BUY AND SELL APPLICATION

A-Level NEA Project

Hamza Latif
Student ID: LAT18003410
Centre Number:

Table of Contents

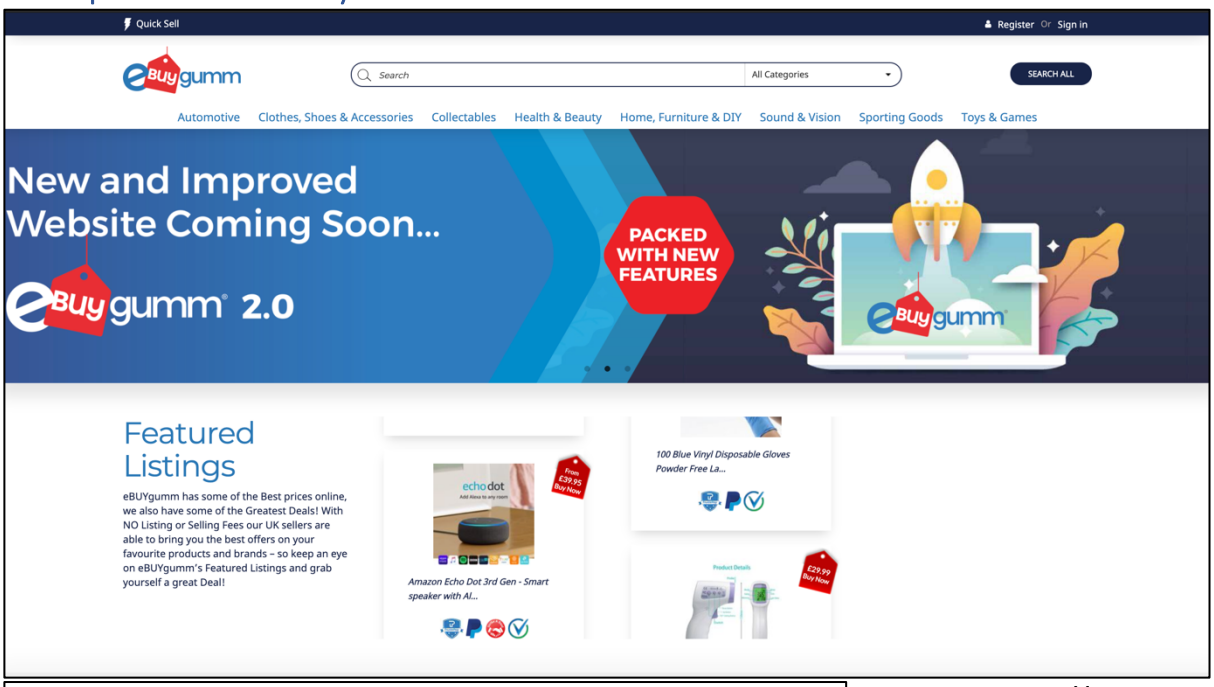
ANALYSIS

Project Background

My project will consist of an application where it is possible to buy and sell electronics items. I have always been interested in programming an application that could be used in the real life by other people. My initial thought was to develop a game, but I dropped the idea because I thought it was something that many other people would also try to do, and I wanted to create something more personal to me. Therefore, I thought about what applications I use on my phone or laptop. That is when I realised that me and many others use buy & sell application every day. The most popular and successful online stores that come my mind are Amazon & eBay. I don't think that my application will be at the same level of complexity and difficulty of those, but those two are my aspirations, my goals. Creating a buy & sell application is not easy as there are many aspects to consider, for example having an efficient, user-friendly interface where people can easily find what they are looking for; or making sure that the items are displayed correctly, or that the customer details are protected. So, I thought developing a buy & sell application would be a great challenge for me and it will help me enhance my programming skills.

My application will be divided in two different sectors. First of all, the user will have to create an account in order to login. In the first part, customer can have a look at the available products. There will be a table showing the items, a description with all the details and the price alongside. Customers will be able to buy with credit cards, which will only work if a valid number is inserted. In another section, the user will be able to list items that he would like to sell. These items will then show up in the buying section and will disappear after they are sold out.

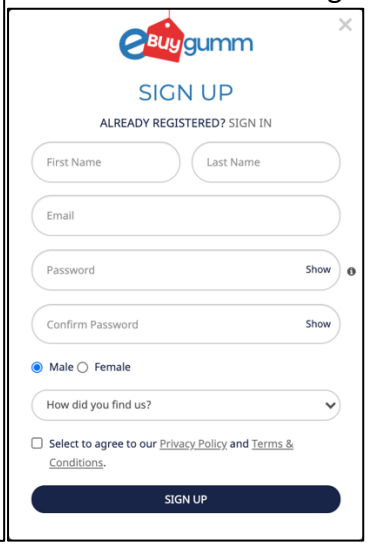
Description of Current System



Homepage

Seller Section

Login



One of the most famous and popular UK buy and sell websites is ebuygumm.co.uk. The website is very user-friendly, it is very simple, and it uses good colours combination. This is very important as a good choice of colours can make the use of the website more pleasant. Each colour has a different effect on people's mind, and each colour is used for a different purpose. In this website the dominant colour is blue. Blue is used to calm and relax the mind. The website presents at the top all the different categories of products. This facilitates the user experience, as he can easily find whatever he needs. At the top left corner, you can access the selling part. Firstly, you need to create an account and login. There are two buttons on the top-right corner of the page, register or login. To register, the user has to enter his name and surname, an email and create a password. The website also

asks your gender and how you found them. Being able to create an account is an essential feature, as it allows you to store your data somewhere and keep a register of what you have bought/sold. In the selling section the website shows you the items you are currently selling, their value. On the left-hand side of the page, there are all the options needed for your selling account, for example, your listings, your orders and your returns & cancellations.

The website is very well built, however there are somethings I would like to improve or change in order to make it more suitable to my project. When registering the user will only need to input his name, email and create a password. The password will be checked to see if it meets the requirements, if it doesn't the user will have to re-enter the password. I would remove the question about the gender and how they found the app, because I think they are not necessary and relevant in my case.
In the selling section I would add a graphs/charts section where it shows me how my selling is been going lately, if it is going well or bad. Also, it would show me which items are the most sold and which are the most looked at. This will help the seller understand his situation and it would work as a feedback, where he can see what items are selling better, so he can list them, and what items are not doing well.
The application will only have electronic related items, such as tv, phones, gaming consoles, therefore there will be less categories at the top.
When the user is on the homepage, he will be able to see all products and he will be able to filter them, for example "Price low to high" or "Highest rated".

End User Interview Questions

Identification of End User

This application would be used by two different people: buyers and sellers. The buyers will use the buying part of the application. Their feedback will be fundamental as I expect most of the user will be part of this group. The other type of end user is sellers. Sellers are as important as buyers, because without them there would be no items available to buy. Therefore, it is essentials that I get feedback and tips from my potential customers in order to create an efficient and functional application.

Interview Questions

Buyers:

1. **What is the first thing you notice when you open a buying and selling application?**
    - The design of the application is the first thing I notice alongside its reactiveness. If an application is not running smoothly, therefore it keeps freezing or it takes too long to load, I most probably will not use that application again. As of the design, I prefer a minimalistic design, with simple logos and images. The texts should be big enough in order to be visible to everyone, so it is not a problem for people with eyesight issues. The whole application should be based on a maximum of 5 matching colours.

2. **What would you like to see in the homepage of the application?**
    - I should be able to see a high-resolution image of the product, with the possibility of seeing two more images. There should be a short description of the

item, highlighting the main features of the product. Finally, the price should be there, clearly visible.

3. **Would you like the idea of having an account where all your data is saved, for example credit card details or shipping details?**
    - Yes, I think that would be a great addition. If the user could connect his account to his Facebook account or to his google account it would be exceptional, as it would make things easier. For credit card details, I will only save them if I know that my data is protected and secure. Therefore, I would like to know that my details are safe before saving them into an account.

4. **What other features would you like to see?**
    - When I'm on the homepage, I would like to be able to filter the products based on different aspects, for example price low to high or vice versa, or most sold to least sold. I would also like to see products categorized, for example smartphones all together, TVs all together etc.

Sellers:

1. **What do you think is the most important aspect of a selling account?**
    - The most important thing for me is being able to easily control my account. For example, I should be able to see what items I'm currently selling, how many I have sold, how many are still left, what items I already dispatched, what items I still need to dispatch, which items has been collected, which items still need to be collected. Lastly, if there are any returns questions.

2. **Would you like to see graphs, charts that show you how your business is going?**
    - I think that would be a great addition. A graph showing which of my items are the most sold or which are the most looked. This would help me, as a seller, decide which items I should keep selling and which I should stop.

3. **What other features would you like to see?**
    - I think there should be a way of communication between the seller and the customer. This could be, for example, the ability of posting comments underneath the products. This will work as a feedback for the seller, and it would give the possibility to the customer to ask questions.

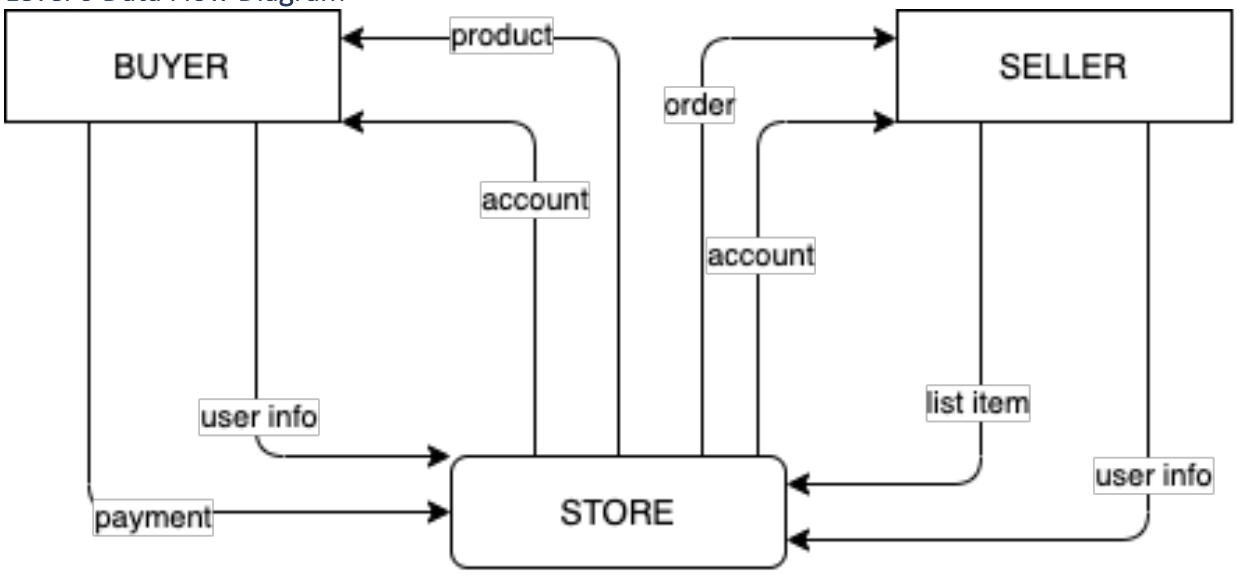
User Needs and Limitations

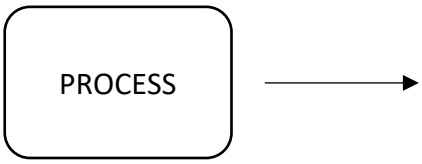
From the interview, we can understand what the users would like to see in the application. The first thing users notice is the design and aesthetics, as suggested above, therefore the application will have a minimalistic and simple design. This will make the application work more smoothly, giving a better experience to the user. When in the homepage, the user should be able to see clearly the product alongside its description and price. When opening the application, the user will have to create an account. This account will be used to save its data, like password and credit card details. As suggested above, the user will be able to filter the products which will be categorised.
On the other side, in the seller section the user will easily be able to see what products he's selling, how much he has spent and how much he made as profit. The user will be able to check which items have been delivered, but unfortunately, he will not be able to see if there

are any return request, as this feature is not present in the system. The seller sections will have graphs, which will help the user.

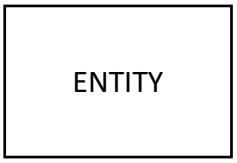
Data Flow Diagrams

Level 0 Data Flow Diagram

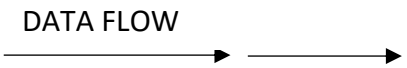




Store where a buyer can purchase items sold by a seller. Every buyer can also be a seller.

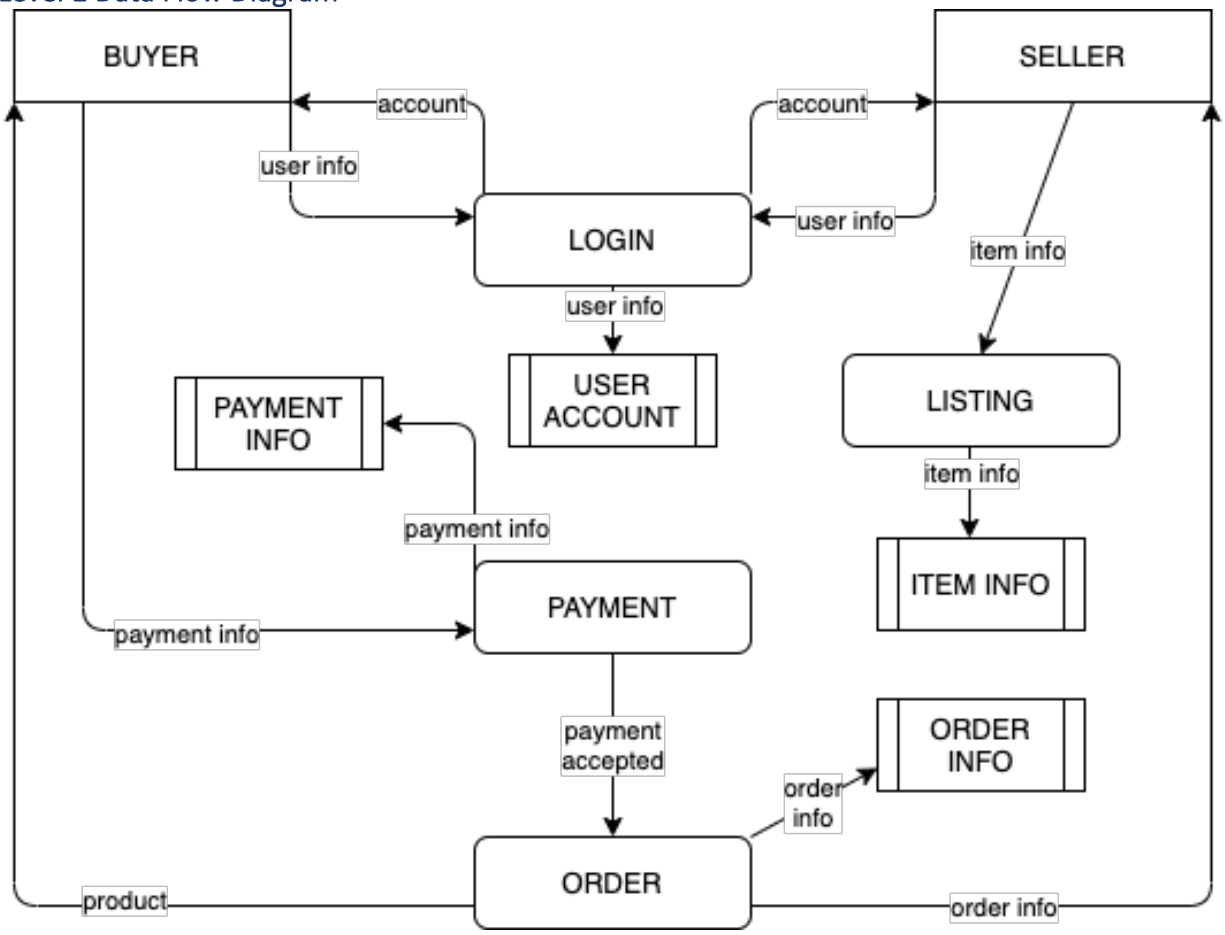


Different types of users who will make use of the system, in this case a buyer and a seller.



Indicates in which direction the data is going. “User info” means that the user inputs his name, email and password into the system and he receives an account back.

Level 1 Data Flow Diagram



PROCESS →

There are many processes in this case. First of all, the user needs to login, then he needs to make a payment in order to receive an order. The seller can list items. These are the main processes in this online store.

ENTITY →

Different types of users who will make use of the system, in this case a buyer and a seller.

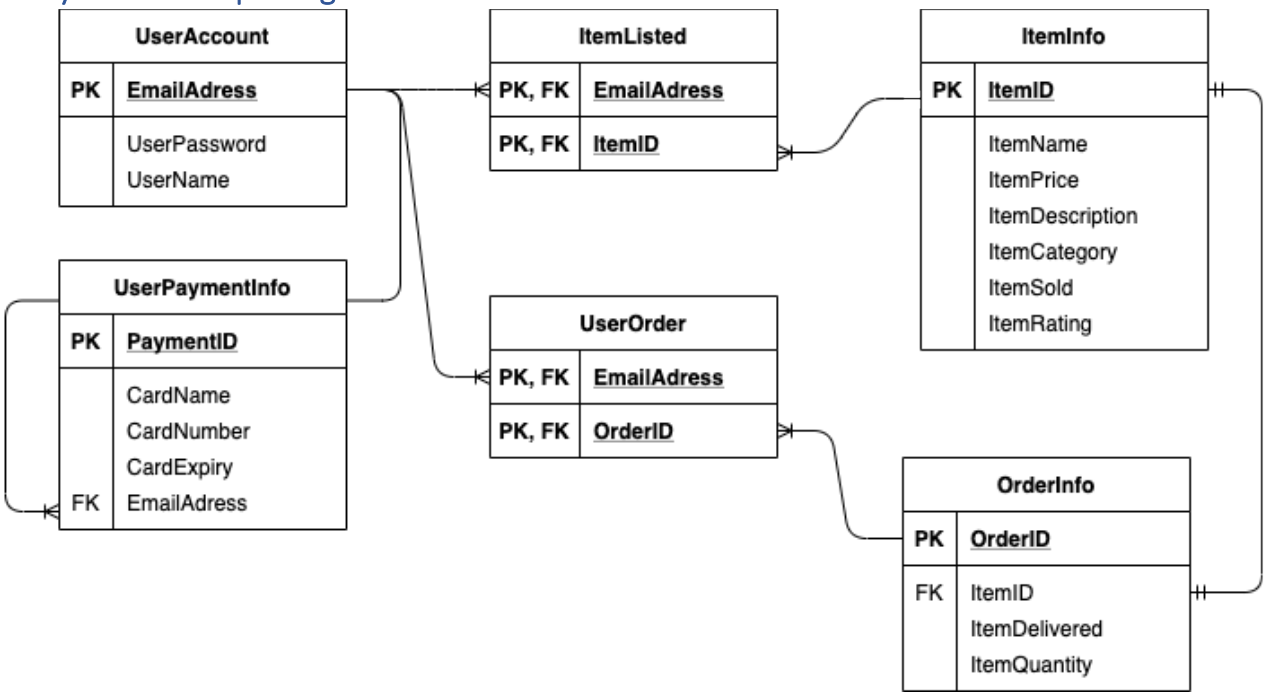
DATA FLOW → →

Indicates in which direction the data is going. For example, buyer inputs 'user info' which goes into the login process. The user then receives back an account from the login process.

DATA STORE →

This indicates where the data is stored. For example, the 'User Account' data store keep all the 'User Info'. Every time someone tries to login the system will check if the account already exists. If it is not present in the 'User Account' data store then the user has to register.

Entity Relationships Diagram



First of all, the user needs to enter his details in order to make an account. User details are stored in the entity “UserAccount”. If the user tries to register an already existent account, an error message will occur. When the user lists a new item, its info is stored in the entity “ItemInfo”. Since one user can list many items, there is a one-to-many relationship. A middle entity, “ItemListed”, is used to link what items each user has listed. One user can make many orders, so the relationship is one-to-many. Each order contains only one item; therefore, the relationship is one-to-one. “OrderInfo” entity is used to store the orders info, which are the itemID and whether it is delivered or not. A middle entity “UserOrder” is used to keep note of each order made by the user. Finally, when the user pays for an item, he has to enter his credit card credentials, which are stored in the “UserPaymentInfo” entity. Each user can have more than one payment method, so the relationship is one-to-many.

Data Dictionary

ENTITY NAME	DATA NAME	DATA TYPE	DESCRIPTION/LIMITATIONS	RELATIONSHIP
UserAccount	EmailAddress	String	Valid email entered by the user. Each user has one email. Two accounts cannot have the same email. The email can contain numbers but not symbols.	Primary Key
	UserName	String	Username. Cannot contain any numbers or symbols.	
	UserPassword	String	User’s password. Can contain letters, numbers and symbols. The password must be minimum 8 characters.	

ItemInfo	ItemID	Integer	Each item listed will have its own ID. The id will consist of 5 random digits. Two items cannot have the same ID.	Primary Key
	ItemName	String	The name of the item. It can contain numbers and symbols as well.	
	ItemPrice	Float	How much the item is selling for.	
	ItemDescription	String	A small description of the item sold. The maximum length is 100 characters.	
	ItemCategory	String	The categories available are “phones”, “laptops” and “TVs”.	
	ItemSold	Integer	How many of that item has been sold.	
	ItemRating	Float	Value must be between 0 and 5.	
ItemListed	EmailAddress			Primary Key Foreign Key
	ItemID			Primary Key Foreign Key
OrderInfo	OrderID	Integer	Each order made by the user has its own unique ID, represented by a value	Primary Key
	ItemID			Foreign Key
	ItemDelivered	Boolean	For this application, a random number will be generated to see whether the item is delivered or not. If the number is 1 then the item is delivered, and the value is going to be “true”. If the value is 0 then the is not delivered, and the values is going to be “false”	
	ItemQuantity	Integer	The user has to enter how many ‘pieces’ he wants to buy of that item. Value must be greater than 0.	
UserOrder	EmailAddress			Primary Key Foreign Key
	OrderID			Primary Key Foreign Key
UserPaymentInfo	PaymentID	Integer	Each payment has its own unique ID, represented by a value.	Primary Key
	CardName	String	Name on the credit card number. Can be different	

			from the name used by the user to register.	
	CardNumber	Integer	Must be a 13 digits long number. Cannot contain any letter or symbol. Must be a valid credit card number.	
	CardExpiry	Time	Any date that it is not the same as the day the user is inputting it.	
	EmailAddress			Foreign Key

Objectives

General Objectives

1. Create an application where users can buy or sell electronic devices.
2. Create a user-friendly interface with a minimalistic design.
3. Divide the application in two sections, one for the buyers and one for the sellers.
4. Give the opportunity to create an account to the user.
5. The application will be easy to navigate, run quickly and efficiently.

Specific Objectives

1. **Create an application where users can buy or sell electronic devices:**
   - The user will be able to buy different types of electronics devices, such as TVs, phones, gaming consoles and laptops.
   - On the other side, a seller will be able to sell different types of electronic devices.
2. **Create a user-friendly interface with a minimalistic design:**
   - The application will have a simple and minimalistic design. The text will be of adequate size, therefore clear and easy to read for everyone.
   - It will have a defined scheme of colour, which will be based on the colour blue with red accents to catch the user attention, which will calm the mind and aid concentration.
3. **Divide the application in two sections, one for the buyers and one for the sellers:**
   - After logging in, the user will be re-directed to the homepage. The homepage will be the buyer section. The user will be able to change to the seller by clicking the “sell” button on the top right corner.
   - In the homepage, the user will be able to see the products, their images and their description + price. The buyer will be able to search anything by using the search bar at the top. The buyer will also be able to filter his search by price, name, popularity or rating. At the top of the homepage, there will be different categories of products.
   - When buying a product, the user will have to enter an address and his credit card details. If the credit card is invalid, an error message is displayed and the user has to re-enter the details. If the payment is made successfully, an invoice will be displayed on-screen. The user can then take a screenshot of it.
   - In the user section, the user will have a menu at the left. From there, he will be able to see which products he is selling, what products he has sold, how much he has spent and how much profit he has made. There will be also a section with graphs. These graphs will show which products sold the most, which are making

the most profit. These should help the seller decide what to sell and what not to sell.

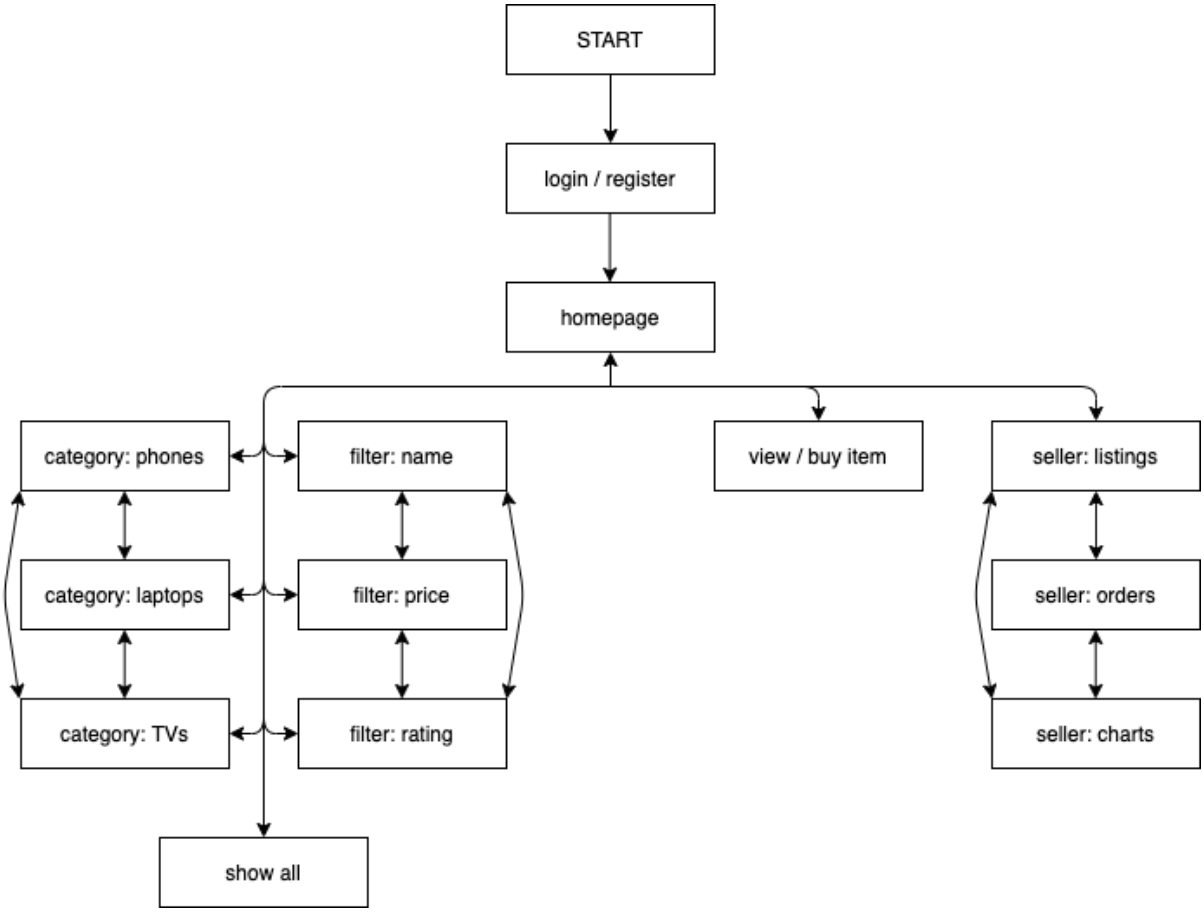
4. **Give the opportunity to create an account to the user:**
    - When the user opens the application, he will have to create an account or sign in if he already registered.
    - When registering, the user will have to enter his/her name, email and password. The password will have to meet requirements, if it doesn't the user will have to re-enter the password.
    - When logging in, the application will check that the email and password entered by the user match those in the database. If it doesn't, the user will have to re-enter email and password.

