Caleb Johnstone

Grade 11

Information Technology

Practical Assessment Task – Phase 2



Table of Contents

[Data Dictionary 3](#_Toc492222525)

[Database Design 6](#_Toc492222526)

[Entity-relationship diagram 6](#_Toc492222527)

[Data Capturing 6](#_Toc492222528)

[GUI 6](#_Toc492222529)

[Log In Form 6](#_Toc492222530)

[Sign Up form 6](#_Toc492222531)

[Ordering Form 7](#_Toc492222532)

[Order Details Form 7](#_Toc492222533)

[Shopping Cart (draft orders) 8](#_Toc492222534)

[Administrator Page 8](#_Toc492222535)

[TOE Chart 8](#_Toc492222536)

[IPO Table 11](#_Toc492222537)

[Input 11](#_Toc492222538)

[Validation / Error Catching 12](#_Toc492222539)

[Processing 13](#_Toc492222540)

[Output 16](#_Toc492222541)

# Data Dictionary

|  |  |  |
| --- | --- | --- |
| **Var / Array / Text file** | **Data type / Number of items etc.** | **What will it be used for?** |
| questions.txt | Text file  5 items | To store the five questions to be used as a back-up, if the user forgets their account password |
| sAnswer | string | The answer to the chosen question will be stored in the database as the field “Answer” in “tblClients” |
| iQuestion | integer | The question number chosen to identify the question that was chosen when creating the account |
| sQuestion | string | To store the question of the corresponding question number selected and display in a message dialogue |
| sClient | string | To store the client’s wanted username |
| sAddress | string | The address that will be delivered to, if they choose for the order to be delivered instead of collecting the order at the LAN party |
| sPassword | string | To store the client’s password that will be paired with their username |
| sRetype | string | To compare with the first password entered. This password will be used to confirm that the user has entered the password correctly, as they might mistype and not be aware of it because of the password being displayed in stars |
| bDeliver | boolean | To store whether the client wants an order to be delivered to them or collect it themselves at the LAN party |
| iNum | integer | To store the quantity of a product that a client is ordering |
| OrderDate | date | To store the date when the client placed an order |
| cPart | char | To identify which part of the order the product is. This will be used to conjoin to the order number. |
| iOrders | integer | To count the number of records |
| sOrderID | string | To store the order ID. This will be the order number (iOrder) conjoined with the part char (cPart) |
| iProd | integer | To identify which product was selected |
| sUser | String | The username entered in during logging in |
| sPasscode | string | The password entered during logging in |
| sProduct | string | Store which product the user wants to buy |
| iBuyNum | integer | Store the number of a specific product to be ordered |
| arrNum | Integer array | To store the number of each product being ordered |
| arrProducts | String array | To store the names of the products being ordered |
| iCart | integer | The number of the draft order from the user’s shopping cart that the user selects to order |
| sFieldSort | string | To store the field that the user will be able to sort by in their order history |
| sPart | string | To store parts of the queries that the administrator can perform |
| sQuery | string | To store the administrator’s entire requested query |
| bDesc | boolean | To sort the user’s order history in descending order rather than in the normal (default) ascending order |

# Database Design

## Entity-relationship diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **tblClients** |  | **tblOrders** |  | **tblMerch** |
| ClientID | 1 | OrderID |  | ProdName |
| Address | ∞ | ClientID |  | Price |
| Username |  | Delivery | 1 | ProductID |
| Password |  | ProductID | ∞ | TotalSold |
| QuestionID |  | Quantity |  |  |
| Answer |  | Date |  |  |

