PROJECT DESCRIPTION:

The Shopping cart application allows the user to login only using the Facebook login credentials. The authenticated users can be both sellers and buyers. When a user wants to sell a product, the Sell tab in the Application. The user will be giving the details like Item name, category, description, price and also the image which is saved in varbinary format in the database. The buyer will be displayed with the details of the products along with the seller detail like facebook url of the seller. When the user selects a product to add to the cart, it will be added to it. On the cart page, the user will be given three button: Checkout to continue with payment, continue shopping to view more products, or remove a product from cart. On clicking remove product, the product will be removed from both cart view and the table. On clicking checkout, the selected item details will be added to the seller product details table and order table and displayed to the user. The user will be redirected to the payment method which is Paypal. On successful payment, a mail will be sent to the buyer with the order details.

Packages used:

Facebook.7.0.6

Microsoft.Owin.3.1.0

Microsoft.AspNet.Identity.Owin.2.2.1

Microsoft.AspNet.Mvc.5.2.3

Microsoft.Owin.Secutiry.3.0.1

Controller code:

Selling a product:

//To sell a product

public ActionResult SellProduct()

{

SellProductDetail selldet = new SellProductDetail();

return View(selldet);

}

//post data from view form to the database

[HttpPost]

public ActionResult SellProduct(SellProductDetail sdet,HttpPostedFileBase image)

{

try

{

var db = new ShoppingAppFBEntities2();

sdet.ItemImage = new byte[image.ContentLength];

image.InputStream.Read(sdet.ItemImage, 0, image.ContentLength); //convert image to varbinary format in database

sdet.SellerID = Session["id"].ToString();

db.SellProductDetails.Add(sdet); //saves data to the database

db.SaveChanges();

}

catch (DbEntityValidationException dbEx)

{

foreach (var validationErrors in dbEx.EntityValidationErrors)

{

foreach (var validationError in validationErrors.ValidationErrors)

{

Trace.TraceInformation("Property: {0} Error: {1}",

validationError.PropertyName,

validationError.ErrorMessage);

}

}

}

return View(sdet);

}

Display products:

//displays products available for buying

public ActionResult BuyProduct()

{

var db = new ShoppingAppFBEntities2();

var item = (from d in db.SellProductDetails select d).ToList(); //lists the products available for buying from the database

return View(item);

}

Add to cart:

//add the selected item to cart table

public ActionResult AddtoCart(int id)

{

SellProductDetail prod = db.SellProductDetails.SingleOrDefault(p => p.ItemID == id);

var cartItem = db.CartDetails.SingleOrDefault(c => c.ItemID == id);

if (cartItem == null)

{

cartItem = new CartDetail //saves the selected item to cart table based on item id

{

ItemID = prod.ItemID,

Count = 1,

SellerID = Session["id"].ToString()

//SellerID = "1234"

};

db.CartDetails.Add(cartItem);

}

else

{

// If the item does exist in the cart,

// then add one to the quantity

cartItem.Count++;

db.Entry(cartItem).State = System.Data.Entity.EntityState.Modified;

}

db.SaveChanges();

return RedirectToAction("Viewcart", "Home", new { id = cartItem.ItemID }); //redirects to viewcart and display details of selected products

}

Checkout:

//Saves the confirmed items details to the order table

public ActionResult Checkout(int id, int tot)

{

// var result = (from d in db.CartDetails where d.ItemID == id select d).FirstOrDefault();

OrderDetail ord = new OrderDetail();

ord.ItemID = id;

ord.SellerID = Session["id"].ToString();

ord.BuyerID = Session["id"].ToString();

ord.Total = tot.ToString();

db.OrderDetails.Add(ord);

db.SaveChanges();

// return RedirectToAction(

return RedirectToAction("SellerDetails","Home",new { ItemID1 = ord.ItemID }); //saves the details of item and the buyer in the buyer details

}

Save ordered details to buyer table:

//saves details to buyer product details table

public ActionResult SellerDetails(int ItemID1)

{

SellProductDetail prod = db.SellProductDetails.FirstOrDefault(p => p.ItemID == ItemID1);

// Session["ItemID1"] = prod.ItemID.ToString();

OrderDetail odet = db.OrderDetails.FirstOrDefault(o => o.ItemID == prod.ItemID);

var bdet = db.BuyerProductDetails.FirstOrDefault(c => c.ItemID == ItemID1);

if (bdet == null)

{

bdet = new BuyerProductDetail

{

ItemID = odet.ItemID,

OrderID = odet.OrderID,

ItemName = prod.ItemName,

ItemCategory = prod.ItemCategory,

ItemDescription = prod.ItemDescription,

Total = odet.Total,

BuyerID = odet.BuyerID,

SellerID = odet.SellerID

};

db.BuyerProductDetails.Add(bdet);

db.SaveChanges();

Session["itemid"] = bdet.ItemID.ToString();

ViewBag.orderid = bdet.OrderID;

}

return RedirectToAction("Payment", "Home"); //redirects to the payment page

}

Payment Controller:

//Payment page

public ActionResult Payment()

{

return View();

}

Placing order successfully and email the details after payment:

//A mail wil be sent upon successful payment from admin to the buyer with the ordered product details

public ActionResult Success()

{

string id = Session["itemid"].ToString();

SmtpClient server = new SmtpClient("smtp.gmail.com");

server.Port = 587;

server.UseDefaultCredentials = false;

server.Credentials = new NetworkCredential("smadhu0893@gmail.com", "madhu@201103045");

server.EnableSsl = true;

MailMessage email = new MailMessage();

email.From = new MailAddress("smadhu0893@gmail.com");

var toaddress = Session["email"].ToString();

email.To.Add(toaddress);

List<BuyerProductDetail> bdet = (from st in db.BuyerProductDetails where st.ItemID.ToString()==id select st).ToList();

email.Subject = "Order Placed";

foreach (var item in bdet) //ordered items will be retrieved from the table and sent in mail

{

string Body = "Hi ," + "\n" + "Your order has been placed and the details are given below: " + "\n" +"Order ID:"+"\t"+item.OrderID+"\t" +"Item ID: "+"\t"+item.ItemID +"\t" +"Item Name: " +"\t"+ item.ItemName +"\n"+ "Category:" +"\t"+ item.ItemCategory+ "\t"+"Description:"+"\t"+item.ItemDescription+"\t"+"Total:"+ "\t"+item.Total + " ";

email.Body = Body;

}

try

{

server.Send(email);

}

catch (SmtpFailedRecipientException error)

{

return View(error.FailedRecipient);

}

return RedirectToAction("RemovefromSell","Home",new { id = Session["itemid"] });

}

Remove ordered item from Product table on checking out:

public ActionResult RemovefromSell(int id)

