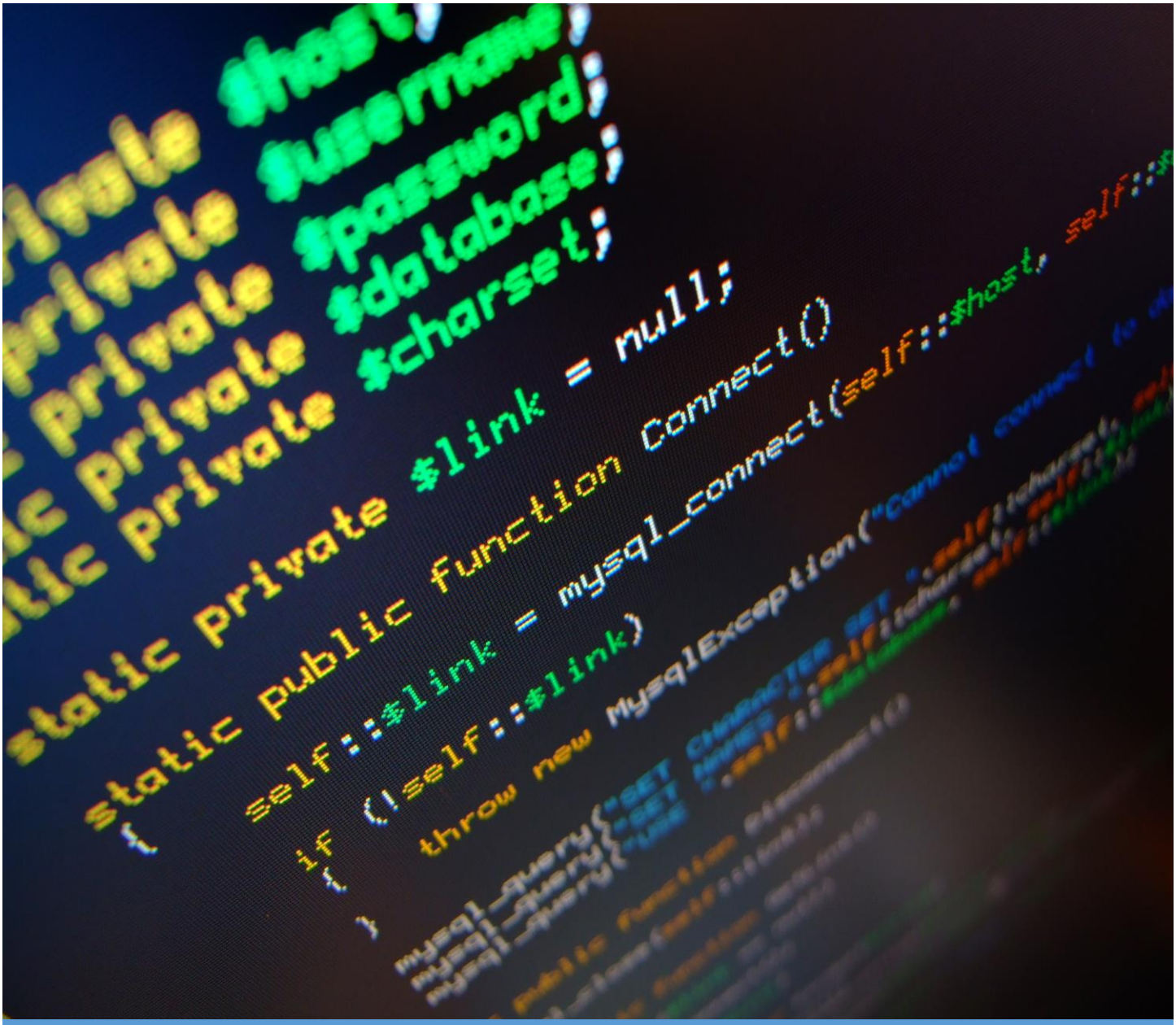


Auction Informatics System

Information Processing Technology

The construction and implementation of a Auction Database System used for the peer to peer interfacing of item bidding.

Lachlan Stevens St. Patrick's College Mrs. Agius

Contents

Auction databases extend to any database that includes the exchange of items in an online environment. These back-ends allow at a fundamental level the inner workings of websites such as EBay and Etsy that each person knows and loves. During the course of this project; a list of very specific processes will be followed (commonly referred to as the 'Software Development Cycle') to produce a database implementation of these Auctioning Systems. Successful implementation will result in a product that is able to incorporated and used to mimic these levels and demonstrate their workings. All database schemas will be implemented using Microsoft Access.

Identification

For this task, the chosen database schema will be the manipulation and implementation of an Auctioning System. This will allow a target audience of St. Patrick's College and all of its subsidiaries; the ability to browse and sell items within the college. This project will run under the name 'Pro Auction Industries'; giving the user of the database a real world insight into how the famed databases of EBay and Etsy are successfully constructed and implemented each day. Within any information system there extends to an incorporation of the Universe of Discourse. The universe of discourse representing the effective communication of data between various pieces of computing; this is helpful due to the fact that computers do not function with the same level of flexibility as is depicted by human communication; with this effective use of communication giving birth to a computing system without any acts of human colloquy and a clear **unambiguous** method of inferring meaning to data.

This then extends to the construction of the various Schema (extended from the word 'schematic') as an outline for the data inside of the resultant information system will be read, as it to quote from the textbook these three levels are:

- **Internal schema** — rules governing physical storage and manipulation of data.
- **External schema** — rules governing how information is viewed by a user.
- **Conceptual schema** — rules governing the type of information that can be stored and the restrictions on how it can be changed.

From the above definition, the resultant universe of discourse of the database auction schema will extend to purely being the access to:

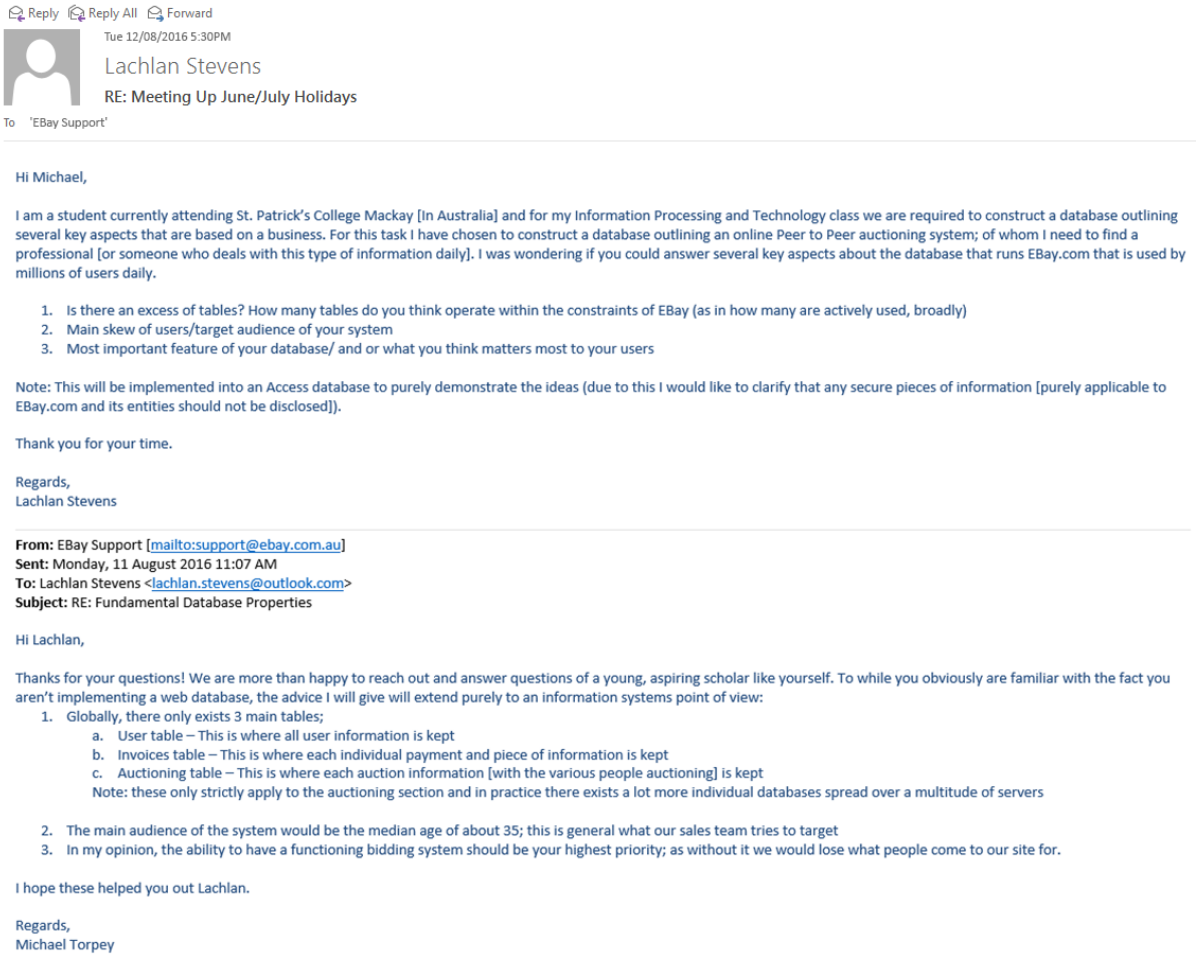
- User information [Name, Address, CNC information]
- Item information [Item, Price, seller]
- Sold information [Item, Price, buyer]
- Bidding information [Item, Bid-Number]

This universe will only extend to inclusion of database (auction) specific items and will neglect any information that doesn't relate to the directly to the act of transfer of items. Having this strict definition of the data that will be implemented into the system is key to maintain and constructing a database that accurately reflects the planned schema.