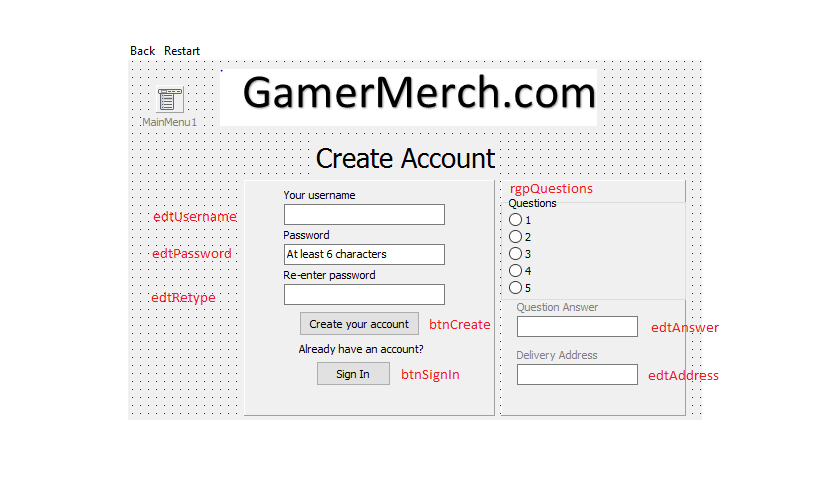
# Data Capturing

# GUI

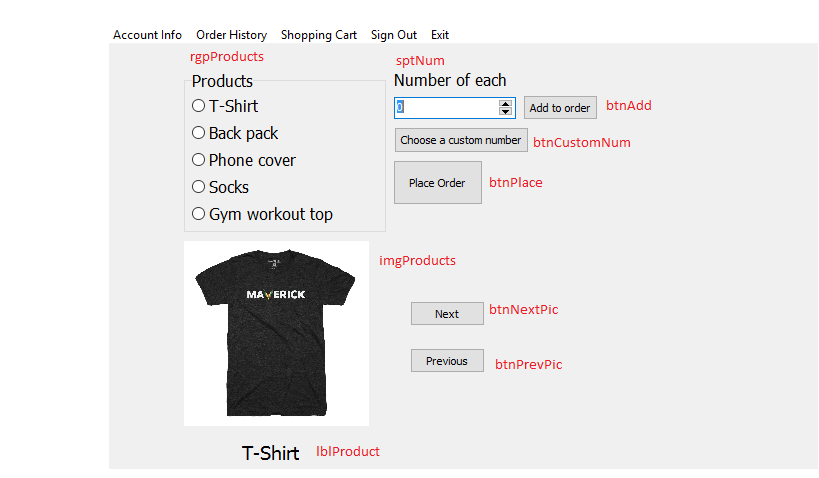
### Log In Form



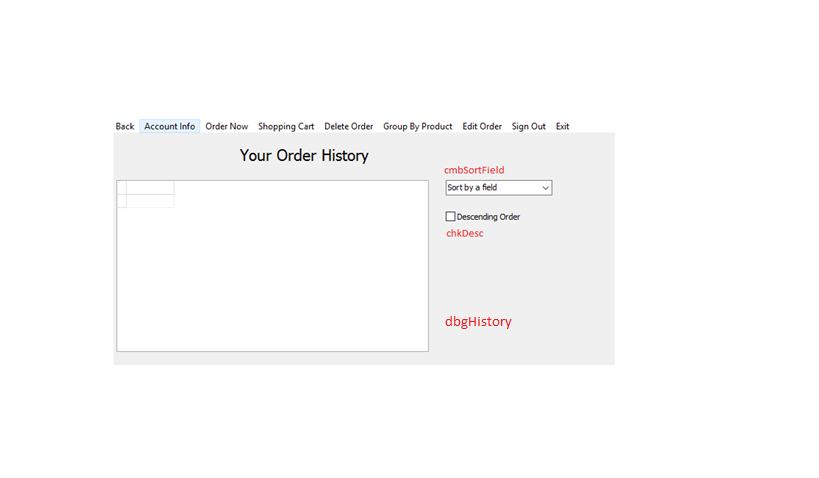
### Sign Up form



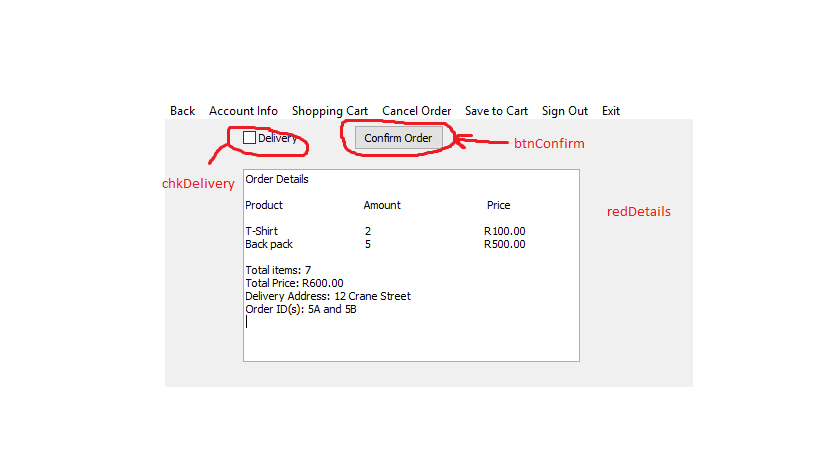
