**Bachelor’s Degree Report (B.Sc.)**

**AUN SAFESHOP**

***A Project Report***

**By**

**Usman Idris KWARU (A00017145)**

**&**

**Olaoluwa NWOBODO (A00016734)**

**Supervisor: Narasimha Rao Vajjhala (PhD)**

**School of Information, Technology & Computing American University of Nigeria**

**May 2018**

**AUN SAFESHOP**

**Usman Idris KWARU &**

**Olaoluwa NWOBODO**

***BSc Report***

**Project Report submitted in**

**Partial fulfillment of the requirements for the Degree of Bachelor in Software Engineering**

**Supervisor: Narasimha Rao Vajjhala (PhD)**

**School of Information, Technology & Computing American University of Nigeria**

**May 2018**

# DECLARATION

We declare that this project work is carried out by ourselves and has not been previously submitted for the degree. And that the report is written unaided in our own words, apart from any quoted material which we identified clearly in the correct manner and fully acknowledged work by others. The work and the report were carried out under the guidance of Dr. Narasimha Rao Vajjhala.

Student's Name:

Usman Idris Signature.....................................

Student's Name:

Olaoluwa Nwobodo Signature.....................................

Supervisor's Name:

Dr. Narasimha Rao Vajjhala Signature.....................................

Program Chair's Name:

Dr. Narasimha Rao Vajjhala Signature.....................................

Dean's Name:

Dr. Mathias Fonkam Signature.....................................

SDP Coordinator’s Name:

Dr. Abubakar Sadiq Hussaini Signature.....................................

# ABSTRACT

Trust is a major factor in the internet business. This is not surprising as it is very convenient for cyber criminals to swindle unsuspecting customers. AUN SafeShop is an online store for the American University of Nigeria (AUN) Community. AUN SafeShop is a platform for AUN staff, students and faculty to buy and sell new or fairly used items among themselves. This platform is a lot safer for members of the AUN Community because it will be a near transparent system and it will be very difficult for people to be duped. This is because sufficient information would be provided about buyers and sellers of SafeShop (with their permission). People would be able to sell items of their choice on SafeShop, they would also be able to buy items from other sellers. The only items that would be available for sale would be items provided by the AUN community. Another major function of SafeShop would be auctioning and bidding. A seller can decide not to sell an item at a fixed price; he/she can decide to auction it. In this type of event, people can bid on the item(s) and it is therefore sold to the highest bidder. It is a fact that students get to a point in the semester where they need money, and are willing to sell items. AUN SafeShop provides the perfect avenue to such students so that they do not have to go from door to door looking for buyers. Sellers can register on SafeShop and immediately start selling different items. This platform is also directed at bringing students together, especially because a lot of students would prefer to meet with the seller and see the item before paying for it. AUN SafeShop would eliminate the ambiguity of anonymous sellers because it would be running in a small community, thus making it trustworthy and safe.

# ACKNOWLEDGEMENT

With special recognition, we would like to express our sincere gratitude and thanks to our beloved parents whom their support took us this far. We really thank you for all the support you gave us in all our academics and social life. Furthermore, we would like to specially thank Mr. Narasimha Rao Vajjhala and Dr. Abubakar Sadiq Hussaini whom graced us with their continuous and valuable support. God bless you all.

Table of Contents

[DECLARATION 1](#_bookmark0)

[ABSTRACT 2](#_bookmark1)

[ACKNOWLEDGEMENT 3](#_bookmark2)

[LIST OF FIGURES 5](#_bookmark3)

[CHAPTER ONE 6](#_bookmark4)

* 1. [INTRODUCTION 6](#_bookmark5)
	2. [BACKGROUND AND MOTIVATION 6](#_bookmark6)
	3. [PROBLEM STATEMENT 6](#_bookmark7)
	4. [PROJECT OBJECTIVE 7](#_bookmark8)
	5. [OUTLINE OF THE REPORT 7](#_bookmark9)

[CHAPTER TWO 8](#_bookmark10)

1. [REQUIREMENTS 8](#_bookmark11)
	1. [FUNCTIONAL REQUIREMENTS 8](#_bookmark12)
	2. [NON-FUNCTIONAL REQUIREMENT 9](#_bookmark13)
	3. [TECHNOLOGIES USED 10](#_bookmark14)
	4. [USECASE DIAGRAM 10](#_bookmark15)
	5. [USE CASE DESCRIPTION 11](#_bookmark17)

[CHAPTER THREE 15](#_bookmark18)

1. [DESIGN 15](#_bookmark19)
	1. [ARCHITECTURAL DESIGN 15](#_bookmark20)
	2. [METHODOLOGY 16](#_bookmark22)

[CHAPTER FOUR 18](#_bookmark23)

1. [CONCLUSION AND FUTURE WORK 18](#_bookmark24)
	1. [CONCLUSION 18](#_bookmark25)
	2. [FUTURE IMPLEMENTATIONS 19](#_bookmark26)

[References 20](#_bookmark27)

[Appendix 21](#_bookmark28)

# LIST OF FIGURES

[Figure 1 Use case Diagram 11](#_bookmark16)

[Figure 2 Three-tier Architecture 15](#_bookmark21)

# CHAPTER ONE

# INTRODUCTION

## BACKGROUND AND MOTIVATION

Through our stay at AUN, we noticed a particular trend among members of the AUN community, especially the students. We noticed that toward the end of the semester AUN students are in need of cash, and as such are constantly searching for who to buy either, used, fairly used or new items they were willing to do away with. The issue became they had to go from person to person looking for buyers. This was not nearly efficient because students rarely ever found buyers through that method, while the buyers were there, getting to them was the difficult part. Sometimes students needed money to travel, or their meal plans finished and they needed cash. While it is possible for students to use their ID cards to swipe everywhere on the main campus, this is not the case when students go outside the main campus and thus need money that is not tied to their e-Wallet account. This problem amongst others, inspired our decision to take on the challenge and solve this problem.

## PROBLEM STATEMENT

Several members of the AUN community need to buy certain goods from others want to sell, this is made extremely difficult because there is not enough advertisement for these products that they want to sell or buy. The process of finding a buyer, or seller is excruciating, time- consuming and inefficient.

## PROJECT OBJECTIVE

The aim of this project is to ease the process of finding sellers and buyers for different products by creating awareness of said products. SafeShop is an online shopping and auctioning system. This web application would be accessible online and members of the AUN community would be able to login with thier AUN emails and immediately see products that are available for sale. An added advantage of SafeShop is that auctioning system. One more objective of SafeShop is to bring out the true value in product by making it possible for people to auction products and buyers will be able to make bids on products

## OUTLINE OF THE REPORT

Chapter 1 clearly introduces the system to be developed as well as its problems and the objectives.

Chpater 2 states the requirements needed to develop the system. These requirements range from the functional to non-functional requirements.

Chapter 3 outline the design and development of this system. It also talks about the architecture used in developing the system.

Chapter 4 talks about the future implementation for the system as well as the conclusion.

# CHAPTER TWO

# REQUIREMENTS

The requirements gives a detailed description of the system functionalities (what it should do) as well as what it must have. This accumulates to the behavior of the system (functional requirement) and the limits under which the system functions (non-functional requirement). It also states the devices needed for the development of the system.