To gauge an accurate understanding of how all of these elements were designed to work together a storyboard was constructed on grid paper (Appendix 1), this storyboard showed a basic layout in which these elements could be fitted and it allowed an overview as to how the database system will come together; This also allowed the first depiction of the product to be laid out in front so an accurate depiction of what this product will need could be analysed and constructed.

Once design had been settled upon, analysis of the market was needed to see what worked and what didn't work within these specific types of programs. Following from the research, a professional was

contacted as EBay in order to offer various insights as to how their systems work and what their priority's as a company were for their Auctioning System. Refer below for the email conversation between myself and the client.



Reply Reply All Forward

Tue 12/08/2016 5:30PM

Lachlan Stevens

RE: Meeting Up June/July Holidays

To 'EBay Support'

Hi Michael,

I am a student currently attending St. Patrick's College Mackay [In Australia] and for my Information Processing and Technology class we are required to construct a database outlining several key aspects that are based on a business. For this task I have chosen to construct a database outlining an online Peer to Peer auctioning system; of whom I need to find a professional [or someone who deals with this type of information daily]. I was wondering if you could answer several key aspects about the database that runs EBay.com that is used by millions of users daily.

1. Is there an excess of tables? How many tables do you think operate within the constraints of EBay (as in how many are actively used, broadly)
2. Main skew of users/target audience of your system
3. Most important feature of your database/ and or what you think matters most to your users

Note: This will be implemented into an Access database to purely demonstrate the ideas (due to this I would like to clarify that any secure pieces of information [purely applicable to EBay.com and its entities should not be disclosed]).

Thank you for your time.

Regards,
Lachlan Stevens

From: EBay Support [mailto:support@ebay.com.au]
Sent: Monday, 11 August 2016 11:07 AM
To: Lachlan Stevens <lachlan.stevens@outlook.com>
Subject: RE: Fundamental Database Properties

Hi Lachlan,

Thanks for your questions! We are more than happy to reach out and answer questions of a young, aspiring scholar like yourself. To while you obviously are familiar with the fact you aren't implementing a web database, the advice I will give will extend purely to an information systems point of view:

1. Globally, there only exists 3 main tables;
   a. User table – This is where all user information is kept
   b. Invoices table – This is where each individual payment and piece of information is kept
   c. Auctioning table – This is where each auction information [with the various people auctioning] is kept
   Note: these only strictly apply to the auctioning section and in practice there exists a lot more individual databases spread over a multitude of servers
2. The main audience of the system would be the median age of about 35; this is general what our sales team tries to target
3. In my opinion, the ability to have a functioning bidding system should be your highest priority; as without it we would lose what people come to our site for.

I hope these helped you out Lachlan.

Regards,
Michael Torpey

Conversation with EBay representative

As can be noted in the above conversation; the EBay representative outlined that very minimal tables were used to effectively depict the product, as a convolution of tables will lead to a convoluted database design system. Another note to make is that **their** target audience was 30 due to mainly being open to such a broad range of clients, spanning from young ages into the older ages. This allows for (inside of the schema constructed for this project) the age ranges to purely extend to the discussed **Universe of Discourse**. Finally, the EBay representative spoke about the most important feature being what the users come to the website for, a working auctioning system. EBay's auctioning system is one of a kind, it stands out from the competition in the fact that it is quick and it just works. Effectively mimicking this inside of the resultant database will ensure its success in the ability to efficaciously target the desired audience.

Conceptualisation

During the development of this project; the intrinsic goal will be to ensure that the user will be able to navigate and effectively use the database program. This effectively means that the user should be able to open the database, place a bid on an item to then go to selling their own items. They should also be able to browse items they have sold and view seller information on said items. All of this information alongside the guidelines can be noted in the above Identification subsection of this report.

Over the course of this project, the main goals of this program will be:

- **Construction of a working auction system**
- **View and modify personal data**
- **View seller information**
- **Modify seller items**

Each part of this database will be constructed from the fundamentals of human computer interface, whereby each part will place humans first and allow the forms to be sculpted around the needs of the user as opposed to the needs of the software. The human computer interface effectively outlines the communication between the human user and a computer system, effectively incorporating this into the design will allow anyone the ability to pick up and manipulate the resultant program without taking various convoluted extra steps. Once these features are all implemented, the database will be able to be finalised and accessible by the clientele. To completely implemented this information system in a timely manner an effective methodology will need to be incorporated to have a structured plan to each section of the task. This extending to constructing the rationale for development (e.g. depicting what this information system will include over its competitors). While access databases are not commonly used within the business world. Several key features can be implemented to make it more effective for the users.

Key Features

The first of these features is a log in encryption hash system. Including this encryption step into the database will ensure that no user is able to be “hacked” or have their details stolen by some hacker trying to get into the database. This will be implemented through a unique serial token issuer (USTI) whereby a user’s details (their password) will be hashed against a private key [located within the database compiled code] to then be hashed with the public key which the public user form will have available. This alongside salting the hash will allow for a security whereby even if the data from the database was taken, the resultant information regarding the users log in will be useless.

Another key feature that will allow the ability to stand out among competitors is the addition of search box, the implemented search box will be triggered the moment the user starts typing, allowing them to easily see and backspace any results they are looking for. This will allow the user a more immersed experience as they won’t have to obstruct from what they are doing to press another button to purely perform the same task. Additionally, each form will be constructed with a ‘VBA-first’ structure where each form will be constructed to ensure it is able to be as fast as possible inside of Access. VBA of course standing for Visual Basic Access (the programming language that runs behind access and its various constructions); The VBA language was chosen over the alternatives of macros and other miscellaneous graphical based method is that purely down on a constitutional level **access is code**, this will lead to an under convolution in the resultant code constructed as there is no middle layer of graphical abstraction of the methodological steps.

For this project, the program Microsoft Access will be the main entity of this Database assignment, while there are better choices for database construction (For example MySQL alongside web

programming languages such as Python or HTML) Microsoft access is good for this task due to it pertaining to a packaged deal. This meaning that all of the database (forms, tables, querying engine) is all placed and located within the software suite. This being extremely good as there is no wasted time of this project getting the prerequisite programs working and will allow for simple transfer of the database to the client's computer. Alongside this, the overall code demand will be able to be minimized as there is no external code being used to just configure the interface as the interface is designed using Windows iconic Graphical Window API. Additionally, Adobe Photoshop will be incorporated to handle any logos or images used within the constructed database.

Limitations

Due to time constraints laid upon this task, various intrinsic information system features for an auction database will need to be excluded. This mainly revolving around the implementation of a shipping system alongside a feedback system for each user. The main reason for excluding these features is it would add another layer of time to configuration between forms alongside in real life both of these are commonly shared with a third party company which specialises in the construction and manipulation of user data like that. For example, a shipping company system (USPS) would commonly be used to track user shipments as opposed to the individual company itself offering the portal to check shipments. Additionally, user feedback can be shifted to an online conversational platform such as Disqus. This removing the limitations placed upon this task, further aligning it with real world ideals.

Data Recovery

During the course of this project whenever changes are made to the report or project they will be saved to both a project USB and be synced up with Google Drive. This is to ensure that in the event of a technological failure, a backup exists.

Formulisation

Once the conceptualisation section of this task has been constructed, it is time to create and formulate the information schema and its various components. To begin, a Gaant chart will be used that outlines the various pieces of the project and where it sits in the overall construction process. Refer below for constructed Gaant Chart:

Item	Week 11 Jul					Week 18 Jul				
	Monday (11)	Tuesday (12)	Wednesday (13)	Thursday (14)	Friday (15)	Monday (18)	Tuesday (19)	Wednesday (20)	Thursday (21)	Friday (22)
Define Project Goals	Individual	Individual								
Define Hardware And Software Goals		Individual								
Research Target Audience		Individual	Individual	Individual	Individual					
Define Asset Management strategy			Individual	Individual	Individual	Individual	Individual			
Construct Initial Storyboard						Individual	Individual			
Estimate costs							Individual	Individual		
Plan the testing strategies							Individual	Individual	Individual	Individual
Plan project delviery										
Finalise initial storyboard										
Construct final storyboard										
Begin making database										
Make database										
Fix Errors										
Have ready for delivery										
Alpha Testing										
Beta Testing										
Write Report	Individual	Individual	Individual	Individual	Individual	Individual	Individual	Individual	Individual	Individual

Gaant Chart 1

Refer to Appendix 2 for further outline of each other Gaant Chart items (following the course of this project)

Once the timeline schedule has been laid out, the formulation of the Access database information schema can be constructed, to begin the entities that are part of the system are laid out:

These entities being a crucial part of the universe of discourse (UoD) extending to only entities that relate directly to the auction stimulus. Therefore; the database entities for this project will merely be:

- **User and their applicable information (address, email, password, username, loggedInStatus, phoneNumber, currentBalanceOfAccount and CNCInformation)**
- **Bidding status of each bid, enlisting its own table to keep track of each user placing each separate bid entity**
- **Item information (name, description, sellerID, initBidTime, endBidTime, pictureOfItem)**

When compiled into these separate entities they can be placed and constructed into elementary sentences which will then describe the relationships between these entities to one another, effectively constructing the **conceptual schema** of the information system. Refer below for the resultant Elementary Sentences.

Elementary Sentences

firstName(name) has userID (number)
lastName(name) has userID (number)
address(address) has userID (number)
phone(phoneNumber) has userID (number)
userName(name) has userID (number)
password(password) has userID (number)
loggedIn(boolean) has userID (number)
currentBalance(currency) has userID (number)
CNCInformation(number) has userID (number)
email(emailAddress) has userID (number)
userType(name) has userID (number)

itemName(name) has bidID(number)
itemDescription(description) has bidID(number)
sellerID(number) has bidID(number)
picture(pictureLocation) has bidID(number)
initBidTime(time) has bidID(number)
initBid(currency) has bidID(number)
endBidTime(time) has bidID(number)

userID(number) has bidID(number)
itemID(number) has bidID(number)
price(currency) has bidID(number)
bidTime(time) has bidID(number)

userID(number) has soldItemsID(number)
itemID(number) has soldItemsID(number)
soldPrice(currency) has soldItemsID(number)

soldPrice has (currency) where (itemID(number) and sellerID(number))
userType(name) has user(administrator)
userType(name) has user(user)

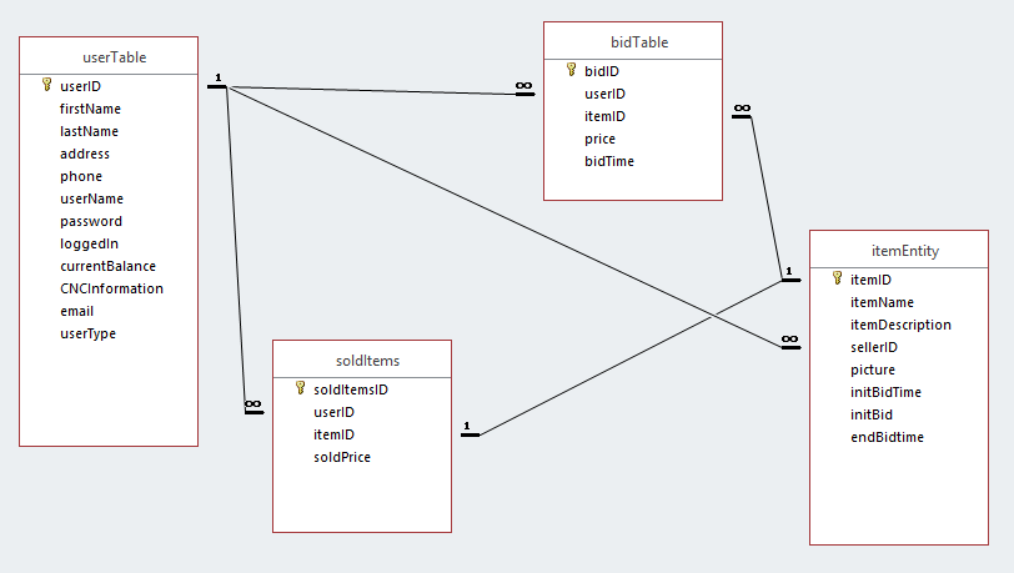
Once elementary sentences have been constructed they can easily be converted into a visual dictation of these ideas through the use of a conceptual schema; this schema essentially depicts the written information above in a visual medium. This making it a very effective tool for database developers to then integrate these systems into their projects.

Conceptual Schema

For the resultant conceptual schema of this data refer to Appendix 3.1 with the optimal normal form (O.N.F.) shading being depicted in the following Appendix 3.2.

Relational Schema

Once these resultant conceptual ideals have been constructed, the various tables can be tabulated. These tables will depict what is known as the relational schema and is what the information system will actually incorporate in order to effectively allow the system to work. Refer below for the respective relational schema of this information system.



Relational Schema
of database

From the construction of the relational schema, the complete data and table definitions can be constructed, as can be noted below. These give a brief overview of each data table location inside of the database and become crucial when replicating the database at a future interval.

Data Dictionary (Table Definitions)

	Variable Name	Data Type	Description	Possible Values	Restrictions
userTable	userID	Integer	Unique serial for user identification	1,2,3, etc.	Can't be null, has to be unique
	firstName	Short String	First name of user	"Lachlan"	Has to contain at least one character
	lastName	Short String	Last name of user	"Stevens"	Has to contain at least one character
	Address	Short String	Address of user	"12 Highview Close"	Has to contain at least one character
	Phone	Integer	Phone number of user	0400 123 456	Can't contain letters
	username	Short String	Username of user	"stevlach"	Has to contain at least one character
	Password	Short String	Password of user	"password"	Has to contain at least one character
	loggedIn	Boolean	Boolean depicts whether user logged in or not	0,1	Only 1 or 0 can be represented
	currentBalance	Float	Shows current balance in users account	$3.14	Only numerical values
	CNCInformation	Integer	Credit Card number	1234123421	Only numerical values
	Email	Short String	Email of user	info@example.com	Has to contain one character
	userType	Short String	Type of user	User, Administrator	Only these two options
itemEntity	itemID	Integer	Unique serial for Item identification	1,2,3, etc.	Can't be null, has to be unique
	itemName	Short String	Item Name	"Chair"	Has to contain at least one character
	itemDescription	Long String	Item Description	"Good deal, cheap price"	Has to contain at least one character
	sellerID	Integer	Unique serial for Seller identification	1,2,3, etc.	Can't be null, has to be unique
	Picture	Long String	Location of item picture	"C:/users/example/picture.png"	Has to be directory
	initBidTime	Date/Time	Time of auction start	16/01/19 4:00:00pm	Has to be date and time
	initBid	Float	Value of Initial Bid	$3.14	Only numerical values
	endBidTime	Date/Time	Time of auction end	16/01/19 5:00:00pm	Has to be date and time
soldItems	soldItemsID	Integer	Unique serial for sold Items identification	1,2,3, etc.	Can't be null, has to be unique
	userID	Integer	Unique serial for user identification	1,2,3, etc.	Can't be null, has to be unique

	itemID	Integer	Unique serial for item identification	1,2,3, etc.	Can't be null, has to be unique
	soldPrice	Float	Price sold at auction	$3.14	Only numerical values
bidTable	bidID	Integer	Unique serial for bid identification	1,2,3, etc.	Can't be null, has to be unique
	userID	Integer	Unique serial for user identification	1,2,3, etc.	Can't be null, has to be unique
	itemID	Integer	Unique serial for item identification	1,2,3, etc.	Can't be null, has to be unique
	price	Float	Bid Price	$3.14	Only numerical values
	bidTime	Date/Time	Time of Bid	16/01/19 4:20:00pm	Has to be date and time

Once the data definitions have been defined for the table, their definitions in access can be noted below:

userTable (Table Definition)

userTable

Field Name	Data Type	Description (Optional)
userID	AutoNumber	Unique serial for user identification
firstName	Short Text	First name of user
lastName	Short Text	Last name of user
address	Short Text	Address of user
phone	Short Text	Phone number of user
userName	Short Text	Username of user
password	Short Text	Password of user
loggedIn	Yes/No	Boolean depicts whether user logged in or not
currentBalance	Currency	Shows current balance in users account
CNCInformation	Short Text	Credit Card number
email	Short Text	Email of user
userType	Short Text	Type of user

itemEntity (Table Definition)

itemEntity

Field Name	Data Type	Description (Optional)
itemID	AutoNumber	Unique serial for Item identification
itemName	Short Text	Item Name
itemDescription	Short Text	Item Description
sellerID	Number	Unique serial for Seller identification
picture	Short Text	Location of item picture
initBidTime	Date/Time	Time of auction start
initBid	Currency	Value of Initial Bid
endBidtime	Date/Time	Time of auction end

soldItems (Table Definition)

soldItems

Field Name	Data Type	Description (Optional)
soldItemsID	AutoNumber	Unique serial for sold Items identification
userID	Number	Unique serial for user identification
itemID	Number	Unique serial for item identification
soldPrice	Currency	Price sold at auction

bidTable (Table Definition)

bidTable

Field Name	Data Type	Description (Optional)
bidID	AutoNumber	Unique serial for bid identification
userID	Number	Unique serial for user identification
itemID	Number	Unique serial for item identification
price	Currency	Bid Price
bidTime	Date/Time	Time of Bid

Conceptual Schema Assumptions

During the creation of this conceptual schema it was assumed that only the functional bidding system was needed (as previous discussed upon first noting the viability of product). Based on this assumption several key features were dismissed, notably the admission of the shipping and a peer to peer feedback system. This was to minimize time and overall feasibility of product within the various depicted time constraints placed upon this task. Based upon these assumptions the respective Conceptual/Relational Schemas have been effectively constructed.

Menu Tree

Refer to Appendix 4.1 for written diagram of the menu tree of the forms inside of this information system. For any extra graphical depictions of the finalised storyboard (what the forms are prospectively going to look like, refer to Appendix 1)

Prospective SQL Queries

Refer to Appendix 5.1 for the prospective written SQL queries of this database to offer full functionality to its users.

Implementation

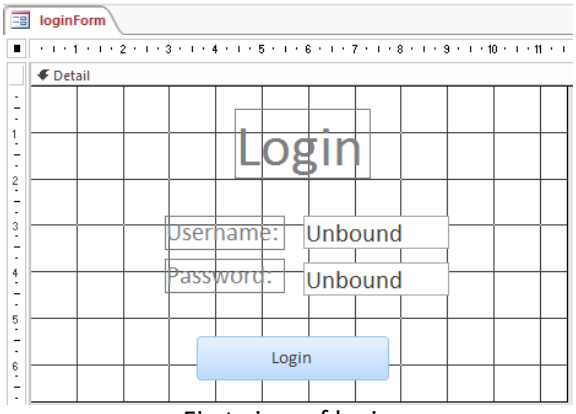
Once each part of the conceptualisation was laid out, it was time to enter all of the data into an access database. To begin the basic tables were constructed which various pieces of data given as placeholders. Once the tables were constructed and their relevant relationships were made (in accordance to the conceptualisation above); it was time to transition into actually producing the product that the client will see at the end product. For this project the fake logo of the company will be:

Mock-up logo for a
representative auction company

Having a logo to be used inside of the design will allow for a more immersive product to be produced, as it enlists into the user an aura of professionalism with the product they are using.

Login Form

With the finalised storyboard it was time to transition this project into access and construct the Information System. To begin this transformation a simple methodology was devised, mimic the design (wireframe) in the visual program and see how well the elements worked together; from an application point of view. As can be noted below it was quite quick to construct the Login form within Access by merely dragging and dropping the respective elements to their locations as depicted on the storyboard. This effective use of WinAPI allowed a fast and efficient working environment. Note below for the representation of the Login form within access.



First view of login form within access

This original form abstraction was constructed through a simple button and, several labels and two unbounded input boxes. A VBA event was then added to the button in order to query the database in order to register whether or not a user has successfully logged in. This followed the logic of checking that the username box was full, to then checking if the password box had text placed in it. From here the database was queried **specifically** taking note to neglect entries if the logged in value was already = 1. This would allow multiple users to access the database at once (if it were to be shared on a networked drive); which would in turn allow a multitude of users the capacity to place bets and auction items as regarded to being the main goal of this project.

From here if there was a correct result (whereby both user name and password both matched with the user not already being logged in) the loggedIn Boolean was updated with the command of:

```
SQL = "UPDATE userTable " & _
      "SET userTable.loggedIn = 1 " & _
      "WHERE userTable.userName ='" & Me.usernameTextBox.Value & "'"
```

Updating of loggedIn boolean

Which essentially updates the loggedIn Boolean ensuring that no other user can log in under that account while the first login remains logged in. From here the Login form was set to be invisible which allows the variables of the input boxes to remain readable by other external queries by other forms.

To ensure that when the database was closed any user was logged out an extra event was added to the login form, so that when it remained invisible in the background it could act as an event holder, which would execute right before the database closed. This event was the Form_Unload event within access. This would essentially just quickly reset the values of who was logged in, to ensure no loss or corruption of any data would incur. Refer below for this code:

```
Private Sub Form_Unload(Cancel As Integer)
    ' Used to test if query will run that will run before exiting
    ' MsgBox "Sure you want to quit?", vbCritical, "Exit Confirmation"

    ' Run sql which will reset all logged in users
    Dim SQL As String

    SQL = "UPDATE userTable " & _
          "SET userTable.loggedIn = 0"

    CurrentDb.Execute SQL

End Sub
```

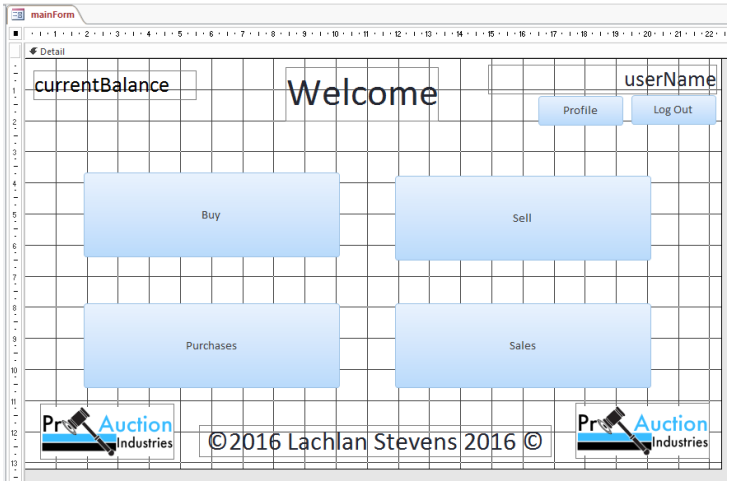
`

Form_Unload code

Once this logic was added to the form it was a mere process of having it open up the main form right after placing itself to be invisible in order to have the next part of the user tree continued. Refer to Appendix 6 for complete VBA code.

Main Form

Upon successful construction of the login form, the main navigation form should be constructed. This can be achieved by following the same preceding steps (as demonstrated before) whereby the WinAPI graphical form functions are implemented in accordance to the elements as depicted within the storyboard (Refer to Appendix 1). This gained design view results of:



Main form Design View
(From storyboard)

On the surface, this is a perfect implementation of the storyboard proving just how applicable and user friendly it is by following the C.A.R.P. Design principles alongside the Human Computer Interface Techniques teased throughout the design. From this, the overall code that runs this form is considerably more involved than from the previous login form, due to the authentication having to take place among the form. This can be summarised by the following process; When the form_load event is triggered upon the form loading the form runs a query towards the database trying to find the details about the current logged in user (from the previous form). This information is found by reopening (un-hiding) the login form behind the new main form to then explicitly extract the login form information to effectively query the respective information. This of course resulting in the First Name and Last name and the Current Balance of the logged in user all the while knowing that security of the database hasn't been flawed. This is due to no active details being transferred over a network with all of it happening locally within access. From this point the form has been successfully loaded and it goes into a waiting mode awaiting user input to proceed further.