### Ordering Form

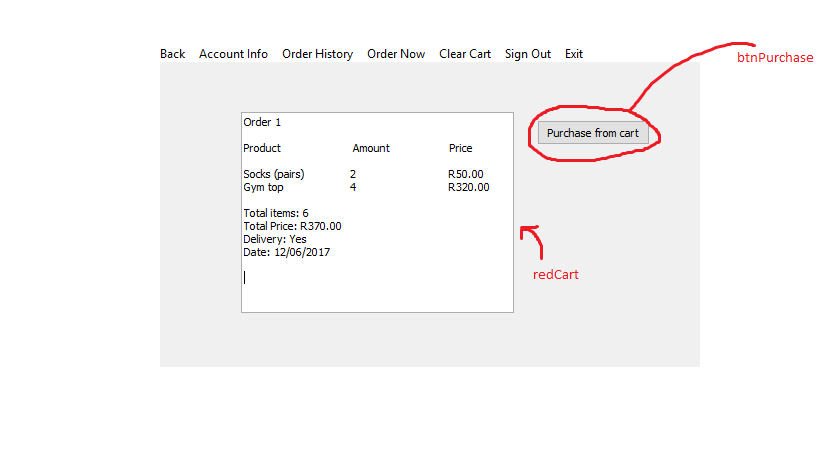


Order History Form

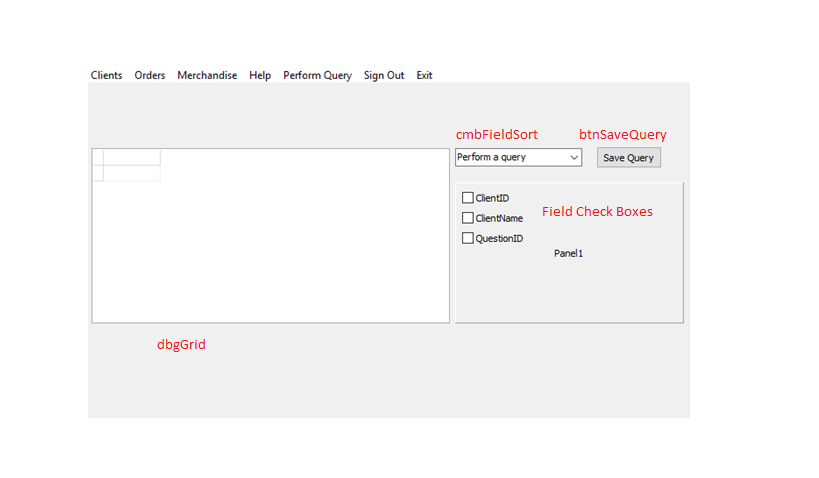


### Order Details Form



Shopping Cart (draft orders)

