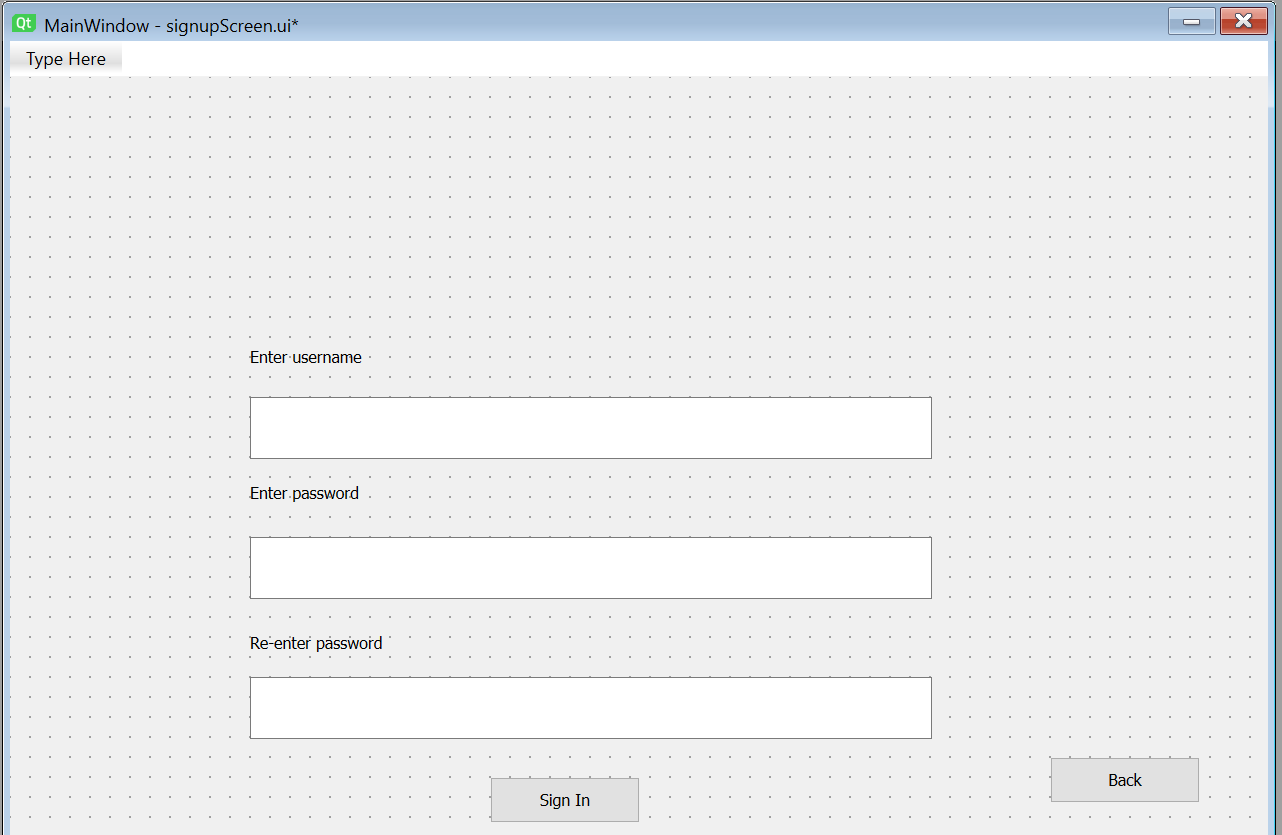
To create the screens, I used pyqt5 designer

