**PROJECT REPORT ON**

**“Let’s Shopping Electronic Store”**

**SUBMITTED BY**

**MR. SUNIL B PATEL**

**Seat No. 1130730**

**UNDER THE GUIDANCE OF**

**PROF. DIVYA SHETTY**

**TO**

**UNIVERSITY OF MUMBAI**

**IN PARTIAL FULLFILMENT OF**

**T.Y.B.SC (INFORMATION TECHNOLOGY)**

**SEM –VI**

**ACADEMIC YEAR 2014-2015**

**GURU NANAK**

**COLLEGE OF ARTS, SCIENCE & COMMERCE,**

**G.T.B NAGAR , MUMBAI -400037**

**ACKNOWLEDGEMENT**

I have great pleasure in presenting this project entitled “**Let’s Shopping Electronic Store**” And I grab this opportunity to convey our immense regards toward all the distinguished people who have their invaluable contribution in the hour of need.

First of all I express my vote of think to our **Principle Dr.Vijay V. Dabholkar**.

I would also like to thank our **Project guide: Miss. Divya Shetty** for their timely support in the completion of this project along with her full support and cheerful encouragement.

I would also thank for giving her frank opinions and various inputs in long discussion on the project which helped me shape the software, effectively improve the quality of the software.

I higly onlige thankful to the entire teaching staff of the Information Technology who have spared efforts in making the project a successful endeavor.

Last but not the least I extends our sincere thanks and appreciations to our family without their support and motivation this project was not possible

**Index**

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Title** | **Page No.** |
| **1** | **Introduction** |  |
| **2** | **System Study** |  |
|  | 2.1 Existing System |  |
|  | 2.2 Drawbacks in the Existing System |  |
| **3** | **System Analysis** |  |
|  | 3.1 Proposed System |  |
|  | 3.2 Scope |  |
|  | 3.3 Need for the Proposed System |  |
|  | 3.4 Feasibility Study |  |
|  | 3.4.1 Technical Feasibility |  |
|  | 3.4.2 Financial Feasibility |  |
|  | 3.4.3 Operational Feasibility |  |
| **4** | **Selected Software** |  |
|  | 4.1 Introduction to .NET Framework |  |
|  | 4.2 ASP.NET |  |
|  | 4.3 C#.NET |  |
|  | 4.4 Sql Server |  |
| **5** | **Requirement Analysis** |  |
|  | 5.1 User Class and Characteristics |  |
|  | 5.2 Functional Requirements |  |
|  | 5.3 Performance Requirements |  |
|  | 5.4 Non Functional Requirements |  |
|  | 5.5 External Interfaces Requirements |  |
|  | 5.6 General Constraints, Assumptions,  Dependencies, Guidelines |  |
| **6** | **System Design Specifiation** |  |
|  | 6.1 Architectural design |  |
|  | 6.1.1 Data Flow Diagrams |  |
|  | 6.2 ER Diagram |  |
|  | 6.3 Activity Diagram |  |
|  | 6.4 Gantt Chart |  |
| **7** | **Testing** |  |
| **8** | **Coding** |  |
| **9** | **Screen shots** |  |
| **10** | **Limitations of project** |  |
| **11** | **Future expansions** |  |
| **12** | **Bibliography** |  |

Introduction

**1. INTRODUCTION**

Online shopping is the process whereby consumers directly buy goods, services etc. from a seller interactively in real-time without an intermediary service over the internet.

Online shopping is the process of buying goods and services from merchants who sell on the Internet. Since the emergence of the World Wide Web, merchants have sought to sell their products to people who surf the Internet. Shoppers can visit web stores from the comfort of their homes and shop as they sit in front of the computer.Consumers buy a variety of items from online stores. In fact, people can purchase just about anything from companies that provide their products online. Books, clothing, household appliances, toys, hardware, software, and health insurance are just some of the hundreds of products consumers can buy from an online store.

Many people choose to conduct shopping online because of the convenience. For example, when a person shops at a brick-and-mortar store, she has to drive to the store, find a parking place, and walk throughout the store until she locates the products she needs. After finding the items she wants to purchase, she may often need to stand in long lines at the cash register.

Despite the convenience of online shopping, not everyone chooses to purchase items and services online. Some people like the idea of physically going to a store and experiencing the shopping process. They like to touch the merchandise, try on clothing, and be around other people. Online shopping doesn't permit shoppers to touch products or have any social interaction. It also doesn't allow them to take the merchandise home the same day they buy it.

Online shopping allows you to browse through endless possibilities, and even offers merchandise that's unavailable in stores. If you're searching for a niche product that may not be distributed locally, you're sure to find what you're looking for on the internet. What's even more useful is the ability to compare items, similar or not, online. You can search through multiple stores at the same time, comparing material quality, sizes and pricing simultaneously.

Shopping via the internet eliminates the need to sift through a store's products with potential buys like pants, shirts, belts and shoes all slung over one arm. Online shopping also eliminates the catchy, yet irritating music, as well as the hundreds, if not thousands, of other like-minded individuals who seem to have decided to shop on the same day.

Say 'goodbye' to the days when you stood in line waiting, and waiting, and waiting some more for a store clerk to finally check out your items. Online shopping transactions occur instantly-saving you time to get your other errands done! Additionally, unlike a store, online shopping has friendly customer service representatives available 24 hours a day, 7 days a week to assist you with locating, purchasing and shipping your merchandise.

**System Study**

**2. SYSTEM STUDY**

Information systems projects’ originate from many reasons: to achieve greater speed in processing data, better accuracy and improved consistency, faster information retrieval, integration of business areas, reduced cost and better security. The sources also vary project proposals originate with department managers, senior executives and systems analysis.

Sometimes the real origin is an outside source, such as a government agency which stipulates a systems requiremetns the organisattion must meet. When the request is made, the first systems activity, the preliminary investigation, begins. The activity has three parts: request clarification, feasibility study and request approval

**2.1 Existing System:**

The existing system was an automated system. But It was found to be inefficient in meeting the growing demands of population.

**2.2 Drawbacks in the existing systems:**

* Disadvantage of the existing system:
* Time Consuming
* Expensive
* Needed an agent
* We have to out for that.

**System Analysis** **3. SYSTEM ANALYSIS**

* This system is all about the converting the shopping

system from manual to online.

* Customer can buy products online after login to the site.
* Administrator is adding product to database.
* Administrator can edit or delete the products
* from the database.
* After buying and making payment the products are

send to customers address that he has given.

* Customer can write feedback for the product or services.
* Admin can see daily sell and feedback given by customer.
* Administrator is adding the delivery report to the database.
* Both admin and customer can see the delivery report**.**

**3.1 Purpose:**

Online shopping tries to enhance access to care and improve the continuity and efficiency of services. Depending on the specific setting and locale, case managers are responsible for a variety of tasks, ranging from linking clients to services to actually providing intensive shopping and delivery services themselves

**Main objective**

* To shop wile in the comfort of your own home ,without having to step out of the door.

sell at lower rate due to less over head.

* provide home delivery free of cost.
* No wait to see the products if someone else is taking that.

**3.2 Scope:**

This product has great future scope. Online shopping Internet software developed on and for the Windows and later versions environments and Linux OS. This project also provides security with the use of Login-id and Password, so that any unauthorized users can not use your account. The only Authorized that will have proper access authority can access the software.

**3.3 Need for the proposed system:**

The online shopping (HOME SHOP) is an easy to maintain, ready to run, scalable, affordable and reliable cost saving tool from Software Associates suited for small, medium, and large shopping complex and shopping malls.

**Features and Benefits:**

* Providing security
* Low cost
* Basic computer knowledge required
* Configurable and extensible application UI design
* he proposed system can be used even by the naïve users and it does not require any educational level, experience, and technical expertise in computer field but it will be of good use if the user has the good knowledge of how to operate a computer.

**3.4 Feasibility study:**

* A feasibility study is a short, focused study, which aims to answer a number of questions:
* Does the system contribute to the overall objectives of the organizations?
* Can the system be implemented using current technology and within given cost and schedule constrains?
* Can the system be integrated with systems which are already in place?

**3.4.1 Technical Feasibility:**

* Is the project feasibility within the limits of current technology?
* Does the technology exist at all?
* Is it available within given resource constraints (i.e., budget, schedule)?

**3.4.2 Financial Feasibility:**

* Is the project possible, given resource constraints?
* Are the benefits that will accrue from the new system worth the costs?
* What are the savings that will result from the system, including tangible and intangible ones?
* What are the development and operational costs?

**3.4.3 Operational Feasibility:**

Define the urgency of the problem and the acceptability of any solution; if the system is developed, will it be used? Includes people-oriented and social issues: internal issues, such as manpower problems, labour objections, manager resistance, organizational conflicts and policies; also external issues, including social acceptability, legal aspects and government regulations.

In preliminary investigation feasibility study has three aspects..

* Technical Feasibilty
* Operational Feasibilty
* Economical Feasibility

**Technical Feasibility**

Technical issues involved are the necessary technology existence, technical guarantees of accuracy, reliability, ease of access, data security, aspects of future expansion.

* Technology exists to develop a system.
* The proposed system is capable of holding data to be used.
* The proposed system is capable of providing adequate response and regardless of the number of users.
* The proposed system being modular to the administrator, if he/she wants can add more features in the future and as well as be able to expand the system.
* As far as the hardware and software is concerned, the proposed system is completely liable with proper backup and security.

Hence, we can say that the proposed system is technically feasible.

**Operational Feasibility**

If the system meets the requirements of the customers and the administrator we can say that the system is operationally feasible.

The proposed system will be beneficial only if it can be turned into a system which will meet the requirements of the store when it is developed and installed, and there is sufficient support from the users

The proposed system will improve the total performance.

* Customers here are the most important part of the system and the proposed system will provide them with a convenient mode of operation for them.
* The proposed system will be available to the customers throughout the globe.
* The proposed system will provide a better market for different dealers.

Hence, the proposed system is operationally feasible.

**Economical Feasibility**

Economic Feasibility is the most frequently used method for evaluating the effectiveness of the proposed system if the benefits of the proposed system outweighs the cost then the decision is made to design and implement the system.

* The cost of hardware and software is affordable.
* High increase in the amount of profit earned by going global.
* Easy and cheap maintenance of the system possible.

Very cheap price for going global

**FEATURES OF THE PROPOSED SYSTEM**

* The proposed system is flexible both for the administrators and the customers visiting the website.
* The proposed system provides a unique platform for different silk vendors to interact using the same platform.
* The proposed system allows easy promotion of the site through emails and newsletters.
* The proposed system gives information about the delivery and present status of their orders.
* Management of data is easy.
* Security is provided wherever necessary.

**PROPOSED SYSTEM**

In the proposed website there are different parts or modules which are summarized as follows

**CUSTOMER REGISTRATION*:***

Customers are required to register on the website before they can do the shopping. The website also provides several features for the non-registered user. Here they can choose their id and all the details regarding them are collected and a mail is sent to the email address for confirmation.

**SHOPPING CART*:***

Shopping cart module tries to simulate the working of a store where user can view each design, color, size and price of the product available. The items they like can be added to the logical cart and can be removed if not required later. Billing and other payment related matters are handled here.

**ADMINISTRATION:**

This is the part of the website where the administrators can add delete or update the product information. Administrators are also responsible for adding and deleting the customers from the website. In addition, newsletter and promotions are also handled by the site administrator via e-mail

**SEARCH :**

This facility is provided to both registered and unregistered user. User can search for the availability and type of products available on the website.

**Selected Software**

**4. SELECTED SOFTWARE**

**4.1 INTRODUCTION TO .NET Framework**

The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet. The .NET Framework is designed to fulfill the following objectives:

* To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
* To provide a code-execution environment that minimizes software deployment and versioning conflicts.
* To provide a code-execution environment that guarantees safe execution of code, including code created by an unknown or semi-trusted third party.
* To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
* To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
* To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

The .NET Framework has two main components: the common language runtime and the .NET Framework class library. The common language runtime is the foundation of the .NET Framework. You can think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management, and Remoting, while also enforcing strict type safety and other forms of code accuracy that ensure security and robustness. In fact, the concept of code management is a fundamental principle of the runtime. Code that targets the runtime is known as managed code, while code that does not target the runtime is known as unmanaged code. The class library, the other main component of the .NET Framework, is a comprehensive, object-oriented collection of reusable types that you can use to develop applications ranging from traditional command-line or graphical user interface (GUI) applications to applications based on the latest innovations provided by ASP.NET, such as Web Forms and XML Web services.

The .NET Framework can be hosted by unmanaged components that load the common language runtime into their processes and initiate the execution of managed code, thereby creating a software environment that can exploit both managed and unmanaged features. The .NET Framework not only provides several runtime hosts, but also supports the development of third-party runtime hosts.

For example, ASP.NET hosts the runtime to provide a scalable, server-side environment for managed code. ASP.NET works directly with the runtime to enable Web Forms applications and XML Web services, both of which are discussed later in this topic.

Internet Explorer is an example of an unmanaged application that hosts the runtime (in the form of a MIME type extension). Using Internet Explorer to host the runtime enables you to embed managed components or Windows Forms controls in HTML documents. Hosting the runtime in this way makes managed mobile code (similar to Microsoft® ActiveX® controls) possible, but with significant improvements that only managed code can offer, such as semi-trusted execution and secure isolated file storage.

The following illustration shows the relationship of the common language runtime and the class library to your applications and to the overall system. The illustration also shows how managed code operates within a larger architecture.

**FEATURES OF THE COMMON LANGUAGE RUNTIME**

The common language runtime manages memory, thread execution, code execution, code safety verification, compilation, and other system services. These features are intrinsic to the managed code that runs on the common language runtime.

With regards to security, managed components are awarded varying degrees of trust, depending on a number of factors that include their origin (such as the Internet, enterprise network, or local computer). This means that a managed component might or might not be able to perform file-access operations, registry-access operations, or other sensitive functions, even if it is being used in the same active application.

The runtime enforces code access security. For example, users can trust that an executable embedded in a Web page can play an animation on screen or sing a song, but cannot access their personal data, file system, or network. The security features of the runtime thus enable legitimate Internet-deployed software to be exceptionally featuring rich.

The runtime also enforces code robustness by implementing a strict type- and code-verification infrastructure called the common type system (CTS). The CTS ensures that all managed code is self-describing. The various Microsoft and third-party language compilers

Generate managed code that conforms to the CTS. This means that managed code can consume other managed types and instances, while strictly enforcing type fidelity and type safety.

In addition, the managed environment of the runtime eliminates many common software issues. For example, the runtime automatically handles object layout and manages references to objects, releasing them when they are no longer being used. This automatic memory management resolves the two most common application errors, memory leaks and invalid memory references.

The runtime also accelerates developer productivity. For example, programmers can write applications in their development language of choice, yet take full advantage of the runtime, the class library, and components written in other languages by other developers. Any compiler vendor who chooses to target the runtime can do so. Language compilers that target the .NET Framework make the features of the .NET Framework available to existing code written in that language, greatly easing the migration process for existing applications.

While the runtime is designed for the software of the future, it also supports software of today and yesterday. Interoperability between managed and unmanaged code enables developers to continue to use necessary COM components and DLLs.

The runtime is designed to enhance performance. Although the common language runtime provides many standard runtime services, managed code is never interpreted. A feature called just-in-time (JIT) compiling enables all managed code to run in the native machine language of the system on which it is executing. Meanwhile, the memory manager removes the possibilities of fragmented memory and increases memory locality-of-reference to further increase performance.

Finally, the runtime can be hosted by high-performance, server-side applications, such as Microsoft® SQL Server™ and Internet Information Services (IIS). This infrastructure enables you to use managed code to write your business logic, while still enjoying the superior performance of the industry's best enterprise servers that support runtime hosting.

**.NET FRAMEWORK CLASS LIBRARY**

The .NET Framework class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is object oriented, providing types from which your own managed code can derive functionality. This not only makes the .NET Framework types easy to use, but also reduces the time associated with learning new features of the .NET Framework. In addition, third-party components can integrate seamlessly with classes in the .NET Framework.

For example, the .NET Framework collection classes implement a set of interfaces that you can use to develop your own collection classes. Your collection classes will blend seamlessly with the classes in the .NET Framework.

As you would expect from an object-oriented class library, the .NET Framework types enable you to accomplish a range of common programming tasks, including tasks such as string management, data collection, database connectivity, and file access. In addition to these common tasks, the class library includes types that support a variety of specialized development scenarios. For example, you can use the .NET Framework to develop the following types of applications and services:

* Console applications.
* Scripted or hosted applications.
* Windows GUI applications (Windows Forms).
* ASP.NET applications.
* XML Web services.
* Windows services.

For example, the Windows Forms classes are a comprehensive set of reusable types that vastly simplify Windows GUI development. If you write an ASP.NET Web Form application, you can use the Web Forms classes.

**CLIENT APPLICATION DEVELOPMENT**

Client applications are the closest to a traditional style of application in Windows-based programming. These are the types of applications that display windows or forms on the desktop, enabling a user to perform a task. Client applications include applications such as word processors and spreadsheets, as well as custom business applications such as data-entry tools, reporting tools, and so on. Client applications usually employ windows, menus, buttons, and other GUI elements, and they likely access local resources such as the file system and peripherals such as printers.

Another kind of client application is the traditional ActiveX control (now replaced by the managed Windows Forms control) deployed over the Internet as a Web page. This application is much like other client applications: it is executed natively, has access to local resources, and includes graphical elements.

In the past, developers created such applications using C/C++ in conjunction with the Microsoft Foundation Classes (MFC) or with a rapid application development (RAD) environment such as Microsoft® Visual Basic®. The .NET Framework incorporates aspects of these existing products into a single, consistent development environment that drastically simplifies the development of client applications.

The Windows Forms classes contained in the .NET Framework are designed to be used for GUI development. You can easily create command windows, buttons, menus, toolbars, and other screen elements with the flexibility necessary to accommodate shifting business needs.

For example, the .NET Framework provides simple properties to adjust visual attributes associated with forms. In some cases the underlying operating system does not support changing these attributes directly, and in these cases the .NET Framework automatically recreates the forms. This is one of many ways in which the .NET Framework integrates the developer interface, making coding simpler and more consistent.

Unlike ActiveX controls, Windows Forms controls have semi-trusted access to a user's computer. This means that binary or natively executing code can access some of the resources on the user's system (such as GUI elements and limited file access) without being able to access or compromise other resources. Because of code access security, many applications that once needed to be installed on a user's system can now be safely deployed through the Web. Your applications can implement the features of a local application while being deployed like a Web page.

**4.2 ASP.NET**

**SERVER APPLICATION DEVELOPMENT**

Server-side applications in the managed world are implemented through runtime hosts. Unmanaged applications host the common language runtime, which allows your custom managed code to control the behavior of the server. This model provides you with all the features of the common language runtime and class library while gaining the performance and scalability of the host server.

The following illustration shows a basic network schema with managed code running in different server environments. Servers such as IIS and SQL Server can perform standard operations while your application logic executes through the managed code.

**SERVER-SIDE MANAGED CODE**

ASP.NET is the hosting environment that enables developers to use the .NET Framework to target Web-based applications. However, ASP.NET is more than just a runtime host; it is a complete architecture for developing Web sites and Internet-distributed objects using managed code. Both Web Forms and XML Web services use IIS and ASP.NET as the publishing mechanism for applications, and both have a collection of supporting classes in the .NET Framework.

XML Web services, an important evolution in Web-based technology, are distributed, server-side application components similar to common Web sites. However, unlike Web-based applications, XML Web services components have no UI and are not targeted for browsers such as Internet Explorer and Netscape Navigator. Instead, XML Web services consist of reusable software components designed to be consumed by other applications, such as traditional client applications, Web-based applications, or even other XML Web services. As a result, XML Web services technology is rapidly moving application development and deployment into the highly distributed environment of the Internet.

If you have used earlier versions of ASP technology, you will immediately notice the improvements that ASP.NET and Web Forms offers. For example, you can develop Web Forms pages in any language that supports the .NET Framework. In addition, your code no longer needs to share the same file with your HTTP text (although it can continue to do so if you prefer). Web Forms pages execute in native machine language because, like any other managed application, they take full advantage of the runtime. In contrast, unmanaged ASP pages are always scripted and interpreted. ASP.NET pages are faster, more functional, and easier to develop than unmanaged ASP pages because they interact with the runtime like any managed application.

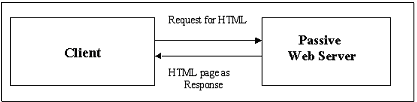
The .NET Framework also provides a collection of classes and tools to aid in development and consumption of XML Web services applications. XML Web services are built on standards such as SOAP (a remote procedure-call protocol), XML (an extensible data format), and WSDL ( the Web Services Description Language). The .NET Framework is built on these standards to promote interoperability with non-Microsoft solutions.