Proposed Solution

My application will be developed in VB.Net with the support of Microsoft Access, which will be used to create tables. VB.Net will be used to create the user interface, which includes the homepage, the seller page with graphs and charts, and the login functionality. I will be using Microsoft Access as well, as it facilitates the use of databases. In my project I have to store all the information that the user inputs, such as his login credentials or his credit cards details. This will make it easier to check if the data inserted by the user matches the one in the database, or when a new user creates an account it will be easier to keep record of it. It will also be useful when making charts and graphs for the seller. This will help me keep the project organised and easily adjustable.

DESIGN

Hierarchy chart



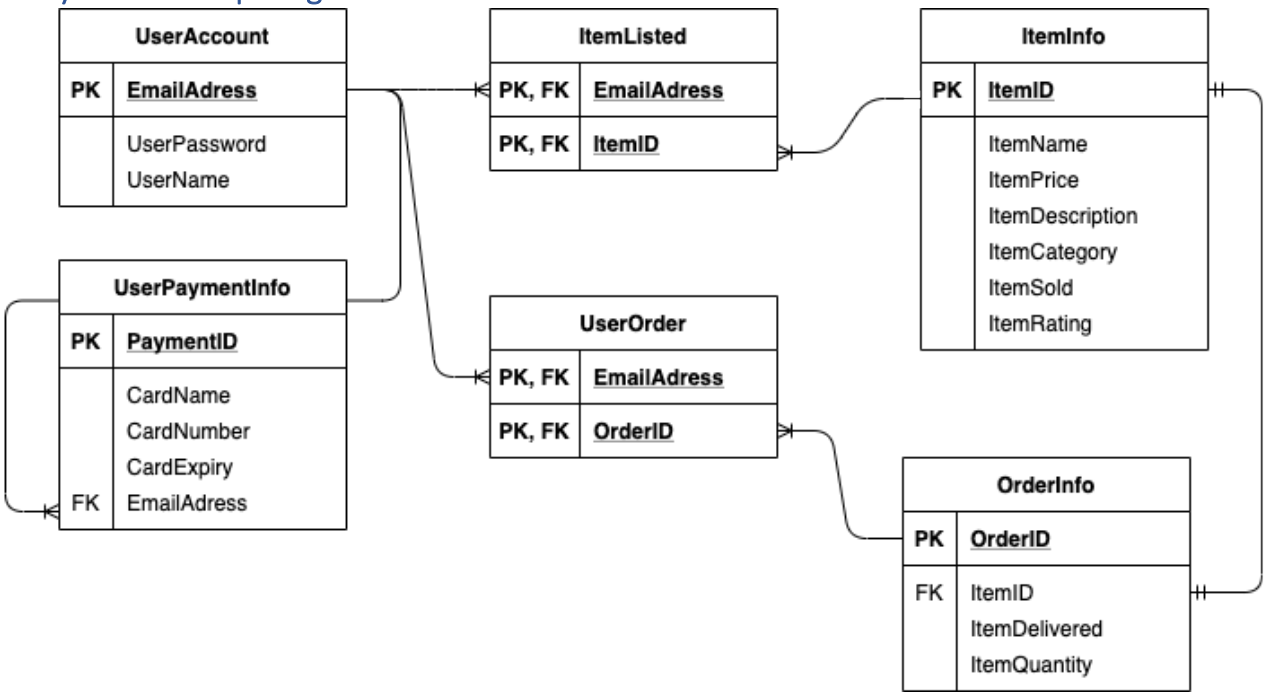
This is an overview of how the different modules in the system are linked together. When the application is opened, the login / register page shows up. Here the user can register himself and create an account to login. If the account is present in the database then the user can continue, if not then the user has to register. After logging in, the user ends up on the homepage. Here all the products are displayed. From the homepage, the user can filter and categories the product. The tree filters are price, name, rating. The categories are phones, laptops and TVs. The user can see all the item unorganised. From the homepage the user can view the item, which has a more detailed look on the product, and buy it in the same page. From every single page it is possible to go to seller section. The first page that comes up is the listing section. Here the user can see what item he has listed and list new items. The user can then go to the orders or charts section as he prefers. In the orders section, the user can see which order are delivered and which are not. In the charts section,

the user can see two tables. One shows the most sold items and the other shows the highest rated items. The user can always go to the homepage from the seller section.

IPSO Table

INPUT	PROCESS	STORAGE	OUTPUT
Name	User Registration	Storing name, email and password in 'UserAccount' entity	Items listed info
Email	User Login	Storing credit card name, number and expiry in 'UserPaymentInfo' entity	User listed items table
Password	Connect with database	Storing item name, price, description, category, rating in 'ItemInfo' entity	User delivered orders table
Credit Card Name	Display Items		User not delivered orders table
Credit Card Number	Show items filtered by price		User most sold items table
Credit Card Expiry Date	Show items filtered by name		User highest rated items table
Item Name	Show items filtered by rating		
Item Price	Show phones category items		
Item Description	Show laptops category items		
Item Category	Show TVs category items		
Item Rating	Show items info		
Item Quantity	Take user payment info		
	Confirm payment		
	Show seller listing section		
	List new item		
	Show seller order section		
	Show seller charts section		

Entity Relationship Diagram



Normalization

Normalization is a process used to come up with the best possible design for a database. Tables Should be organised so that there are no duplicate items in any table. The process should allow complex queries to be made. There are three different stages for normalization: First, Second and Third Normal Form (1NM, 2NM and 3NM).
A table is in 1NM when there are no repeating attributes in the same table. All attributes must be atomic, meaning they can only consist of one data item.
A table is in 2NM when it is in 1NM and it contains no partial dependencies. This can happen when there is a composite keys table, for example “ItemListed”.
A table is in 3NM when it is in 2NM and there are no non-key dependencies, meaning every attribute depends on the primary key only.

UNF – Unnormalized Form

(EmailAddress, UserPassword, UserName, Item1, Item2, ItemName1, ItemName2, ItemPrice1, ItemPrice2, ItemDescription1, ItemDescription2, ItemCategory1, ItemCategory2, ItemSold1, UtemRating1, ItemRating2, Payment1, Payment2, CardName1, CardName2, CardNumber1, CardNumber2, CardExpiry1, CardExpiry2, Order1, Order2, ItemDelivered1, ItemDelivered2)

This is the single unnormalized entity for the database. There are several repeating items, so the database must be normalized.

1NM – First Normal Form

UserAccount (EmailAddress, Password, Name)

ItemInfo (ItemID, ItemName, ItemPrice, ItemDescription, ItemCategory, ItemSold, ItemRating)
UserPaymentInfo (PaymentID, CardName, CardNumber, CardExpiry)
OrderInfo (OrderID, ItemDelivered, ItemQuantity, *ItemID*)

All the repeating attributes have been placed in different entities. Because some relationships are one to many or many to many, the database should be in second form.

2NM – Second Normal Form

UserAccount (EmailAddress, Password, Name)
ItemListed (*EmailAddress, ItemID*)
ItemInfo (ItemID, ItemName, ItemPrice, ItemDescription, ItemCategory, ItemSold, ItemRating)
UserPaymentInfo (*PaymentID,* CardName, CardNumber, CardExpiry, *EmailAdress*)
OrderInfo (OrderID, ItemDelivered, ItemQuantity, *ItemID*)
UserOrder (*EmailAddress, OrderID*)

New entities have been created where there was a many to many relationships. As there are no non-key dependencies, there is no need for a third normal form.

Data Dictionary Validation

ENTITY NAME	DATA NAME	DATA TYPE	DATA EXAMPLE	DATA VALIDATION
UserAccount	EmailAddress	String	Abc123@gmail.com	The email entered must have the '@' sign, and it should end with '.com'
	UserName	String	Hamza	The name cannot have numbers and signs in it
	UserPassword	String	Password1	Password length must be between 8-15 characters long. It can contain letters, numbers and signs.
ItemInfo	ItemID	Integer	1	Consecutive number generated by the computer. Starts from 1 and goes on. Each item has a different ID number.
	ItemName	String	iPhone	Can be anything
	ItemPrice	Float	£9.00	Must be a decimal number. If it is a whole number, it should end with '.00'

	ItemDescription	String	iPhone 12 mini. 64GB storage.	Small description of maximum 100 characters.
	ItemCategory	String	Phones	The tree available categories are ‘phones’, ‘laptops’ and ‘TVs’
	ItemSold	Integer	1	Every time an order is made, the number increases by 1.
	ItemRating	Integer	2	Integer between 0 and 5.
ItemListed	EmailAddress			
	ItemID			
OrderInfo	OrderID	Integer	2	Consecutive number generated by the computer. Starts from 1 and goes on. Each order has a different ID number.
	ItemID			
	ItemDelivered	Boolean	‘True’	Can either be ‘true’ or ‘false’
	ItemQuantity	Integer	5	Any integer greater than 0.
UserOrder	EmailAddress			
	OrderID			
UserPaymentInfo	PaymentID	Integer	1	Consecutive number generated by the computer. Starts from 1 and goes on. Each order has a different ID number.
	CardName	String	Hamza	Cannot contain any number or sign. Can be different from name used for the account.
	CardNumber	Integer	1234567812345678	Length must be 16 digits.
	CardExpiry	Time	01/09/2025	Date should be in the format ‘dd/mm/year’
	EmailAddress			

HCI

HCI design overview

The whole system will be based around 3 colours: white, light blue and dark blue. White will be used for the background as it a neutral colour, which signify cleanliness and purity, as well as being a colour easy to match. Light blue will be used for buttons, and dark blue will be used at the top as the background for the logo and the search bar. I have decided to use

the colour blue for its effects and influences on the mind and feelings of people. Blue calls to mind feelings of serenity and calmness. It is often seen as a sign of security and reliability. I chose to use a lighter version of blue for the buttons because it is easier to see, and it draws the user attention to it. Therefore, when the user opens up the program he knows exactly where to press. The darker blue is used to create contrast with the main part of the program, which has a white background.

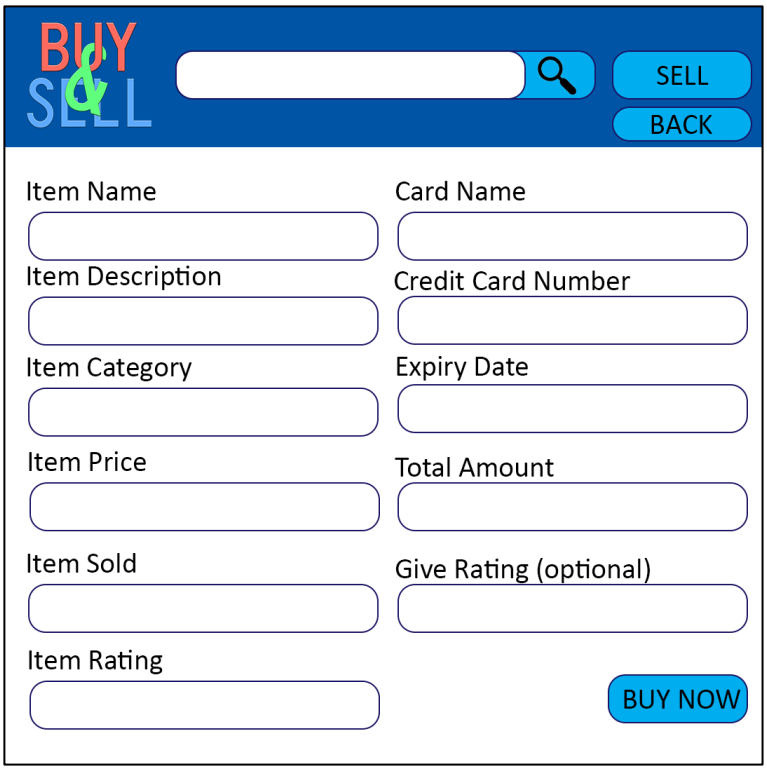
The font I will be using is Century Gothic. It is clean and minimalistic, which resembles the overall aesthetic of the program. Since it is commonly used, the user will not have difficulties when reading. When writing titles or heading the text will be in uppercase while the rest will be in lowercase.

In conclusion, the program will have a minimalistic design. The whole program will be based around 3 colours and 1 font, so that it looks nicer to the eye and makes the overall experience more enjoyable.

HCI sample design

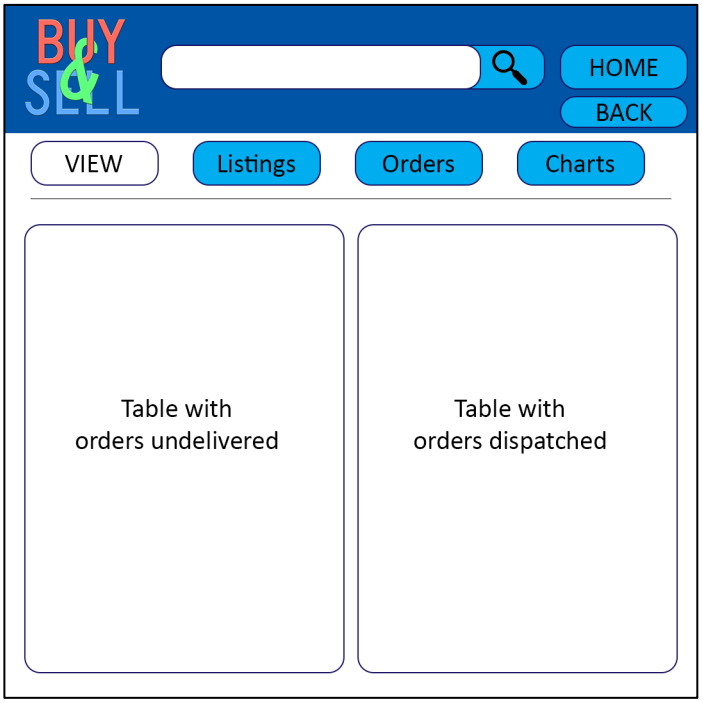
This is the login page. When the user starts the application, this is where he will end up first. The screen is divided in two sections. At the top we can find the logo, which has no functionality and it can't be pressed, it just represents the name of the application. At the bottom we find the registration section, on the left side, and the login section, on the right side. If the user uses the application for the first time, he will have to register first by entering his name, email and password. When the user presses the "Register" buttons, the credentials are saved in the database and the user now has an account. After registering, the user can login, by entering his email and password and pressing the "Login" button. If the email and password are the same as in the database, the user will access the application.

This is the homepage. From here the user can look at items, search items and go to the seller section. At the top we find a search bar where the user can enter an item he wants to see and if it is present in the database it will show up, otherwise no item will show up. Next to it there is the “Sell” button, which will take the user to the seller section. Down we have the main page. The filter section allows the user to filter the item in order by “price”, which is going to be lower to higher, “name”, in alphabetical order, and “rating”, from higher to lower. In the category section the user will be able to filter items by categories, by clicking on the type of item he wants to see. At the bottom the item will be displayed. For each item the ID, name, description, price, sold and rating will be present. Each item will have a “view item” button, which will lead to a detailed look at the product and payment.



When the user clicks the “view item” button on the homepage this is the page that comes up. In the upper section the “back” button takes you back to the homepage. On the left side all the information of the item will be shown. On the right side the user has to enter his payment details, card name, card number and expiry date. The “buy now” button allows the user to buy the item. At the bottom the user can give a rating to the item by inserting a number between 0 and 5, however this is optional and the user is not forced to do it.

This is the homepage of the seller section. It is the first page that comes up when clicking the “sell” button at the top. Here the user can see his listings already made in a table on the left side. On the right side the item can list a new item by entering name, description, price, category and images of the item and afterwards pressing the “list item” button. At the top the “home” button takes the user back to the homepage.



This is the orders section. Here the user can see his undelivered orders on the left and on the right his orders already dispatched.

This is the last part in the seller section. Here the user can see a table ordered by most sold items on the left side. On the right side a table ordered by highest rated items.

SQL Queries Sample

User Registration: Add user info into database to create an account

InEmail, InPassword, InName = User Input

‘INSERT INTO UserAccount(EmailAddress, Password, Name) VALUES (InEmail, InPassword, InName)

User Login: Check if the user info matches the one in the database

InEmail, InPassword = User Input

‘SELECT EmailAddress AND Password
FROM UserAccount(EmailAddress, Password, Name)
WHERE UserAccount.EmailAddress = InEmailAddress AND
UserAccount.Password = InPassword’

If nothing is selected, then it means that the user has to create an account first or the credentials are wrong.

Search Item: When the user searches for an item, the system will check if there are any similar items in the database and will output them

Search = User Input

‘SELECT * FROM ItemInfo WHERE ItemInfo.ItemName LIKE %Search%”

If nothing is selected, then it means there are no similar items.

Filter Items by Price: The system will filter all the items available in crescent order of price, from the cheapest to the most expensive.

‘SELECT * FROM ItemInfo ORDER BY ItemPrice ASC’

Filter Items by Name: The system will filter all the items available in alphabetically order.

‘SELECT * FROM ItemInfo ORDER BY ItemName ASC’

Filter Items by Rating: The system will filter all the items available in decrescent order of rating, from the highest rated to the lowest.

‘SELECT * FROM ItemInfo ORDER BY ItemRating DESC’

Display Phones Category: Select all the items in the phones category

‘SELECT * FROM ItemInfo WHERE ItemCategory LIKE “Phones” ‘

Display Laptops Category: Select all the items in the laptops category

‘SELECT * FROM ItemInfo WHERE ItemCategory LIKE “Laptops” ‘

Display TVs Category: Select all the items in the TVs category

‘SELECT * FROM ItemInfo WHERE ItemCategory LIKE “TVs” ‘

Display item in the homepage: All the items in the database will be shown in the homepage

FOR EACH ItemID IN ItemInfo

‘SELECT * FROM ItemInfo’

Update item rating in the database: When the user gives a rating, the item rating will be updated in the database.

NewItemRating = User Input
ItemIDViewed = Item ID of the item the user wants to see

‘UPDATE ItemInfo SET ItemRating = (ItemRating + NewItemRating) /2 Where ItemID = ItemIDViewed’

Add payment info into database: Check if the user is entering his payment info for the first time.

InCName, InCNumber, InCExpiry = User Input
‘INSERT INTO UserPaymentInfo(CardName, CardNumber, CardExpiry) VALUES (InCName, InCNumber, InCExpiry)’

Show table with item listed: Show all the items listed by the user

‘SELECT * FROM ItemInfo WHERE ItemInfo.ItemID = ItemListed.ItemID AND ItemListed.EmailAddress = Login.EmailAddress’

Add new item in the database: When the user lists a new item, it is added into the database

InIName, InIPrice, InIDescription, InICategory, InIImages = User Input

‘INSERT INTO ItemInfo(ItemName, ItemPrice, ItemDescription, ItemCategory)
VALUES (InIName, InIPrice, InIDescription, InICategory,)’

Show undelivered items table: The user can view which orders are still undelivered

‘SELECT * FROM OrderInfo WHERE ItemDelivered = “false”’

Show dispatched items table: The user can see which orders have been dispatched.

'SELECT * FROM OrderInfo WHERE ItemDelivered = "true"

Show listed items in order of most sold: The user can see which of his listed items have sold the most

'SELECT * FROM ItemInfo WHERE ItemInfo.ItemID = ItemListed.ItemID
ORDERED BY ItemSold DESC'

Show listed items in order of rating: The user can see which of his listed items have the highest rating

'SELECT * FROM ItemInfo WHERE ItemInfo.ItemID = ItemListed.ItemID
ORDERED BY Rating DESC'

Pseudocode

In this section I will write a pseudocode for each HCI sample screen, which can be found in the 'HCI' section.

Registration/Login: In the first screen the user can register if it is the first time he is using the application, or login if he already registered.

```
#registration
InName, InEmail, InPassword = USER INPUT
Registration = 'true'

While Registration = 'true'
 For Each EmailAddress in UserAccount
    If EmailAddress = InEmail then
      Output -> "Account already exists, please login or use another email address."
      Registration = 'false'
 End For
End While

If Registration = 'true' then
  'INSERT INTO UserAccount(EmailAddress, Password, Name) VALUES (InEmail, InPassword,
InName)'

#login
InEmail, InPassword = USER INPUT
Login = 'false'

While Login = 'false'
  For Each EmailAddress in UserAccount
   If EmailAddress = InEmail and Password = InPassword then
    Output -> "Login successful"
```

```
    Login = 'true'
    UserOnline = InEmail
  End For
End While

If Login = 'false' then
   Output -> "Unsuccessful login. Please enter email and password again"
```

Homepage: The homepage is composed of many small functions, which are search bar, view filtered item, view item by category, view all items and go into the seller page.

```
#searchbar
Search = USER INPUT
DisplayItem = dictionary (name as string, description as string, price as float, rating as float, image as long)


For Each ItemID in ItemInfo
  DisplayItem.Add('SELECT * FROM ItemInfo WHERE ItemInfo.ItemName LIKE Search')
  Output -> DisplayItem
End For

If result of query is null then
Output -> "There are no items that matches your search"


#filtered items
Filter = USER INPUT

If Filter = 'Price' then
 For Each ItemID in ItemInfo
  DisplayItem.Add('SELECT * FROM ItemInfo ORDER BY ItemPrice ASC')
  Output -> DisplayItem
Elseif Filter = 'Name' then
 For Each ItemID in ItemInfo
  DisplayItem.Add('SELECT * FROM ItemInfo ORDER BY ItemName ASC')
  Output -> DisplayItem
Elseif Filter = 'Rating' then
 For Each ItemID in ItemInfo
  DisplayItem.Add('SELECT * FROM ItemInfo ORDER BY ItemRating DESC')
  Output -> DisplayItem
End For

#view item by category
Category = USER INPUT

If Category = 'Phones' then
```

```
 For Each ItemID in ItemInfo
  DisplayItem.Add('SELECT * FROM ItemInfo WHERE ItemCategory LIKE "Phones" ')
  Output -> DisplayItem
Elseif Category = 'Laptops' then
 For Each ItemID in ItemInfo
  DisplayItem.Add('SELECT * FROM ItemInfo WHERE ItemCategory LIKE "Laptops" ')
  Output -> DisplayItem
Elseif Category = 'TVs' then
 For Each ItemID in ItemInfo
  DisplayItem.Add('SELECT * FROM ItemInfo WHERE ItemCategory LIKE "TVs" ')
  Output -> DisplayItem
End For

If result of query is null then
Output -> "There are no items in this category"


#view all items
For Each ItemID in ItemInfo
  DisplayItem.Add('SELECT * FROM ItemInfo)
  Output -> DisplayItem
End For

If result of query is null then
Output -> "There are no items available right now"

#go to seller section
Go to List new item/Show user listings section
```

View/buy item: In this screen the user can see the item he wants solely. The user can then add a rating.

```
#display info of item viewed
ItemIDViewed = Item ID of the item the user wants to see

Output -> 'SELECT * FROM ItemInfo WHERE ItemID = ItemIDViewed'
InRating = USER INPUT

#update rating of item in the database
'UPDATE ItemInfo SET ItemRating = (ItemRating + InRating) / 2 WHERE ItemID =
ItemIDViewed'

#add payment info
InCName, InCNumber, InCExpiry = USER INPUT
ValidCard = 'False'
```

If length InCNumber = 16 AND InCExpiry > Date.Today then
  ValidCArd = ‘True’

If ValidCard = ‘True’ then
  ‘INSERT INTO UserPaymentInfo(CardName, CardNumber, CardExpiry) VALUES (InCName, InCNumber, InCExpiry) ’

#increase number of items sold in database
‘UPDATE ItemInfo SET ItemSold = ItemSold + 1 WHERE ItemID = ItemIDViewed’

#create new order if payment is successful
ItemDelivery = Random number between 0 and 1
For Each EmailAddress in ItemListed
 If ItemID = ItemIDViewed then
  InEmailAddress = EmailAddress
End For
If ItemDelivery = 1 then
  InItemDelivered = ‘True’
Else
  InItemDelivered = ‘False’

‘INSERT INTO OrderInfo(ItemID, ItemDelivered) VALUES (ItemIDViewed, InItemDelivered) WHERE UserOrder.EmailAddress = InEmailAddress’

List new item/Show user listings: Here the user will be able to see in a table the item he has listed and list new items.

#view user listed items table
Output -> ‘SELECT * FROM ItemInfo WHERE ItemInfo.ItemID = ItemListed.ItemID AND EmailAddress = UserOnline’

If result of query is null then
  Output -> “You have no item listed”

#list new item
InIName, InIPrice, InIDescription, InICategory, = USER INPUT
ValidListing = ‘False’

If InICategory = “Phones” or “Laptops” or “TVs” AND length(InIDescription) <= 100 then
  ValidListing = ‘True’

If ValidListing = ‘True’ then
  ‘INSERT INTO ItemInfo(ItemID, ItemName, ItemPrice, ItemDescription, ItemCategory) VALUES (ItemID+1,InIName, InIPrice, InIDescription, InICategory) ’
  ‘INSERT INTO ItemListed(EmailAddress, ItemID) Values (UserOnline, ItemID+1)

Show user orders: The user can see two tables. The first has all the undelivered orders, the other has all the dispatched orders.

#undelivered orders table
Output -> 'SELECT * FROM UserOrder WHERE EmailAddress = UserOnline AND OrderInfo.ItemDelivered = 'True')

#Dispatched orders table
Output -> 'SELECT * FROM UserOrder WHERE EmailAddress = UserOnline AND OrderInfo.ItemDelivered = 'False')

Show user charts: In this section the user will see two tables. One will show all the item listed ordered by most sold, the other all the item listed ordered by highest rated.

#most sold items table
Output -> 'SELECT * FROM ItemInfo WHERE ItemInfo.ItemID = ItemListed.ItemID AND ItemListed.EmailAddress = UserOnline ORDERED BY ItemSold DESC'

#highest rated items table
Output -> 'SELECT * FROM ItemInfo WHERE ItemInfo.ItemID = ItemListed.ItemID AND ItemListed.EmailAddress = UserOnline ORDERED BY Rating DESC'

Test Strategy

Three types of data are usually used to test:

- **Normal:** typical data that the user would normally enter. For example, number 5 in the range 0 to 10.
- **Boundary:** data that is on the extreme of a range, which could cause errors. For example, numbers 0 and 10 in the range 0 to 10.
- **Erroneous:** data that is outside of the range, which should definitely cause errors. For example, number 15 in the range 0 to 10.

When I will be testing my application, I will use all three types of data to see that the program works as it should.