Main menu screen



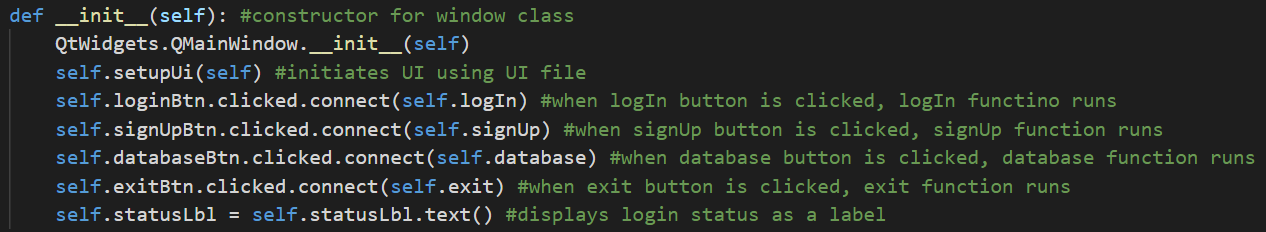
Login screen



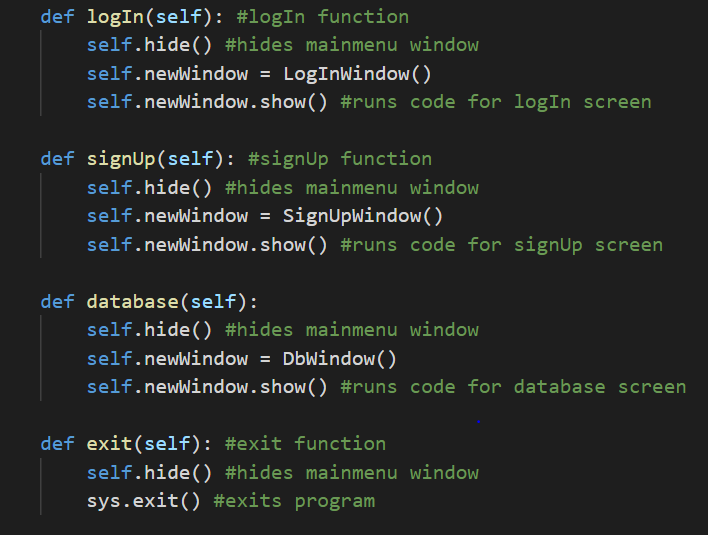
Signup screen

Stage 3 – first screen

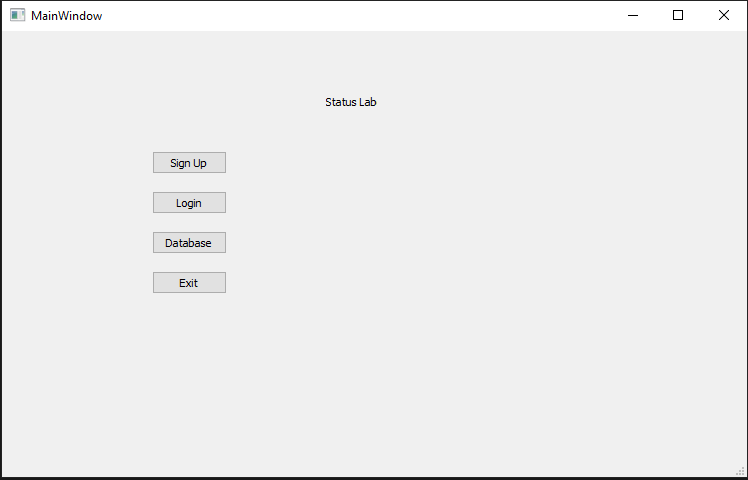
I started off by coding the main menu. From here, the user can access any screen available. I decided to do this screen first because it is the easiest to code and would also be needed to test the back buttons for the rest of the screens.

I started off by initiating the class for the screen.

After I initiated the class, I started coding the events the trigger when the buttons get pressed



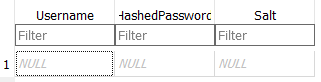
The only testing that could be done at this point was opening the window. Whenever any buttons were pressed other than the exit button, the program would crash because I didn’t have any code for the other windows to run. When the program was run, this is what appeared:



Stage 4 – creating the database

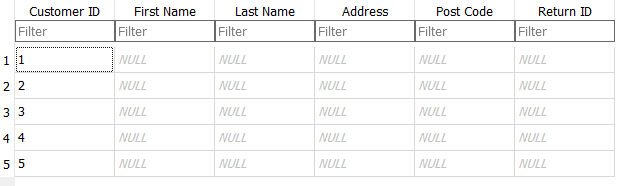
Once the main menu was coded, I needed to have a database to read from and write to. Originally, I was just going to have a separate database for the account information table. But I decided to have all the different tables in the same database.

The first table was the user info one. It only needed 3 columns, one for username, one for hashed password and one for the salt. The password is stored in its hashed form for security and the salt is also held in the table to be used during logging in.

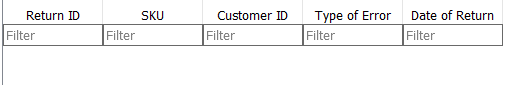


The next tables were the only other tables needed: the customer ID, error info and product info tables. These were all made to hold the same information as intended in the design section of the write up. The tables share items such as IDs, these are foreign keys and will help link the tables together.