For example, the Web Services Description Language tool included with the .NET Framework SDK can query an XML Web service published on the Web, parse its WSDL description, and produce C# or Visual Basic source code that your application can use to become a client of the XML Web service. The source code can create classes derived from classes in the class library that handle all the underlying communication using SOAP and XML parsing. Although you can use the class library to consume XML Web services directly, the Web Services Description Language tool and the other tools contained in the SDK facilitate your development efforts with the .NET Framework.

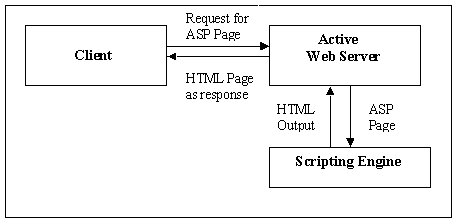
If you develop and publish your own XML Web service, the .NET Framework provides a set of classes that conform to all the underlying communication standards, such as SOAP, WSDL, and XML. Using those classes enables you to focus on the logic of your service, without concerning yourself with the communications infrastructure required by distributed software development.

Finally, like Web Forms pages in the managed environment, your XML Web service will run with the speed of native machine language using the scalable communication of IIS.

**A Passive Web Server stores static Web Pages**



**An active Web Server generates dynamic Web Pages**



**ACTIVE SERVER PAGES.NET**

ASP.NET is a programming framework built on the common language runtime that can be used on a server to build powerful Web applications. ASP.NET offers several important advantages over previous Web development models:

* **Enhanced Performance**. ASP.NET is compiled common language runtime code running on the server. Unlike its interpreted predecessors, ASP.NET can take advantage of early binding, just-in-time compilation, native optimization, and caching services right out of the box. This amounts to dramatically better performance before you ever write a line of code.
* **World-Class Tool Support**. The ASP.NET framework is complemented by a rich toolbox and designer in the Visual Studio integrated development environment. WYSIWYG editing, drag-and-drop server controls, and automatic deployment are just a few of the features this powerful tool provides.
* **Power and Flexibility**. Because ASP.NET is based on the common language runtime, the power and flexibility of that entire platform is available to Web application developers. The .NET Framework class library, Messaging, and Data Access solutions are all seamlessly accessible from the Web. ASP.NET is also language-independent, so you can choose the language that best applies to your application or partition your application across many languages. Further, common language runtime interoperability guarantees that your existing investment in COM-based development is preserved when migrating to ASP.NET.
* **Simplicity**. ASP.NET makes it easy to perform common tasks, from simple form submission and client authentication to deployment and site configuration. For example, the ASP.NET page framework allows you to build user interfaces that cleanly separate application logic from presentation code and to handle events in a simple, Visual Basic - like forms processing model. Additionally, the common language runtime simplifies development, with managed code services such as automatic reference counting and garbage collection.
* **Manageability**. ASP.NET employs a text-based, hierarchical configuration system, which simplifies applying settings to your server environment and Web applications. Because configuration information is stored as plain text, new settings may be applied without the aid of local administration tools. This "zero local administration" philosophy extends to deploying ASP.NET Framework applications as well. An ASP.NET Framework application is deployed to a server simply by copying the necessary files to the server. No server restart is required, even to deploy or replace running compiled code.
* **Scalability and Availability**. ASP.NET has been designed with scalability in mind, with features specifically tailored to improve performance in clustered and multiprocessor environments. Further, processes are closely monitored and managed by the ASP.NET runtime, so that if one misbehaves (leaks, deadlocks), a new process can be created in its place, which helps keep your application constantly available to handle requests.
* **Customizability and Extensibility**. ASP.NET delivers a well-factored architecture that allows developers to "plug-in" their code at the appropriate level. In fact, it is possible to extend or replace any subcomponent of the ASP.NET runtime with your own custom-written component. Implementing custom authentication or state services has never been easier.
* **Security**. With built in Windows authentication and per-application configuration, you can be assured that your applications are secure.

**LANGUAGE SUPPORT**

The Microsoft .NET Platform currently offers built-in support for three languages: C#, Visual Basic, and JScript.

**WHAT IS ASP.NET WEB FORMS?**

The ASP.NET Web Forms page framework is a scalable common language runtime programming model that can be used on the server to dynamically generate Web pages.

Intended as a logical evolution of ASP (ASP.NET provides syntax compatibility with existing pages), the ASP.NET Web Forms framework has been specifically designed to address a number of key deficiencies in the previous model. In particular, it provides:

* The ability to create and use reusable UI controls that can encapsulate common functionality and thus reduce the amount of code that a page developer has to write.
* The ability for developers to cleanly structure their page logic in an orderly fashion (not "spaghetti code").
* The ability for development tools to provide strong WYSIWYG design support for pages (existing ASP code is opaque to tools).

ASP.NET Web Forms pages are text files with an .aspx file name extension. They can be deployed throughout an IIS virtual root directory tree. When a browser client requests .aspx resources, the ASP.NET runtime parses and compiles the target file into a .NET Framework class. This class can then be used to dynamically process incoming requests. (Note that the .aspx file is compiled only the first time it is accessed; the compiled type instance is then reused across multiple requests).

An ASP.NET page can be created simply by taking an existing HTML file and changing its file name extension to .aspx (no modification of code is required). For example, the following sample demonstrates a simple HTML page that collects a user's name and category preference and then performs a form postback to the originating page when a button is clicked:

ASP.NET provides syntax compatibility with existing ASP pages. This includes support for <% %> code render blocks that can be intermixed with HTML content within an .aspx file. These code blocks execute in a top-down manner at page render time.

**CODE-BEHIND WEB FORMS**

ASP.NET supports two methods of authoring dynamic pages. The first is the method shown in the preceding samples, where the page code is physically declared within the originating .aspx file. An alternative approach--known as the code-behind method--enables the page code to be more cleanly separated from the HTML content into an entirely separate file.

**INTRODUCTION TO ASP.NET SERVER CONTROLS**

In addition to (or instead of) using <% %> code blocks to program dynamic content, ASP.NET page developers can use ASP.NET server controls to program Web pages. Server controls are declared within an .aspx file using custom tags or intrinsic HTML tags that contain a **runat="server"** attributes value. Intrinsic HTML tags are handled by one of the controls in the **System.Web.UI.HtmlControls** namespace. Any tag that doesn't explicitly map to one of the controls is assigned the type of **System.Web.UI.HtmlControls.HtmlGenericControl**.

Server controls automatically maintain any client-entered values between round trips to the server. This control state is not stored on the server (it is instead stored within an **<input type="hidden">** form field that is round-tripped between requests). Note also that no client-side script is required.

In addition to supporting standard HTML input controls, ASP.NET enables developers to utilize richer custom controls on their pages. For example, the following sample demonstrates how the **<asp:adrotator>** control can be used to dynamically display rotating ads on a page.

1. ASP.NET Web Forms provide an easy and powerful way to build dynamic Web UI.
2. ASP.NET Web Forms pages can target any browser client (there are no script library or cookie requirements).
3. ASP.NET Web Forms pages provide syntax compatibility with existing ASP pages.
4. ASP.NET server controls provide an easy way to encapsulate common functionality.
5. ASP.NET ships with 45 built-in server controls. Developers can also use controls built by third parties.
6. ASP.NET server controls can automatically project both up level and down level HTML.
7. ASP.NET templates provide an easy way to customize the look and feel of list server controls.
8. ASP.NET validation controls provide an easy way to do declarative client or server data validation.

**4.3 C#.NET**

**ADO.NET OVERVIEW**

ADO.NET is an evolution of the ADO data access model that directly addresses user requirements for developing scalable applications. It was designed specifically for the web with scalability, statelessness, and XML in mind.

ADO.NET uses some ADO objects, such as the **Connection** and **Command** objects, and also introduces new objects. Key new ADO.NET objects include the **DataSet**, **DataReader**, and **DataAdapter**.

The important distinction between this evolved stage of ADO.NET and previous data architectures is that there exists an object -- the **DataSet** -- that is separate and distinct from any data stores. Because of that, the **DataSet** functions as a standalone entity. You can think of the DataSet as an always disconnected recordset that knows nothing about the source or destination of the data it contains. Inside a **DataSet**, much like in a database, there are tables, columns, relationships, constraints, views, and so forth.

A **DataAdapter** is the object that connects to the database to fill the **DataSet**. Then, it connects back to the database to update the data there, based on operations performed while the **DataSet** held the data. In the past, data processing has been primarily connection-based. Now, in an effort to make multi-tiered apps more efficient, data processing is turning to a message-based approach that revolves around chunks of information. At the center of this approach is the **DataAdapter**, which provides a bridge to retrieve and save data between a **DataSet** and its source data store. It accomplishes this by means of requests to the appropriate SQL commands made against the data store.

The XML-based **DataSet** object provides a consistent programming model that works with all models of data storage: flat, relational, and hierarchical. It does this by having no 'knowledge' of the source of its data, and by representing the data that it holds as collections and data types. No matter what the source of the data within the **DataSet** is, it is manipulated through the same set of standard APIs exposed through the **DataSet** and its subordinate objects.

While the **DataSet** has no knowledge of the source of its data, the managed provider has detailed and specific information. The role of the managed provider is to connect, fill, and persist the **DataSet** to and from data stores. The OLE DB and SQL Server .NET Data Providers (System.Data.OleDb and System.Data.SqlClient) that are part of the .Net Framework provide four basic objects: the **Command**, **Connection**, **DataReader** and **DataAdapter**. In the remaining sections of this document, we'll walk through each part of the **DataSet** and the OLE DB/SQL Server .NET Data Providers explaining what they are, and how to program against them.

The following sections will introduce you to some objects that have evolved, and some that are new. These objects are:

* **Connections**. For connection to and managing transactions against a database.
* **Commands**. For issuing SQL commands against a database.
* **DataReaders**. For reading a forward-only stream of data records from a SQL Server data source.
* **DataSets**. For storing, Remoting and programming against flat data, XML data and relational data.
* **DataAdapters**. For pushing data into a **DataSet**, and reconciling data against a database.

When dealing with connections to a database, there are two different options: SQL Server .NET Data Provider (System.Data.SqlClient) and OLE DB .NET Data Provider (System.Data.OleDb). In these samples we will use the SQL Server .NET Data Provider. These are written to talk directly to Microsoft SQL Server. The OLE DB .NET Data Provider is used to talk to any OLE DB provider (as it uses OLE DB underneath).

**Connections:**

Connections are used to 'talk to' databases, and are represented by provider-specific classes such as **SqlConnection**. Commands travel over connections and resultsets are returned in the form of streams which can be read by a **DataReader** object, or pushed into a **DataSet** object.

**Commands:**

Commands contain the information that is submitted to a database, and are represented by provider-specific classes such as **SqlCommand**. A command can be a stored procedure call, an UPDATE statement, or a statement that returns results. You can also use input and output parameters, and return values as part of your command syntax. The example below shows how to issue an INSERT statement against the **Northwind** database.

**DataReaders:**

The **DataReader** object is somewhat synonymous with a read-only/forward-only cursor over data. The **DataReader** API supports flat as well as hierarchical data. A **DataReader** object is returned after executing a command against a database. The format of the returned **DataReader** object is different from a recordset. For example, you might use the **DataReader** to show the results of a search list in a web page.

**DATASETS AND DATAADAPTERS:**

**DataSets**  
The **DataSet** object is similar to the ADO **Recordset** object, but more powerful, and with one other important distinction: the **DataSet** is always disconnected. The **DataSet** object represents a cache of data, with database-like structures such as tables, columns, relationships, and constraints. However, though a **DataSet** can and does behave much like a database, it is important to remember that **DataSet** objects do not interact directly with databases, or other source data. This allows the developer to work with a programming model that is always consistent, regardless of where the source data resides. Data coming from a database, an XML file, from code, or user input can all be placed into **DataSet** objects. Then, as changes are made to the **DataSet** they can be tracked and verified before updating the source data. The **GetChanges** method of the **DataSet** object actually creates a second **DatSet** that contains only the changes to the data. This **DataSet** is then used by a **DataAdapter** (or other objects) to update the original data source.

The **DataSet** has many XML characteristics, including the ability to produce and consume XML data and XML schemas. XML schemas can be used to describe schemas interchanged via WebServices. In fact, a **DataSet** with a schema can actually be compiled for type safety and statement completion.

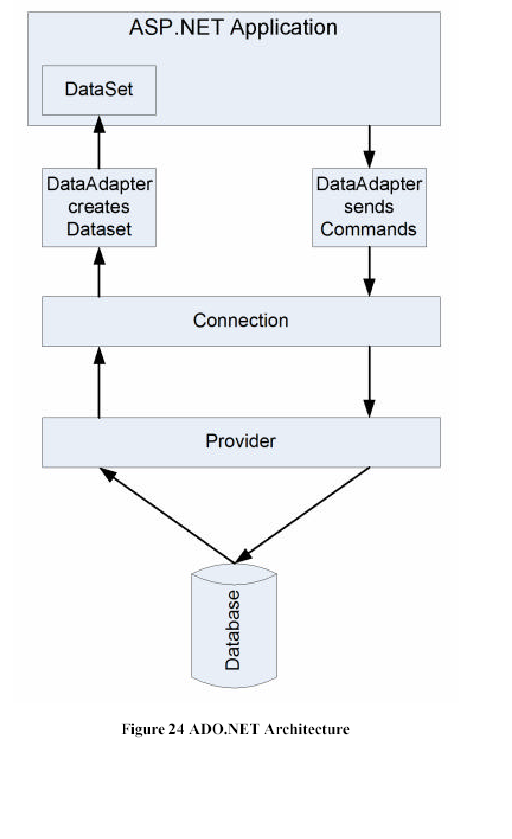
**DATAADAPTERS (OLEDB/SQL)**

The **DataAdapter** object works as a bridge between the **DataSet** and the source data. Using the provider-specific **SqlDataAdapter** (along with its associated **SqlCommand** and **SqlConnection**) can increase overall performance when working with a Microsoft SQL Server databases. For other OLE DB-supported databases, you would use the **OleDbDataAdapter** object and its associated **OleDbCommand** and **OleDbConnection** objects.

The **DataAdapter** object uses commands to update the data source after changes have been made to the **DataSet**. Using the **Fill** method of the **DataAdapter** calls the SELECT command; using the **Update** method calls the INSERT, UPDATE or DELETE command for each changed row. You can explicitly set these commands in order to control the statements used at runtime to resolve changes, including the use of stored procedures. For ad-hoc scenarios, a **CommandBuilder** object can generate these at run-time based upon a select statement. However, this run-time generation requires an extra round-trip to the server in order to gather required metadata, so explicitly providing the INSERT, UPDATE, and DELETE commands at design time will result in better run-time performance.

1. ADO.NET is the next evolution of ADO for the .Net Framework.
2. ADO.NET was created with n-Tier, statelessness and XML in the forefront. Two new objects, the **DataSet** and **DataAdapter**, are provided for these scenarios.
3. ADO.NET can be used to get data from a stream, or to store data in a cache for updates.
4. There is a lot more information about ADO.NET in the documentation.
5. Remember, you can execute a command directly against the database in order to do inserts, updates, and deletes. You don't need to first put data into a **DataSet** in order to insert, update, or delete it.
6. Also, you can use a **DataSet** to bind to the data, move through the data, and navigate data relationships

ADO.NET follows the below process, below Figure, to connect to the database and retrieve data to the application.

****

**4.4 SQL SERVER**

A database management, or DBMS, gives the user access to their data and helps them transform the data into information. Such database management systems include dBase, paradox, IMS, SQL Server and SQL Server. These systems allow users to create, update and extract information from their database.

A database is a structured collection of data. Data refers to the characteristics of people, things and events. SQL Server stores each data item in its own fields. In SQL Server, the fields relating to a particular person, thing or event are bundled together to form a single complete unit of data, called a record (it can also be referred to as raw or an occurrence). Each record is made up of a number of fields. No two fields in a record can have the same field name.

During an SQL Server Database design project, the analysis of your business needs identifies all the fields or attributes of interest. If your business needs change over time, you define any additional fields or change the definition of existing fields.

**SQL SERVER TABLES**

SQL Server stores records relating to each other in a table. Different tables are created for the various groups of information. Related tables are grouped together to form a database.

**PRIMARY KEY**

Every table in SQL Server has a field or a combination of fields that uniquely identifies each record in the table. The Unique identifier is called the Primary Key, or simply the Key. The primary key provides the means to distinguish one record from all other in a table. It allows the user and the database system to identify, locate and refer to one particular record in the database.

**RELATIONAL DATABASE**

Sometimes all the information of interest to a business operation can be stored in one table. SQL Server makes it very easy to link the data in multiple tables. Matching an employee to the department in which they work is one example. This is what makes SQL Server a relational database management system, or RDBMS. It stores data in two or more tables and enables you to define relationships between the table and enables you to define relationships between the tables.

**FOREIGN KEY**

When a field is one table matches the primary key of another field is referred to as a foreign key. A foreign key is a field or a group of fields in one table whose values match those of the primary key of another table.

**REFERENTIAL INTEGRITY**

Not only does SQL Server allow you to link multiple tables, it also maintains consistency between them. Ensuring that the data among related tables is correctly matched is referred to as maintaining referential integrity.

**DATA ABSTRACTION**

A major purpose of a database system is to provide users with an abstract view of the data. This system hides certain details of how the data is stored and maintained. Data abstraction is divided into three levels.

**Physical level**: This is the lowest level of abstraction at which one describes how the data are actually stored.

**Conceptual Level**: At this level of database abstraction all the attributed and what data are actually stored is described and entries and relationship among them.

**View level**: This is the highest level of abstraction at which one describes only part of the database.

**ADVANTAGES OF RDBMS**

* Redundancy can be avoided
* Inconsistency can be eliminated
* Data can be Shared
* Standards can be enforced
* Security restrictions ca be applied
* Integrity can be maintained
* Conflicting requirements can be balanced
* Data independence can be achieved.

**DISADVANTAGES OF DBMS**

A significant disadvantage of the DBMS system is cost. In addition to the cost of purchasing of developing the software, the hardware has to be upgraded to allow for the extensive programs and the workspace required for their execution and storage. While centralization reduces duplication, the lack of duplication requires that the database be adequately backed up so that in case of failure the data can be recovered.

**FEATURES OF SQL SERVER (RDBMS**)

SQL SERVER is one of the leading database management systems (DBMS) because it is the only Database that meets the uncompromising requirements of today’s most demanding information systems. From complex decision support systems (DSS) to the most rigorous online transaction processing (OLTP) application, even application that require simultaneous DSS and OLTP access to the same critical data, SQL Server leads the industry in both performance and capability

SQL SERVER is a truly portable, distributed, and open DBMS that delivers unmatched performance, continuous operation and support for every database.

SQL SERVER RDBMS is high performance fault tolerant DBMS which is specially designed for online transactions processing and for handling large database application.

SQL SERVER with transactions processing option offers two features which contribute to very high level of transaction processing throughput, which are

* The row level lock manager

**ENTERPRISE WIDE DATA SHARING**

The unrivaled portability and connectivity of the SQL SERVER DBMS enables all the systems in the organization to be linked into a singular, integrated computing resource.

**PORTABILITY**

SQL SERVER is fully portable to more than 80 distinct hardware and operating systems platforms, including UNIX, MSDOS, OS/2, Macintosh and dozens of proprietary platforms. This portability gives complete freedom to choose the database sever platform that meets the system requirements.

**OPEN SYSTEMS**

SQL SERVER offers a leading implementation of industry –standard SQL. SQL Server’s open architecture integrates SQL SERVER and non –SQL SERVER DBMS with industries most comprehensive collection of tools, application, and third party software products SQL Server’s Open architecture provides transparent access to data from other relational database and even non-relational database.

**DISTRIBUTED DATA SHARING**

SQL Server’s networking and distributed database capabilities to access data stored on remote server with the same ease as if the information was stored on a single local computer. A single SQL statement can access data at multiple sites. You can store data where system requirements such as performance, security or availability dictate.

**UNMATCHED PERFORMANCE**

The most advanced architecture in the industry allows the SQL SERVER DBMS to deliver unmatched performance.

**SOPHISTICATED CONCURRENCY CONTROL**

Real World applications demand access to critical data. With most database Systems application becomes “contention bound” – which performance is limited not by the CPU power or by disk I/O, but user waiting on one another for data access . SQL Server employs full, unrestricted row-level locking and contention free queries to minimize and in many cases entirely eliminates contention wait times.

**NO I/O BOTTLENECKS**

SQL Server’s fast commit groups commit and deferred write technologies dramatically reduce disk I/O bottlenecks. While some database write whole data block to disk at commit time, SQL Server commits transactions with at most sequential log file on disk at commit time, On high throughput systems, one sequential writes typically group commit multiple transactions. Data read by the transaction remains as shared memory so that other transactions may access that data without reading it again from disk. Since fast commits write all data necessary to the recovery to the log file, modified blocks are written back to the database independently of the transaction commit, when written from memory to disk.