Note below the login form code (for authentication of users):

```
Private Sub Form_Load()
    DoCmd.OpenForm "loginForm", , , , , acHidden
    Dim SQL As String

    SQL = "SELECT * from userTable " & _
          "WHERE userTable.userName = '" & _
          Forms![loginForm].usernameTextBox.value & "'"

    Me.loggedInUser.Caption = CurrentDb.OpenRecordset(SQL).Fields("firstName").value + " " + CurrentDb.OpenRecordset(SQL).Fields("lastName").value
    Me.currentBalanceLabel.Caption = FormatCurrency(CurrentDb.OpenRecordset(SQL).Fields("currentBalance").value, 2)
    Forms![loginForm].Visible = False
End Sub
```

Authentication code
(for logged in users)

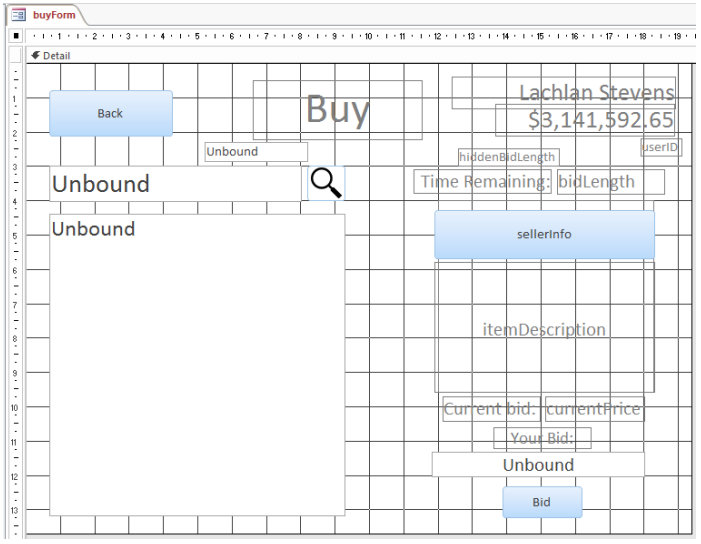
Once the user is logged in it is simply a matter of manipulating blank buttons for their click events to be triggered to open and close to take them further along into the program. The only other interesting thing about this form is the Logout button. The log out button is more technologically involved due to the fact that when it is clicked, the login form is reopened which will then clear the logged in user status alongside clearing the input boxes on the username and password fields (within the login form). The main form is then closed and with the login form being reopened in order to depict a clean log off; ready for a new user to log in. Refer below for the VBA code of these events. (Appendix 6 has total code of program)

```
Private Sub logoutButton_Click()
    Dim SQL As String
    DoCmd.OpenForm "loginForm"
    SQL = "UPDATE userTable " & _
          "SET userTable.loggedIn = 0 " & _
          "WHERE userTable.userName = '" & _
          Forms![loginForm].usernameTextBox.value & "'"

    CurrentDb.Execute SQL
    Forms![loginForm].Visible = False

    DoCmd.Close acForm, "mainForm", acSaveYes

    Forms![loginForm].usernameTextBox.value = ""
    Forms![loginForm].passwordTextBox.value = ""

    Forms![loginForm].usernameTextBox.SetFocus

    Forms![loginForm].Visible = True
End Sub
```

Logout Code

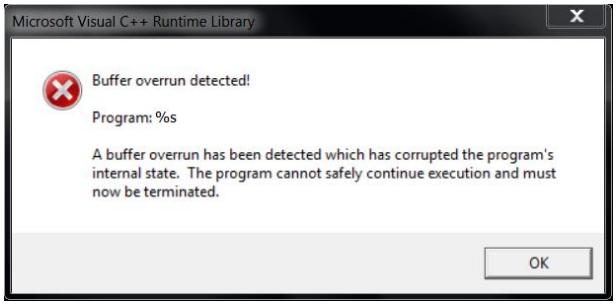
Buy Form

Upon successful completion of the main form; the buy form must be constructed. The buy form extends to all of the actual “auction” processes that occur within the database as it is the portal each prospective buyer will enter in order to place bids and effectively win items. As previously noted, the design to implement the form was purely manipulation of the WinAPI and its various components in order to produce a product visually similar to the one depicted within the storyboard. From here the various pieces of logic was added to the components in order to make them work as envisioned in the conceptualisation process. As can be noted below the design implementation of the buy form follows exactly what was noted in the storyboard.



Buying Form
(Design View)

Out of each form, this was the hardest to get working (in regards to the logic). This is purely due to the countdown at the top being from Days to Hours to Minutes to Seconds. Having a countdown be calculated based on a set of two days proved to be a lot more tedious and time consuming than any part of this project. The main problem was that access kept giving an overflow error:



Buffer Overflow Error

While this error is quite vague it was eventually figured out to being constructed due to how the dates were being converted, to longs; which then separated down to extended float variables which then overflowed and left them being outside of the respective buffer in access. While this was a messy solution to calculating the interval between two dates, a new method was devised which was settled on. Using the inbuilt timeDiff function, the amount of seconds were calculated with a modulo function being used to separate between the hours, minutes and second parts of the timing. This can be seen contrasted and implemented below(*Note* Buffer overflow caused by looping across function, something crucial to having an updated working countdown function):

```
' Set bid time difference
Dim dblNumDays  As Double
Dim dblNumHours As Double
Dim dblNumMins  As Double
Dim dblNumSeconds As Double

Dim dblNumDaysFinal  As Integer
Dim dblNumHoursFinal As Integer
Dim dblNumMinsFinal  As Integer
Dim dblNumSecondsFinal As Integer

dblNumDays = Abs(CDbl(Now()) - CDbl(CurrentDb.OpenRecordset(SQL).Fields("endBidTime").Value))
dblNumHours = dblNumDays * 24 - (Fix(dblNumDays) * 24)
dblNumMins = dblNumHours * 60 - (Fix(dblNumHours) * 60)
dblNumSeconds = dblNumMins * 60 - (Fix(dblNumMins) * 60)

dblNumDaysFinal = Int(dblNumDays)
dblNumHoursFinal = Int(dblNumHours)
dblNumMinsFinal = Int(dblNumMins)
dblNumSecondsFinal = Int(dblNumSeconds)

Me.bidLength.Caption = dblNumDaysFinal & ":" & dblNumHoursFinal & ":" & dblNumMinsFinal & ":" & dblNumSecondsFinal
```

Original Attempt of time calculation
(Resulted in Buffer Overflow)

```
' Already called db
Dim shortDate As Date
Dim shortTime As Date

shortDate = Format(Me.hiddenBidLength.Caption, "dd/mm/yyyy")
shortTime = Format(Me.hiddenBidLength.Caption, "hh:mm:ss")

Dim dblNumDays  As Integer
Dim dblNumHours As Integer
Dim dblNumMins  As Integer
Dim dblNumSeconds As Integer

dblNumDays = DateDiff("d", Format(Now(), "dd/mm/yyyy"), shortDate)
If dblNumDays > 0 Then
    ' Positive Number
    shortTime = Format("23:59:59", "hh:mm:ss")
Else
    ' Do nothing
End If
dblNumHours = DateDiff("s", Format(Now(), "hh:mm:ss"), shortTime) / 3600
dblNumMins = (((DateDiff("s", Format(Now(), "hh:mm:ss"), shortTime)) / 60)) Mod 60
dblNumSeconds = (DateDiff("s", Format(Now(), "hh:mm:ss"), shortTime) + 30) Mod 60

Me.bidLength.Caption = dblNumDays & ":" & dblNumHours & ":" & dblNumMins & ":" & dblNumSeconds

If dblNumSeconds < 0 Then
    ' Item should be sold, sell to highest bidder (and/or back to seller)
    sellItem Me.itemListBox.Column(0), FormatCurrency(Me.currentBid.Caption, 2)
Else
    ' Do nothing
End If
```

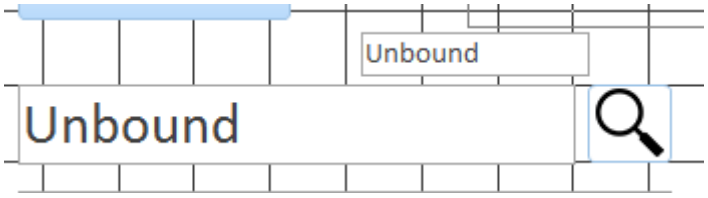
Revised Calculation
(Doesn't Cause Errors)