## FUNCTIONAL REQUIREMENTS

The functional requirements specify the behavior of the system (what it should do) which includes:

* + - Unregistered users shall be able to register as a seller, buyer or both.
		- Unregistered users shall be able to view products on the website without registering
		- Registered users shall be able to login to the website
		- Logged in users shall be able to add items to cart
		- Logged in users shall be able to delete items from cart
		- Logged in users shall be able to check out
		- Logged in users shall be able to buy items
		- Logged in users shall be able to make bids on biddable items
		- Logged in users shall be able to upload pictures and description of products they want to sell
		- Logged in users shall be able to fill a form to lodge a complaint to the developers of AUN SafeShop
		- Logged in users shall be able to logout
		- Logged in users shall be able to view bids made by others for a particular product, only if said user has made a bid
		- Admin users can send warnings to violating users
		- Admin users can deactivate accounts of violating users without prior warning

## NON-FUNCTIONAL REQUIREMENT

The following are the non-functional requirements of the system.

**Accessibility**: The system is expected to work on any browser (Google Chrome, Opera Browser, Mozilla Firefox, Microsoft Edge et cetera). The system is expected to work on desktop systems (Windows OS, Mac OS, Linux OS), and will be accessible through internet connection.

**Usability**: The system should be easy to use and navigate. The system is expected to have a reasonable response time (2 seconds or less)

**Reliability**: The system should be able to host up to 400 logged in users at the same time **Scalability**: The system should be built in such a manner that it is scalable to a larger user base in the future.

**Price**: The system is intended to be free as the technology used in development did not cost money.

## TECHNOLOGIES USED

* + - SQL Server
		- Visual Studio 2012
		- HTML
		- ASP.NET
		- CSS
		- C-Sharp (C#)
		- JQuery
		- Bootstrap
		- Ajax
		- JavaScript

## USECASE DIAGRAM

The use-case diagram below clearly shows the functionalities of the system and which actors are related to each functionality.



**Figure 1 Use case Diagram**

## USE CASE DESCRIPTION

|  |  |
| --- | --- |
| **Identifier and Name:** | **Login** |
| **Initiator:** | BuyerSeller |
| **Goal:** | To gain access to the website |
| **Pre-condition:** | The Buyer and Seller have an account |
| **Post-condition:** | The User is logged in. |
| **Assumptions:** | The user is registered as a Buyer or Seller |

|  |  |
| --- | --- |
| **Main Success Scenario:** | 1. User opens the website
2. Clicks on login
3. User enters his or her credentials
4. User clicks on the login button
5. User gains access to the website
 |

|  |  |
| --- | --- |
| **Identifier and Name:** | **Bidding on items** |
| **Initiator:** | Buyer |
| **Goal:** | To make bids on items |
| **Pre-condition:** | The Buyer is logged in. |
| **Post-condition:** | The buyer makes a bid |
| **Assumptions:** | The item is auctioned off by the seller |
| **Main Success Scenario:** | 1. The buyer clicks on the item he or she wants to bid on
2. The buyer enters the amount he/she wishes to purchase the item for.
3. The buyer clicks on bid
 |

|  |  |
| --- | --- |
| **Identifier and Name:** | **Register** |
| **Initiator:** | BuyerSeller |
| **Goal:** | To successfully register to the website |
| **Pre-condition:** | User has AUN email (…@aun.edu.ng) |
| **Post-condition:** | The user has login credentials |
| **Assumptions:** | The user is a student or staff of AUN |
| **Main Success Scenario:** | 1. User clicks on the register on the site |

1. User fills in the details on the register page
2. User clicks on register
3. User is registered to the website
4. User receives a confirmation email

|  |  |
| --- | --- |
| **Identifier and Name:** | **Sell Items** |
| **Initiator:** | Seller |
| **Goal:** | The seller posts his items on the website |
| **Pre-condition:** | The seller is logged in. |
| **Post-condition:** | Items are posted on the website |
| **Assumptions:** | The user has a seller account |
| **Main Success Scenario:** | 1. The seller logs in to the website
2. The seller clicks on sell
3. The seller fills in the details of the item to sell
4. The seller uploads pictures of the item to sell
5. Chooses whether the item is to be auctioned or not
6. The seller posts the item on the website
 |

|  |  |
| --- | --- |
| **Identifier and Name:** | **Buy Items** |
| **Initiator:** | Buyer |
| **Goal:** | The buyer purchases one or more items |
| **Pre-condition:** | The Buyer is logged in. |
| **Post-condition:** | Items are purchased |
| **Assumptions:** | The user is registered and has an account |
| **Main Success Scenario:** | 1. The buyer clicks on an item
2. The buyer clicks on add to cart
3. The buyer clicks on cart
4. The buyer clicks checkout
 |

# CHAPTER THREE 3 DESIGN

## ARCHITECTURAL DESIGN

This system has three-tire architectural design with Presentation Layer, Web-Server and the Data Layer. The Presentation Layer is the client side which is the user interface and physical appearance. The Web-Server is a special computer and a program host that serves the website and makes it available to users. Data Layer (Database) this is a layer that unifies the communication between the webserver and the database to achieve database abstraction. Below is a diagram showing all the three layers and how they are connected to one another.



**Figure 2 Three-tier Architecture**

## METHODOLOGY

The methodology used in developing this system is Incremental development. Incremental Software development methodology is a process of development where systems are being developed as a protoptype at the initial stage and then more features are gradually added to satisfy the requirement of the system. This methodology is chosen because the system is a small system, the developers concerned are students, staff and faculty (not experts programmers) and the system requirements changes with time.

The main software development methodology used in developing this system is the Agile Development Methodology. The agile methodology releases the constraints on requirments and gives the developers more room for changing requirements, because the only constant thing about software is changing requirements. Due to this methodology, we were able to change correct errors made in the database design and interface design stage, while already implementing. Just like the name “agile” suggests, we can easily go back and forth throughout the development process, unlike the waterfall model, ,which does not allow this freedom.

The waterfall model generally demands that one we have passed a certain stage, e.g. requirements collection, we could not go back. Our main reason for not using the waterfall model was because as student developers, we cannot completely avoid making design and implementation errors, and when those errors occur, we would be stuck with them and that would be detrimental to the system being developed.

Sevaral methodologies have originated from the agile methodology that we also essential to the success of this project. One of which is the Feature Driven Development methodology.

Feature driven development is a branch of agile software development that focuses the development process on the functionality of the system, rather than the overall product/project. This software development methodology requires developers and users to work very closely

together. First of all, an overall model of what the system should look like. Then a features list

has to be created, hence feature driven development, this is a very essential stage in the process, the features list is to ensure that the developer can track their progress. A plan is developed for each feature, then the features are designed. This methodology is centered around features and therefore each feature is treated as a module. It is only after the features are designed, that they start being developed.

There is a lot of benefits with using the FDD, one of which is the ability to easily track the progress of the project. Tracking the project is as easy as during basic math calculations, involving how many features have been implemented, compared to how many need to be implemented. One major disadvantage of FDD is the heavy reliance on the lead programmer, because it is centered on functionality, the lead programmer often acts as the lead designer, mentor and coordinator, while also performing his duties as head programmer [1].