**5. SYSTEM REQUIREMENTS SPECIFICATIONS**

System requirements are expressed in a software requirement document. The Software requirement specification (SRS) is the official statement of what is required of the system developers. This requirement document includes the requirements definition and the requirement specification. The software requirement document is not a design document. It should set out what the system should do without specifying how it should be done. The requirement set out in this document is complete and consistent.

The software specification document satisfies the following:-

* It specifies the external system behaviors.
* It specifies constraints on the implementation.
* It is easy to change.
* It serves as reference tool for system maintainers.
* It record forethought about the life cycle of the system.
* It characterizes acceptable response to undesired events.

**5.1 User Class and Characteristics:**

* General public
* Customers
* Administrator
* General public can use the system to see the product,their prices and quantity available.
* General user can not buy the products.
* Customers are using for viewing and buying the products.
* Customer can also write feedbacks for products and services
* Administrators can add,edit & delete products.and provide services to the customer.
* Administrator can see the daily sell. Can also see the the feedback given by the customer.
* Administrator maintaining the deliveries.

**5.2 Functional Requirements:**

The System must provide following functionalities—

* Keeping records of admission of customers.
* keeping the records of products.
* keeping the daily sell .
* Storing the feedback given by the customer.
* keeping details about the product it is delivered

or not. etc.

* Storing the items selected by the customer in the temporary storage.

**5.3 Performance Requirements:**

In order to maintain an acceptable speed at maximum number of uploads allowed from a particular customer will be any number of users can access the system at any time. Also connections to the servers will be based on the criteria of attributes of the user like his location, and server will be working whole 24X 7 times.

**5.4 Non Functional Requirements:**

Following Non-functional requirements will be there in the Insurance on internet:

* Secure access of confidential data (customer’s details).
* 24 X 7 availability.
* Better component design to get better performance at peak time.

Flexible service based architecture will be highly desirable for future extension Non functional requirements define system properties and constraints It arise through user needs, because of budget constraints or organizational policies, or due to the external factors such as safety regulations, privacy registration and so on.

Various other Non-functional requirements are:

1. Security

2. Reliability

3. Maintainability

4. Portability

5. Extensibility

6. Reusability

7. Application Affinity/Compatibility

8. Resource Utilization

**5.5 External Interface Requirements:**

**5.5.1 User Interface:**

User of the system will be provided with the Graphical user interface, there is no command line interface for any functions of the product. The user will get 2 pages

1. Login page followed by Password

**5.5.2 Hardware Interface**:

Hardware requirements for Insurance on internet will be same for both the parties which are follows:

Processor: - Pentium I or above.

RAM: - 128 MB or above.

HD: - 20 GB or above.

NIC: - For each party

**5.5.3 Software Interface**:-

Software required to make working of product is:-

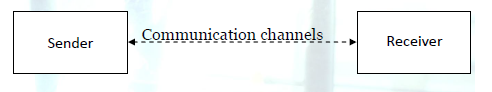
Front end- visual studio 2012

Back end- sql server 2012

**5.5.4 Communication Interfaces**

The two parties should be connected through either by LAN or WAN for the communication.

Communication channels



**5.6 General Constraints, Assumptions, Dependencies, Guidelines:**

**5.6.1 General Constraints**

The interface will be in English only.

The system is working for single server.

Sender

Receiver

There is no maintainability or backup so availability will get affected.

The system is a single user system.

GUI features available.

**5.6.2 Assumptions and Dependencies**

The product does require back-end database server MySQL for storing the username and password for different types of user of the system as well as various databases regarding various insurance information.

**Assumptions:**

User must be trained for basic computer functionalities.

User must have the basic knowledge of English

The system must be able to respond to database software within reasonable time.

**SYSTEM DESIGN SPECIFICATION**

1. **SYSTEM DESIGN SPECIFICATION**

**6.1 ARCHITECTURAL DESIGN**

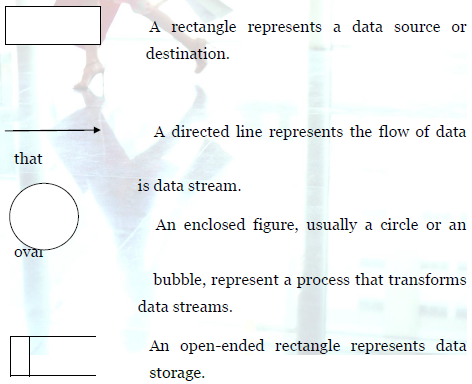
**6.1.1 DATA FLOW DIAGRAMS**:

Data flow diagrams (DFD) was first developed by LARRY CONSTANTINE as way representing system requirements in a graphical form; this lead to modular design. A DFD describes what data flow (logical) rather than how they are processed, so it does not depend on hardware, software, data structure or file organization. It is also known as ‘bubble chart’.

A Data Flow Diagrams is a structured analysis and design tool that can be used for flowcharting in place of, or in association with, information-oriented and process-oriented systems flowcharts. A DFD is a network that describes the flow of data and the processes that change, or transform, data throughout a system. This network is constructed by using a set of symbols that do not imply a physical implementation. It has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design phase that functionality decomposes the requirement specifications down to the lowest level of detail.

20

The symbols used to prepare DFD do not imply a physical implementation, a DFD can be considered to an abstract of the logic of an information-oriented or a process-oriented system flow-chart. For these reasons DFDs are often referred to as logical data flow diagrams. The four basic symbols used to construct data flow diagrams are shown below:

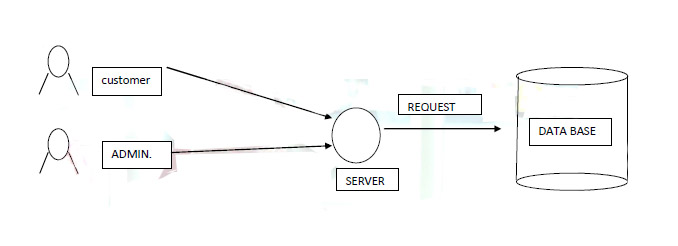


These are symbols that represent data flows, data sources, data transformations and data storage. The points at which data are transformed are represented by enclosed figures, usually circles, which are called nodes. The principle processes that take place at nodes are:

1. Combining data streams

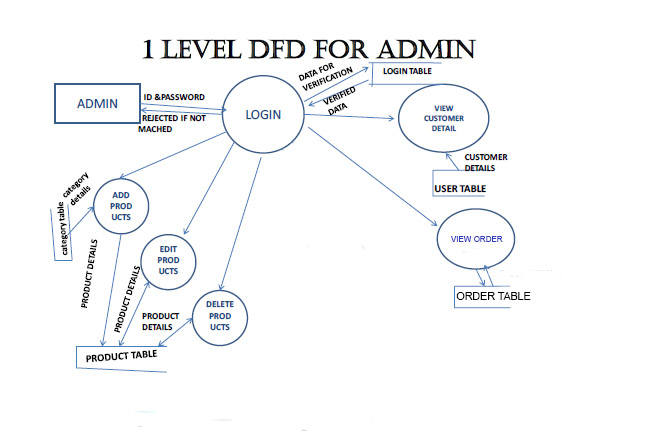
2. Splitting data streams

3. Modifying data streams

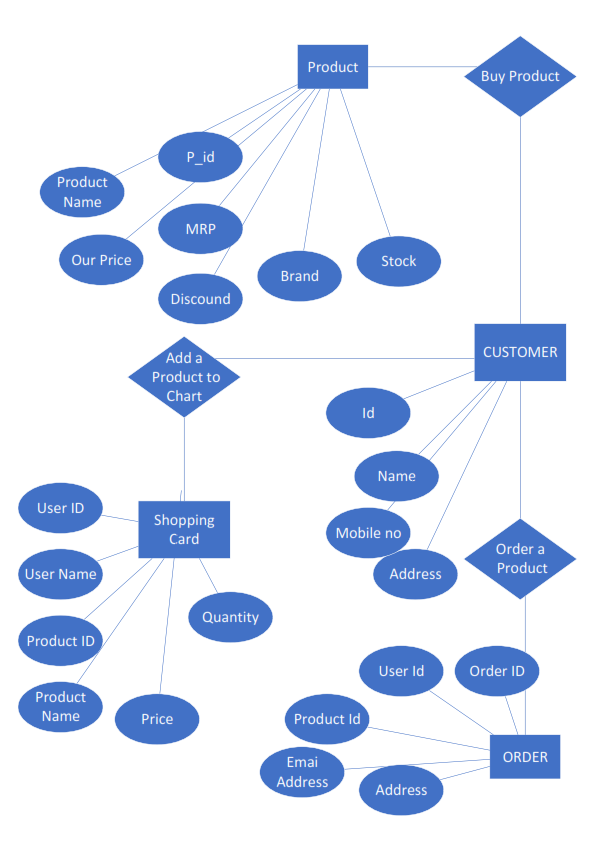


CAD(CONTEXT ANALYSIS DIAGRAM)



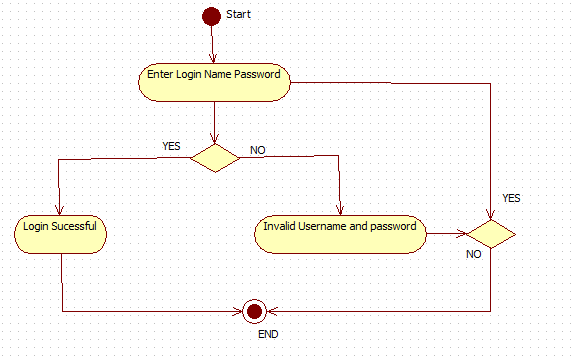


**6.2 ER Diagram**

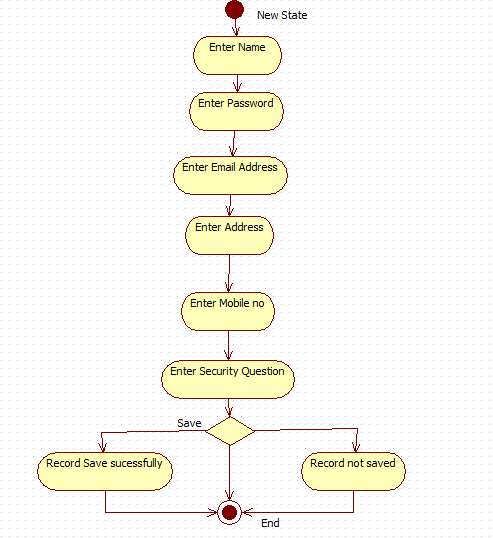
****

**6.3 Activity Diagram**

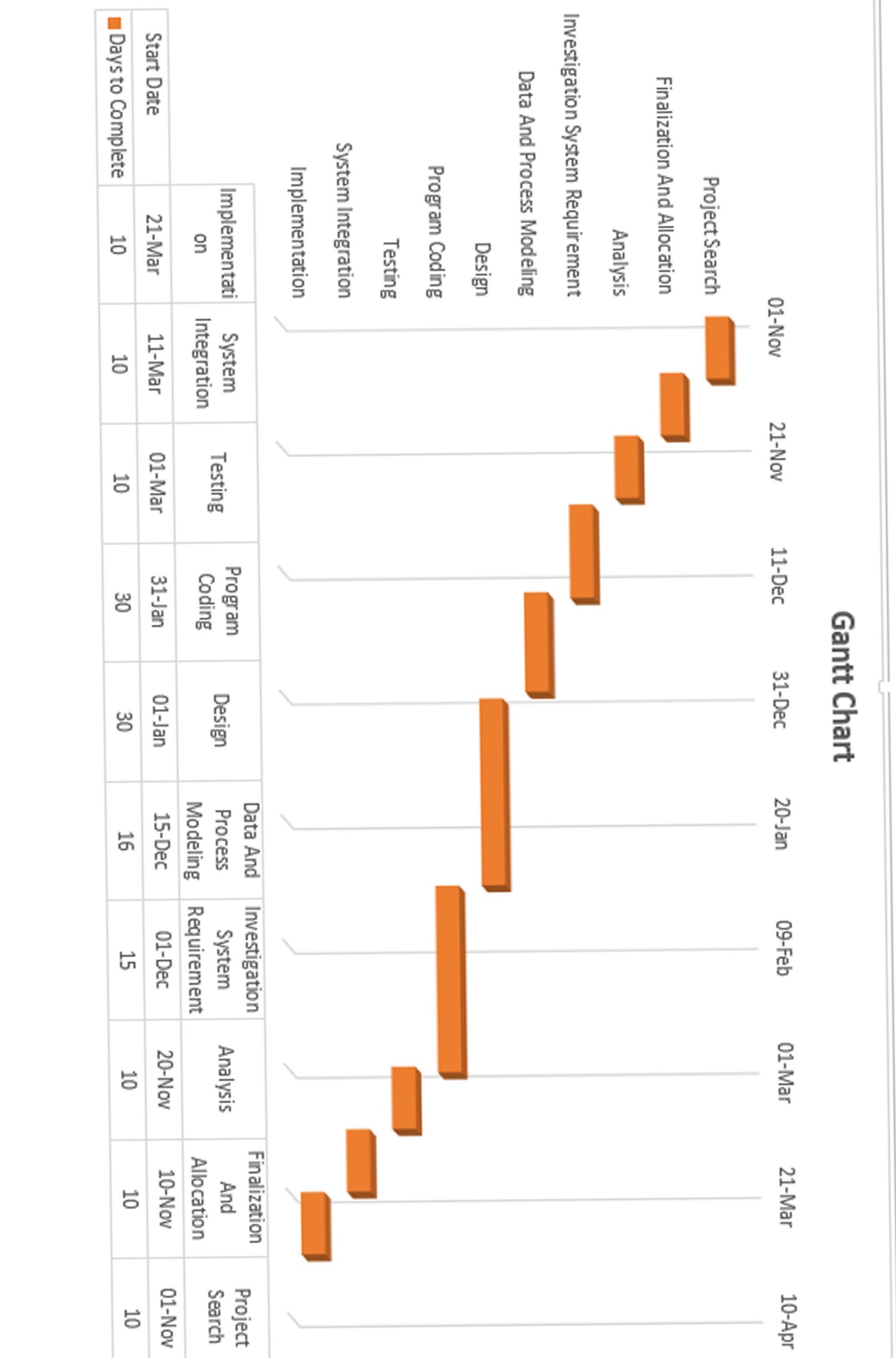
1:Activity Diagram for login :-

****

2: Activity diagram for Add new account :-



**6.4 Gantt Chart**

****

**Testing**

**7. TESTING**

Software Testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test, with respect to the context in which it is intended to operate. Software Testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs. It can also be stated as the process of validating and verifying that a software program/application/product meets the business and technical requirements that guided its design and development, so that it works as expected and can be implemented with the same characteristics.

Software Testing, depending on the testing method employed, can be implemented at any time in the development process, however the most test effort is employed after the requirements have been defined and coding process has been completed.

**7.1 Unit Testing:**

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing we have is white box oriented and some modules the steps are conducted in parallel.

**1. WHITE BOX TESTING**

This type of testing ensures that

* All independent paths have been exercised at least once
* All logical decisions have been exercised on their true and false sides
* All loops are executed at their boundaries and within their operational bounds
* All internal data structures have been exercised to assure their validity.

To follow the concept of white box testing we have tested each form .we have created independently to verify that Data flow is correct, All conditions are exercised to check their validity, All loops are executed on their boundaries.

**2. BASIC PATH TESTING**

Established technique of flow graph with Cyclomatic complexity was used to derive test cases for all the functions. The main steps in deriving test cases were:

Use the design of the code and draw correspondent flow graph.

Determine the Cyclomatic complexity of resultant flow graph, using formula:

V(G)=E-N+2 or

V(G)=P+1 or

V(G)=Number Of Regions

Where V(G) is Cyclomatic complexity,

E is the number of edges,

N is the number of flow graph nodes,

P is the number of predicate nodes.

Determine the basis of set of linearly independent paths.

**3. CONDITIONAL TESTING**

In this part of the testing each of the conditions were tested to both true and false aspects. And all the resulting paths were tested. So that each path that may be generate on particular condition is traced to uncover any possible errors.

**4. DATA FLOW TESTING**

This type of testing selects the path of the program according to the location of definition and use of variables. This kind of testing was used only when some local variable were declared. The *definition-use chain* method was used in this type of testing. These were particularly useful in nested statements.

**5. LOOP TESTING**

In this type of testing all the loops are tested to all the limits possible. The following exercise was adopted for all loops:

* All the loops were tested at their limits, just above them and just below them.
* All the loops were skipped at least once.
* For nested loops test the inner most loop first and then work outwards.
* For concatenated loops the values of dependent loops were set with the help of connected loop.
* Unstructured loops were resolved into nested loops or concatenated loops and tested as above.

Each unit has been separately tested by the development team itself and all the input have been validated.

.

**7.2 Integration Testing**

Integration testing, also known as integration and testing (I&T), is a software development process which program units are combined and tested as groups in multiple ways. In this context, a unit is defined as the smallest testable part of an application. Integration testing can expose problems with the interfaces among program components before trouble occurs in real-world program execution. Integration testing is a component of Extreme Programming (XP), a pragmatic method of software development that takes a meticulous approach to building a product by means of continual testing and revision.

There are two major ways of carrying out an integration test, called the bottom-up method and the top-down method. Bottom-up integration testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds. In top-down integration testing, the highest-level modules are tested first and progressively lower-level modules are tested after that. In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing.

**7.3 Validation testing**

At the validation level, testing focuses on user visible actions and user recognizable output from the system. Validations testing is said to be successful when software functions in a manner that can be reasonably expected by the customer. Two types of validation testing

* **Alpha testing** is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.
* **Beta testing** comes after alpha testing. Versions of the software, known as beta version, are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users
* **Gray box testing** Grey box testing is the combination of black box and white box testing. Intention of this testing is to find out defects related to bad design or bad implementation of the system.it is used for web application

**Coding**

**8. Coding**

**Login.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Home.Master" AutoEventWireup="true" CodeBehind="Login.aspx.cs" Inherits="Online\_Shopping\_Backup.Login" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<div class="row" >

<div class="col-lg-3"></div>

<div class="col-lg-4">

<h3>Sign in to continue to Shopping</h3>

<div class="form-group" >

<label>Username</label>

<asp:TextBox CssClass="form-control" ID="txtusername" placeholder="Enter Username" runat="server" ></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ControlToValidate="txtusername" ErrorMessage="Name is Required"></asp:RequiredFieldValidator>

</div>

<div class="form-group">

<label> Password</label>

<asp:TextBox ID="txtpassword" placeholder="Enter Password" CssClass="form-control" runat="server" TextMode="Password"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server" ControlToValidate="txtpassword" ErrorMessage="Password is Required"></asp:RequiredFieldValidator>

</div>

<%-- Height="38px" Width="105px"--%>

<asp:Button ID="btn\_submit" CssClass="btn btn-primary" runat="server" Text="Login" OnClick="btn\_submit\_Click" />

<asp:Label runat="server" ID="lblerror"></asp:Label>

<asp:HyperLink CssClass="btn btn-primary" ID="hyper\_link\_forget\_password" NavigateUrl= "~/Pages/forget\_password.aspx" Text="Forget Password" runat="server"></asp:HyperLink>

</div>

</div>

<div class="col-lg-5">

</div>

</asp:Content>

**Forgetpassword.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Home.Master" AutoEventWireup="true" CodeBehind="forget\_password.aspx.cs" Inherits="Online\_Shopping\_Backup.Pages.forget\_password" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<div class="container">

<div class="row"">

<div class="col-lg-3">

</div>

<div class="col-lg-4" >

<div class="form-group">

<h2 style="padding-top:0px;padding-bottom:20px;margin:0px">Having trouble signing in ?</h2>

<label>Username</label>

<asp:TextBox CssClass="form-control" ID="txt\_username" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ErrorMessage="Username is required" ControlToValidate="txt\_username"></asp:RequiredFieldValidator>

</div>

<div class="form-group">

<label>Question</label>

<asp:DropDownList ID="DropDownList1" runat="server">

<asp:ListItem>What is your favorite color</asp:ListItem>

<asp:ListItem>What is the name of your school</asp:ListItem>

<asp:ListItem>What is the name of your first School teacher</asp:ListItem>

</asp:DropDownList>

</div>

<div class="form-group">

<label>Answer</label>

<asp:TextBox CssClass="form-control" ID="txt\_answer" runat="server"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server" ErrorMessage="Answere is requird" ControlToValidate="txt\_answer"></asp:RequiredFieldValidator>

</div>

<asp:Button CssClass="btn btn-primary" ID="btn\_submit" runat="server" Text="Submit" OnClick="btn\_submit\_Click" />