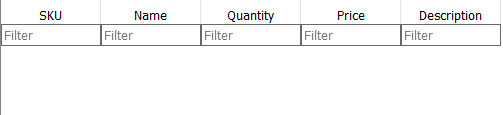
Customer info



Error info table

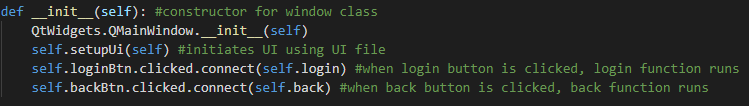


Product info table

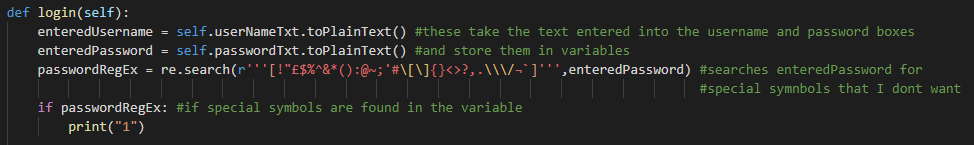


Stage 5 – login screen

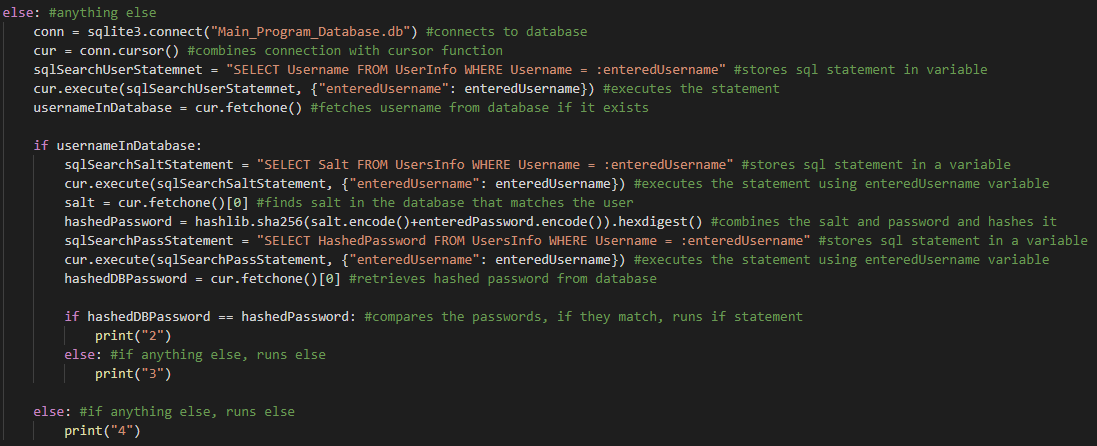
The next screen I coded was the login screen. I did this so I could test that information was correctly being stored in the database.



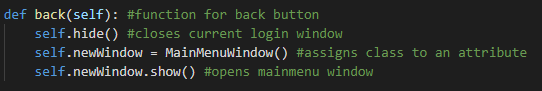
After the class was initiated, I created the login function. The login function will contain most of the code for the class. The function will take the user input, connect to the database, search for username and password and allow entry to the program.



The start of the function just takes the username and password from the input boxes on the UI and stores them in variables called enteredUsername and enteredPassword. It also searches enteredPassword using the regular expression. If the symbols in the regular expression are found in the variable then an if statement is triggered. The statement isn’t coded yet, there is just a print statement to check that it triggers correctly, but it should cause a pop up to appear saying something like “password contains forbidden characters”. If enteredPassword doesn’t have any of the “forbidden” characters then the program connects to the database.

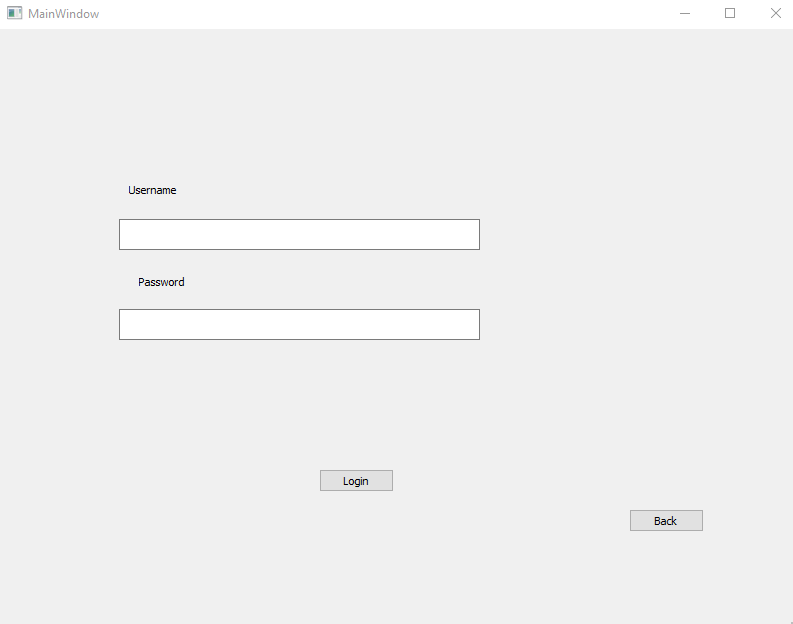


This part of the function connects to the database and searches for the username that was entered. If the username is found, it goes on to take the salt from the database, hashes the entered password and checks if the hashed entered password and the hashed password in the database match. If they do match then the code will continue, if they don’t match then the user will have to re-enter their login details.