We did not use the crystal methodology because it emphazing on people, interaction and communication. This indicates that it is more suitable for larger projects i.e. projects that require a project team of over 20 people and considering that weare only two, it was not the most appropriate of methodologies. The Dynamic Systems Development Methodology (DSDM) is another branch of agile development methodology that played a big role in our project. This is because the DSDM focuses on building just enough at each stage to progress to the next stage. This methodology is very efficient because it so efficiently handles changing business requirements.The Lean development methodology would also not have been as effective as much of its emphasis is placed on cutting cost, waste and effort. This is another system that would be much more appropriate in large, industrial sized projects, but would not be very useful in small projects like SafeShop.

# CHAPTER FOUR

1. **CONCLUSION AND FUTURE WORK**

## CONCLUSION

In conclusion, the system is a web application that will help members of the AUN community to advertise their products and find buyers. The system would also enable potential buyers to find potential sellers with goods of interest. The system is also important for those who are undecided about the price to set their products can put the products up for auctioning. Buyers would be able to make bids on biddable products.

There was a lot to take away from this project, one of the things we learned was the importance of planning. While there were hiccups along the way, we were able to scale through them because we had enough time to maneuver. It was refreshing to finally create, manage and execute a project on our own. Though we did the same thing in CIE 321 – IT Project Management, it was not quite the same. One of the factors that contributed to the stress of this project was the workload, not from SDP, but from other courses that we were taking concurrently. Time management had be extremely spot on else we would not have meet as much dealines as we did. Scheduled meetings were difficult enough to arrange, talk less of impromptu meetings that popped up at unexected times.

We also sharpened our skills as software engineers as writing code is one thing, but writing code to solve problems is another. Several of the functionality that we implemented in this project are functions we have never implemented before, so it was not like we could reuse some code, we

had to create code from scratch and that required not only thinking like software engineers, it also required thinking that problem solvers.

One more thing we learned is something we like to call the “Reality of Implementation”. Though its obvious that implementation is always more difficult than the documents suggest, we discovered that there are certain problems with implementation that we cannot foresee, we can only discover them while actually implementing the software. These valuable lessons can only be gotten through practical and we are greatly that we were given the opportunity to discover these problems ourselves and get our feet wet.

## FUTURE IMPLEMENTATIONS

Some functionalities and enhancements are expected to be part of future realeases. One of these would be adding a profile picture for the users, buyers and sellers. Secondly, we plan to create a mobile app that will be accessible irrespective of the platform (Android and iOS) so as to enable those that would rather access it from a mobile phone, though it will be accessible from a mobile browser. We also plan to minimise restrictions on the system to accomodate non-members of the AUN community for it bo used not only in Aun, but in Yola and other parts of Adamawa State. We plan to work on a voice search functionality and also to integrate it with the official AUN domain i.e (\*.aun.edu.ng).

We planned to implement a pay-with-paypal payment method during the course of this project, but we did not take into consideration that products being paid for should already be on the PayPal website. This roadblock hindered us from going through the the PayPal payment plan. We plan to fix this issue and implement it in future releases also.

# References

|  |  |
| --- | --- |
| [1] | Wikidot.com, “FDD - Agile - Feature Driven Development,” Wikidot.com, 20 August 2011. [Online]. Available: [http://sliitmscfdd.wikidot.com/introduction-to-fdd.](http://sliitmscfdd.wikidot.com/introduction-to-fdd) [Acedido em 30 April 2018]. |
| [2] | F. T. Zuhora, “The Beginner Guide to setup a Multivendor eCommerce Website,” 12 September 2017. [Online]. Available: https:/[/www.theme](http://www.themexpert.com/blog/the-beginner-guide-)x[pert.com/blog/the-beginner-guide-](http://www.themexpert.com/blog/the-beginner-guide-) to-setup-a-multivendor-ecommerce-website. [Acedido em 26 January 2018]. |
| [3] | N. Agarwal, “How To Build A Multi Vendor eCommerce Website,” 11 January 2018. [Online]. Available: https:/[/www.inkthemes.com/how](http://www.inkthemes.com/how-to-build-a-multi-vendor-ecommerce-)-[to-build-a-multi-vendor-ecommerce-](http://www.inkthemes.com/how-to-build-a-multi-vendor-ecommerce-) website/. [Acedido em 26 January 2018]. |
| [4] | M. Ginsburg, J. Gebauer e A. Segev, “Multi-Vendor Electronic Catalogs to Support,” em*Twelfth International Bled Electronic Commerce Conference*, Slovenia, 1999. |
| [5] | K. D. Gupta, “E-COMMERCE DATABASE DESIGN ENTITY RELATIONSHIP DIAGRAM,” 20 January 2011. [Online]. Available: https://kishordgupta.wordpress.com/2011/01/20/e-commerce-database-design-entity- relation-ship-diagram/. [Acedido em 17 February 2018]. |
| [6] | B. Williams, “Auction Data Model,” Dabase Answers, 2001. [Online]. Available: [http://www.databaseanswers.org/data\_models/auction/index.htm.](http://www.databaseanswers.org/data_models/auction/index.htm) [Acedido em 17 Februaru 2018]. |
| [7] | Microsoft, “Visual Studio Library,” Microsoft, 2018. [Online]. Available: https://msdn.microsoft.com/en-us/library/jj159353.aspx. |
| [8] | Microsoft, “The Official Forums for Microsoft ASP.NET,” Neudesic, LLC, 2018. [Online]. Available: https://forums.asp.net/. |
| [9] | I. Spaanjaars, Beginning ASP.NET 4.5: in C# and VB, Indiana: John Wiley & Sons, Inc., 2013. |
| [10] | A. Stellman e J. Greene, Head First C#, 2nd ed., Sebastopol: O’Reilly Media, Inc., 2010. |
| [11] | D. Esposito, Programming Microsoft ASP.NET 4, Washington: Microsoft Press, 2011. |
| [12] | B. Robart, thenewboston, 03 2016. [Online]. Available: https:/[/www](http://www.youtube.com/watch?v=qgGIqRFvFFk&list=PL6gx4Cwl9DGBlmzzFcLgDhKT).[youtube.com/watch?v=qgGIqRFvFFk&list=PL6gx4Cwl9DGBlmzzFcLgDhKT](http://www.youtube.com/watch?v=qgGIqRFvFFk&list=PL6gx4Cwl9DGBlmzzFcLgDhKT) TfNLfX1IK. [Acedido em 07 08 2016]. |
| [13] | O. E. I. a. C. A. O. Daramola, “Implementation of an intelligent course advisory expert system,” *Internation Journal of Advanceed Research in Artificial Intelligence,* vol. 3, nº 5, pp. 6-12, 2014. |
| [14] | N. W. a. F. K. Kamoun, “A decision-tree-based system for student academic advising and planning in information systems progrtammes,” *International Journal of Business Information Systems,* vol. 5, nº 1, pp. 1-18, 2010. |
| [15] | Djangoproject, “Djangoproject,” Djangoproject, 03 12 2015. [Online]. Available:https://docs.djangoproject.com/en/1.10/. [Acedido em 06 08 2016]. |

# Appendix

### Buyer Login Form Codes

using System;

using System.Collections.Generic; using System.Linq;