</div>

<div class="col-lg-5">

</div>

</div>

</div>

</asp:Content>

**View\_Order.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Home.Master" AutoEventWireup="true" CodeBehind="View\_Order.aspx.cs" Inherits="Online\_Shopping\_Backup.View\_Order" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" CellPadding="4" DataKeyNames="order\_id" DataSourceID="SqlDataSource1" ForeColor="#333333" GridLines="None" Height="191px" Width="672px">

<AlternatingRowStyle BackColor="White" />

<Columns>

<asp:BoundField DataField="order\_id" HeaderText="order\_id" InsertVisible="False" ReadOnly="True" SortExpression="order\_id" />

<asp:BoundField DataField="user\_id" HeaderText="user\_id" SortExpression="user\_id" />

<asp:BoundField DataField="user\_name" HeaderText="user\_name" SortExpression="user\_name" />

<asp:BoundField DataField="product\_id" HeaderText="product\_id" SortExpression="product\_id" />

<asp:BoundField DataField="email\_address" HeaderText="email\_address" SortExpression="email\_address" />

<asp:BoundField DataField="mobile\_no" HeaderText="mobile\_no" SortExpression="mobile\_no" />

<asp:BoundField DataField="payment\_type" HeaderText="payment\_type" SortExpression="payment\_type" />

<asp:BoundField DataField="address" HeaderText="address" SortExpression="address" />

</Columns>

<EditRowStyle BackColor="#2461BF" />

<FooterStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />

<HeaderStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />

<PagerStyle BackColor="#2461BF" ForeColor="White" HorizontalAlign="Center" />

<RowStyle BackColor="#EFF3FB" />

<SelectedRowStyle BackColor="#D1DDF1" Font-Bold="True" ForeColor="#333333" />

<SortedAscendingCellStyle BackColor="#F5F7FB" />

<SortedAscendingHeaderStyle BackColor="#6D95E1" />

<SortedDescendingCellStyle BackColor="#E9EBEF" />

<SortedDescendingHeaderStyle BackColor="#4870BE" />

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:ShoppingConnectionString %>" SelectCommand="SELECT \* FROM [main\_order]"></asp:SqlDataSource>

</asp:Content>

**View\_user\_Database.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Home.Master" AutoEventWireup="true" CodeBehind="View\_user\_Database.aspx.cs" Inherits="Online\_Shopping\_Backup.View\_user\_Database" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" DataKeyNames="name" DataSourceID="SqlDataSource1" CellPadding="4" ForeColor="#333333" GridLines="None" Height="198px" Width="718px">

<AlternatingRowStyle BackColor="White" />

<Columns>

<asp:BoundField DataField="id" HeaderText="User ID" InsertVisible="False" ReadOnly="True" SortExpression="id" />

<asp:BoundField DataField="name" HeaderText="Username" ReadOnly="True" SortExpression="name" />

<asp:BoundField DataField="role" HeaderText="Role" SortExpression="role" />

<asp:BoundField DataField="address" HeaderText="Address" SortExpression="address" />

<asp:BoundField DataField="mobile\_no" HeaderText="Mobile No" SortExpression="mobile\_no" />

<asp:BoundField DataField="Email" HeaderText="Email" SortExpression="Email" />

</Columns>

<EditRowStyle BackColor="#2461BF" />

<FooterStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />

<HeaderStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />

<PagerStyle BackColor="#2461BF" ForeColor="White" HorizontalAlign="Center" />

<RowStyle BackColor="#EFF3FB" />

<SelectedRowStyle BackColor="#D1DDF1" Font-Bold="True" ForeColor="#333333" />

<SortedAscendingCellStyle BackColor="#F5F7FB" />

<SortedAscendingHeaderStyle BackColor="#6D95E1" />

<SortedDescendingCellStyle BackColor="#E9EBEF" />

<SortedDescendingHeaderStyle BackColor="#4870BE" />

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:ShoppingConnectionString %>" SelectCommand="SELECT [name], [id], [role], [address], [mobile\_no], [Email] FROM [login\_data]"></asp:SqlDataSource>

</asp:Content>

**order\_summary.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Home.Master" AutoEventWireup="true" CodeBehind="order\_summary.aspx.cs" Inherits="Online\_Shopping\_Backup.order\_summary" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<div class="container">

<div class="row">

<div class="col-lg-2">

</div>

<div class="col-lg-8">

<h1>Order Summary</h1>

<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" DataSourceID="SqlDataSource1" GridLines="None" Height="341px" Width="663px">

<Columns>

<asp:TemplateField>

<ItemTemplate>

<asp:Image ID="Image1" runat="server" ImageUrl='<%#Eval("image") %>' />

</ItemTemplate>

</asp:TemplateField>

<asp:TemplateField HeaderText="Product name">

<ItemTemplate>

<div style="height:30px;width:400px">

<asp:Label ID="Label1" runat="server" Text='<%#Eval("Product\_name") %>'></asp:Label>

</div>

<br />

</ItemTemplate>

</asp:TemplateField>

<asp:BoundField DataField="price" HeaderText="price" SortExpression="price" />

<asp:BoundField DataField="quantity" HeaderText="quantity" SortExpression="quantity" />

</Columns>

</asp:GridView>

</div>

<div class="col-lg-2">

</div>

</div>

<div class="row">

<div class="col-lg-2">

</div>

<div class="col-lg-5">

<h2>Select Shipping Address And Payment option</h2>

<div class="form-group">

<label> Name </label>

<asp:TextBox CssClass="form-control" ID="txt\_username" runat="server"></asp:TextBox>

</div>

<div class="form-group">

<label>Email Address </label> <asp:TextBox ID="txt\_email" CssClass="form-control" runat="server"></asp:TextBox>

</div>

<div class="form-group">

<label> Mobile no </label><asp:TextBox CssClass="form-control" ID="txt\_mobile\_no" runat="server"></asp:TextBox>

</div>

<div class="form-group">

<label> Address</label> <asp:TextBox ID="txt\_Address" runat="server" TextMode="MultiLine" Rows="8" Columns="63"> </asp:TextBox>

</div>

<div class="form-group">

<label>Payment Option </label>

<asp:DropDownList ID="ddl\_payment\_option" runat="server">

<asp:ListItem Value="Cash">Cash on delivery</asp:ListItem>

</asp:DropDownList>

</div>

<asp:Button ID="Button1" CssClass="btn btn-primary" runat="server" Text="Book" OnClick="Button1\_Click"/>

</div>

<div class="col-lg-5">

</div>

</div>

</div>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:ShoppingConnectionString%>" SelectCommand="SELECT [Product\_name], [image], [price], [quantity] FROM [cart\_product] WHERE ([user\_id] = @user\_id) ORDER BY [price]">

<SelectParameters>

<asp:CookieParameter CookieName="u\_id" Name="user\_id" Type="Int32" />

</SelectParameters>

</asp:SqlDataSource>

</asp:Content>

**order\_single.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Home.Master" AutoEventWireup="true" CodeBehind="order\_single.aspx.cs" Inherits="Online\_Shopping\_Backup.order\_single" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<asp:GridView ID="GridView1" runat="server" DataSourceID="SqlDataSource1" AutoGenerateColumns="False" GridLines="None" Height="207px" Width="677px">

<Columns>

<asp:TemplateField>

<ItemTemplate>

<asp:ImageButton ID="ImageButton1" runat="server" ImageUrl='<%# Eval("Brand\_image") %>' />

</ItemTemplate>

</asp:TemplateField>

<asp:BoundField DataField="Product\_name" HeaderText="Product name" SortExpression="Product\_name" />

<asp:BoundField DataField="Our\_Prize" HeaderText="Total" SortExpression="Our\_Prize" />

</Columns>

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:ShoppingConnectionString %>" SelectCommand="SELECT [Product\_name], [Brand\_image], [Our\_Prize] FROM [main\_product] WHERE ([P\_ID] = @P\_ID)">

<SelectParameters>

<asp:QueryStringParameter Name="P\_ID" QueryStringField="product\_id" Type="Int32" />

</SelectParameters>

</asp:SqlDataSource>

<h2>Select Shipping Address And Payment option</h2>

<div class="row">

<div class="col-lg-2">

</div>

<div class="col-lg-4">

<div class="form-group">

<label>Name </label>

<asp:TextBox CssClass="form-control" ID="txt\_username" runat="server"></asp:TextBox>

</div>

<div class="form-group">

<label>Email Address </label>

<asp:TextBox CssClass="form-control" ID="txt\_email" runat="server"></asp:TextBox></td>

</div>

<div class="form-group">

<label>Mobile no </label>

<asp:TextBox CssClass="form-control" ID="txt\_mobile\_no" runat="server"></asp:TextBox></td>

</div>

<div class="form-group">

<label>Address </label>

<asp:TextBox CssClass="form-control" ID="txt\_Address" runat="server" TextMode="MultiLine" Rows="8" Columns="35"></asp:TextBox>

</div>

<div class="form-group">

<label>Payment Option </label>

<asp:DropDownList ID="ddl\_payment\_option" runat="server">

<asp:ListItem Value="Cash">Cash on delivery</asp:ListItem>

</asp:DropDownList>

</div>

<asp:Button ID="Button1" CssClass="btn btn-primary" runat="server" Text="Book" OnClick="Button1\_Click" />

</div>

<div class="col-lg-6">

</div>

</div>

</asp:Content>

**Home.Master**

<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="Home.master.cs" Inherits="Online\_Shopping\_Backup.Home" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title></title>

<style>

.btn\_img {

position: absolute;

top: 0px;

}

#top\_list ul {

margin: 0px;

padding: 0px;

}

#top\_list ul li {

float: right;

display: block;

color: white;

padding: 0px 10px;

margin: 0px;

list-style-type: none;

width: auto;

}

#top\_list li a {

color: white;

text-decoration: none;

}

#top\_list li a:hover {

color: white;

}

#list\_menu li a {

color: white;

text-decoration: none;

}

#list\_menu li a:hover {

color: blue;

background-color: white;

}

</style>

<script src="../Scripts/bootstrap.min.js" type="text/javascript"></script>

<link rel="stylesheet" type="text/css" href="CSS/bootstrap.css" />

<link href="css/skins/light\_blue.css" rel="stylesheet" type="text/css" />

<link href="css/dcmegamenu.css" runat="server" rel="stylesheet" type="text/css" />

<script type="text/javascript" src="../Scripts/jquery-1.9.1.min.js"></script>

<script type='text/javascript' src='../js/jquery.hoverIntent.minified.js'></script>

<script type='text/javascript' src='../js/jquery.dcmegamenu.1.3.3.js'></script>

</head>

<body style="background-color:#f7f5ee" >

<div style="max-width:100%;overflow-x:hidden;overflow-y:hidden">

<form id="form1" runat="server" >

<%-- <%--This is Header--%>

<script type="text/javascript">

$(document).ready(function ($) {

$('#mega-menu-1').dcMegaMenu({

rowItems: '3',

speed: 0,

effect: 'slide',

event: 'click',

fullWidth: true

});

$('#mega-menu-2').dcMegaMenu({

rowItems: '1',

speed: 'fast',

effect: 'fade',

event: 'click'

});

$('#mega-menu-3').dcMegaMenu({

rowItems: '2',

speed: 'fast',

effect: 'fade'

});

$('#mega-menu-4').dcMegaMenu({

rowItems: '3',

speed: 'fast',

effect: 'fade'

});

$('#mega-menu-5').dcMegaMenu({

rowItems: '4',

speed: 'fast',

effect: 'fade'

});

$('#mega-menu-6').dcMegaMenu({

rowItems: '3',

speed: 'slow',

effect: 'slide',

event: 'click',

fullWidth: true

});

$('#mega-menu-7').dcMegaMenu({

rowItems: '3',

speed: 'fast',

effect: 'slide'

});

$('#mega-menu-8').dcMegaMenu({

rowItems: '3',

speed: 'fast',

effect: 'fade'

});

$('#mega-menu-9').dcMegaMenu({

rowItems: '3',

speed: 'fast',

effect: 'fade'

});

});

</script>

<header style="background-color: #005387; height: 140px">

<div class="row" style="padding-top:5px">

<div class="col-lg-8">

</div>

<div class="col-lg-4">

<div id="top\_list">

<ul>

<li>

<asp:ImageButton ID="btn\_submit" runat="server" ImageUrl="~/Images/Admin\_task.png" OnClick="ImageButton1\_Click" Visible="false" />

</li>

<li>

<asp:HyperLink ID="hyper\_id" NavigateUrl="~/Pages/Registration.aspx" runat="server" Text="SignUp"></asp:HyperLink>

</li>

<li>

<asp:LoginStatus ID="LoginStatus1" runat="server" LogoutAction="Redirect" LogoutPageUrl="~/Pages/Thankyou.aspx" />

</li>

<li>

<asp:Label ID="lbl\_name" runat="server"></asp:Label>

</li>

<li>

<asp:Button ID="btn\_Edit\_Profile" runat="server" Visible="false" Text="Edit Profile" Height="15px" Width="80px" OnClick="btn\_Edit\_Profile\_Click" BackColor="#005387" BorderColor="#005387" BorderStyle="None" />

</li>

</ul>

</div>

</div>

</div>

<div class="row" style="height: 70px">

<div style="padding-top: 20px">

<div class="col-lg-3" style="padding-left:60px">

<asp:ImageButton CssClass="img\_home\_button" ID="img\_home\_button" runat="server" ImageUrl="~/Images/Site\_name.png" PostBackUrl="~/Pages/Home.aspx" Width="213px" />

</div>

<div class="col-lg-7">

<asp:TextBox ID="txtSearch" runat="server" Width="469px" Height="30px" placeholder="Search for a Product e.g Moto e,Micromax"></asp:TextBox>

<asp:ImageButton ID="btn\_Img\_search" CssClass="btn\_img" runat="server" ImageUrl="~/Images/search.png" Height="30px" OnClick="btn\_Img\_search\_Click" />

</div>

<div class="col-lg-2">

<asp:ImageButton ID="img\_shopping\_chart" runat="server" ImageUrl="~/Images/Cart.png" OnClick="img\_shopping\_chart\_Click" Height="32px" Width="105px" />

</div>

</div>

</div>

<div class="row" style="padding-top:25px" >

<div class="col-lg-12">

<div class="demo-container" style="margin: 0px; padding: 0px;">

<div class="light-blue" style="margin: 0px; padding: 0px">

<ul id="mega-menu-8" class="mega-menu" style="margin: 0px; padding: 0px">

<li>

<asp:HyperLink ID="Hyper\_Mobiles" Text="Mobiles" runat="server" NavigateUrl="~/Pages/Product\_title.aspx?type\_of\_product=Mobiles"></asp:HyperLink>

<ul>

<li>

<asp:HyperLink ID="Hyper\_Mobiles\_motorola" Text="Motorola" runat="server" NavigateUrl="~/Pages/single\_page.aspx?type\_of\_product=Mobiles&Brand=Motorola" ></asp:HyperLink>

</li>

<li>

<asp:HyperLink ID="Hyper\_Mobiles\_Mi" runat="server" Text="Mi" NavigateUrl="~/Pages/single\_page.aspx?type\_of\_product=Mobiles&Brand=Mi" ></asp:HyperLink>

</li>

<li>

<asp:HyperLink ID="Hyper\_Mobiles\_Samsung" runat="server" Text="Samsung" NavigateUrl="~/Pages/single\_page.aspx?type\_of\_product=Mobiles&Brand=Samsung" ></asp:HyperLink>

</li>

<li>

<asp:HyperLink ID="Hyper\_Mobiles\_Sony" runat="server" Text="Sony" NavigateUrl="~/Pages/single\_page.aspx?type\_of\_product=Mobiles&Brand=Sony" ></asp:HyperLink>

</li>

<li>

<asp:HyperLink ID="Hyper\_Mobiles\_Micromax" runat="server" Text="Micromax" NavigateUrl="~/Pages/single\_page.aspx?type\_of\_product=Mobiles&Brand=Micromax" ></asp:HyperLink>

</li>

</ul>

</li>

<li>

<asp:HyperLink ID="Hyper\_Tablet" Text="Tablet" runat="server" NavigateUrl="~/Pages/Product\_title.aspx?type\_of\_product=Tablet" ></asp:HyperLink>

<ul>

<li>

<asp:HyperLink ID="Hyper\_Tablet\_Apple" runat="server" Text="Apple" NavigateUrl="~/Pages/single\_page.aspx?type\_of\_product=Tablet&Brand=Apple" ></asp:HyperLink>

</li>

<li>

<asp:HyperLink ID="Hyper\_Tablet\_Lenovo" runat="server" Text="Lenovo" NavigateUrl="~/Pages/single\_page.aspx?type\_of\_product=Tablet&Brand=Lenovo" ></asp:HyperLink>

</li>

<li>

<asp:HyperLink ID="Hyper\_Tablet\_Assus" runat="server" Text="Asus" NavigateUrl="~/Pages/single\_page.aspx?type\_of\_product=Tablet&Brand=Asus" ></asp:HyperLink>

</li>

</ul>

</li>

<li>

<asp:HyperLink ID="Hyper\_Laptops" Text="Laptops" runat="server" NavigateUrl="~/Pages/Product\_title.aspx?type\_of\_product=Laptops" ></asp:HyperLink>

<ul>

<li>

<asp:HyperLink ID="Hyper\_Laptop\_HP" runat="server" Text="HP" NavigateUrl="~/Pages/single\_page.aspx?type\_of\_product=Laptops&Brand=HP" ></asp:HyperLink>

</li>

<li>

<asp:HyperLink ID="Hyper\_Laptop\_Acer" runat="server" Text="Acer" NavigateUrl="~/Pages/single\_page.aspx?type\_of\_product=Laptops&Brand=Acer" ></asp:HyperLink>

</li>

<li>

<asp:HyperLink ID="Hyper\_Laptop\_Dell" runat="server" Text="Dell" NavigateUrl="~/Pages/single\_page.aspx?type\_of\_product=Laptops&Brand=Dell" ></asp:HyperLink>

</li>

</ul>

</li>

<li>

<asp:HyperLink ID="Hyper\_Computer\_Accessories" Text="Computer Accessories" runat="server" NavigateUrl="~/Pages/Product\_title.aspx?type\_of\_product=Computer\_Accessories" ></asp:HyperLink>

<ul>

<li>

<asp:HyperLink ID="Hyper\_Computer\_Accessories\_Mouse" runat="server" Text="Mouse" NavigateUrl="~/Pages/single\_page.aspx?type\_of\_product=Computer\_Accessories&Brand=Mouse" ></asp:HyperLink>

</li>

<li>

<asp:HyperLink ID="Hyper\_Computer\_Accessories\_Pendrives" runat="server" Text="Pendrives" NavigateUrl="~/Pages/single\_page.aspx?type\_of\_product=Computer\_Accessories&Brand=Pendrives" ></asp:HyperLink>

</li>

</ul>

</li>

<li>

<asp:HyperLink ID="Hyper\_Televisions" Text="Televisions" runat="server" NavigateUrl="~/Pages/Product\_title.aspx?type\_of\_product=Televisions" ></asp:HyperLink>

<ul>

<li>

<asp:HyperLink ID="Hyper\_Televisions\_Sony" runat="server" Text="Sony" NavigateUrl="~/Pages/single\_page.aspx?type\_of\_product=Televisions&Brand=Sony" ></asp:HyperLink>

</li>

<li>

<asp:HyperLink ID="Hyper\_Televisions\_LG" runat="server" Text="LG" NavigateUrl="~/Pages/single\_page.aspx?type\_of\_product=Televisions&Brand=LG" ></asp:HyperLink>

</li>

</ul>

</li>

</ul>

</div>

</div>

</div>

</div>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:ShoppingConnectionString %>" SelectCommand="SELECT [Type\_of\_product], [Brand] FROM [main\_product]"></asp:SqlDataSource>

</header>

<div class="container" >

<asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">

</asp:ContentPlaceHolder>

</div>

<%-- <div class="container" style="margin-top:10px" >

</div>--%>

</form>

</div>

</body>

</html>

**Edit\_Profile.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Home.Master" AutoEventWireup="true" CodeBehind="Edit\_Profile.aspx.cs" Inherits="Online\_Shopping\_Backup.Edit\_Profile" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False" DataKeyNames="name" DataSourceID="SqlDataSource1" Height="130px" Width="599px" CellPadding="4" ForeColor="#333333" GridLines="None">

<AlternatingRowStyle BackColor="White" />

<Columns>

<asp:CommandField ShowEditButton="True" />

<asp:BoundField DataField="mobile\_no" HeaderText="Mobile No" SortExpression="mobile\_no" />

<asp:BoundField DataField="Email" HeaderText="Email" SortExpression="Email" />

<asp:BoundField DataField="address" HeaderText="Address" SortExpression="address" />