Upon successful implementation of the countdown timing for the bids placed into the system, it was merely a process of copying the authentication from the previous form (main form) into the buy form to then configure the search function. Another thing which made this form quite involved was its reliance on a search function. In order to implement the search function (as can be noted in the handwritten SQL notes revised in conceptualisation processed). To implement the search function an external query needed to be constructed using the design tools offered by Microsoft access. Inside of this query the following was constructed:

```
SELECT itemEntity.itemID, itemEntity.itemName
FROM itemEntity LEFT JOIN soldItems ON itemEntity.itemID = soldItems.itemID
WHERE (((itemEntity.itemName) Like "*" & [forms]![buyForm]![searchHiddenInput] & "*") AND ((soldItems.soldItemsID) Is Null))
ORDER BY itemEntity.itemName;
```

SQL Query to run Auction Search

The resultant query depicted above extends to the connection between the way the form is working. In essence the query cant actively read from an in-focus text box; and it can't read from a label (due to integral access limitations placed upon their software) meaning the only solution to actively read an input from a query is to replace the value of the text into an elemental control that isn't dictated by the WinAPI control interface. By constructing a new hidden text input box right behind the input box of the search function, its value could be manipulated using code that updated it whenever the text inside of the input box was changed to then retrieve SQL results which would then populate the list box underneath with the various results. This idea can be noted with code below:



Hidden textbox right next to search

```
Me.itemListBox.Requery

Me.itemListBox = Me.itemListBox.ItemData(0)
Me.itemListBox.SetFocus
Me.itemListBox.Selected(0) = True
```

Re-query of respective list-box

Once this search was implemented across the system the respective list box event could be added which would respectively update each value dictated on the form. This would effectively be achieved by running a population update across the fields whenever the list box item-select event was triggered. With the list box effectively updated, it was merely a process of adding the event to populate each value with the respective database value, as can be noted in the code below:

```
Private Sub updateItems()
On Error GoTo errUpdateItems
    ' Now that row value has been found (representing itemID in itemEntityTable),
    ' Call SQL query to populate other fields.
    Dim SQL As String
    Dim bidSQL As String

    SQL = "SELECT itemEntity.*, userTable.* " & _
          "FROM userTable INNER JOIN itemEntity ON userTable.userID = itemEntity.sellerID " & _
          "WHERE itemEntity.itemID = " & Me.itemListBox.Column(0)

    bidSQL = "Select userTable.userName,bidTable.bidTime, bidTable.price " & _
             "FROM userTable INNER JOIN bidTable ON userTable.userID = bidTable.userID " & _
             "WHERE bidTable.itemID = " & Me.itemListBox.Column(0) & " " & _
             "ORDER BY bidTable.bidTime DESC;"
    ' NEED:
    ' Item Description
    ' Picture Location
    ' initBid
    ' endBidTime

    ' Set item description
    Me.itemDescription.Caption = CurrentDb.OpenRecordset(SQL).Fields("itemDescription").value
    ' Set button text
    Me.sellerInfo.Caption = CurrentDb.OpenRecordset(SQL).Fields("userName").value
    ' Set image
    Me.pictureBox.picture = CurrentDb.OpenRecordset(SQL).Fields("picture").value
    ' Set latest Bid
    If CurrentDb.OpenRecordset(bidSQL).EOF Then
        ' Set to initial Bid as none have been made yet
        Me.currentBid.Caption = FormatCurrency(CurrentDb.OpenRecordset(SQL).Fields("initBid").value, 2)
    Else
        ' Set to latest Bid
        Me.currentBid.Caption = FormatCurrency(CurrentDb.OpenRecordset(bidSQL).Fields("price").value, 2)
    End If
    ' Set bid time difference
    Me.hiddenBidLength.Caption = CurrentDb.OpenRecordset(SQL).Fields("endBidTime").value
    If Me.hiddenBidLength.Caption = CurrentDb.OpenRecordset(SQL).Fields("endBidTime").value Then
        ' Already called db
        Dim shortDate As Date
        Dim shortTime As Date

        shortDate = Format(Me.hiddenBidLength.Caption, "dd/mm/yyyy")
        shortTime = Format(Me.hiddenBidLength.Caption, "hh:mm:ss")

        Dim dblNumDays  As Integer
        Dim dblNumHours As Integer
        Dim dblNumMins  As Integer
        Dim dblNumSeconds As Integer

        dblNumDays = DateDiff("d", Format(Now(), "dd/mm/yyyy"), shortDate)
        If dblNumDays > 0 Then
            ' Positive Number
            shortTime = Format("23:59:59", "hh:mm:ss")
        Else
            ' Do nothing
        End If
        dblNumHours = DateDiff("s", Format(Now(), "hh:mm:ss"), shortTime) / 3600
        dblNumMins = DateDiff("s", Format(Now(), "hh:mm:ss"), shortTime) / 60 Mod 60
        dblNumSeconds = DateDiff("s", Format(Now(), "hh:mm:ss"), shortTime) Mod 60

        Me.bidLength.Caption = dblNumDays & ":" & dblNumHours & ":" & dblNumMins & ":" & dblNumSeconds
```

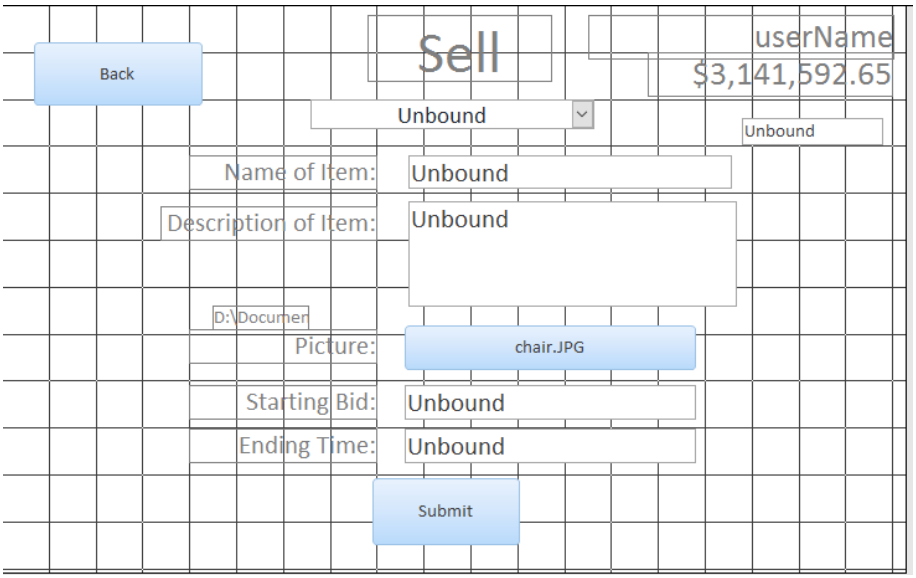
Code depicting on list-box change event

Once list box had been added, there was only one more major addition to be made in order to have this auction form working to its fullest. The ability to place bids was crucial. In order to get his working the input box was sanitised from any user input that wasn't money/currency related; to then have the value only matter if it was a feasible bid to make (e.g. able to be afforded/was high enough to beat competitors). Once this was decided the bid was then placed by running an SQL query adding an entry to the bidding table. In order to determine a winner, the auction was waited until its time was at zero unto which the awardee and transaction would be recorded and each seller and buyer would receive a message box outlining what they received. This logic can be found depicted within VBA code in Appendix 6.

Sell Form

By this point of the creation of the information system there was a lot of foundation code placed, so all of the remaining forms were just abstractions of the previous forms with different information being sorted for. For example, within the sales form the only new piece of code was the addition of a new SQL query in order to sort and unionise the location of a new product entry.

Following the previous standard of implementing the various forms into access (by following storyboard with the graphical tools offered by access) the sell form was able to be implemented quite efficiently, as depicted below:



Implementation of design of sell form

Several things to note about the above image, the hidden label denoted with “D:\Documen” as noted above. This label was added in order for the full location of the image to be noted and saved while not interrupting the user experience. Among this, as discussed there is nothing new to this form other than the addition of the Union within the combo-box query.

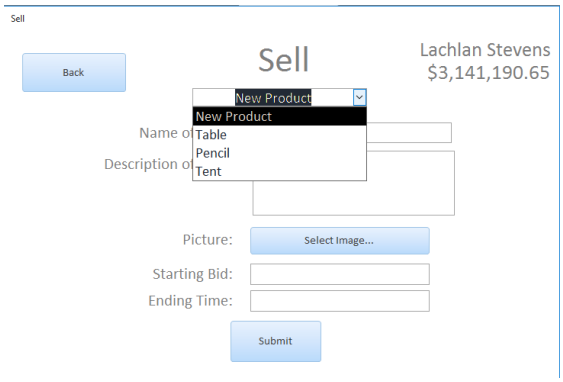
At this point there would be a question of why the original access form controls weren’t used in order to modify and place entries into the form. The reasoning for this is due to the limited control gained through using the default access controls. This is in the sense that for example controlling the location of a picture wouldn’t be possible through the admission of a button in a classical utilisation of Access and its default controls. Due to these limitations VBA (Visual Basic for Applications) has been chosen as the main architecture for this project.

From here the combo-box query was as denoted below:

```
SELECT itemEntity.itemID, itemName
FROM itemEntity
LEFT JOIN soldItems ON soldItems.itemID = itemEntity.itemID
WHERE itemEntity.sellerID =  [forms]![sellForm]![currentUserID]
AND soldItems.itemID IS NULL
ORDER BY itemEntity.itemID
UNION SELECT -1, "New Product"
FROM itemEntity;
```

Combo-Box query

The above query of course **always** having an index of -1 on top representing a new product, this allowing the user to easily manipulate what was trying to be accomplished within the respective form. This when implemented into the form deems the following:



Implementation of Combo-Box Query

The only other buttons on this form represent the reading of an open file dialog in order to retrieve the location of the respective image chosen, this code being noted below as:

```
Private Sub fileSelection_Click()
    Dim fileDialog As Object
    Set fileDialog = Application.fileDialog(msoFileDialogOpen)
    With fileDialog
        .AllowMultiSelect = False
        .Title = "Please Select Item Image"
        .Filters.Clear
        .Filters.Add "Images", "*.png; *.jpg; *.jpeg", 1
    End With

    If fileDialog.Show Then
        fileDir = fileDialog.SelectedItems(1)

        ' Set to picturePath
        Me.fullPath.Caption = fileDir

        Dim fileName As String
        fileName = Split(fileDir, "\")(UBound(Split(fileDir, "\")))

        ' Rename button to filename
        Me.fileSelection.Caption = fileName
    End If
End Sub
```

File Dialog VBA Code

Upon reaching this point the final code would be within the submit button, which collates each separate input box entity and uploads it to its respective place within the database. Refer below for the code that implements the submit button.

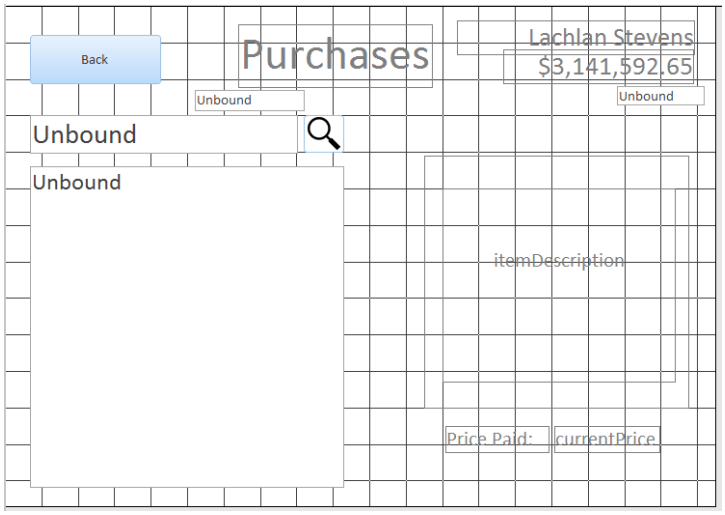
```
Private Sub submitButton_Click()
    Dim SQL As String
    Me.sellingItemsList.SetFocus
    If sellingItemsList.Text = "New Product" Then
        SQL = "INSERT INTO itemEntity (itemName, itemDescription, sellerID, picture, initBidTime, initBid, endBidTime) " & _
          "VALUES ('" & Me.bidInput.value & "', '" & Me.bidDescription.value & "', " & Me.currentUserID.value & _
          ", '" & Me.fullPath.Caption & "' ,'" & Now() & "' ," & Me.bidStarting.value & " ,'" & Me.endBidTime.value & "');"

        CurrentDb.Execute SQL

        MsgBox "Successfully Added Item"
    Else
        SQL = "UPDATE itemEntity " & _
            "SET itemName='" + Me.bidInput.value + "', itemDescription='" + Me.bidDescription.value + "', " & _
              "sellerID='" + CStr(Me.currentUserID.value) + "', picture='" + CStr(Me.fullPath.Caption) + "', " & _
              "initBidTime='" + CStr(Now()) + "', initBid='" + CStr(Me.bidStarting.value) + "', " & _
              "endBidTime='" + Me.endBidTime.value + "' " & _
            "WHERE itemName='" + sellingItemsList.Text + "';"
        CurrentDb.Execute SQL
        MsgBox "Successfully Appended Item"
    End If
    Me.sellingItemsList.Requery
    Me.sellingItemsList.SetFocus
    Me.sellingItemsList.Text = Me.bidInput.value
    Me.submitButton.SetFocus
End Sub
```

Submit Button Code

Upon successful integration with the respective submit button each user can successfully sell and bid items at the online auction. The following forms will allow the user to view and review their winnings/selling's.

Purchases Form

The purchases form is merely an abstraction to the workings of the buying form, without the bidding option. This is in the sense where the major design schematics are copied directly from the buy form with only minor discrepancies in the design being noted (removal of the countdown of the respective auction [due to the user already owning the item]). As discussed, the design was implemented by following the storyboard in order to get the general interface working across the database. Refer below for the resultant depiction of the storyboard:



Purchases Form depiction

Following this, it was merely a process of repeating all of the steps depicted in the buy form with only the removal of some VBA code to rid the new form of the Buy Form features. The resultant SQL query entered into the new list box was as follows:

```
SELECT soldItems.itemID, itemEntity.itemName
FROM itemEntity INNER JOIN soldItems ON itemEntity.itemID = soldItems.itemID
WHERE (((itemEntity.itemName) Like "*" & forms!purchasesForm!searchHiddenInput & "*") And ((soldItems.userID)=forms!purchasesForm!userID))
ORDER BY itemEntity.itemName;
```

SQL for purchase List box

Note the only above difference is the table the resultant information is being pulled on, referring to the current user ID instead of the Sellers ID (as will be demonstrated in the following form).

Sold Form

The sold form is merely an abstraction of the above Purchases form, as it allows the user the ability to view Sales instead of Purchases. From this, the design is the same following the same process as before (from the storyboard) the resultant form looks remarkably close to the one depicted above; this is a key piece of design which allows the user to feel safe and comforted with a familiar interface whenever using or manipulating the respective software. Due to these analogous forms the code and construction is the exact same, minus for the only difference being the SQL query which is executed within the Sales List Box. This SQL query generally being:

```
SELECT soldItems.itemID, itemEntity.itemName
FROM itemEntity INNER JOIN soldItems ON itemEntity.itemID = soldItems.itemID
WHERE (((itemEntity.itemName) Like "*" & [forms]![soldForm]![searchHiddenInput] & "*") And ((itemEntity.sellerID)=[forms]![soldForm]![userID]))
ORDER BY itemEntity.itemName;
```

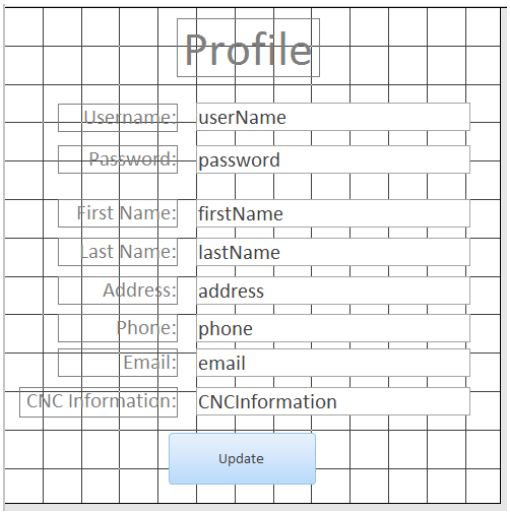
SQL for sold List box

As can be noted above, the only difference is indeed the userID as opposed to the seller ID and the table is being pulled from sold Items.

Another addition to note about the following three forms, Sold, Buy and Purchases is they have a hidden feature whereby the picture that is displayed of the item; when clicked reveals a hidden interface to access and view the information of the seller or the item (in regards to the purchases form). In the other forms it just generalises to the overview of the description of the item. This was accomplished by having two separate events triggered, one on the click of the picture and the other on the click of the description. In both cases it hides the other and shows the other set. This extending to the effect that when the user clicks on the image there is a subtle change of information, making it an effective trick of maximising the utilisation space depicted on the form to the user.