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls; using System.Data;

using System.Data.SqlClient; using System.Configuration;

public partial class realLogin : System.Web.UI.Page

{

SqlConnection con = new SqlConnection(ConfigurationManager.ConnectionStrings["AUNSafeShopDBConnectionString"

].ConnectionString);

protected void Page\_Load(object sender, EventArgs e)

{

if (Session["buyerId"] != null )

{

Response.Redirect("~/BuyerSide/WebShop.aspx");

}

}

protected void btnLogin\_Click(object sender, EventArgs e)

{

con.Open();

string sData = txtPassword.Text; string encoded;

byte[] encData\_byte = new byte[sData.Length];

encData\_byte = System.Text.Encoding.UTF8.GetBytes(sData); string encodedData = Convert.ToBase64String(encData\_byte); encoded = encodedData.ToString();

try

{

SqlCommand cmd = con.CreateCommand(); cmd.CommandType = CommandType.Text;

cmd.CommandText = "select \* from Buyer where Email='" + txtEmail.Text + "' and Password='" + encoded + "'";

cmd.ExecuteNonQuery(); DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd); da.Fill(dt);

int tot = Convert.ToInt32(dt.Rows.Count.ToString());

if (tot == 1)

{

foreach (DataRow dr in dt.Rows)

{

Session["buyerId"] = dr["BuyerId"].ToString(); Session["buyerName"] = dr["FirstName"].ToString();

}

if (Session["currentPage"] == null)

{

Response.Redirect("~/BuyerSide/WebShop.aspx");

}

else

{

Response.Redirect(Convert.ToString(Session["currentPage"]));

}

}

else

{

litStatus.Text = "Wrong email or password";

}

}

catch (Exception ex)

{

litStatus.Text = ex.Message;

}

con.Close();

}

}

### Seller Login Form Codes

using System;

using System.Collections.Generic; using System.Linq;

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls; using System.Data;

using System.Data.SqlClient; using System.Configuration;

public partial class Seller\_Side\_Account\_SellerLogin : System.Web.UI.Page

{

SqlConnection con = new SqlConnection(ConfigurationManager.ConnectionStrings["AUNSafeShopDBConnectionString"

].ConnectionString);

protected void Page\_Load(object sender, EventArgs e)

{

}

protected void btnLogin\_Click(object sender, EventArgs e)

{

con.Open();

string sData = txtPassword.Text; string encoded;

byte[] encData\_byte = new byte[sData.Length];

encData\_byte = System.Text.Encoding.UTF8.GetBytes(sData); string encodedData = Convert.ToBase64String(encData\_byte); encoded = encodedData.ToString();

try

{

SqlCommand cmd = con.CreateCommand(); cmd.CommandType = CommandType.Text;

cmd.CommandText = "select \* from Seller where Email='" + txtEmail.Text + "' and Password='" + encoded + "'";

cmd.ExecuteNonQuery(); DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd); da.Fill(dt);

int tot = Convert.ToInt32(dt.Rows.Count.ToString());

if (tot == 1)

{

foreach (DataRow dr in dt.Rows)

{

Session["sellerId"] = dr["SellerId"].ToString(); Session["sellerName"] = dr["FirstName"].ToString();

}

Response.Redirect("~/SellerSide/Dashboard.aspx");

}

else

{

litStatus.Text = "Wrong email or password";

}

}

catch (Exception ex)

{

litStatus.Text = ex.Message;

}

finally

{

con.Close();

}

}

}

### Add Product Form Codes

using System;

using System.Collections.Generic; using System.Linq;

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls; using System.Data;

using System.Data.SqlClient; using System.Configuration;

public partial class Add\_Product : System.Web.UI.Page

{

SqlConnection con = new SqlConnection(ConfigurationManager.ConnectionStrings["AUNSafeShopDBConnectionString"

].ConnectionString);

protected void Page\_Load(object sender, EventArgs e)

{

if (Session["sellerId"] == null)

{

Response.Redirect("~/SellerSide/Account/SellerLogin.aspx");

}

}

protected void submitProduct\_Click(object sender, EventArgs e)

{

con.Open();

string sellerId = Session["sellerId"].ToString();

int prTypeId = Convert.ToInt32(dropdownCategories.SelectedValue); int productId;

string generatedName, generatedName1, imageLocation, imageLocation1;

try

{

SqlCommand cmd = con.CreateCommand(); cmd.CommandType = CommandType.Text;

cmd.CommandText = "insert into Product (ProductName, Price, Description, Quantity, ProductTypeId, SellerId) values('" + productName.Text + "', '" + price.Text + "', '" + productDescription.Text + "', '" + quantity.Text + "', '" + prTypeId + "', '" + sellerId + "'); select scope\_identity();";

productId = Convert.ToInt32(cmd.ExecuteScalar());

generatedName = PictureName.GetRandomPassword(10).ToString();

fuPicture1.SaveAs(Request.PhysicalApplicationPath + "./Images/" + generatedName + fuPicture1.FileName.ToString());

imageLocation = "Images/" + generatedName + fuPicture1.FileName.ToString();

if (fuPicture2.HasFile == true)

{

generatedName1 = PictureName.GetRandomPassword(10).ToString(); fuPicture2.SaveAs(Request.PhysicalApplicationPath + "./Images/" + generatedName +

fuPicture2.FileName.ToString());

imageLocation1 = "Images/" + generatedName + fuPicture2.FileName.ToString();

SqlCommand cmd2 = con.CreateCommand(); cmd2.CommandType = CommandType.Text; cmd2.CommandText =

"insert into Picture (ProductId, Image) values('" + productId + "', '" + imageLocation.ToString() + "'); insert into Picture (ProductId, Image) values('" + productId + "', '" + imageLocation1.ToString() + "')";

cmd2.ExecuteNonQuery();

}

else

{

SqlCommand cmd2 = con.CreateCommand(); cmd2.CommandType = CommandType.Text;

cmd2.CommandText = "insert into Picture (ProductId, Image) values('" + productId + "', '" + imageLocation.ToString() + "')";

cmd2.ExecuteNonQuery();

}

Response.Write(@"<script language='javascript'>alert('Inserted Successfully');</script>");

}

catch (Exception ex)

{

litStatus.Text = ex.Message;

}

}

}

### Auction Product Form Codes

using System;

using System.Collections.Generic; using System.Linq;

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls; using System.Data;

using System.Data.SqlClient;

using System.Configuration;

public partial class Auction\_Product : System.Web.UI.Page

{

SqlConnection con = new SqlConnection(ConfigurationManager.ConnectionStrings["AUNSafeShopDBConnectionString"

].ConnectionString);

protected void Page\_Load(object sender, EventArgs e)

{

if (Session["sellerId"] == null)

{

Response.Redirect("~/SellerSide/Account/SellerLogin.aspx");

}

}

protected void auctionSubmit\_Click(object sender, EventArgs e)

{

DateTime bidStartDate = DateTime.Now; string sellerId = Session["sellerId"].ToString();

int prTypeId = Convert.ToInt32(dropdownCategories.SelectedValue); int productId;

string generatedName, generatedName1, imageLocation, imageLocation1; DateTime bidEndDate = Convert.ToDateTime(txtbidEndDate.Text); con.Open();

try

{

SqlCommand cmd = con.CreateCommand(); cmd.CommandType = CommandType.Text;

cmd.CommandText = "insert into Product (ProductName, StartingBid, BidStartDate, BidEndDate, Description, ProductTypeId, SellerId) values('" + productName.Text + "', '" + auctionStartingPrice.Text + "', '" + bidStartDate + "', '" + bidEndDate + "', '" + productDescription.Text + "', '" + prTypeId + "', '" + sellerId + "'); select scope\_identity();";

productId = Convert.ToInt32(cmd.ExecuteScalar());

generatedName = PictureName.GetRandomPassword(10).ToString(); fuPicture1.SaveAs(Request.PhysicalApplicationPath + "./Images/" + generatedName +

fuPicture1.FileName.ToString());

imageLocation = "Images/" + generatedName + fuPicture1.FileName.ToString();

if (fuPicture2.HasFile == true)

{

generatedName1 = PictureName.GetRandomPassword(10).ToString(); fuPicture2.SaveAs(Request.PhysicalApplicationPath + "./Images/" + generatedName +

fuPicture2.FileName.ToString());

imageLocation1 = "Images/" + generatedName + fuPicture2.FileName.ToString();

SqlCommand cmd2 = con.CreateCommand(); cmd2.CommandType = CommandType.Text; cmd2.CommandText =

"insert into Picture (ProductId, Image) values('" + productId + "', '" + imageLocation.ToString() + "'); insert into Picture (ProductId, Image) values('" + productId + "', '" + imageLocation1.ToString() + "')";

cmd2.ExecuteNonQuery();

}

else

{

SqlCommand cmd2 = con.CreateCommand(); cmd2.CommandType = CommandType.Text;

cmd2.CommandText = "insert into Picture (ProductId, Image) values('" + productId + "', '" + imageLocation.ToString() + "')";

cmd2.ExecuteNonQuery();

}

Response.Write(@"<script language='javascript'>alert('Product Added Successfully');</script>");

}

catch (Exception ex)

{

litStatus.Text = ex.Message;

}

}

}

### WebShop Form Codes

using System;

using System.Collections.Generic; using System.Linq;

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls; using System.Data;

using System.Data.SqlClient; using System.Configuration;

public partial class WebShop\_WebShop : System.Web.UI.Page

{

SqlConnection con = new SqlConnection(ConfigurationManager.ConnectionStrings["AUNSafeShopDBConnectionString"

].ConnectionString);

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

if (Request.QueryString["search"] != null)

{

FillSearchListView();

}

else

{

FillListView();

}

}

}

private void FillSearchListView()

{

string search = Request.QueryString["search"].ToString(); con.Open();

SqlCommand cmd = con.CreateCommand(); cmd.CommandType = CommandType.Text;

cmd.CommandText = "SELECT Product.ProductId, Product.ProductName, Product.Price, Product.StartingBid, Picture.PictureId, Picture.Image FROM Product INNER JOIN Picture ON Product.ProductId = Picture.ProductId WHERE Product.ProductName like('%" + search + "%') and (Picture.PictureId IN (SELECT MIN(PictureId) AS Expr1 FROM Picture AS Picture\_1 GROUP BY ProductId)) ORDER BY NEWID() ";

cmd.ExecuteNonQuery(); DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd); da.Fill(dt);

ListView1.DataSource = dt; ListView1.DataBind();

if (dt.Rows.Count == 0)

{

Response.Write(@"<script language='javascript'>alert('No items found');</script>");

}

con.Close();

}

private void FillListView()

{

con.Open();

SqlCommand cmd = con.CreateCommand(); cmd.CommandType = CommandType.Text;

cmd.CommandText = "SELECT Product.ProductId, Product.ProductName, Product.Price, Product.StartingBid, Picture.PictureId, Picture.Image FROM Product INNER JOIN Picture ON Product.ProductId = Picture.ProductId WHERE (Picture.PictureId IN (SELECT MIN(PictureId) AS Expr1 FROM Picture AS Picture\_1 GROUP BY ProductId)) ORDER BY NEWID() ";

cmd.ExecuteNonQuery(); DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd); da.Fill(dt);

ListView1.DataSource = dt;

ListView1.DataBind();

con.Close();

}

protected void ListView1\_ItemDataBound(object sender, ListViewItemEventArgs e)

{

if (e.Item.ItemType == ListViewItemType.DataItem)

{

Button buy = new Button(); Button bid = new Button();

buy = (Button)e.Item.FindControl("btnBuy"); bid = (Button)e.Item.FindControl("btnBid");

if (!DataBinder.Eval(e.Item.DataItem, "Price").ToString().Equals(""))

{

buy.Visible = true; bid.Visible = false;

}

else

{

buy.Visible = false; bid.Visible = true;

}

}

}

protected void ListView1\_ItemCommand(object sender, ListViewCommandEventArgs e)

{

if (e.CommandName == "buy")

{

Response.Redirect("~/BuyerSide/Description.aspx?id=" + e.CommandArgument);

}

else if (e.CommandName == "bid")

{

Response.Redirect("~/BuyerSide/Bidding.aspx?id=" + e.CommandArgument);

}

}

}

### Bidding Form Codes

using System;

using System.Collections.Generic; using System.Linq;

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls; using System.Data;

using System.Data.SqlClient; using System.Configuration;

public partial class Bidding\_Bidding : System.Web.UI.Page

{

SqlConnection con = new SqlConnection(ConfigurationManager.ConnectionStrings["AUNSafeShopDBConnectionString"

].ConnectionString);

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

if (Session["buyerId"] != null)

{

FillPage();

}

else

{

Response.Redirect("~/realLogin.aspx");

}

}

}

protected void makeBid\_Click(object sender, EventArgs e)

{

if (lblTimer.Text != "Bidding is Closed")

{

int buyerId = Convert.ToInt32(Session["buyerId"]);

int productId = Convert.ToInt32(Request.QueryString["id"]); con.Open();

SqlCommand cmd1 = con.CreateCommand(); cmd1.CommandType = CommandType.Text;

cmd1.CommandText = "select \* from Bids where BuyerId ='" + buyerId + "' and ProductId = '" + productId + "' ";

cmd1.ExecuteNonQuery();

DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd1); da.Fill(dt);

int tot = Convert.ToInt32(dt.Rows.Count.ToString());

if (tot > 0)

{

SqlCommand cmd2 = con.CreateCommand(); cmd2.CommandType = CommandType.Text;

cmd2.CommandText = "update Bids set BidAmount ='" + txtUserBid.Text + "' where BuyerId = '" + buyerId + "' and ProductId = '" + productId + "' ";

cmd2.ExecuteNonQuery(); FillGridView();

}

else

{

SqlCommand cmd3 = con.CreateCommand(); cmd3.CommandType = CommandType.Text;

cmd3.CommandText = " insert into Bids (ProductId, BuyerId, BidAmount) values ('" + productId + "', '" + buyerId + "', '" + txtUserBid.Text + "') ";

cmd3.ExecuteNonQuery();

}

FillGridView();

}

}

private void FillPage()

{

if (Request.QueryString["id"] != null)

{

int id = Convert.ToInt32(Request.QueryString["id"].ToString()); con.Open();

SqlCommand cmd = new SqlCommand("select Product.ProductName, Product.Description, Product.StartingBid, Product.BidEndDate, Picture.Image, Seller.FirstName, Seller.LastName, Seller.PhoneNo, Seller.Email from Picture INNER JOIN Product ON Picture.ProductId = Product.ProductId INNER JOIN Seller ON Product.SellerId = Seller.SellerId where Product.ProductId = '" + id + "'", con);

DataTable dt = new DataTable();

SqlDataAdapter sqlDa = new SqlDataAdapter(cmd);

sqlDa.Fill(dt);

if (dt.Rows.Count > 0)

{

imgProduct1.ImageUrl = "~/" + dt.Rows[0]["Image"].ToString(); lblName.Text = dt.Rows[0]["ProductName"].ToString(); lblBidEndDate.Text = dt.Rows[0]["BidEndDate"].ToString(); Session["EndDate"] = dt.Rows[0]["BidEndDate"].ToString(); lblPrice.Text = dt.Rows[0]["StartingBid"].ToString(); lblDescription.Text = dt.Rows[0]["Description"].ToString(); lblSellerName.Text = dt.Rows[0]["FirstName"].ToString() + " " +

dt.Rows[0]["LastName"].ToString();

lblEmail.Text = dt.Rows[0]["Email"].ToString(); lblPhoneNo.Text = dt.Rows[0]["PhoneNo"].ToString(); Pic2.Visible = false;

DateTime end = Convert.ToDateTime(dt.Rows[0]["BidEndDate"].ToString()); DateTime now = DateTime.Now;

TimeSpan timeLeft = end.Subtract(now);

if (timeLeft.Days <= 0 && timeLeft.Hours <= 0 && timeLeft.Minutes <= 0 && timeLeft.Seconds <= 0)

{

txtUserBid.Visible = false; makeBid.Visible = false;

cmd.CommandText = "UPDATE Bids SET Bids.WinStatus = 'Winner' WHERE Bids.BidAmount = (SELECT MAX(BidAmount) FROM Bids) and Bids.ProductId = '" + id + "' ";

cmd.ExecuteNonQuery(); FillWinnerGridView();

}

else

{

FillGridView();

}

}

else if (dt.Rows.Count > 1)

{

imgProduct1.ImageUrl = "~/" + dt.Rows[0]["Image"].ToString(); imgProduct2.ImageUrl = "~/" + dt.Rows[1]["Image"].ToString(); lblName.Text = dt.Rows[0]["ProductName"].ToString(); lblBidEndDate.Text = dt.Rows[0]["BidEndDate"].ToString(); Session["EndDate"] = dt.Rows[0]["BidEndDate"].ToString(); lblPrice.Text = dt.Rows[0]["StartingBid"].ToString(); lblDescription.Text = dt.Rows[0]["Description"].ToString();

DateTime end = Convert.ToDateTime(Session["EndDate"].ToString()); DateTime now = DateTime.Now;

TimeSpan timeLeft = end.Subtract(now);

lblTimer.Text = timeLeft.Days.ToString() + " Days " + timeLeft.Hours.ToString() + " Hours " + timeLeft.Minutes.ToString() + " Minutes " + timeLeft.Seconds.ToString() + " Second";

if (timeLeft.Days <= 0 && timeLeft.Hours <= 0 && timeLeft.Minutes <= 0 && timeLeft.Seconds <= 0)

{

lblTimer.Text = "Bidding is Closed"; txtUserBid.Visible = false; makeBid.Visible = false;

cmd.CommandText = "UPDATE Bids SET Bids.WinStatus = 'Winner' WHERE Bids.BidAmount = (SELECT MAX(BidAmount) FROM Bids) and Bids.ProductId = '" + id + "' ";

cmd.ExecuteNonQuery();

FillWinnerGridView();

}

else

{

FillGridView();

}

}

con.Close();

}

else

{

Response.Redirect("~/Buyerside/Webshop.aspx");

}

}

protected void Timer1\_Tick(object sender, EventArgs e)

{

DateTime end = Convert.ToDateTime(Session["EndDate"].ToString()); DateTime now = DateTime.Now;

TimeSpan timeLeft = end.Subtract(now);

lblTimer.Text = timeLeft.Days.ToString() + " Days " + timeLeft.Hours.ToString() + " Hours " + timeLeft.Minutes.ToString() + " Minutes " + timeLeft.Seconds.ToString() + " Second";

if (timeLeft.Days <= 0 && timeLeft.Hours <= 0 && timeLeft.Minutes <= 0 && timeLeft.Seconds <= 0)

{

lblTimer.Text = "Bidding is Closed"; Timer1.Enabled = false;

}

}

private void BidButton()

{

if (Session["buyerId"] == null)

{

Session["currentPage"] = Request.Url; Response.Redirect("~/realLogin.aspx");

}

else

{

DateTime end = Convert.ToDateTime(Session["EndDate"].ToString()); DateTime now = DateTime.Now;

TimeSpan timeLeft = end.Subtract(now);

if (lblTimer.Text != "Bidding is Closed")

{

int buyerId = Convert.ToInt32(Session["buyerId"]);

int productId = Convert.ToInt32(Request.QueryString["id"]); con.Open();

SqlCommand cmd1 = con.CreateCommand(); cmd1.CommandType = CommandType.Text;

cmd1.CommandText = "select \* from Bids where BuyerId ='" + buyerId + "' and ProductId = '" + productId + "' ";

cmd1.ExecuteNonQuery();

DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd1); da.Fill(dt);

int tot = Convert.ToInt32(dt.Rows.Count.ToString());

if (tot > 0)

{

SqlCommand cmd2 = con.CreateCommand(); cmd2.CommandType = CommandType.Text;

cmd2.CommandText = "update Bids set BidAmount ='" + txtUserBid.Text + "' where BuyerId = '" + buyerId + "' and ProductId = '" + productId + "' ";

cmd2.ExecuteNonQuery();

}

else

{

SqlCommand cmd3 = con.CreateCommand(); cmd3.CommandType = CommandType.Text;

cmd3.CommandText = " insert into Bids (ProductId, BuyerId, BidAmount) values ('" + productId + "', '" + buyerId + "', '" + txtUserBid.Text + "') ";

cmd3.ExecuteNonQuery();

}

FillGridView();

}

}

}

private void FillGridView()

{

int productId = Convert.ToInt32(Request.QueryString["id"].ToString());

SqlCommand cmd = con.CreateCommand(); cmd.CommandType = CommandType.Text;

cmd.CommandText = "SELECT Buyer.FirstName AS [First Name], Buyer.LastName AS [Last Name], Bids.BidAmount AS [Bid Amount], Bids.WinStatus AS [Win Status] FROM Bids INNER JOIN Buyer ON Bids.BuyerId = Buyer.BuyerId and Bids.ProductId = '" + productId + "' ORDER BY CAST(Bids.BidAmount as int) desc ";

cmd.ExecuteNonQuery(); DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd); da.Fill(dt);

GridView1.DataSource = dt; GridView1.DataBind();

}

private void FillWinnerGridView()

{

int productId = Convert.ToInt32(Request.QueryString["id"].ToString()); string name;