<asp:BoundField DataField="password" Visible="false" HeaderText="Password" SortExpression="password" />

<asp:BoundField DataField="name" HeaderText="Username" ReadOnly="True" SortExpression="name" />

<asp:TemplateField HeaderText="Password">

<EditItemTemplate>

<asp:TextBox ID="txt\_password" runat="server" TextMode="Password" Text='<%#Bind("password") %>'></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ErrorMessage="Password is Required" ControlToValidate="txt\_password"></asp:RequiredFieldValidator>

</EditItemTemplate>

</asp:TemplateField>

</Columns>

<EditRowStyle BackColor="#2461BF" />

<FooterStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />

<HeaderStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />

<PagerStyle BackColor="#2461BF" ForeColor="White" HorizontalAlign="Center" />

<RowStyle BackColor="#EFF3FB" />

<SelectedRowStyle BackColor="#D1DDF1" Font-Bold="True" ForeColor="#333333" />

<SortedAscendingCellStyle BackColor="#F5F7FB" />

<SortedAscendingHeaderStyle BackColor="#6D95E1" />

<SortedDescendingCellStyle BackColor="#E9EBEF" />

<SortedDescendingHeaderStyle BackColor="#4870BE" />

</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:ShoppingConnectionString %>" DeleteCommand="DELETE FROM [login\_data] WHERE [name] = @name" InsertCommand="INSERT INTO [login\_data] ([name], [password], [address], [mobile\_no], [Email]) VALUES (@name, @password, @address, @mobile\_no, @Email)" SelectCommand="SELECT [name], [password], [address], [mobile\_no], [Email] FROM [login\_data] WHERE ([id] = @id)" UpdateCommand="UPDATE [login\_data] SET [password] = @password, [address] = @address, [mobile\_no] = @mobile\_no, [Email] = @Email WHERE [name] = @name">

<DeleteParameters>

<asp:Parameter Name="name" Type="String" />

</DeleteParameters>

<InsertParameters>

<asp:Parameter Name="name" Type="String" />

<asp:Parameter Name="password" Type="String" />

<asp:Parameter Name="address" Type="String" />

<asp:Parameter Name="mobile\_no" Type="String" />

<asp:Parameter Name="Email" Type="String" />

</InsertParameters>

<SelectParameters>

<asp:CookieParameter CookieName="u\_id" Name="id" Type="Int32" />

</SelectParameters>

<UpdateParameters>

<asp:Parameter Name="password" Type="String" />

<asp:Parameter Name="address" Type="String" />

<asp:Parameter Name="mobile\_no" Type="String" />

<asp:Parameter Name="Email" Type="String" />

<asp:Parameter Name="name" Type="String" />

</UpdateParameters>

</asp:SqlDataSource>

</asp:Content>

**Home.aspx**

<%@ Page Title="" Language="C#" MasterPageFile="~/Home.Master" AutoEventWireup="true" CodeBehind="Home.aspx.cs" Inherits="Online\_Shopping\_Backup.Pages.Home" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<%-- <link rel="stylesheet" type="text/css" href="../CSS/css\_container.css" />

<link rel="stylesheet" type="text/css" href="../CSS/Main\_Style.css" />--%>

<%-- Javascript file --%>

<script type="text/javascript" src="../Scripts/jquery-1.9.1.min.js"></script>

<script type="text/javascript" src="../Scripts/jquery.bxslider.min.js"></script>

<%-- CSS FILE --%>

<link rel="stylesheet" type="text/css" href="../CSS/jquery.bxslider.css" />

<link rel="stylesheet" type="text/css" href="../CSS/css\_container.css" />

<script type="text/javascript">

$(document).ready(function () {

$('.bxslider').bxSlider();

});

</script>

<style type="text/css">

#description {

height:150px;

width:150px;

}

#slider\_postion {

position:absolute;

top:0px;

left:100px;

}

</style>

<section>

<div class="row">

<div class="col-lg-2">

</div>

<div class="col-lg-8">

<div id="slider">

<ul class="bxslider">

<li><img src="../Images/1.PNG" /></li>

<li><img src="../Images/2.PNG" /></li>

<li><img src="../Images/3.PNG" /></li>

<li><img src="../Images/4.PNG" /></li>

</ul>

</div>

</div>

<div class="col-lg-2">

</div>

<%--this is slider --%>

</div>

<%--this is deal of the day--%>

<div class="row">

<div class="col-lg-2">

</div>

<div class="col-lg-8">

<asp:DataList ID="DataList1" runat="server" DataSourceID="SqlDataSource1" RepeatColumns="4" RepeatDirection="Horizontal" Width="783px" Height="305px" DataKeyField="P\_ID">

<ItemTemplate>

<asp:ImageButton ID="ImageButton1" runat="server" ImageUrl='<%# Eval("Brand\_image") %>' PostBackUrl='<%#"Mobile\_description.aspx?P\_ID="+Eval("P\_ID")%>' />

<br />

<div id="product\_name">

<asp:Label ID="Label1" runat="server" Text='<%# Eval("Product\_name") %>'></asp:Label>

</div>

<div id="price">

<del>

<asp:Label ID="MRPLabel" runat="server" Text='<%# Eval("MRP") %>' />

<br />

</del>

Rs:<asp:Label ID="Our\_PrizeLabel" runat="server" Text='<%# Eval("Our\_Prize") %>' />

<br />

</div>

</ItemTemplate>

</asp:DataList>

</div>

<div class="col-lg-2">

<asp:AdRotator ID="AdRotator1" runat="server" DataSourceID="XmlDataSource1" />

<asp:XmlDataSource ID="XmlDataSource1" runat="server" DataFile="~/Pages/add\_source.xml"></asp:XmlDataSource>

</div>

<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%$ ConnectionStrings:ShoppingConnectionString %>" SelectCommand="SELECT [P\_ID], [Brand\_image], [MRP], [Product\_name], [Our\_Prize] FROM [main\_product] WHERE ([best\_offer] = @best\_offer)">

<SelectParameters>

<asp:Parameter DefaultValue="1" Name="best\_offer" Type="Int32" />

</SelectParameters>

</asp:SqlDataSource>

</div>

</section>

</asp:Content>

**Login.aspx.cs**

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;

using System.Web.Security;

namespace Online\_Shopping\_Backup

{

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

{

SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["ShoppingConnectionString"].ConnectionString);

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

{

}

protected void btn\_submit\_Click(object sender, EventArgs e)

{

SqlCommand cmd = new SqlCommand("select \* from login\_data", conn);

conn.Open();

if (conn.State == ConnectionState.Open)

{

SqlDataReader sdr = cmd.ExecuteReader();

while (sdr.Read())

{

if (sdr[1].ToString().Equals(txtusername.Text) && sdr[2].ToString().Equals(txtpassword.Text))

{

Response.Cookies["uname"].Value = sdr[1].ToString();

Response.Cookies["u\_id"].Value = sdr[0].ToString();

Session["Role"] = sdr[3].ToString();

FormsAuthentication.RedirectFromLoginPage(txtusername.Text, false);

Response.Redirect("~/Pages/Home.aspx");

}

else

{

lblerror.Text = "Please enter write username and password";

}

}

}

else

{

}

}

}

}

**Data\_Insert.aspx.cs**

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;

namespace Online\_Shopping\_Backup

{

public partial class Data\_Insert : System.Web.UI.Page

{

SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["ShoppingConnectionString"].ConnectionString);

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

{

}

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

{

string Product\_path = Server.MapPath("~/Product/");

string productimg\_path = Server.MapPath("~/Product\_Img/");

FileUpload1\_brand\_image.SaveAs(Product\_path + FileUpload1\_brand\_image.FileName);

FileUpload2\_img\_path\_1.SaveAs(productimg\_path + FileUpload2\_img\_path\_1.FileName);

FileUpload3\_img\_path\_2.SaveAs(productimg\_path + FileUpload3\_img\_path\_2.FileName);

FileUpload4\_img\_path\_3.SaveAs(productimg\_path + FileUpload4\_img\_path\_3.FileName);

FileUpload5\_img\_path\_4.SaveAs(productimg\_path + FileUpload5\_img\_path\_4.FileName);

string brand\_file\_name = "~/Product/" + FileUpload1\_brand\_image.FileName;

string product\_img1 = "~/Product\_Img/" + FileUpload2\_img\_path\_1.FileName;

string product\_img2 = "~/Product\_Img/" + FileUpload3\_img\_path\_2.FileName;

string product\_img3 = "~/Product\_Img/" + FileUpload4\_img\_path\_3.FileName;

string product\_img4 = "~/Product\_Img/" + FileUpload5\_img\_path\_4.FileName;

String ddlbrand = ddlbrand\_name.SelectedValue;

String ddltypeofproduct = ddl\_type\_of\_product.SelectedValue;

String query = "insert into main\_product values(@product\_name,@Mrp,@Our\_Prise,@Discount,@Brand,@Brand\_img,@Type\_of\_product,@Imgpath1,@Imgpath2,@Imgpath3,@Imgpath4,@Detail,@Stock,@new\_arrival,@best\_seller,@best\_offer)";

conn.Open();

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@product\_name", txt\_product\_name.Text);

cmd.Parameters.AddWithValue("@Mrp", txt\_MRP.Text);

cmd.Parameters.AddWithValue("@Our\_Prise", txt\_OUR\_prise.Text);

cmd.Parameters.AddWithValue("@Discount", txt\_Discount.Text);

cmd.Parameters.AddWithValue("@Brand", ddlbrand);

cmd.Parameters.AddWithValue("@Brand\_img", brand\_file\_name);

cmd.Parameters.AddWithValue("@Type\_of\_product", ddltypeofproduct);

cmd.Parameters.AddWithValue("@Imgpath1", product\_img1);

cmd.Parameters.AddWithValue("@Imgpath2", product\_img2);

cmd.Parameters.AddWithValue("@Imgpath3", product\_img3);

cmd.Parameters.AddWithValue("@Imgpath4", product\_img4);

cmd.Parameters.AddWithValue("@Detail", txt\_detail.Text);

cmd.Parameters.AddWithValue("@Stock", txt\_stock.Text);

cmd.Parameters.AddWithValue("@new\_arrival", txt\_new\_arrival.Text);

cmd.Parameters.AddWithValue("@best\_seller", txt\_best\_seller.Text);

cmd.Parameters.AddWithValue("@best\_offer", txt\_best\_offer.Text);

cmd.ExecuteNonQuery();

conn.Close();

Response.Redirect("~/Data\_Insert.aspx");

}

}

}

**Registration.aspx.cs**

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;

namespace Online\_Shopping\_Backup.Pages

{

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

{

SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["ShoppingConnectionString"].ConnectionString);

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

{

}

protected void btn\_submit\_Click(object sender, EventArgs e)

{

try

{

SqlCommand cmd = new SqlCommand("insert into login\_data values(@name,@password,@role,@address,@mobile\_no,@email,@Answer)", conn);

cmd.Parameters.AddWithValue("@name", txt\_name.Text);

cmd.Parameters.AddWithValue("@password", txt\_password.Text);

cmd.Parameters.AddWithValue("@role", "user");

cmd.Parameters.AddWithValue("@address", txt\_address.Text);

cmd.Parameters.AddWithValue("@mobile\_no", txt\_mobile\_no.Text);

cmd.Parameters.AddWithValue("@email", txt\_Email\_address.Text);

cmd.Parameters.AddWithValue("@Answer", txt\_answer.Text);

conn.Open();

cmd.ExecuteNonQuery();

Response.Redirect("~/Pages/Home.aspx");

}

catch {

lbl\_error.Text = "Chose another username this username exist";

}

}

}

}

**forget\_password.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Net.Mail;

using System.Data;

using System.Data.SqlClient;

using System.Configuration;

namespace Online\_Shopping\_Backup.Pages

{

public partial class forget\_password : System.Web.UI.Page

{

SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["ShoppingConnectionString"].ConnectionString);

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

{

}

protected void btn\_submit\_Click(object sender, EventArgs e)

{

SqlCommand cmd = new SqlCommand("select \* from login\_data where name='"+txt\_username.Text+"' ", conn);

conn.Open();

string answer="";

string email = "";

string password = "";

if (conn.State == ConnectionState.Open)

{

SqlDataReader sdr = cmd.ExecuteReader();

while (sdr.Read())

{

answer = sdr["Answer"].ToString();

email= sdr["Email"].ToString();

password = sdr["password"].ToString();

}

if(txt\_answer.Text.ToUpper().Equals(answer.ToUpper()))

{

try

{

//create the mail message

MailMessage mail = new MailMessage();

//set the addresses

mail.From = new MailAddress("sunil64500@gmail.com");

mail.To.Add(email);

//set the content

mail.Subject = "Save your Password";

mail.Body = "This is your password " + password + " Please save it carefully";

//send the message

SmtpClient smtp = new SmtpClient();

smtp.Send(mail);

}

catch

{

}

}

conn.Close();

}

}

}

}

**add\_admin.aspx.cs**

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;

namespace Online\_Shopping\_Backup

{

public partial class add\_admin : System.Web.UI.Page

{

SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["ShoppingConnectionString"].ConnectionString);

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

{

}

protected void btn\_submit\_Click(object sender, EventArgs e)

{

SqlCommand cmd = new SqlCommand("insert into login\_data values(@name,@password,@role,@address,@mobile\_no,@email)", conn);

string admin = ddl\_admin.SelectedValue;

cmd.Parameters.AddWithValue("@name", txt\_name.Text);

cmd.Parameters.AddWithValue("@password", txt\_password.Text);

cmd.Parameters.AddWithValue("@role", admin);

cmd.Parameters.AddWithValue("@address", txt\_address.Text);

cmd.Parameters.AddWithValue("@mobile\_no", txt\_mobile\_no.Text);

cmd.Parameters.AddWithValue("@email", txt\_Email\_address.Text);

conn.Open();

cmd.ExecuteNonQuery();

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

}

}

}

**add\_cart.aspx.cs**

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;

using System.Diagnostics.Eventing.Reader;

namespace Online\_Shopping\_Backup

{

public partial class add\_cart : System.Web.UI.Page

{

SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["ShoppingConnectionString"].ConnectionString);

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

{

}

protected void btn\_place\_order\_Click(object sender, ImageClickEventArgs e)

{

int u\_id = Convert.ToInt32(Request.Cookies["u\_id"].Value);

string product\_id = "";

conn.Open();

string query = "SELECT STUFF(( SELECT ','+ [product\_id] AS [text()] FROM [Shopping].[dbo].[cart\_product] WHERE [user\_id] = '" + u\_id + "' FOR XML PATH('') ), 1, 1,'')";

SqlCommand cmd = new SqlCommand(query, conn);

var result = cmd.ExecuteScalar();

if (!Convert.IsDBNull(result))

product\_id = (string)result;

conn.Close();

Response.Redirect("~/order\_summary.aspx?product\_id=" + product\_id);

}

}

}

**Admin.aspx.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Online\_Shopping\_Backup

{

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

{

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

{

}

protected void btn\_delete\_Product\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Delete\_Product.aspx");

}

protected void btn\_Add\_Product\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Data\_Insert.aspx");

}

protected void btn\_Add\_Admin\_Click(object sender, EventArgs e)

{

Response.Redirect("~/add\_admin.aspx");

}

protected void btn\_Edit\_Product\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Edit\_Product.aspx");

}

protected void btn\_view\_order\_Click(object sender, EventArgs e)

{

Response.Redirect("~/View\_Order.aspx");

}

protected void btn\_user\_database\_Click(object sender, EventArgs e)

{

Response.Redirect("~/View\_user\_Database.aspx");

}

}

}

**Home.master.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace Online\_Shopping\_Backup

{

public partial class Home : System.Web.UI.MasterPage

{

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

{

try

{

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

lbl\_name.Text = "Welcome " + Request.Cookies["uname"].Value;

if (Role != null)

{

if (Role == "admin")

{

btn\_submit.Visible = true;

}

else

{

btn\_submit.Visible = false;

}

}

else

{

btn\_submit.Visible = false;

}

if (Role != null)

{

if (Role == "user" || Role == "admin")

{

btn\_Edit\_Profile.Visible = true;

}

else

{

btn\_Edit\_Profile.Visible = false;

}

}

else

{

btn\_Edit\_Profile.Visible = false;

}

}

catch

{

}

//navigation();

}

protected void img\_shopping\_chart\_Click(object sender, ImageClickEventArgs e)

{

string u\_id = "";

try

{

u\_id = Request.Cookies["u\_id"].Value;

}

catch

{ }

Response.Redirect("~/add\_cart.aspx?u\_id=" + u\_id);

}

protected void ImageButton1\_Click(object sender, ImageClickEventArgs e)

{

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

}

protected void btn\_Img\_search\_Click(object sender, ImageClickEventArgs e)

{

Response.Redirect("~/Pages/Search\_result.aspx?Search=" + txtSearch.Text);

}

protected void btn\_Edit\_Profile\_Click(object sender, EventArgs e)

{

Response.Redirect("~/Edit\_Profile.aspx");

}

}

}

**order\_single.aspx.cs**

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;

namespace Online\_Shopping\_Backup

{

public partial class order\_single : System.Web.UI.Page

{

SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["ShoppingConnectionString"].ConnectionString);

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

{

DataTable dt = new DataTable();

conn.Open();

SqlCommand sqlCmd = new SqlCommand("SELECT \* from login\_data WHERE id = @id", conn);

SqlDataAdapter sqlDa = new SqlDataAdapter(sqlCmd);

string user\_id = Request.Cookies["u\_id"].Value;

sqlCmd.Parameters.AddWithValue("@id", user\_id);

sqlDa.Fill(dt);

if (dt.Rows.Count > 0)

{

txt\_username.Text = dt.Rows[0]["name"].ToString(); //Where ColumnName is the Field from the DB that you want to display

txt\_Address.Text = dt.Rows[0]["address"].ToString();

txt\_email.Text = dt.Rows[0]["Email"].ToString();

txt\_mobile\_no.Text = dt.Rows[0]["mobile\_no"].ToString();

}

conn.Close();

}

protected void Button1\_Click(object sender, EventArgs e)

{

string product\_id = Request.QueryString["product\_id"];

string user\_id = Request.Cookies["u\_id"].Value;

string payment\_type = ddl\_payment\_option.SelectedValue;

string query = "insert into main\_order values(@user\_id,@user\_name,@product\_id,@email\_address,@mobile\_no,@payment\_type,@address)";

conn.Open();

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@user\_id", user\_id);

cmd.Parameters.AddWithValue("@user\_name", txt\_username.Text);

cmd.Parameters.AddWithValue("@product\_id", product\_id);

cmd.Parameters.AddWithValue("@email\_address", txt\_email.Text);

cmd.Parameters.AddWithValue("@mobile\_no", txt\_mobile\_no.Text);

cmd.Parameters.AddWithValue("@payment\_type", payment\_type);

cmd.Parameters.AddWithValue("@address", txt\_Address.Text);

cmd.ExecuteNonQuery();

conn.Close();

Response.Redirect("~/Pages/Thank\_you.aspx");

}

}

}

**order\_summary.aspx.cs**

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;

namespace Online\_Shopping\_Backup

{

public partial class order\_summary : System.Web.UI.Page

{

SqlConnection conn = new SqlConnection(ConfigurationManager.ConnectionStrings["ShoppingConnectionString"].ConnectionString);

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

{

DataTable dt = new DataTable();

conn.Open();

SqlCommand sqlCmd = new SqlCommand("SELECT \* from login\_data WHERE id = @id", conn);

SqlDataAdapter sqlDa = new SqlDataAdapter(sqlCmd);

string user\_id = Request.Cookies["u\_id"].Value;

sqlCmd.Parameters.AddWithValue("@id", user\_id);

sqlDa.Fill(dt);

if (dt.Rows.Count > 0)

{

txt\_username.Text = dt.Rows[0]["name"].ToString(); //Where ColumnName is the Field from the DB that you want to display

txt\_Address.Text = dt.Rows[0]["address"].ToString();

txt\_email.Text = dt.Rows[0]["Email"].ToString();

txt\_mobile\_no.Text = dt.Rows[0]["mobile\_no"].ToString();

}

conn.Close();

}

protected void Button1\_Click(object sender, EventArgs e)

{

string product\_id = Request.QueryString["product\_id"];

string user\_id = Request.Cookies["u\_id"].Value;

string payment\_type = ddl\_payment\_option.SelectedValue;

string query = "insert into main\_order values(@user\_id,@user\_name,@product\_id,@email\_address,@mobile\_no,@payment\_type,@address)";

conn.Open();

SqlCommand cmd = new SqlCommand(query, conn);

cmd.Parameters.AddWithValue("@user\_id",user\_id);

cmd.Parameters.AddWithValue("@user\_name",txt\_username.Text);

cmd.Parameters.AddWithValue("@product\_id",product\_id);

cmd.Parameters.AddWithValue("@email\_address",txt\_email.Text);

cmd.Parameters.AddWithValue("@mobile\_no",txt\_mobile\_no.Text);

cmd.Parameters.AddWithValue("@payment\_type",payment\_type);

cmd.Parameters.AddWithValue("@address",txt\_Address.Text);

cmd.ExecuteNonQuery();

conn.Close();

Response.Redirect("~/Pages/Thank\_you.aspx");

}

}

}

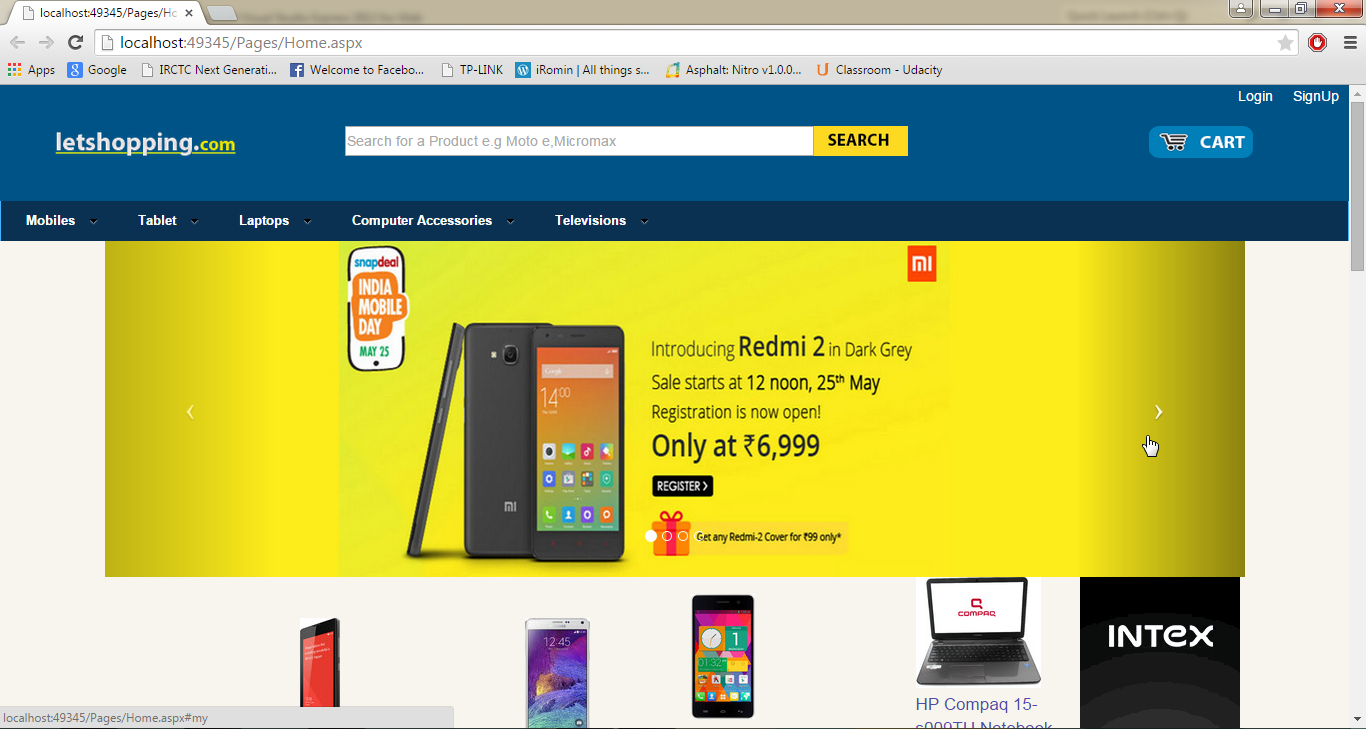
**Screen shots**

**9.Screen Shots**

1. **Home page**

**Form name:** **Home.aspx**

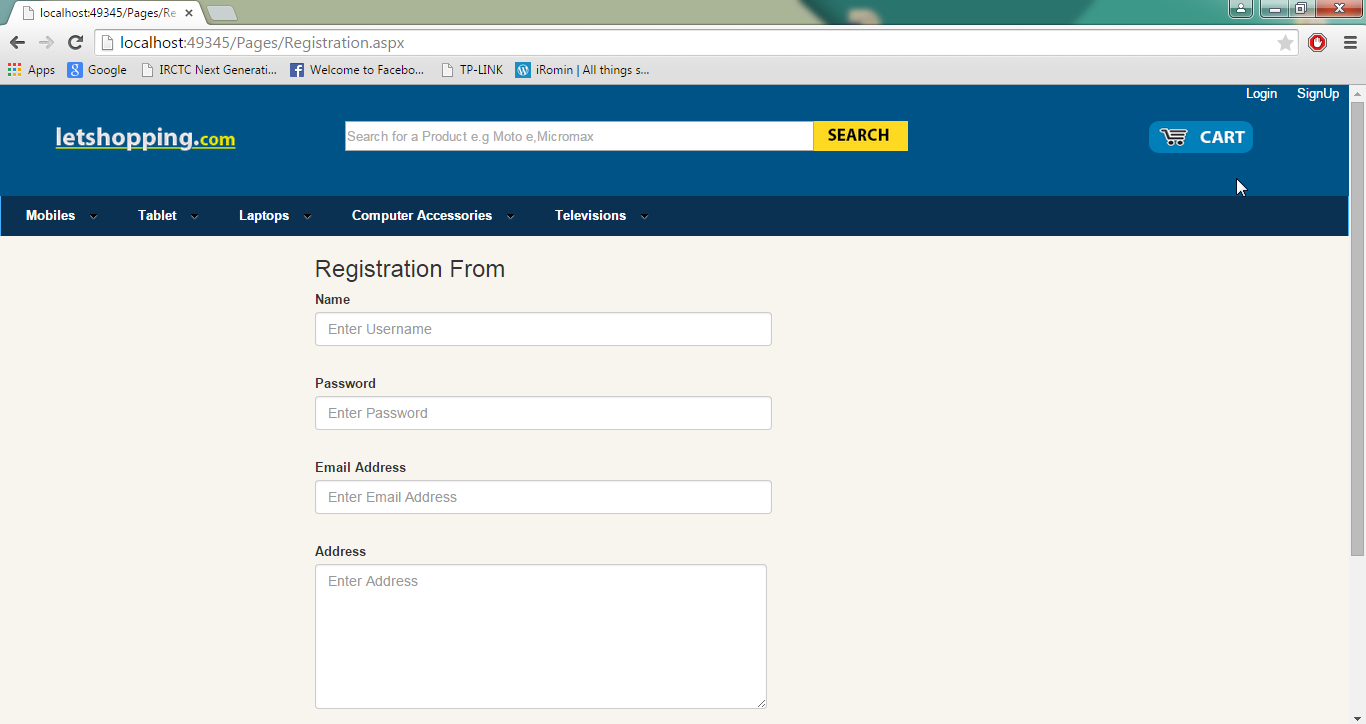
**Description : Use to show homepage**

****

1. **Registration Page**

**Form name:** **Registration.aspx**

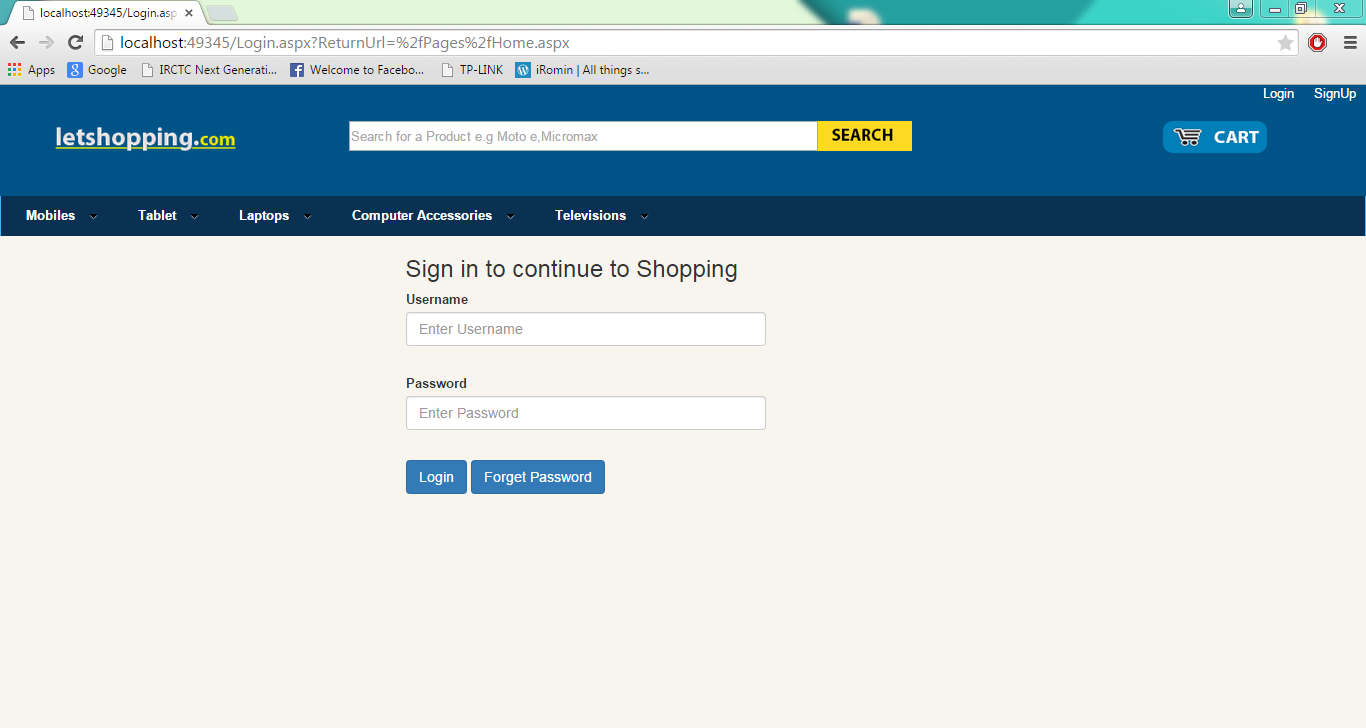
**Desription :Use to Register the user**

****

**3.Login Page**

**Form name:** **Login.aspx**

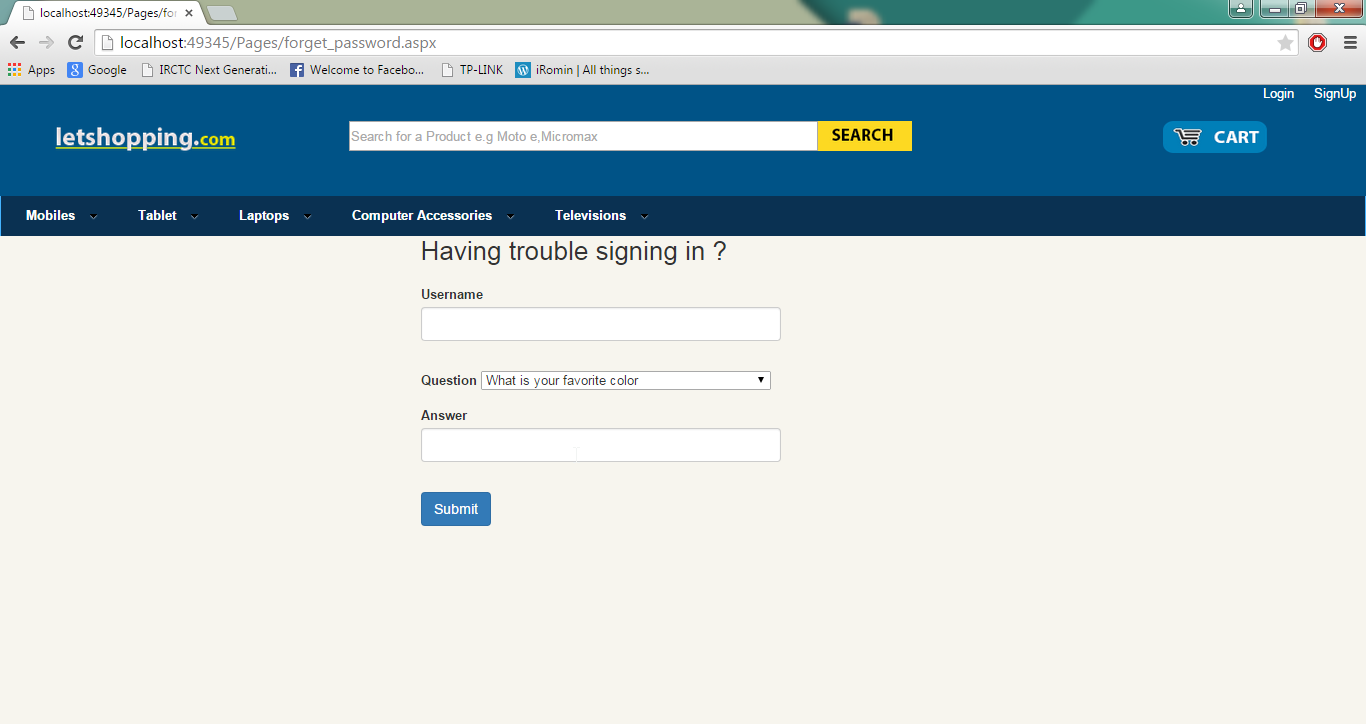
**Description : Use to login in Website**

****

**4.Forget Password**

**Form name:** **forget\_password.aspx**

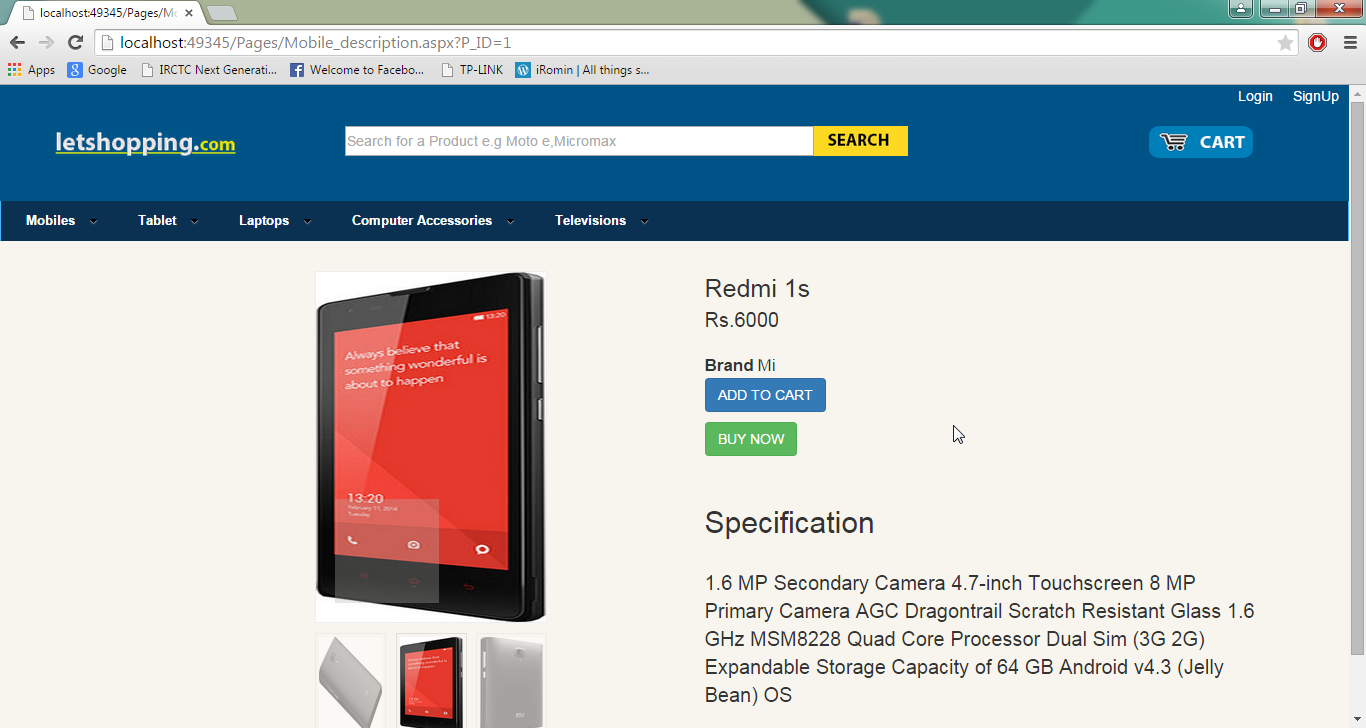
**Description :Use to recover the password**