User Profile/Registration Form

While the main forms have been discussed, the final forms within this information system are the user project and registration forms; each of which playing a crucial role in the user having the ability to keep their information updated. The same process was followed of implementing the forms, with quite a minimal amount of VBA being needed to appropriately pull off the updating process. Refer below for the constructed form version of the user profile/registration storyboard:



Design view of Profile list

As can be noted, this form is bounded natively to the access controls. This is due to the access natively having the support for updating a multitude of text input boxes. This being very helpful in maintaining a minimum amount of code used on such minor forms.

For populated tables refer to Appendix 7.

Testing

While the project was in the midst of being created a comprehensive job was done to test and test locally (pre-alpha phase) where many errors were found, with the some of the most notable being Syntax errors with the HTML being shown in the abstract of the Server signage interface, this of course being solved by implementing Regular Expressions into the code allowed for more accurate sorting and replacing of the string.

Self-Testing

Once at the end of the process of construction of the database, the login form was decidedly revamped to feature more of the logo alongside attracting a broader audience with an encapsulating

```
If (IsNull(DLookup("userName", "userTable", "[userName] ='" & Me.usernameTextBox.value & "' And [loggedIn] = 0"))) Or _
(IsNull(DLookup("password", "userTable", "[password] ='" & MD5_string(Me.passwordTextBox.value) & "'"))) Then
```

view. Alongside this, the security of the login form was not being hashed so an MD5 hashing system was constructed within VBA which allowed the text given in the input boxes to be converted to an MD5 hash which effectively means if hackers managed to get hold of all of the data in the database, the individual passwords or user access won't be at risk. Refer below for the VBA implementation and changes to the login form.

Revamped Login Screen

As can be noted in the above login screen design, it encapsulates more contrast into the design, this is crucial as due to accordance with the C.A.R.P. principles Contrast is one of the main things that draw a user's attention in. This alongside Repetition (as depicted across each of the form designs), Alignment and Proximity (dictated through the combination of like elements and their specific alignment and centring across the form). Overall, these two changes have greatly increased the overall approachability of the database and the security of the database. Two things which are of the upmost goals when constructing this information system.

Alpha Testing

Although, no matter how much pre alpha testing occurs nothing suffices from having fellow students test and give feedback on the functioning prototypes of the software. Fellow classmates, Alex VanLint and Duron Prinsloo were asked to give feedback on the software and various observations made about the program. Refer to Appendix 8.1 – 8.2.

Originally, in the alpha tests it was noted that the images loaded slowly; resulting in a slow experience overall; While this can be stacked up to merely being a problem with the networking experienced the day of testing, after restarting the entire database and resyncing everything to the server these experiences were gone (as noted by the Beta Testers of this product). Overall, the responses seemed very good and positive with no real changes being suggested to change the product at this point in time.

Beta Testing

Beta Testing refers to feedback given by Kit O'Brien (Appendix 8.3) and Matt Stewart (Appendix 8.4). This feedback was an improvement over the last as it proved that the previous problems were gone and the program was functioning as expected. None of these testers gave adequate improvement to be made to the software stating that it worked extremely well at the present time.

Evaluation

When first taking up this project, it quickly became prevalent how big it was going to be. There were a lot of complicated parts working together meaning there wasn't time to perfect just one part, there needed to exist a working prototype, which was then added on. This was accomplished in part, by creating each part of the program slowly and on a part per part basis; resulting in very modular, easy to implement code. Overall, this code functions very efficiently with not many resources being wasted on unnecessary computations.

Extensions to this software in the future would be to transfer onto a web based medium. Removing the limitation of access will allow for an unprecedented reach in regards to the application of the product; although stating that a **localised** database such as Access has its strengths in the fact it is good for mere local area network situation (such as a local school auction across devices).

Alongside this another useful inclusion that wasn't able to be achieved within the given time period was the addition of an auto populate command. Where the form would automatically populate with the information that is already in the database and the user would just need to edit it. Having this feature would save the user a lot of time when manipulating the program with it overall relating to the Human Computer Interface of the software. The human computer interface for this project was very highly accredited purely because it follows the strict rulings of C.A.R.P. and Schneiderman. C.A.R.P. was followed all over the program, firstly with the use of Contrast in the login Interface. Effectively using a gradient to highlight the title on the Server interface allowed the user to gain a very obvious perception to what the text wrote. This along with the Z rule on that page meant the very last thing the person would see would be the submit button.

Following from this, Alignment was effectively demonstrated through the implementation of the grid elements, separating each element and having their own clear spots helped the user feel a sense of consistency throughout the program which draws their appeal to the program stronger. Following from this, Repetition was shown throughout all of the interfaces. This was achieved through the consistent submit button down the bottom of the page and the overall repetition in the way the elements were laid out (in a grid).

Finally, Proximity was incorporated by allowing each of the relevant pieces of content to stay together within 'proximity' of each other. This again allows the user of the software feel a sense of consistency and safety when using the program as it allows them to step back and feel in control of what the computer is doing.

Moving on from this, Shneiderman was a very famous computer scientist who developed 8 rules to follow for a successful human computer interaction. These rules begin with striving for consistency, this is true for this project as consistency across all forms were followed. The next rule was 'Enable frequent users to use shortcuts', this is untrue for this project but could be taken as an improvement in the future. Having shortcuts allows more fluent users a faster way around the program resulting in a better user experience. Next is 'Offer informative feedback'. Having quick simple feedback is key to having a successful program. This is achieved within this project by incorporating the use of message boxes to subtly alert the user of the current status of the program.

Following, the next rule is 'Design dialog to yield closure'; having a form like within this project which opens up and tells the user they have successfully updated the form to then have the form close, follows this rule. Having this sense of closure welcomes the user to make changes as they know when they make changes it is saved and it is done. From this, 'Offer simple error handling' this is done in the fact that whenever there is an error on the application it is gracefully echoed to the console. Having this means the program won't crash in the middle of something but it also means if there is something going awry with the program an administrator can simply open it up within a console and retrieve the input from there.

'Permit easy reversal of actions', incorporating this into the program allows the user to feel safe and comfortable with using it. Unfortunately, a lot of the mechanisms involved with setting that up are beyond the scope of this project; but could be implemented in future versions. 'Support internal locus of control'; in other words, make a responsive and quick interface. This is achieved within this project as the interface between both one user and another is seamless and both allow for active real time updates of the program, with no lag being experienced between any part of the communication between them.

Finally, 'Reduce short-term memory load.' This is achieved in this project by incorporating the use of the digital sign to display all of the information. Doing so, allows users to see the information in front of them without thinking about it. Overall, around about 6 of those 8 rules were successfully followed making this an interface that is statistically effective in accordance to the Schneiderman and C.A.R.P. Design rules. Having implemented all of the above rules correctly, this leads to a conclusion that a successful product has been produced.

Appendix

Appendix 1.1 (Drawn Storyboard [Login & Registration/Profile Form]):

See Attached Document

Appendix 1.2 (Drawn Storyboard [Main & Buy Form]):

See Attached Document

Appendix 1.3 (Drawn Storyboard [Sell Form]):

See Attached Document

Appendix 1.4 (Drawn Storyboard [Purchases and Sales Form]):

See Attached Document

Appendix 2.1 (Gaant Chart 1):

See Attached Document

Appendix 2.2 (Gaant Chart 2):

See Attached Document

Appendix 2.3 (Gaant Chart 3):

See Attached Document

Appendix 3.1 (Conceptual Schema):

See Attached Document

Appendix 3.2 (O.N.F. Conceptual Schema):

See Attached Document

Appendix 4.1 (Menu Tree)

See Attached Document

Appendix 5.1 (Written SQL Queries Prospectively to be used)

See Attached Document

Appendix 6 (Total VBA Code)

See Attached Document

Appendix 7 (Total Forms, Tabulated Data, Access Database Data)

See Attached Document

Appendix 8.1 (Duron Prinsloo – Alpha Testing)

See Attached Document

Appendix 8.2 (Alex VanLint – Alpha Testing)

See Attached Document

Appendix 8.3 (Kit O'Brien – Beta Testing)

See Attached Document

Appendix 8.4 (Matt Stewart – Beta Testing)

See Attached Document

Bibliography

Anon, 2016. *CARP Design Principles | Digest Web Design | Free Website Design Lessons, HTML, CSS*. [online] Digestwebdesign.com. Available at: <http://digestwebdesign.com/carp_design_principles.html> [Accessed 14 Aug. 2016].

Anon, 2016. *Shneiderman's "Eight Golden Rules of Interface Design" | Design Principles FTW*. [online] Designprinciplesftw.com. Available at: <http://www.designprinciplesftw.com/collections/shneidermans-eight-golden-rules-of-interface-design> [Accessed 12 Aug. 2016].