TEST NUMBER	TEST DESCRIPTION	EXPECTED OUTCOME
LOGIN PAGE		
1a	Check if all the data validation works and an error comes up when the data is not valid	All the validations should take place and all the error messages should show up
1b	Check that the email the user is trying to register does not exist in the database, if it already exists an error message should come up	If the email already exists than the user should not be able to register

1c	Check that the register button works only if all the data inserted is valid, an error message should come up if that is not the case	Only when all the data inserted is correct/valid the button should work
1d	Check that the data is inserted in the right table in the database	The data is inserted in the table UserAccount and the values are in the right columns
1e	Check that the email the user is trying to login with exists in the database and the password matches when the user tries to login, if that is not the case then an error message should come up	The login should only happen when the email already exists in the database and the password matches
HOMEPAGE		
2a	Check that Sell button works	The Seller-Listings page should come up
2b	Check that the Show All button works	All the item listed should show on the Data Grid unsorted
2c	Check that the Search button works	Every item that has the “searched word” in their name should show on the Data Grid
2d	Check that the Price button works	The items in the list should be filtered by price low to high
2e	Check that the Name button works	The items in the list should be filtered by name alphabetically
2f	Check that the Rating button works	The items in the list should be filtered by rating high to low
2g	Check that the Phones button works	Only the items in the Phones category should show
2h	Check that the Laptops button works	Only the items in the Laptops category should show
2i	Check that the TVs button works	Only the items in the TVs category should show

2j	Check that the View Item at the end of each row works and the right values are passed to the next form	A new page should come up where the user can see the details of the item he wanted to see
VIEW/BUY ITEM		
3a	Check that the right values of the item are displayed and the user can't edit them	The details of the item that the user wants to see should be displayed in the right textboxes and the user should not be able to edit them
3b	Check that the Sell button works	The Seller-Listings page should come up
3c	Check that the Back button works	The Homepage should come up
3d	Check that the Search button works	An error message should come up
3e	Check if all the data validation works and an error comes up when the data is not valid	All the validations should take place and all the error messages should show up
3f	Check that the Buy button works and the data is stored correctly in the database	When the user presses the Buy button, the user payment info should be stored in the UserPaymentInfo table. The itemsold and itemrating fields should be updated in the ItemInfo table. A new order should be created, the itemID and whether it was delivered or not should be stored in the OrderInfo table, the seller email and orderID should be stored in the UserOrder table. If the data is not valid then an error message should come up.
SELLER-LISTINGS		
4a	Check that the Home button works	The Homepage should come up
4b	Check that the Search button works	An error message should come up
4c	Check that the Listings button works	An error message should come up

4d	Check that the Orders button works	The Seller-Orders page should come up
4e	Check that the Charts button works	The Seller-Charts page should come up
4f	Check that the Show button works	All the item listed by the user should show in the DataGrid
4g	Check if all the data validation works and an error comes up when the data is not valid	All the validations should take place and all the error messages should show up
4h	Check that the List Item button works, and the data is stored correctly in the database	When the user presses the List Item button, all the item values (name, description, price, category) should be stored in the ItemInfo table, the user email and itemID of the item just stored should be stored in the ItemListed table. If the data is not valid then an error message should come up.
SELLER-ORDERS		
5a	Check that the Home button works	The Homepage should come up
5b	Check that the Search button works	An error message should come up
5c	Check that the Listings button works	The Seller-Listings page should come up
5d	Check that the Orders button works	An error message should come up
5e	Check that the Charts button works	The Seller-Charts page should come up
5f	Check that the left Show button works	All the orders received by the user which have been delivered should show on the Data Grid
5g	Check that the right Show button works	All the orders received by the user which have been delivered should show on the Data Grid
SELLER-CHARTS		
6a	Check that the Home button works	The Homepage should come up
6b	Check that the Search button works	An error message should come up

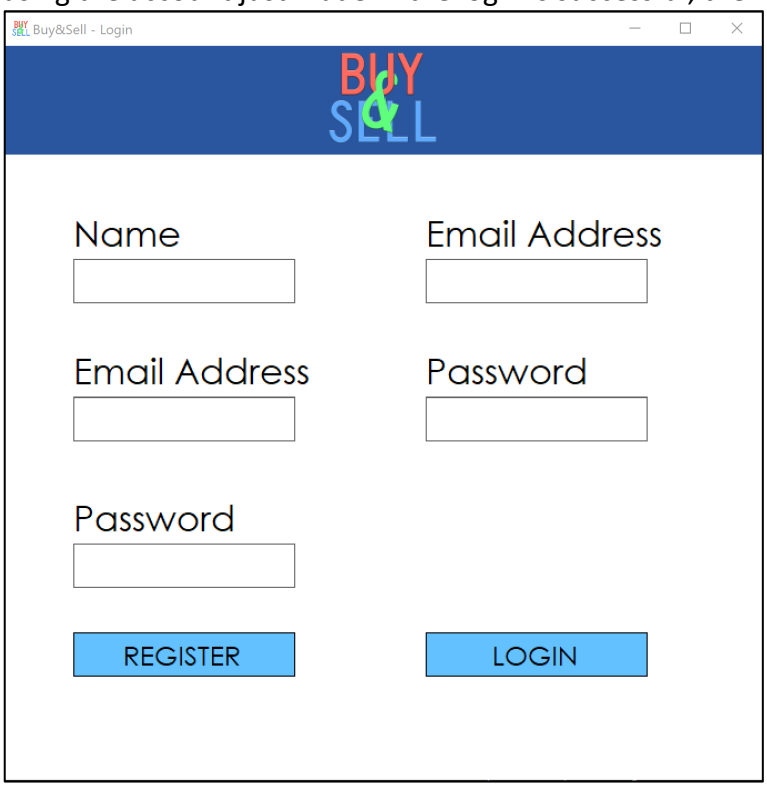
6c	Check that the Listings button works	The Seller-Listings page should come up
6d	Check that the Orders button works	The Seller-Orders page should come up
6e	Check that the Charts button works	An error message should come up
6f	Check that the left Show button works	All the items listed by the user filtered by most sold should show on the Data Grid
6g	Check that the right Show button works	All the items listed by the user filtered by most rated should show on the Data Grid

TECHNICAL SOLUTION

I have implemented my project by using Windows Forms. The project has been coded in VB.Net. I have used Microsoft Access to create the database, as it works efficiently with the coding language. I have used SQL Statements to read/write data in the database.

Login

In this form, the user can register to create an account. After registering, the user can login using the account just made. If the login is successful, the homepage form will be loaded.




```
1.  Imports System.Text.RegularExpressions
```

```
2.  Imports System.Data.OleDb
3.  Public Class Form1
4.
5.      'When user presses the register button, values are stored in the database
6.      Private Sub RegisterBtn_Click(sender As Object, e As EventArgs) Handles Registe
    rBtn.Click
7.
8.          If RegistrationDetailsCheck() = True And ExistingEmailCheck() = False Then
     'Check if values entered by users are valid
9.              Dim con As New OleDb.OleDbConnection
10.
11.             Dim cmd As New OleDb.OleDbCommand
12.
13.             'Open a connection with the database to insert the values in the right
    table
14.             con.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=P
    rojectDatabase.accdb;Persist Security Info=True"
15.
16.             con.Open()
17.
18.             Dim Sql As String = "INSERT INTO UserAccount(EmailAddress,UserPassword,
    UserName) VALUES('" & Me.RegisterEmail.Text & "','" & Me.RegisterPassword.Text & "'
    , '" & Me.RegisterName.Text & "' )"
19.             cmd = New OleDb.OleDbCommand(Sql, con)
20.             cmd.ExecuteNonQuery()
21.
22.             MsgBox("You have been registered! You can now login.")
23.
24.             'Remove any values in the Email,Password,Name textbox
25.             RegisterEmail.Text = ""
26.             RegisterPassword.Text = ""
27.             RegisterName.Text = ""
28.
29.             con.Close()  'Close connection
30.         ElseIf ExistingEmailCheck() = True Then  'If the email already exists in th
    e databse, he has use another one
31.             MsgBox("Email already exists! Please use another one.")
32.
33.             RegisterEmail.Text = ""
34.         Else
35.             MsgBox("Please enter valid details in order to register!")
36.
37.             RegisterEmail.Text = ""
38.             RegisterPassword.Text = ""
39.             RegisterName.Text = ""
40.         End If
41.
42.     End Sub
43.
44.     'Check if the values entered in the Registration Name, Email and Password textb
    ox are correct
45.     Public Function RegistrationDetailsCheck() As Boolean
46.         If Me.NameErrorProvider.GetError(Me.RegisterName) = "" And Me.EmailErrorPro
    vider.GetError(Me.RegisterEmail) = "" And Me.PasswordErrorProvider.GetError(Me.Regi
    sterPassword) = "" Then
47.             Return True
48.         Else
49.             Return False
50.         End If
51.     End Function
52.
53.     'Check if the Email entered by the user already exists in the database
54.     Public Function ExistingEmailCheck() As Boolean
55.         Dim Table_ As String = "UserAccount"
56.         Dim query As String = "SELECT EmailAddress FROM UserAccount WHERE EmailAddr
    ess = '" & Me.RegisterEmail.Text & "'"
```

```
57.
58.         'Open a connection with the database, and create a table with the email equ
    al to the user input, if it exists
59.         Dim MDBConnString_ As String = "Provider=Microsoft.ACE.OLEDB.12.0;Data Sour
    ce=ProjectDatabase.accdb;Persist Security Info=True"
60.         Dim ds As New DataSet
61.         Dim cnn As OleDbConnection = New OleDbConnection(MDBConnString_)
62.         cnn.Open()
63.         Dim cmd As New OleDbCommand(query, cnn)
64.         Dim da As New OleDbDataAdapter(cmd)
65.         da.Fill(ds, Table_)
66.         cnn.Close()
67.
68.         'If the table has 0 rows, it means that the email does not exists in the da
    tabse and can be used
69.         If ds.Tables(0).Rows.Count > 0 Then
70.             Return True
71.         Else
72.             Return False
73.         End If
74.
75.     End Function
76.
77.     'The user can only input characters in the Name textbox, any other input is not
     accepted
78.     Private Sub RegisterName_KeyPress(sender As Object, e As KeyPressEventArgs) Han
    dles RegisterName.KeyPress
79.         If Char.IsNumber(e.KeyChar) = True Or Char.IsPunctuation(e.KeyChar) = True
    Or Char.IsSymbol(e.KeyChar) = True Then
80.             e.Handled = True
81.         End If
82.     End Sub
83.
84.     'Check if there is a value in the Name textbox, if there is not an error comes
    up
85.     Private Sub RegisterName_Validating(sender As Object, e As System.ComponentMode
    l.CancelEventArgs) Handles RegisterName.Validating
86.         If Len(Me.RegisterName.Text) = 0 Then
87.             Me.NameErrorProvider.SetError(Me.RegisterName, "Please enter a name!")

88.             Return
89.         Else
90.             Me.NameErrorProvider.SetError(Me.RegisterName, "")
91.         End If
92.     End Sub
93.
94.     'Check if the Email entered is in the right format, for example aa@aa.aa
95.     Public Function RegistrationValidateEmail(RegisterEmail) As Boolean
96.         Dim email As New Regex("([\w-+]+(?:\.[\w-+]+)*@(?:[\w-]+\.)+[a-zA-
    Z]{2,7})")
97.         If email.IsMatch(Me.RegisterEmail.Text) Then
98.             Return True
99.         Else
100.                Return False
101.            End If
102.        End Function
103.
104.        'If the email is not in the right format an error comes up
105.        Private Sub RegisterEmail_Validating(sender As Object, e As System.Compo
    nentModel.CancelEventArgs) Handles RegisterEmail.Validating
106.            If RegistrationValidateEmail(Me.RegisterEmail.Text) = False Then
107.                Me.EmailErrorProvider.SetError(Me.RegisterEmail, "Enter a valid
    email!")
108.                Return
109.            Else
110.                Me.EmailErrorProvider.SetError(Me.RegisterEmail, "")
```

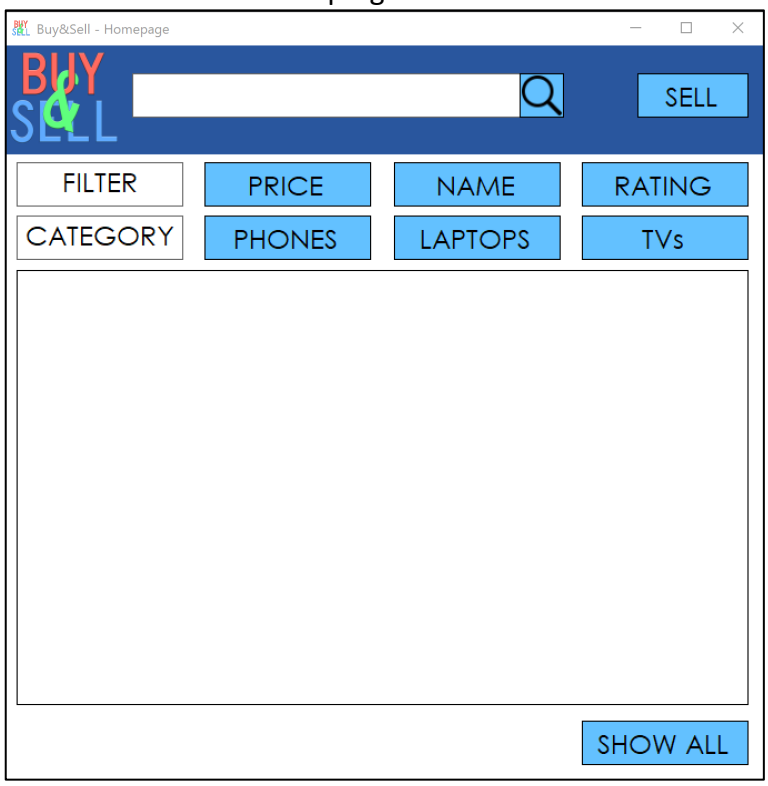
```
111.                End If
112.            End Sub
113.
114.            'Check the password length, if it is not between 8-
    15 characters an error comes up
115.            Private Sub RegisterPassword_Validating(sender As Object, e As System.Co
    mponentModel.CancelEventArgs) Handles RegisterPassword.Validating
116.                If Len(Me.RegisterPassword.Text) = 0 Then
117.                    Me.PasswordErrorProvider.SetError(Me.RegisterPassword, "Please e
    nter a password! Must be 8-15 characters")
118.                    Return
119.                ElseIf Len(Me.RegisterPassword.Text) < 8 Then
120.                    Me.PasswordErrorProvider.SetError(Me.RegisterPassword, "Password
     is too short! Must be 8-15 characters")
121.                    Return
122.                ElseIf Len(Me.RegisterPassword.Text) > 15 Then
123.                    Me.PasswordErrorProvider.SetError(Me.RegisterPassword, "Password
     is too long! Must be 8-15 chracters")
124.                    Return
125.                Else
126.                    Me.PasswordErrorProvider.SetError(Me.RegisterPassword, "")
127.                End If
128.            End Sub
129.
130.            'Check if the Email entered is in the right format, for example aa@aa.aa

131.            Public Function LoginValidateEmail(LoginEmail) As Boolean
132.                Dim email As New Regex("([\w-+]+(?:\.[\w-+]+)*@(?:[\w-]+\.)+[a-zA-
    Z]{2,7})")
133.                If email.IsMatch(Me.LoginEmail.Text) Then
134.                    Return True
135.                Else
136.                    Return False
137.                End If
138.            End Function
139.
140.            'If the email is not in the right format an error comes up
141.            Private Sub LoginEmail_Validating(sender As Object, e As System.Componen
    tModel.CancelEventArgs) Handles LoginEmail.Validating
142.                If LoginValidateEmail(Me.LoginEmail.Text) = False Then
143.                    Me.EmailErrorProvider.SetError(Me.LoginEmail, "Enter a valid ema
    il!")
144.                    Return
145.                Else
146.                    Me.EmailErrorProvider.SetError(Me.LoginEmail, "")
147.                End If
148.            End Sub
149.
150.            'Check the password length, if it is not between 8-
    15 characters an error comes up
151.            Private Sub LoginPassword_Validating(sender As Object, e As System.Compo
    nentModel.CancelEventArgs) Handles LoginPassword.Validating
152.                If Len(Me.LoginPassword.Text) = 0 Then
153.                    Me.PasswordErrorProvider.SetError(Me.LoginPassword, "Please ente
    r a password!")
154.                    Return
155.                ElseIf Len(Me.LoginPassword.Text) < 8 Then
156.                    Me.PasswordErrorProvider.SetError(Me.LoginPassword, "Password is
     too short!")
157.                    Return
158.                ElseIf Len(Me.LoginPassword.Text) > 15 Then
159.                    Me.PasswordErrorProvider.SetError(Me.LoginPassword, "Password is
     too long!")
160.                    Return
161.                Else
162.                    Me.PasswordErrorProvider.SetError(Me.LoginPassword, "")
```

```
163.                End If
164.            End Sub
165.
166.            'Check if the values entered in the Login Email and Password textbox are
     correct
167.            Public Function LoginDetailsCheck() As Boolean
168.                If Me.EmailErrorProvider.GetError(Me.LoginEmail) = "" And Me.Passwor
    dErrorProvider.GetError(Me.LoginPassword) = "" Then
169.                    Return True
170.                Else
171.                    Return False
172.                End If
173.            End Function
174.
175.            'When the user presses the Login button, it is checked that the email an
    d password pair correspond in the database
176.            Private Sub LoginBtn_Click(sender As Object, e As EventArgs) Handles Log
    inBtn.Click
177.
178.                Dim LoginDone As Boolean = False
179.                Dim Homepage As New Form2
180.
181.                Dim Table_ As String = "UserAccount"
182.                Dim query As String = "SELECT EmailAddress,UserPassword FROM UserAcc
    ount WHERE EmailAddress = '" & Me.LoginEmail.Text & "'"
183.
184.                'Open a connection with the database, and create a table with the em
    ail and password equal to the user input, if it exists
185.                Dim MDBConnString_ As String = "Provider=Microsoft.ACE.OLEDB.12.0;Da
    ta Source=ProjectDatabase.accdb;Persist Security Info=True"
186.                Dim ds As New DataSet
187.                Dim cnn As OleDbConnection = New OleDbConnection(MDBConnString_)
188.                cnn.Open()
189.                Dim cmd As New OleDbCommand(query, cnn)
190.                Dim da As New OleDbDataAdapter(cmd)
191.                da.Fill(ds, Table_)
192.                cnn.Close()
193.
194.                If ds.Tables(0).Rows.Count = 0 Then  'If there are no rows in the ta
    ble, that means the email does not exists in the database
195.                    MsgBox("Invalid Email!")
196.                Else
197.                    If Convert.ToString(ds.Tables(0).Rows(0)("UserPassword")) = Me.L
    oginPassword.Text Then 'If the password inserted by the user matches the one in the
     table, then the user can login
198.                        MsgBox("Successful login!")
199.                        LoginDone = True
200.                    Else
201.                        MsgBox("Wrong password!") 'If the password does not match, t
    he the user has to login again
202.                        LoginPassword.Text = ""
203.                    End If
204.                End If
205.
206.                'When the login is done, the next page is shown
207.                If LoginDone = True Then
208.                    Me.Hide()
209.                    Homepage.Show()
210.                End If
211.            End Sub
212.        End Class
```

Homepage

Here the user can see all the item listed. Search for items using the search bar, filter/order the products by using the buttons at the top. From here the user can view an item in more detail, by clicking the button at the end of each row, or go to the seller section, by clicking the 'sell' button at the top right corner.




```
1.  Imports System.Data.OleDb
2.  Public Class Form2
3.
4.      'When the user clicks the SELL button, it goes to the seller section
5.      Private Sub SellBtn_Click(sender As Object, e As EventArgs) Handles SellBtn.Cli
    ck
6.          Dim SellerListings As New Form3
7.
8.          Me.Hide()
9.          SellerListings.Show()
10.
11.     End Sub
12.
13.     'Reads all the item listed in the database and shows them in a datagridview or
    "table"
14.     Private Sub ShowAllBtn_Click(sender As Object, e As EventArgs) Handles ShowAllB
    tn.Click
15.
16.         Dim columnButton As New DataGridViewButtonColumn
17.
18.         'add VIEW button at the end of each row
19.         columnButton.Text = "VIEW"
20.         columnButton.UseColumnTextForButtonValue = True
21.         columnButton.FlatStyle = FlatStyle.Flat
22.         columnButton.DefaultCellStyle.BackColor = Color.FromArgb(99, 193, 255)
23.
24.
25.         Dim Table_ As String = "ItemListed"
```

```
26.         Dim query As String = "SELECT ItemInfo.ItemID, ItemName, ItemPrice, ItemDes
    cription, ItemCategory, ItemSold, ItemRating, ItemListed.EmailAddress FROM ItemInfo
     INNER JOIN ItemListed ON ItemInfo.ItemID = ItemListed.ItemID;"
27.
28.         'Open connection with database and fill in a table with all the item listed
    
29.         Dim MDBConnString_ As String = "Provider=Microsoft.ACE.OLEDB.12.0;Data Sour
    ce=ProjectDatabase.accdb;Persist Security Info=True"
30.         Dim ds As New DataSet
31.         Dim cnn As OleDbConnection = New OleDbConnection(MDBConnString_)
32.         cnn.Open()
33.         Dim cmd As New OleDbCommand(query, cnn)
34.         Dim da As New OleDbDataAdapter(cmd)
35.         da.Fill(ds, Table_)
36.         cnn.Close()
37.
38.         'Assign header name for each column in the table
39.         DataGridView2.Rows.Clear()
40.         DataGridView2.ColumnCount = 8
41.         DataGridView2.Columns(0).Name = "ID"
42.         DataGridView2.Columns(1).Name = "Name"
43.         DataGridView2.Columns(2).Name = "Price"
44.         DataGridView2.Columns(3).Name = "Description"
45.         DataGridView2.Columns(4).Name = "Category"
46.         DataGridView2.Columns(5).Name = "Sold"
47.         DataGridView2.Columns(6).Name = "Rating"
48.         DataGridView2.Columns(7).Name = "Listed By"
49.
50.         'Fill values in the datagridview with the values in the table
51.         Dim rowText As String()
52.         Dim t1 As DataTable = ds.Tables(Table_)
53.         Dim row As DataRow
54.         For Each row In t1.Rows
55.             rowText = New String() {row(0), row(1), row(2), row(3), row(4), row(5),
     row(6), row(7)}
56.             DataGridView2.Rows.Add(rowText)
57.         Next
58.
59.         DataGridView2.Columns.Add(columnButton)
60.
61.         'Make the table not sortable from the column headers
62.         For Each DataGridView2Columns In DataGridView2.Columns
63.             DataGridView2Columns.SortMode = DataGridViewColumnSortMode.NotSortable
    
64.         Next
65.     End Sub
66.
67.     'When the user presses the Price button, all the item listed are filtered by pr
    ice lower to higher
68.     Private Sub PriceBtn_Click(sender As Object, e As EventArgs) Handles PriceBtn.C
    lick
69.
70.         Dim columnButton As New DataGridViewButtonColumn
71.
72.         columnButton.Text = "VIEW"
73.         columnButton.UseColumnTextForButtonValue = True
74.         columnButton.FlatStyle = FlatStyle.Flat
75.         columnButton.DefaultCellStyle.BackColor = Color.FromArgb(99, 193, 255)
76.
77.         Dim Table_ As String = "ItemListed"
78.         Dim query As String = "SELECT ItemInfo.ItemID, ItemName, ItemPrice, ItemDes
    cription, ItemCategory, ItemSold, ItemRating, ItemListed.EmailAddress FROM ItemInfo
     INNER JOIN ItemListed ON ItemInfo.ItemID = ItemListed.ItemID ORDER BY ItemPrice AS
    C;"
79.
```

```
80.         Dim MDBConnString_ As String = "Provider=Microsoft.ACE.OLEDB.12.0;Data Sour
    ce=ProjectDatabase.accdb;Persist Security Info=True"
81.         Dim ds As New DataSet
82.         Dim cnn As OleDbConnection = New OleDbConnection(MDBConnString_)
83.         cnn.Open()
84.         Dim cmd As New OleDbCommand(query, cnn)
85.         Dim da As New OleDbDataAdapter(cmd)
86.         da.Fill(ds, Table_)
87.         cnn.Close()
88.
89.         DataGridView2.Rows.Clear()
90.         DataGridView2.ColumnCount = 8
91.         DataGridView2.Columns(0).Name = "ID"
92.         DataGridView2.Columns(1).Name = "Name"
93.         DataGridView2.Columns(2).Name = "Price"
94.         DataGridView2.Columns(3).Name = "Description"
95.         DataGridView2.Columns(4).Name = "Category"
96.         DataGridView2.Columns(5).Name = "Sold"
97.         DataGridView2.Columns(6).Name = "Rating"
98.         DataGridView2.Columns(7).Name = "Listed By"
99.
100.                Dim rowText As String()
101.                Dim t1 As DataTable = ds.Tables(Table_)
102.                Dim row As DataRow
103.                For Each row In t1.Rows
104.                    rowText = New String() {row(0), row(1), row(2), row(3), row(4),
    row(5), row(6), row(7)}
105.                    DataGridView2.Rows.Add(rowText)
106.                Next
107.
108.                DataGridView2.Columns.Add(columnButton)
109.
110.                For Each DataGridView2Columns In DataGridView2.Columns
111.                    DataGridView2Columns.SortMode = DataGridViewColumnSortMode.NotSo
    rtable
112.                Next
113.            End Sub
114.
115.            'When the user presses the Name button, all the item listed are filtered
     by name alphabetically
116.            Private Sub NameBtn_Click(sender As Object, e As EventArgs) Handles Name
    Btn.Click
117.
118.                Dim columnButton As New DataGridViewButtonColumn
119.
120.                columnButton.Text = "VIEW"
121.                columnButton.UseColumnTextForButtonValue = True
122.                columnButton.FlatStyle = FlatStyle.Flat
123.                columnButton.DefaultCellStyle.BackColor = Color.FromArgb(99, 193, 25
    5)
124.
125.                Dim Table_ As String = "ItemListed"
126.                Dim query As String = "SELECT ItemInfo.ItemID, ItemName, ItemPrice,
    ItemDescription, ItemCategory, ItemSold, ItemRating, ItemListed.EmailAddress FROM I
    temInfo INNER JOIN ItemListed ON ItemInfo.ItemID = ItemListed.ItemID ORDER BY ItemN
    ame;"
127.
128.                Dim MDBConnString_ As String = "Provider=Microsoft.ACE.OLEDB.12.0;Da
    ta Source=ProjectDatabase.accdb;Persist Security Info=True"
129.                Dim ds As New DataSet
130.                Dim cnn As OleDbConnection = New OleDbConnection(MDBConnString_)
131.                cnn.Open()
132.                Dim cmd As New OleDbCommand(query, cnn)
133.                Dim da As New OleDbDataAdapter(cmd)
134.                da.Fill(ds, Table_)
135.                cnn.Close()
```

```
136.
137.                 DataGridView2.Rows.Clear()
138.                 DataGridView2.ColumnCount = 8
139.                 DataGridView2.Columns(0).Name = "ID"
140.                 DataGridView2.Columns(1).Name = "Name"
141.                 DataGridView2.Columns(2).Name = "Price"
142.                 DataGridView2.Columns(3).Name = "Description"
143.                 DataGridView2.Columns(4).Name = "Category"
144.                 DataGridView2.Columns(5).Name = "Sold"
145.                 DataGridView2.Columns(6).Name = "Rating"
146.                 DataGridView2.Columns(7).Name = "Listed By"
147.
148.                 Dim rowText As String()
149.                 Dim t1 As DataTable = ds.Tables(Table_)
150.                 Dim row As DataRow
151.                 For Each row In t1.Rows
152.                     rowText = New String() {row(0), row(1), row(2), row(3), row(4),
    row(5), row(6), row(7)}
153.                     DataGridView2.Rows.Add(rowText)
154.                 Next
155.
156.                 DataGridView2.Columns.Add(columnButton)
157.
158.                 For Each DataGridView2Columns In DataGridView2.Columns
159.                     DataGridView2Columns.SortMode = DataGridViewColumnSortMode.NotSo
    rtable
160.                 Next
161.             End Sub
162.
163.             'When the user presses the Rating button, all the item listed are filter
    ed by rating higher to lower
164.             Private Sub RatingBtn_Click(sender As Object, e As EventArgs) Handles Ra
    tingBtn.Click
165.
166.                 Dim columnButton As New DataGridViewButtonColumn
167.
168.                 columnButton.Text = "VIEW"
169.                 columnButton.UseColumnTextForButtonValue = True
170.                 columnButton.FlatStyle = FlatStyle.Flat
171.                 columnButton.DefaultCellStyle.BackColor = Color.FromArgb(99, 193, 25
    5)
172.
173.                 Dim Table_ As String = "ItemListed"
174.                 Dim query As String = "SELECT ItemInfo.ItemID, ItemName, ItemPrice,
    ItemDescription, ItemCategory, ItemSold, ItemRating, ItemListed.EmailAddress FROM I
    temInfo INNER JOIN ItemListed ON ItemInfo.ItemID = ItemListed.ItemID ORDER BY ItemR
    ating DESC;"
175.
176.                 Dim MDBConnString_ As String = "Provider=Microsoft.ACE.OLEDB.12.0;Da
    ta Source=ProjectDatabase.accdb;Persist Security Info=True"
177.                 Dim ds As New DataSet
178.                 Dim cnn As OleDbConnection = New OleDbConnection(MDBConnString_)
179.                 cnn.Open()
180.                 Dim cmd As New OleDbCommand(query, cnn)
181.                 Dim da As New OleDbDataAdapter(cmd)
182.                 da.Fill(ds, Table_)
183.                 cnn.Close()
184.
185.                 DataGridView2.Rows.Clear()
186.                 DataGridView2.ColumnCount = 8
187.                 DataGridView2.Columns(0).Name = "ID"
188.                 DataGridView2.Columns(1).Name = "Name"
189.                 DataGridView2.Columns(2).Name = "Price"
190.                 DataGridView2.Columns(3).Name = "Description"
191.                 DataGridView2.Columns(4).Name = "Category"
192.                 DataGridView2.Columns(5).Name = "Sold"
```

```
193.                DataGridView2.Columns(6).Name = "Rating"
194.                DataGridView2.Columns(7).Name = "Listed By"
195.
196.                Dim rowText As String()
197.                Dim t1 As DataTable = ds.Tables(Table_)
198.                Dim row As DataRow
199.                For Each row In t1.Rows
200.                    rowText = New String() {row(0), row(1), row(2), row(3), row(4),
    row(5), row(6), row(7)}
201.                    DataGridView2.Rows.Add(rowText)
202.                Next
203.
204.                DataGridView2.Columns.Add(columnButton)
205.
206.                For Each DataGridView2Columns In DataGridView2.Columns
207.                    DataGridView2Columns.SortMode = DataGridViewColumnSortMode.NotSo
    rtable
208.                Next
209.            End Sub
210.
211.            'When the user presses the Phones button, only the items in the Phones c
    ategory are shown
212.            Private Sub PhonesBtn_Click(sender As Object, e As EventArgs) Handles Ph
    onesBtn.Click
213.
214.                Dim columnButton As New DataGridViewButtonColumn
215.
216.                columnButton.Text = "VIEW"
217.                columnButton.UseColumnTextForButtonValue = True
218.                columnButton.FlatStyle = FlatStyle.Flat
219.                columnButton.DefaultCellStyle.BackColor = Color.FromArgb(99, 193, 25
    5)
220.
221.                Dim Table_ As String = "ItemListed"
222.                Dim query As String = "SELECT ItemInfo.ItemID, ItemName, ItemPrice,
    ItemDescription, ItemCategory, ItemSold, ItemRating, ItemListed.EmailAddress FROM I
    temInfo INNER JOIN ItemListed ON ItemInfo.ItemID = ItemListed.ItemID WHERE ItemCate
    gory = '" & "Phones" & "'"
223.
224.                Dim MDBConnString_ As String = "Provider=Microsoft.ACE.OLEDB.12.0;Da
    ta Source=ProjectDatabase.accdb;Persist Security Info=True"
225.                Dim ds As New DataSet
226.                Dim cnn As OleDbConnection = New OleDbConnection(MDBConnString_)
227.                cnn.Open()
228.                Dim cmd As New OleDbCommand(query, cnn)
229.                Dim da As New OleDbDataAdapter(cmd)
230.                da.Fill(ds, Table_)
231.                cnn.Close()
232.
233.                DataGridView2.Rows.Clear()
234.                DataGridView2.ColumnCount = 8
235.                DataGridView2.Columns(0).Name = "ID"
236.                DataGridView2.Columns(1).Name = "Name"
237.                DataGridView2.Columns(2).Name = "Price"
238.                DataGridView2.Columns(3).Name = "Description"
239.                DataGridView2.Columns(4).Name = "Category"
240.                DataGridView2.Columns(5).Name = "Sold"
241.                DataGridView2.Columns(6).Name = "Rating"
242.                DataGridView2.Columns(7).Name = "Listed By"
243.
244.                Dim rowText As String()
245.                Dim t1 As DataTable = ds.Tables(Table_)
246.                Dim row As DataRow
247.                For Each row In t1.Rows
248.                    rowText = New String() {row(0), row(1), row(2), row(3), row(4),
    row(5), row(6), row(7)}
```

```
249.                 DataGridView2.Rows.Add(rowText)
250.             Next
251.
252.             DataGridView2.Columns.Add(columnButton)
253.
254.             For Each DataGridView2Columns In DataGridView2.Columns
255.                 DataGridView2Columns.SortMode = DataGridViewColumnSortMode.NotSo
    rtable
256.             Next
257.         End Sub
258.
259.         'When the user presses the Laptops button, only the items in the Laptops
     category are shown
260.         Private Sub LaptopsBtn_Click(sender As Object, e As EventArgs) Handles L
    aptopsBtn.Click
261.
262.             Dim columnButton As New DataGridViewButtonColumn
263.
264.             columnButton.Text = "VIEW"
265.             columnButton.UseColumnTextForButtonValue = True
266.             columnButton.FlatStyle = FlatStyle.Flat
267.             columnButton.DefaultCellStyle.BackColor = Color.FromArgb(99, 193, 25
    5)
268.
269.             Dim Table_ As String = "ItemListed"
270.             Dim query As String = "SELECT ItemInfo.ItemID, ItemName, ItemPrice,
    ItemDescription, ItemCategory, ItemSold, ItemRating, ItemListed.EmailAddress FROM I
    temInfo INNER JOIN ItemListed ON ItemInfo.ItemID = ItemListed.ItemID WHERE ItemCate
    gory = '" & "Laptops" & "'"
271.
272.             Dim MDBConnString_ As String = "Provider=Microsoft.ACE.OLEDB.12.0;Da
    ta Source=ProjectDatabase.accdb;Persist Security Info=True"
273.             Dim ds As New DataSet
274.             Dim cnn As OleDbConnection = New OleDbConnection(MDBConnString_)
275.             cnn.Open()
276.             Dim cmd As New OleDbCommand(query, cnn)
277.             Dim da As New OleDbDataAdapter(cmd)
278.             da.Fill(ds, Table_)
279.             cnn.Close()
280.
281.             DataGridView2.Rows.Clear()
282.             DataGridView2.ColumnCount = 8
283.             DataGridView2.Columns(0).Name = "ID"
284.             DataGridView2.Columns(1).Name = "Name"
285.             DataGridView2.Columns(2).Name = "Price"
286.             DataGridView2.Columns(3).Name = "Description"
287.             DataGridView2.Columns(4).Name = "Category"
288.             DataGridView2.Columns(5).Name = "Sold"
289.             DataGridView2.Columns(6).Name = "Rating"
290.             DataGridView2.Columns(7).Name = "Listed By"
291.
292.             Dim rowText As String()
293.             Dim t1 As DataTable = ds.Tables(Table_)
294.             Dim row As DataRow
295.             For Each row In t1.Rows
296.                 rowText = New String() {row(0), row(1), row(2), row(3), row(4),
    row(5), row(6), row(7)}
297.                 DataGridView2.Rows.Add(rowText)
298.             Next
299.
300.             DataGridView2.Columns.Add(columnButton)
301.
302.             For Each DataGridView2Columns In DataGridView2.Columns
303.                 DataGridView2Columns.SortMode = DataGridViewColumnSortMode.NotSo
    rtable
304.             Next
```

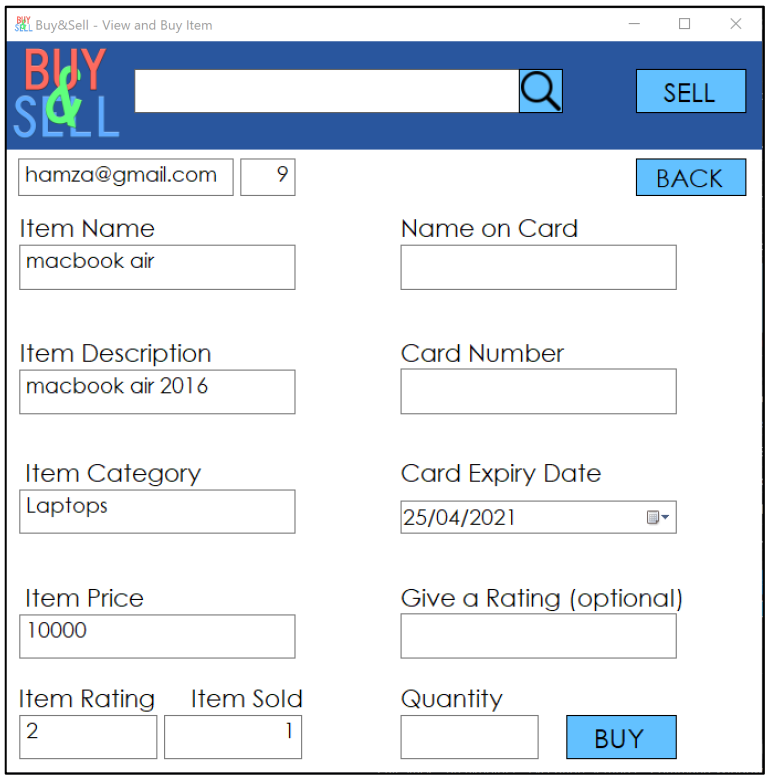
```
305.            End Sub
306.
307.            'When the user presses the TVs button, only the items in the TVs categor
    y are shown
308.            Private Sub TvBtn_Click(sender As Object, e As EventArgs) Handles TvBtn.
    Click
309.
310.                Dim columnButton As New DataGridViewButtonColumn
311.
312.                columnButton.Text = "VIEW"
313.                columnButton.UseColumnTextForButtonValue = True
314.                columnButton.FlatStyle = FlatStyle.Flat
315.                columnButton.DefaultCellStyle.BackColor = Color.FromArgb(99, 193, 25
    5)
316.
317.                Dim Table_ As String = "ItemListed"
318.                Dim query As String = "SELECT ItemInfo.ItemID, ItemName, ItemPrice,
    ItemDescription, ItemCategory, ItemSold, ItemRating, ItemListed.EmailAddress FROM I
    temInfo INNER JOIN ItemListed ON ItemInfo.ItemID = ItemListed.ItemID WHERE ItemCate
    gory = '" & "TVs" & "'"
319.
320.                Dim MDBConnString_ As String = "Provider=Microsoft.ACE.OLEDB.12.0;Da
    ta Source=ProjectDatabase.accdb;Persist Security Info=True"
321.                Dim ds As New DataSet
322.                Dim cnn As OleDbConnection = New OleDbConnection(MDBConnString_)
323.                cnn.Open()
324.                Dim cmd As New OleDbCommand(query, cnn)
325.                Dim da As New OleDbDataAdapter(cmd)
326.                da.Fill(ds, Table_)
327.                cnn.Close()
328.
329.                DataGridView2.Rows.Clear()
330.                DataGridView2.ColumnCount = 8
331.                DataGridView2.Columns(0).Name = "ID"
332.                DataGridView2.Columns(1).Name = "Name"
333.                DataGridView2.Columns(2).Name = "Price"
334.                DataGridView2.Columns(3).Name = "Description"
335.                DataGridView2.Columns(4).Name = "Category"
336.                DataGridView2.Columns(5).Name = "Sold"
337.                DataGridView2.Columns(6).Name = "Rating"
338.                DataGridView2.Columns(7).Name = "Listed By"
339.
340.                Dim rowText As String()
341.                Dim t1 As DataTable = ds.Tables(Table_)
342.                Dim row As DataRow
343.                For Each row In t1.Rows
344.                    rowText = New String() {row(0), row(1), row(2), row(3), row(4),
    row(5), row(6), row(7)}
345.                    DataGridView2.Rows.Add(rowText)
346.                Next
347.
348.                DataGridView2.Columns.Add(columnButton)
349.
350.                For Each DataGridView2Columns In DataGridView2.Columns
351.                    DataGridView2Columns.SortMode = DataGridViewColumnSortMode.NotSo
    rtable
352.                Next
353.            End Sub
354.
355.            'When the user presses the Search button, the items that have the 'user
    search' in their name in any position are shown
356.            Private Sub SearchBtn_Click(sender As Object, e As EventArgs) Handles Se
    archBtn.Click
357.
358.                Dim columnButton As New DataGridViewButtonColumn
359.
```

```
360.                columnButton.Text = "VIEW"
361.                columnButton.UseColumnTextForButtonValue = True
362.                columnButton.FlatStyle = FlatStyle.Flat
363.                columnButton.DefaultCellStyle.BackColor = Color.FromArgb(99, 193, 25
    5)
364.
365.                Dim Table_ As String = "ItemListed"
366.                Dim query As String = "SELECT ItemInfo.ItemID, ItemName, ItemPrice,
    ItemDescription, ItemCategory, ItemSold, ItemRating, ItemListed.EmailAddress FROM I
    temInfo INNER JOIN ItemListed ON ItemInfo.ItemID = ItemListed.ItemID WHERE ItemName
     LIKE '%" & UserSearch.Text & "%'"
367.
368.                Dim MDBConnString_ As String = "Provider=Microsoft.ACE.OLEDB.12.0;Da
    ta Source=ProjectDatabase.accdb;Persist Security Info=True"
369.                Dim ds As New DataSet
370.                Dim cnn As OleDbConnection = New OleDbConnection(MDBConnString_)
371.                cnn.Open()
372.                Dim cmd As New OleDbCommand(query, cnn)
373.                Dim da As New OleDbDataAdapter(cmd)
374.                da.Fill(ds, Table_)
375.                cnn.Close()
376.
377.                DataGridView2.Rows.Clear()
378.                DataGridView2.ColumnCount = 8
379.                DataGridView2.Columns(0).Name = "ID"
380.                DataGridView2.Columns(1).Name = "Name"
381.                DataGridView2.Columns(2).Name = "Price"
382.                DataGridView2.Columns(3).Name = "Description"
383.                DataGridView2.Columns(4).Name = "Category"
384.                DataGridView2.Columns(5).Name = "Sold"
385.                DataGridView2.Columns(6).Name = "Rating"
386.                DataGridView2.Columns(7).Name = "Listed By"
387.
388.                Dim rowText As String()
389.                Dim t1 As DataTable = ds.Tables(Table_)
390.                Dim row As DataRow
391.                For Each row In t1.Rows
392.                    rowText = New String() {row(0), row(1), row(2), row(3), row(4),
    row(5), row(6), row(7)}
393.                    DataGridView2.Rows.Add(rowText)
394.                Next
395.
396.                DataGridView2.Columns.Add(columnButton)
397.
398.                For Each DataGridView2Columns In DataGridView2.Columns
399.                    DataGridView2Columns.SortMode = DataGridViewColumnSortMode.NotSo
    rtable
400.                Next
401.            End Sub
402.
403.            'When the user clicks the VIEW button at the end of a row, it shows anot
    her form
404.            Private Sub DataGridView2_CellClick(sender As Object, e As DataGridViewC
    ellEventArgs) Handles DataGridView2.CellClick
405.                If DataGridView2.Columns(e.ColumnIndex).Index = 8 Then  'Check that
    the user has clicked the VIEW button
406.                    Dim ItemView As New Form6
407.
408.                    Dim index As Integer
409.                    index = e.RowIndex
410.                    Dim SelectedRow As New DataGridViewRow
411.                    SelectedRow = DataGridView2.Rows(index)
412.
413.                    'Pass the values of the item to the next form and assign them to
     a textbox not editable, and shows them on screens
```

```
414.                     ItemView.ItemIDTextBox.Text = SelectedRow.Cells(0).Value.ToStrin
    g
415.                     ItemView.ItemNameTextBox.Text = SelectedRow.Cells(1).Value.ToStr
    ing
416.                     ItemView.ItemPriceTextBox.Text = SelectedRow.Cells(2).Value.ToSt
    ring
417.                     ItemView.ItemDescriptionTextBox.Text = SelectedRow.Cells(3).Valu
    e.ToString
418.                     ItemView.ItemCategoryTextBox.Text = SelectedRow.Cells(4).Value.T
    oString
419.                     ItemView.ItemSoldTextBox.Text = SelectedRow.Cells(5).Value.ToStr
    ing
420.                     ItemView.ItemRatingTextBox.Text = SelectedRow.Cells(6).Value.ToS
    tring
421.                     ItemView.ListedByTextBox.Text = SelectedRow.Cells(7).Value.ToStr
    ing
422.
423.                     ItemView.CardExpiryDatePicker.Value = Date.Today  'Assign date a
    s the date today
424.
425.                     Me.Hide()
426.                     ItemView.Show()
427.
428.                 End If
429.             End Sub
430.         End Class
```

View-Buy Item

In this form the user can buy the item he would like. He needs to enter valid details, and then a payment is made, and an order is created for the seller. From here, the user can go back to the homepage, by using the 'back' button, or go to the seller section by using the 'sell' button.



```
1.  Public Class Form6
2.      'When the user clicks the Back button, it goes to the Homepage
3.      Private Sub BackBtn_Click(sender As Object, e As EventArgs) Handles BackBtn.Cli
    ck
4.          Dim Homepage As New Form2
5.
6.          Me.Hide()
7.          Homepage.Show()
8.      End Sub
9.
10.     'When the user clicks the Sell button, it goes to the seller section
11.     Private Sub SellBtn_Click(sender As Object, e As EventArgs) Handles SellBtn.Cli
    ck
12.         Dim SellerListings As New Form3
13.
14.         Me.Hide()
15.         SellerListings.Show()
16.     End Sub
17.
18.     'When the user clicks the Search button, an error comes up as there is nothing
    to search in this page
19.     Private Sub SearchBtn_Click(sender As Object, e As EventArgs) Handles SearchBtn
    .Click
20.         MsgBox("Can't search on this page!")
21.     End Sub
22.
23.     'Allows the user to only enter characters in the Name textbox
24.     Private Sub CardNameTxtBox_KeyPress(sender As Object, e As KeyPressEventArgs) H
    andles CardNameTxtBox.KeyPress
25.         If Char.IsNumber(e.KeyChar) = True Or Char.IsPunctuation(e.KeyChar) = True
    Or Char.IsSymbol(e.KeyChar) = True Then
26.             e.Handled = True
27.         End If
28.     End Sub
29.
30.     'Allows the user to only enter numbers in the Card Number textbox
31.     Private Sub CardNumberTxtBox_KeyPress(sender As Object, e As KeyPressEventArgs)
     Handles CardNumberTxtBox.KeyPress
32.         If Char.IsLetter(e.KeyChar) = True Or Char.IsSymbol(e.KeyChar) = True Or Ch
    ar.IsWhiteSpace(e.KeyChar) = True Or Char.IsPunctuation(e.KeyChar) = True Then
33.             e.Handled = True
34.         End If
35.     End Sub
36.
37.     'Allows the user to only enter numbers in the Rating textbox
38.     Private Sub UserRatingTxtBox_KeyPress(sender As Object, e As KeyPressEventArgs)
     Handles UserRatingTxtBox.KeyPress
39.         If Char.IsLetter(e.KeyChar) = True Or Char.IsSymbol(e.KeyChar) = True Or Ch
    ar.IsWhiteSpace(e.KeyChar) = True Or Char.IsPunctuation(e.KeyChar) = True Then
40.             e.Handled = True
41.         End If
42.     End Sub
43.
44. 'Allows the user to only enter numbers in the Quantity textbox
45.     Private Sub QuantityTxtBox_KeyPress(sender As Object, e As KeyPressEventArgs) H
    andles QuantityTxtBox.KeyPress
46.         If Char.IsLetter(e.KeyChar) = True Or Char.IsSymbol(e.KeyChar) = True Or Ch
    ar.IsWhiteSpace(e.KeyChar) = True Or Char.IsPunctuation(e.KeyChar) = True Then
47.             e.Handled = True
48.         End If
49.     End Sub
50.
51.     'Checks if there is a value in the Name textbox, if there is not then an error
    comes up
52.     Private Sub CardNameTxtBox_Validating(sender As Object, e As System.ComponentMo
    del.CancelEventArgs) Handles CardNameTxtBox.Validating
```

```
53.         If Len(Me.CardNameTxtBox.Text) = 0 Then
54.             Me.NameErrorProvider.SetError(Me.CardNameTxtBox, "Please enter a name!"
    )
55.             Return
56.         Else
57.             Me.NameErrorProvider.SetError(Me.CardNameTxtBox, "")
58.         End If
59.     End Sub
60.
61.     'Checks if there is a value in the Card Number textbox and it is 16 digits long
    , if it is not then an error comes up
62.     Private Sub CardNumberTxtBox_Validating(sender As Object, e As System.Component
    Model.CancelEventArgs) Handles CardNumberTxtBox.Validating
63.         If Len(Me.CardNumberTxtBox.Text) = 0 Then
64.             Me.CardNumberErrorProvider.SetError(Me.CardNumberTxtBox, "Please enter
    a valid number!")
65.             Return
66.         ElseIf Len(Me.CardNumberTxtBox.Text) <> 16 Then
67.             Me.CardNumberErrorProvider.SetError(Me.CardNumberTxtBox, "Number must b
    e 16 digits long!")
68.             Return
69.         Else
70.             Me.CardNumberErrorProvider.SetError(Me.CardNumberTxtBox, "")
71.         End If
72.     End Sub
73.
74.     'Checks if the rating input by the user is between 0 and 5, if it is not then a
    n error comes up
75.     Private Sub UserRatingTxtBox_Validating(sender As Object, e As System.Component
    Model.CancelEventArgs) Handles UserRatingTxtBox.Validating
76.         If Len(Me.UserRatingTxtBox.Text) = 0 Then  'As the rating is optional, not
    having a value in the rating textbox is not considered an error
77.             Me.RatingErrorProvider.SetError(Me.UserRatingTxtBox, "")
78.             Return
79.         ElseIf Me.UserRatingTxtBox.Text > 5 Or Me.UserRatingTxtBox.Text < 0 Then
80.             Me.RatingErrorProvider.SetError(Me.UserRatingTxtBox, "Value must be bet
    ween 0 and 5!")
81.             Return
82.         Else
83.             Me.RatingErrorProvider.SetError(Me.UserRatingTxtBox, "")
84.         End If
85.     End Sub
86.
87.     'Checks if the date input by the user is after or equal to the current date, if
     it is not then an error comes up
88.     Private Sub CardExpiryDatePicker_Validating(sender As Object, e As System.Compo
    nentModel.CancelEventArgs) Handles CardExpiryDatePicker.Validating
89.         If Me.CardExpiryDatePicker.Value < Date.Today Then
90.             Me.DateErrorProvider.SetError(Me.CardExpiryDatePicker, "Please enter a
    valid date!")
91.             Return
92.         Else
93.             Me.DateErrorProvider.SetError(Me.CardExpiryDatePicker, "")
94.         End If
95.     End Sub
96.
97. 'Checks if there is a value in the Quantity textbox and it is greater than
    0, if it is not then an error comes up
98.     Private Sub QuantityTxtBox_Validating(sender As Object, e As System.ComponentMo
    del.CancelEventArgs) Handles QuantityTxtBox.Validating
99.         If Len(Me.QuantityTxtBox.Text) = 0 Then
100.                Me.QuantityErrorProvider.SetError(Me.QuantityTxtBox, "Please
    enter a number!")
101.                Return
102.            ElseIf Me.QuantityTxtBox.Text = 0 Then
```

```
103.                     Me.QuantityErrorProvider.SetError(Me.QuantityTxtBox, "Quantity m
    ust be greater than 0.")
104.                     Return
105.                 Else
106.                     Me.QuantityErrorProvider.SetError(Me.QuantityTxtBox, "")
107.                 End If
108.             End Sub
109.
110.             'Checks if all the values entered by the user are valid
111.             Public Function CardDetailsCheck() As Boolean
112.                 If Me.NameErrorProvider.GetError(Me.CardNameTxtBox) = "" And Me.Card
    NumberErrorProvider.GetError(Me.CardNumberTxtBox) = "" And Me.DateErrorProvider.Get
    Error(Me.CardExpiryDatePicker) = "" And Me.QuantityErrorProvider.GetError(Me.Quanti
    tyTxtBox) = "" Then
113.                     Return True
114.                 Else
115.                     Return False
116.                 End If
117.             End Function
118.
119.             'Random number generator that decides whether an item is delivered or no
    t
120.             Public Function ItemDeliveryRandom() As Boolean
121.                 Dim randint As Integer = Int(Rnd())
122.
123.                 If randint = 1 Then  'If the value equals 1 the the item is delivere
    d, otherwise it is not
124.                     Return True
125.                 Else
126.                     Return False
127.                 End If
128.             End Function
129.
130.             'When the user clicks the Buy button, all the details are stored in the
    database and the seller receives a new order
131.             Private Sub BuyBtn_Click(sender As Object, e As EventArgs) Handles BuyBt
    n.Click
132.                 If CardDetailsCheck() = True Then  'Check if all details are valid
133.
134.                     Dim itemsoldupdated As Integer = Int(ItemSoldTextBox.Text) + Int
    (QuantityTxtBox.Text)  'Increment the number of item sold by the quantity bought
135.                     Dim itemratingupdated As Integer
136.                     Dim itemsum As Integer
137.
138.                     If UserRatingTxtBox.Text <> "" Then
139.                         itemsum = Int(ItemRatingTextBox.Text) + Int(UserRatingTxtBox
    .Text)  'If the user has given a rating then the updated rating is equal to the mea
    n of the old and new rating
140.                         itemratingupdated = itemsum \ 2
141.                     Else
142.                         itemratingupdated = ItemRatingTextBox.Text  'If the user has
     not given a rating, then it remains the same
143.                     End If
144.
145.                     Dim con As New OleDb.OleDbConnection
146.                     Dim cmd As New OleDb.OleDbCommand
147.
148.                     con.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data S
    ource=ProjectDatabase.accdb;Persist Security Info=True"
149.
150.                     'Insert the details entered by the user into the UserPaymentInfo
     table in the database
151.                     con.Open()
152.                     Dim Sql As String = "INSERT INTO UserPaymentInfo(CardName,CardNu
    mber,CardExpiry,
```

```
    EmailAdress) VALUES('" & Me.CardNameTxtBox.Text & "', '" & Me.CardNumberTxtBox.Text
     & "', '" & Me.CardExpiryDatePicker.Text & "', '" & Form1.LoginEmail.Text & "' )"
153.                    cmd = New OleDb.OleDbCommand(Sql, con)
154.                    cmd.ExecuteNonQuery()
155.                    con.Close()
156.
157.                    'Update the ItemSold and ItemRating fields in the database of th
    e item just bought
158.                    con.Open()
159.                    Dim Sql2 As String = "UPDATE ItemInfo SET ItemSold = " & itemsol
    dupdated & ", ItemRating = " & itemratingupdated & " WHERE (ItemID = " & Me.ItemIDT
    extBox.Text & ")"
160.                    cmd = New OleDb.OleDbCommand(Sql2, con)
161.                    cmd.ExecuteNonQuery()
162.                    con.Close()
163.
164.                    'Create an order in the database by storing the ItemID of the it
    em just bought and whethter it has been delivered or not
165.                    con.Open()
166.                    Dim Sql3 As String = "INSERT INTO OrderInfo(ItemID,ItemDelivered
    ) VALUES(" & Me.ItemIDTextBox.Text & "," & ItemDeliveryRandom() & ")"
167.                    cmd = New OleDb.OleDbCommand(Sql3, con)
168.                    cmd.ExecuteNonQuery()
169.                    con.Close()
170.
171.                    'Store the order in the "UserOrder table so the seller can see t
    hat he has a new order
172.                    con.Open()
173.                    Dim Sql4 As String = "INSERT INTO UserOrder(EmailAdress,OrderID)
     VALUES('" & Me.ListedByTextBox.Text & "', " & UserOrderID() & ")"
174.                    cmd = New OleDb.OleDbCommand(Sql4, con)
175.                    cmd.ExecuteNonQuery()
176.                    con.Close()
177.
178.                    MsgBox("Succesful Payment!")
179.
180.                    'Reset the value of all the textboxes and date picker
181.                    CardNameTxtBox.Text = ""
182.                    CardNumberTxtBox.Text = ""
183.                    CardExpiryDatePicker.Value = Date.Today
184.                    UserRatingTxtBox.Text = ""
185.                Else
186.                    MsgBox("Enter valid details!")
187.                End If
188.
189.        'Get the value of OrderID of the order just made by the user
190.        Public Function UserOrderID() As Integer
191.
192.            Dim Table_ As String = "OrderInfo"
193.            Dim query As String = "SELECT MAX(OrderID) FROM OrderInfo"
194.
195.            Dim MDBConnString_ As String = "Provider=Microsoft.ACE.OLEDB.12.0;Data S
    ource=ProjectDatabase.accdb;Persist Security Info=True"
196.            Dim ds As New DataSet
197.            Dim cnn As OleDb.OleDbConnection = New OleDb.OleDbConnection(MDBConnStri
    ng_)
198.            cnn.Open()
199.            Dim cmd As New OleDb.OleDbCommand(query, cnn)
200.            Dim da As New OleDb.OleDbDataAdapter(cmd)
201.            da.Fill(ds, Table_)
202.            cnn.Close()
203.
204.            Return ds.Tables(0).Rows(0)("Expr1000")
205.        End Function
206.
207.        End Sub
```