****

**5. Buy now**

**Form name:** **Mobile\_description.aspx**

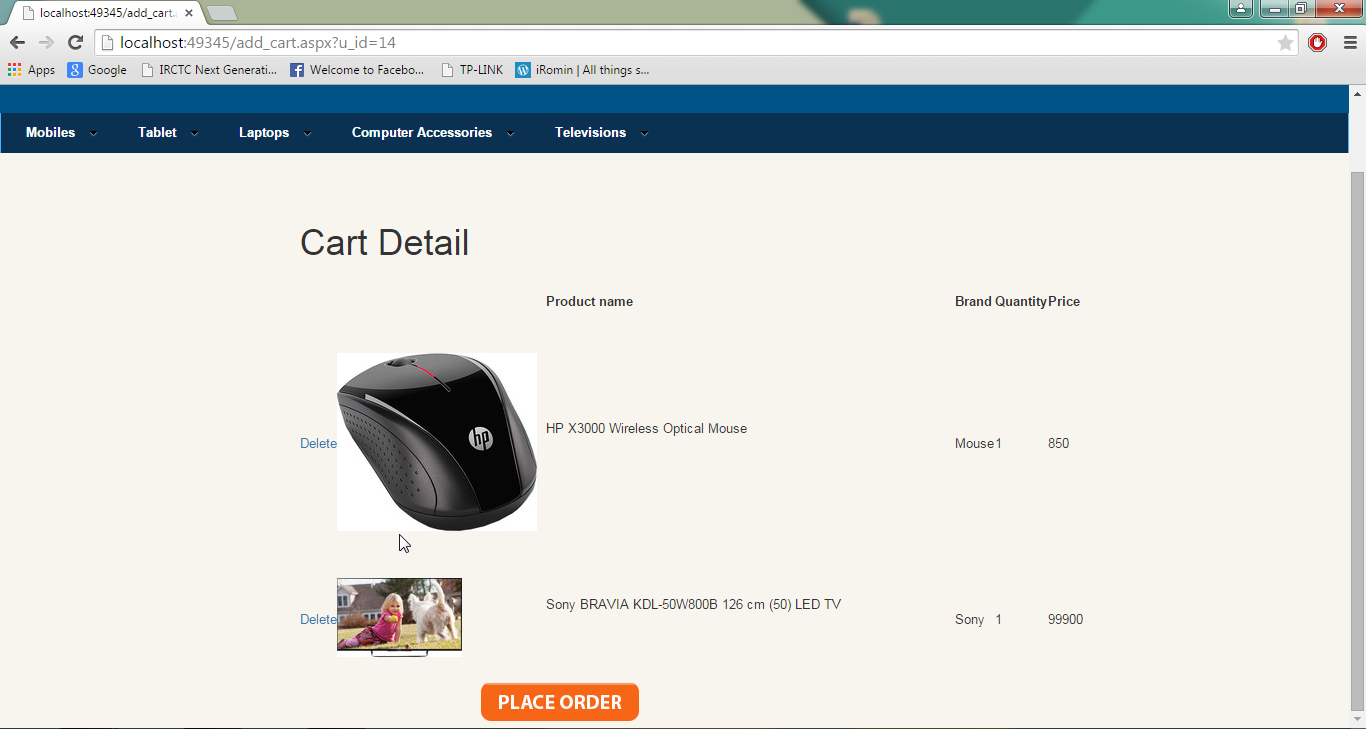
**Description :Use to buy product**

****

**6. Shopping Cart**

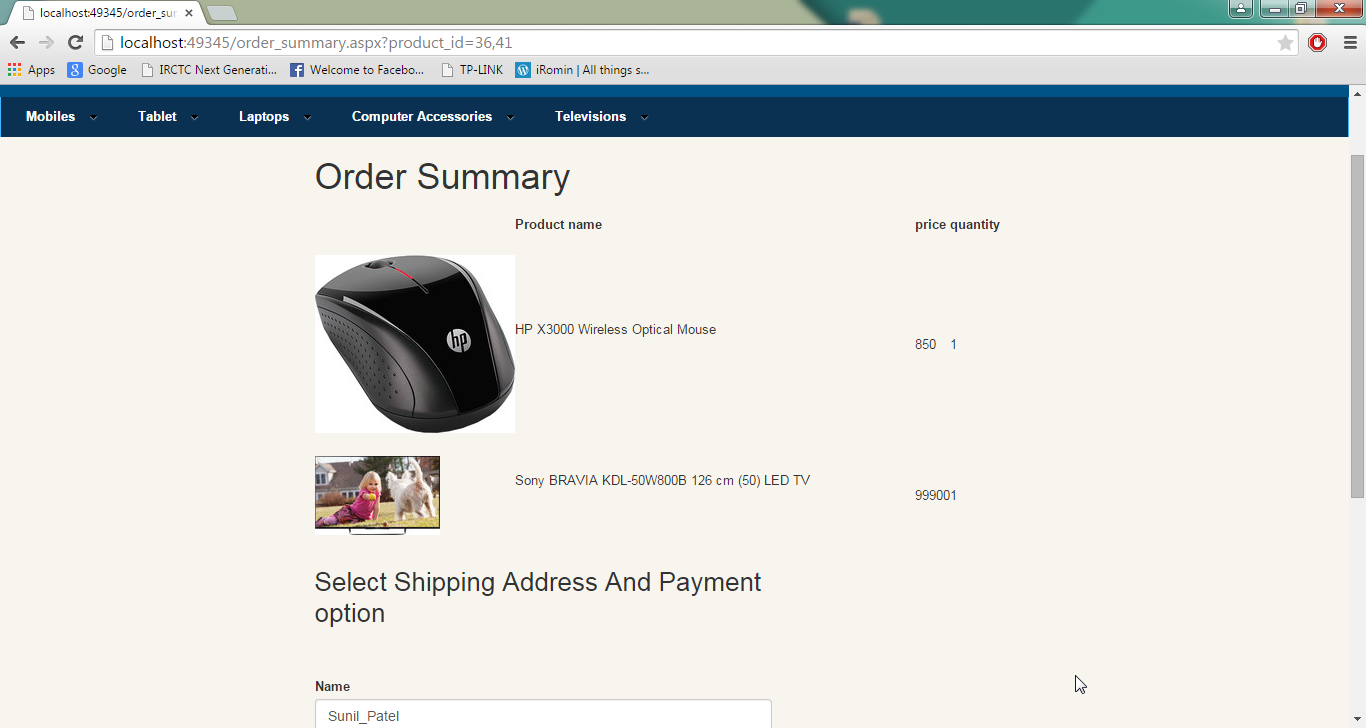
**Form name:** **add\_cart.aspx**

**Description :Use to add a product on Cart**

****

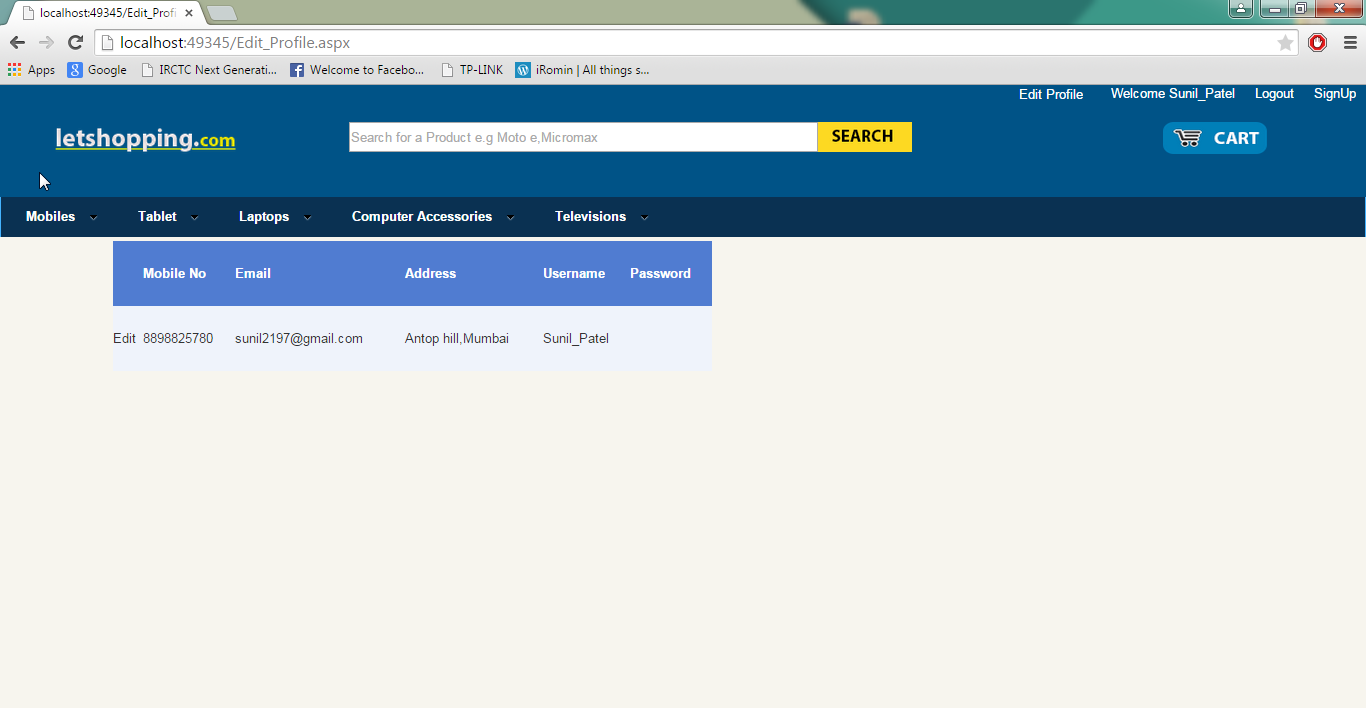
**7. Order Summary**

**Form name:** **order\_summary.aspx**

** Description : Use to for Product summary at the time of purchase**

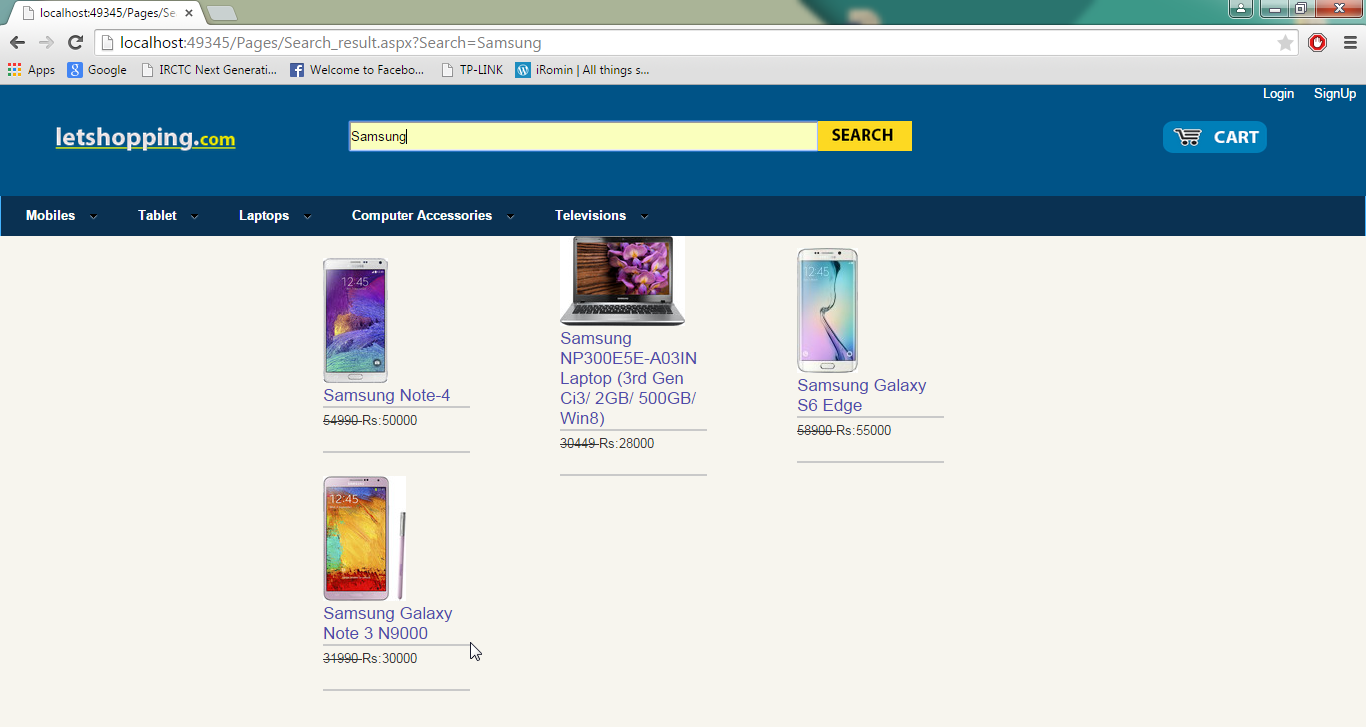
**8. Edit Profile**

**Form name:** **Edit\_Profile.aspx**

** Description :Use to Edit the user profile**

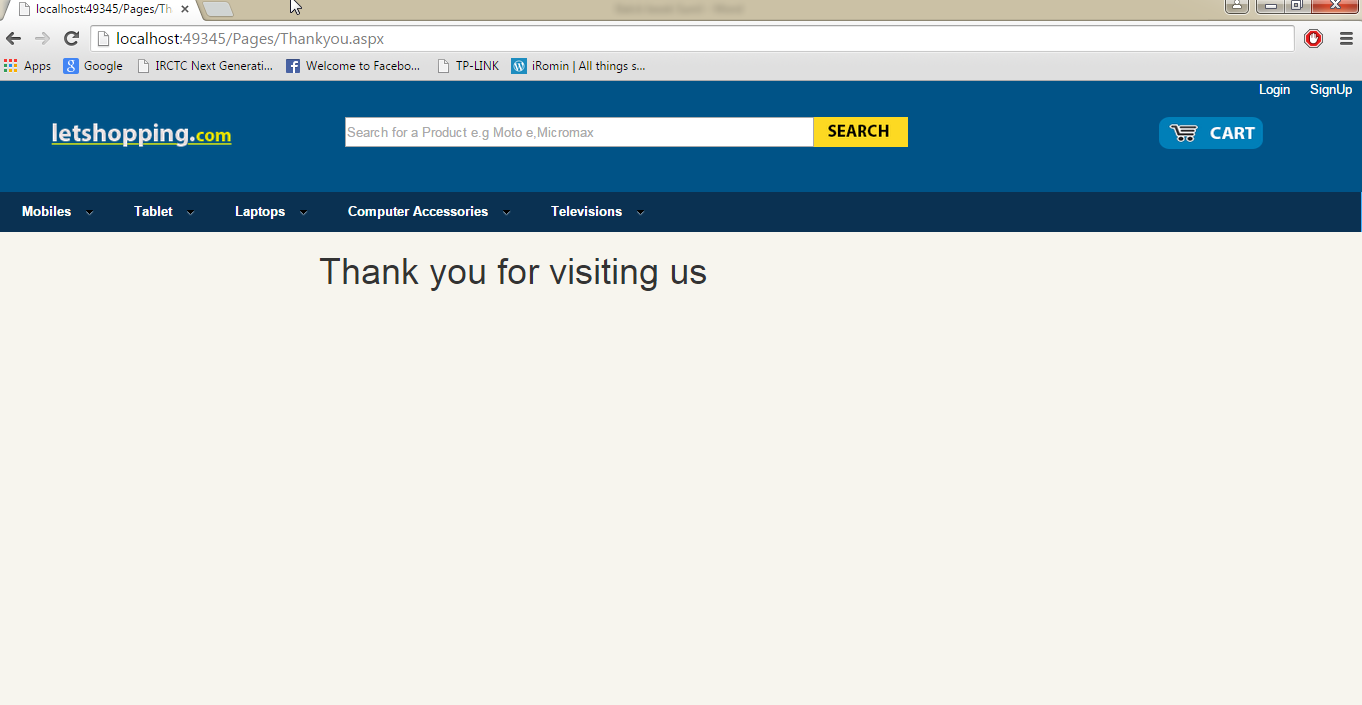
**9.Search Product**

**Form name:** **Search\_result.aspx**

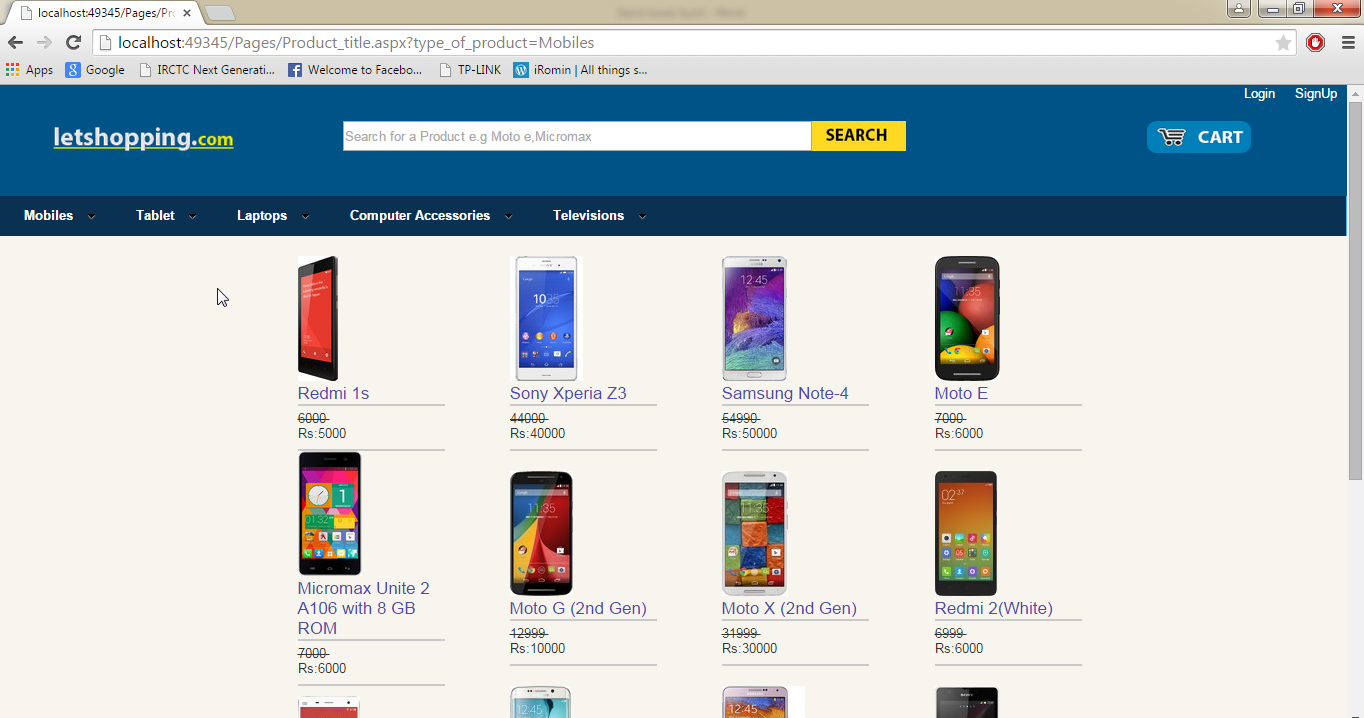
** Description: Use to search a product**