SqlCommand cmd = con.CreateCommand(); cmd.CommandType = CommandType.Text;

cmd.CommandText = "SELECT Buyer.FirstName AS [First Name], Buyer.LastName AS [Last Name], Bids.BidAmount AS [Bid Amount], Bids.WinStatus AS [Win Status] FROM Bids INNER JOIN Buyer ON Bids.BuyerId = Buyer.BuyerId and Bids.ProductId = '" + productId + "' where Bids.WinStatus = 'Winner' ORDER BY CAST(Bids.BidAmount as int) desc ";

cmd.ExecuteNonQuery(); DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd); da.Fill(dt);

if (dt.Rows.Count == 1)

{

name = dt.Rows[0]["First Name"].ToString(); if (Session["buyerName"].ToString() == name)

{

lbl1.Visible = true; lbl2.Visible = true;

}

}

GridView1.DataSource = dt; GridView1.DataBind();

}

}

### Description Form Codes

using System;

using System.Collections.Generic; using System.Linq;

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls; using System.Data;

using System.Data.SqlClient; using System.Configuration;

public partial class Buyer\_Side\_Description : System.Web.UI.Page

{

SqlConnection con = new SqlConnection(ConfigurationManager.ConnectionStrings["AUNSafeShopDBConnectionString"

].ConnectionString);

protected void Page\_Load(object sender, EventArgs e)

{

FillPage();

}

private void FillPage()

{

if (Request.QueryString["id"] != null)

{

int id = Convert.ToInt32(Request.QueryString["id"].ToString()); con.Open();

SqlCommand cmd = new SqlCommand("select Picture.Image, Product.ProductName, Product.Description, Product.Price, Product.Quantity, Seller.FirstName, Seller.LastName, Seller.PhoneNo, Seller.Email from Picture INNER JOIN Product ON Picture.ProductId = Product.ProductId INNER JOIN Seller ON Product.SellerId = Seller.SellerId where Product.ProductId = '" + id + "'", con);

DataTable dt = new DataTable();

SqlDataAdapter sqlDa = new SqlDataAdapter(cmd); sqlDa.Fill(dt);

if (dt.Rows.Count == 1)

{

imgProduct1.ImageUrl = "~/" + dt.Rows[0]["Image"].ToString(); lblTitle.Text = dt.Rows[0]["ProductName"].ToString();

Session["productName"] = dt.Rows[0]["ProductName"].ToString(); lblDescription.Text = dt.Rows[0]["Description"].ToString(); lblPrice.Text = dt.Rows[0]["Price"].ToString();

Session["price"] = dt.Rows[0]["Price"].ToString(); lblItemNr.Text = dt.Rows[0]["Quantity"].ToString();

Session["quantity"] = dt.Rows[0]["Quantity"].ToString(); lblSellerName.Text = dt.Rows[0]["FirstName"].ToString(); secondImage.Visible = false;

imgProduct2.Visible = false;

}

else if (dt.Rows.Count > 1)

{

imgProduct1.ImageUrl = "~/" + dt.Rows[0]["Image"].ToString(); imgProduct2.ImageUrl = "~/" + dt.Rows[1]["Image"].ToString(); lblTitle.Text = dt.Rows[0]["ProductName"].ToString(); Session["productName"] = dt.Rows[0]["ProductName"].ToString(); lblDescription.Text = dt.Rows[0]["Description"].ToString(); lblPrice.Text = dt.Rows[0]["Price"].ToString();

Session["price"] = dt.Rows[0]["Price"].ToString(); lblItemNr.Text = dt.Rows[0]["Quantity"].ToString(); lblSellerName.Text = dt.Rows[0]["FirstName"].ToString() + " " +

dt.Rows[0]["LastName"].ToString();

lblEmail.Text = dt.Rows[0]["Email"].ToString(); lblPhoneNo.Text = dt.Rows[0]["PhoneNo"].ToString(); Session["quantity"] = dt.Rows[0]["Quantity"].ToString();

}

if (Convert.ToInt32(Session["quantity"]) < 1)

{

btnAdd.Enabled = false;

}

con.Close();

}

else

{

Response.Redirect("~/Buyerside/Webshop.aspx");

}

}

protected void btnAdd\_Click(object sender, EventArgs e)

{

if (Session["buyerId"] == null)

{

Session["currentPage"] = Request.Url; Response.Redirect("~/realLogin.aspx");

}

else

{

int buyerId = Convert.ToInt32(Session["buyerId"].ToString()); DateTime orderDate = DateTime.Now;

string orderStatus = "Order en route";

int productId = Convert.ToInt32(Request.QueryString["id"].ToString()); int buyerQuantity = Convert.ToInt32(txtQuantity.Text);

int quantityRemaining = Convert.ToInt32(Session["quantity"].ToString()); int orderId;

con.Open();

SqlCommand cmd = new SqlCommand("select Address from Buyer where BuyerId = '"

+ buyerId + "'", con);

DataTable dt = new DataTable();

SqlDataAdapter sqlDa = new SqlDataAdapter(cmd); sqlDa.Fill(dt);

string address = dt.Rows[0]["Address"].ToString();

if (quantityRemaining >= buyerQuantity && buyerQuantity > 0)

{

SqlCommand cmd1 = con.CreateCommand(); cmd1.CommandType = CommandType.Text;

cmd1.CommandText = "insert into [Order] (BuyerId, OrderDate, OrderStatus, DeliveryAddress) values('" + buyerId + "', '" + orderDate + "', '" + orderStatus + "', '" + address + "'); select scope\_identity();";

orderId = Convert.ToInt32(cmd1.ExecuteScalar());

SqlCommand cmd2 = con.CreateCommand(); cmd2.CommandType = CommandType.Text;

cmd2.CommandText = "insert into OrderDetails (ProductId, OrderId, Quantity) values('" + productId + "', '" + orderId + "', '" + txtQuantity.Text.ToString() + "') ";

cmd2.ExecuteNonQuery();

lblResult.Text = "Inserted to cart Successfully"; Response.Write(@"<script language='javascript'>alert('Inserted to cart

Successfully');</script>");

}

else

{

if (buyerQuantity <= 0)

{

lblResult.Text = "Quantity cannot be 0 or less";

Response.Write(@"<script language='javascript'>alert('Quantity cannot be 0 or

less');</script>");

}

else

{

lblResult.Text = "Quantity has to be less than the amount available";

Response.Write(@"<script language='javascript'>alert('Quantity has to be less than the amount available');</script>");

}

}

con.Close(); FillPage();

}

}

}

### Shopping Cart Form Codes

using System;

using System.Collections.Generic; using System.Linq;

using System.Web; using System.Web.UI;

using System.Web.UI.WebControls; using System.Data;

using System.Data.SqlClient; using System.Configuration;

public partial class BuyerSide\_ShoppingCart : System.Web.UI.Page

{

SqlConnection con = new SqlConnection(ConfigurationManager.ConnectionStrings["AUNSafeShopDBConnectionString"

].ConnectionString);

protected void Page\_Load(object sender, EventArgs e)

{

if (Session["buyerId"] == null)

{

Response.Redirect("~/realLogin.aspx");

}

else

{

if (!IsPostBack)

{

FillCartPage();

}

}

}

protected void ListView1\_ItemCanceling(object sender, ListViewCancelEventArgs e)

{

ListView1.EditIndex = -1; FillCartPage();

}

protected void ListView1\_ItemDeleting(object sender, ListViewDeleteEventArgs e)