{

var sell = db.SellProductDetails.SingleOrDefault(c => c.ItemID == id);

if (sell != null)

{

db.SellProductDetails.Remove(sell);

db.SaveChanges();

}

else

{

TempData["message"] = "Fail";

}

// return RedirectToAction("Viewcart", "Home", new { id = cartit.ItemID }); ;

return View("Success");

}

Remove selected item from cart:

//removes from cart when remove from cart is clicked

public ActionResult Remove(int id)

{

var cartit = db.CartDetails.SingleOrDefault(c => c.ItemID == id);

if (cartit != null)

{

if (cartit.Count > 1) //decreases the quantity by 1 when there are multiple items

{

cartit.Count--;

}

else

{

db.CartDetails.Remove(cartit); //removes the only item from cart

}

db.SaveChanges();

}

else

{

TempData["message"] = "Fail";

}

return RedirectToAction("Viewcart", "Home", new { id = cartit.ItemID });

}

PayPal Authentication:

Web.config file:

<appSettings>

<add key="PDTToken" value="wx4npw3jn6ysj6hk$3ba69c8cac3e21a6bc04ba654e58aa0b"/>

<add key="PayPalUserName" value="madhus15@xyz.com"/>

<add key="returnurl" value="https://localhost:44333//Home/Success"/>

<add key="PayPalSubmitUrl" value="https://www.sandbox.paypal.com/cgi-bin/webscr"/>

</appSettings>

To enable paypal gateway transaction, PDTHolder class is used.

public class PDTHolder

{

public double GrossTotal { get; set; }

public int InvoiceNumber { get; set; }

public string PaymentStatus { get; set; }

public string PayerFirstName { get; set; }

public double PaymentFee { get; set; }

public string BusinessEmail { get; set; }

public string PayerEmail { get; set; }

public string TxToken { get; set; }

public string PayerLastName { get; set; }

public string ReceiverEmail { get; set; }

public string ItemName { get; set; }

public string Currency { get; set; }

public string TransactionId { get; set; }

public string SubscriberId { get; set; }

public string Custom { get; set; }

private static string authToken, txToken, query, strResponse;

public static PDTHolder Success(string tx)

{

authToken = WebConfigurationManager.AppSettings["PDTToken"];

txToken = tx;

query = string.Format("cmd=\_notify-synch&tx={0}&at={1}", txToken, authToken);

string url = WebConfigurationManager.AppSettings["PayPalSubmitUrl"];

HttpWebRequest req = (HttpWebRequest)WebRequest.Create(url);

req.Method = "POST";

req.ContentType = "application/x-www-form-urlencoded";

req.ContentLength = query.Length;

StreamWriter stOut = new StreamWriter(req.GetRequestStream(), System.Text.Encoding.ASCII);

stOut.Write(query);

stOut.Close();

if (strResponse.StartsWith("SUCCESS"))

return PDTHolder.Parse(strResponse);

return null;

}

private static PDTHolder Parse(string postData)

{

String sKey, sValue;

PDTHolder ph = new PDTHolder();

try

{

String[] StringArray = postData.Split('\n');

int i;

for (i = 1; i < StringArray.Length - 1; i++)

{

String[] StringArray1 = StringArray[i].Split('=');

sKey = StringArray1[0];

sValue = HttpUtility.UrlDecode(StringArray1[1]);

switch (sKey)

{

case "mc\_gross":

ph.GrossTotal = Convert.ToDouble(sValue);

break;

case "invoice":

ph.InvoiceNumber = Convert.ToInt32(sValue);

break;

case "payment\_status":

ph.PaymentStatus = Convert.ToString(sValue);

break;

case "first\_name":

ph.PayerFirstName = Convert.ToString(sValue);

break;

case "mc\_fee":

ph.PaymentFee = Convert.ToDouble(sValue);

break;

case "business":

ph.BusinessEmail = Convert.ToString(sValue);

break;

case "payer\_email":

ph.PayerEmail = Convert.ToString(sValue);

break;

case "Tx Token":

ph.TxToken = Convert.ToString(sValue);

break;

case "last\_name":

ph.PayerLastName = Convert.ToString(sValue);

break;

case "receiver\_email":

ph.ReceiverEmail = Convert.ToString(sValue);

break;

case "item\_name":

ph.ItemName = Convert.ToString(sValue);

break;

case "mc\_currency":

ph.Currency = Convert.ToString(sValue);

break;

case "txn\_id":

ph.TransactionId = Convert.ToString(sValue);

break;

case "custom":

ph.Custom = Convert.ToString(sValue);

break;

case "subscr\_id":

ph.SubscriberId = Convert.ToString(sValue);

break;

}

}

return ph;

}

catch (Exception ex)

{

throw ex;

}

}

}

PayPal payment redirection page:

@{

Layout = null;

}

@using System.Configuration

<html>

<body>

<h3>Payment Method:</h3>

<h2>Pay using your paypal account</h2>

<form method="post" action="@ConfigurationManager.AppSettings["PayPalSubmitUrl"]">

<input type="hidden" name="upload" value="1" />

<input type="hidden" name="return" value="@ConfigurationManager.AppSettings["returnurl"]" />

<input type="hidden" name="cmd" value="\_cart" />

<input type="hidden" name="business" value="@ConfigurationManager.AppSettings["PayPalUsername"]" />

<div class="form-group">

@\* redirects to paypal page \*@

<input type="image" src="https://www.paypal.com/en\_US/i/btn/btn\_xpressCheckout.gif" />

</div>

</form>

</body>

</html>

Facebook Authentication:

Install Facebook package, Facebook owin security package.

Startup.Auth.cs:

using System;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.Owin;

using Microsoft.Owin;

using Microsoft.Owin.Security.Cookies;

using Microsoft.Owin.Security.Google;

using Owin;

using ShoppingAppFB.Models;

using Microsoft.Owin.Security.Facebook;

using System.Threading.Tasks;

public void ConfigureAuth(IAppBuilder app)

{

app.UseFacebookAuthentication(new FacebookAuthenticationOptions

{

AppId = "189731444877054",

AppSecret = "980f7062072a8d1ccbc972a78bb2ad92",

Scope = { "email" },

Provider = new FacebookAuthenticationProvider

{

OnAuthenticated = context =>

{

context.Identity.AddClaim(new System.Security.Claims.Claim("FacebookAccessToken", context.AccessToken));

return Task.FromResult(true);

}

}

});

Get facebook user details and save to User Details table:

[AllowAnonymous]

public async Task<ActionResult> ExternalLoginCallback(string returnUrl)

{

var loginInfo = await AuthenticationManager.GetExternalLoginInfoAsync();

if (loginInfo == null)