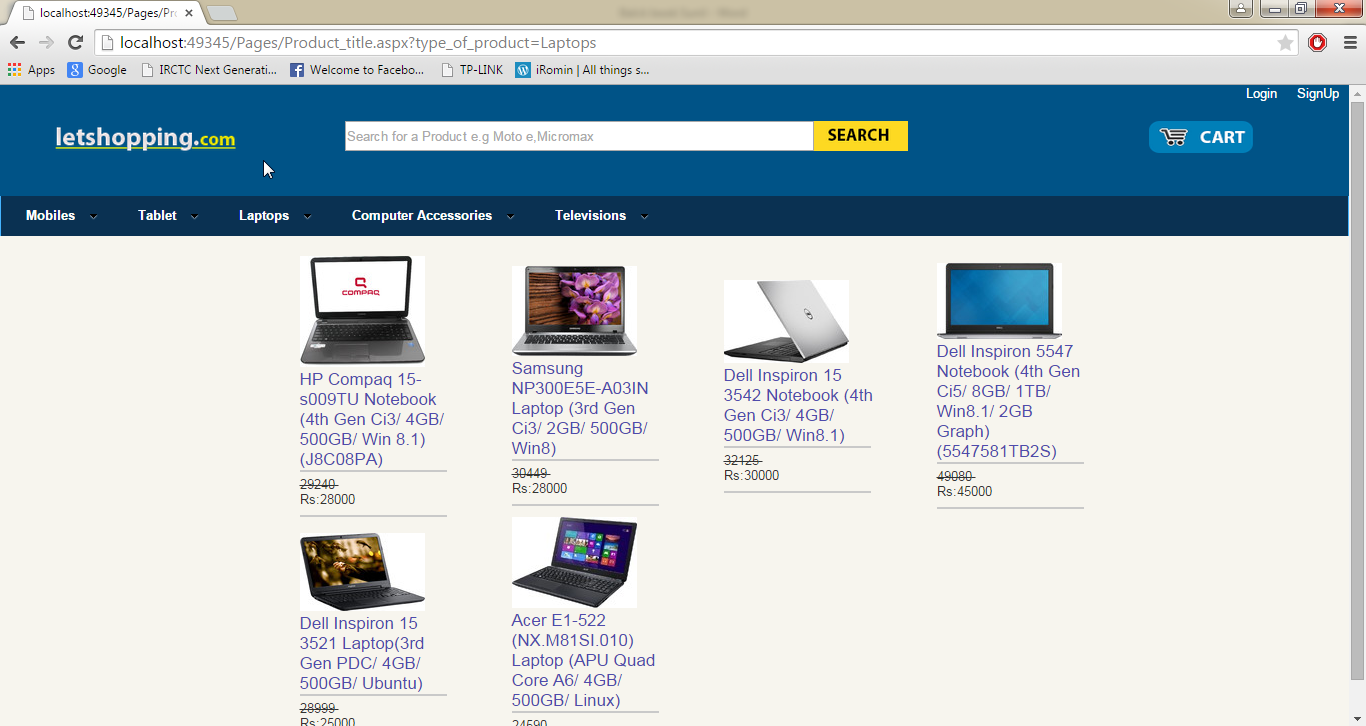
**9. Logout screen**

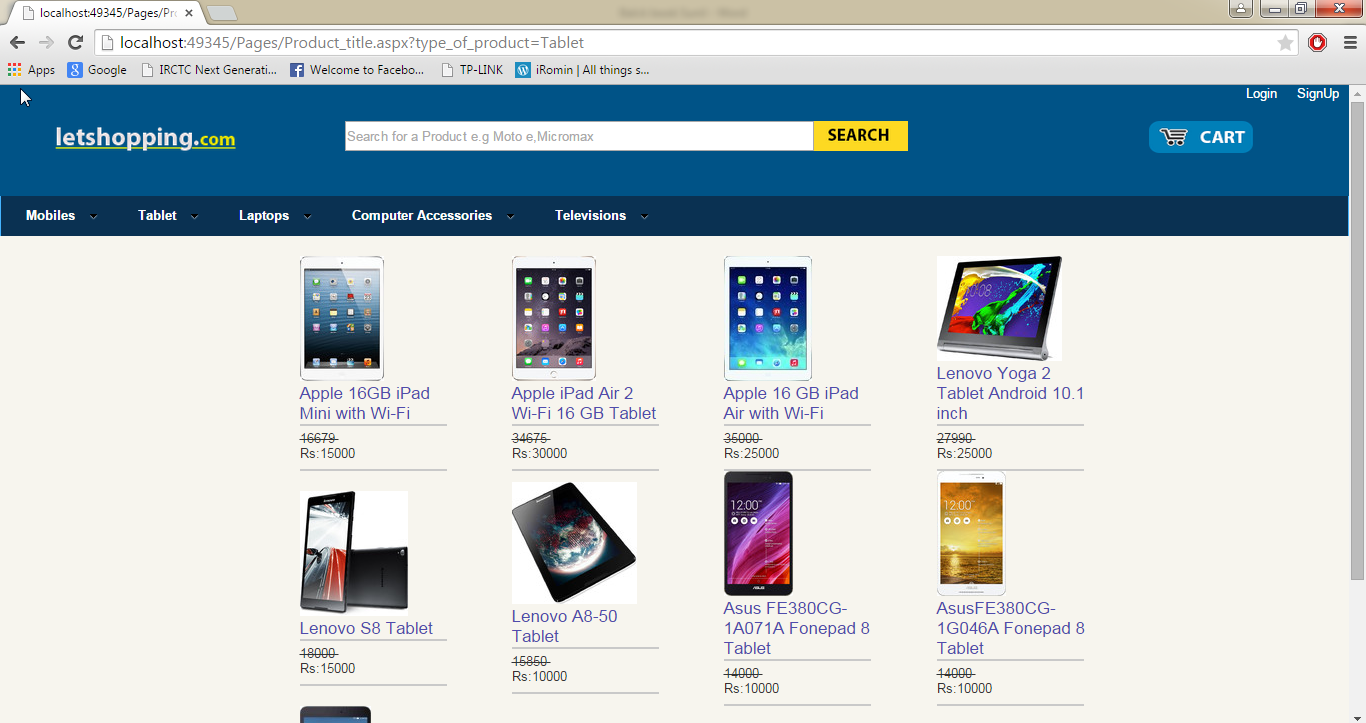
**Form name: Thankyou.aspx**

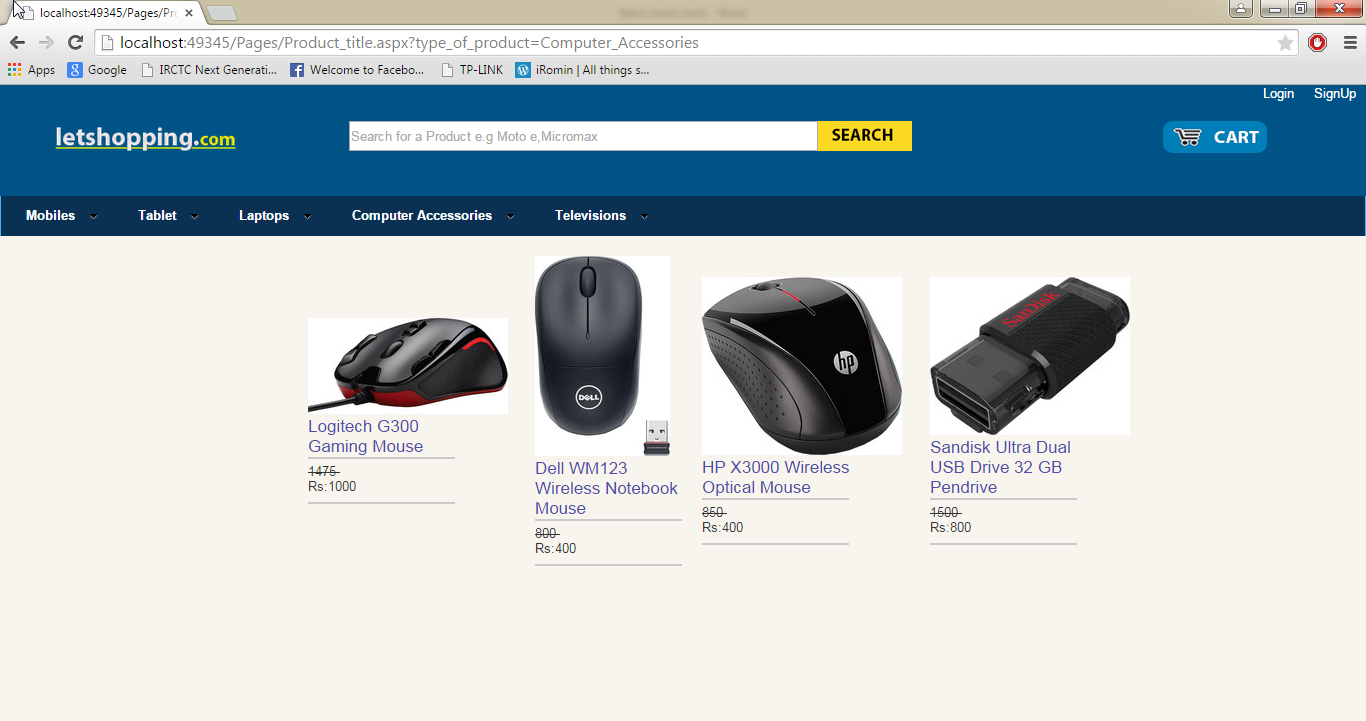
**Description: Thank you show at time of logout**

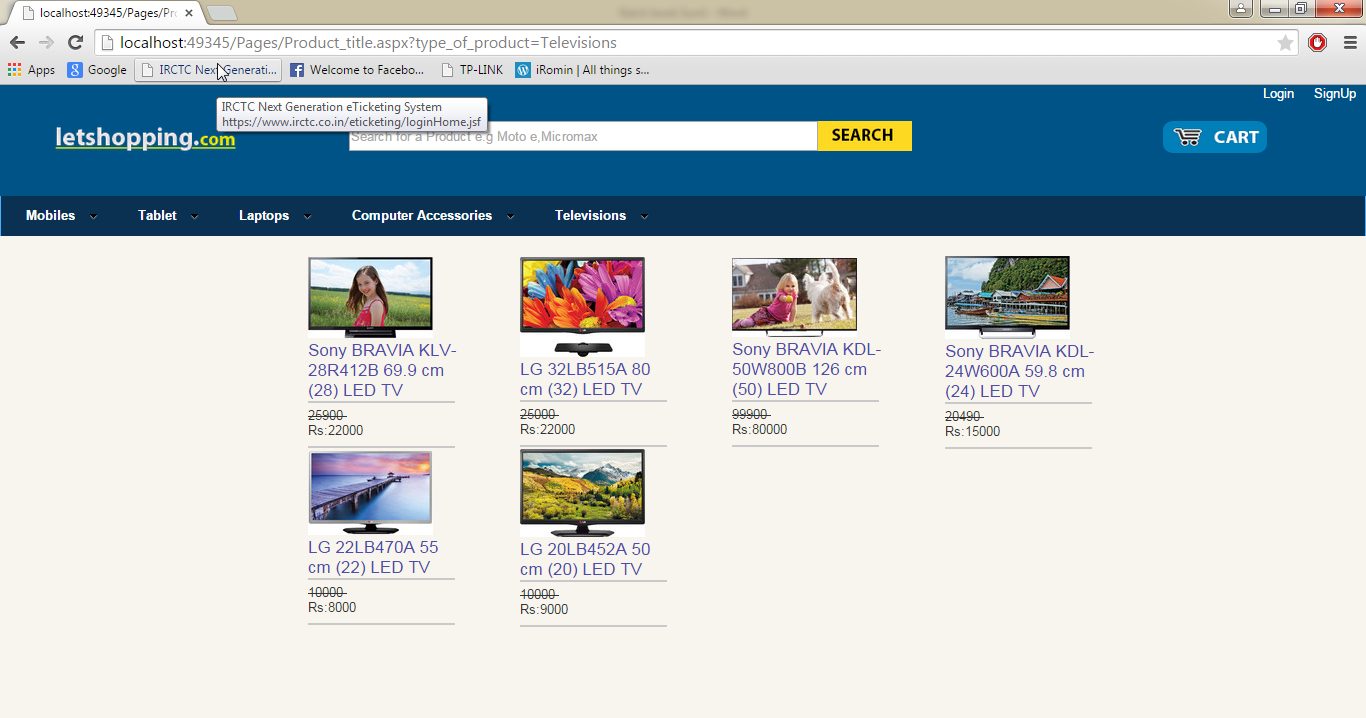
**10. Mobile**

****

**12.Laptop**

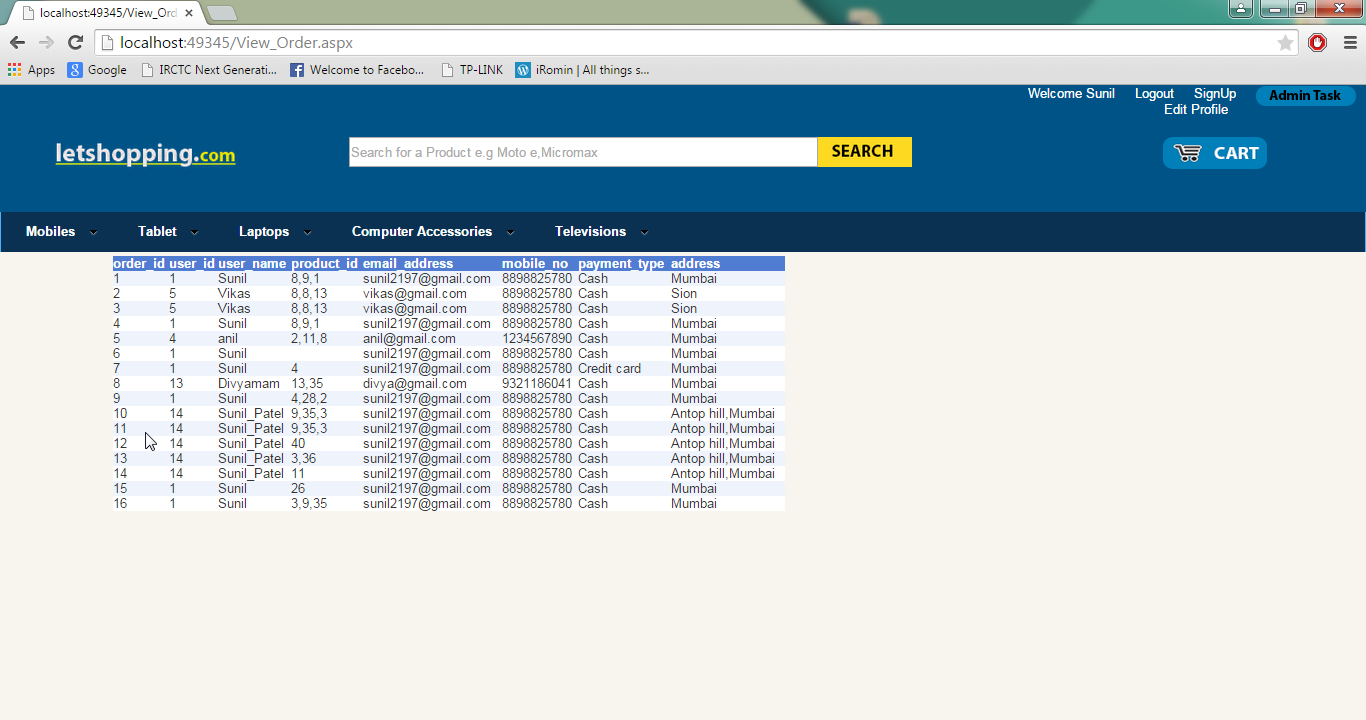
**13.Tablet**

**14.** **Computer Accessories**

**15.** **Televisions**

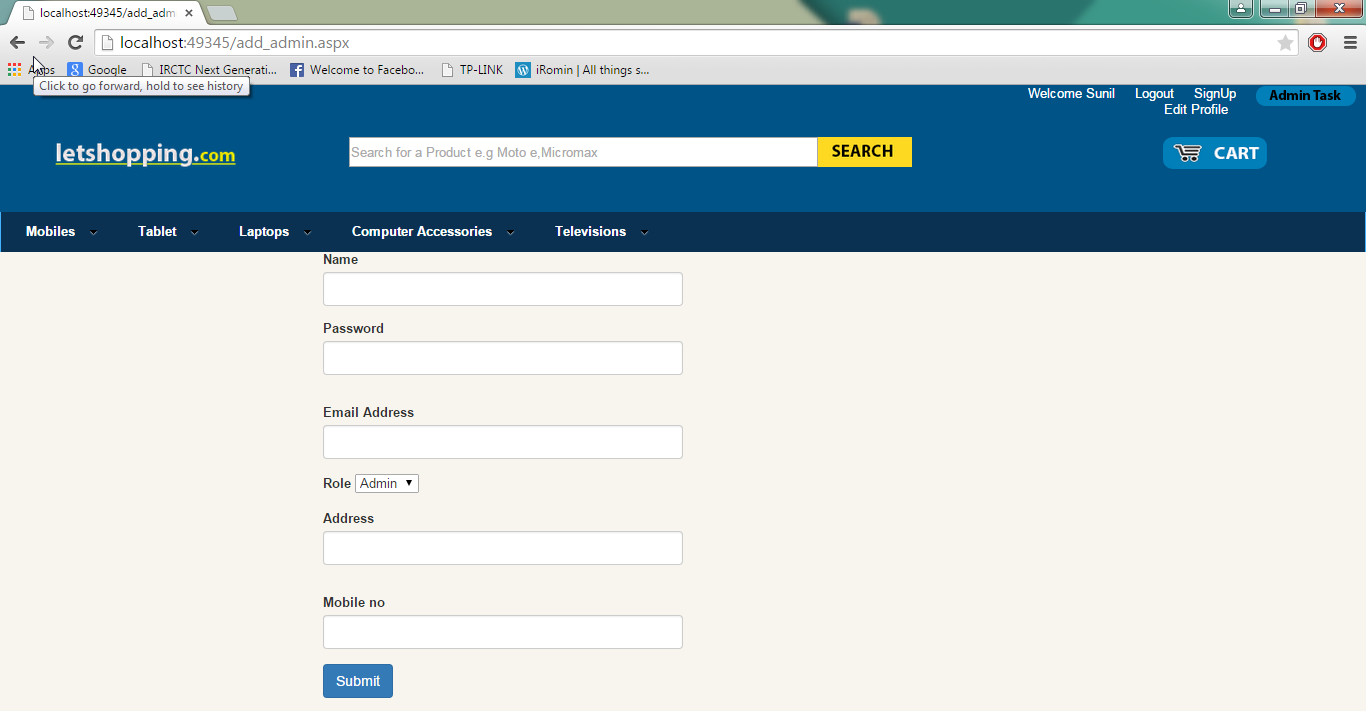
**16.View Order**

**Form name:** **View\_Order.aspx**

**Description: To view the customer Order**

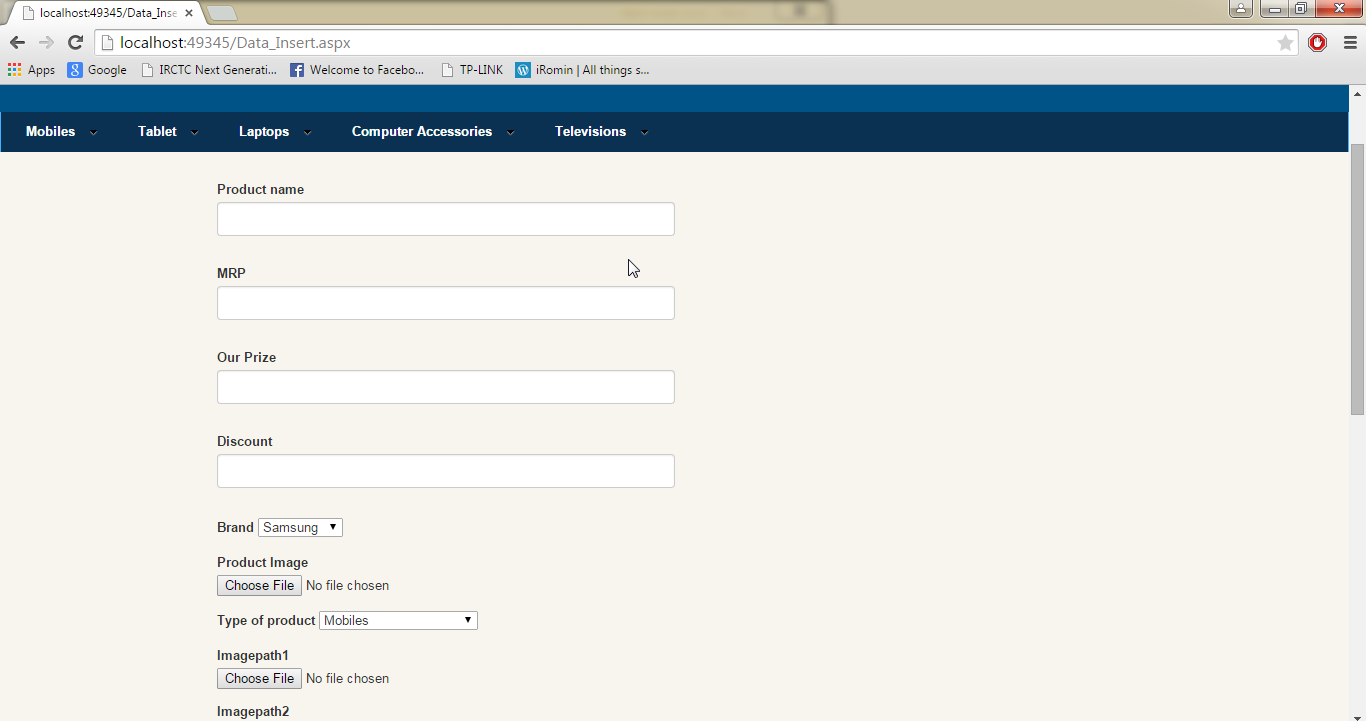
**17.Add Admin**

**Form name:** **add\_admin.aspx**

**Description:This page use to add admin**

**18.Add Product**

**From name:** **Data\_Insert.aspx**

**Description: Use to add a product**

**Limitations of project**

**10. LIMITATIONS OF PROJECT**

* **Enjoyment of retail shopping lost**

Many enjoy shopping with others and it is often a good way to make social connections. When shopping independently online, the enjoyment is lost.

* **Privacy and security issues**

Privacy is the number one reason that non-online shoppers do not shop online. Almost 95% of Web users have declined to provide personal information to Web sites at one time or another when asked (Hoffman, Novak, & Peralta, 1999). Another recent study has found that privacy was the top concern of customers while security ranked bottom (Schaupp & Belanger, 2005). This proves that many do not trust the privay of the Internet and are concerned with their credit card fauds, unwanted solicitation, and use of their information for other purposes. Security of Web sites is not the top concern because many shop on Web sites that they trust so that other factors appear to be more important than security.

* **Access to the Internet and computer necessary**

Because one needs money to buy a computer and to have internet connection, online shopping seems to be limited people of reasonable amount of income. Also, since it is harder to learn computer at an older age, the elderly people tend shop at traditional retail stores.

* **Product category risk**

Product category risk is related to functional products such as apparel, perfume, and electronics, that have functions that cannot fully be expeirenced online. Online shopper are worried that the products will not be what they have expected by viewing online. This is a clear disadvantage of onlins shopping because it shows that "[t]he likelihood of purchasing on the Internet decreases with increases in product risk" (Bhatnagar, Misra, & Rao, 20000, p. 100). Apparels in particular had negative rating in online shopping because of it is difficult to feel and see the texture of color online that is incomparable to going to a retail store, even with magnifying tools online. Also, one cannot try on a clothing before buying it online, so it would be very inconvenient if the size did not fit the person and he/she had to return it.

Too many choices Although having access to a very large number of products is highly desirable, consumers have limited cognitive resources and may simply be unable to process the potentially vast amounts of information about these alternatives (Haubl & Trifts, 2000). Online stores need to provide the variety in an organized way that will facilitate shopping online

**Future expansions**

**11. FUTURE EXPANSION**

1. Secure registration and profile management facilities for Customers
2. Browsing through the e-Mall to see the items that are there in each category of products like Apparel, Kitchen accessories, Bath accessories, Food items etc.
3. Adequate searching mechanisms for easy and quick access to particular products and services.
4. Creating a Shopping cart so that customers can shop ‘n’ no. of items and checkout finally with the entire shopping carts
5. Customers should be able to mail the Shop about the items they would like to see in the Shop
6. Regular updates to registered users of the website about new arrivals.
7. Secured mechanism for checking out from the Shop (Credit card verification mechanism)
8. Updates to customers about the recently added items in the shop through various mechanisms.
9. Uploading ‘Most Purchased’ Items in each category of products in the Shop like Apparel, Kitchen accessories, Bath accessories, Food items etc.
10. Strategic data and graphs for Administrators and Shop owners about the items that are popular in each category and age group
11. Give special discounts to Premier customers
12. Shop employees are responsible for internal affairs like processing orders, assure home delivery, getting customer's delivery-time feedback, updating order's status and answering client's queries online.
13. Feedback mechanism, so that customers can give feedback for the product or service which they have purchased. Also facility rating of individual products by relevant customers. Also feedback can be given on the performance of particular vendors and the entire mall as well.
14. Adequate payment mechanism and gateway for all popular credit cards, cheques and other relevant payment options, as available from time to time.

**Initial non functional requirements will be: -**

1. Secure access of confidential data (user’s details). SSL can be used.
2. 24 X 7 availability

3. Better component design to get better performance at peak time

4. Flexible service based architecture will be highly desirable for future extension

5. Advertisement space where it will effectively catch the customer’s attention and as a source of revenue.

**12**. **BIBLIOGRAPHY**

**Books:**

1. The main books , which were consulted for the project development, are :
2. Guide to Microsoft Visual Studio– Peter Norton
3. Introduction of Microsoft SQL Server : Aptech
4. MSDN Library
5. Software Engineering – Roger S. Pressman, K. K. Aggarwal

**Websites:**

1. <http://getbootstrap.com/css/>
2. <http://stackoverflow.com/questions/tagged/asp.net>
3. <http://asp.net-tutorials.com/>
4. http://www.w3schools.com/css/