The last part of the login screen so far is the back button. The button is linked the back function and this just closes the login window and opens the main menu window.

When the code is run the windows appears as it should and the back button to the main menu works. Once the signup page is working, I can test whether the login button works or not.



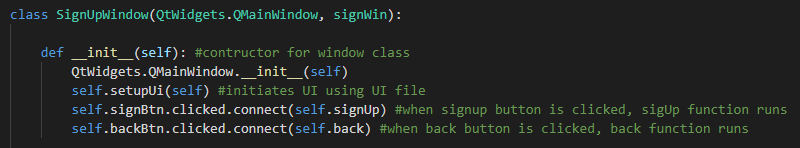
When testing the login screen, I used the username Yes1234567# and the password yes123. If the username and password are correct then a “2” should be outputted otherwise I should get a “4” or a “3”. A “4” means that the username isn’t in the database and a “3” means that the hashed passwords don’t match.



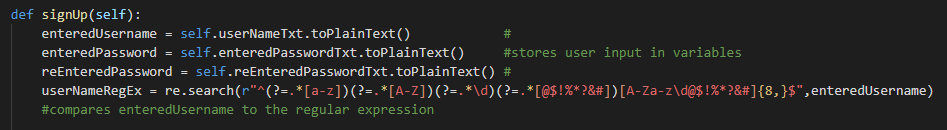
A “2” was outputted meaning the login system works.

Stage 6 – signup screen

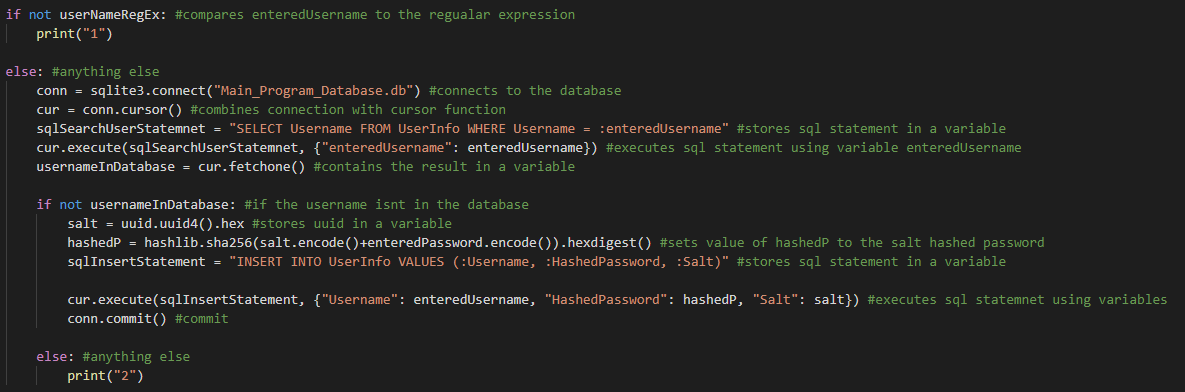
The signup screen is the screen that will let users create accounts and store their login details in the database.



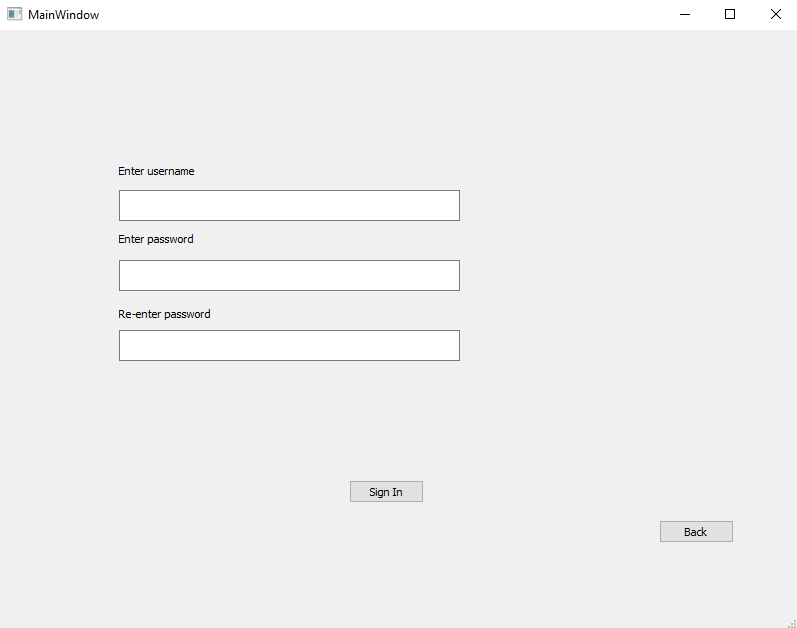
The first function after the class was initiated was the actual signup function which triggers when the signup button is pressed. This function will take the entered username and password, search for the username in the database to make sure that the user didn’t already exist then hash the password.



The first part of the function stores the users entered username and password in variables. The username is then searched with a regular expression so that it doesn’t contain certain characters and is of a certain length. This works as input sanitation.

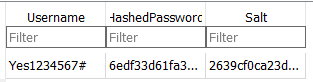


This part of the function connects to the database if the username meets the regular expression. After that, the database is searched for the username to check if a user with that name already exists. If the user doesn’t exist then then it goes on to combine the password with a salt and hash it. The salt is also stored so that it can be used when logging in.



When the code is run the page appears as it should and the back button is fully functional. I tested the code with 2 different usernames: Yes1234567# and just yes. If the regular expression works as I understand it to then only Yes1234567# should be allowed through and if yes is used, the program should return “1”. The password used doesn’t matter because there is no regular expression but I used yes123.

This worked as expected. When Yes1234567# was used, it was saved into the database with the hashed password and when yes was used the code outputted 1



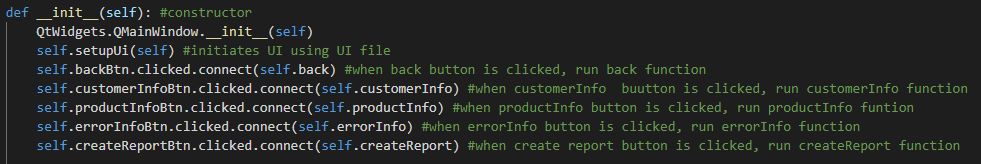
User data being saved in the database



1 being outputted when yes was used as username

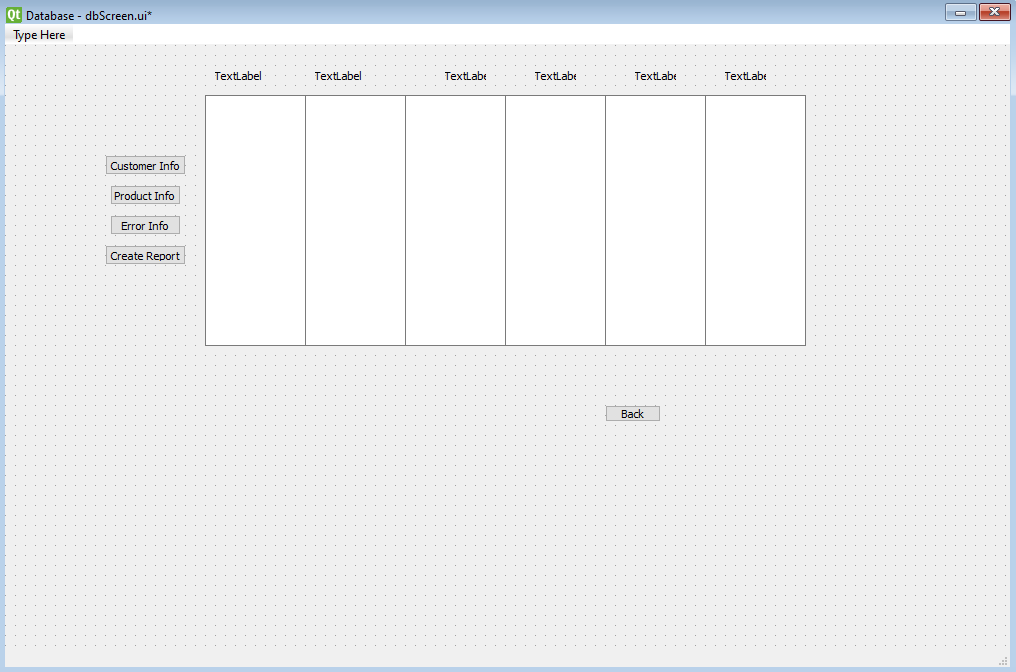
Stage 7 – database screen

The last screen was the database screen. This was probably the most difficult to figure out how to code but was probably one of the quickest too code.

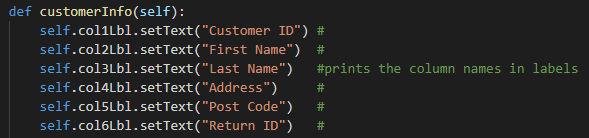


This part of the code linked all the buttons on the page with functions so that when they are pressed, the correct code runs.

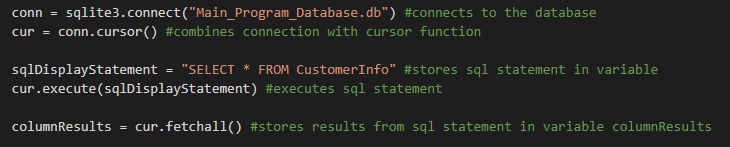
Once I had the class created, I started on the screen. I searched for direct ways of displaying databases using pyqt5 but found out that I can use labels and text boxes to display the table.



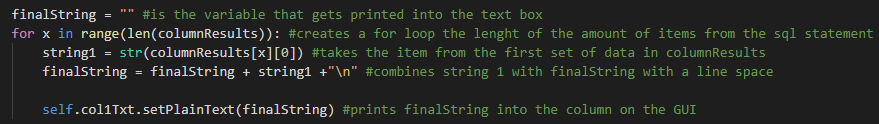
The GUI ended up looking like this. In pyqt designer.



To display the names of the columns on the table I put the name changes in the function. The titles displayed depended on which table was selected through the button pressed.



The next step was extracting the information from the table in the database. I did this by using an SQL statement to select all of the data from the table and store the results in a variable called columnResults.

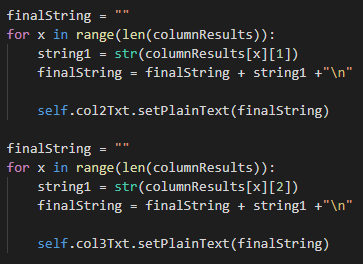


This is how the data would end up in the table. It takes all the data from the first column in the table and turns it into a string with an indent after each line. This is the code for one column but for the rest of the columns this code just needed to be copied and it slightly modified.

The 0 needs to be increased by 1 for each new column.

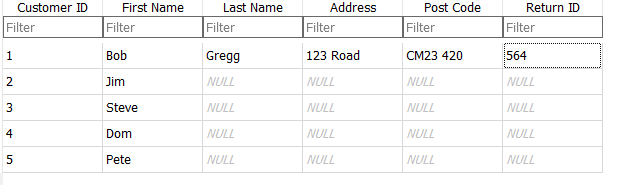
The 1 also needs to be increased by 1 for each new column.

So the next 2 columns end up looking like this

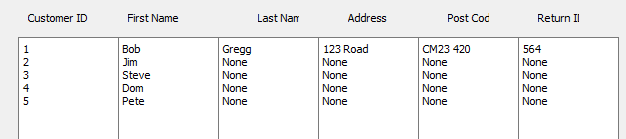


To test whether it worked or not I put some random data in the customerinfo table. When the code was run and the customer info table was selected the correct data should be displayed.

Table of data



How it was displayed in the GUI



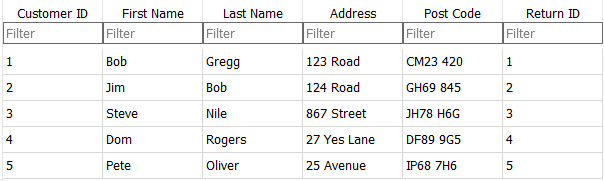
When the code was run the data was displayed in the correct way. The rest of the code was easily repeatable for each table.

**Evaluation**

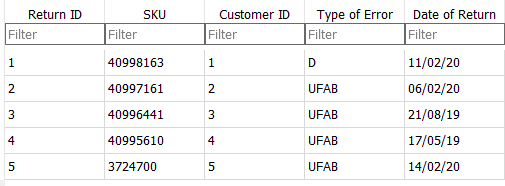
Testing with real data

The testing that could be done with real data was rather limited because the program isn’t complete but real data could still be put into the tables. I used DB Browser to put data in. The data used was real data from the stakeholder. Obviously, I can’t use real customers details so that was all made up.

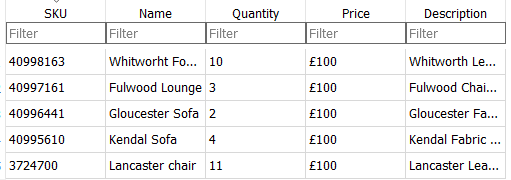
After acquiring the real data, I realised that I’d made my tables small, the spreadsheets that they use contains much more information about the errors and returns. Their table holds SKU, supplier, item range, item description, fabric SKU, fabric colour, order number, store, regen date, inspection, inspected by, qty ordered, fault description, fault code, top reason, top\_co, qa\_x\_ref, liability, SOD order, entry date, post code, channel, customer name, shipped date, level, shape, PO and acxess order.



This shows the customer info table with fake names and addresses entered. I wasn’t allowed real customers names.

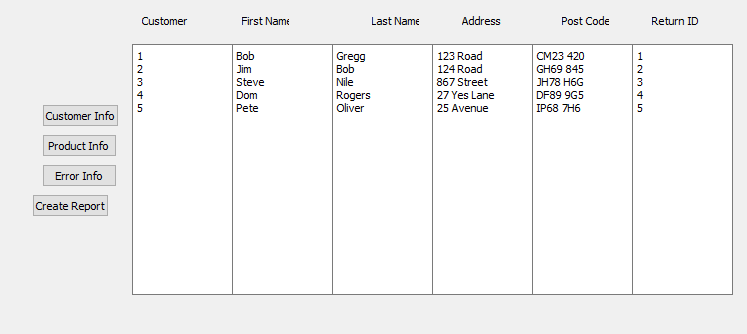


This is the error info table, containing real data. For type of error I used the qa\_x\_ref from the original data which is a error or fault code. D stands for damage and is normally a problem with the structure, spills or marks from shipping. UFAB stands for fabric or leather issue. This is when there is a problem with the fabric of the item, either it being wrong or falling off.

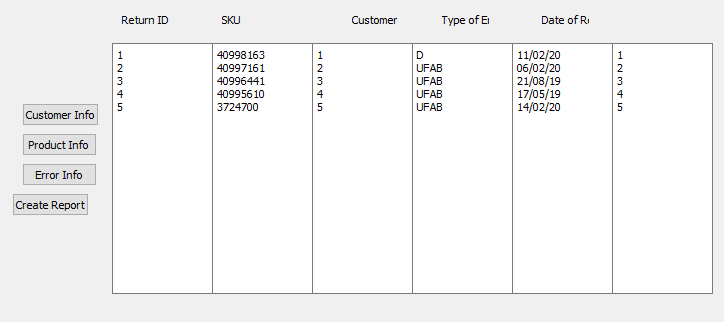


This is the product info table. I took the top five items from the original table that I was given. The table didn’t contain the price so I just set every item to £100 even though that, like the customers, is unrealistic.

When the real data was entered into the tables, everything the program was coded to do worked as intended and didn’t crash. The tables were correctly displayed.

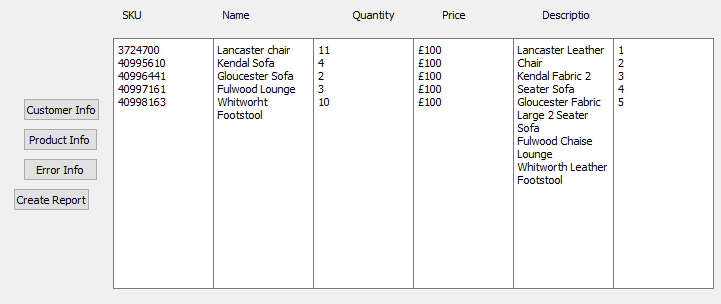


Result of customer info button press

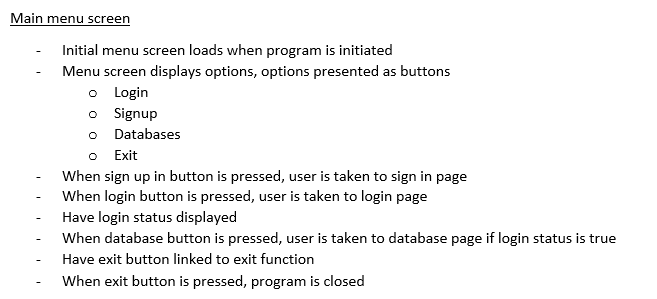


Result of error info button press

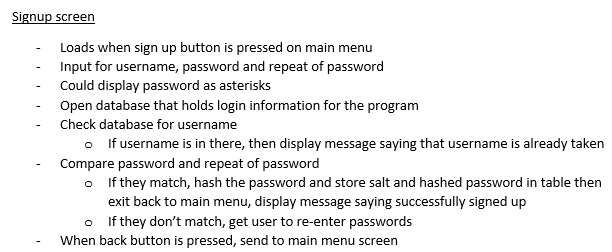
The only problem that was encountered was that items like descriptions couldn’t fit in the text box without moving onto another line. This meant that the product descriptions didn’t line up with the rest of the rows.



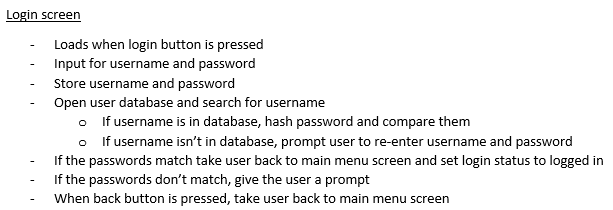
Success criteria



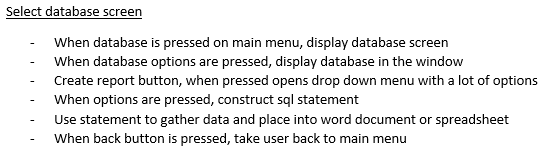
The main menu screen currently meets all of the success criteria except for the parts relating to the login status which isn’t coded. The database button does work, just irrelevant of the login status and login status isn’t displayed.



The entered passwords aren’t displayed as asterisks, this would make the program more secure though and the same thing could also be implemented into the login screen. The program doesn’t compare the entered password with the re-entered password and the user isn’t shown any pop ups when they should because they aren’t coded and if they signup successfully they aren’t taken back to the main menu.



Almost all of the success criteria is met for this screen but login status isn’t changed when successfully logged in, prompts aren’t given when passwords don’t match and the user isn’t taken back to the main menu.

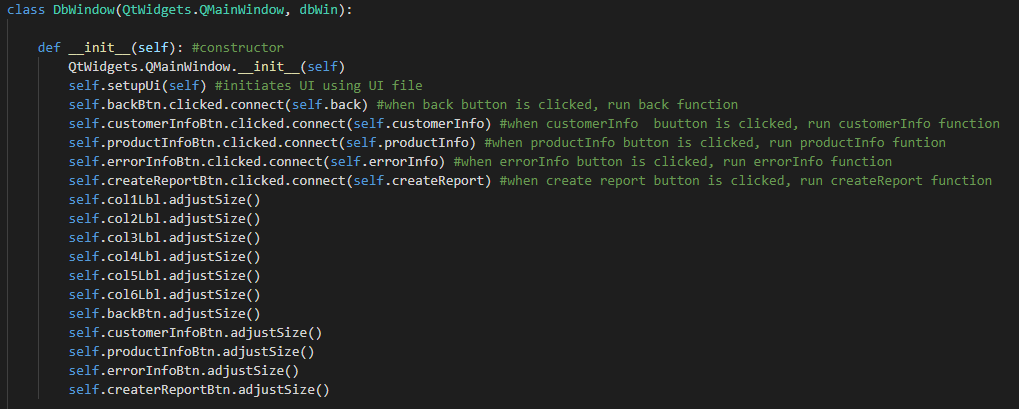


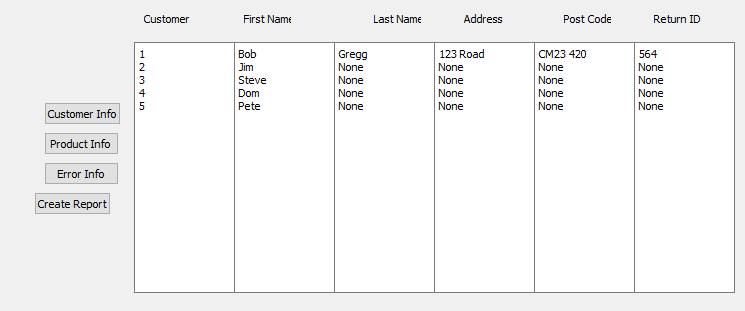
Only the first two and the last item from the success criteria were completed. The rest simply wasn’t coded in.

Problems

The first problem I noticed was that all the windows have MainWindow at the top of them except the database page. This is down to me not realising how to change this until I got to constructing the database page.

With some of the titles and buttons, parts of the words are missing, this is fixed by using .adjustSize() in the constructor of the class.

The start of the database window class would look like this instead which the .adjustSize() section.



And the actual window now looks like this.

Another problem is that the database display part of the code can be simplified by putting all the separate for statements into 1 and increasing the numbers as part of the for statement.

Two very small problems are that the signup button on the signup page says sign in which means the same thing as log in and needs to be changed and that the buttons are often not the same size and not aligned which makes the GUI look clumsy and not very well put together.

The Error Info table doesn’t need to have the Customer ID column. It holds no purpose in that table because the return ID links the return to the customer.