Seller Listings

In this form the user can see all his item listed and list new items as well, by entering the details of the product. From here, the user can go to the other sections, orders and charts, or to the homepage, by clicking the 'home' button.


```
1.  Imports System.Text.RegularExpressions
2.  Imports System.Data.OleDb
3.  Public Class Form3
4.      'When the user presses the Listings button, a messaege comes up, as the user is
     already on the Listings page
5.      Private Sub ListingsBtn_Click(sender As Object, e As EventArgs) Handles Listing
    sBtn.Click
6.          MsgBox("You are already on the listings section!")
7.      End Sub
8.
9.      'When the user presses the Orders button, the Orders page comes up
10.     Private Sub OrdersBtn_Click(sender As Object, e As EventArgs) Handles OrdersBtn
    .Click
11.         Dim SellerOrders As New Form4
12.
13.         Me.Hide()
```

```
14.         SellerOrders.Show()
15.     End Sub
16.
17.     'When the user presses the Charts buttton, the Charts page comes up
18.     Private Sub ChartsBtn_Click(sender As Object, e As EventArgs) Handles ChartsBtn
    .Click
19.         Dim SellerCharts As New Form5
20.
21.         Me.Hide()
22.         SellerCharts.Show()
23.     End Sub
24.
25.     'When the user presses the Home button, the Homepage comes up
26.     Private Sub HomeBtn_Click(sender As Object, e As EventArgs) Handles HomeBtn.Cli
    ck
27.         Dim Homepage As New Form2
28.
29.         Me.Hide()
30.         Homepage.Show()
31.     End Sub
32.
33.     'When the user presses the Search button, an error comes up as it is not possib
    le to search on this page
34.     Private Sub SearchBtn_Click(sender As Object, e As EventArgs) Handles SearchBtn
    .Click
35.         MsgBox("Can't search on this page!")
36.     End Sub
37. 'When the user presses the Show button, all the item listed alongside their info ar
    e shown on screen in the datagrid
38. Private Sub ShowDataGridBtn_Click(sender As Object, e As EventArgs) Handles ShowDat
    aGridBtn.Click
39.
40.     Dim Table_ As String = "ItemListed"
41.     Dim query As String = "SELECT EmailAddress, ItemInfo.ItemID, ItemName, ItemPric
    e, ItemDescription, ItemCategory, ItemSold, ItemRating FROM ItemInfo INNER JOIN Ite
    mListed ON ItemInfo.ItemID = ItemListed.ItemID WHERE EmailAddress = '" & Form1.Logi
    nEmail.Text & "'" 'Replace the query with your one
42.
43.     Dim MDBConnString_ As String = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=P
    rojectDatabase.accdb;Persist Security Info=True"
44.     Dim ds As New DataSet
45.     Dim cnn As OleDbConnection = New OleDbConnection(MDBConnString_)
46.     cnn.Open()
47.     Dim cmd As New OleDbCommand(query, cnn)
48.     Dim da As New OleDbDataAdapter(cmd)
49.     da.Fill(ds, Table_)
50.     cnn.Close()
51.
52.     DataGridView1.Rows.Clear()
53.     DataGridView1.ColumnCount = 8
54.     DataGridView1.Columns(0).Name = "Email"
55.     DataGridView1.Columns(1).Name = "ID"
56.     DataGridView1.Columns(2).Name = "Name"
57.     DataGridView1.Columns(3).Name = "Price"
58.     DataGridView1.Columns(4).Name = "Description"
59.     DataGridView1.Columns(5).Name = "Category"
60.     DataGridView1.Columns(6).Name = "Sold"
61.     DataGridView1.Columns(7).Name = "Rating"
62.
63.     Dim rowText As String()
64.     Dim t1 As DataTable = ds.Tables(Table_)
65.     Dim row As DataRow
66.     For Each row In t1.Rows
67.         rowText = New String() {row(0), row(1), row(2), row(3), row(4), row(5), row
    (6), row(7)}
68.         DataGridView1.Rows.Add(rowText)
```

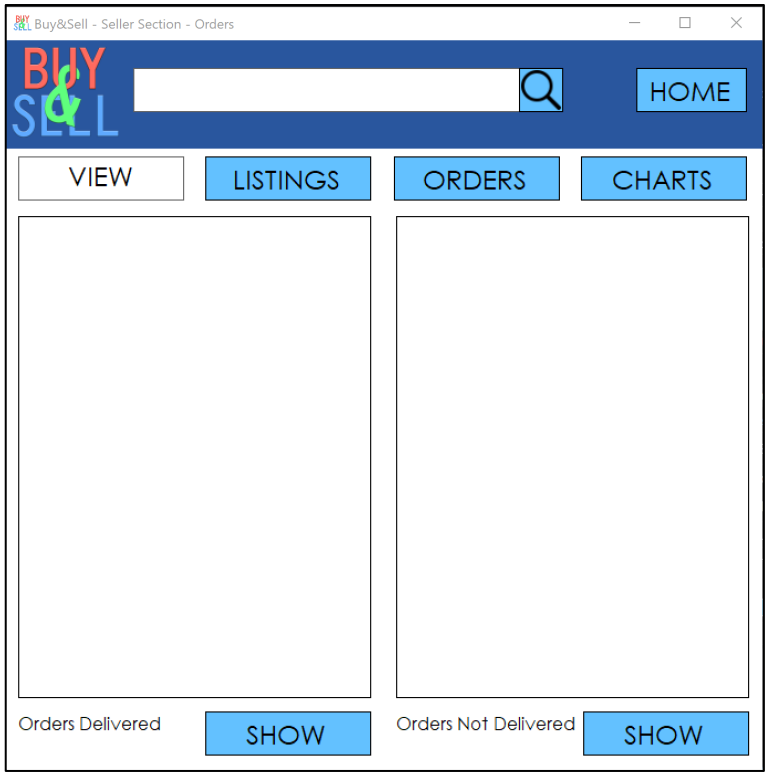
```
69.     Next
70.
71.     For Each DataGridView1Columns In DataGridView1.Columns
72.         DataGridView1Columns.SortMode = DataGridViewColumnSortMode.NotSortable
73.     Next
74. End Sub
75.
76. 'Checks that the there is a value in the Description textbox and it is less than 10
    0 characters, if that is not the case then an error comes up
77. Private Sub ItemDescription_Validating(sender As Object, e As System.ComponentModel
    .CancelEventArgs) Handles ItemDescription.Validating
78.     If Len(ItemDescription.Text) = 0 Then
79.         Me.DescriptionErrorProvider.SetError(Me.ItemDescription, "Please enter a de
    scription! Must be less than 100 characters")
80.         Return
81.     ElseIf Len(ItemDescription.Text) > 100 Then
82.         Me.DescriptionErrorProvider.SetError(Me.ItemDescription, "Description is to
    o long! Must be less than 100 characters")
83.         Return
84.     Else
85.         Me.DescriptionErrorProvider.SetError(Me.ItemDescription, "")
86.     End If
87. End Sub
88.
89. 'Allows the user to only enter numbers in the Price textbox
90. Private Sub ItemPrice_KeyPress(sender As Object, e As KeyPressEventArgs) Handles It
    emPrice.KeyPress
91.     If Char.IsLetter(e.KeyChar) = True Or Char.IsSymbol(e.KeyChar) = True Or Char.I
    sWhiteSpace(e.KeyChar) = True Then
92.         e.Handled = True
93.     End If
94. End Sub
95.
96. 'Check that the input in the Price textbox is in the right format, for example, 1.1
    1, the dot represent the decimal point
97. Public Function PriceFormatValidation() As Boolean
98.     Dim email As New Regex("^[$]?[0-9]*(\.)?[0-9]?[0-9]?$")
99.     If email.IsMatch(Me.ItemPrice.Text) Then
100.                Return True
101.            Else
102.                Return False
103.            End If
104.        End Function
105.        'Checks that there is a value in the Price textbox and it is in the right fo
    rmat, if that is not the case then an error comes up
106.        Private Sub ItemPrice_Validating(sender As Object, e As System.ComponentMode
    l.CancelEventArgs) Handles ItemPrice.Validating
107.            If Len(Me.ItemPrice.Text) = 0 Then
108.                Me.PriceErrorProvider.SetError(Me.ItemPrice, "Please enter a price,
    in the format 0.00")
109.                Return
110.            ElseIf PriceFormatValidation() = False Then
111.                Me.PriceErrorProvider.SetError(Me.ItemPrice, "Please enter a valid p
    rice, in the format 0.00.")
112.            Else
113.                Me.PriceErrorProvider.SetError(Me.ItemPrice, "")
114.            End If
115.        End Sub
116.
117.        'Checks that there is a value in the Name textbox, if that is not the case t
    hen an error comes up
118.        Private Sub ItemName_Validating(sender As Object, e As System.ComponentModel
    .CancelEventArgs) Handles ItemName.Validating
119.            If Len(Me.ItemName.Text) = 0 Then
120.                Me.NameErrorProvider.SetError(Me.ItemName, "Please enter a name!")
121.                Return
```

```
122.            Else
123.                Me.NameErrorProvider.SetError(Me.ItemName, "")
124.            End If
125.        End Sub
126.
127.        'Checks that all the value entered are valid
128.        Public Function ItemInfoCheck() As Boolean
129.            If Me.NameErrorProvider.GetError(Me.ItemName) = "" And Me.DescriptionErr
    orProvider.GetError(Me.ItemDescription) = "" And Me.PriceErrorProvider.GetError(Me.
    ItemPrice) = "" And CategoryListBox.SelectedIndex <> -1 Then
130.                Return True
131.            Else
132.                Return False
133.            End If
134.        End Function
135.
136.        'Insert the values entered in the database
137.        Private Sub ListItemBtn_Click(sender As Object, e As EventArgs) Handles List
    ItemBtn.Click
138.
139.            If ItemInfoCheck() = True Then
140.                Dim con As New OleDb.OleDbConnection
141.
142.                Dim cmd As New OleDb.OleDbCommand
143.
144.                con.ConnectionString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Sourc
    e=ProjectDatabase.accdb;Persist Security Info=True"
145.
146.                con.Open()
147.                Dim Sql As String = "INSERT INTO ItemInfo(ItemName,ItemPrice,ItemDes
    cription,ItemCategory) VALUES('" & Me.ItemName.Text & "','" & Me.ItemPrice.Text & "
    ', '" & Me.ItemDescription.Text & "', '" & Me.CategoryListBox.SelectedItem.ToString
    () & "' )"
148.                cmd = New OleDb.OleDbCommand(Sql, con)
149.                cmd.ExecuteNonQuery()
150.                con.Close()
151.
152.                con.Open()
153.                Dim Sql1 As String = "INSERT INTO ItemListed(EmailAddress,ItemID) VA
    LUES('" & Form1.LoginEmail.Text & "', " & ItemListedID() & ")"
154.                cmd = New OleDb.OleDbCommand(Sql1, con)
155.                cmd.ExecuteNonQuery()
156.                con.Close()
157.
158.                MsgBox("Your item has been listed!")
159.
160.                ItemName.Text = ""
161.                ItemPrice.Text = ""
162.                ItemDescription.Text = ""
163.                CategoryListBox.SelectedIndex = -1
164.            Else
165.                MsgBox("Please enter valid details!")
166.            End If
167.
168.        End Sub
169.
170.        'Get the value of ItemID of the item just listed by the user
171.        Public Function ItemListedID() As Integer
172.
173.            Dim Table_ As String = "ItemInfo"
174.            Dim query As String = "SELECT MAX(ItemID) FROM ItemInfo"
175.
176.            Dim MDBConnString_ As String = "Provider=Microsoft.ACE.OLEDB.12.0;Data S
    ource=ProjectDatabase.accdb;Persist Security Info=True"
177.            Dim ds As New DataSet
```

```
178.            Dim cnn As OleDb.OleDbConnection = New OleDb.OleDbConnection(MDBConnStri
    ng_)
179.            cnn.Open()
180.            Dim cmd As New OleDb.OleDbCommand(query, cnn)
181.            Dim da As New OleDb.OleDbDataAdapter(cmd)
182.            da.Fill(ds, Table_)
183.            cnn.Close()
184.
185.            Return ds.Tables(0).Rows(0)("Expr1000")
186.        End Function
187.        Class
188.
```

Seller Orders

In this form the user can see his delivered orders in the right table and his undelivered orders in the left table. From here the user can go to the other sections, listings or charts, or to the homepage, by using the 'home' button.




```
1.  Imports System.Data.OleDb
2.  Public Class Form4
3.      'When the user presses the Listings button, the Listings page comes up
4.      Private Sub ListingsBtn_Click(sender As Object, e As EventArgs) Handles Listing
    sBtn.Click
5.          Dim SellerListings As New Form3
6.
7.          Me.Hide()
8.          SellerListings.Show()
9.      End Sub
10.
11.     'When the user presses the Orders button, a message comes up, as the user is al
    ready on the Orders page
12.     Private Sub OrdersBtn_Click(sender As Object, e As EventArgs) Handles OrdersBtn
    .Click
13.         MsgBox("You are already on the orders section!")
14.
15.     End Sub
```

```
16.
17.     'When the user presses the Charts button, the Charts page comes up
18.     Private Sub ChartsBtn_Click(sender As Object, e As EventArgs) Handles ChartsBtn
    .Click
19.         Dim SellerCharts As New Form5
20.
21.         Me.Hide()
22.         SellerCharts.Show()
23.     End Sub
24.
25.     'When the user presses the Home button, the Homepage page comes up
26.     Private Sub HomeBtn_Click(sender As Object, e As EventArgs) Handles HomeBtn.Cli
    ck
27.         Dim Homepage As New Form2
28.
29.         Me.Hide()
30.         Homepage.Show()
31.     End Sub
32.
33.     'When the user presses the Search button, an error comes up as it is not possib
    le to search on this page
34.     Private Sub SearchBtn_Click(sender As Object, e As EventArgs) Handles SearchBtn
    .Click
35.         MsgBox("Can't search on this page!")
36.     End Sub
37.
38.     'When the user presses the Show button, all the item listed that have been deli
    vered are shown on screen in the datagrid
39.     Private Sub ShowDataGrid1Btn_Click(sender As Object, e As EventArgs) Handles Sh
    owDataGrid1Btn.Click
40.
41.         Dim Table_ As String = "UserOrder"
42.         Dim query As String = "SELECT EmailAddress, OrderInfo.OrderID, ItemID, Item
    Delivered FROM OrderInfo INNER JOIN UserOrder ON OrderInfo.OrderID = UserOrder.Orde
    rID WHERE ItemDelivered = True AND EmailAddress = '" & Form1.LoginEmail.Text & "'"

43.
44.         Dim MDBConnString_ As String = "Provider=Microsoft.ACE.OLEDB.12.0;Data Sour
    ce=ProjectDatabase.accdb;Persist Security Info=True"
45.         Dim ds As New DataSet
46.         Dim cnn As OleDbConnection = New OleDbConnection(MDBConnString_)
47.         cnn.Open()
48.         Dim cmd As New OleDbCommand(query, cnn)
49.         Dim da As New OleDbDataAdapter(cmd)
50.         da.Fill(ds, Table_)
51.         cnn.Close()
52.
53.         DataGridView1.Rows.Clear()
54.         DataGridView1.ColumnCount = 4
55.         DataGridView1.Columns(0).Name = "Email"
56.         DataGridView1.Columns(1).Name = "OrderID"
57.         DataGridView1.Columns(2).Name = "ItemID"
58.         DataGridView1.Columns(3).Name = "Delivered"
59.
60.         Dim rowText As String()
61.         Dim t1 As DataTable = ds.Tables(Table_)
62.         Dim row As DataRow
63.         For Each row In t1.Rows
64.             rowText = New String() {row(0), row(1), row(2), row(3)}
65.             DataGridView1.Rows.Add(rowText)
66.         Next
67.
68.         For Each DataGridView1Columns In DataGridView1.Columns
69.             DataGridView1Columns.SortMode = DataGridViewColumnSortMode.NotSortable

70.         Next
```

```
71.     End Sub
72.
73.     'When the user presses the Show button, all the item listed that have not been
    delivered are shown on screen in the datagrid
74.     Private Sub ShowDataGrid2Btn_Click(sender As Object, e As EventArgs) Handles Sh
    owDataGrid2Btn.Click
75.
76.         Dim Table_ As String = "UserOrder"
77.         Dim query As String = "SELECT EmailAddress, OrderInfo.OrderID, ItemID, Item
    Delivered FROM OrderInfo INNER JOIN UserOrder ON OrderInfo.OrderID = UserOrder.Orde
    rID WHERE ItemDelivered = False AND EmailAddress = '" & Form1.LoginEmail.Text & "'"

78.
79.         Dim MDBConnString_ As String = "Provider=Microsoft.ACE.OLEDB.12.0;Data Sour
    ce=ProjectDatabase.accdb;Persist Security Info=True"
80.         Dim ds As New DataSet
81.         Dim cnn As OleDbConnection = New OleDbConnection(MDBConnString_)
82.         cnn.Open()
83.         Dim cmd As New OleDbCommand(query, cnn)
84.         Dim da As New OleDbDataAdapter(cmd)
85.         da.Fill(ds, Table_)
86.         cnn.Close()
87.
88.         DataGridView2.Rows.Clear()
89.         DataGridView2.ColumnCount = 4
90.         DataGridView2.Columns(0).Name = "Email"
91.         DataGridView2.Columns(1).Name = "OrderID"
92.         DataGridView2.Columns(2).Name = "ItemID"
93.         DataGridView2.Columns(3).Name = "Delivered"
94.
95.         Dim rowText As String()
96.         Dim t1 As DataTable = ds.Tables(Table_)
97.         Dim row As DataRow
98.         For Each row In t1.Rows
99.             rowText = New String() {row(0), row(1), row(2), row(3)}
100.                DataGridView2.Rows.Add(rowText)
101.            Next
102.
103.            For Each DataGridView2Columns In DataGridView2.Columns
104.                DataGridView2Columns.SortMode = DataGridViewColumnSortMode.NotSo
    rtable
105.            Next
106.        End Sub
107.
108.    End Class
```

Seller Charts

In this form the user can see his most sold items in the left table and his highest rated items in the right table. From here the user can go to the other sections, listings and orders, or to the homepage, by clicking the 'home' button.


```
1.  Imports System.Data.OleDb
2.  Public Class Form5
3.      'When the user presses the Listings button, the Listings page comes up
4.      Private Sub ListingsBtn_Click(sender As Object, e As EventArgs) Handles Listing
    sBtn.Click
5.          Dim SellerListings As New Form3
6.
7.          Me.Hide()
8.          SellerListings.Show()
9.      End Sub
10.
11.     'When the user presses the Orders button, the Orders page comes up
12.     Private Sub OrdersBtn_Click(sender As Object, e As EventArgs) Handles OrdersBtn
    .Click
13.         Dim SellerOrders As New Form4
14.
```

```
15.         Me.Hide()
16.         SellerOrders.Show()
17.     End Sub
18.
19.     'When the user presses the Orders button, a message comes up, as the user is al
    ready on the Orders page
20.     Private Sub ChartsBtn_Click(sender As Object, e As EventArgs) Handles ChartsBtn
    .Click
21.         MsgBox("You are already on the charts section!")
22.
23.     End Sub
24.
25.     'When the user presses the Home button, the Homepage page comes up
26.     Private Sub HomeBtn_Click(sender As Object, e As EventArgs) Handles HomeBtn.Cli
    ck
27.         Dim Homepage As New Form2
28.
29.         Me.Hide()
30.         Homepage.Show()
31.     End Sub
32.
33.     'When the user presses the Search button, an error comes up as it is not possib
    le to search on this page
34.     Private Sub SearchBtn_Click(sender As Object, e As EventArgs) Handles SearchBtn
    .Click
35.         MsgBox("Can't search on this page!")
36.     End Sub
37.
38.     'When the user presses the Show button, all the item listed ordered by most sol
    d are shown on screen in the datagrid
39.     Private Sub ShowDataGrid1Btn_Click(sender As Object, e As EventArgs) Handles Sh
    owDataGrid1Btn.Click
40.
41.         Dim Table_ As String = "ItemListed"
42.         Dim query As String = "SELECT EmailAddress, ItemInfo.ItemID, ItemName, Item
    Price, ItemDescription, ItemCategory, ItemSold, ItemRating FROM ItemInfo INNER JOIN
     ItemListed ON ItemInfo.ItemID = ItemListed.ItemID WHERE EmailAddress = '" & Form1.
    LoginEmail.Text & "' ORDER BY ItemSold DESC;"
43.
44.         Dim MDBConnString_ As String = "Provider=Microsoft.ACE.OLEDB.12.0;Data Sour
    ce=ProjectDatabase.accdb;Persist Security Info=True"
45.         Dim ds As New DataSet
46.         Dim cnn As OleDbConnection = New OleDbConnection(MDBConnString_)
47.         cnn.Open()
48.         Dim cmd As New OleDbCommand(query, cnn)
49.         Dim da As New OleDbDataAdapter(cmd)
50.         da.Fill(ds, Table_)
51.         cnn.Close()
52.
53.         DataGridView1.Rows.Clear()
54.         DataGridView1.ColumnCount = 8
55.         DataGridView1.Columns(0).Name = "Email"
56.         DataGridView1.Columns(1).Name = "ID"
57.         DataGridView1.Columns(2).Name = "Name"
58.         DataGridView1.Columns(3).Name = "Price"
59.         DataGridView1.Columns(4).Name = "Description"
60.         DataGridView1.Columns(5).Name = "Category"
61.         DataGridView1.Columns(6).Name = "Sold"
62.         DataGridView1.Columns(7).Name = "Rating"
63.
64.         Dim rowText As String()
65.         Dim t1 As DataTable = ds.Tables(Table_)
66.         Dim row As DataRow
67.         For Each row In t1.Rows
68.             rowText = New String() {row(0), row(1), row(2), row(3), row(4), row(5),
     row(6), row(7)}
```

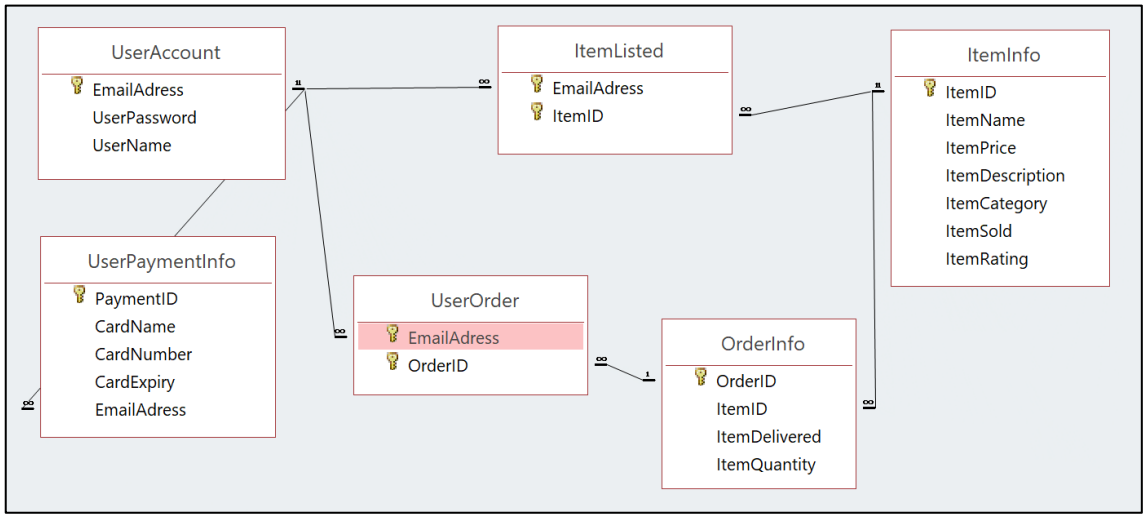
```
69.             DataGridView1.Rows.Add(rowText)
70.         Next
71.
72.         For Each DataGridView1Columns In DataGridView1.Columns
73.             DataGridView1Columns.SortMode = DataGridViewColumnSortMode.NotSortable

74.         Next
75.     End Sub
76.
77.     'When the user presses the Show button, all the item listed ordered by most rat
    ed are shown on screen in the datagrid
78.     Private Sub ShowDataGrid2Btn_Click(sender As Object, e As EventArgs) Handles Sh
    owDataGrid2Btn.Click
79.
80.         Dim Table_ As String = "ItemListed"
81.         Dim query As String = "SELECT EmailAddress, ItemInfo.ItemID, ItemName, Item
    Price, ItemDescription, ItemCategory, ItemSold, ItemRating FROM ItemInfo INNER JOIN
     ItemListed ON ItemInfo.ItemID = ItemListed.ItemID WHERE EmailAddress = '" & Form1.
    LoginEmail.Text & "' ORDER BY ItemRating DESC"
82.
83.         Dim MDBConnString_ As String = "Provider=Microsoft.ACE.OLEDB.12.0;Data Sour
    ce=ProjectDatabase.accdb;Persist Security Info=True"
84.         Dim ds As New DataSet
85.         Dim cnn As OleDbConnection = New OleDbConnection(MDBConnString_)
86.         cnn.Open()
87.         Dim cmd As New OleDbCommand(query, cnn)
88.         Dim da As New OleDbDataAdapter(cmd)
89.         da.Fill(ds, Table_)
90.         cnn.Close()
91.
92.         DataGridView2.Rows.Clear()
93.         DataGridView2.ColumnCount = 8
94.         DataGridView2.Columns(0).Name = "Email"
95.         DataGridView2.Columns(1).Name = "ID"
96.         DataGridView2.Columns(2).Name = "Name"
97.         DataGridView2.Columns(3).Name = "Price"
98.         DataGridView2.Columns(4).Name = "Description"
99.         DataGridView2.Columns(5).Name = "Category"
100.                DataGridView2.Columns(6).Name = "Sold"
101.                DataGridView2.Columns(7).Name = "Rating"
102.
103.                Dim rowText As String()
104.                Dim t1 As DataTable = ds.Tables(Table_)
105.                Dim row As DataRow
106.                For Each row In t1.Rows
107.                    rowText = New String() {row(0), row(1), row(2), row(3), row(4),
    row(5), row(6), row(7)}
108.                    DataGridView2.Rows.Add(rowText)
109.                Next
110.
111.                For Each DataGridView2Columns In DataGridView2.Columns
112.                    DataGridView2Columns.SortMode = DataGridViewColumnSortMode.NotSo
    rtable
113.                Next
114.            End Sub
115.        End Class
```

Database

Here we can see all the relationships between the tables, and what kind of data is accepted by each field of each table.