### Administrator Page



# TOE Chart

|  |  |  |
| --- | --- | --- |
| **Task** | **Object** | **Event** |
| **Get User Sign In Information** |  |  |
| Username | edtUser |  |
| Password | edtPasscode |  |
| User help | btnHelp | OnClick |
| See password being entered | btnSee | OnClick |
| Sign In | btnSignIn | OnClick |
| Sign Up (create an account) | btnSignUp | OnClick |
| **Get User Sign Up Information** |  |  |
| Username | edtUsername |  |
| Set Password | edtPassword |  |
| Confirm password | edtRetype |  |
| Sign Up (create the account) | btnCreate | OnClick |
| Sign In | btnSignIn | OnClick |
| Restart Page (clear information) | Restart - Menu option | OnClick |
| Enter delivery address | edtAddress |  |
| Select a question number to find out what the question is | rgpQuestions  Message Dialogue | OnDrawItem |
| Enter answer for the chosen question | edtAnswer |  |
| **Ordering** |  |  |
| Select products to buy | rgpProducts | OnClick |
| Specify number of each product | sptNum | OnChange |
| Add a product to a pending order | btnAdd | OnClick |
| Place Order | btnOrder | OnClick |
| Choose a custom amount of a product | btnCustomNum | OnClick |
| Display images of the products | imgProducts | btnNextPic. OnClick  btnPrevPic. OnClick |
| Change image to the next product in the list | btnNextPic | OnClick |
| Change image to the previous product in the list | btnPrevPic | OnClick |
| Change the label of the product image | lblProduct | btnNextPic. OnClick  btnPrevPic. OnClick |
| **Display user’s order history** |  |  |
| Sort by a specific field | Filter History - Menu option | OnClick |
| Enter field to sort by | Input Box | Filter History. OnClick |
| Sort ascending or descending | Input Box | Filter History. OnClick |
| **Confirm Order Details** |  |  |
| Cancel Order | Cancel Order – Menu option | OnClick |
| Save order as a draft to the shopping cart | Save to Cart – Menu option | OnClick |
| Get delivery status | chkDelivery | OnClick |
| Confirm that all order details are correct | btnConfirm | OnClick |
| Display order details | redDetails | btnOrder. OnClick |
| Save order details to the shopping cart | Save to cart – Menu option | OnClick |
| **Shopping Cart Orders** |  |  |
| Purchase a draft order | btnPurchase | OnClick |
| Display draft orders | redCart |  |
| **Administrator Page** |  |  |
| Display Clients table | Clients – Menu option | OnClick |
| Display Orders table | Orders – Menu option | OnClick |
| Display Merchandise table | Merchandise – Menu option | OnClick |
| Perform queries on the table currently being displayed | Queries – Menu option | OnClick |
| Select which query to perform | cmbQuery | OnDrawItem |
| Select which fields you want to perform the query on | Field Check Boxes | OnClick |
| Save a part of a query | btnSaveQuery | OnClick |
| Values and operations on the selected field(s) | Input Boxes | btnSaveQuery.OnClick |
| **A user’s order history** |  |  |
| Sort by a specific field | cmbFieldSort | OnDrawItem |
| Sort by ascending or descending order | chkDesc | OnClick |
| Group by product name | Group By Product – Menu option | OnClick |

# IPO Table

## Input

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Part of program** | **What** | **Var Name** | **Data Structure** | **Global/ Local?** | **Object** |
| **Sign In** | Username | sUser | string | global | edtUser |
| Password | sPasscode | string | global | edtPasscode |
| **Sign Up** | Username | sUser | string | global | edtUsername |
| Password | sPassword | string | local | edtPassword |
| Confirmed password | sRetype | string | local | edtRetype |
| QuestionID | iQuestionID | integer | local | rgpQuestion |
| Answer to the chosen back-up question | sAns | string | local | edtAnswer |
| **Placing an order** | Products being bought | arrProducts | String array | global | rgpProducts |
| Number of a product being bought | iBuyNum | integer | local | sptNum |
| Number of each product being bought | arrNum | Integer array | global | “Number of each” spin edits |
| Custom number of a product | iCustom | integer | local | btnCustomNum  Input Box |
| **Order History** | Field name to group by | sField | string | local | cmbSortField |
| Ascending or descending order when displaying the sorted table | bDesc | boolean | local | chkDesc |
| **Order Details** | Whether an order will be for delivery or not (will be picked up at the LAN party) | bDelivery | boolean | global | chkDelivery |
| **Shopping Cart** | Which draft order number the user wants to actually order | iCart | integer | local | btnPurchase  Input Box |
| **Administrator page** | Which query to perform | sQuery | string | local | cmbFieldSort |
| Whether to perform the chosen query on a specific field or not | Eg: bClientID | Booleans | local | Field Check Boxes |

## Validation / Error Catching

|  |  |  |
| --- | --- | --- |
| **What** | **How: Algorithms** | **Message** |
| edtUser | sUser -> edtUser.Text  qryMerch.SQL.Text -> ‘Select \* from tblClients Where ClientName = “‘ + sUser + ‘”’  qryMerch.Open  iNum -> tblClients.RecordCount  if iNum = 0 then  Display -> Message | “That is not a valid username” |
| edtPasscode | (continue from above algorithm)  Else  Begin  sPasscode -> edtPasscode.Text  qryMerch.SQL.Text -> ‘Select \* from tblClients Where ClientName = “‘ + sUser + ‘” and Password = “’ + sPassword + ‘”’  qryMerch.Open  iNum -> tblClients.RecordCount  if iNum = 0 then  Display -> Message  end | “That is a wrong password for your username” |
| edtUsername | sUser -> edtUser.Text  qryMerch.SQL.Text -> ‘Select \* from tblClients Where ClientName = “‘ + sUser + ‘”’  qryMerch.Open  iNum -> tblClients.RecordCount  if iNum = 1 then  Display -> Message | “That user name is already in use” |
| edtPassword | sPassword -> edtPassword.Text  if Length(sPassword) < 6 then  Display -> Message | “Your password needs to be at least 6 characters long” |
| edtRetype | sPassword -> edtPassword.Text  sRetype -> edtRetype.Text  if not sPassword = sRetype then  Display -> Message | “The passwords you type in must match” |
| chkDelivery | If not chkDelivery.Checked then  Message Dialogue | “Are you sure that you don’t want delivery?” |

## Processing

|  |  |  |
| --- | --- | --- |
| **Part of program** | **What** | **How: Algorithms** |
| **Sign In** | Display the password’s characters in stars, to keep it confidential | In the password edit box’s OnChange event:  Begin  sPasscode -> edtPasscode.Text  sKeep -> sPasscode  iCount -> Length(sPasscode)  sDisplay -> ‘’  for k -> 1 to iCount do  begin  sDisplay -> sDisplay + ‘’  end  edtPasscode.Text -> sDisplay  end |
| Notify the user if their desired username is already taken | * Use a select and the record count afterwards to see if the username has been used * If it has been used – display a message saying so |
| See the characters that are being typed in for the password entry | Have a counter that will be increased every time “btnSee” is clicked.  Increase counter  If counter = even then  Begin  Display in stars  End  Else  Display sKeep |
| If the username exists and the password entered is a correct match, the user order page should be displayed | In the sign in button:  frmOrder.ShowModal |
| Click the sign up button – show the sign up form | frmSignUp.ShowModal |
| **Sign Up** | If the username is available and the two passwords entered equal each other, have the required length, then a record must be added to “tblClients” with the other information entered | Using an insert in SQL |
| **Placing an order** | Change the default value of the spin edits when a product is selected | When an option is selected then the default text value of the spin edit will change from 0 to 1 |
| Add the names of the products being bought to an array | arrProducts[k] -> rgpProducts.Items[rgpProducts.ItemIndex] |
| Add the number of each product to an array | arrNum[k] -> sptNum.Text |
| **Order History** | Use string building to determine which field to order by (sort by) | sField -> cmbSortField.Items[cmbSortField.ItemIndex]  qryMerch.SQL.Text -> ‘Select \* from TABLE Order By “‘ + sField + ‘”’  qryMerch.Open |
| If “chkDesc” was selected, then the ordering by should be done in descending order | Same as above, but the following must be added to the select:  + ‘ Desc’ |
| **Order Details** | Set a value for the delivery field, depending on whether “chkDelivery” was selected or not | In the insert, the field called: “Delivery” will be passed the value of chkDelivery.Checked  Eg:  bDeliver -> chkDelievry.Checked  qryMerch.SQL.Text -> ‘Insert into tblOrders (ClientName, ProductID, Quantity, Delivery, Date) Values (“johns”, 1, 2, ‘ + bDelivery + ‘, 12/03/2012)’  qryMerch.ExecSql |
| **Shopping Cart** | Extracting the shopping cart details of the user from a text file called “ShopCart.txt”, by reading from the text file | A loop will be used to extract the data that was stored (written) at an earlier time when the user saved the order as a draft in the shopping cart. The draft orders will be stored in the format:  **ClientID#DraftNumber#ProductID#Amount#Delivery#Date** |
| **Administrator page** | If a check box is selected then that field should be added to perform a particular query on, which will be from “cmbSortField” | Using string building in the forming of the SQL statement to build the wanted query. Taking the text of the check boxes and “cmbSortField” and adding that to the query in the appropriate placing and order |
| Saving the individual parts of a query as a string | Using a button called “btnSaveQuery” to store each part of the query in local string variables |

## Output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Part of program** | **What** | **Purpose** | **Object** | **How: Format/Data Type** |
| **Sign In** | Output an error message, if an user enters an username that is not in the existing database’s table of clients | Only allow existing users to be able to access the rest of the program and have the user to rather create an account | Message Dialogue | Output in the message dialogue that the username does not exist |
| **Sign Up** | When a question number is selected, the corresponding question will be outputted | To be used as a back-up, if the user forgets their password, as a recovery option | Message Dialogue | Output the corresponding question from the text file “Questions.txt” |
| **Placing an order** | Output the pictures of what the merchandise looks like. | So that the user can know what the products they are buying actually look like | imgProducts | Having a counter that will change every time the next and previous picture buttons are clicked. The names of the pictures in the folder will be numbered. This will allow the picture to change when these two buttons are clicked |
| Output the name of the merchandise product currently being displayed | So the user knows what product they are looking at | lblProduct | Using the same counter as above with the array “arrProdPic” and using the counter to display the correct index |
| **Order History** | Output the sorted tables selected by the user | So the user can see the information of the table “tblOrder” sorted. This is so that they can view their own order history according to different criteria | dbgHistory | Using the order by function in SQL and string building with the selected field name and descending order or not |
| **Order Details** | Display a message dialogue to confirm that the user knows they have not selected the delivery option | This will be to ensure that not selecting this option was purposeful and not just user error (the user forgot to check it) | chkDelivery | Once “btnConfirm” is clicked, the message dialogue will be automatically displayed, asking the user if they are sure they do not want delivery |
| Output the details of the order | So the user can confirm that they are going to order what they want and can make any necessary adjustments | redDetails | Output the product name, quantity and price of each product in neat columns.  Also displaying the total price of the order, total number of items to be bought, delivery address and the order numbers underneath |
| **Shopping Cart** | Display the extracted details from the text file: “ShopCart.txt” in a rich edit | Display draft orders to the user so that they can decide to go through with the order or clear the draft from the shopping cart | redCart | Reading from the text file, as in processing, by splitting up the information and storing it in local variables to be displayed in the rich edit |
| **Administrator page** | Displaying the original tables of data from the database (“Merchandise Database”) and showing the results of the selected queries to the user | This is so that the administrator can figure out information and do analysis on the products being sold | dbgGrid | Using the SQL statements constructed in processing |