{

int orderId = int.Parse(ListView1.DataKeys[e.ItemIndex].Values[2].ToString()); con.Open();

SqlCommand cmd = con.CreateCommand(); cmd.CommandType = CommandType.Text;

cmd.CommandText = "delete from [Order] FROM [Order] INNER JOIN OrderDetails ON [Order].OrderId = OrderDetails.OrderId where [Order].OrderId = '" + orderId + "' ";

cmd.ExecuteNonQuery(); con.Close(); FillCartPage();

}

protected void ListView1\_ItemEditing(object sender, ListViewEditEventArgs e)

{

ListView1.EditIndex = e.NewEditIndex; FillCartPage();

}

protected void ListView1\_ItemUpdating(object sender, ListViewUpdateEventArgs e)

{

int buyerId = Convert.ToInt32(Session["buyerId"].ToString()); ListViewDataItem item = ListView1.Items[e.ItemIndex];

int productId = int.Parse(ListView1.DataKeys[e.ItemIndex].Values[0].ToString()); TextBox quantity = (TextBox)item.FindControl("txtQuantity");

TextBox address = (TextBox)item.FindControl("txtDeliveryAddress");

string quantityDatabase = "select Product.Quantity from Product where Product.ProductId = '" + productId + "' ";

string updateOrder = "UPDATE [Order] SET [Order].DeliveryAddress = '" + address.Text

+ "' FROM [Order] INNER JOIN OrderDetails ON [Order].OrderId = OrderDetails.OrderId and [Order].BuyerId = '" + buyerId + "' and OrderDetails.ProductId = '" + productId + "' INNER JOIN Product ON OrderDetails.ProductId = Product.ProductId ";

string updateOrderDetails = "UPDATE OrderDetails SET OrderDetails.Quantity = '" + quantity.Text + "' FROM [Order] INNER JOIN OrderDetails ON [Order].OrderId = OrderDetails.OrderId and [Order].BuyerId = '" + buyerId + "' and OrderDetails.ProductId = '" + productId + "' INNER JOIN Product ON OrderDetails.ProductId = Product.ProductId ";

int quantityRemaining; con.Open();

SqlCommand cmd1 = con.CreateCommand(); cmd1.CommandType = CommandType.Text; cmd1.CommandText = quantityDatabase; cmd1.ExecuteNonQuery();

DataTable dTable = new DataTable(); SqlDataAdapter dAdap = new SqlDataAdapter(cmd1); dAdap.Fill(dTable);

quantityRemaining = Convert.ToInt32(dTable.Rows[0]["Quantity"].ToString()); con.Close();

if (Convert.ToInt32(quantity.Text) < quantityRemaining)

{

con.Open();

SqlTransaction trans = con.BeginTransaction();

try

{

SqlCommand cmd = con.CreateCommand(); cmd.Transaction = trans; cmd.CommandType = CommandType.Text; cmd.CommandText = updateOrder; cmd.ExecuteNonQuery(); cmd.CommandText = updateOrderDetails; cmd.ExecuteNonQuery();

trans.Commit();

}

catch (Exception ex)

{

trans.Rollback(); litStatus.Text = ex.Message;

}

finally

{

con.Close(); ListView1.EditIndex = -1; FillCartPage();

}

}

else

{

ListView1.EditIndex = -1; FillCartPage();

Response.Write(@"<script language='javascript'>alert('We only have" + quantityRemaining + " items left.');</script>");

}

private void FillCartPage()

{

int buyerId = Convert.ToInt32(Session["buyerId"]); int total = 0;

con.Open();

SqlCommand cmd = con.CreateCommand(); cmd.CommandType = CommandType.Text;

cmd.CommandText = "SELECT Picture.Image, Product.ProductId, Product.ProductName, OrderDetails.OrderDetailsId, OrderDetails.Quantity, [Order].OrderId, [Order].DeliveryAddress, cast(OrderDetails.Quantity as int) \* cast(Product.Price as int) AS 'ItemTotal' FROM [Order] INNER JOIN OrderDetails ON [Order].OrderId = OrderDetails.OrderId and [Order].BuyerId = '" + buyerId + "' and [Order].PaymentType IS NULL INNER JOIN Product ON OrderDetails.ProductId = Product.ProductId INNER JOIN Picture ON Product.ProductId = Picture.ProductId WHERE (Picture.PictureId IN (SELECT MIN(PictureId) AS Expr1 FROM Picture AS Picture\_1 GROUP BY ProductId)) ";

cmd.ExecuteNonQuery(); DataTable dt = new DataTable();

SqlDataAdapter da = new SqlDataAdapter(cmd); da.Fill(dt);

foreach (DataRow dr in dt.Rows)

{

total += Convert.ToInt32(dr["ItemTotal"].ToString());

}

lblTotal.Text = total.ToString(); ListView1.DataSource = dt; ListView1.DataBind();

con.Close();

if (ListView1.Items.Count > 0)

{

btnContinue.Visible = true; btnPayOnDelivery.Visible = true; Tot.Visible = true;

or.Visible = true;

}

else

{

btnPayOnDelivery.Visible = false; btnContinue.Visible = false; Tot.Visible = false;

or.Visible = false;

}

}

protected void btnContinue\_Click(object sender, EventArgs e)

{

Response.Redirect("~/BuyerSide/WebShop.aspx");

protected void btnPayOnDelivery\_Click(object sender, EventArgs e)

{

string paymentType = "Pay on Delivery";

foreach (ListViewDataItem item in ListView1.Items)

{

int orderId = int.Parse(ListView1.DataKeys[item.DataItemIndex].Values[2].ToString()); int productId =

int.Parse(ListView1.DataKeys[item.DataItemIndex].Values[0].ToString()); Label quantity = item.FindControl("QuantityLabel") as Label;

int buyerQuantity = Convert.ToInt32(quantity.Text);

"' ";

con.Open();

SqlCommand com = con.CreateCommand(); com.CommandType = CommandType.Text;

com.CommandText = "select Quantity from Product where ProductId = '" + productId +

com.ExecuteNonQuery();

DataTable dataTable = new DataTable();

SqlDataAdapter dataAdapter = new SqlDataAdapter(com); dataAdapter.Fill(dataTable);

int productQuantity = Convert.ToInt32(dataTable.Rows[0]["Quantity"].ToString()); string quantityLeft = Convert.ToString(productQuantity - buyerQuantity);

if (productQuantity >= buyerQuantity && buyerQuantity > 0)

{

SqlTransaction trans = con.BeginTransaction();

try

{

SqlCommand cmd = con.CreateCommand(); cmd.Transaction = trans; cmd.CommandType = CommandType.Text;

cmd.CommandText = "update [Order] set [Order].PaymentType = '" + paymentType

+ "' where [Order].OrderId = '" + orderId + "' "; cmd.ExecuteNonQuery();

cmd.CommandText = "update Product set Quantity = '" + quantityLeft + "' where ProductId = '" + productId + "' ";

cmd.ExecuteNonQuery();

trans.Commit();

}

catch (Exception ex)

{

}

}

else

trans.Rollback(); litStatus.Text = ex.Message;

{

litStatus.Text = "We don't have that much in stock. Thank you for patronizing us and

we hope you will return when we have more in stock.";

}

con.Close();

}

litStatus.Text = "Successful"; FillCartPage();

}

}