Database Relationships



UserAccount table

UserAccount

Field Name	Data Type
EmailAdress	Short Text
UserPassword	Short Text
UserName	Short Text

ItemInfo table

ItemInfo

Field Name	Data Type
ItemID	AutoNumber
ItemName	Short Text
ItemPrice	Currency
ItemDescription	Short Text
ItemCategory	Short Text
ItemSold	Number
ItemRating	Number

ItemListed table

ItemListed

Field Name	Data Type
EmailAdress	Short Text
ItemID	Number

OrderInfo table

OrderInfo

Field Name	Data Type
OrderID	AutoNumber
ItemID	Number
ItemDelivered	Yes/No
ItemQuantity	Number

UserOrder table

UserOrder

Field Name	Data Type
EmailAdress	Short Text
OrderID	Number

UserPaymentInfo table

UserPaymentInfo

Field Name	Data Type
PaymentID	AutoNumber
CardName	Short Text
CardNumber	Number
CardExpiry	Date/Time
EmailAdress	Short Text

Techniques/models used

Aggregate SQL Function

Used the MAX function to get the ID of the item just inserted in the database. Also the AVG function has been used to calculate the rating of an item, after the user gives rates an item he purchased.

Pattern Matching
In order to check whether the user has entered a valid email in the right format, the value entered by the user is matched to a Regular Expression. If the input does not match the Regex 'email' then it is not valid. The same method has been used to check if the user has entered a correct price format when entering the price of a listing item.

Simple Mathematical Algorithms
Simple calculations and algorithms are used throughout the project, for example sum, length of a text, average, random number.

Multi-dimensional Arrays
The project uses a lot of tables, which can be considered as two-dimensional arrays. When 'passing' a table from Ms Access to VB it is 'passed' as an array: table (row, column). This is useful as it can be used to check whether a value exists already in the database or not; or get a specific value from the table.

Simple Data-Model in Database
The project uses 6 tables which are linked together. The tables are linked in order to create different types of relationships between them, for example a one-to-many or many-to-many relationship.

TESTING

Test Strategy

I will be testing my project by going through each form of my project, there are in total 6 forms. For each form I will test all the possible action that the user can take and check that everything works. When the user enters an invalid input an error message shows up next to the textbox, and when the user presses a button that doesn't lead to anything an error message comes up in a message box. I will also test that data is written in the database in the right tables and it is read from the right tables, without causing any error.
When testing the first form, login, I will check that all the user inputs are valid before registering/login the user. When the input is not valid an error should show up on screen. I will also check that the values entered by the user are stored in the correct table in the database.
When testing the second form, homepage, I will check that the data is displayed correctly in the DataGrid. I will check that every button does the right thing, for example Price button orders the items by price lower to higher.
When testing the third form, view/buy item, I will check that the right values have been passed to the form and they are not editable by the user. Each input in the textboxes should be valid and an error should show up when they are not. When pressing the Buy button, the user payment info should be stored in the right table and a new order should be made for the seller.
When testing the fourth form, seller listings, I will check that only the item listed by the current user shows on the Data Grid and all the information is available. Each input in the

textboxes should be valid and an error should show up when they are not. When the user lists the item, all the information about the item should be stored in the database.
When testing the fifth form, seller orders, I will check that only the orders of the current user show up and they in the right Data Grid, delivered in the right and not delivered in the left one.
When testing the sixth form, seller charts, I will check that only the item listed by the current user show up an in the right Data Grid, most sold in the right and highest rated in the left.
All tests, where applicable, will be done by using normal, boundary and erroneous data to make sure the program works as expected.

Test Plan

TEST NUMBER	TEST DESCRIPTION	EXPECTED OUTCOME	ACTUAL OUTCOME
LOGIN PAGE			
1a	Check if all the data validation works and an error comes up when the data is not valid	All the validations should take place and all the error messages should show up	Works as expected
1b	Check that the email the user is trying to register does not exist in the database, if it already exists an error message should come up	If the email already exists than the user should not be able to register	Works as expected
1c	Check that the register button works only if all the data inserted is valid, an error message should come up if that is not the case	Only when all the data inserted is correct/valid the button should work	Works as expected
1d	Check that the data is inserted in the right table in the database	The data is inserted in the table UserAccount and the values are in the right columns	Works as expected
1e	Check that the email the user is trying to login with exists in the database and the password matches when the	The login should only happen when the email already exists in the database and the password matches	Works as expected

	user tries to login, if that is not the case then an error message should come up		
HOMEPAGE			
2a	Check that Sell button works	The Seller-Listings page should come up	Works as expected
2b	Check that the Show All button works	All the item listed should show on the Data Grid unsorted	Works as expected
2c	Check that the Search button works	Every item that has the “searched word” in their name should show on the Data Grid	Works as expected
2d	Check that the Price button works	The items in the list should be filtered by price low to high	Works as expected
2e	Check that the Name button works	The items in the list should be filtered by name alphabetically	Works as expected
2f	Check that the Rating button works	The items in the list should be filtered by rating high to low	Works as expected
2g	Check that the Phones button works	Only the items in the Phones category should show	Works as expected
2h	Check that the Laptops button works	Only the items in the Laptops category should show	Works as expected
2i	Check that the TVs button works	Only the items in the TVs category should show	Works as expected
2j	Check that the View Item at the end of each row works and the right values are passed to the next form	A new page should come up where the user can see the details of the item he wanted to see	Works as expected
VIEW/BUY ITEM			
3a	Check that the right values of the item are displayed, and	The details of the item that the user wants to see should be displayed in the	Works as expected

	the user can't edit them	right textboxes and the user should not be able to edit them	
3b	Check that the Sell button works	The Seller-Listings page should come up	Works as expected
3c	Check that the Back button works	The Homepage should come up	Works as expected
3d	Check that the Search button works	An error message should come up	Works as expected
3e	Check if all the data validation works and an error comes up when the data is not valid	All the validations should take place and all the error messages should show up	Works as expected
3f	Check that the Buy button works, and the data is stored correctly in the database	When the user presses the Buy button, the user payment info should be stored in the UserPaymentInfo table. The itemsold and itemrating fields should be updated in the ItemInfo table. A new order should be created, the itemID and whether it was delivered or not should be stored in the OrderInfo table, the seller email and orderID should be stored in the UserOrder table. If the data is not valid then an error message should come up.	Works as expected
SELLER-LISTINGS			
4a	Check that the Home button works	The Homepage should come up	Works as expected
4b	Check that the Search button works	An error message should come up	Works as expected

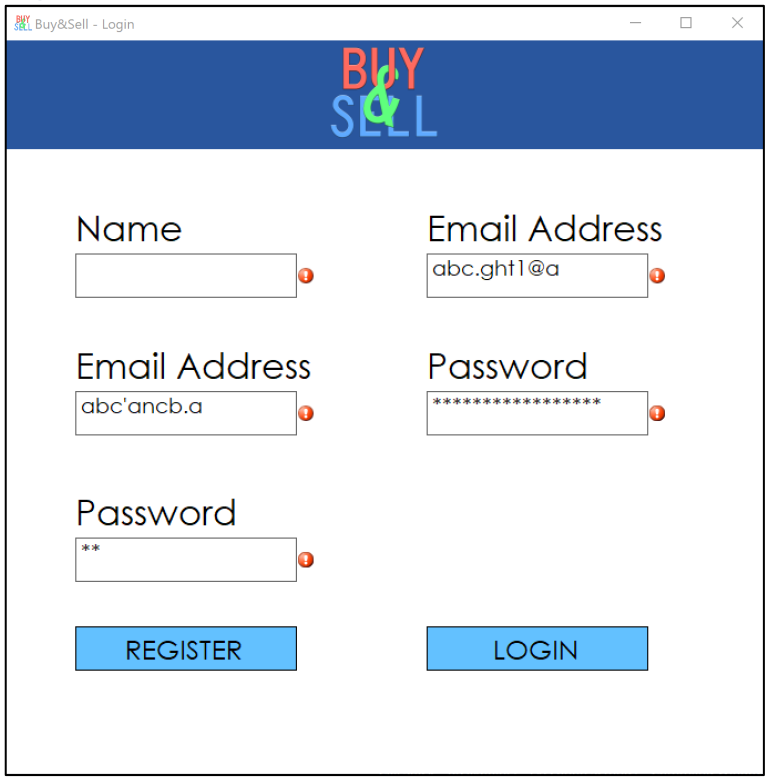
4c	Check that the Listings button works	An error message should come up	Works as expected
4d	Check that the Orders button works	The Seller-Orders page should come up	Works as expected
4e	Check that the Charts button works	The Seller-Charts page should come up	Works as expected
4f	Check that the Show button works	All the item listed by the user should show in the DataGrid	Works as expected
4g	Check if all the data validation works and an error comes up when the data is not valid	All the validations should take place and all the error messages should show up	Works as expected
4h	Check that the List Item button works, and the data is stored correctly in the database	When the user presses the List Item button, all the item values (name, description, price, category) should be stored in the ItemInfo table, the user email and itemID of the item just stored should be stored in the ItemListed table. If the data is not valid then an error message should come up.	Works as expected
SELLER-ORDERS			
5a	Check that the Home button works	The Homepage should come up	Works as expected
5b	Check that the Search button works	An error message should come up	Works as expected
5c	Check that the Listings button works	The Seller-Listings page should come up	Works as expected
5d	Check that the Orders button works	An error message should come up	Works as expected

5e	Check that the Charts button works	The Seller-Charts page should come up	Works as expected
5f	Check that the left Show button works	All the orders received by the user which have been delivered should show on the Data Grid	Works as expected
5g	Check that the right Show button works	All the orders received by the user which have been delivered should show on the Data Grid	Works as expected
SELLER-CHARTS			
6a	Check that the Home button works	The Homepage should come up	Works as expected
6b	Check that the Search button works	An error message should come up	Works as expected
6c	Check that the Listings button works	The Seller-Listings page should come up	Works as expected
6d	Check that the Orders button works	The Seller-Orders page should come up	Works as expected
6e	Check that the Charts button works	An error message should come up	Works as expected
6f	Check that the left Show button works	All the items listed by the user filtered by most sold should show on the Data Grid	Works as expected
6g	Check that the right Show button works	All the items listed by the user filtered by most rated should show on the Data Grid	Works as expected

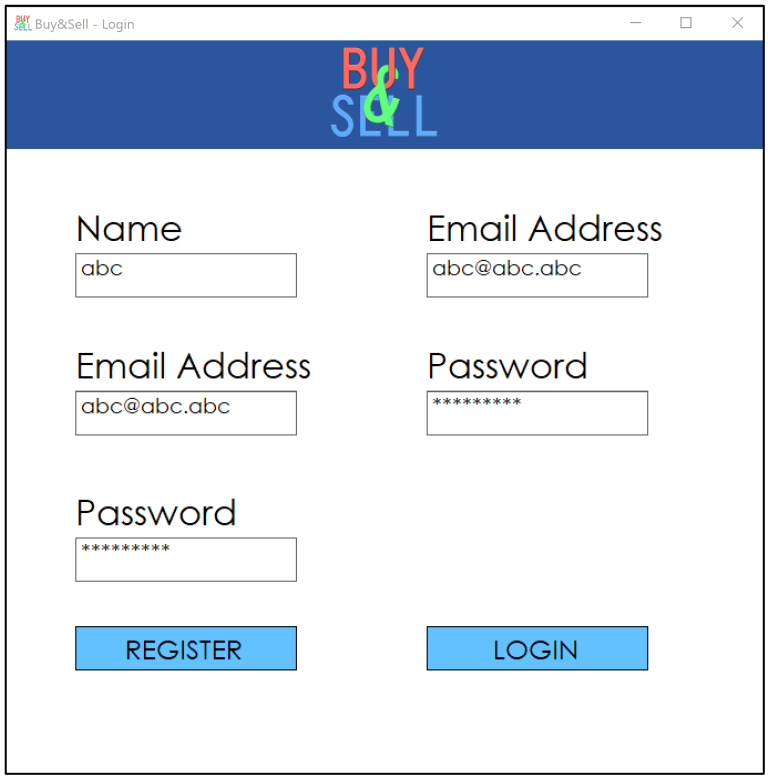
Testing Evidence

There are two types of testing evidence. The first one is screenshots. I have taken screenshots, where applicable, for each test. The second one is a video of me using the application.

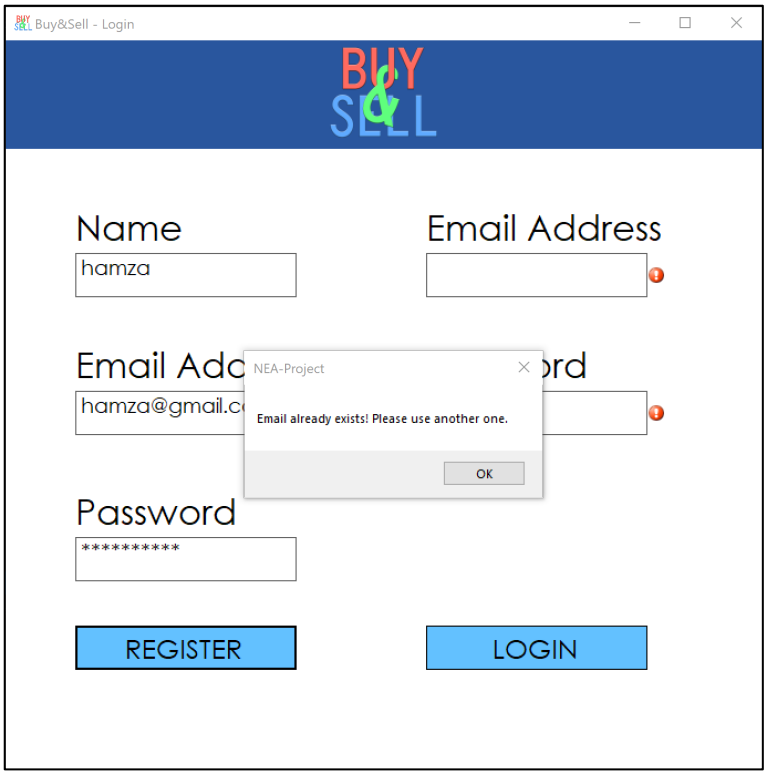
Login



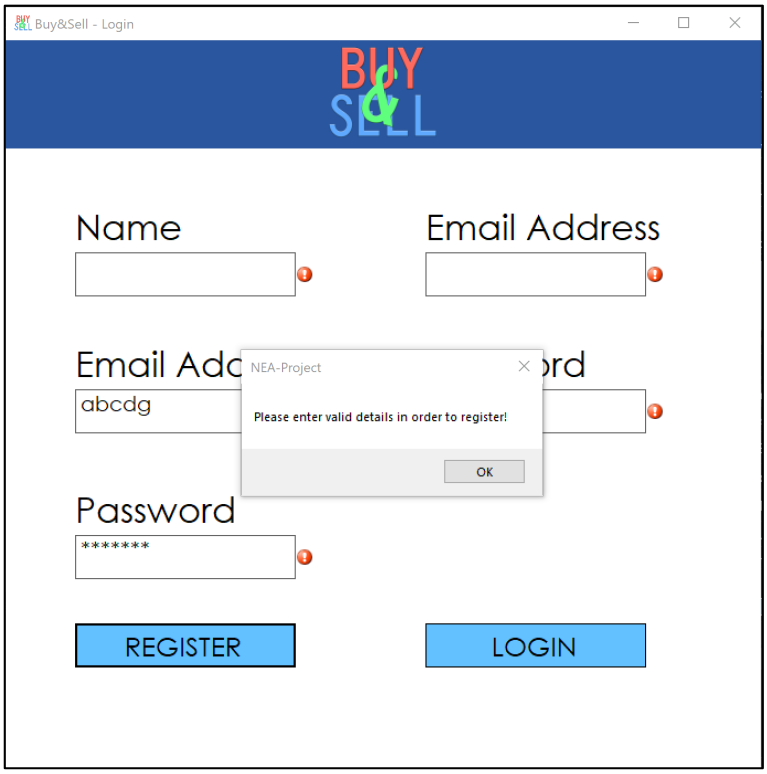
Test 1a



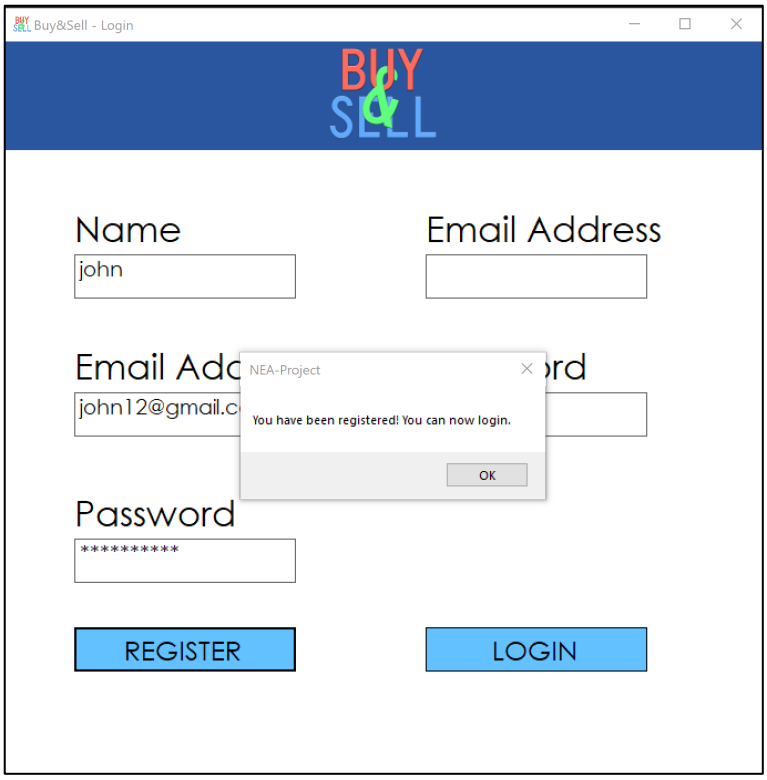
Test 1a



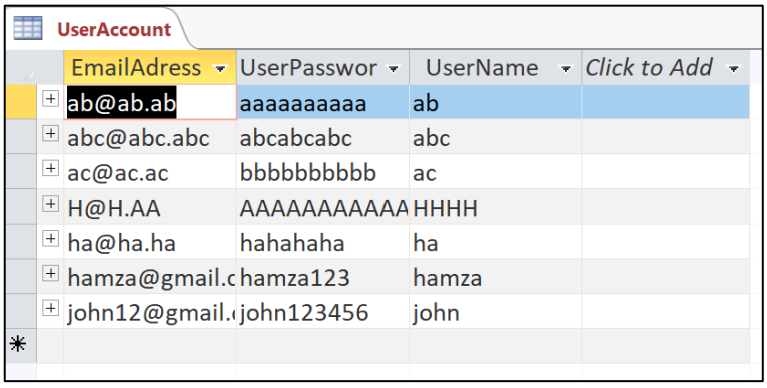
Test 1b



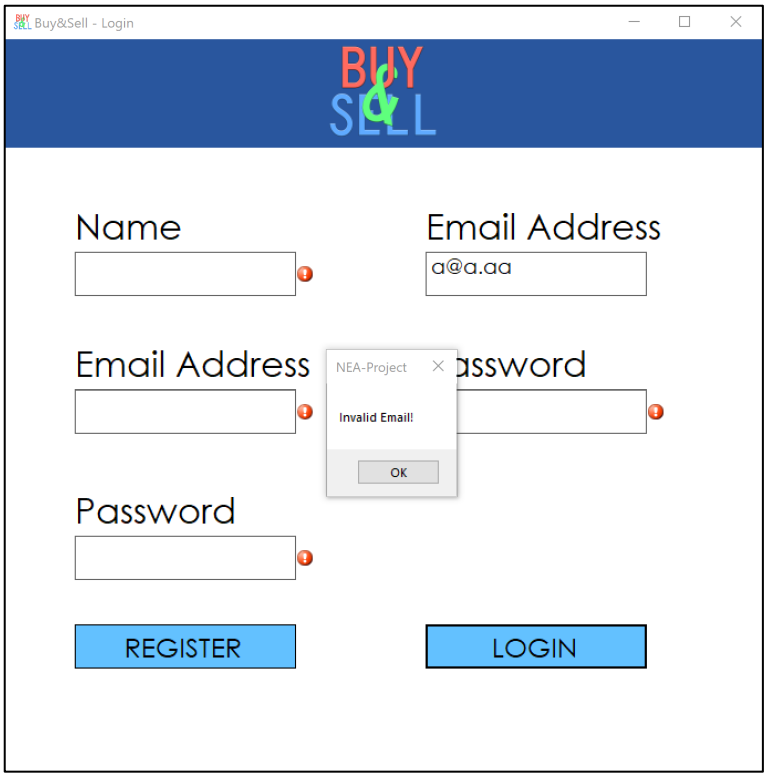
Test 1c



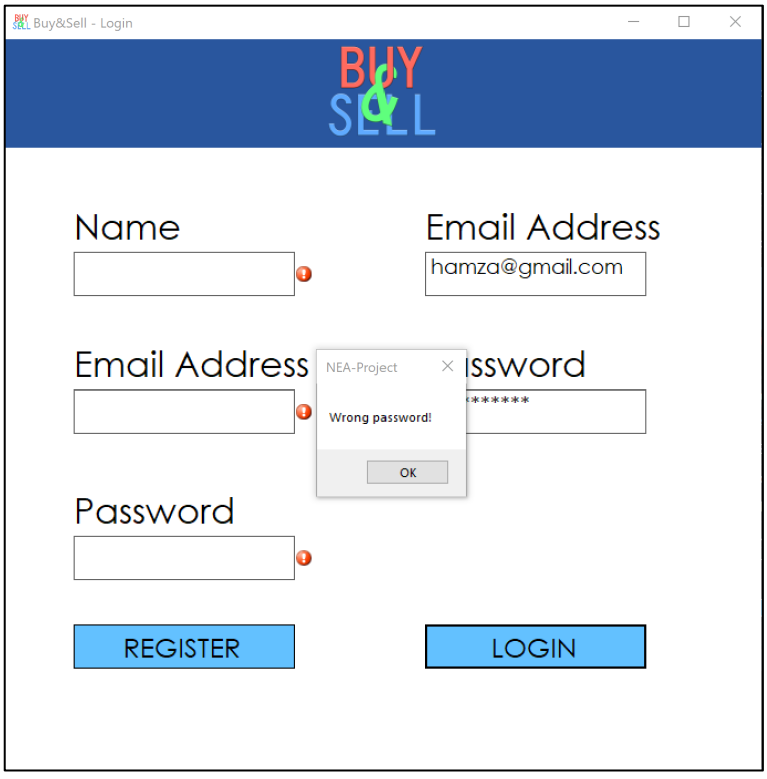
Test 1c



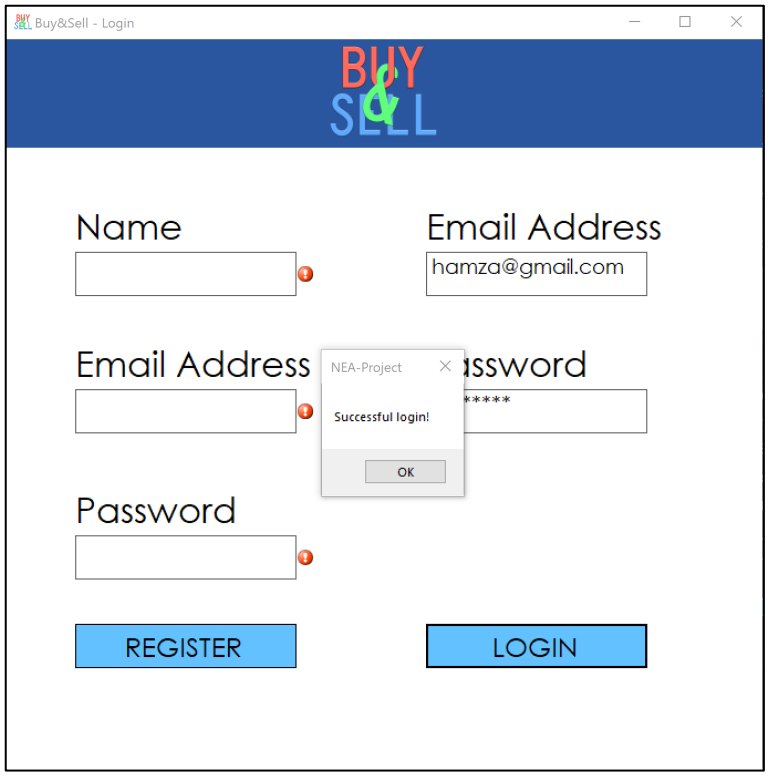
Test 1d



Test 1e



Test 1e


Buy&Sell - Login
BUY
&
SELL
Name
Email Address
hamza@gmail.com
Email Address
NEA-Project
Successful login!
OK
Password
*****
Password
REGISTER
LOGIN

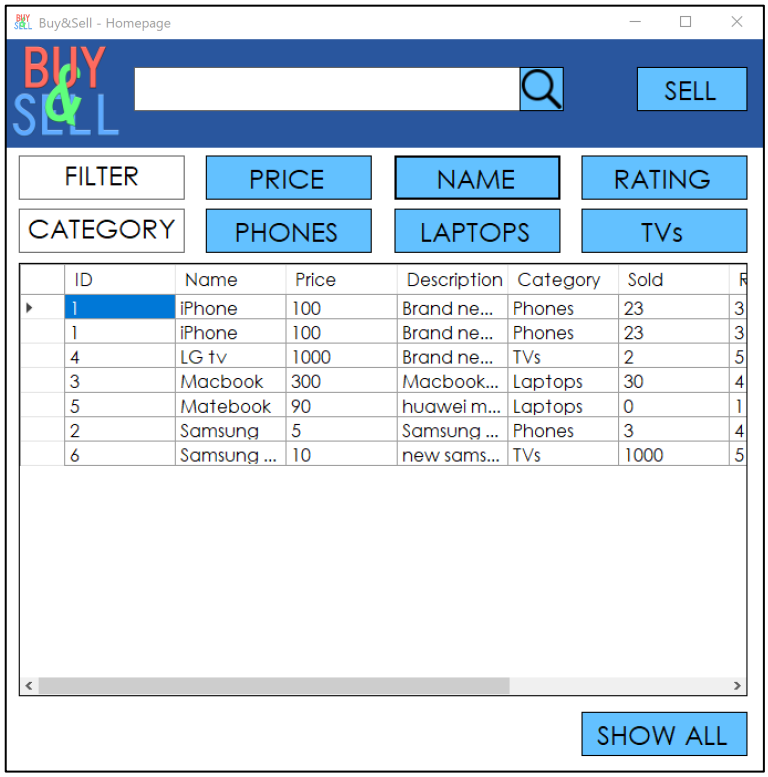
Test 1e

Homepage

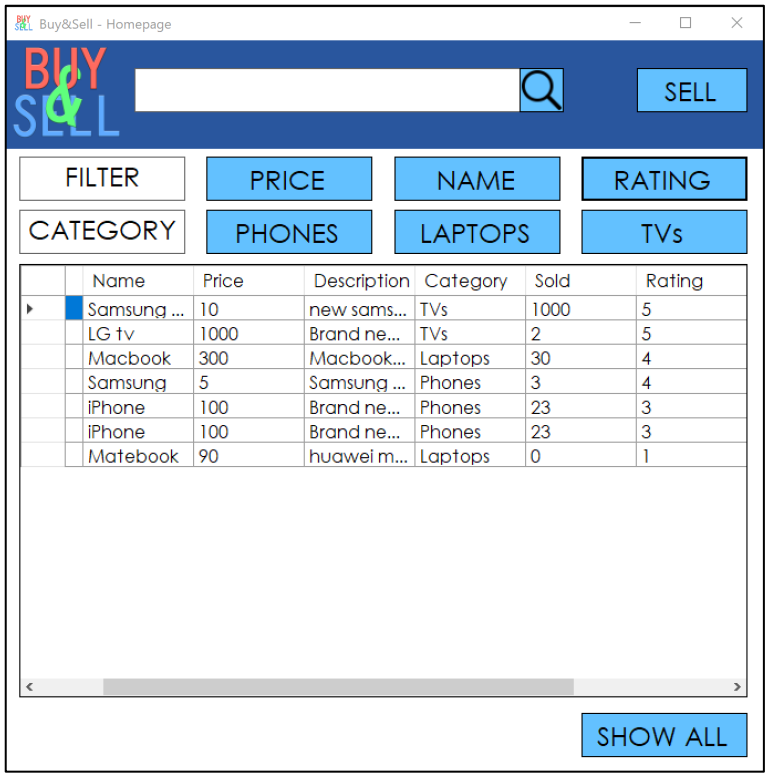
Test 2b

Test 2c

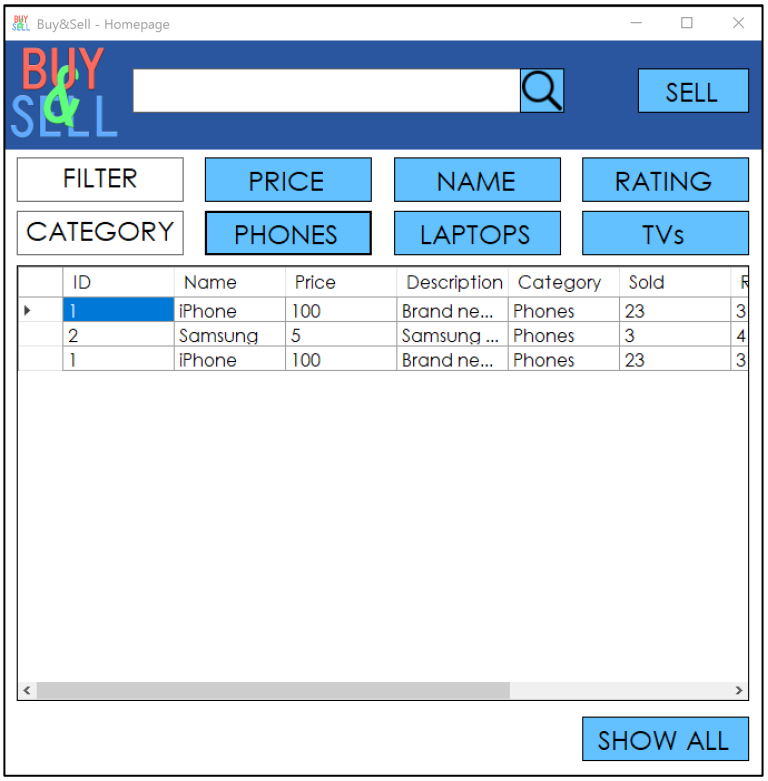
Test 2d



Test 2e

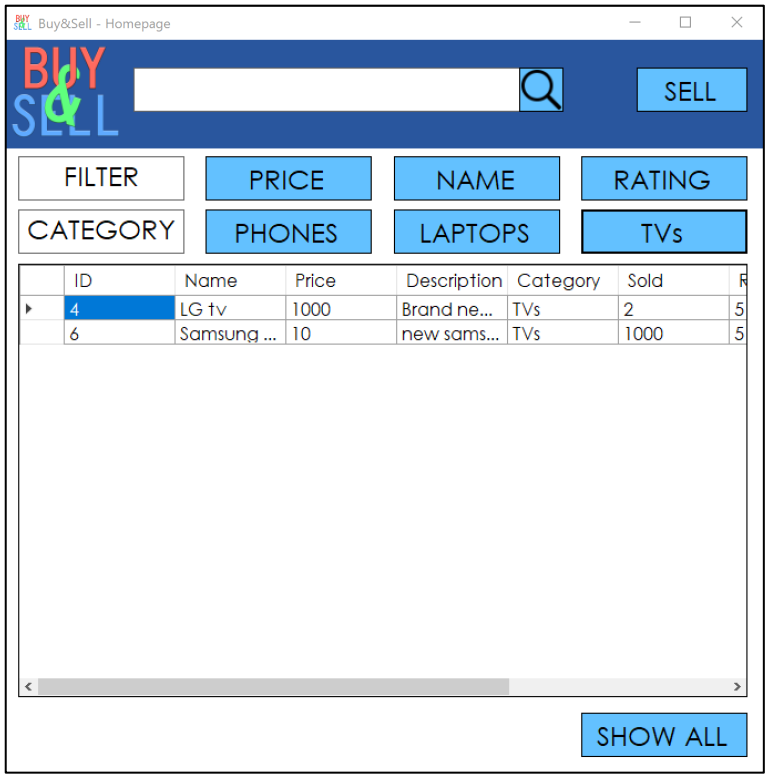


Test 2f



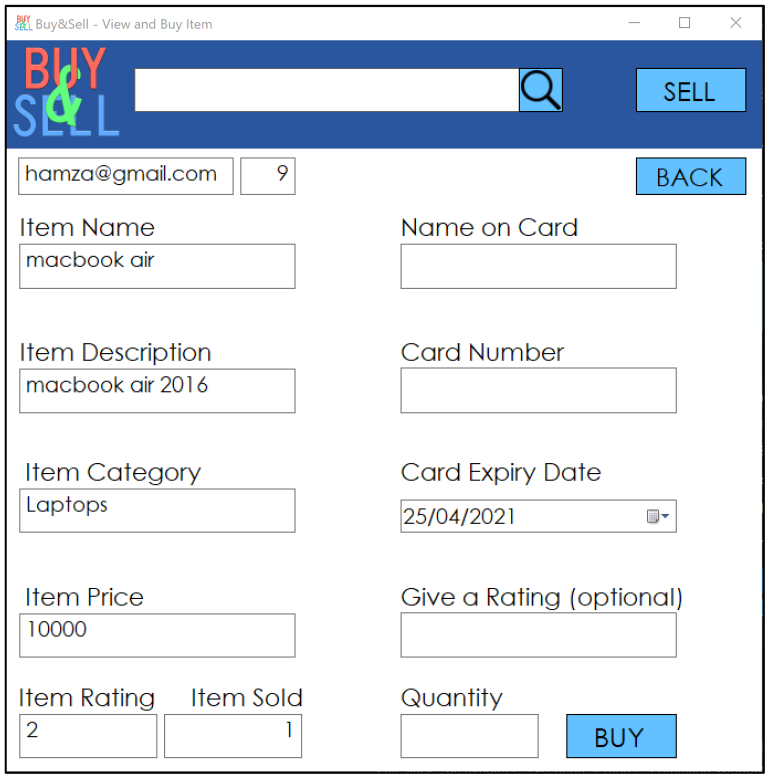
Test 2g

Test 2h

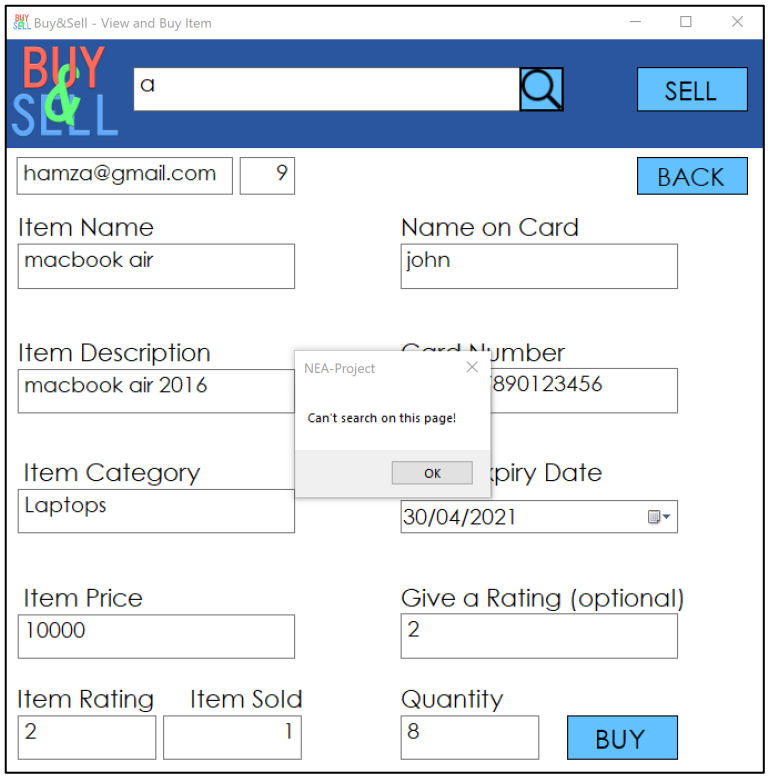


Test 2i

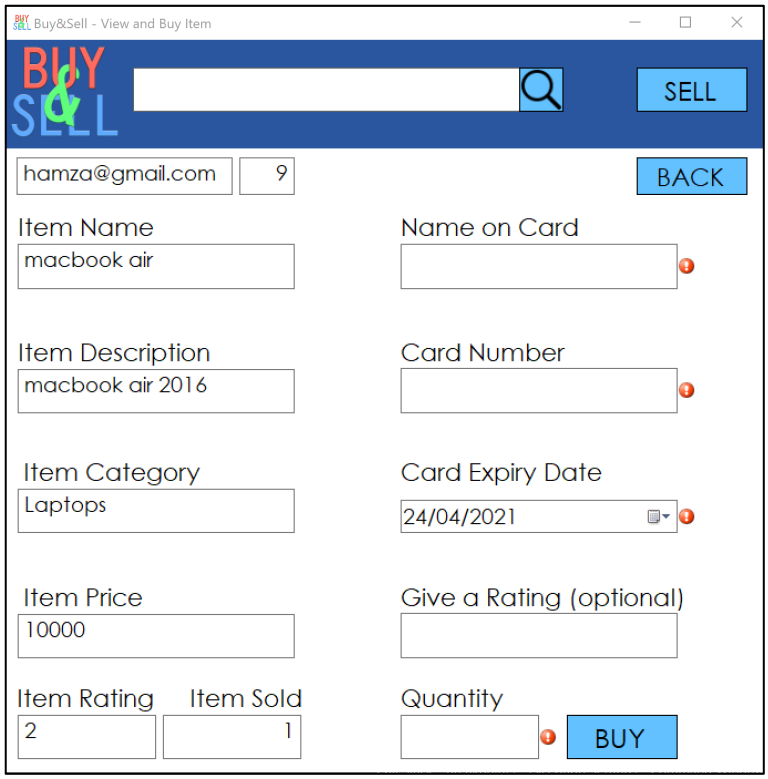
View-Buy Item



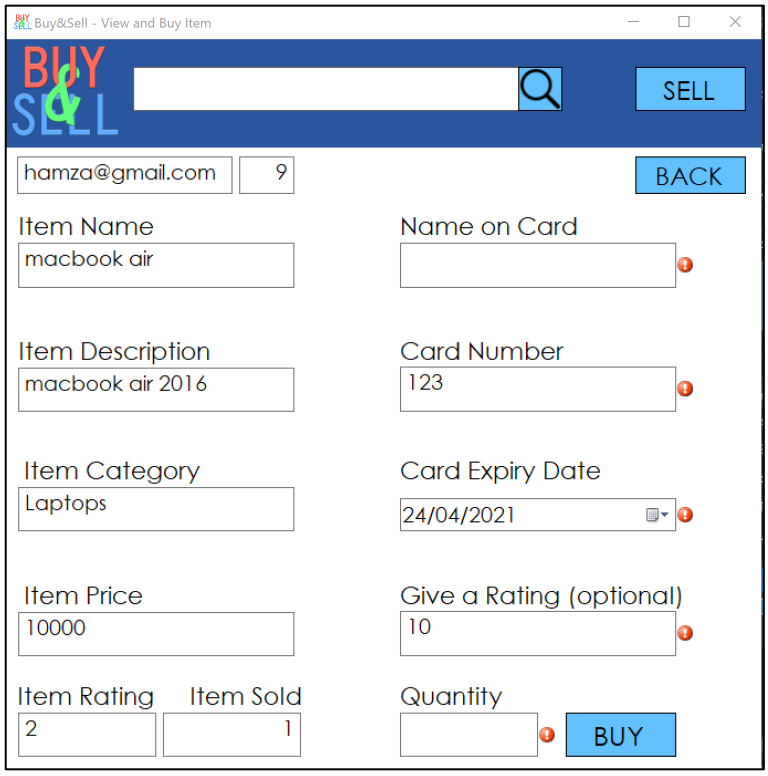
Test 3a



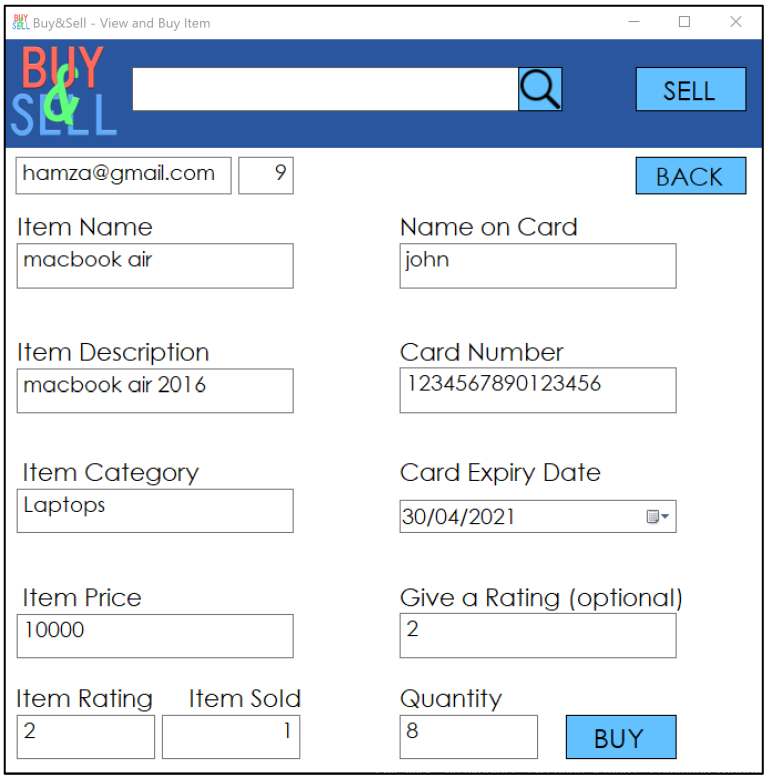
Test 3d



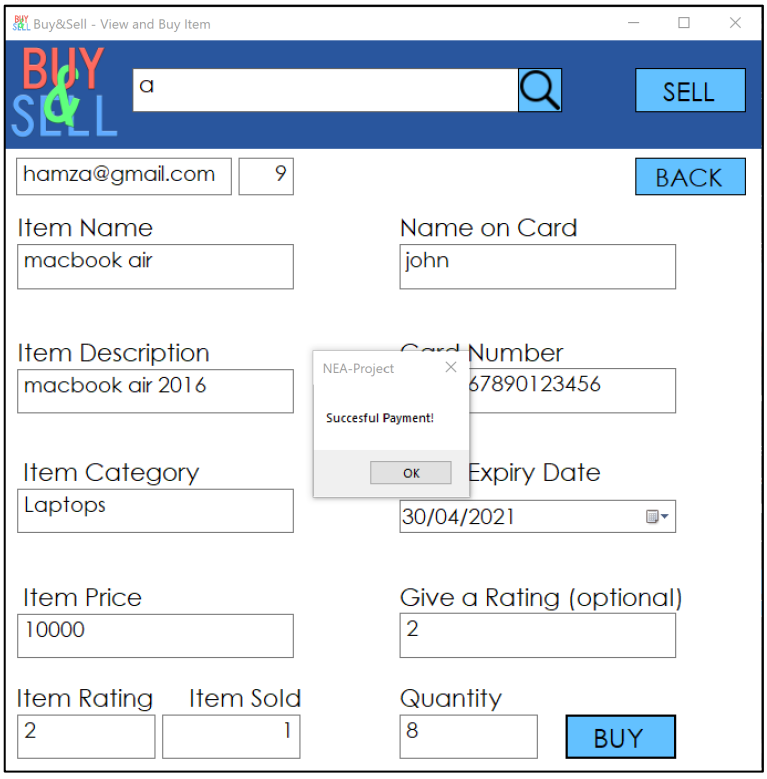
Test 3e



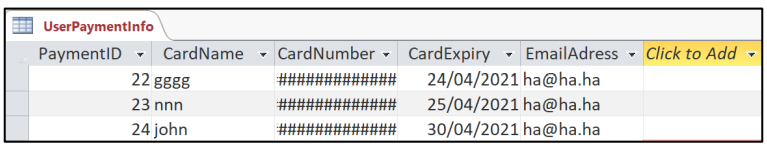
Test 3e



Test 3e



Test 3f



UserPaymentInfo

PaymentID	CardName	CardNumber	CardExpiry	EmailAdress	Click to Add
22	gggg	############	24/04/2021	ha@ha.ha	
23	nnn	############	25/04/2021	ha@ha.ha	
24	john	############	30/04/2021	ha@ha.ha	

Test 3f

OrderInfo

OrderID	ItemID	ItemDelivere	ItemQuantity	Click to Add
22	1	☑		
23	2	☐		
24	3	☑		
25	4	☐		
26	5	☑		
27	6	☐		
28	1	☐		
29	8	☐		
30	9	☐		
31	10	☐	8	
32	10	☐	10	
33	9	☐	8	

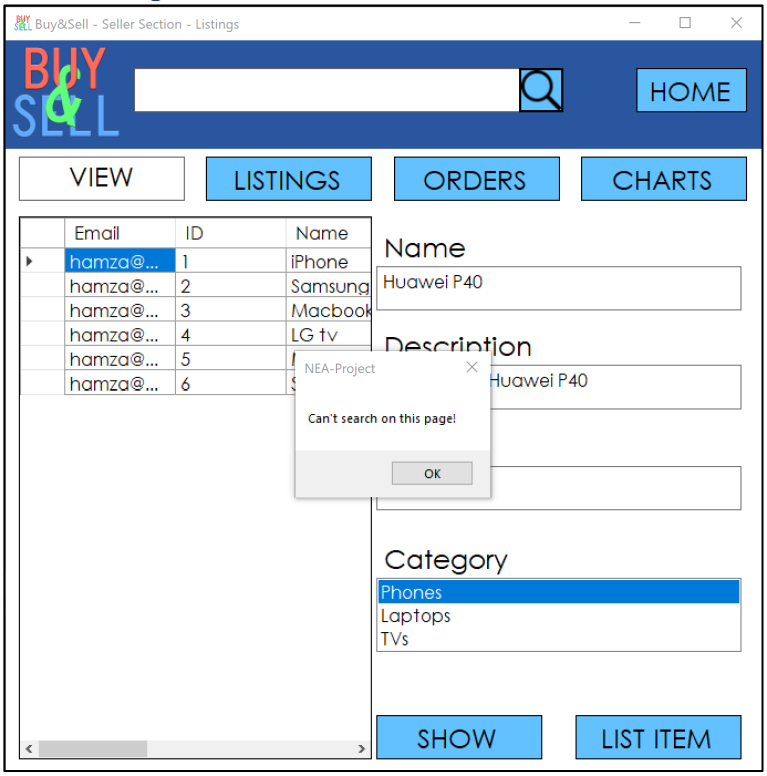
Test 3f

UserOrder

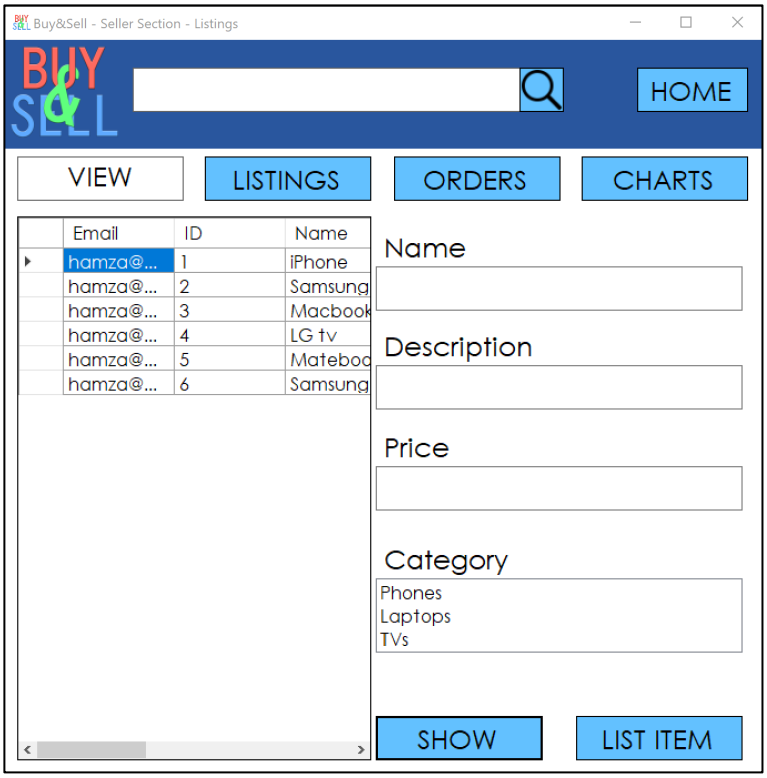
EmailAdress	OrderID	Click to Add
hamza@gmail.c	23	
hamza@gmail.c	24	
hamza@gmail.c	25	
hamza@gmail.c	26	
hamza@gmail.c	27	
hamza@gmail.c	28	
hamza@gmail.c	29	
hamza@gmail.c	30	
hamza@gmail.c	31	
hamza@gmail.c	32	
hamza@gmail.c	33	

Test 3f

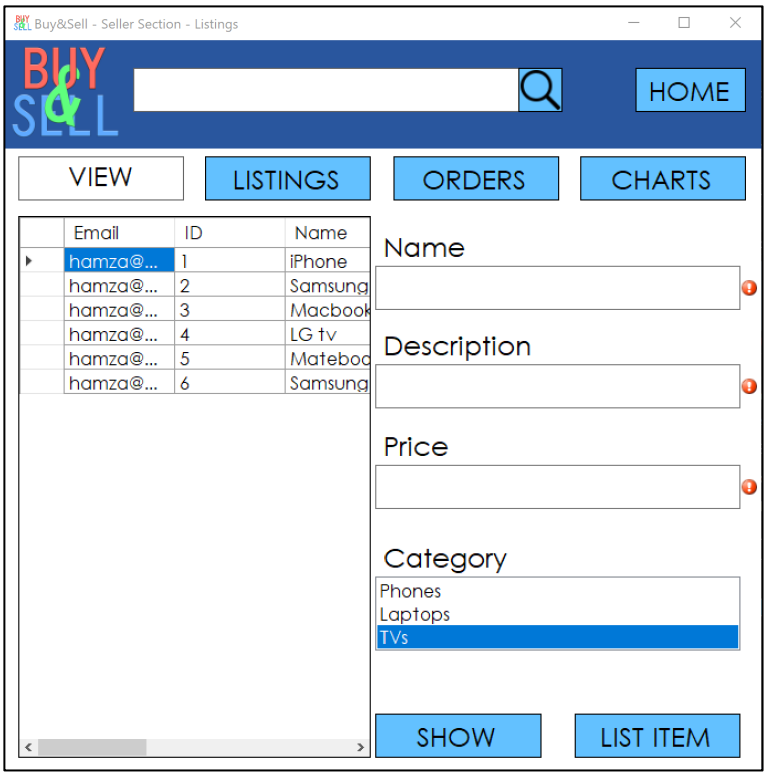
Seller Listings



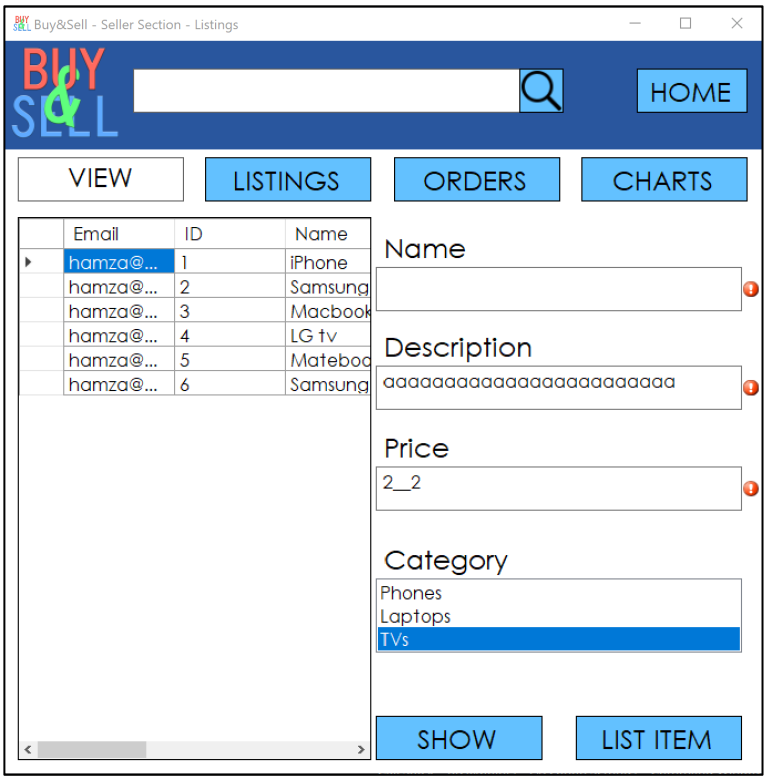
Test 4b



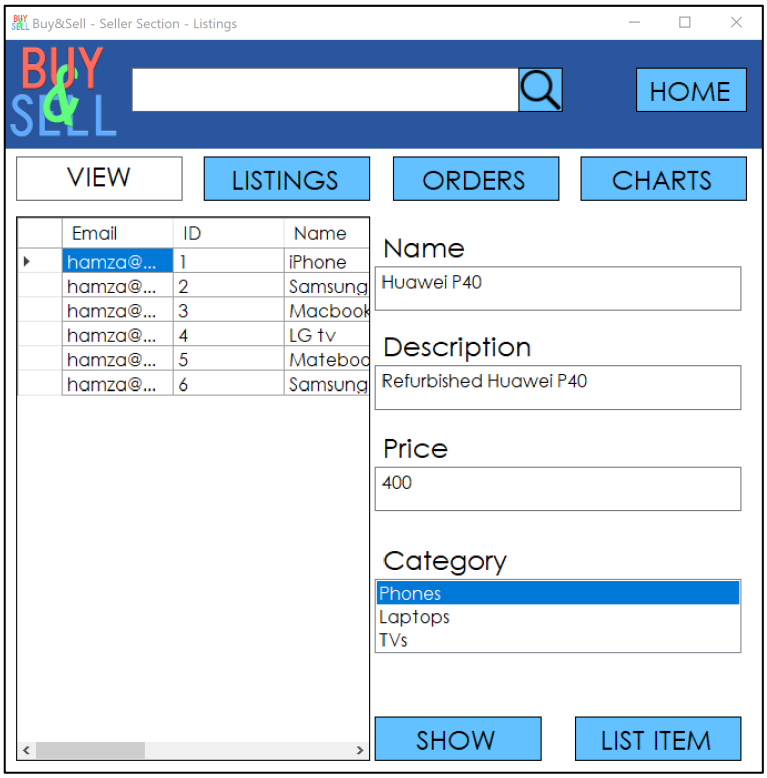
Test 4f



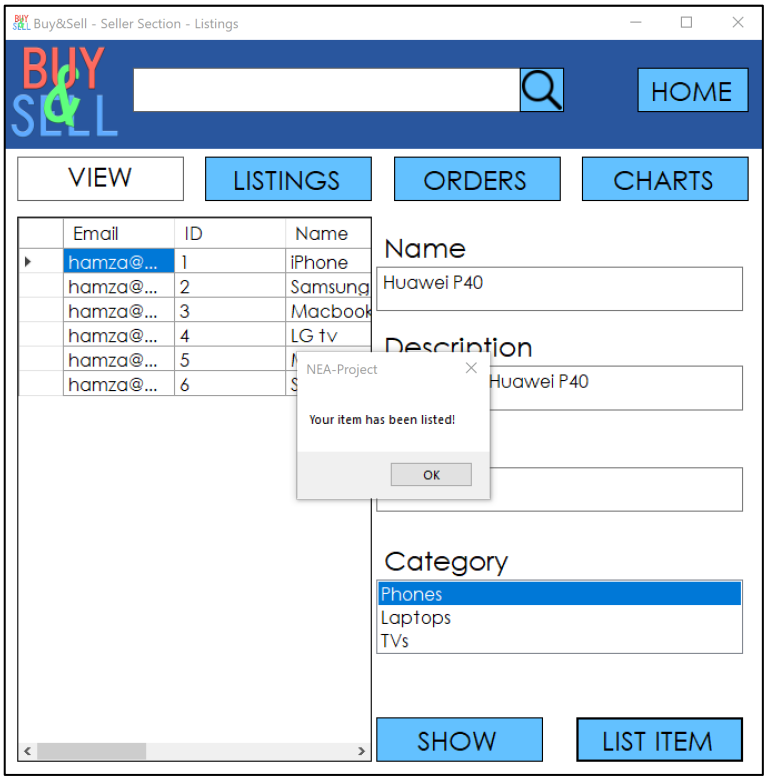
Test 4g



Test 4g



Test 4g



Test 4h

ItemInfo

ItemID	ItemName	ItemPrice	ItemDescript	ItemCategor	ItemSold	ItemRating	Click to Add
1	iPhone	£100.00	Brand new ipho	Phones	24	3	
2	Samsung	£5.00	Samsung S20	Phones	3	4	
3	Macbook	£300.00	Macbook Pro	Laptops	30	4	
4	LG tv	£1,000.00	Brand new LG t	TVs	2	5	
5	Matebook	£90.00	huawei matebo	Laptops	0	1	
6	Samsung tv	£10.00	new samsung t	TVs	1000	5	
8	Huawei P40	£400.00	Refurbished Hu	Phones	0	0	
(New)		£0.00			0	0	

Test 4h

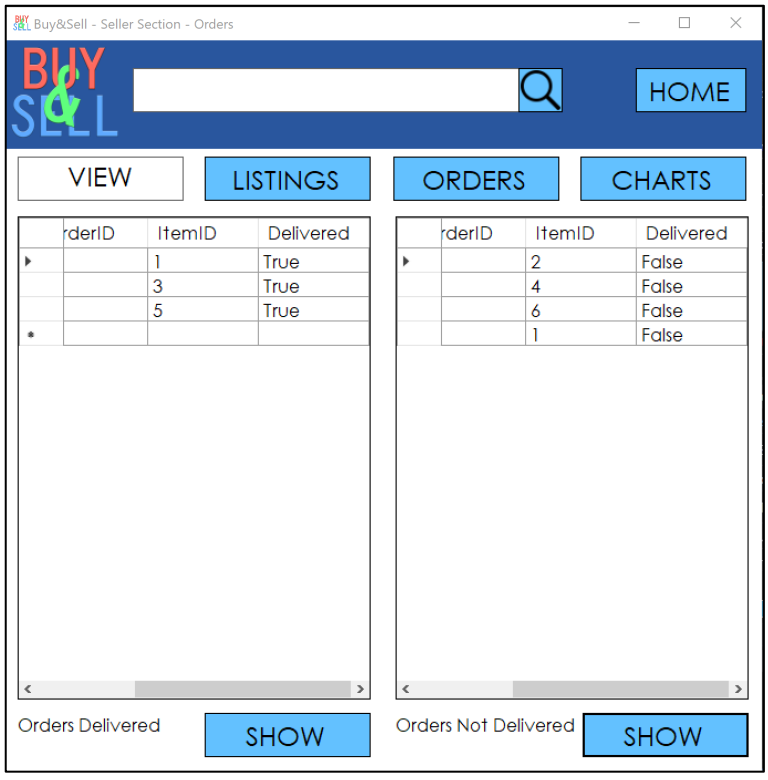
ItemListed

EmailAdress	ItemID	Click to Add
abc@abc.abc	1	
hamza@gmail.c	1	
hamza@gmail.c	2	
hamza@gmail.c	3	
hamza@gmail.c	4	
hamza@gmail.c	5	
hamza@gmail.c	6	
hamza@gmail.c	8	
	0	

Test 4h

Seller Orders

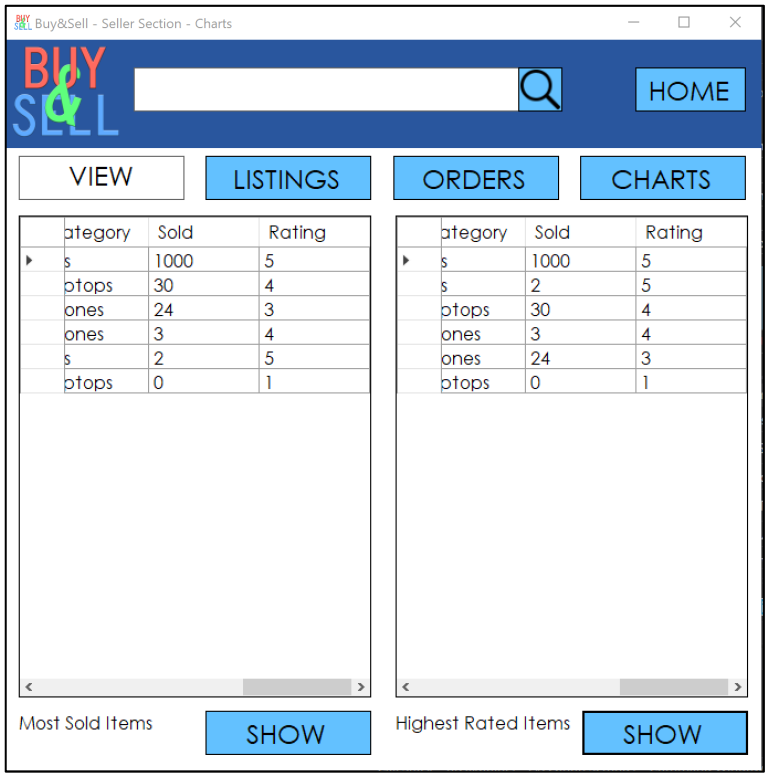
Test 5b



Test 5f/5g

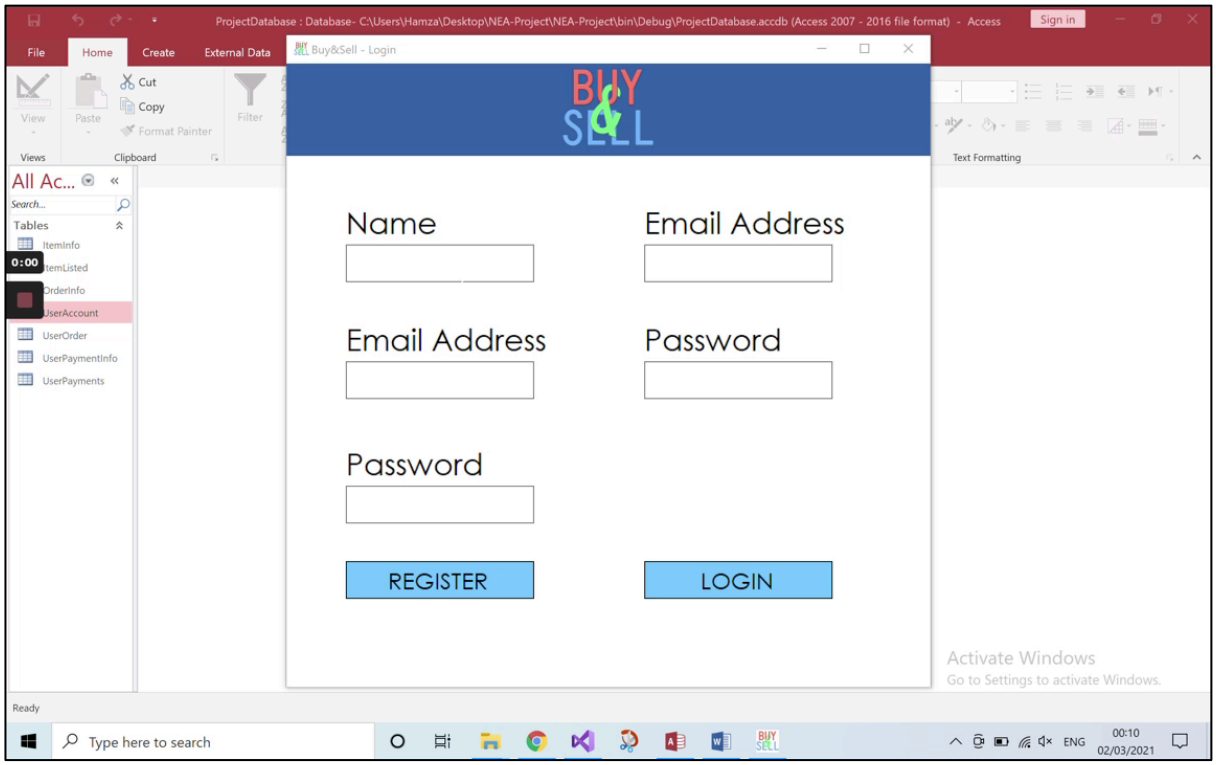
Seller Charts

Test 6b



Test 6f/6g

Test Video



In this video I go through each test. I also show that the data is saved into the database. The video can also be found on YouTube. The link is: https://youtu.be/Gz6Fp_cZot0
*The test video was made before a small change was made to the program. The video does not show that the user can decide the quantity of the item he wants to buy. Another change is that the UserPayments table has been removed from the database, as it was not necessary.

Third Party Test

TEST	SUCCESS (YES,NO)
Register by entering valid details in the textboxes. Login using the same values you used to register. Press login button and successfully login	Yes
View all the items in the homepage by clicking the “Show” button. And filter/order the table by using the appropriate buttons. Search items in the table. Go to the seller section	Yes
Click the “View item’ button at the end of each row and go to the view/buy section. Display the correct information about the item and not being able to edit them. Buy the item by entering the correct values.	Yes
Go to seller-listings section by pressing the “Sell” button from any window. See all the	Yes

listings in the table, only made by the current user. List an item by entering valid values in the textboxes.	
Go to seller-orders section. See all the delivered orders in the right table. See all the not delivered orders in the left table.	Yes
Go to seller-charts section. See all the item listed by the current user ordered by most sold in the right table. See all the item listed by the current user ordered by highest rated in the left table.	Yes
Was the application easy to navigate and understand?	Yes
Is there something you would like to see improved?	I would like to see images in the homepage for each item, possibly more than one. Give the possibility to the seller to eliminate or edit a listing he has made. Finally see different charts in the charts section.

EVALUATION

Overall Effectiveness of Solution

General Objectives

1. **Create an application where users can buy or sell electronic devices.**

I believe I have met this objective as I have created an application where a user can buy items or sell them. In the homepage the user can see all the items available and then buy them. He can also go to the seller section, where he can list items he would like to sell as well.

2. **Create a user-friendly interface with a minimalistic design.**

I believe I have met this objective. The application uses a very minimal design, based on different shades of blue. The design is consistent throughout the whole application. Every button has the same colour, to not confuse the user. This is supported by my third-party user, who shares the same opinions regarding the design.

3. **Divide the application in two sections, one for the buyers and one for the sellers.**

I have met this objective. The application is divided in two distinct section, one to buy items and the other one to sell items. When the user logins, he finds himself in the homepage. Here he can see all the items he can buy. The user can click on an item to see it in more detail and buy it. The user can then go to the seller section. Here he can list items to sell them, he can view all his items listed as well. He can also manage his orders, by seeing which order has been delivered and which not.

4. **Give the opportunity to create an account to the user.**

I have met this objective. When the user opens the application, he needs to register himself to create an account. Each account has a different email address, so every account is unique. After registering the user can use the account to login.

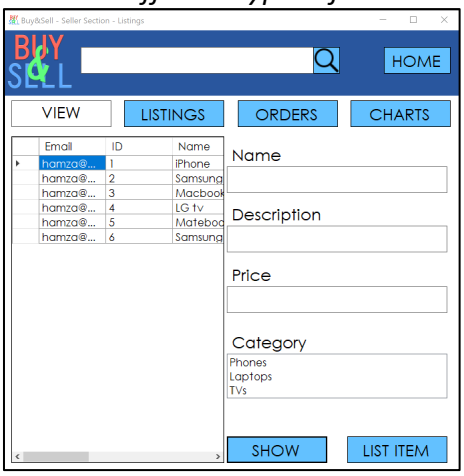
5. **The application will be easy to navigate, run quickly and efficiently.**

I have met this objective. Supported by the third-party feedback, the application runs smoothly, it is easy to navigate and understand.

Specific Objectives

1. ***Create an application where users can buy or sell electronic devices:***

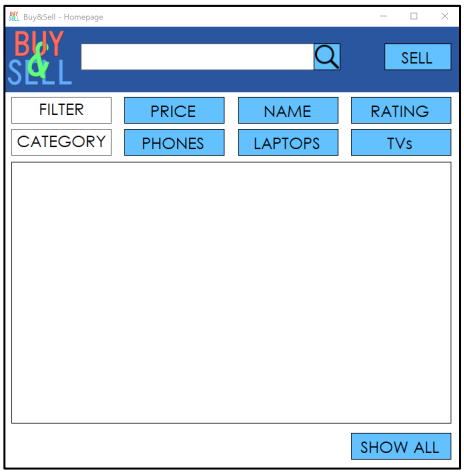
- *The user will be able to buy different types of electronics devices, such as TVs, phones, gaming consoles and laptops.*
- *On the other side, a seller will be able to sell different types of electronic devices.*

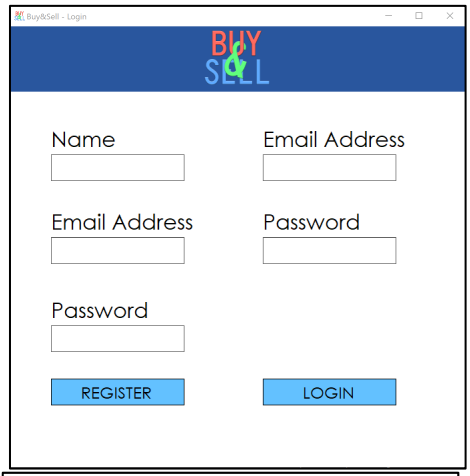


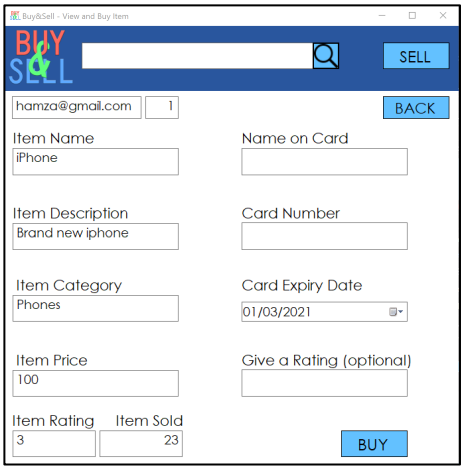
As seen in the screenshots, I have met this objective. On the right side, we can see that the user can see different types of electronic items and buy them afterwards. On the left side, we can see that the user can sell different types of items.

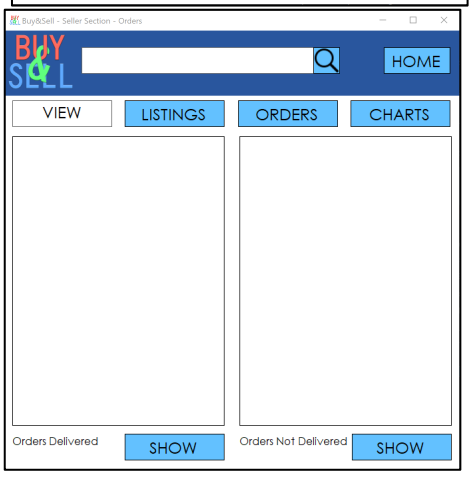
2. ***Create a user-friendly interface with a minimalistic design:***

- *The application will have a simple and minimalistic design. The text will be of adequate size, therefore clear and easy to read for everyone.*
- *It will have a defined scheme of colour, which will be based on the colour blue with red accents to catch the user attention, which will calm the mind and aid concentration.*







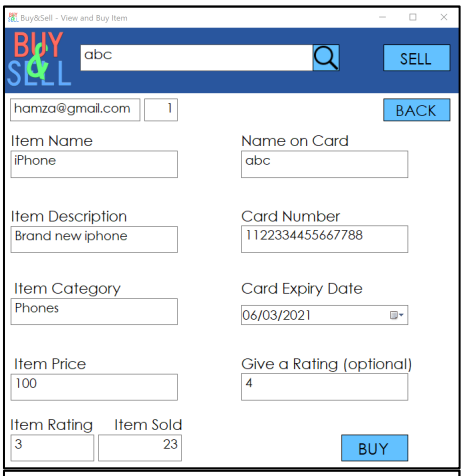


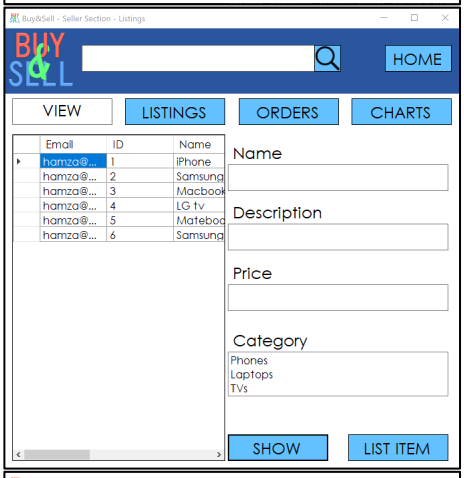
As we can wee from the screenshots, I have met this objective. The style of the application is consistent throughout the whole system. Each screen has the same banner at the top. The font is the same on all pages, and the text is easy to read. There is only red accent in the logo, as I thought that it was not necessary to add more of it. All the buttons have the same colour and design.

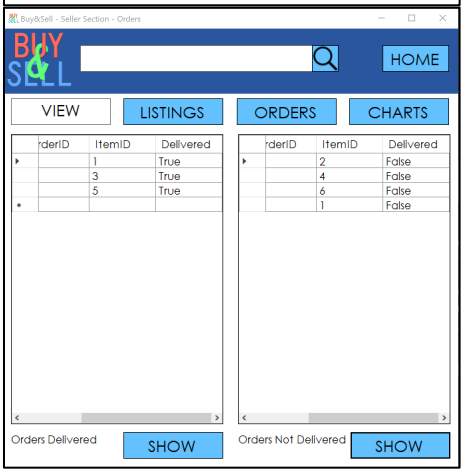
3. ***Divide the application in two sections, one for the buyers and one for the sellers:***

- *After logging in, the user will be re-directed to the homepage. The homepage will be the buyer section. The user will be able to change to the seller by clicking the "sell" button on the top right corner.*

- *In the homepage, the user will be able to see the products, their images and their description + price. The buyer will be able to search anything by using the search bar at the top. The buyer will also be able to filter his search by price, name, popularity or rating. At the top of the homepage, there will be different categories of products.*
- *When buying a product, the user will have to enter an address and his credit card details. If the credit card is invalid, an error message is displayed and the user has to re-enter the details. If the payment is made successfully, an invoice will be displayed on-screen. The user can then take a screenshot of it.*
- *In the user section, the user will have a menu at the left. From there, he will be able to see which products he is selling, what products he has sold, how much he has spent and how much profit he has made. There will be also a section with graphs. These graphs will show which products sold the most, which are making the most profit. These should help the seller decide what to sell and what not to sell.*





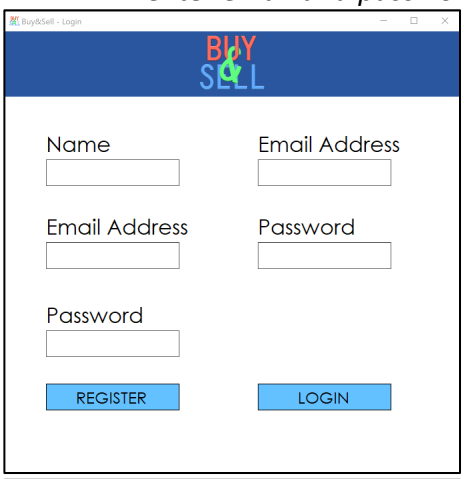


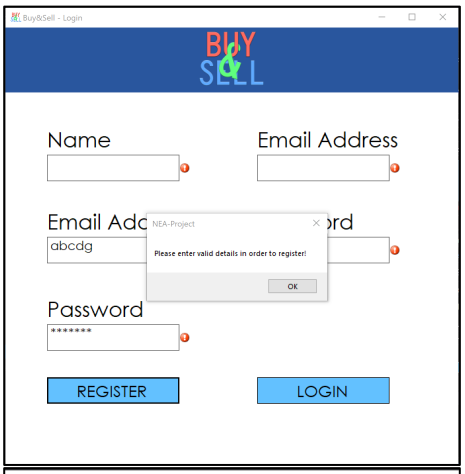
As we can see from the screenshot, I have mostly met the objective. When the user login he can see all the different items. He can search and filter them. From the homepage the user can go to the seller section. When viewing an item, the suer has to enter a valid credit card number and a valid expiry date in order to buy the item. However, a receipt is not shown at the end of the payment. In the seller section, the user has three different windows to choose from. In the first he can see all the item listed, their info and list new items. In the second one, the user can see all his orders. And in the last one the user can see his most sold items and his highest rated items.

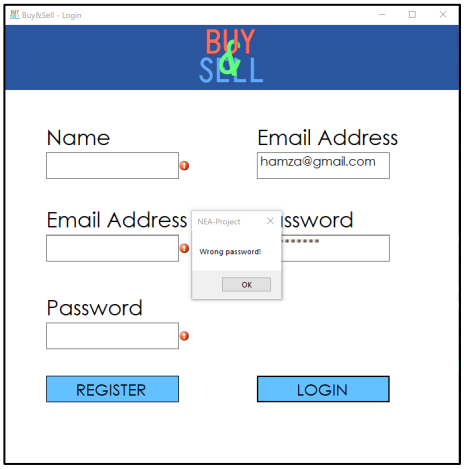
4. ***Give the opportunity to create an account to the user:***

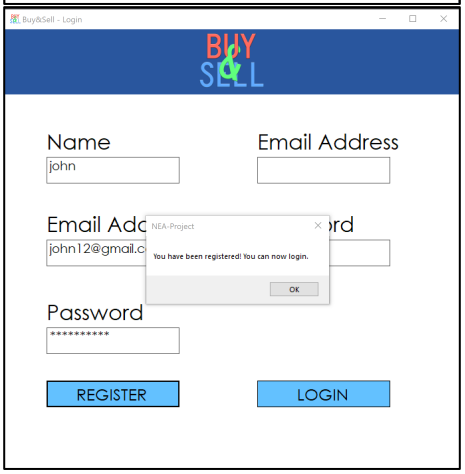
- *When the user opens the application, he will have to create an account or sign in if he already registered.*

- *When registering, the user will have to enter his/her name, email and password. The password will have to meet requirements, if it doesn't the user will have to re-enter the password.*
- *When logging in, the application will check that the email and password entered by the user match those in the database. If it doesn't, the user will have to re-enter email and password.*









As seen from the screenshots, I have met the objectives. When the user opens the app, he has to register himself in order to login. When registering, all the details must meet the conditions, if that does not happen then an error message is displayed. The user can only register when all the details are valid. After registering, the user can login to the system. The values entered must match in the database, if not then an error message is displayed.

Analysis of Third-Party Feedback and Suggestions

In general, the feedback was positive from the different users. Most of them enjoyed using the application, the system was easy to understand and navigate. There are some things they suggested to add in order to make the experience better. First of all, nearly all of them said to add pictures/images of the items in the homepage. This will allow the buyer to see the item he is buying. The seller will then have to upload pictures as well when listing an item. This would be optional, and there would be a maximum of five pictures per item. This was initially part of the project, but for time factors I was not able to insert this feature.

Another suggestion was to add the possibility to edit/delete item listed by the user. This could be done by allowing the user to edit/eliminate the data in the Data Grid and then press an “edit” button in order to save the changes. This would be useful as sometimes mistakes can be made when listing an item, or when the item is sold out the seller can decide to remove it. Finally, they suggested to add charts in the “charts” section instead of tables. I think adding histograms would be useful as the user can easily see, for example, which item has the highest rating. It would also look nicer and more elegant.

Potential Improvements and Extensions

Alongside the suggestions made by third party users, there are some improvements I would like to make.

The best improvement in my opinion would be to move the application online, as well as the database. This would make the app more accessible to people, increasing its possibilities of succeeding. Having an online database has its advantages. For example, it is more secure, the data can be accessed by anyone at anytime from anywhere. It also has its disadvantages, for example, no internet connection means no access to the database. Same applies to the application, if someone does not have access to internet, he can’t use the application.

Another improvement would be to add more categories of items people can sell. Extend the app from being a marketplace of electronic devices to a marketplace of all kinds of products. This would bring more attention to the platform.

One important development would be to protect the data of the user when inserting it in the database. Not all the data would have to be protected. I think only the password of accounts and the card number of every payment stored. One way I would avoid doing this is by using a Caesar cipher. This is the most basic and most insecure type of encryption. Letters of the alphabet are shifted by a fixed amount, for example, all the letters are shifted by 3, so A would become D and so forth. A more secure way would be using a Vernam cipher encryption. Encryption/decryption is made bit by bit using an XOR operation with the shared key, which must be destroyed afterwards. The shared key must be random and used one time only.

Lastly, I would make the application more detailed. For example, when registering the user has to enter his date of birth, or when buying an item, he would have to enter his address. All this data could then be used to study the application, in order to understand in more detail its customers. Or for example, create a ‘basket’ where people can put different items and pay for them together, instead of buying them separately.