{

return RedirectToAction("Login");

}

if (loginInfo.Login.LoginProvider == "Facebook")

{

var identity = AuthenticationManager.GetExternalIdentity(DefaultAuthenticationTypes.ExternalCookie);

var access\_token = identity.FindFirstValue("FacebookAccessToken");

// var dob = identity.Equals("user\_birthday");

var fb = new FacebookClient(access\_token);

dynamic myInfo = fb.Get("/me?fields=email,first\_name,id");

string email = myInfo.email;

string name = myInfo.first\_name;

string id = myInfo.id;

//var id = myInfo.id;

Session["email"] = email;

Session["name"] = name;

Session["id"] = id;

Session["SellerID"] = id;

UserDetail det = new UserDetail();

using (var db = new ShoppingAppFBEntities2())

{

var results = (from c in db.UserDetails where c.UserID == id select c).SingleOrDefault();

if (results == null)

{

det.UserEmail = email;

det.UserName = name;

det.UserID = id;

db.UserDetails.Add(det);

db.SaveChanges();

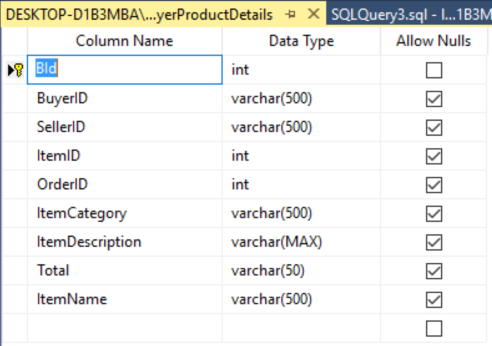
}

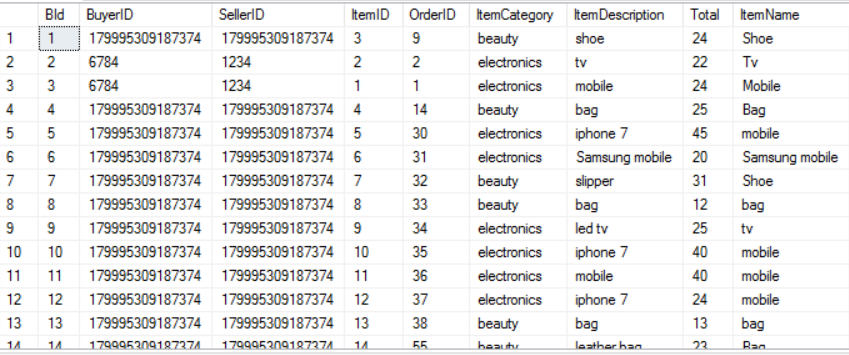
}

}

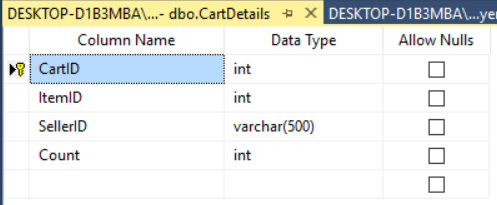
Database design and values:

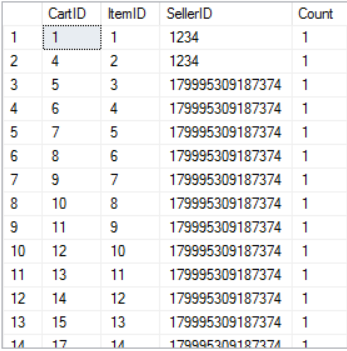
BuyerProductDetails:



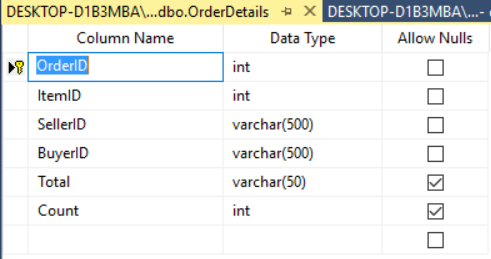


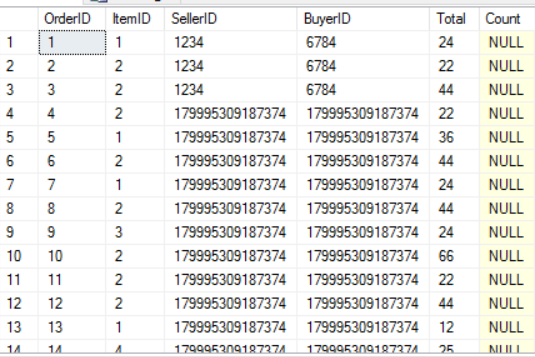
CartDetails:



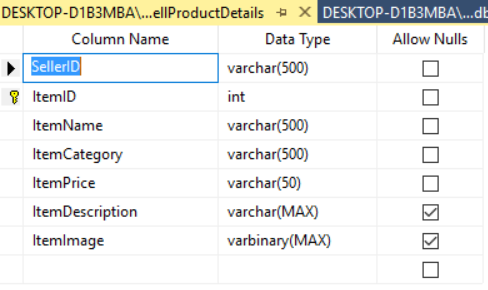


OrderDetails:



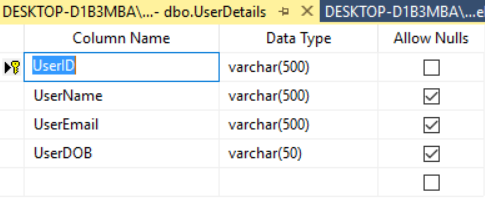


SellProductDetails:





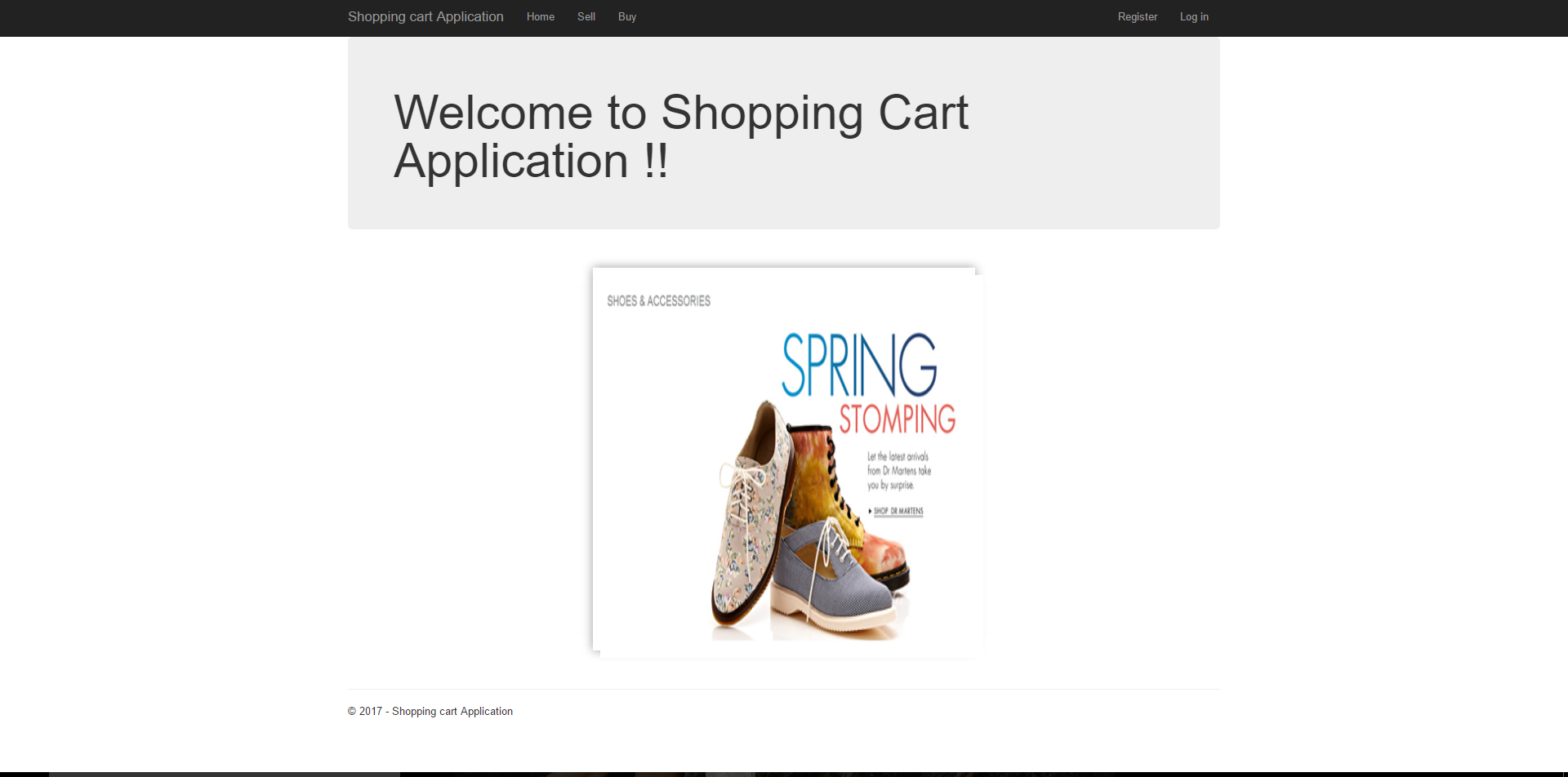
UserDetails:



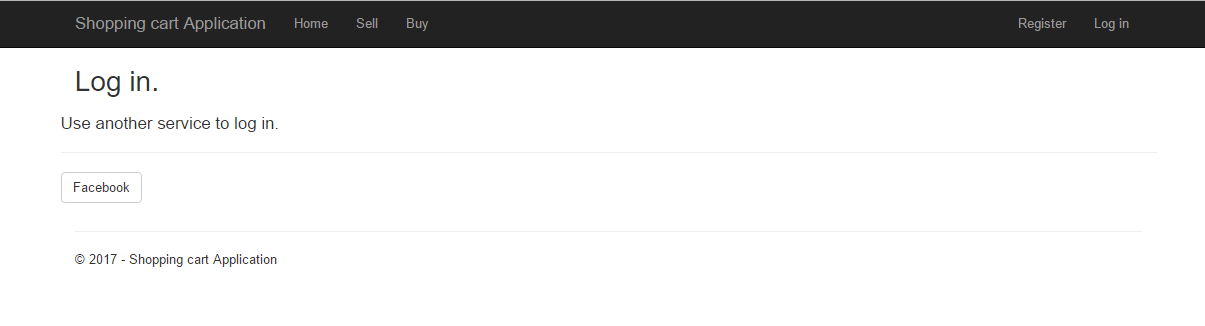


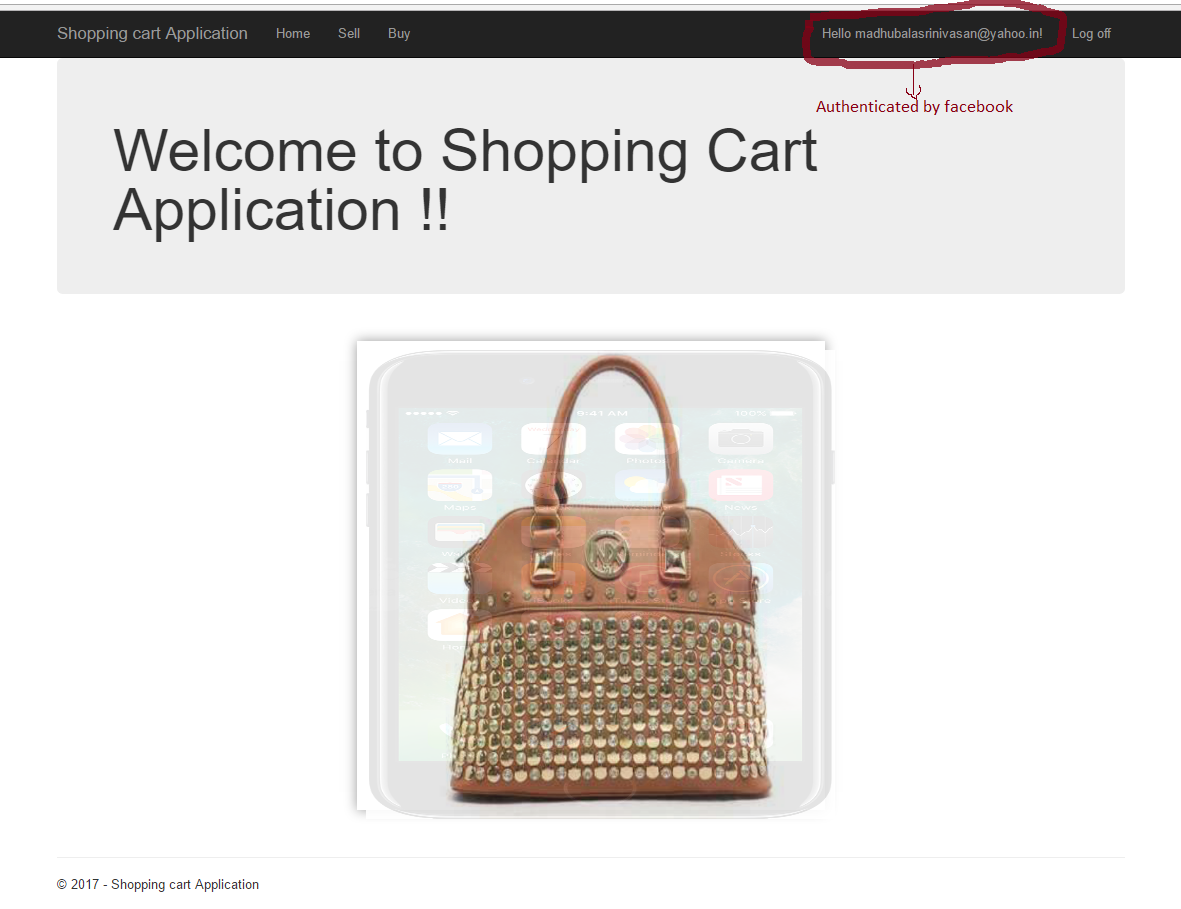
Screenshot of Result:

Home Page:

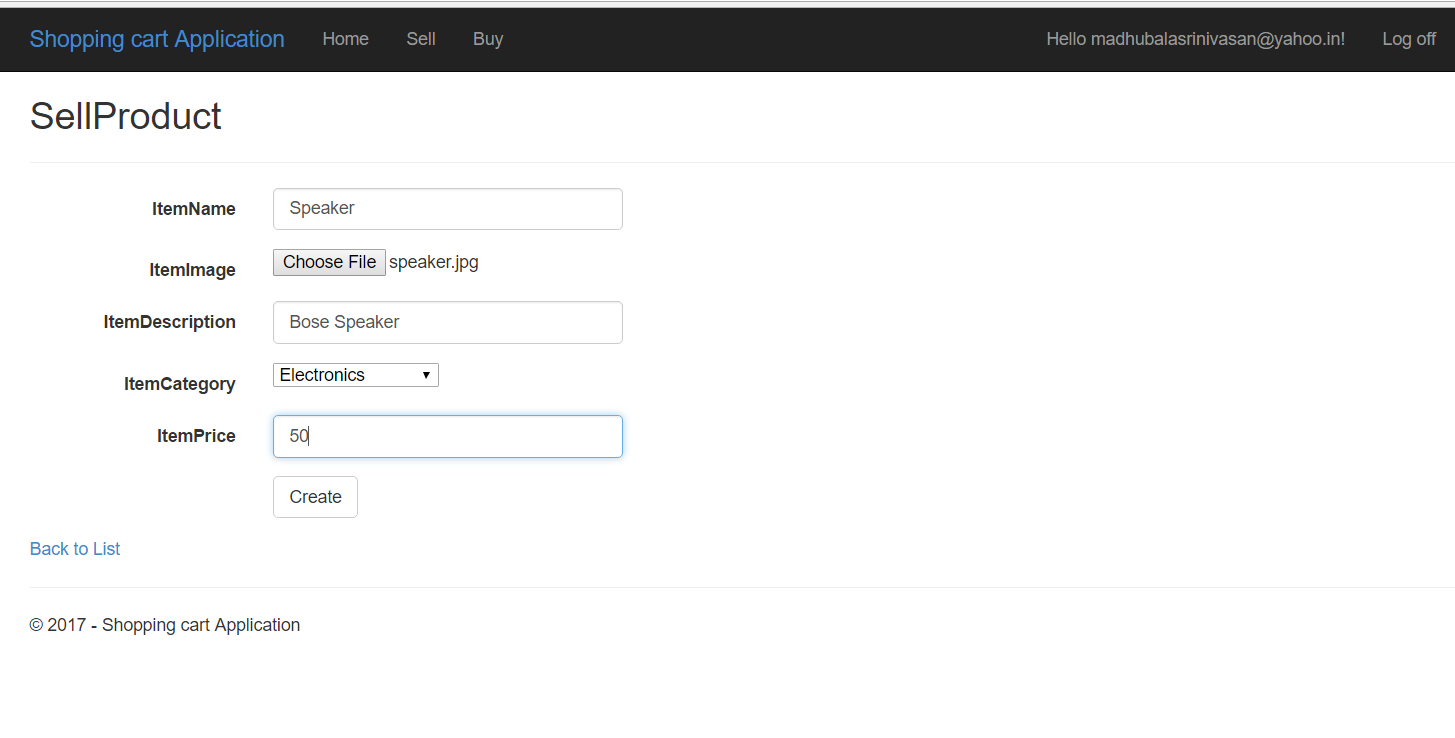


Login Page:

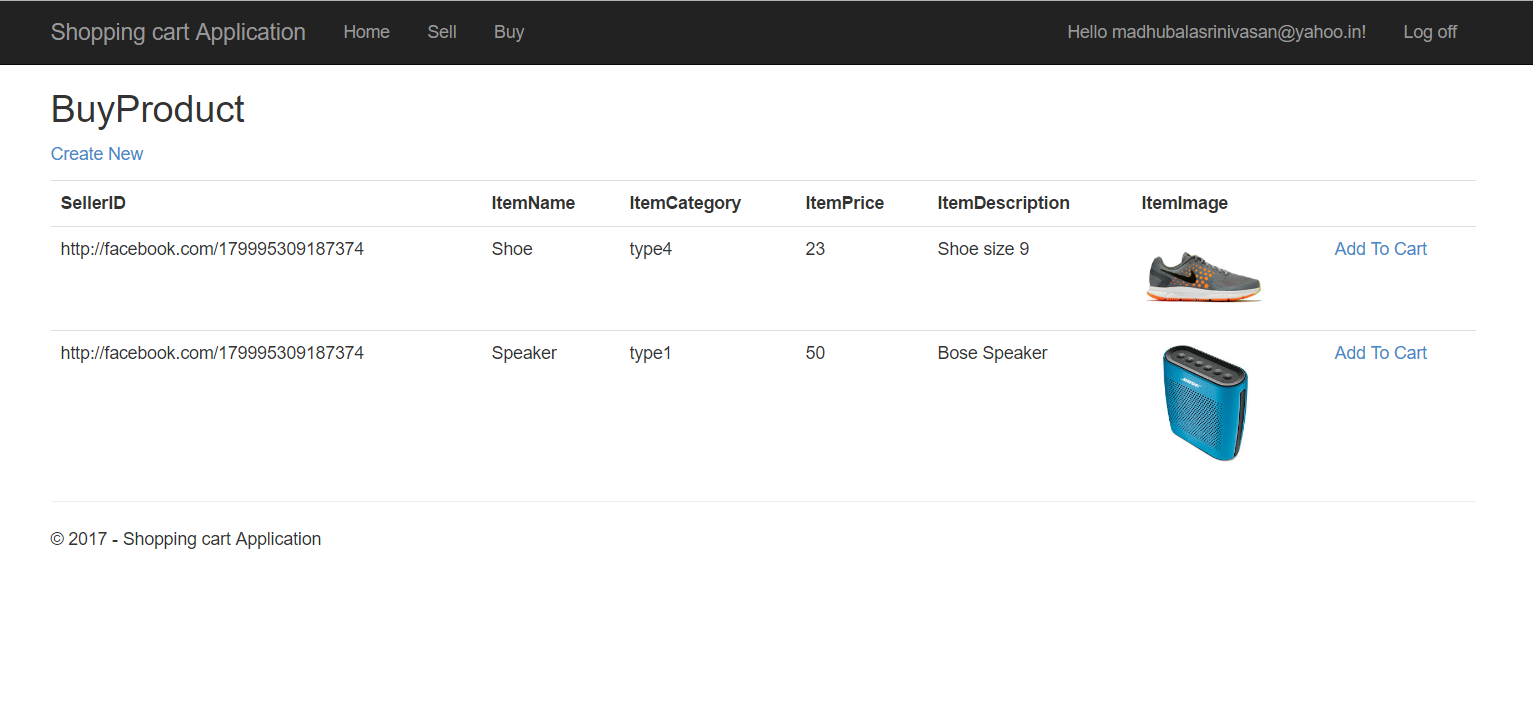


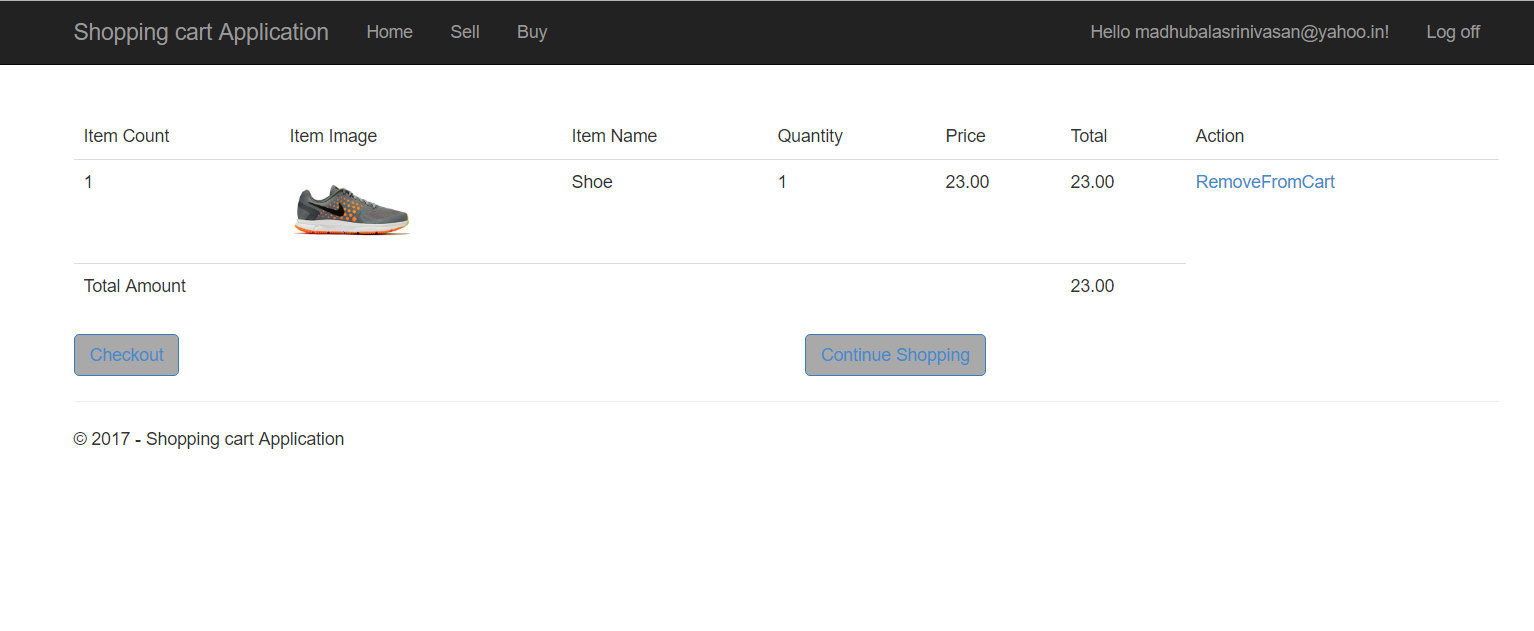


Sell a Product Page:

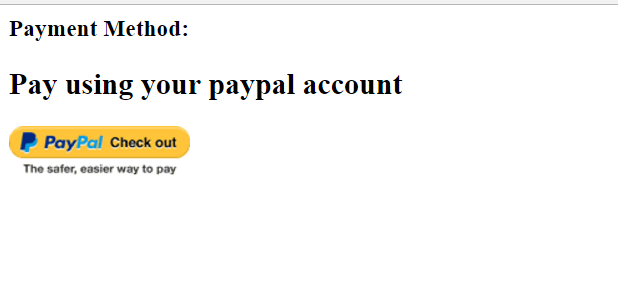


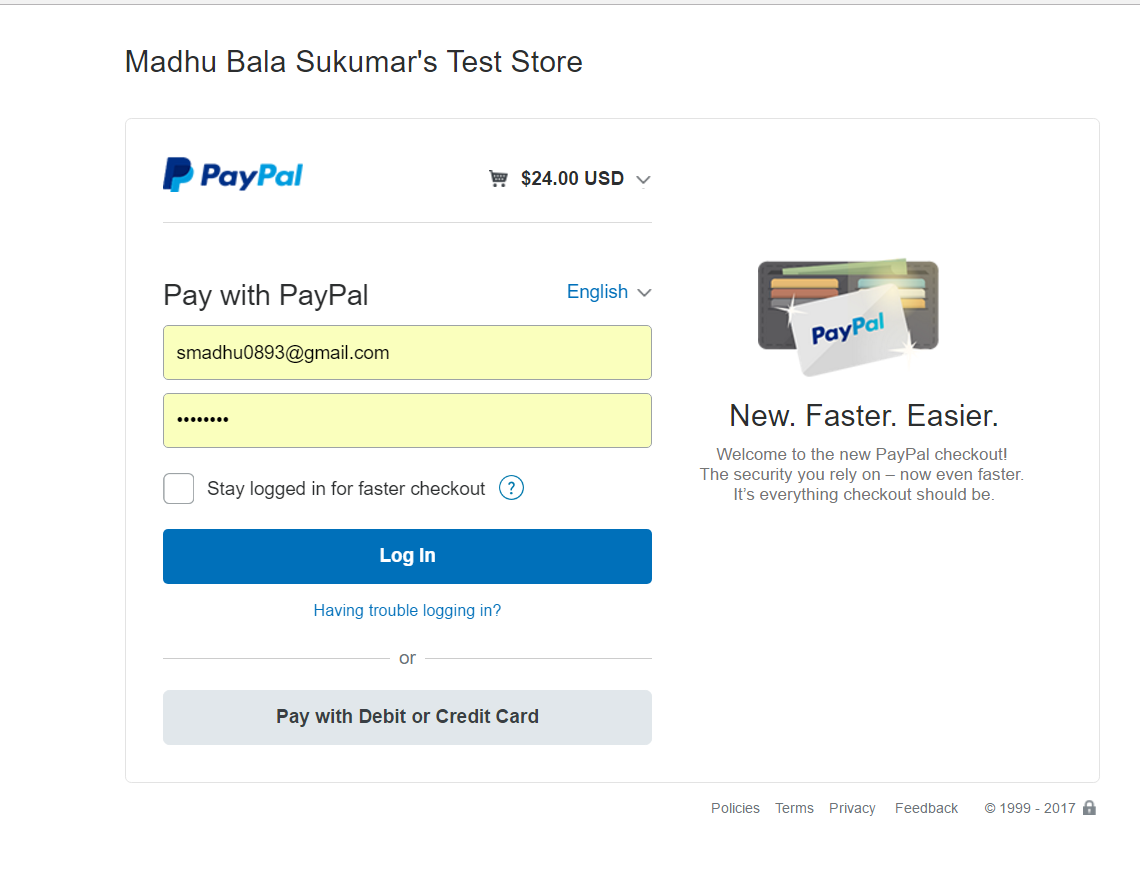
Product Details Page:

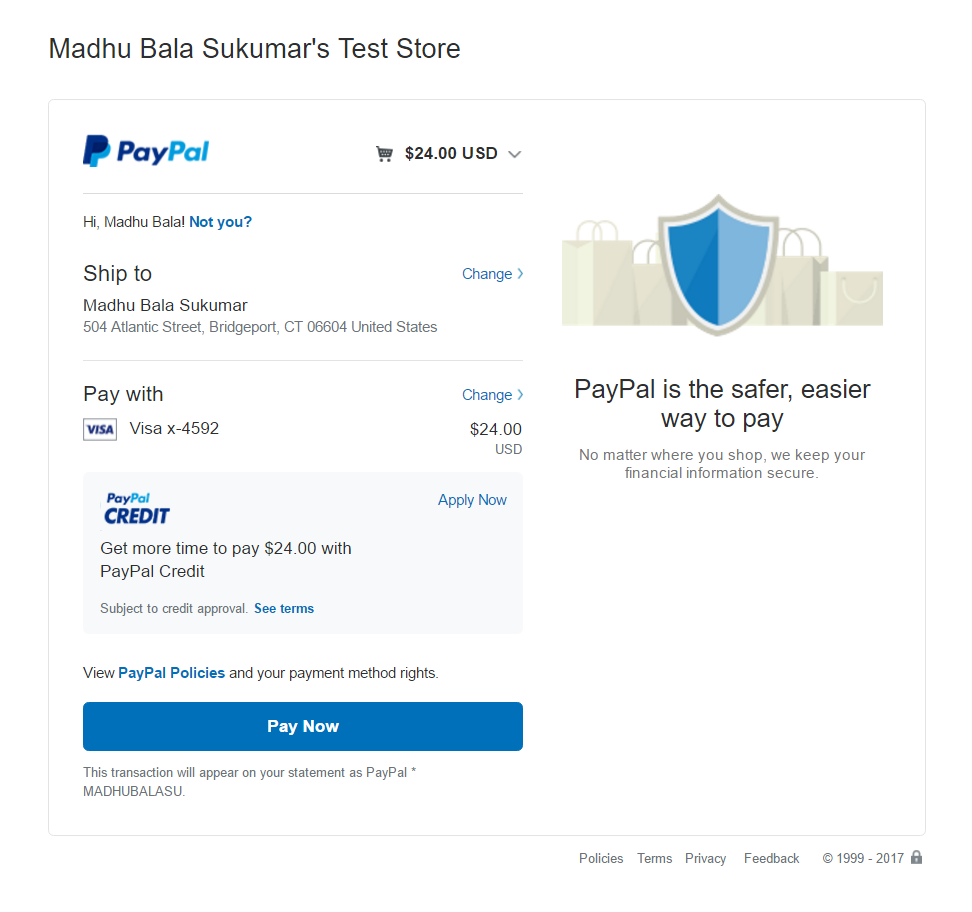


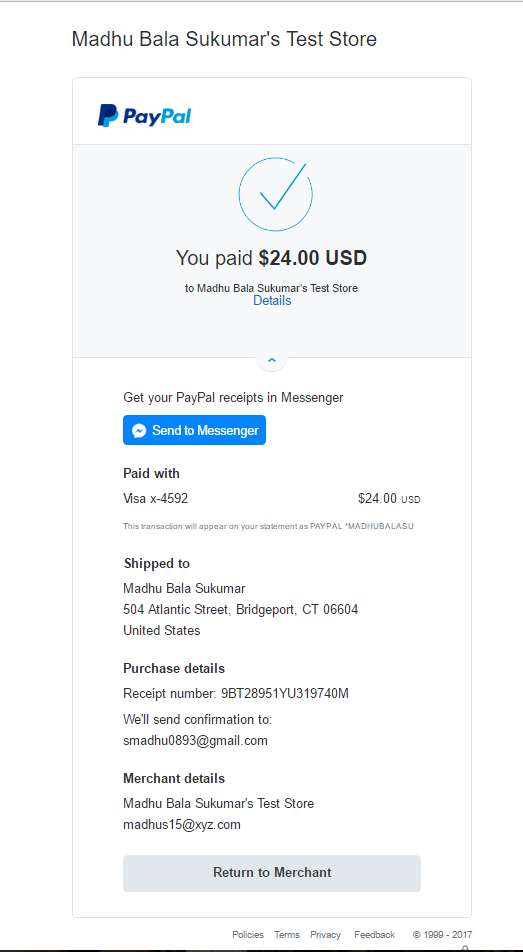
View Cart Page: 

Payment Page:

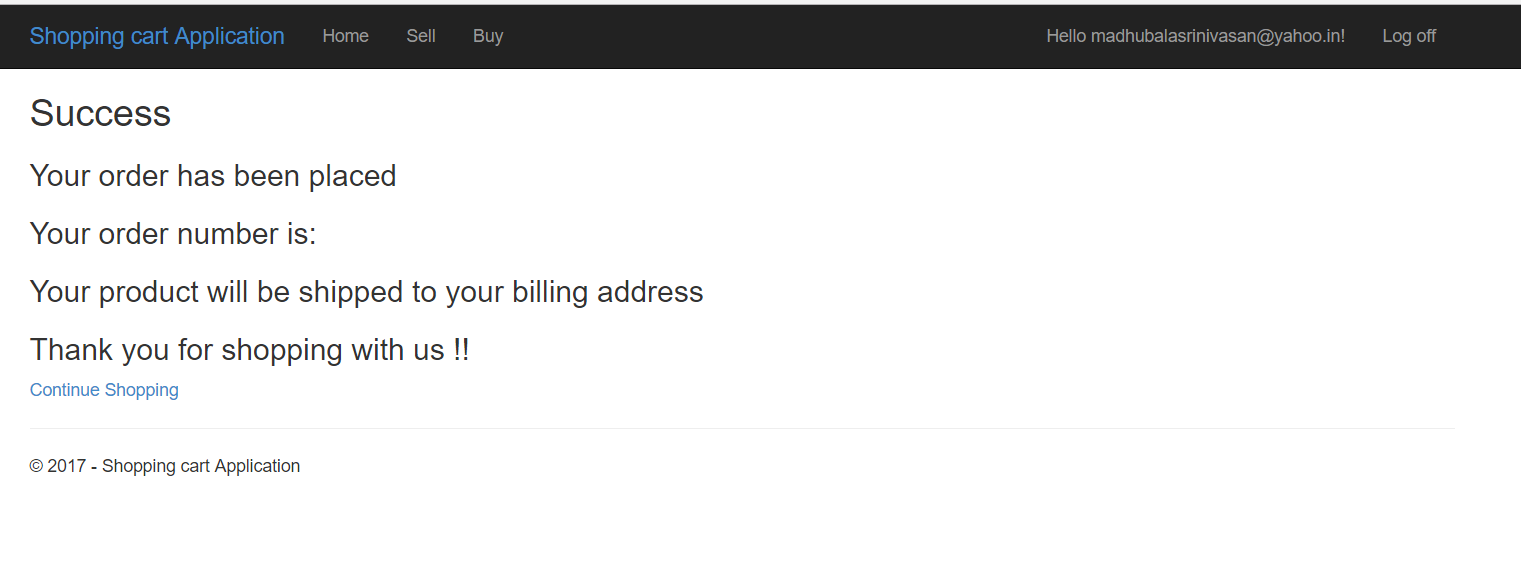


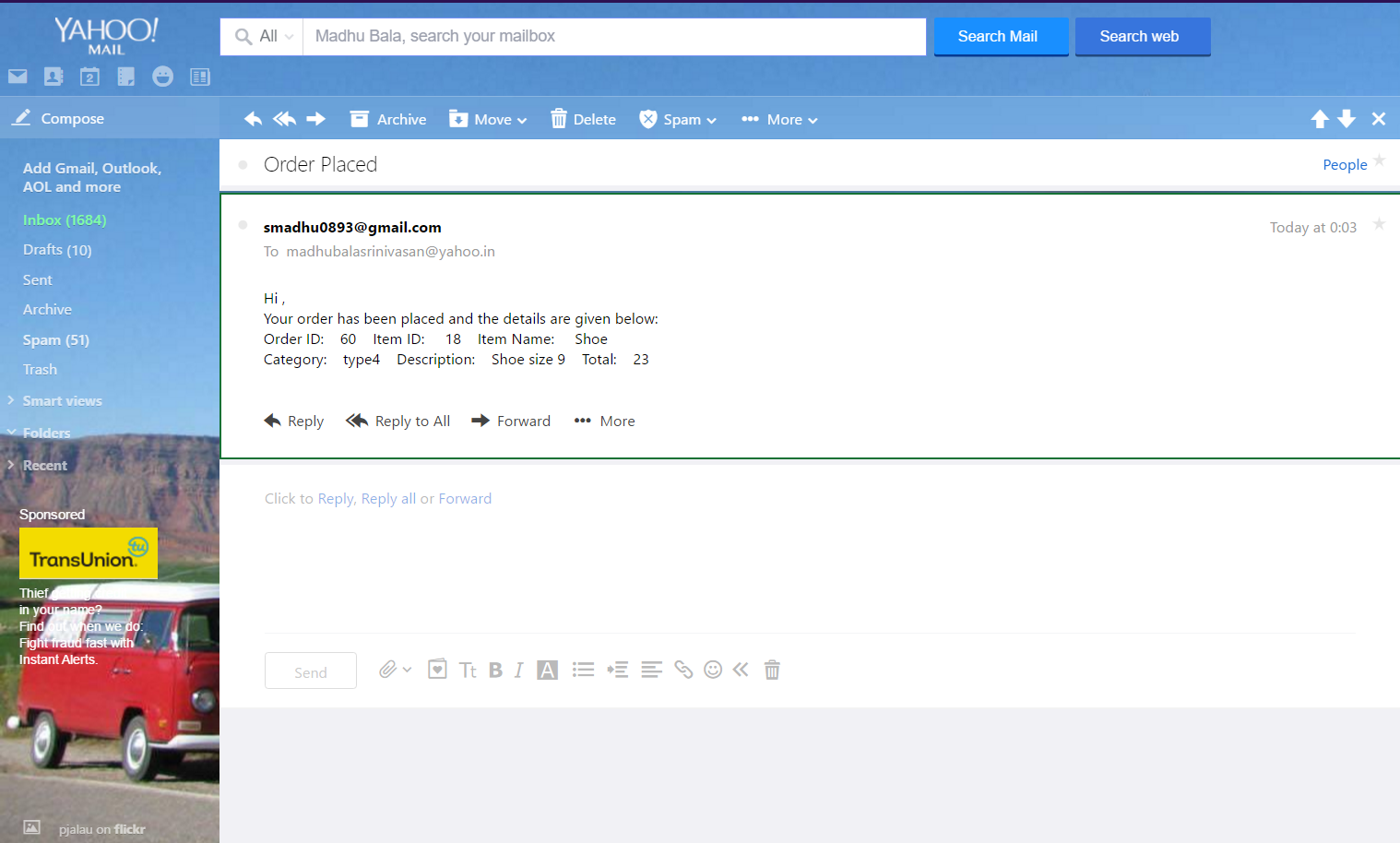






Successful Payment Page:





Checkedout item removed from available items:

